Edge Geography is XNLP-hard for Pathwidth and in XP for Tree-Partition Width

Thobias Kvalvik Høivik
Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Førde, Norway

Erlend Raa Vågset
Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Førde, Norway

Abstract

Directed Edge Geography and Undirected Edge Geography are classical PSPACE-complete two-player graph games in which players alternately make moves along edges, deleting each one after use; the first player unable to move loses. We prove that both problems are XNLP-hard when parameterized by pathwidth, addressing a question raised by Bodlaender over 30 years ago. On the positive side, we observe that Directed Edge Geography is fixed-parameter tractable when parameterized by treewidth and maximum degree. We also prove that both problems are in XP on simple graphs when parameterized by tree-partition width. These results develop modern lower-bound and decomposition-based algorithmic methods for width-based questions in PSPACE-complete graph games.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Geography games, graph games, pathwidth, parameterized complexity, XNLP-hardness, PSPACE-complete games, tree-partition width

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.114

Related Version *Full Version*: `https://arxiv.org/abs/2607.03189` [21]

Supplementary Material
Software (Source Code): `https://github.com/ThobiasKH/GeographyXPAlgorithm` [20]
archived at `swh:1:dir:138f852ee494fb59e4a77b073493b0e6a1788c53`

Funding Thobias Kvalvik Høivik acknowledges support from StudentFORSK at Western Norway University of Applied Sciences. Erlend Raa Vågset acknowledges support from the HVL Robotics research group at Western Norway University of Applied Sciences.

Acknowledgements We thank the anonymous reviewers for their careful reading and for their thoughtful and constructive feedback, which helped us improve the exposition and clarify several aspects of the paper.

1 Introduction

Many natural computational problems lie beyond NP, yet their parameterized complexity is still relatively poorly understood [12, 2]. Among them, PSPACE-hard problems are especially challenging, since their complexity is often driven by long sequences of interacting choices and evolving states. Such phenomena arise in puzzles such as Sokoban and Rush Hour [11, 16], in generalized games such as Chess and Go [17, 23], in motion planning [22, 27], in task and classical planning [28, 9], and in temporal reasoning [26]. In this paper, we study Directed Edge Geography and Undirected Edge Geography, classical PSPACE-complete graph games [19, 18] that offer a natural setting in which to ask what can and cannot be captured by structural width parameters.

We are not the first to study PSPACE-hard problems from a parameterized perspective. Bäckström and Jonsson [2] develop finer complexity analyses for planning; Mouawad et al. [24] study natural process parameters for reconfiguration; and Bonnet et al. [8] analyze short winning-strategy problems in games. A particularly relevant line of study for us is quantified reasoning. Chen [10] gives an early tractability result for the *quantified constraint satisfaction problem* (QCSP) under bounded treewidth, and Pan and Vardi [25] show that for *quantified Boolean formulas* (QBF), width remains useful when paired with additional control over alternation. Atserias and Oliva [1], however, show that ordinary graph width alone can still be too weak, proving PSPACE-completeness even at constant pathwidth. This in turn motivates the prefix-aware viewpoint of Eiben, Ganian, and Ordyniak [13], who introduce decomposition parameters that reflect the dependency structure induced by the quantifier prefix.

In contrast to quantified reasoning, where graph structure and dependency structure need not align, Edge Geography is played directly on a graph. The issue is therefore not representational mismatch, but whether graph width alone can control an evolving game state. Bodlaender showed that stronger restrictions can make the problem easier. In particular, Edge Generalized Geography is solvable in linear time on graphs of bounded treewidth and bounded maximum degree, and on directed graphs whose underlying undirected graph is a cactus [3]. However, these results do not explain what width bounds alone imply.

