Theoretical Computer Science 1028 (2025) 115029

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Parameterizing path partitions ☆

Henning Fernau [a], Florent Foucaud [b,*], Kevin Mann [a], Utkarsh Padariya [c], Rajath Rao K.N. [c]

[a] Universität Trier, Fachbereich IV, Informatikwissenschaften, Germany
[b] Université Clermont Auvergne, CNRS, Clermont Auvergne INP, Mines Saint-Étienne, LIMOS, 63000 Clermont-Ferrand, France
[c] International Institute of Information Technology Bangalore, India

ARTICLE INFO

Keywords:
Path partitions
NP-hardness
Parameterized complexity
Neighborhood diversity
Directed neighborhood diversity
Vertex cover parameterization

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 SPP and IPP 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, as well as for other special subclasses of directed graphs (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. We also give an explicit algorithm for the vertex cover parameterization of PP. 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. We also lift the mentioned neighborhood diversity and dual parameterization results to directed graphs; here, we need to define a proper novel notion of directed neighborhood diversity. As we also show, most of our results also transfer to the case of covering by edge-disjoint paths, and purely covering.

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), partitioning the vertex set by independent sets (COLORING), and covering the vertex set by paths or cycles [59]. 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 [36], Berge's path partition conjecture [9,46] and a conjecture by Magnant and Martin [58]. These studies are motivated by applications in

☆ This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.

* Corresponding author.
E-mail addresses: fernau@uni-trier.de (H. Fernau), florent.foucaud@uca.fr (F. Foucaud), mann@uni-trier.de (K. Mann), utkarsh.prafulchandra@iiitb.ac.in (U. Padariya), rajath.rao@iiitb.ac.in (R. Rao K.N.).

https://doi.org/10.1016/j.tcs.2024.115029
Received 31 August 2023; Received in revised form 1 November 2024; Accepted 6 December 2024

Available online 14 December 2024
0304-3975/© 2024 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

diverse areas such as code optimization [11], machine learning / AI [78], transportation networks [79], bioinformatics [53], parallel computing [70], and program testing [66]. 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*, focusing 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 (directed) raph G is a partitioning of the vertex set into (directed) 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 (often under the names Path Cover or Hamiltonian Completion) on many graph classes, see for example [10,15,19,22,32,41,52,68] and references therein.

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 for a summary of our results. 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 the neighborhood diversity and vertex cover number.

The three problems considered are all NP-hard. They are mostly studied on undirected graphs, and unless stated otherwise, the following references are for undirected graphs. PP can be seen as an extension of Hamiltonian Path and is thus NP-hard, even for one path (while for one path, SPP and IPP are very easy, as it suffices to check whether the whole graph is a path). Similarly, IPP is NP-hard, even for two paths [55]. Recently, SPP was proved to be NP-hard [60], but note that, as opposed to the two other problems, it is polynomial-time solvable for any constant number of paths by an XP algorithm [26]. On trees, the three problems are equivalent, and are solvable in polynomial time (as shown in the 1970s in [10,41,52,74], see also [32] for a more recent improved algorithm). As a special case of Hamiltonian Path, it is known that PP is NP-hard for many graph classes such as planar graphs [38] or chordal bipartite graphs [65]. On the other hand, it can be solved in polynomial time for some graph classes, such as cographs [15], cocomparability graphs [19] or graphs whose blocks are cycles, complete graphs or complete bipartite graphs [68] (this includes trees, cactii and block graphs). We refer to [68] for further references. SPP remains NP-hard on split graphs but becomes polynomial-time solvable on cographs and chain graphs [14]. IPP can also be solved in linear time for graphs whose blocks are cycles, complete graphs or complete bipartite graphs [69] and on cographs and chain graphs [14], but no further positive results for special graph classes seem to be known. For directed graphs, it is known that PP can be solved in polynomial time for DAGs [18, Probl. 26-2] but IPP is NP-hard on DAGs, even for three paths [53].

For a detailed survey on these types of problems (both partitioning and covering versions), see [59], and also [2].

Although every shortest path is an induced path and every induced path is a path, the reverse are not true. In this sense, induced paths can be seen as an intermediate notion between shortest paths and unrestricted paths. However, this intuition may fail in certain contexts. Indeed, some seemingly simple problems related to paths are easy for unrestricted or shortest paths, but become hard for induced paths. For example, given a graph G and three vertices x, y, z, determining whether there is an induced path from vertex x to vertex y going through vertex z is NP-complete [63], while this task is polynomial-time both for unrestricted paths and for shortest paths.

Further related work. As sketched in [32], the PP problem is equivalent to the Hamiltonian Completion problem, asking to add at most k edges/arcs to a (directed) graph to guarantee the existence of a Hamiltonian path. Besides Hamiltonian Path, PP is also directly connected to other graph problems such as $L(2,1)$-labellings [39] or the metric dimension [3]. It is the object of the Gallai-Milgram theorem, stating that for any minimal path partition of a directed graph, there is an independent set intersecting each path [36], see also [23, Section 2.5].

Variations of PP and IPP where the paths must have a prescribed length are also studied, see for example [56,64,76] for undirected graphs, and [17,28] for directed graphs.

The covering versions of PP and IPP (where the paths need not necessarily be disjoint) seem to be very little studied [59], however we can mention some works for PP on DAGs, which is polynomial-time solvable and has numerous practical applications (see [12] and references therein). On the other hand, the covering version of SPP was recently studied (often under the name Isometric Path Cover), see [67] for a linear-time algorithm on block graphs, [14] for algorithms on cographs and chain graphs, [26] for an XP algorithm, [13] for NP-hardness on chordal graphs and approximation algorithms for chordal graphs and other classes, and [78] for a $\log n$-factor approximation algorithm. This problem is connected to the *Cops and Robber game*, where a robber and a set of cops alternatively move in a graph (a move is to reach a neighboring vertex); the robber tries to evade the cops indefinitely, while they try to reach (catch) the robber. As shown in [1,30], if we are given a set of shortest paths covering all the vertices, we can assign a cop to each path and let him patrol along it (while staying as close as possible to the robber). Eventually, they will catch the robber, and thus the smallest size of such a shortest path cover is an upper bound to the smallest number of cops required to catch the robber. This problem is also connected to decomposition results in structural graph theory [25] and to machine learning applications [78,77].

The versions of PP and SPP where, on the other hand, covering is not required (and the endpoints of the solution paths, called *terminals*, are prescribed in the input) are widely studied as Disjoint Paths (DP) [71] (sometimes also called Linkage) and Disjoint Shortest Paths (DSP) [7,57]. Both these problems are NP-complete and have applications in network routing problems and VLSI design [72]. DP in particular has been extensively studied, due to its deep connections with structural graph theory: as part of the celebrated Graph Minor project, Robertson and Seymour showed that for undirected graphs, DP is in FPT, parameterized by

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, DG and SG referring to undirected graphs, directed graphs and special graph classes, respectively. Our results are highlighted in bold face.

parameter	PP	SPP	IPP
none (UG/DG)	NP-c. [37]	NP-c. [60]	NP-c. [55]
none (bipartite UG)	NP-c. [51,55]	**NP-c.**	open
solution size k (UG)	paraNP-h. [37]	**in XP**	paraNP-h. [55]
solution size k (DG)	paraNP-h. [37]	open **in XP (SG)**	paraNP-h. [55]
solution size k (DAG)	polynomial, see [18], Problem 26-2	**NP-c.** **W[1]-h.** **in XP**	**NP-c.** **W[1]-h.** paraNP-h. [53]
neighborhood diversity (UG)	FPT [34]	**FPT**	**FPT**
neighborhood diversity (DG)	open	**FPT**	**FPT**
vertex cover number (UG/DG)	**FPT**	**FPT**	**FPT**
dual $n-k$ (UG/DG)	**FPT**	**FPT**	**FPT**

the number of paths [71], contrasting PP. An improved FPT algorithm was given in [49]. Algorithms were also designed for planar directed graphs [31] (FPT), graphs of bounded directed treewidth [47] (XP) and other classes [5]. Recently, DSP on undirected graphs, which has applications in artificial intelligence [43], was shown to have an XP algorithm and to be W[1]-hard when parameterized by the number of paths [57]; an improved algorithm was given in [7], and the directed graph case is studied in [8].

A strengthening of the DP problem (motivated by the task of detecting induced subgraphs or induced subdivisions of graphs) is studied under the name Induced Disjoint Paths [40,48,61]: here, given the terminals, one is looking for disjoint unrestricted paths connecting the terminals, but moreover, no edge should connect two vertices on different paths. This problem is NP-hard even for two paths [48].

Some further restricted variations around disjoint paths are studied for example in [6] (for unrestricted paths) and in [4] (for induced paths).

Our contribution. Table 1 summarizes known results about the three problems, with our results are highlighted in bold. We fill in most of the hitherto 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 (on undirected graphs, DAGs, and special classes of directed graphs). This is surprising as both PP and IPP are NP-hard on undirected graphs for $k = 1$ and for $k = 2$, respectively, see [37] and [55]. To prove this, we use existing XP (or faster) algorithms for the related Disjoint (Shortest) Paths problem. 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 [18, Probl. 26-2]), 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 and have maximum degree 4.[1] We strengthen this result using a similar construction as in the classic proof of DP being W[1]-hard on DAGs by Slivkins [75], to show that IPP and SPP are W[1]-hard on DAGs. As mentioned above, Manuel recently showed that SPP is NP-hard on undirected graphs [60]. We extend this result to show that SPP is NP-hard even when restricted to bipartite graphs that are sparse (they have degeneracy at most 5). The complexity of these problems has not yet 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 the technique of Integer Linear Programming. Also, we lift these considerations to directed graphs. To this end, we introduce the notion of *directed neighborhood diversity* in this paper, which may be useful for other problems on directed graphs. We also obtain easy FPT algorithms for PP, IPP and SPP on both undirected and directed graphs when parameterized by the vertex cover number of the (underlying) graph. We further obtain a non-trivial FPT algorithm for PP on undirected graphs G with vertex cover number $\mathsf{vc}(G)$, with the improved running time $\mathcal{O}^*\left(2^{\mathcal{O}(\mathsf{vc}(G)\log(\mathsf{vc}(G)))}\right)$. Moreover, when considering the dual parameterization (graph order minus number of paths), we show that all three variants, PP, IPP, and SPP, are in FPT for undirected graphs. This is also the case for directed graphs (except for PP, which is left open). Finally, we also remark that many of our results apply to the purely "vertex covering" versions of PP, IPP, and SPP, and to their intermediate "vertex covering using edge-disjoint paths" versions.

It is interesting to note the differences in the complexities of the three problems on different input classes. For example, we will see that SPP can be solved in XP-time on undirected graphs when parameterized by solution size, 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, as any shortest path is induced, IPP can be seen as an intermediate problem between PP and SPP. Thus, it is not too surprising that, when the complexity

[1] We remark that in the conference version of this paper [29], this result was stated by mistake for graphs of maximum degree 3, but the proof is the same here and the statement is corrected.

of the three problems differs, IPP sometimes behaves like PP, and sometimes, like SPP. It would be interesting to find a situation in which all three problems behave differently.

Organization of the paper. We start with formally defining the studied problems in section 2. Then we focus our attention on NP-hardness results in section 3, showing that IPP and SPP are NP-hard when the input graph is a DAG, and that SPP is NP-hard when the input graph is a bipartite graph. Our main result showing that IPP and SPP are W[1]-hard for the standard parameter k (number of paths) is found in section 4. We give an XP algorithm for SPP in section 5. We then give an FPT algorithm for IPP and SPP when parameterized by neighborhood diversity in section 6, as well as a non-trivial direct FPT algorithm for PP, parameterized by vertex cover. These results are then applied in section 7 to prove FPT results for the dual parameterization. In section 8, we discuss the (edge-disjoint) covering versions of the problems, and argue that most of our results translate to this setting as well. We conclude in section 9.

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.

We say that a sub-graph G' of $G = (V, E)$ *spans* G if its vertex set is V. In other words, G' is a partial graph of G.

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?

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.

INDUCED PATH PARTITION (IPP for short)
Input: A graph G, a non-negative integer k
Problem: Are there pairwise vertex-disjoint paths $P_1, \ldots, P_{k'}$, with $k' \leq k$, that are induced paths, and such that, together, these paths span G?

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 $\mathrm{diam}(G)$.

SHORTEST PATH PARTITION (SPP for short)
Input: A graph G, a non-negative integer k
Problem: Are there pairwise vertex-disjoint paths $P_1, \ldots, P_{k'}$, with $k' \leq k$, that are shortest paths, and such that, together, these paths span G?

Remark 2.1. The following known combinatorial properties are helpful to connect the different yet related 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.

Omitting the vertex-disjointness condition, we arrive at three *covering versions* 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?

INDUCED PATH COVER (IPC for short)
Input: A graph G, a non-negative integer k
Problem: Are there paths $P_1, \ldots, P_{k'}$, with $k' \leq k$, that are induced paths and such that, together, these paths span G?

SHORTEST PATH COVER (SPC for short)
Input: A graph G, a non-negative integer k
Problem: Are there paths $P_1, \ldots, P_{k'}$, with $k' \leq k$, that are shortest paths and such that, together, these paths span G?

Let us also define the non-covering versions of PP, IPP and SPP, that will be used in the paper.

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?

Analogously to DP, we can define the problems DISJOINT INDUCED PATHS[2] and DISJOINT SHORTEST PATHS as follows.

DISJOINT INDUCED PATHS (DIP 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 induced 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?

DISJOINT SHORTEST PATHS (DSP 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 shortest 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?

Remark 2.2. The problems DIP and DP are *equivalent* when the inputs are undirected graphs or DAGs (in the sense that the Yes-instances are the same for both problems), but not for general directed graphs.

Proof. Namely, if an input is a Yes-instance of DIP, then trivially it is also a Yes-instance of DP. But also if an input is a Yes-instance of UDP or DAGDP, then it is also a Yes-instance of UDIP or DAGDIP, as we could take each path P in the solution $\mathbb{P}$ of UDP or DAGDP and find a path P' induced by some subset of vertices of P; this can be done by starting a breadth-first search at one endpoint of the path P to reach the other endpoint and choosing a shortest path. This way, we get a set $\mathbb{P}'$ of induced paths as a solution to UDIP or DAGDIP. This does not hold true for general directed graphs, as a solution to DDP does not imply a solution to DDIP. For example, consider a directed graph with three vertices a_1, a_2, a_3 and arcs $(a_1,a_2),(a_2,a_3),(a_3,a_1)$ with $(s_1,t_1) = (a_1,a_3)$ as the prescribed path endpoints. □

Interestingly, and contrasting Remark 2.2, we will see that the complexities of DAGPP and DAGIPP differ drastically, see Table 1.

2.1. Graph notions

A graph is *d-degenerate* if every induced subgraph has a vertex of degree at most d. The *vertex cover number* of a graph G is the smallest size of a vertex cover of G, that is, a set S of vertices of G such that each edge of G intersects S. The *neighborhood diversity* [54] of a graph G (or just $\mathsf{nd}(G)$) is the number of equivalence classes of the following equivalence relation: 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, which we call *neighborhood diversity classes*, form either cliques or independent sets and all vertices in one class are pairwise *twins* (vertices with either the same closed neighborhood or the same open neighborhood) [50]. More information on this parameter can be found in [50]. Notice that V can be partitioned into vertices of the same type in linear time using partition refinement [44, Algorithm 2].

3. NP-hardness results

This section contains several NP-hardness reductions for the studied problems.