We address this question from both a hardness and an algorithmic perspective. For hardness, we study pathwidth and related linear-width parameters, where XNLP is the relevant complexity framework [14, 15, 7, 6]. For algorithms, we consider rooted tree partitions as a stronger decomposition model [29]. Our main results are as follows:

1. As our main result, we prove that Directed Edge Geography is XNLP-hard when parameterized by pathwidth. By a parameter-preserving reduction to the undirected setting, the same also holds for Undirected Edge Geography. Using this transfer together with Bodlaender's bounded-treewidth-and-degree algorithm for Edge Generalized Geography [3], we also obtain that Directed Edge Geography is fixed-parameter tractable when parameterized by treewidth together with maximum degree.
2. On the positive side, we prove that Directed Edge Geography and Undirected Edge Geography on simple graphs, given together with a rooted tree partition of bounded width, are solvable in XP. Combined with the known XP algorithm for computing such decompositions, this yields XP algorithms parameterized by tree-partition width. We also implemented a prototype of the algorithm for correctness validation against a minimax solver; details are given in the full version and in the project repository.

▶ Remark. Due to space constraints, Sections 3 and 5 are presented here in abbreviated form. Their full technical developments are given in the full version [21]. The shorter transfer argument of Section 4 is included in its entirety.

2 Preliminaries

We briefly introduce key definitions, problems and notation. Throughout, we assume all graphs to be finite and simple and we may use the shorthand $[k] := \{1, \ldots, k\}$.

2.1 Edge Geography

▶ **Definition 1** (Edge Geography). *An instance of Edge Geography consists of a graph $G = (V, E)$ together with a designated start vertex $s \in V$, where G is either directed or undirected. Two players alternately move a token, initially placed at s. If the token is*

currently at v, a move consists of selecting an unused edge from v to some vertex u, moving the token to u, and deleting that edge. In the directed variant, the chosen edge must be an outgoing edge $(v, u) \in E$; in the undirected variant, it must be an incident edge $\{v, u\} \in E$. The first player unable to move loses.

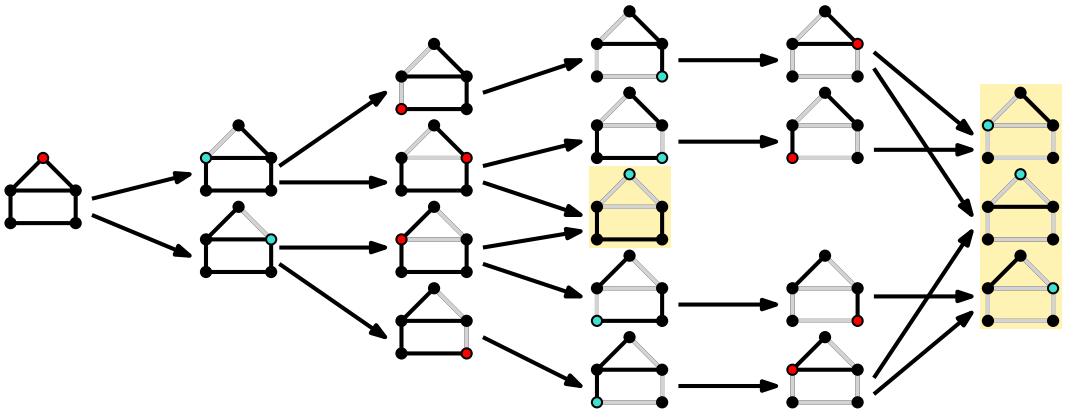

Figure 1 All possible states for a given geography game. This game DAG shows that it is the turquoise player that always becomes "stuck" and hence will always lose the game for this particular graph and initial position.

▶ **Problem 2** (Directed/Undirected Edge Geography).

Input: A directed or undirected graph G and a start vertex $s \in V(G)$.
Question: Does Player 1 have a winning strategy?

Move/edge notation. If the token is at some vertex $v \in V(G)$, then a move consists of choosing an outgoing edge with some endpoint, e.g. $w \in V(G)$. When we write $v \to w$, we mean the move which corresponds to moving the token from vertex v to w, deleting the edge (v, w) ($\{v, w\}$ when referring to the undirected variant), resulting in the opposing player being the one to make a move at w. We also use this notation to refer to the edge (v, w) itself. Often there will only be one legal move from a given vertex. Suppose $v, w, u \in V(G)$ such that there is only one outgoing edge from w, going to u, and that there is an edge from v to w. Then we will often write $v \to w \to u$ to denote the forced sequence that arises from the player to move at v choosing to play $v \to w$, since the opposing player is forced to move the token from w to u. Similarly, when we have edges connected in this way where the intermediate vertex has outdegree exactly 1 we will also often use this notation.

2.2 Graphs and pathwidth

▶ **Definition 3.** *A* path decomposition *of a graph $G = (V, E)$ is a sequence of vertex sets $(X_1, \ldots, X_r)$ such that:*

1. *for every vertex $v \in V$, there exists $i \in [r]$ with $v \in X_i$;*
2. *for every edge $\{u, v\} \in E$, there exists $i \in [r]$ with $\{u, v\} \subseteq X_i$;*
3. *for every vertex $v \in V$, the set $\{\, i \mid v \in X_i \,\}$ forms a contiguous interval of $[r]$.*

We will also use the notion of a *nice path decomposition*: a path decomposition $(B_1, \ldots, B_r)$ is nice if $B_1 = B_r = \emptyset$ and, for every $i \in [r-1]$, the bags B_i and B_{i+1} differ in exactly one vertex. It is well-known that every path decomposition of width k can be transformed in polynomial time into a nice path decomposition of the same width.

▶ **Definition 4.** *The* width *of a path decomposition* $(X_1, \ldots, X_r)$ *is* $\max_i |X_i| - 1$. *The* pathwidth *of a graph* G*, denoted* $\mathrm{pw}(G)$*, is the minimum width over all path decompositions of* G*. The pathwidth of a directed graph* $G = (V, E)$*, is the pathwidth of its underlying undirected graph.*

2.3 Parameterized complexity

A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$. An instance is a pair (x, k), where k is the parameter. A parameterized problem is in *FPT* if it can be decided in time $f(k) \cdot |x|^{O(1)}$ for some computable function f. A parameterized problem is in *XP* if it can be decided in time $|x|^{f(k)}$ for some computable function f.

We will work with the class XNLP, consisting of parameterized problems solvable by a nondeterministic Turing machine in time $f(k) \cdot |x|^{O(1)}$ and space $O(k \log |x|)$. This class was introduced under the notation $\mathrm{N}[f\mathrm{poly}, f \log]$ by Elberfeld, Stockhusen, and Tantau [14], and has subsequently been studied under the name XNLP, e.g. by Bodlaender et al. [7]. We will also use the following two results of Bodlaender [3].

▶ **Theorem 5.** *For every fixed* $k, d \geq 1$*,* Edge Generalized Geography *can be solved in time* $O(n)$ *on graphs of treewidth at most* k *and maximum degree at most* d*, provided together with a tree decomposition of width at most* k*.*

▶ **Theorem 6.** Edge Generalized Geography *can be solved in time* $O(n)$ *on directed graphs whose underlying undirected graph is a cactus.*

In particular, since pathwidth bounds treewidth from above, Bodlaender's first theorem immediately yields linear-time solvability of Undirected Edge Geography on graphs of bounded pathwidth and bounded maximum degree, provided with a path decomposition of bounded width.

3 XNLP-hardness of Directed Edge Geography

We prove XNLP-hardness of Directed Edge Geography by a parameterized logspace reduction from Chosen Maximum Outdegree, which is known to be XNLP-complete when parameterized by pathwidth [4]. Following [4, Section 2.3], we assume that all integers in the source instance are encoded in unary.

Source problem. Let G be an undirected graph. An *orientation* of G assigns a direction to each edge. For an orientation ω and a vertex x, let $\theta_\omega(x)$ denote the set of edges directed out of x, and let $\mathrm{Ori}(G)$ denote the set of all orientations of G.

▶ **Problem 7** (Chosen Maximum Outdegree).