3.1. SPP and IPP on DAGs

A well-known result related to PC in DAGs is Dilworth's theorem: the minimal size of a directed path cover equals the maximal cardinality of an independent set (or, in the original language of posets, the minimum size of a chain cover is equal to the maximum

[2] This problem should not be confused with INDUCED DISJOINT PATH, see [40,48,61], where it is required that any two (unrestricted) solution paths have no adjacent vertices.

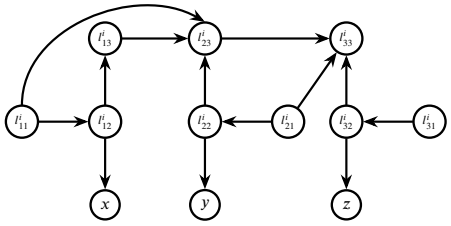


Fig. 1. The vertex gadget, replacing v_i in G with nine vertices in G', as defined in the proof of Theorem 3.1.

size of an antichain) [24]. Fulkerson [33] gave a constructive proof of this theorem, from which it follows that the DAGPC problem can be reduced to a maximum matching problem in a bipartite graph. See also [12] for further information. Even DAGPP can be solved in polynomial time by a similar method [18, Problem 26-2]. We show that, in contrast, DAGSPP and DAGIPP are NP-hard even when restricted to planar bipartite DAGs.

Theorem 3.1. *DAGSPP and DAGIPP are NP-hard even when the inputs are restricted to planar bipartite DAGs of maximum degree 4.*

Proof. Our reduction is adapted from [64,76]. We reduce from the Planar 3-Dimensional Matching problem, or Planar 3-DM, which is NP-complete (see [27]), 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,[3] 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 $1 \le j, k \le 3$ and with edges as shown in Fig. 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 4 in which every shortest/induced path is of length at most 3, and the underlying undirected graph of G' is bipartite.*

Proof. The first four conditions are easy to see in the given construction. We will prove the bipartiteness claim by giving a proper 2-coloring. We will color all vertices which represent the elements of sets X and Z with color 1, and all vertices which represent elements of set Y with color 2. Then we observe that within the gadget $H(v_i)$, the vertices $l_{12}, l_{23}, l_{21}, l_{32}$ have color 2, and the remaining vertices have color 1. This coloring is a proper 2-coloring of G'. ◇

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

Proof. Since $|V(G')| = 3p + 9q$, if G' can be partitioned into $p + 3q$ shortest/induced paths, we observe that the average path length has to be 2. Observe also that none of the paths can contain more than one of the original elements of $X \cup Y \cup Z$.

The only possible shortest/induced paths of length 3 are of the form $P^i = l^i_{12} l^i_{13} l^i_{23} l^i_{33}$, but if one takes this path into the solution, to cover vertex l^i_{11}, one has to include this vertex as a single vertex path, hence the average path length of such a solution would be strictly less than 2. We can also eliminate paths $P^i_1 = l^i_{11} l^i_{23} l^i_{33}$ and $P^i_2 = l^i_{22} l^i_{23} l^i_{33}$ from the solution of G' by similar arguments. More precisely, taking P^i_1 would force us to also take the path $l^i_{12} l^i_{13}$, and taking P^i_2 even makes us select the singe vertex path l^i_{21}. Again, the average path length of such a solution would be strictly less than 2.

In summary, each shortest or induced path in the solution contains exactly three vertices, and each gadget $H(v_i)$ is partitioned into P_3-paths in one of the two ways shown in Fig. 2.

We now describe how to read off a solution of the original Planar 3-DM instance as described by G from a shortest/induced path partition of G' into $p + 3q$ shortest/induced paths. If a gadget $H(v_i)$ is partitioned as in (2) in Fig. 2, we select that triple $v_i = (x, y, z) \in T$ into the solution S of G, else we do not select the triple into the solution. Since the solution covers all vertices of G', it will cover in particular all vertices from $X \cup Y \cup Z$; thus, the selected set S of triples will form a Planar 3-DM solution.

[3] This planar embedding can be computed in linear time if necessary [73].

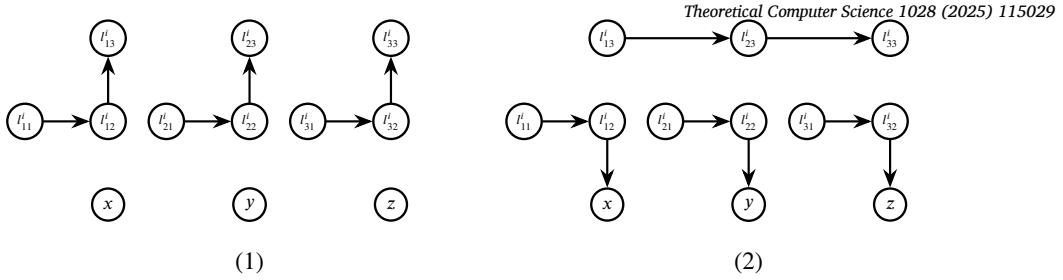


Fig. 2. Two different vertex partitions of a $H(v_i)$ gadget into 3-vertex paths, corresponding to different triple selections in the construction of Theorem 3.1.

Conversely, if S is a solution of the PLANAR 3-DM instance specified by G, then for $v_i = (x, y, z) \in S$, partition the corresponding $H(v_i)$ as described in (2) of Fig. 2, and partition every other $H(v_j)$ as shown in (1) of Fig. 2. This provides a shortest/induced path partition of G' into $p + 3q$ shortest paths. ◇

This completes our proof of the theorem. □

3.2. SPP for bipartite undirected graphs

Next, we prove that USPP 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-USPP on bipartite graphs to USPP on bipartite graphs. 4-USPP 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-USPP is NP-hard on bipartite graphs (Lemma 3.5) by a reduction from 4-UIPP (also known as INDUCED P_4-PARTITION) on bipartite graphs. 4-UIPP 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 3.4 *([64]). 4-UIPP is NP-hard for bipartite graphs of maximum degree* 3.

Lemma 3.5. *4-USPP is NP-hard for bipartite graphs of maximum degree* 3.

Proof. This follows from Lemma 3.4 and the fact that, in a bipartite graph, a path of length three is an induced path if and only if it is a shortest path, see Remark 2.1. □

Theorem 3.6. *USPP is NP-hard, even for bipartite 5-degenerate graphs with diameter 4.*

Proof. To prove this claim, we use Lemma 3.5. Given an instance of 4-USPP, 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 USPP.

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. See Fig. 3 for an illustration. This describes E' of G'.

Claim 3.7. *$G' = (V', E')$ is bipartite and 5-degenerate.*

Proof. 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 $B' = B \cup \{x_1, x_3, x_5, y_2, y_4\}$ (see Fig. 3).

Hence, G' is bipartite. Now we prove that G' is 5-degenerate. As the maximum degree of G is 3 and as the construction of G' will add two more neighbors to each vertex of $A \cup B$, vertices from $A \cup B$ have degree at most 5 in G'. Moreover, vertices x_1, x_3, x_5, y_1, y_3 and y_5 have degree 1 or 2 in G'. Thus, any induced subgraph containing a vertex other than x_2, x_4, y_2, y_4 has some vertex of degree at most 5. Moreover, the subgraph induced by these four vertices is a 4-cycle. Thus, G' is 5-degenerate. ◇

Claim 3.8. *Any shortest path of G', not starting and ending with one of the vertices from $\{y_1, y_5, x_1, x_5\}$ has at most* 4 *vertices. Also, G' has a diameter of* 4.

Proof. In G', any vertex $v \in A' \setminus \{y_1, y_5\}$, is at a distance at most 2 away from any vertex in A' and at a distance at most 3 from a vertex in B'. Similarly, any vertex $u \in B' \setminus \{x_1, x_5\}$ is at a distance at most 2 away from a vertex in B' and at a distance at most 3 from a vertex in A'. From the above arguments, we can see that any shortest path with 5 vertices has to start and end with one of the following vertices: $\{y_1, y_5, x_1, x_5\}$. We can see the distance between any two vertices in the set is 4. ◇

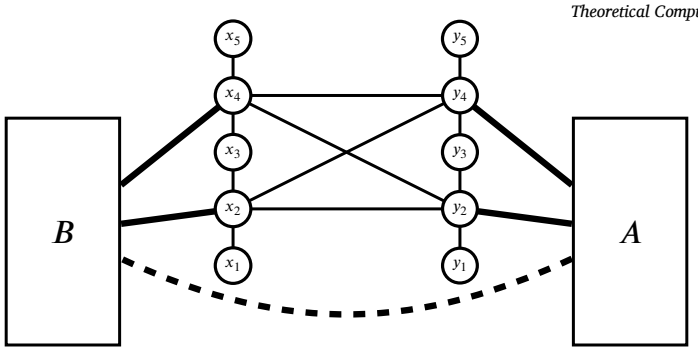


Fig. 3. A sketch of $G' = (V', E')$; thick lines mean that all the edges across two sets are present. The thick dashed line represents all edges that have been present between A and B already in G.

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

Claim 3.9. *There can be only two shortest paths in G' that have five vertices and that can simultaneously exist in a path partition.*

Proof. The above claim is true, as the starting and ending vertices of shortest path having 5 vertices must start and end at $\{y_1, y_5, x_1, x_5\}$. ◇

Notice that Claim 3.7 and Claim 3.8 together guarantee the additional properties of the constructed graph G' that have been claimed in Theorem 3.6.

Claim 3.10. *Let $u, v \in V$ have distance $d \leq 3$ in G. Then, they also have distance d in G'. Hence, if P is a shortest path on at most four vertices in G, then P is also a shortest path in G'.*

Proof. This observation is trivial if $d = 0, 1$; it follows for $d = 2$ as (also) the graph G' is bipartite by Claim 3.7. For $d = 3$, the distance remains the same, as no edge is added between $u \in A$ and $v \in B$ in G'. ◇

Now, we claim that G is a Yes-instance of 4-USPP if and only if G' has a shortest path partitioning of cardinality $k' = k + 2$, where $|V| = 4k$. For the forward direction, let $\mathbb{P}$ be any solution of 4-USPP for G containing k shortest paths. To construct a solution $\mathbb{P}'$ of USPP 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 $\mathbb{P}$. By Claim 3.10, every path $P \in \mathbb{P}$ is in fact a shortest path in G'. Hence, $\mathbb{P}'$ 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 $\mathbb{P}'$, where $|\mathbb{P}'| = k' = k + 2$. As $|V'| = 4k + 10$ by construction, we know by Claim 3.8 and Claim 3.9 that $\mathbb{P}'$ contains k paths of length three and two paths of length 4. The only two possible paths of length 4 following the condition are $\{x_1x_2x_3x_4x_5, y_1y_2y_3y_4y_5\}$, otherwise we would create independent vertices y_3 or x_3. Hence,

$$\{x_1x_2x_3x_4x_5, y_1y_2y_3y_4y_5\} \subseteq \mathbb{P}'$$

and the rest of the paths of $\mathbb{P}'$ are of length 3 and consists of vertices from V only. Let $\mathbb{P} = \mathbb{P}' \setminus \{x_1x_2x_3x_4x_5, y_1y_2y_3y_4y_5\}$. As the k paths of $\mathbb{P}$ are each of length 3 also in G (by Remark 2.1) and as they cover V completely, $\mathbb{P}$ provides a solution to the 4-USPP instance G. □

4. W[1]-hardness for DAGSPP and DAGIPP

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. *DAGSPP and DAGIPP are W[1]-hard when parameterized by solution size.*

The following reduction is non-trivially adapted from [75]. The reduction starts out from Clique and produces an array-like layout of the input graph as an instance of DAGSPP (or of DAGIPP), as can be seen in the figures on the following pages. The construction is quite technical and is now described in details.

Proof. We define a parameterized reduction from Clique to DAGSPP (or to DAGIPP), all parameterized by solution size. We focus on DAGSPP, but all the arguments work for DAGIPP as well. Let (G, k) be an instance of Clique, where $k \in \mathbb{N}$ and $G = (V, E)$,

with $V = [n]$ for simplicity. We construct an equivalent instance (G', k') of the DAGSPP problem, where $G' = (V', E')$ is a DAG and $k' = \frac{k\cdot(k-1)}{2} + 3k$. We will now formally describe V' and E' in the following and later explain their functionalities. To avoid trivialities as well as many corner cases, we tacitly assume $k \geq 2$ in the following.

We define the vertex set first. For $i \in [k]$ and $u \in [n]$, let

$$V^{i,u} = \{a_l^{i,u}, b_l^{i,u} \mid l \in [k] \setminus i\}$$

be the vertex set of our *main gadgets* $G^{i,u}$. For $i \in [k]$, the *dummy gadgets* $G^{i,0}$ and $G^{i,n+1}$ have vertex sets

$$V^{i,0} = \{a_l^{i,u}, b_l^{i,u} \mid l \in \{1,2\}\} \text{ and } V^{i,n+1} = \{a_l^{i,u}, b_1^{i,n+1}, b_2^{i,n+1} \mid l \in [k+2]\},$$

respectively. Then, define

$$V_{\text{gadgets}} := \bigcup_{i\in[k]} \bigcup_{u\in\{0\}\cup[n+1]} V^{i,u}.$$

As we think of the gadgets being arranged in an array, we also speak of the i^{th} *row* (containing $V^{i,u}$ and more) in the following, as well as of the u^{th} *column*. For each row $i \in [k]$, we also have *row start terminals* s_i, s'_i (collected in the set V_{rst}) and *row target terminals* t_i, t'_i (collected in the set V_{rtt}). Also, for each $i \in [k]$, we have $2(k-i)$ *column start terminals* $s_{i,j}$ and $s'_{i,j}$, where $k \geq j \geq i+1$, (collected in the set V_{cst}), as well as $2(i-1)$ *column target terminals* $t_{j,i}$ and $t'_{j,i}$, with $1 \leq j \leq i-1$ (collected in the set V_{ctt}). This gives the set of terminals $V_{\text{terminals}} := V_{\text{rst}} \cup V_{\text{rtt}} \cup V_{\text{cst}} \cup V_{\text{ctt}}$. Altogether, we have $V' := V_{\text{gadgets}} \cup V_{\text{terminals}}$.

Now, we describe the arc set. Let us first describe the arcs within the gadgets. For the main gadgets $G^{i,u}$, with $i \in [k]$ and $u \in [n]$, we set

$$E^{i,u} = E^{i,u}_{\text{down}} \cup E^{i,u}_{\text{right}} \cup E^{i,u}_{\text{bridge}},$$

where $E^{i,u}_{\text{down}} = \{(a_j^{i,u}, b_j^{i,u}) \mid j \in [k] \setminus \{i\}\}$, $E^{i,u}_{\text{right}} = \{(a_j^{i,u}, a_{j+1}^{i,u}), (b_j^{i,u}, b_{j+1}^{i,u}) \mid j \in [k] \setminus \{i-1, i\}\}$, and $E^{i,u}_{\text{bridge}} = \{(a_{i-1}^{i,u}, a_{i+1}^{i,u}), (b_{i-1}^{i,u}, b_{i+1}^{i,u})\}$. As visualized in Fig. 4, one can think of an array layout also for $G^{i,u} = (V^{i,u}, E^{i,u})$; then $e \in E^{i,u}_{\text{down}}$ go downwards, $e \in E^{i,u}_{\text{right}}$ point to the right, in principle always to the next vertex in the index ordering, except for the i^{th} column within $G^{i,u}$ that is bridged by $E^{i,u}_{\text{bridge}}$.

For the dummy gadgets $G^{i,0}$, we have $E^{i,0} = \{(a_1^{i,0}, a_2^{i,0}), (b_1^{i,0}, b_2^{i,0})\}$ and for the dummy gadgets $G^{i,n+1}$, $E^{i,n+1} = \{(a_l^{i,n+1}, a_{l+1}^{i,n+1}) \mid l \in [k+1]\} \cup \{(b_1^{i,n+1}, b_2^{i,n+1})\}$. This describes all arcs within the gadgets $G^{i,u} = (V^{i,u}, E^{i,u})$, i.e., $G^{i,u} = G'[V^{i,u}]$ for all $i \in [k]$ and $u \in \{0\} \cup [n+1]$.

Next, we describe (most of) the connections between terminals. For the row terminals, let $E_{\text{rst}} = \{(s_i, s'_i) \mid i \in [k]\}$ and $E_{\text{rtt}} = \{(t'_i, t_i) \mid i \in [k]\}$. Then, $(V_{\text{rst}}, E_{\text{rst}}) = G'[V_{\text{rst}}]$ and $(V_{\text{rtt}}, E_{\text{rtt}}) = G'[V_{\text{rtt}}]$. For the column terminals, let $E_{\text{cst}} = \{(s_{i,j}, s'_{i,j}) \mid 1 \leq i < j \leq k\}$ and $E_{\text{ctt}} = \{(t'_{j,i}, t_{j,i}) \mid 1 \leq j < i \leq k\}$. Then, $(V_{\text{cst}}, E_{\text{cst}}) = G'[V_{\text{cst}}]$ and $(V_{\text{ctt}}, E_{\text{ctt}}) = G'[V_{\text{ctt}}]$.

The remaining description of the set of arcs of G' is dedicated to arcs between certain gadgetries. Here, first we describe all arcs that connect main gadgets with other main gadgets or with dummy gadgets. We have two arcs which connect two consecutive gadgets in a row $i \in [k] \setminus \{1, k\}$:

$$E^i_{\text{consec}} = \{(a_2^{i,0}, a_1^{i,1}), (b_2^{i,0}, b_1^{i,1})\} \cup \{(a_k^{i,u}, a_1^{i,u+1}), (b_k^{i,u}, b_1^{i,u+1}) \mid u \in [n]\}.$$

The two cases $i = 1$ and $i = k$ are special because then, the first (or last, respectively) column are missing in the main gadget.

$$E^1_{\text{consec}} = \{(a_2^{1,0}, a_2^{1,1}), (b_2^{1,0}, b_2^{1,1})\} \cup \{(a_k^{1,u}, a_2^{1,u+1}), (b_k^{1,u}, b_2^{1,u+1}) \mid u \in [n-1]\}$$
$$\cup \{(a_k^{1,n}, a_1^{1,n+1}), (b_k^{1,n}, b_1^{1,n+1})\} \text{ and}$$
$$E^k_{\text{consec}} = \{(a_2^{k,0}, a_1^{k,1}), (b_2^{k,0}, b_1^{k,1})\} \cup \{(a_{k-1}^{k,u}, a_1^{k,u+1}), (b_{k-1}^{1,u}, b_1^{k,u+1}) \mid u \in [n]\}.$$

Moreover, we have some *skipping arcs* between the gadgets, namely, for each row $i \in [k] \setminus \{1, k\}$, we set

$$E^i_{\text{skip}} = \{(a_2^{i,0}, b_1^{i,2})\} \cup \{(a_k^{i,u}, b_1^{i,u+2}) \mid u \in [n-1]\},$$

again with two special cases for $i = 1$ and $i = k$, resulting in

$$E^1_{\text{skip}} = \{(a_2^{1,0}, b_2^{1,2})\} \cup \{(a_k^{1,u}, b_2^{1,u+2}) \mid u \in [n-2]\} \cup \{(a_k^{1,n-1}, b_1^{1,n+1})\} \text{ and}$$
$$E^k_{\text{skip}} = \{(a_2^{k,0}, b_1^{k,2})\} \cup \{(a_{k-1}^{k,u}, b_1^{k,u+2}) \mid u \in [n-1]\}.$$

The arcs from $\bigcup_{i\in[k]}(E^i_{\text{consec}} \cup E^i_{\text{skip}})$ are also illustrated in Fig. 6.

To model the edges of the original graph G, we need the following arcs

$$E_{\text{con}} = \{(b_j^{i,u}, a_l^{j,v}) \mid uv \in E, 1 \leq i < j \leq k\}.$$

These arcs are also illustrated in Fig. 5. Apart from some shortcut arcs defined below, we have all arcs of $G'[V_{\text{gadgets}}]$ collected in

$$E_{\text{gadgets}} = \left(\bigcup_{i\in[k]} \left(\bigcup_{u\in\{0\}\cup[n+1]} E^{i,u} \right) \cup E^i_{\text{skip}} \cup E^i_{\text{consec}} \right) \cup E_{\text{con}} .$$

Next, we have some arcs that connect the row start and row target terminal to the dummy gadget:

$$E^i_{\text{stcon}} = \{(s'_i, a^{i,0}_1), (b^{i,n+1}_2, t'_i) \mid i \in [k]\} .$$

Similarly, we have some arcs that connect column terminals to the main gadgets: for $i \in [k]$, define

$$E^i_{\text{upcon}} = \{(s'_{i,j}, a^{i,u}_j) \mid 1 \le i < j \le k, u \in [n]\} ,$$
$$E^i_{\text{downcon}} = \{(b^{i,u}_j, t'_{j,i}) \mid 1 \le j < i \le k, u \in [n]\} .$$

We then set for all these terminal connectors

$$E_{\text{tercon}} = \bigcup_{i\in k} \left(E^i_{\text{stcon}} \cup E^i_{\text{upcon}} \cup E^i_{\text{downcon}} \right) .$$

We have some arcs that help enforce the start and target vertices of the paths. Their subscripts indicate if they force some row/column start to some row/column target (as in the first case) or other combinations. In the following, let $i \in [k]$.

$$E^i_{\text{frsrt}} = \{(s_i, t_l), (s_i, t'_l), (s'_i, t_l), (s'_i, t'_l) \mid i < l \le k\} ,$$

with $E^k_{\text{frsrt}} = \emptyset$ as a corner case,

$$E^i_{\text{frsct}} = \{(s_i, t_{j,l}), (s_i, t'_{j,l}), (s'_i, t_{j,l}), (s'_i, t_{j,l}) \mid i \le l \le k, j \in [l-1]\} ,$$

with $E^1_{\text{frsct}} = \emptyset$ as a corner case,

$$E^i_{\text{fcsrt}} = \{(s_{i,j}, t_l), (s'_{i,j}, t_l), (s_{i,j}, t'_l), (s'_{i,j}, t'_l) \mid i < j \le k, i \le l \le k\} ,$$

with $E^k_{\text{fcsrt}} = \emptyset$ as a corner case, and

$$E^i_{\text{fcsct}} = \{(s_{i,j}, t_{l,h}), (s_{i,j}, t'_{l,h}), (s'_{i,j}, t_{l,h}), (s'_{i,j}, t'_{l,h}) \mid i < j \le k, i \le h \le k, l \in [h-1], (i,j) \ne (h,l)\} .$$

We also connect the dummy gadget to some target vertices as follows:

$$E^i_{\text{fbrt}} = \{(b^{i,0}_1, t_l), (b^{i,0}_1, t'_l), (b^{i,0}_2, t_l), (b^{i,0}_2, t'_l) \mid i \le l \le k\} ,$$
$$E^i_{\text{fbct}} = \{(b^{i,0}_1, t_{j,l}), (b^{i,0}_1, t'_{j,l}), (b^{i,0}_2, t_{j,l}), (b^{i,0}_2, t_{j,l}) \mid i \le l \le k, j \in [l-1]\} .$$

So we can collect all these enforcing arcs into one arc set:

$$E_{\text{force}} = \bigcup_{i\in[k]} \left(E^i_{\text{frsrt}} \cup E^i_{\text{frsct}} \cup E^i_{\text{fbrt}} \cup E^i_{\text{fbct}} \cup E^i_{\text{fcsrt}} \cup E^i_{\text{fcsct}} \right) .$$

To prove Claim 4.3, we need the following arcs (for row $i \in [k]$) that connect start terminals to the rightmost dummy gadget:

$$E^i_{\text{spfrst}} = \{(s_i, a^{l,n+1}_q), (s'_i, a^{l,n+1}_q) \mid i \le l \le k, q \in [k+2]\} ,$$
$$E^i_{\text{spfcst}} = \{(s_{i,j}, a^{l,n+1}_q), (s'_{i,j}, a^{l,n+1}_q) \mid i \le l \le k, i < j \le k, q \in [k+2]\} .$$

Finally, we also connect the bottom main gadget vertices (and those of the leftmost dummy gadget) to the top vertices of the rightmost dummy gadget, this way defining the set E^i_{spfbot} as

$$\bigcup_{q\in[k+2]} \bigcup_{i<j\le k} \left(\left\{ (b^{i,u}_l, a^{j,n+1}_q) \mid u \in [n], l \in [k] \setminus \{i\} \right\} \cup \{(b^{i,0}_1, a^{j,n+1}_q), (b^{i,0}_2, a^{j,n+1}_q)\} \right) .$$

We collect these arc sets into the set of *shortest path enforcers* that provide the shortcuts mentioned above.

$$E_{\text{spf}} = \bigcup_{i\in[k]} \left(E^i_{\text{spfcst}} \cup E^i_{\text{spfrst}} \cup E^i_{\text{spfbot}} \right) .$$

We can now define

$$E' = E_{\text{rst}} \cup E_{\text{rtt}} \cup E_{\text{cst}} \cup E_{\text{ctt}} \cup E_{\text{gadgets}} \cup E_{\text{tercon}} \cup E_{\text{force}} \cup E_{\text{spf}} .$$

This concludes the formal description of $G' = (V', E')$.

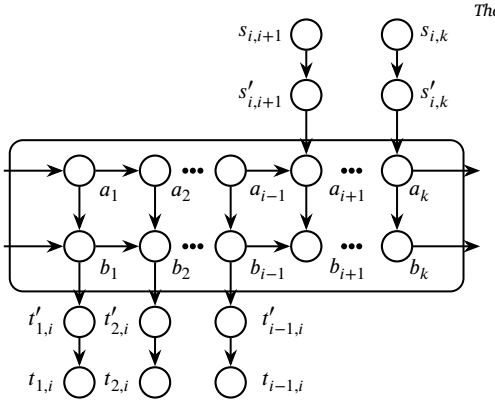


Fig. 4. Gadget $G^{i,u}$, $0 < u \leq n$, $i \in [k]$.

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 DAGSPP 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 the DAGSPP instance. 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, see Fig. 5.

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 row start terminals, connected by an arc $(s_i, s'_i) \in E_{\text{rst}}$, and row target terminals, connected by an arc $(t'_i, t_i) \in E_{\text{rtt}}$. Next, add arcs starting from s_i, s'_i to t_l and t'_l, with $l > i$, see E^i_{frsrt}. Also, for each row, we have $k - i$ *column start terminals*, arcs $(s_{i,j}, s'_{i,j}) \in E_{\text{cst}}$ with $i < j$, and $i - 1$ *column target* terminals, arcs $(t'_{j,i}, t_{j,i}) \in E_{\text{rtt}}$ with $j \in [i-1]$. Also, add arcs from vertices $s_{i,j}$ and $s'_{i,j}$ to t_l, t'_l if $l \geq i$ and to column target terminals (see E^i_{fcsct} in the formal construction).

Gadgets are denoted by $G^{i,u}$, $i \in [k]$, corresponding to $u \in V$. Each gadget $G^{i,u}$ consists of $k-1$ top-level vertices $a^{i,u}_j$ and of $k-1$ bottom-level vertices $b^{i,u}_j$, connected by arcs $(a^{i,u}_j, b^{i,u}_j) \in E^{i,u}_{\text{down}}$, with $j \in [k] \setminus \{i\}$. Due to the natural ordering of the vertices within a gadget, we can also speak of the first vertex on the top level of a gadget or of the last vertex on the bottom level of a gadget. On both levels, the vertices are also connected following their natural 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. The formal description of the edge set $E^{i,u}$ is a bit more technical, because one index of the set $[k]$ is missing in the column indexing. The arcs within a gadget described so far lead to a grid-like drawing of each gadget, see Fig. 4.

To each row $i \in [k]$, we also add *dummy gadgets* $G^{i,0}$ and $G^{i,n+1}$. Gadget $G^{i,0}$ consists of two arcs, $(a^{i,0}_1, a^{i,0}_2)$ and $(b^{i,0}_1, b^{i,0}_2)$, also add arcs from $b^{i,0}_1$ and $b^{i,0}_2$ to t_h, t'_h and to $t'_{l,h}$, $t_{l,h}$ where $l < h$ and $i \leq h$, see the formal definition of E^i_{fbrt} and of E^i_{fbct}. Gadget $G^{i,n+1}$ consists of a directed P_{k+2} and an arc $(b^{i,n+1}_1, b^{i,n+1}_2)$; the P_{k+2}-vertices are named $a^{i,n+1}_j$, with $j \in [k+2]$, where $a^{i,n+1}_{k+2}$ has out-degree zero. The arcs inside the dummy gadgets are formally defined as the sets $E^{i,0}$ and $E^{i,n+1}$ above. We can speak of

$$T^i := \{a^{i,u}_r \mid r \in [k] \setminus \{i\}, u \in [n]\} \cup \{a^{i,0}_1, a^{i,0}_2, a^{i,n+1}_j \mid j \in [k+2]\}$$

as being on the i^{th} *top level*, while the vertices

$$B^i := \{b^{i,u}_r \mid r \in [k] \setminus \{i\}, u \in [n]\} \cup \{b^{i,0}_1, b^{i,0}_2, b^{i,n+1}_1, b^{i,n+1}_2\}$$

form the i^{th} *bottom level*, where $i \in [k]$.

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 $G^{i,u+1}$, etc. Analogously, the vertices of the lower level of the gadgets are connected. For a formal treatment, we refer to the definition of E^i_{consec}. These notions are illustrated in Fig. 4 and Fig. 6. In the figures, for readability we omit the superscripts and simply write $(a_l, b_l) = (a^{i,u}_l, b^{i,u}_l)$ where $l \in [k]$. Also, notice that the mentioned vertex names a_1, b_1, a_k and b_k are rather meant in a symbolic fashion. Clearly, by our construction, the first vertex of the top level of $G_{1,u}$ would be $a^{1,u}_2$, to explicitly mention one exceptional case.

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 $(s'_i, a^{i,0}_1)$ and $(b^{i,n+1}_2, t'_i)$ for row i, see the definition of E^i_{stcon}, as well as *skipping arcs* that allow to skip a gadget $G^{i,u}$ for $u \in V$. Such an arc connects the last vertex of the top level of $G^{i,u-1}$ to the first vertex of the bottom level of $G^{i,u+1}$, see Fig. 6 for an illustration and the definition of E^i_{skip} for a formal treatment.

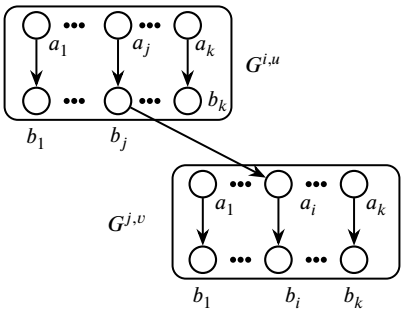


Fig. 5. $G^{i,u}$ connected to $G^{j,v}$, $i < j$.