Input: An undirected graph $G = (V, E)$, a weight function $w : E \to \mathbb{Z}_{>0}$, and a vertex bound $t : V \to \mathbb{Z}_{>0}$.
Question: Is there an orientation $\omega \in \mathrm{Ori}(G)$ such that

$$\sum_{e \in \theta_\omega(v)} w(e) \leq t(v)$$

for every $v \in V(G)$?

We assume that the input is given together with a path decomposition of width at most k.

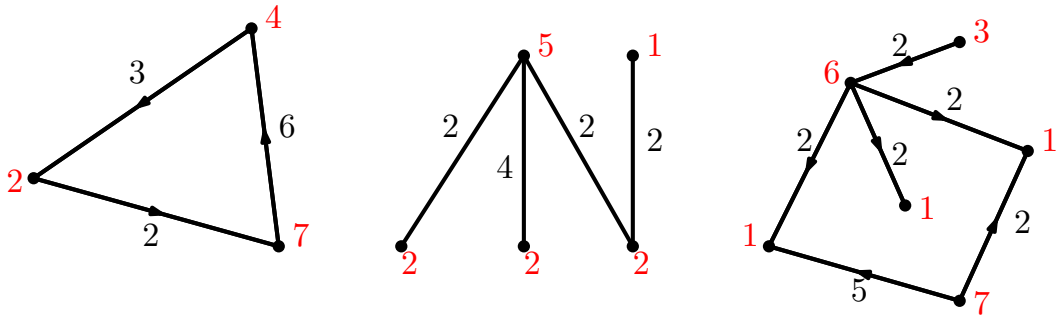

Figure 2 Three instances of Chosen Maximum Outdegree. Vertex bounds $t(x)$ are shown in red. A feasible orientation is shown whenever one exists. The middle instance admits no feasible orientation.

Proof intuition. The reduction encodes an orientation of the input graph by letting Player 1 choose, for each edge gadget, one of two symmetric sides during an initial traversal phase. For an edge $e = \{x, y\}$, traversing the y-side encodes the orientation (x, y), and traversing the x-side encodes (y, x). After all gadgets have been traversed, Player 2 chooses a vertex x in a challenge phase and repeatedly enters the gadget sides corresponding to edges oriented out of x. If the total encoded outgoing weight at x exceeds $t(x)$, then Player 2 can force one more successful challenge excursion than there are available return paths. If the encoded orientation is feasible, then every non-losing challenge excursion consumes one challenge edge and one return path, so Player 2 eventually runs out of non-losing moves.

The edge gadget. For each input edge $e = \{x, y\}$, we construct a directed gadget G_e with two symmetric sides, one corresponding to x and one to y; see Figure 3. Entering one side during the initial phase encodes the orientation of e. Each side contains:

- a forward/backward traversal structure that simulates the weight $w(e)$,
- a merge-and-return path through which the initial traversal leaves the gadget, and
- an escape structure that is used only in the later challenge phase.

The full vertex-and-edge definition is given in the full version.

The full reduction. Given an instance (H, w, t) of Chosen Maximum Outdegree, we build one copy of the edge gadget G_e for each $e \in E(H)$, and chain these gadgets in the order induced by the given path decomposition. Thus the first phase of the game traverses the gadgets one by one and encodes an orientation of every edge of H. After the last gadget, play enters a challenge gadget in which Player 2 chooses a vertex $x \in V(H)$ and enters the sides corresponding to edges oriented out of x. For each vertex x, the challenge gadget provides exactly $t(x)$ return paths. The complete formal construction is given in the full version.

The following lemma summarizes the two key properties of the reduction; the full proof is given in the full version.