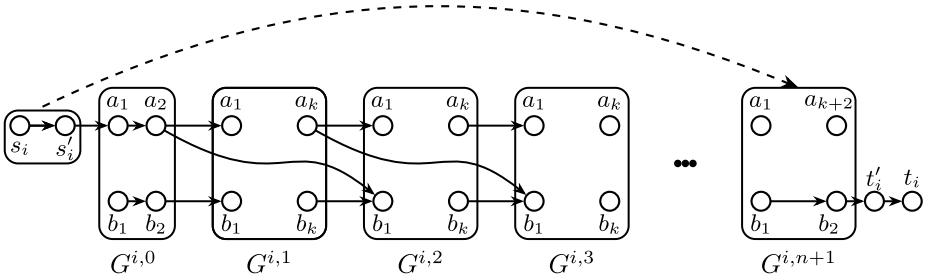


Fig. 6. The i^{th} row (only the first two skipping arcs are shown), the dashed line indicates arcs (shortest paths enforcers) from s_i, s'_i to $a_j^{i,n+1}$, with $j \in [k+2]$.

A **verifier** is a path that routes through one of the vertices of the skipped gadget and connects column terminals $s_{i,j}$ to $t_{i,j}$. In order to implement this, we add the arcs $(b_1^{i,0}, t'_{i,j})$, $(b_1^{i,0}, t_{i,j})$, $(b_2^{i,0}, t_{i,j})$ and $(b_2^{i,0}, t'_{i,j})$ to every gadget in rows i and j. See the formal definition of E^i_{fbct}. To connect the gadget in row i and j, for every edge uv in G, we add an arc $(b_j^{i,u}, a_i^{j,v})$ for each $i < j$, see Fig. 5 for an illustration and the definition of E_{con} for formalities.

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*, see the definition of E_{spf}. For every row i, from all the bottom vertices of gadget $G^{j,u}$, where $j < i$ and $0 \le u \le n$, from every vertex s_l and s'_l of a row start terminal and from every vertex $s_{l,h}$, $s'_{l,h}$ of a column start terminal, where $l \le i$ and $l < h$, add arcs to $a_j^{i,n+1}$, with $j \in [k+2]$, see Fig. 6. The idea is, except for T^i, for all the vertices which can reach $a_j^{i,n+1}$, we add an arc connecting it to $a_j^{i,n+1}$.

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 and end vertex of each path in the solution have the desired properties.

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

Proof. First, recall that $k' = \frac{k\cdot(k-1)}{2} + 3k$. Hence, we have to show that we have $\frac{k\cdot(k-1)}{2} + 2k$ many vertices with in-degree zero. Namely, for each row i, with $i \in [k]$, we have one such vertex as its row start terminal s_i, $k - i$ column start terminals and one vertex of the dummy gadget $b_1^{i,0}$. We also have $\frac{k\cdot(k-1)}{2} + 2k$ vertices with out-degree zero, for each row i, $i \in [k]$, one as the row's target terminal t_i, $i - 1$ as its column target terminals and as a vertex of the dummy gadget $a_{k+2}^{i,n+1}$, where $|V| = n$. ◇

Hence, we know that $k' - k$ start and $k' - k$ 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 vertices of all the paths in the solution.

Claim 4.3. *If a solution $\mathbb{P}$ of the created DAGSPP instance G' is of size k' then, for each row $i \in [k]$, there is a path in $\mathbb{P}$ starting from $v \in T^i$ which covers all the vertices of $\{a_j^{i,n+1} \mid j \in [k+2]\}$.*

Proof. First, we will prove that there is a path in $\mathbb{P}$ starting from $v \in T^i$ which covers at least one vertex from $\{a_j^{i,n+1} \mid j \in [k+2]\}$. We will prove this by contradiction. Assume that there is an spp $\mathbb{P}$ such that there exists a row i, with $i \in [k]$, such that there is no shortest path starting from v, where $v \in T^i$, which covers any of the vertices $a_j^{i,n+1}$, with $j \in [k+2]$. Then, to cover the vertices $a_j^{i,n+1}$, with $j \in [k+2]$, $\mathbb{P}$ needs $k+2$ paths, as every vertex that can reach $a_l^{i,n+1}$, where $l \in [k+2]$, is connected to all the vertices $a_j^{i,n+1}$, where $j \in [k+2]$. Hence, we require $k+2$ paths to just cover $a_j^{i,n+1}$ for $j \in [k+2]$, but this leaves G' with $k' - k - 1$ out-degree

zero vertices. This contradicts that $\mathbb{P}$ is of size k'. Next, it is easy to see that if a path starting at $v \in T^i$ ends at some $a_q^{i,n+1}$ where $q \in [k+1]$, it must cover all the vertices $a_l^{i,n+1}$ where $l < q$, as there is only one shortest path joining v and $a_q^{i,n+1}$. Finally, we can observe that we can extend the path starting at $v \in T^i$ to end at $a_{k+2}^{i,n+1}$, with $i \in [k]$, otherwise it would violate the small size of the partition. ◇

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 ($v \in T^i$) and $s_{i,j}$, with $i < j$. In the next claims, we will show observations about the end vertices of these paths.

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

Proof. Let $\mathbb{P}$ be a DAGSPP solution of size k'. The shortest path from $\mathbb{P}$ that covers t_i cannot have started at s_j, where $j < i$, as this path would consist of two vertices due to the arc (s_j, t_i) and hence $\mathbb{P}$ would also contain a path starting at s'_j, which is a contradiction to its small size. Similar arguments work if the path would start at any column start terminal $s_{l,m}$ or at any $b_1^{l,0}$, with $l < m$, $l \leq i$. Also, the path does not start at $v \in T^m$, with $m \leq i$, as this has to go through $a_1^{m,n+1}$ and there is no path from $a_1^{m,n+1}$ to t_i. Hence, the path should start at vertex s_i. ◇

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 $t_{i,j}$, with $i < j$.*

Proof. After Claim 4.3 and Claim 4.4, we are left with $k + \frac{k(k-1)}{2}$ paths to cover the remaining graph. In the remaining graph, there are $k + \frac{k(k-1)}{2}$ vertices with in-degree 0, these are (1) $s_{i,j}$, for $i < j$ and $j \leq k$ and (2) $b_1^{i,0}, i \in [k]$. The path starting at $b_1^{l,0}$ where $l \in [k]$ and $l \leq i$ cannot cover $t_{i,j}$, as the path would be the arc $(b_1^{l,0}, t_{i,j})$, hence creating one more in-degree 0 vertex, i.e., $b_2^{l,0}$. Also, path starting at $s_{i,j}$ cannot end at $t_{m,l}$ where $1 < m < l \leq k$ for $k \neq i$ and $l \neq j$, due to the arcs E_{fcsct} or as there is no path from $s_{i,j}$ to reach it. Therefore, a path starting from $s_{i,j}$ has to end at $t_{i,j}$, with $i < j$. ◇

Therefore, 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 $t_{i,j}$, with $j > i$. Once we skip a gadget, we are at the bottom level of the row, and hence we cannot skip again. 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 $b_1^{i,0}$ and the top of the row is covered by a path starting at $v \in T^i$ 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 4.6. *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 4.7. *G has a k-clique if and only if G' has a shortest path partition of size k'.*

Proof. If vertices $v_1, v_2, \ldots, v_k$ form a clique in G, then we construct an spp of size k' in G' as follows: for each row i, with $i \in [k]$, we take a path starting at s_i and leading to t_i while skipping gadget G_{i,v_i}; we cover all the vertices on the left of the skipped gadget with a single path starting at $b_1^{i,0}$, and similarly, we cover all the vertices on the right of the skipped gadget with a single path ending at $a_{k+2}^{i,n+1}$. Then, we connect $s_{i,j}$ to $t_{i,j}$, with $i < j$, through the skipped gadget (this is possible because of Claim 4.6).

For the other direction, assume we have an spp $\mathbb{P}$ of size k' in G': include the vertices corresponding to any gadget which was skipped by a path which connects s_i to t_i for $i \in [k]$ into the k-element set C. This forms a clique in G, as any two vertices of $\{v_i, v_j\} \in C$, are distinct and have an edge between them because of the verifiers. ◇

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

5. XP algorithms for SPP

We now present our XP algorithms to prove the following result. We note that the result for USPP was also proved in [26] (with a different method: they first give an upper bound on the treewidth of the graph, and then they apply Courcelle's theorem). The basic idea is to use existing algorithmic results for Disjoint (Shortest) Paths from the literature.

Theorem 5.1. *The following problems are in XP when parameterized by the number of paths: USPP, DAGSPP, and DSPP when restricted to planar directed graphs, to directed graphs of bounded directed treewidth, or when $k = 2$.*[4]

[4] See [47] for the definition of *directed treewidth*.

For our XP algorithms, the following combinatorial result is crucial, that allows us to reduce the problem to DP (without the shortest path requirement).

Lemma 5.2. *Let $G = (V, E)$ be a graph (directed or undirected). 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 $\sum_{i=1}^{k} d(s_i, t_i) = |V| - k$.*

Proof. Let $G = (V, E)$ be a directed or an undirected graph. If V can be partitioned into k vertex-disjoint shortest paths P_i, for $1 \leq i \leq k$, then we can take their endpoints as s_i and t_i. As the paths are shortest and form a partition, $\sum_{i=1}^{k} d(s_i, t_i) = |V| - k$ must hold. Conversely, if there are k vertex-disjoint paths in G, connecting, say, s_i and t_i, for $1 \leq i \leq k$, then the condition $\sum_{i=1}^{k} d(s_i, t_i) = |V| - k$ can only be satisfied if each vertex of the graph belongs to exactly one of the paths and if all the paths are shortest paths. Hence, V can be partitioned into k vertex-disjoint shortest paths. □

Lemma 5.2 allows us to prove Theorem 5.1 by enumerating all possible $X := \{(s_1, t_1), \ldots, (s_k, t_k)\}$, resulting in an instance of DSP. Now, we can either apply known algorithms with XP running time (or better), for either DP or DSP. We give details of this proof strategy in the following.

Proof of Theorem 5.1. Let $G = (V, E)$, together with an integer k, be an instance of SPP (i.e., DSPP, DAGSPP or USPP, the arguments are complete analogous for these cases). By Lemma 5.2, given an instance (G, k) of SPP, with $G = (V, E)$, it is sufficient to iterate (in XP time) through all possible $X := \{(s_1, t_1), \ldots, (s_k, t_k)\}$ where $s_i, t_i \in V$ and $\sum_{i=1}^{i=k} (d(s_i, t_i) + 1) = |V|$. The latter can be checked in polynomial time using standard algorithms for computing the distance. Then, for each such X we form an instance (G, X) of the version of DP (or DSP) corresponding to our initial instance (i.e., DDP, DAGDP or UDP, or alternatively, DDSP, DAGDSP or UDSP). Now, (G, k) is a Yes-instance of SPP, respectively, if and only if some constructed (G, X) instance is a Yes-instance of DP (or of DSP).

We next see in which cases DP or DSP can be solved in XP time by known algorithms.

For DAGs, Fortune et al. [31, Theorem 3] showed that, for arbitrary fixed directed pattern graphs, subgraphs homeomorphic to them can be detected in DAGs in polynomial time (even when the node-mapping is fixed). Hence, taking as a pattern graph a collection of k isolated directed edges and as node-mapping the chosen terminals, the according homeomorphism problem is identical to DAGDP (with fixed k). Hence, the instance (G, X) can be solved in time $\mathcal{O}(|V|^{f(|X|)}) = \mathcal{O}(|V|^{f(k)})$ for some function f. Alternatively, an XP algorithm for DAGDSP is also given in [8].

For planar (directed) graphs, we refer to [21] for an FPT algorithm for DDP, or to [8] for an XP algorithm for DDSP. An XP algorithm is given for DDP on graphs of bounded directed treewidth [47]. For the case of $k = 2$, see [8]. Note that results for other classes exist and can be applied, see e.g. [5].

For the undirected case, we can refer to the seminal paper of Robertson and Seymour [71], yielding a cubic-time algorithm on input (G, X) for UDP, an algorithm that was later improved to be quadratic in [49], yielding again the XP-claim. Alternatively, XP algorithms for UDSP were also given in [7,57].

These results establish the statement. □

Notice that these algorithmic results rule out paraNP-hardness results (in contrast to those existing for PP and IPP) for the corresponding problems.

6. Neighborhood diversity and vertex cover parameterizations

One of the standard structural parameters studied within parameterized complexity is the *vertex cover number*, i.e., the size of the smallest vertex cover that a graph has. 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 [54].

6.1. Neighborhood diversity for USPP and UIPP

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

Proposition 6.1. *If G is connected, then any induced path has length at most $\mathsf{nd}(G)$. In particular,* $\mathrm{diam}(G) \leq \mathsf{nd}(G)$.

Proof. Let P be an induced path with k vertices in G (note that any shortest path, in particular a diametrical path, is also an induced path). Note that if $k \geq 4$, no two vertices of P are twins in G, and so, every vertex of P needs to be in a different neighborhood diversity class, and so, the claim is true (in fact even stronger, $k \leq \mathsf{nd}(G)$). Hence, if $\mathsf{nd}(G) \geq 2$, we are done: G does not contain an induced path of length at least $\mathsf{nd}(G) + 1$ (i.e. with $k \geq \mathsf{nd}(G) + 2$ vertices). If $k \leq 3$, then the endpoints of P may be twins and thus, may belong to the same neighborhood diversity class, if that class forms an independent set (but no other pair of vertices of P may lie in the same class). Thus, if $\mathsf{nd}(G) \leq 2$, G may contain an induced with $\mathsf{nd}(G) + 1$ vertices, but no more. □

We will also use the following result.

Theorem 6.2 *([20, Theorem 6.5]). An* INTEGER LINEAR PROGRAMMING *instance of size L with p variables can be solved using*

$$\mathcal{O}(p^{2.5p+o(p)} \cdot (L + \log M_x) \log(M_x M_c))$$

arithmetic operations and space polynomial in $L + log M_x$, where M_x is an upper bound on the absolute value a variable can take in a solution, and M_c is the largest absolute value of a coefficient.

Now, call two induced 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 neighborhood diversity classes. Any two equivalent induced paths have the same length. Proposition 6.1 and our discussions imply that there are $2^{\mathcal{O}(\mathsf{nd}(G))}$ many equivalence classes of induced paths (+).

Theorem 6.3. *UIPP and USPP are FPT when parameterized by neighborhood diversity.*

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

Recall that there exists a partition of $V(G)$ into d nd-classes $C_1, \ldots, C_d$, and each C_i for $i \in [d]$ either induces a clique or an independent set and all its vertices are mutual twins. As mentioned before, such a partition can be computed in linear time using [44, Algorithm 2].

Compute (in time $2^{\mathcal{O}(\mathsf{nd}(G))}$, by (+)) the set of all $2^{\mathcal{O}(\mathsf{nd}(G))}$ induced path equivalence classes; this can be represented by a set $\mathcal{P}$ of *types* of induced paths (each represented by one representative induced path), in which any two induced paths are not equivalent. For USPP, we discard those classes that do not yield shortest paths. Construct and solve the following Integer Linear Program (ILP), with a variable $z_P \geq 0$ corresponding to each type of path $P \in \mathcal{P}$. Each $P \in \mathcal{P}$ is characterized by a vector $(P^1, \ldots, P^d)$ with $P^j = |C_j \cap P|$ (note that, as argued in the proof of Proposition 6.1, $P^j \leq 2$).

$$\text{minimize} \sum_{P \in \mathcal{P}} z_P$$

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

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

Claim 6.4. *There exists an ipp (respectively spp) of G with k induced (respectively, shortest) paths if and only if the objective function attains the value k in the ILP described above.*

Proof. For the forward direction, consider the induced/shortest paths $\mathbb{P} = \{P_1, P_2, \ldots, P_k\}$ as a solution. To construct a solution of the ILP, for every $P \in \mathcal{P}$, z_P equals the number of paths in $\mathbb{P}$ that are equivalent to P. Clearly, $\sum_{P \in \mathcal{P}} z_P = k$. The constraint is satisfied as every vertex is covered exactly once by $\mathbb{P}$.

For the other direction, suppose that there exist integers $\{z_P \mid P \in \mathcal{P}\}$ which satisfy the ILP and $\sum_{P \in \mathcal{P}} z_P = k$. We can produce an ipp/spp of G with k paths as follows. As long as there exists a set $P \in \mathcal{P}$ with $z_P > 0$, construct a path by picking exactly P^j vertices from each C_j. Since the vertices in C_j form either a clique or an independent set of mutual twins, this choice can safely be arbitrary. Since P corresponds to a valid type of induced/shortest path, the set of selected vertices forms such a path. Remove these vertices from G, and set $z_P := z_P - 1$. This algorithm produces exactly k paths which form an ipp/spp that covers each vertex exactly once because of the satisfied constraints of the ILP. ◇

Notice that the number of variables of the ILP is exactly $|\mathcal{P}|$, which is $2^{\mathcal{O}(\mathsf{nd}(G))}$, by (+). Now, we apply Theorem 6.2 with $p = 2^{\mathcal{O}(\mathsf{nd}(G))}$, $L = \mathcal{O}(n)$, $M_x = M_c = n$ to get our FPT claim. □

It is worth mentioning that by Theorem 6.2, the time consumption of the proposed algorithms is at worst double-exponential in the parameter: ignoring lower-order terms, the growth is $\mathcal{O}\left(2^{2.5d \cdot 2^d}\right)$, while the space consumption is rather single-exponential, $\mathcal{O}(2^d)$. In certain cases, however, the number of induced/shortest path equivalence classes of a graph (that determines that number of variables of the derived ILP) need not be exponential in the neighborhood diversity, which would immediately improve the running time estimates for the proposed algorithm. Nonetheless, it would be interesting to give better parameterized algorithms for these path partitioning problems.

Moreover, using the mentioned estimate of Lampis [54, Lemma 2] that a graph with vertex cover number k has neighborhood diversity at most $2^k + k$, we can immediately infer the following result, where the dependence on the parameter is even worse than that for neighborhood diversity.

Corollary 6.5. *USPP and UIPP are FPT when parameterized by vertex cover number.*

Note that we also have a similar result for UPP, either by a more general approach in [34], or based on a more direct and simple reasoning, as shown later in Proposition 6.15.

6.2. Neighborhood diversity of directed graphs for DSPP and DIPP

Now we want to try to adapt these neighborhood diversity results for the directed case. For this reason we want to introduce the new notion of *directed neighborhood diversity* (dnd for short) that might be of independent interest. To this end, let $N^-(v) := \{x \in V \mid (x,v) \in E\}$ and $N^+(v) := \{x \in V \mid (v,x) \in E\}$ be the in- and out-neighborhood of v, respectively. For two vertices $v, u \in V$, we define $v \sim_{dnd} u$ to hold if the following three conditions are satisfied:

1. $N^-(v) \setminus \{u\} = N^-(u) \setminus \{v\}$,
2. $N^+(v) \setminus \{u\} = N^+(u) \setminus \{v\}$, and
3. $(v,u) \in E \Leftrightarrow (u,v) \in E$.

Lemma 6.6. *For any directed graph $G = (V, E)$, the relation $\sim_{dnd}$ is an equivalence relation.*

Proof. Trivially, $\sim_{dnd}$ is reflexive and symmetric. We are left to show transitivity. Let $v, u, w \in V$ be different vertices with $v \sim_{dnd} u$ and $u \sim_{dnd} w$.

First assume that $(v,u),(u,v) \in E$. This implies $v \in N^-(u) \setminus \{w\} = N^-(w) \setminus \{u\}$ and $v \in N^+(u) \setminus \{w\} = N^+(w) \setminus \{u\}$. Hence, $(v,w),(w,v) \in E$. As $N^-(u) \setminus \{v\} = N^-(v) \setminus \{u\}$, $(u,w),(w,u) \in E$. Thus,

$$\begin{aligned} N^-(v) \setminus \{w\} &= ((N^-(v) \setminus \{u\}) \cup \{u\}) \setminus \{w\} = ((N^-(u) \setminus \{v\}) \cup \{u\}) \setminus \{w\} \\ &= ((N^-(u) \setminus \{w\}) \cup \{u\}) \setminus \{v\} = ((N^-(w) \setminus \{u\}) \cup \{u\}) \setminus \{v\} \\ &= N^-(w) \setminus \{v\}. \end{aligned}$$

Analogously, $N^+(v) \setminus \{w\} = N^+(w) \setminus \{v\}$. Therefore, $v \sim_{dnd} w$.

Now assume $(v,u),(u,v) \notin E$. Then, $v \notin N^-(u) \setminus \{w\} = N^-(w) \setminus \{u\}$ and $v \notin N^+(u) \setminus \{w\} = N^+(w) \setminus \{u\}$. Thus, $(v,w),(w,v) \notin E$. Since $N^-(u) \setminus \{v\} = N^-(v) \setminus \{u\}$, $(u,w),(w,u) \notin E$. Hence,

$$\begin{aligned} N^-(v) \setminus \{w\} &= (N^-(v) \setminus \{u\}) \setminus \{w\} = (N^-(u) \setminus \{v\}) \setminus \{w\} \\ &= (N^-(u) \setminus \{w\}) \setminus \{v\} = (N^-(w) \setminus \{u\}) \setminus \{v\} \\ &= N^-(w) \setminus \{v\}. \end{aligned}$$

Therefore, $\sim_{dnd}$ is transitive and hence an equivalence relation. □

Note that the third condition in the definition of relation $\sim_{dnd}$ is needed to obtain an equivalence relation. To see this, consider for example the directed graph on vertices u, v, w consisting of a transitive triangle, where u is the source and w is the sink. Without the third condition, we would have $u \sim v$ and $v \sim w$, but $u \nsim w$, so $\sim$ would not be an equivalence relation.

Now we want to consider the equivalence classes under this equivalence relation. The previous proof provides an observation for these.

Corollary 6.7. *Let $G = (V, E)$ be a directed graph and $C_1, \ldots, C_d$ be the equivalence classes with respect to $\sim_{dnd}$. Then for each $i \in \{1, \ldots, d\}$, C_i is either an independent set or a bi-directed clique.*

A similar result for the neighborhood diversity on undirected graphs is well-known. There, an equivalence class is either an independent set or a clique. But this is not the only similarity, as we see next.

Remark 6.8. If we take an undirected graph $G = (V, E)$ and build the directed graph $G' = (V, E')$ with $E' := \{(v,u),(u,v) \mid \{v,u\} \in E\}$, then $\sim_{nd} = \sim_{dnd}$, where $\sim_{nd}$ is the underlying relation of the undirected neighborhood diversity. This holds, since $N(v) = N^-(v) = N^+(v)$ and $\{v,u\} \in E$ if and only if $(v,u) \in E'$ if and only if $(u,v) \in E'$, which is the case if and only if $\{u,v\} \in E$.

As we can see, the definition above is a natural way to generalize the neighborhood diversity relation from undirected to directed graphs. Therefore, we define the *directed neighborhood diversity* of a graph $G = (V, E)$ (denoted by $\mathsf{dnd}(G)$) as the number of equivalence classes under $\sim_{dnd}$.

Lemma 6.9. *Let $G = (V, E)$ be a directed graph with directed neighborhood diversity classes $C_1, \ldots, C_{\mathsf{dnd}(G)}$. For any induced path $P = v_1, \ldots, v_\ell$ in G, there is at most one $i \in \{1, \ldots, \mathsf{dnd}(G)\}$ such that $|C_i \cap P| > 1$. In this case, $\{v_1, v_\ell\} = C_i \cap P$.*

Proof. Let $P = v_1 \dots v_\ell$ be an induced directed path and C_i be a directed neighborhood diversity class with $\{v_j, v_k\} \subseteq C_i \cap P$ and $v_j \neq v_k$. Without loss of generality, $j < k$. Assume $j \neq 1$. Then $v_{j-1} \in N^-(v_j) \setminus \{v_k\} = N^-(v_k) \setminus \{v_j\}$. This contradicts the fact that P is induced. Therefore, $j = 1$. Analogously, $k = \ell$ (substitute v_{j-1} by v_{k+1} and N^- by N^+). □

This provides the following analogue of Proposition 6.1 for directed graphs.

Corollary 6.10. *Let G be a directed graph. Any induced path has length at most $\mathsf{dnd}(G)$.*

This is a tight bound. Indeed, for any given $d \in \mathbb{N}$, there exists a directed graph G_d with $\mathrm{diam}(G_d) = \mathsf{dnd}(G)$: For $d = 1$, just use a path of two vertices. For the other cases, define $G_d = (V_d, E_d)$ with $V_d := \{v, v', v_1, \dots, v_{d-1}\}$ and $E_d := \{(v_i, v_{i+1}) \mid i \in \{1, \dots, d-2\}\} \cup \{(v_{d-1}, v), (v_{d-1}, v'), (v, v_1), (v', v_1)\}$. The directed neighborhood diversity classes are $C_i = \{v_i\}$ for $i \in \{1, \dots, d-1\}$ and $C_d = \{v, v'\}$. The only path between v and v' is $v, v_1, \dots, v_{d-1}, v'$.

By this corollary, we can obtain an analogue of Theorem 6.3 for directed graphs as follows:

Theorem 6.11. *DSPP and DIPP are FPT when parameterized by the directed neighborhood diversity of the input graph.*

Proof. The proof is exactly the same as the one of Theorem 6.3, using the crucial fact from Lemma 6.9 which implies that there are at most $2^{\mathcal{O}(\mathsf{dnd}(G))}$ types of induced/shortest directed paths (that can be computed in $2^{\mathcal{O}(\mathsf{dnd}(G))}n$ time), and since each directed neighborhood diversity class is a set of twins, the choice of some vertex in the class is completely arbitrary. Thus, the same ILP as in the proof of Theorem 6.3 provides the desired FPT algorithm. □

Let us compare $\mathsf{dnd}(G)$ with $\mathsf{nd}(U(G))$ of the underlying undirected graph $U(G)$. By the very definitions, we observe:

Proposition 6.12. *For every directed graph G, $\mathsf{nd}(U(G)) \leq \mathsf{dnd}(G)$.*

Note that equality in Proposition 6.12 can be strict, consider for example any tournament G of order n. Since there is exactly one arc between every pair u, v of vertices of G, the third condition for $u \sim_{dnd} v$ is never satisfied, so $\mathsf{dnd}(G) = n$. However, $U(G)$ is a complete graph, so $\mathsf{nd}(U(G)) = 1$.

Similarly to the undirected case, the directed neighborhood diversity number of the underlying graph is upper-bounded by an exponential function of the vertex cover number of the underlying undirected graph. However, the details are a bit different, as shown next.

Proposition 6.13. *For every directed graph G, $\mathsf{dnd}(G) \leq 4^{\mathsf{vc}(U(G))} + \mathsf{vc}(U(G))$.*

Proof. Namely, consider all (not necessarily proper) 4-colorings of the vertex cover set. They should come with the following meaning (with respect to a vertex u in the independent set):

- 0: not touched by any edge incident to u,
- 1: in $N^+(u) \setminus N^-(u)$,
- 2: in $N^-(u) \setminus N^+(u)$,
- 3: in $N^+(u) \cap N^-(u)$.

This idea already shows the claim. Namely, at worst each vertex in a minimum vertex cover forms its own equivalence class, and two vertices u, u' in the independent set (the complement of the vertex cover) are equivalent if and only if they induce the same 4-coloring as defined above. As there are at most $4^{\mathsf{vc}(U(G))}$ many such 4-colorings, this upper-bounds the number of equivalence classes with independent set vertices only. □

This immediately provides the following result, which we will refine and make more explicit in the next subsection.

Corollary 6.14. *DSPP and DIPP are FPT when parameterized by the vertex cover number of the underlying undirected graph.*

We could also define the directed neighborhood diversity by only in-arcs or out-arcs. But different from the undirected case, we must specify if we consider the open or closed neighborhood for the diversity. If we would define the relation $\sim$ for two vertices $v, u \in V$ only by $N^-(v) \setminus \{u\} = N^-(u) \setminus \{v\}$ (completely in analogy to the undirected case), then the transitive triangle would again be a counter-example for the required transitivity. As for $\sim_{dnd}$, we could include the condition $(u, v) \in E$ if and only if $(v, u) \in E$. But even for this modification, there is a counter-example for transitivity of this relation $\sim$: $G = (\{u, v, w\}, E)$ with $E := \{(w, v), (u, v), (v, w)\}$. Here, $v \sim w$ and $v \sim u$ but $w \nsim u$.

So either we try to discuss further adaptations of the notion of neighborhood diversity towards directed graphs, or we can base a definition on the open or closed neighborhoods, e.g., $u \sim_{\text{open}^-} v \iff N^-(u) = N^-(v)$. But we did not find a way such that these

definitions help generalize our algorithms. For example, consider for the open neighborhood (the closed one works analogously) $G = (\{v, u, w, x, y\}, E)$ with $E := \{(y, v), (y, u), (v, w), (v, x)\}$. The three $\sim_{\text{open}^-}$-classes are $C_1 := \{y\}$, $C_2 := \{v, u\}$, $C_3 := \{w, x\}$. So our algorithm would advise to use a path C_1, C_2, C_3 and C_2, C_3. But this is not possible.

6.3. More on vertex cover parameterization

We now study the parameterization by the vertex cover number of the (underlying) graph, which provides better running times than via the more general (directed) neighborhood diversity parameterization of Corollary 6.14.

Proposition 6.15. *UPP, USPP and UIPP are FPT when parameterized by vertex cover number, as they admit single-exponential size kernels. The same holds for directed graphs for DPP, DSPP and DIPP, for the vertex cover number of the underlying graph.*

Proof. Assume that $\mathsf{vc}(G)$ is our problem parameter, given a graph $G = (V, E)$, and let $C \subseteq V$ be a minimum vertex cover (which can be computed in FPT time by [16,45]). Now we claim that if there are more than $2\mathsf{vc}(G)$ many vertices in the independent set $I = V \setminus C$ that have the same type, then at least one of them must form a single-vertex path in a minimum pp $\mathbb{P}$ of G. Namely, consider a vertex $x_0 \in I$ with $x_1, \ldots, x_{2\mathsf{vc}(G)} \in I$ being $2\mathsf{vc}(G)$ many other vertices of the same type, i.e., $N_G(x_0) = N_G(x_i)$ for $i = 1, \ldots, 2\mathsf{vc}(G)$. Any of these vertices x_i (with $i = 0, \ldots, 2\mathsf{vc}(G)$) that is not forming a single-vertex path in $\mathbb{P}$ demands at least one neighbor $y_i \in N_G(x_0)$ as a neighbor on its path. This neighbor y_i can only be 'shared' by one other vertex $x'_i \in I$ on this path, i.e., there might be at most one $i' \neq i$ such that $y_i = y_{i'}$. But as $|N_G(x_0)| \leq \mathsf{vc}(G)$ and as there are (at least) $2\mathsf{vc}(G) + 1$ many vertices of the same type as x_0, so by pigeon hole, at least one vertex must form a single-vertex path in $\mathbb{P}$.

Implementing this reduction rule leaves us with $|I| \leq 2\mathsf{vc}(G) \cdot 2^{\mathsf{vc}(G)}$, as the number of neighborhood equivalence classes within I is clearly bounded by $2^{\mathsf{vc}(G)}$. Hence, we can assume that the size of G is bounded by a function in $\mathsf{vc}(G)$ which shows that UPP, UIPP and USPP parameterized by vertex cover, have a kernel and are hence in FPT.