▶ **Lemma 8** (Gadget behavior). *The reduction has the following properties.*

1. *During the initial traversal phase, once Player 1 chooses a side of an edge gadget, all moves of Player 2 inside that gadget are forced, and play leaves the gadget at its return vertex with Player 2 to move.*
2. *In the challenge phase, challenging a side that was already used in the initial phase is immediately losing for Player 2. Challenging a side that was left untouched yields a non-losing excursion if and only if one challenge edge and one return path are consumed.*

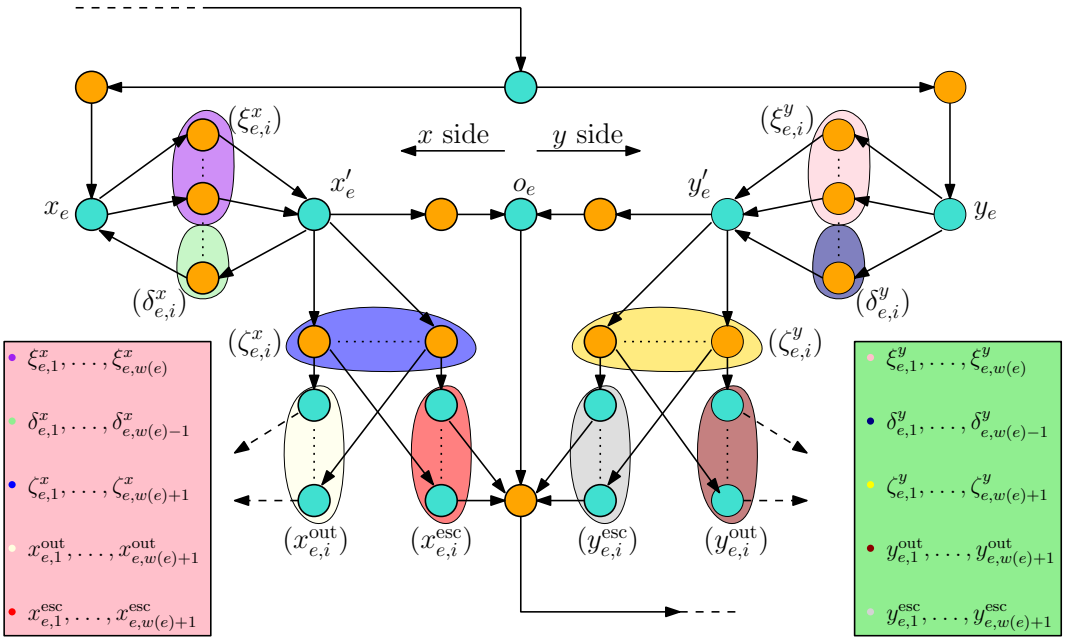

Figure 3 Edge gadget used in the reduction. The x- and y-sides are symmetric; the ξ- and δ-vertices implement the weighted forward/backward traversal, while the ζ-vertices lead either to return vertices $x^{\text{out}}_{e,i}, y^{\text{out}}_{e,i}$ or to escape vertices $x^{\text{esc}}_{e,i}, y^{\text{esc}}_{e,i}$.