Note that the exact same argument holds for directed graphs as well, based on our reasoning in the preceding subsection. □

We can also obtain a direct FPT algorithm for UPP, leading to the following result. It is not that clear how this result can be adapted to the other path partition variants (UIPP and USPP) studied in this paper.

Proposition 6.16. *UPP is FPT when parameterized by vertex cover number, testified by an algorithm running in time* $\mathcal{O}^*\left(\mathsf{vc}(G)! \cdot 2^{\mathsf{vc}(G)}\right)$ *on input* G.

Proof. The proposed algorithm runs as follows on the input graph $G = (V, E)$:

(0) Compute a minimum vertex cover C of the input graph; this can be done in time $\mathcal{O}(c^{\mathsf{vc}(G)})$ for some $c < 1.3$, see [16,45]. We call the complement of C as I, being an independent set.

We also keep track of a 'record partition' $\mathbb{P}$ that we initialize with the path partition where each path is consisting of one vertex.

(1) Iterate through all permutations of the vertices in $C = \{v_1, \ldots, v_r\}$, where $r = \mathsf{vc}(G)$. The idea is that each such permutation describes a sequence of vertices in the order in which each of the paths that are considered in a partition traverse the C-vertices in this order.

(2) For each such permutation π, say, $v_{\pi(1)}, \ldots, v_{\pi(r)}$, walk through all $(r-1)$-dimensional bit-vectors $\bar{b} = b_1 b_2 \cdots b_{r-1}$. These bit-vectors describe where the sequence $v_{\pi(1)}, \ldots, v_{\pi(r)}$ is connected (or not) on a path. More precisely, $b_j = 1$ if and only if $v_{\pi(j)}$ and $v_{\pi(j+1)}$ are on the same path in the path partition $\mathbb{P}(\pi, \bar{b})$ that we are constructing. Therefore, a bit-vector with ℓ zeros will describe a collection of paths with $\ell + 1$ paths that contain vertices of C. Also, if $P \in \mathbb{P}(\pi, \bar{b})$, then there exist $i \leq j$ such that, if we view P as a set of vertices, $P \cap C = \{v_{\pi(i)}, \ldots, v_{\pi(j)}\}$ and for all indices $\iota \in [i, j-1]$, $b_\iota = 1$, while $b_{i-1} = 0$ or $i = 1$ and $b_j = 0$ or $j - 1 = r$.

Remark 6.17. Under the assumption that $I = V \setminus C$ might be much larger than C, as suggested by the exponential-size kernel obtained in Proposition 6.15, this does not relate ℓ to the solution size k, as then k may be in the order of $|I|$ rather than just in the order of $\ell \leq r = |C|$. Yet, we can say that, given a bit-vector $\bar{b}$ with ℓ zeros and assuming that I is sufficiently large, we will obtain at least $(\ell + 1) + (|I| - 2(\ell + 1)) = |I| - (\ell + 1)$ many paths, a number that should be smaller than k, as otherwise we can abort. Namely, by our construction, we have $\ell + 1$ paths containing vertices from C. In order to get as few paths as possible in our path partition, as many vertices from I as possible should be integrated into these paths. In each of these paths, we can integrate at most as many vertices as there are vertices belonging to C, plus one, because vertices from I can occur only as 'intermediate vertices' between two vertices from C, or at the very ends of the paths, because I is an independent set in G by assumption. Hence, at most $2r + (\ell + 1)$ many vertices can be integrated into these $\ell + 1$ many paths. Of these vertices, at most $r + (\ell + 1)$ belong to I, but 'the rest' of I must be covered by single-vertex-paths. However, we could possibly also get more paths in a solution that is consistent with $\bar{b}$, namely, if no vertices from I are integrated in the paths that contain vertices from C. In this extreme case, we are looking at a solution with $|I| + (\ell + 1)$ many paths. In order to integrate as many vertices from I into paths that contain vertices from C, we will suggest to consider a maximum matching in an auxiliary graph, as described in the following.

(3) For all π and $\bar{b}$, create an auxiliary edge-weighted bipartite graph $G' = G'(\pi, \bar{b})$ with edge set E' and vertex set V' containing V, such that C is part of one set C' of the bipartition. Similarly, let $I' = V' \setminus C'$ contain I. Moreover, we will define a weight function $w : E' \to \{1, 2\}$. More precisely, to obtain C', add to C (always) the vertex $v'_{\pi(1)}$ and further vertices $v'_{\pi(i+1)}$ if $b_i = 0$, and possibly add vertices u_i into I if $b_i = 1$ to build I'. For a worked-out example explaining our construction, we refer to Example 6.20.

First assume that $b_i = 1$, i.e., $v_{\pi(i)}$ and $v_{\pi(i+1)}$ should lie on a path in $\mathbb{P}(\pi, \bar{b})$. Connect $v_{\pi(i)}$ and $x \in I$ by an edge of weight 2 if $v_{\pi(i)}x \in E$ and $v_{\pi(i+1)}x \in E$. Moreover, if there is no edge incident to $v_{\pi(i)}$ so far but $v_{\pi(i)}v_{\pi(i+1)} \in E$, then introduce a new vertex u_i (as part of I') and the edge $v_{\pi(i)}u_i$ of weight 1.

Secondly, assume that $b_i = 0$, i.e., $v_{\pi(i)}$ and $v_{\pi(i+1)}$ should not lie on a path in $\mathbb{P}(\pi, \bar{b})$. Still, the path that ends at $v_{\pi(i)}$ (concerning the C-vertices) could contain one more vertex (in I). Therefore, we also connect $v_{\pi(i)}$ and $x \in I$ by an edge of weight 1 if $v_{\pi(i)}x \in E$. Similarly, connect $v_{\pi(r)}$ and $x \in I$ by an edge of weight 1 if $v_{\pi(r)}x \in E$. Also the path that starts at $v_{\pi(i+1)}$ (concerning the C-vertices) could contain one more vertex (in I). To cover this case, we also connect $v_{\pi(i+1)'}$ and $x \in I$ by an edge of weight 1 if $v_{\pi(i+1)}x \in E$. Similarly, we connect $v_{\pi(1)'}$ and $x \in I$ by an edge of weight 1 if $v_{\pi(1)}x \in E$.

This concludes the description of the auxiliary graph $G' = (V', E')$ and the weight function w.

Now, compute a maximum weighted matching M in G'. If the vertices $v_{\pi(i)}, v_{\pi(i+1)}, \ldots, v_{\pi(j)}$ are matched, then this clearly corresponds to a path within G that traverses $v_{\pi(i)}, v_{\pi(i+1)}, \ldots v_{\pi(j)}$ in that order, possibly using vertices from I on its way, including possibly starting or ending in I. Let us make this more precise.

First, we test if there is any edge $v'_{\pi(i+1)}x$ in the matching, while $v_{\pi(i)}x$ was also an edge in G'. This would mean that we found a situation where we guessed (by the bit-vector $\bar{b}$, as $b_i = 0$) that one path ends at $v_{\pi(i)}$ and the next path starts at $v_{\pi(i+1)}$. Our matching proves that we could connect both paths, i.e., the path partition $\mathbb{P}(\pi, \bar{b})$ would not be minimal. Also, there will be another setting of the bit-vector $\bar{b}$ that covers this (better) case. Therefore, we need not consider this case any longer. Similarly, we can skip the case when $v'_{\pi(1)}x$ in the matching, while $v_{\pi(r)}x$ was also an edge in G'.

Now, we describe $\mathbb{P}(\pi, \bar{b})$ more precisely. Our first path P_1 should contain $v_{\pi(1)}$. If $v_{\pi(1)'}$ is matched with some $x_0 \in I$, then P_1 starts with x_0, with the second vertex then being $x_1 = v_{\pi(1)}$, while otherwise it starts with x_1. If $b_1 = 0$, then it might be the case that x_1 is matched with $x_2 \in I$, where P_1 will then end, or x_1 is not matched, which means that P_1 already ends with x_1. If $b_1 = 1$, then we expect that x_1 is matched with some vertex (otherwise, the guessed bit-vector would be wrong and we can stop any further computation of this case here). If x_1 is matched with some $x_2 \in I$, this means that (within G) we find the edges x_1x_2 and x_2x_3, with $x_3 = v_{\pi(2)}$. Otherwise, x_1 is matched with u_1. By construction, this means that there was no $x \in I$ with $x \in N_G(x_1) \cap N_G(v_{\pi(2)})$, but $x_1 \in N_G(v_{\pi(2)})$. Hence, we can take $x_2 = v_{\pi(2)}$ as the next vertex on P_1. Continuing this way, we will construct the first path P_1, whose end is determined by the first j with $b_j = 0$. Then, the second path P_2 will start with $v_{\pi(j)}$ if $v'_{\pi(j)}$ is unmatched and with some $y \in V \setminus C$ if y is matched with $v'_{\pi(j)}$ that is constructed in the very same fashion.

Notice that all vertices of I outside the matching belong to paths of length zero. As the matching was maximum, we see the minimum number of paths (respecting the choice of $\bar{b}$) constructed in this way. More precisely, we claim the following property.

Claim 6.18. *Let M be a maximum weight matching of $G'(\pi, \bar{b}) = (V', E')$ with weight function w as defined above. Then, the path partition $\mathbb{P}(\pi, \bar{b})$ contains $|V| - w(M)$ many paths.*

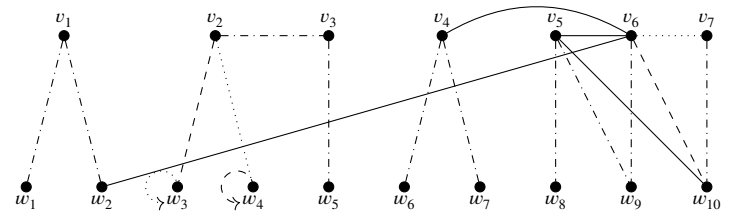
Proof. It is best to think of this claim from the perspective of how many paths can be 'saved' within $\mathbb{P}(\pi, \bar{b})$ in comparison to the solution that consists of single-vertex paths only. If a vertex in $C \cup I$ is not matched, then it means that it forms a single-vertex path. If we find an edge $e = v_{\pi(i)}x \in M$ with $x \in I$, then this means that we see the sub-path $v_{\pi(i)}xv_{\pi(i+1)}$ in $\mathbb{P}(\pi, \bar{b})$, somehow 'merging' three potential single-vertex paths and therefore reducing the number of paths by 2, which is the weight of e. If we find an edge $e = v_{\pi(i)}x \in M$ where $b_i = 0$, i.e., $w(e) = 1$, then the path does not continue to $v_{\pi(i+1)}$ but will end in x. Nonetheless, x does not form a single-vertex path, so again we reduce the number of paths by the weight 1 of e. A similar observation can be made concerning edges between some $v'_{\pi(i)}$ and some $x \in I$, as they treat 'the other end' of the paths. Finally, one can consider the edges $e = v_{\pi(i)}u_i \in M$. Again, $w(e) = 1$. They reflect the possibility to connect two paths that would otherwise end in $v_{\pi(i)}$ and start in $v_{\pi(i+1)}$, respectively. Hence, also in this case we reduce the number of paths by the weight 1 of e. ◇

If the path partition $\mathbb{P}(\pi, \bar{b})$ contains less paths than the current record partition $\mathbb{P}$, we will update $\mathbb{P}$ accordingly. After enumerating all permutations and bit vectors, $\mathbb{P}$ should satisfy the following claim.

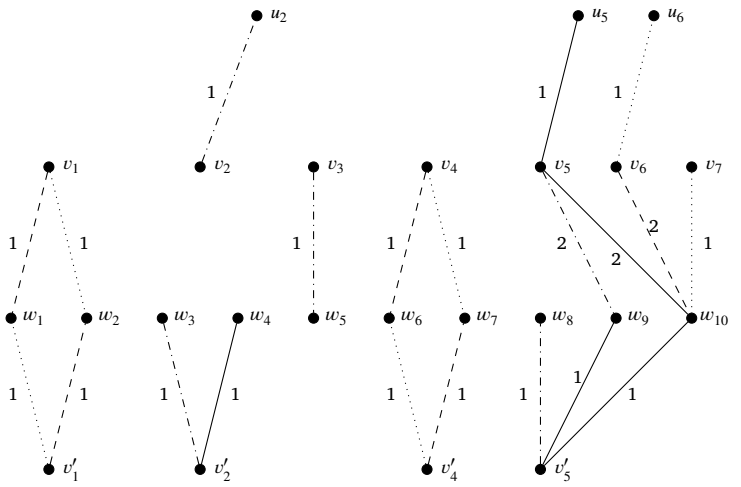
Claim 6.19. *The given graph G contains a path partition with at most k many paths if and only if our algorithm outputs a record partition $\mathbb{P}$ with at most k many paths.*

Proof. It should be clear by what we argued above that our algorithm only outputs valid path partitions. Hence, if it outputs a record partition $\mathbb{P}$ with at most k many paths, then our graph contains such a pp.

Conversely, let $\mathbb{P}$ be a pp with the minimum number of paths. Let k be the number of these paths. If $P_1, \ldots, P_k$ are the paths in $\mathbb{P}$ ordered by decreasing length, then there is some smallest number $k' \leq k$ such that the paths $P_{k'+1}, \ldots, P_k$ contain only vertices from I. Then, by reading the vertices from $C = \{v_1, \ldots, v_r\}$ on the paths $P_1, \ldots, P_{k'}$ 'from left to right', we can define a permutation π on $\{1, \ldots, r\}$. Moreover, we can define a bit-vector $\bar{b}$ that describes how many vertices of C belong to $P_1, \ldots, P_{k'}$. This bit-vector has $k' - 1$ many zeros. One can verify that this solution corresponds to a weighted maximum matching in the auxiliary graph $G'(\pi, \bar{b})$ of weight $|V| - k$. ◇



(a) The original graph G



(b) The resulting bipartite graph G'

Fig. 7. Drawings illustrating Example 6.20.

Notice that Step (0) is executed only once, so that for the running time, it is decisive that we cycle through all permutations and through all settings of the bit-vectors. □

We are now illustrating the construction of the previous proof by an example.

Example 6.20. Consider $G = (V, E)$, with V being decomposed as $C \cup I$, where $C = \{v_1, v_2, \ldots, v_7\}$ and $I = \{w_1, w_2, \ldots, w_{10}\}$. The set C is a vertex cover of G, with E containing the edges $\{v_2v_3, v_4v_6, v_6v_6, v_6v_7\}$ plus several edges connecting C and I, as shown in Fig. 7.

Let us consider the identity as permutation and consider the bit-vector $\bar{b} = (0, 1, 0, 0, 1, 1)$. This lets us consider the graph G' as depicted in Fig. 7. Notice that we use four layers for drawing G'. On the topmost layer, we show the three vertices u_2, u_5, u_6 that we have to introduce thanks to $\bar{b}$. On the second layer, we draw the vertices of C given in the ordering of our permutation, the identity. On the third layer, we draw the vertices from I. Finally, our algorithm enforces us to introduce one more primed vertex whenever a new path is started with respect to C. As $\bar{b}$ contains three zeros, this means that we have to introduce four vertices, which are v'_1, v'_2, v'_4, v'_5. All edges in this bipartite graph G' that are incident to vertices on the topmost or lowermost layer have weight 1. Only edges connecting the second and the third layer may have either weight 1 or 2. In our example, one can find two distinctively different matchings of maximum weight:

- $M_1 = \{v_1w_1, v'_1w_2, v_2u_2, v'_2w_3, v_3w_5, v_4w_6, v'_4w_7, v'_5w_8, v_5w_9, v_6w_{10}\}$, and
- $M_2 = \{v'_1w_1, v_1w_2, v_2u_2, v'_2w_3, v_3w_5, v'_4w_6, v_4w_7, v'_5w_8, v_5w_9, v_6u_6, v_7w_{10}\}$.

We find $w(M_1) = w(M_2) = 12$ and hence, both matchings should correspond to a collection of five paths in the original graph G, as G has 17 vertices. More precisely, M_1 corresponds to

$$P_1 = \{w_1v_1w_2, w_3v_2v_3w_5, w_4, w_6v_4w_7, w_8v_5w_9v_6w_{10}v_7\},$$

while M_2 shows $P_2 = \{w_1v_1w_2, w_3, w_4v_2v_3w_5, w_6v_4w_7, w_8v_5w_9v_6v_7w_{10}\}$. In Fig. 7, we mark M_1 with dashed lines and M_2 with dotted lines; if an edge appears in both matchings, the line is dashed-dotted. We follow a similar convention for marking the corresponding paths P_1 and P_2. We mark single-vertex paths by small self-loops, see w_4 in P_1 and w_3 in P_2.

We could also consider a different bit-vector, say, $\bar{b}' = (0,0,0,0,1,1)$. Although we would now have one more path involving C-vertices, their overall number would not change. As the reader can verify, a possible solution obtained by our matching approach is now

$$P_1' = \{w_1v_1w_2, w_3v_2w_4, w_5v_3, w_6v_4w_7, w_8v_5w_9v_6w_{10}v_7\}.$$

Again, there would be a variation P_2' similar as before.

Observe that the edge v_4v_6 does not play a big role in our construction in this example. We can take this to an extreme by considering the permutation $\pi = (7,1,2,6,3,5,4)$. Then, there are no two vertices $v_{\pi(i)}$ and $v_{\pi(i+1)}$ that are connected by an edge. Even worse, there is no vertex from I adjacent both to $v_{\pi(i)}$ and $v_{\pi(i+1)}$. Therefore, the only reasonable bit-vector is the all-zero-vector $\vec{0}$. This means we have seven paths containing C-vertices in any path partition of $G'(\pi,\vec{0})$. But in fact, our matching approach also yields a matching with 10 edges in $G'(\pi,\vec{0})$, with each edge having weight 1. To realize this, we could take a path partition that contains four paths of length one: v_7w_{10}, v_6w_9, v_3w_5, and v_5w_8, plus three paths of length two: $w_1v_1w_2$, $w_3v_2w_4$, and $w_6v_4w_7$.

But the first choice obviously gave a better result for our aim of minimizing the number of paths in a path partition of G.

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*; see [42]. Moreover, all our problems can be algorithmically solved for each connected component separately, so that we can assume, without loss of generality, 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 combinatorial claims are crucial.

Lemma 7.1. *Let G be an undirected 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.*

Proof. Let M be a matching of G such that M covers $2k$ vertices. We construct a solution $\mathbb{P}$ of PP, IPP and SPP as follows. For every edge $(u,v) \in M$, include the path $\{u,v\}$ into $\mathbb{P}$. Put the rest of the vertices not yet covered by M as single-vertex paths into the solution $\mathbb{P}$. We see that the path collection $\mathbb{P}$ covers every vertex exactly once in G and has cardinality $n-k$. □

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.*

More interestingly, Lemma 7.1 is also valid for directed graphs in the following form as we can observe the directions of the matching edges as prescribed by the given directed graph.

Lemma 7.4. *Let G be a directed graph of order n. If the underlying undirected graph $U(G)$ has a matching that covers $2k$ vertices, then $(G, n-k)$ is a Yes-instance of DPP, DIPP and DSPP.*

Our combinatorial thoughts, along with Corollary 6.5 and Proposition 6.15, allow us to show the following algorithmic result, the main result of this section.

Theorem 7.5. *UPP, UIPP and USPP, as well as DPP, DIPP and DSPP, can be solved in FPT time with respect to the dual parameterization.*

Proof. We first handle the undirected graph case. Let $G = (V,E)$ and k together be an instance of UPP, UIPP, or USPP, where $|V| = n$. As said above, we can assume that G is connected. Next, compute (in polynomial time) a maximum matching M in G. If $|M| > k$, then Lemma 7.1 justifies why $(G, n-k)$ is a Yes-instance. Hence, we can assume that any matching of G has size at most k. This implies (as matchings provide factor-2 approximations for Minimum Vertex Cover) that the vertex cover number $\mathrm{vc}(G)$ is at most $2k$. This allows us to first compute $\mathrm{vc}(G)$ in FPT time; see [16]. Now, Corollary 6.5 and Proposition 6.15 show the claim in the case of UIPP and USPP, as well as for UPP.

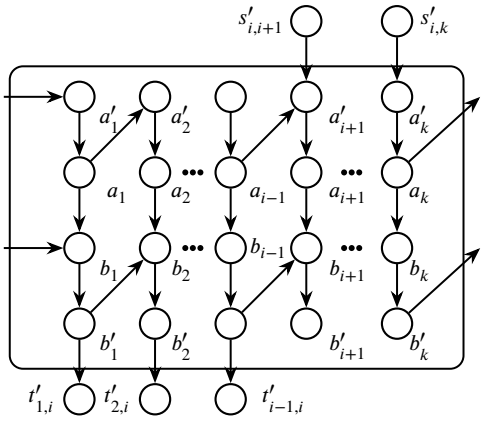


Fig. 8. New gadget $G^{i,u}$, $u \in [n]$, $i \in [k]$.

For the case of directed graphs, we can first argue concerning the underlying undirected graph as in Lemma 7.4. Then, we also make use of the part of Proposition 6.15 for directed graphs. □

8. Covering and edge-disjoint variants

As we mentioned in the introduction, the variants PC, SPC and IPC of the problems, where the paths cover the vertices but do not need to form a partition, are also studied. Many of our results do apply to these problems as well.

Alternatively, in some applications, paths may be required to be edge-disjoint but not necessarily vertex-disjoint, see for example [75], where the problem Edge-Disjoint Paths is studied. Thus, one may define the problems Edge-Disjoint Path Cover (ED-PC), Edge-Disjoint Shortest Path Cover (ED-SPC) and Edge-Disjoint Induced Path Cover (ED-IPC) (as before we include the prefixes U, D, or DAG in the problem name abbreviations if we want to emphasize that the inputs are undirected, directed, or directed acyclic). These can be thought of as intermediate problems between the partition and the covering variants, indeed a (unrestricted, induced or shortest) path partition is an (unrestricted, induced or shortest) edge-disjoint path cover, which is also a (unrestricted, induced or shortest) path cover.

Some of our results apply to these three problems as well with slight modifications of the proofs. Hence, we only exhibit these modifications in the following.

8.1. NP-completeness results for DAGs and bipartite undirected graphs

Theorem 8.1. *DAGSPC, DAGIPC, ED-DAGSPC and ED-DAGIPC are NP-hard even when the inputs are restricted to planar bipartite DAGs of maximum degree 4.*

Proof. We follow the same construction as in Theorem 3.1. Observe that, by the same argument as in the proof of Claim 3.3, each shortest or induced path in any solution of (ED-)DAGSPC/(ED-)DAGIPC must contain exactly three vertices (and thus forms a vertex-partition of the graph), and each gadget $H(v_i)$ is partitioned into P_3-paths in one of the two ways as shown in Fig. 2. The rest of the argumentation is similar to Theorem 3.1. □

Theorem 8.2. *USPC and ED-USPC are NP-hard, even for bipartite 5-degenerate graphs of diameter 4.*

Proof. We follow the same construction as Theorem 3.6 (also see Fig. 3). We can observe that Claim 3.8, Claim 3.9 and Claim 3.7 are true due to the construction. The rest of the proof is similar to Theorem 3.6. □

8.2. W-hardness for DAGs

Our W-hardness result is only adaptable (as far as we can see) for the edge-disjoint path cover variants: for the unrestricted path cover variants, crucial elements of the proof do not work, as the same edges can be used by multiple solution paths. Also, recall that DAGPC is solvable in polynomial time [33].

The only problem with the construction of Theorem 4.1 is that now paths can start from $s_{i,j}$, with $i, j \in [k]$, and go through to a_l, $i \leq l \leq k$, even if a_l belongs to a different path. To prevent this, we modify $G^{i,u}$, where $0 < u \leq n$ and $i \in [k]$, in the construction as shown in Fig. 8. The rest of the proof is similar to Theorem 4.1. This construction is an adaptation of [75], where it was proved that ED-DP is W[1]-hard when parameterized by solution size.

Theorem 8.3. *ED-DAGSPC and ED-DAGIPC (parameterized by solution size) are W[1]-hard.*

8.3. XP algorithms

For our next result, we need the following theorem, indeed the techniques used in Section 5 are not applicable since they are based on using (vertex-) disjoint path problems.

Theorem 8.4 *([26]). Let G be an undirected graph whose vertex set can be covered by at most k (not necessarily disjoint) shortest paths. Then the pathwidth of G is in $O(k \cdot 3^k)$.*

The following corollary was already observed in [26] for the problems USPC and USPP. The same argument shows the following.

Corollary 8.5. *ED-USPC is in XP when parameterized by solution size.*

Proof. Here we cannot use the technique of Theorem 5.1 based on DP and its variants. Corollary 8.5, follows from the same proof used in [26] for USPC: first bound the pathwidth of the graph using Theorem 8.4. Then, as for Theorem 5.1, guess the set X of endpoints of the solution paths (enumerating all the solutions is done in XP time). That way, one can compute the sum s of the pairwise distances between the endpoints in X. Then, one can use the counting version of Courcelle's theorem by expressing the problem in Counting Monadic Second Order Logic (CMSOL) and solve it in FPT time: one expects a partition whose total size is equal to s. □

8.4. Neighborhood diversity and vertex cover parameterizations

Theorem 8.6. *USPC, UIPC, ED-USPC and ED-UIPC (as well as DSPC, DIPC, ED-DSPC and ED-DIPC) are in FPT when parameterized by (directed) neighborhood diversity.*

Proof. This can be obtained with the same proof as the one of Theorem 6.3 by modifying the ILP as follows. Each path $P \in \mathcal{P}$ is characterized by a vector $(t_{P,1,1}, \ldots, t_{P,k,k})$ with $t_{P,i,j} = |E_{i,j} \cap P|$, where $i, j \leq k$. Here $E_{i,j}$ refers to the edges between the i^{th} and j^{th} equivalence class.

For the covering versions, we use:

$$\text{minimize} \sum_{P \in \mathcal{P}} z_P$$
$$\text{subject to} \sum_{P \in \mathcal{P}} z_P \cdot P^j \geq |C_j| \text{ for all } j \in [d],$$

while for the edge-disjoint variants, we write:

$$\text{minimize} \sum_{P \in \mathcal{P}} z_P$$
$$\text{subject to} \sum_{P \in \mathcal{P}} z_P \cdot P^j \geq |C_j| \text{ for all } j \in [d]$$
$$\text{and subject to} \sum_{P \in \mathcal{P}} z_P \cdot t_{P,i,j} \leq |E_{i,j}| \text{ for all } i, j \in [d].$$

For the directed cases, we refer to the arguments leading to Theorem 6.11. □

Corollary 8.7. *USPC, UIPC, ED-USPC and ED-UIPC (as well as DSPC, DIPC, ED-DSPC and ED-DIPC) are in FPT when parameterized by the vertex cover number of the given (or of the underlying undirected) graph.*

8.5. Dual parameters

Theorem 8.8. *UPC, ED-UPC, ED-USPC and ED-UIPC (as well as the directed counterparts ED-DSPC, ED-DIPC, ED-DSPC and ED-DIPC) can be solved in FPT time with respect to the dual parameterization.*

Proof. These results are obtained using the technique of Theorem 7.5 in view of Corollary 8.7, using Proposition 6.15 for (ED-)UPC and Theorem 8.6 for (ED-)USPC and (ED-)UIPC. □

9. Conclusions

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 [26], and our W[1]-hardness result for DAGs can be seen as a first step towards an answer.

Another interesting question is whether SPP is XP on directed graphs? This is the case for undirected graphs, for DAGs and for directed graphs that are planar or have bounded directed treewidth, but we do not know about the general case. In the conference version of this paper, we had asked whether IPP is in XP for DAGs when parameterized by solution size, or if it is paraNP-hard like in the general (and undirected) case. During the revision of this journal paper, the latter possibility has been confirmed in [53]. We have added this reference also in Table 1 because it highlights that in terms of complexity, there are indeed many parallels between undirected graphs and DAGs concerning SPP/IPP problems.

We do not know whether IPP is NP-hard on undirected bipartite graphs. Other classes, like planar graphs, could also be interesting to study in this regard.

We have seen that IPP and SPP admit FPT algorithms when parameterized by (directed) neighborhood diversity. Can we obtain such algorithms for other (e.g., more general) parameters? This holds on undirected graphs (UPP) for the more general parameter modular-width [34]; we do not know if this stays true for directed graphs.

Regarding the parameterized algorithms that we obtained for vertex cover and neighborhood diversity, can they be improved? For PP parameterized by vertex cover, we managed to reduce the dependency to $\mathcal{O}^*(\mathsf{vc}! \cdot 2^{\mathsf{vc}})$, which is $\mathcal{O}^*\left(2^{\mathsf{vc}\log \mathsf{vc}}\right)$ (Proposition 6.16). It is not clear if this can be improved to, say, $\mathcal{O}^*(2^{\mathsf{vc}})$. Also, it would be interesting to obtain a similar running time (or better) for SPP and IPP as well. Furthermore, we do not know whether polynomial kernelizations exist for these problems.

Similarly, 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.

As we have indicated, some of our results also apply to the covering variants of the three problems (where the paths need not be disjoint). On the other hand, approximation algorithms have been designed for SPC, the covering version of SPP, in [78] (for general undirected graphs) and [13] (for chordal graphs and related undirected graph classes). Are there such algorithms for SPP (and IPP) as well?

Another line of research is *reconfiguration*. In particular, shortest path reconfiguration in undirected graphs has been extensively studied, see [35] for a recent piece of work that also gives a nice survey, but directed graphs have attracted less attention. Also, the reconfiguration of path partitions as studied in this paper has not been touched so far, although this has natural applications in reconfiguring supply networks. A possible definition of a single reconfiguration step could be to select two paths P, Q from the current path partition $\mathbb{P}$ and build two other paths P', Q' partitioning the graph induced by the vertex sets of P, Q, so that one obtains another path partition $\mathbb{P}'$ by replacing P, Q in $\mathbb{P}$ by P', Q'.

We also note that some of the techniques used in this paper for the IPP problem likely can be applied to the Hole Partition problem (where we want to partition a graph into induced cycles). The packing version of this problem was studied in [62].

SPP could also be studied for edge-weighted graphs, as studied for example in [8].

CRediT authorship contribution statement

Henning Fernau: Conceptualization, Investigation, Methodology, Project administration, Supervision, Validation, Writing – original draft, Writing – review & editing. **Florent Foucaud:** Conceptualization, Investigation, Methodology, Project administration, Supervision, Validation, Writing – original draft, Writing – review & editing. **Kevin Mann:** Investigation, Writing – original draft, Writing – review & editing. **Utkarsh Padariya:** Investigation, Writing – original draft, Writing – review & editing. **Rajath Rao K.N.:** Investigation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This paper is an extended version of a paper presented at CIAC 2023 [29]. The research of the second author was financed by the French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25), the International Research Center “Innovation Transportation and Production Systems” of the I-SITE CAP 20-25, and by the ANR project GRALMECO (ANR-21-CE48-0004). The research of the last author was funded by a DAAD-WISE Scholarship 2022.