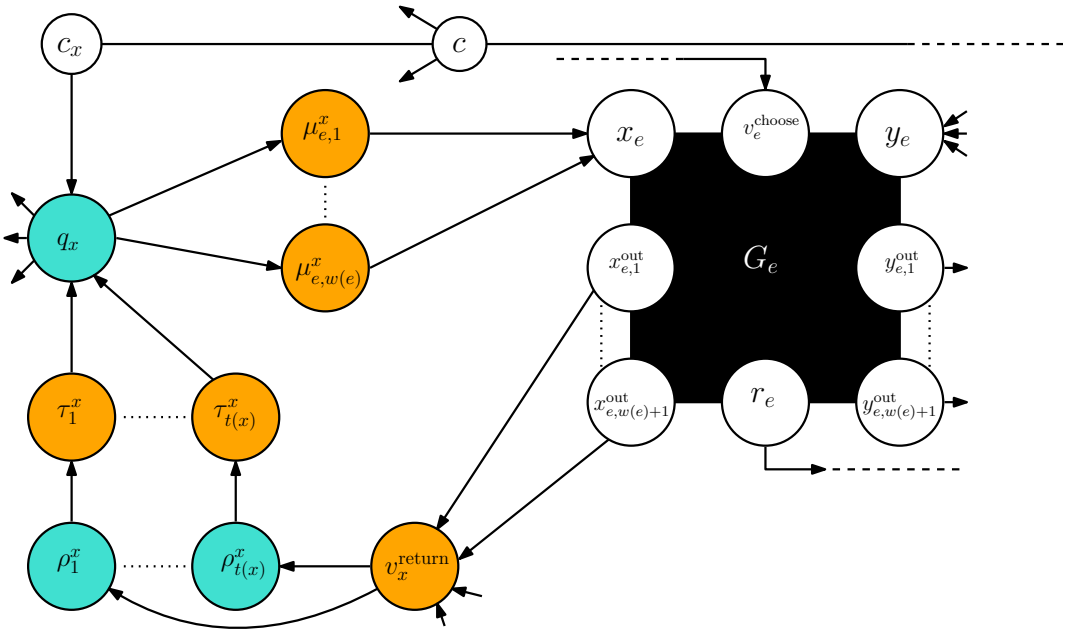

Figure 4 The challenge gadget for a vertex x connected to an edge gadget G_e, with the behavior of G_e abstracted away. Notice how, due to the parity swap after the last edge gadget, Player 2 is now the turquoise player. This is the case inside of G_e also.

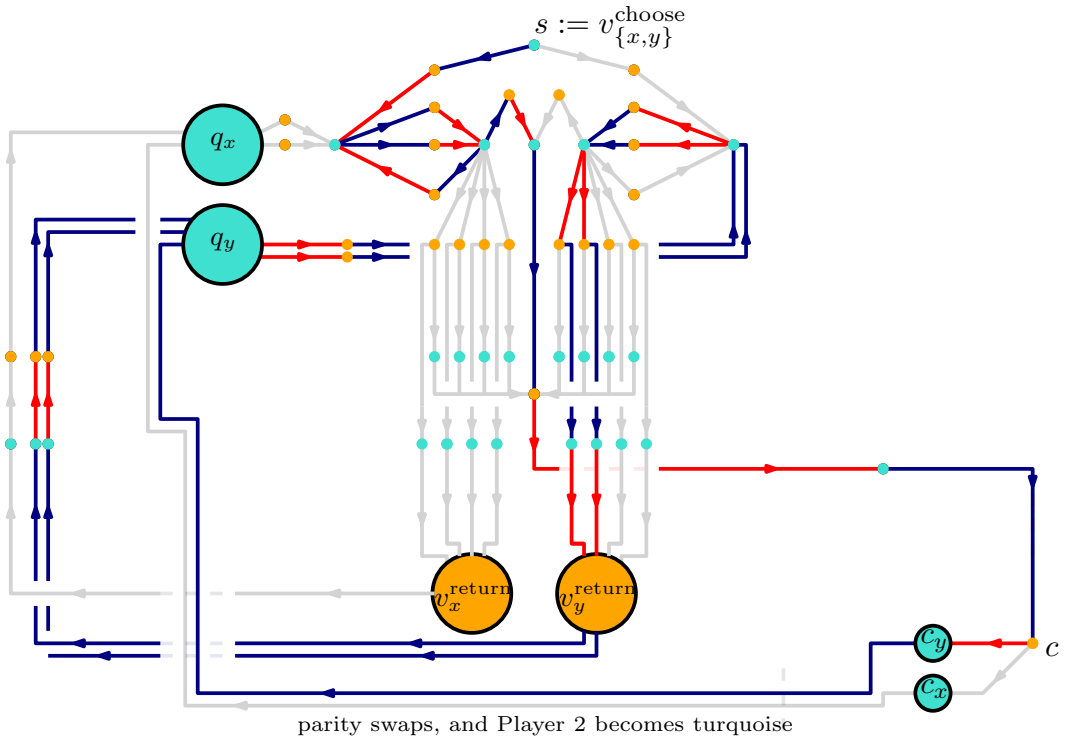

Figure 5 A reduced instance of our input problem for which there is one edge $\{x, y\}$ with weight 2, and $t(x) = 1$ and $t(y) = 2$. This is a feasible instance and we observe that Player 1 has a winning strategy, with Player 2 to move at q_y with no legal continuation. In this figure we assume not only that the players are playing optimally, but that they want to stay alive "as long as possible". The edges played by Player 1 are colored navy and those played by Player 2 are colored red. In particular, observe that in the phase where the edge is oriented, Player 2 only has one legal response every time they move.

▶ **Theorem 9.** *Directed Edge Geography is XNLP-hard when parameterized by pathwidth.*

Proof sketch. We reduce from Chosen Maximum Outdegree.

Correctness. Suppose first that (H, w, t) is a yes-instance, and fix a feasible orientation $\omega \in \text{Ori}(H)$. Player 1 uses the initial traversal phase to encode ω by choosing, for each edge gadget, the side corresponding to the head of the oriented edge. By Lemma 8, all responses of Player 2 inside the gadgets are forced, and after all gadgets have been processed, play reaches the challenge gadget with Player 2 to move.

Now Player 2 chooses a vertex x. The only non-losing challenges are exactly those corresponding to edges directed out of x, and there are $\sum_{e \in \theta_\omega(x)} w(e)$ such challenge opportunities in total. Each non-losing challenge consumes exactly one return path, and the number of return paths available at x is $t(x)$. Since ω is feasible, we have

$$\sum_{e \in \theta_\omega(x)} w(e) \leq t(x),$$

so eventually all non-losing challenges are exhausted and Player 2 loses.

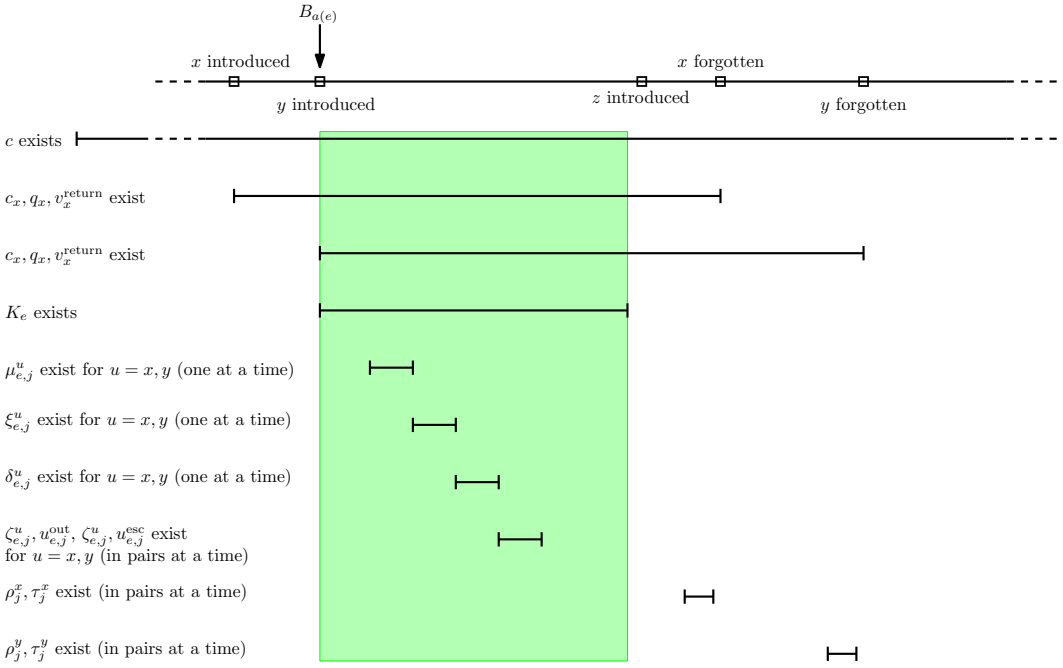

Figure 6 Schematic view of the pathwidth-preserving construction for an edge gadget $e = \{x, y\}$. The top timeline indicates the intervals during which x and y are present. The rows below indicate the corresponding lifetimes of the anchor vertices, together with the global vertex c. The green block indicates the dedicated contiguous block of bags in the constructed decomposition where e is realized: throughout K_e, which is meant to represent the constant-size core of the edge gadget, is present together with S_i, while the auxiliary vertices are introduced only locally, one at a time or in pairs. Finally, before a vertex is forgotten, the path-return vertices are realized in pairs at a time.