Data availability

No data was used for the research described in the article.

References

[1] M. Aigner, M. Fromme, A game of cops and robbers, Discrete Appl. Math. 8 (1) (1984) 1–12.

[2] G. Andreatta, F. Mason, Path covering problems and testing of printed circuits, Discrete Appl. Math. 62 (1–3) (1995) 5–13.
[3] J. Araújo, J. Bensmail, V. Campos, F. Havet, A.K. Maia de Oliviera, N. Nisse, A. Silva, On finding the best and worst orientations for the metric dimension, Algorithmica 85 (10) (2023) 2962–3002.
[4] J. Araújo, V.A. Campos, A.K. Maia, I. Sau, A. Silva, On the complexity of finding internally vertex-disjoint long directed paths, Algorithmica 82 (6) (2020) 1616–1639.
[5] Jørgen Bang-Jensen, Tilde My Christiansen, Alessandro Maddaloni, Disjoint paths in decomposable digraphs, J. Graph Theory 85 (2) (2017) 545–567.
[6] R. Belmonte, T. Hanaka, M. Kanzaki, M. Kiyomi, Y. Kobayashi, Y. Kobayashi, M. Lampis, H. Ono, Y. Otachi, Parameterized complexity of (A,ℓ)-path packing, Algorithmica 84 (4) (2022) 871–895.
[7] M. Bentert, A. Nichterlein, M. Renken, P. Zschoche, Using a geometric lens to find k-disjoint shortest paths, SIAM J. Discrete Math. 37 (3) (2023) 1674–1703.
[8] K. Bérczi, Y. Kobayashi, The directed disjoint shortest paths problem, in: K. Pruhs, C. Sohler (Eds.), 25th Annual European Symposium on Algorithms, ESA, in: LIPIcs, vol. 87, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 13:1–13:13.
[9] C. Berge, Path partitions in directed graphs, in: North-Holland Mathematics Studies, vol. 75, Elsevier, 1983, pp. 59–63.
[10] F.T. Boesch, S. Chen, J.A.M. McHugh, On covering the points of a graph with point disjoint paths, in: Graphs and Combinatorics, Springer, 1974, pp. 201–212.
[11] F.T. Boesch, J.F. Gimpel, Covering points of a digraph with point-disjoint paths and its application to code optimization, J. ACM 24 (2) (1977) 192–198.
[12] M. Cáceres, M. Cairo, B. Mumey, R. Rizzi, A.I. Tomescu, Sparsifying, shrinking and splicing for minimum path cover in parameterized linear time, in: ACM-SIAM Symposium on Discrete Algorithms, SODA, SIAM, 2022, pp. 359–376.
[13] D. Chakraborty, A. Dailly, S. Das, F. Foucaud, H. Gahlawat, S.K. Ghosh, Complexity and algorithms for isometric path cover on chordal graphs and beyond, in: Proceedings of the 33rd International Symposium on Algorithms and Computation, ISAAC, in: LIPIcs, vol. 248, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 12:1–12:17.
[14] D. Chakraborty, H. Müller, S. Ordyniak, F. Panolan, M. Rychlicki, Covering and partitioning of split, chain and cographs with isometric paths, in: R. Královic, A. Kucera (Eds.), 49th International Symposium on Mathematical Foundations of Computer Science, MFCS, in: LIPIcs, vol. 306, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 39:1–39:14.
[15] G.J. Chang, D. Kuo, The L(2, 1)-labeling problem on graphs, SIAM J. Discrete Math. 9 (2) (1996) 309–316.
[16] J. Chen, I.A. Kanj, G. Xia, Improved upper bounds for vertex cover, Theor. Comput. Sci. 411 (40–42) (2010) 3736–3756.
[17] Y. Chen, Z.-Z. Chen, C. Kennedy, G. Lin, Y. Xu, A. Zhang, Approximating the directed path partition problem, Inf. Comput. 297 (2024) 105150.
[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, Third Edition, 3rd edition, The MIT Press, 2009.
[19] D.G. Corneil, B. Dalton, M. Habib, LDFS-based certifying algorithm for the minimum path cover problem on cocomparability graphs, SIAM J. Comput. 42 (3) (2013) 792–807.
[20] M. Cygan, F. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015.
[21] M. Cygan, D. Marx, M. Pilipczuk, M. Pilipczuk, The planar directed k-vertex-disjoint paths problem is fixed-parameter tractable, in: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS, IEEE Computer Society, 2013, pp. 197–206.
[22] P. Damaschke, J.S. Deogun, D. Kratsch, G. Steiner, Finding Hamiltonian paths in cocomparability graphs using the bump number algorithm, Order 8 (4) (1992) 383–391.
[23] R. Diestel, Graph Theory, 5th edition, Graduate Texts in Mathematics, vol. 173, Springer, 2017.
[24] R.P. Dilworth, A decomposition theorem for partially ordered sets, in: Classic Papers in Combinatorics, Springer, 2009, pp. 139–144.
[25] V. Dujmović, G. Joret, P. Micek, P. Morin, T. Ueckerdt, D.R. Wood, Planar graphs have bounded queue-number, J. ACM 67 (4) (2020) 1–38.
[26] M. Dumas, F. Foucaud, A. Perez, I. Todinca, On graphs coverable by k shortest paths, SIAM J. Discrete Math. 38 (2) (2024) 1840–1862.
[27] M.E. Dyer, A.M. Frieze, Planar 3DM is NP-complete, J. Algorithms 7 (2) (1986) 174–184.
[28] H. Eto, S. Kawaharada, G. Lin, E. Miyano, T. Ozdemir, Directed path partition problem on directed acyclic graphs, in: A.A. Rescigno, U. Vaccaro (Eds.), Combinatorial Algorithms - 35th International Workshop, IWOCA, in: LNCS, vol. 14764, Springer, 2024, pp. 314–326.
[29] H. Fernau, F. Foucaud, K. Mann, U. Padariya, K.N. Rajath Rao, Parameterizing path partitions, in: M. Mavronicolas (Ed.), Algorithms and Complexity - 13th International Conference, CIAC, in: LNCS, vol. 13898, Springer, 2023, pp. 187–201.
[30] D.C. Fisher, S.L. Fitzpatrick, The isometric number of a graph, J. Comb. Math. Comb. Comput. 38 (1) (2001) 97–110.
[31] S. Fortune, J.E. Hopcroft, J. Wyllie, The directed subgraph homeomorphism problem, Theor. Comput. Sci. 10 (1980) 111–121.
[32] D.S. Franzblau, A. Raychaudhuri, Optimal Hamiltonian completions and path covers for trees, and a reduction to maximum flow, ANZIAM J. 44 (2) (2002) 193–204.
[33] D.R. Fulkerson, Note on Dilworth's decomposition theorem for partially ordered sets, Proc. Am. Math. Soc. 7 (4) (1956) 701–702.
[34] J. Gajarský, M. Lampis, S. Ordyniak, Parameterized algorithms for modular-width, in: G.Z. Gutin, S. Szeider (Eds.), Parameterized and Exact Computation - 8th International Symposium, IPEC, in: LNCS, vol. 8246, Springer, 2013, pp. 163–176.
[35] K. Gajjar, A.V. Jha, M. Kumar, A. Lahiri, Reconfiguring shortest paths in graphs, in: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, the Twelfth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022, AAAI Press, 2022, pp. 9758–9766.
[36] T. Gallai, A.N. Milgram, Verallgemeinerung eines graphentheoretischen Satzes von Rédei, Acta Sci. Math. 21 (3–4) (1960) 181–186.
[37] M.R. Garey, D.S. Johnson, Computers and Intractability, Freeman, 1979.
[38] M.R. Garey, D.S. Johnson, R.E. Tarjan, The planar Hamiltonian circuit problem is NP-complete, SIAM J. Comput. 5 (4) (1976) 704–714.
[39] J.P. Georges, D.W. Mauro, M.A. Whittlesey, Relating path coverings to vertex labellings with a condition at distance two, Discrete Math. 135 (1–3) (1994) 103–111.
[40] P.A. Golovach, D. Paulusma, E.J. van Leeuwen, Induced disjoint paths in AT-free graphs, J. Comput. Syst. Sci. 124 (2022) 170–191.
[41] S. Goodman, S. Hedetniemi, On the Hamiltonian completion problem, in: Graphs and Combinatorics, Springer, 1974, pp. 262–272.
[42] J. Guo, F. Hüffner, R. Niedermeier, A structural view on parameterizing problems: distance from triviality, in: R. Downey, M. Fellows, F. Dehne (Eds.), International Workshop on Parameterized and Exact Computation IWPEC 2004, in: LNCS, vol. 3162, Springer, 2004, pp. 162–173.
[43] L. Guo, Y. Deng, K. Liao, Q. He, T. Sellis, Z. Hu, A fast algorithm for optimally finding partially disjoint shortest paths, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI, ijcai.org, 2018, pp. 1456–1462.
[44] M. Habib, C. Paul, L. Viennot, A synthesis on partition refinement: a useful routine for strings, graphs, boolean matrices and automata, in: M. Morvan, C. Meinel, D. Krob (Eds.), STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science, in: LNCS, vol. 1373, Springer, 1998, pp. 25–38.
[45] D.G. Harris, N.S. Narayanaswamy, A faster algorithm for vertex cover parameterized by solution size, in: O. Beyersdorff, M.M. Kanté, O. Kupferman, D. Lokshtanov (Eds.), 41st International Symposium on Theoretical Aspects of Computer Science, STACS, in: LIPIcs, vol. 289, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 40:1–40:18.
[46] I.B.-A. Hartman, Variations on the Gallai-Milgram theorem, Discrete Math. 71 (2) (1988) 95–105.
[47] T. Johnson, N. Robertson, P.D. Seymour, R. Thomas, Directed tree-width, J. Comb. Theory, Ser. B 82 (1) (2001) 138–154.
[48] K.-I. Kawarabayashi, Y. Kobayashi, A linear time algorithm for the induced disjoint paths problem in planar graphs, J. Comput. Syst. Sci. 78 (2) (2012) 670–680.
[49] K.-I. Kawarabayashi, Y. Kobayashi, B. Reed, The disjoint paths problem in quadratic time, J. Comb. Theory, Ser. B 102 (2) (2012) 424–435.
[50] M. Koutecký, Solving hard problems on neighborhood diversity, Master thesis, Charles University in Prague, Chech Republic, Facultas Mathematica Physicaque, Department of Applied Mathematics, April 2013.

[51] M.S. Krishnamoorthy, An NP-hard problem in bipartite graphs, ACM SIGACT News 7 (1) (1975) 26.
[52] S. Kundu, A linear algorithm for the Hamiltonian completion number of a tree, Inf. Process. Lett. 5 (2) (1976) 55–57.
[53] M. Lafond, V. Moulton, Path partitions of phylogenetic networks, Theor. Comput. Sci. 1024 (2025) 114907.
[54] M. Lampis, Algorithmic meta-theorems for restrictions of treewidth, Algorithmica 64 (1) (2012) 19–37.
[55] H.-O. Le, V.B. Le, H. Müller, Splitting a graph into disjoint induced paths or cycles, Discrete Appl. Math. 131 (1) (2003) 199–212.
[56] S. Li, W. Yu, Z. Liu, A local search algorithm for the k-path partition problem, Optim. Lett. 18 (1) (2024) 279–290.
[57] W. Lochet, A polynomial time algorithm for the k-disjoint shortest paths problem, in: ACM-SIAM Symposium on Discrete Algorithms, SODA, SIAM, 2021, pp. 169–178.
[58] C. Magnant, D.M. Martin, A note on the path cover number of regular graphs, Australas. J. Comb. 43 (2009) 211–218.
[59] P.D. Manuel, Revisiting path-type covering and partitioning problems, CoRR ArXiv preprint, arXiv:1807.10613, 2018.
[60] P.D. Manuel, On the isometric path partition problem, Discuss. Math., Graph Theory 41 (4) (2021) 1077–1089.
[61] B. Martin, D. Paulusma, S. Smith, E.J. van Leeuwen, Few induced disjoint paths for H-free graphs, Theor. Comput. Sci. 939 (2023) 182–193.
[62] D. Marx, Chordless cycle packing is fixed-parameter tractable, in: F. Grandoni, G. Herman, P. Sanders (Eds.), 28th Annual European Symposium on Algorithms, ESA, Dagstuhl, Germany, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 173, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 71:1–71:19.
[63] M. Mezzini, On the complexity of finding chordless paths in bipartite graphs and some interval operators in graphs and hypergraphs, Theor. Comput. Sci. 411 (7–9) (2010) 1212–1220.
[64] J. Monnot, S. Toulouse, The path partition problem and related problems in bipartite graphs, Oper. Res. Lett. 35 (5) (2007) 677–684.
[65] H. Müller, Hamiltonian circuits in chordal bipartite graphs, Discrete Math. 156 (1–3) (1996) 291–298.
[66] S.C. Ntafos, S.L. Hakimi, On path cover problems in digraphs and applications to program testing, IEEE Trans. Softw. Eng. SE-5 (5) (1979) 520–529.
[67] J.-J. Pan, G.J. Chang, Isometric-path numbers of block graphs, Inf. Process. Lett. 93 (2) (2005) 99–102.
[68] J.-J. Pan, G.J. Chang, Path partition for graphs with special blocks, Discrete Appl. Math. 145 (3) (2005) 429–436.
[69] J.-J. Pan, G.J. Chang, Induced-path partition on graphs with special blocks, Theor. Comput. Sci. 370 (1–3) (2007) 121–130.
[70] S.S. Pinter, Y. Wolfstahl, On mapping processes to processors in distributed systems, Int. J. Parallel Program. 16 (1) (1987) 1–15.
[71] N. Robertson, P.D. Seymour, Graph minors XIII. The disjoint paths problem, J. Comb. Theory, Ser. B 63 (1) (1995) 65–110.
[72] A. Schrijver, L. Lovasz, B. Korte, H.J. Promel, R.L. Graham, Paths, Flows, and VLSI-Layout, Springer-Verlag, Berlin, Heidelberg, 1990.
[73] W.-K. Shih, W.-L. Hsu, A new planarity test, Theor. Comput. Sci. 223 (1–2) (1999) 179–191.
[74] Z. Skupién, Path partitions of vertices and hamiltonicity of graphs, in: Proceedings of the Second Czechoslovakian Symposium on Graph Theory, Prague, 1974.
[75] A. Slivkins, Parameterized tractability of edge-disjoint paths on directed acyclic graphs, SIAM J. Discrete Math. 24 (1) (2010) 146–157.
[76] G. Steiner, On the k-path partition of graphs, Theor. Comput. Sci. 290 (3) (2003) 2147–2155.
[77] M. Thiessen, T. Gärtner, Online learning of convex sets on graphs, in: M.-R. Amini, S. Canu, A. Fischer, T. Guns, P. Kralj Novak, G. Tsoumakas (Eds.), Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD, Proceedings, Part IV, in: LNCS, vol. 13716, Springer, 2022, pp. 349–364.
[78] M. Thiessen, T. Gärtner, Active learning of convex halfspaces on graphs, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, J. Wortman Vaughan (Eds.), Proceedings of the 35th Conference on Neural Information Processing Systems, NeurIPS, vol. 34, Curran Associates, Inc., 2021, pp. 23413–23425.
[79] C. Wang, Y. Song, G. Fan, H. Jin, L. Su, F. Zhang, X. Wang, Optimizing cross-line dispatching for minimum electric bus fleet, IEEE Trans. Mob. Comput. 22 (4) (2023) 2307–2322.