Conversely, suppose (H, w, t) is a no-instance. Whatever Player 1 does in the initial phase, the choices of sides encode some orientation ω. Since (H, w, t) is infeasible, there exists a vertex x with

$$\sum_{e \in \theta_\omega(x)} w(e) > t(x).$$

Player 2 chooses this vertex in the challenge phase. By Lemma 8, each non-losing challenge consumes exactly one return path, and Player 2 has strictly more such challenges available than the number of return paths at x. Hence Player 2 can force one final successful challenge after all return paths have been exhausted, leaving Player 1 without a move. Therefore Player 2 wins.

Pathwidth preservation. Let $(B_1, \ldots, B_r)$ be a nice path decomposition of H of width k. We order the edge gadgets according to the first bag containing both endpoints of the corresponding input edge. For each bag B_i, we keep a set of *anchor vertices*: the global challenge vertex together with the challenge vertices associated with the original vertices currently present in B_i. Since $|B_i| \leq k + 1$, the number of anchor vertices in a bag is $O(k)$.

Each edge gadget is then realized in a contiguous block of bags in which the relevant anchor vertices are kept throughout, while the auxiliary gadget vertices are introduced and forgotten locally. Similarly, when an original vertex x is forgotten, the $t(x)$ return paths associated with x are realized in a short contiguous block before the anchor vertices of x are removed. This yields a valid path decomposition of the reduction graph of width $O(k)$. The full construction and verification are given in the full version.

The construction is computable by a deterministic parameterized logspace transducer. Indeed, the output graph can be streamed by scanning the input path decomposition and enumerating edge gadgets in the order of the first bag containing both endpoints. Each output vertex and edge is specified by a constant-size type label, an input vertex or edge, and a counter bounded by $w(e)+O(1)$ or $t(v)+O(1)$. Since all integers in the source instance are encoded in unary, the output has polynomial size and these counters use only $O(\log n)$ bits. The path decomposition witnessing the $O(k)$ pathwidth bound is generated analogously.

Together with the correctness argument and the pathwidth bound above, this proves the claimed XNLP-hardness. ◀

4 Reducing to the undirected variant

We now transfer hardness from the directed game to the undirected one.

▶ **Lemma 10.** *Let G be a directed graph with $\mathrm{pw}(G) = k$. There exists an undirected graph G' such that:*

1. *the* Directed Edge Geography *instance on G is equivalent to the* Undirected Edge Geography *instance on G';*
2. *each directed edge of G is replaced by a constant-size gadget;*
3. *$\mathrm{pw}(G') \leq f(k)$ for some function $f(k) = O(k^2)$.*

Proof. For each directed edge $(u, v) \in E(G)$, replace it by the well-known constant-size gadget from the *Undirected Edge Geography* paper [18, Theorem 2.1] that simulates a directed edge: it admits traversal from u to v, while entering from v results in a losing position under optimal play.

Let $(X_1, \ldots, X_r)$ be a path decomposition of G of width k. For each edge (u, v), let I_{uv} be the interval of bags containing both u and v. Place every vertex of the gadget replacing (u, v) in all bags indexed by I_{uv}.

All gadget edges are covered inside the corresponding interval, and contiguity is immediate. In a fixed bag X_i, there are at most $\binom{k+1}{2} = O(k^2)$ pairs of vertices from X_i, and each contributes only a constant number of gadget vertices. Hence each bag increases in size by at most $O(k^2)$. ◀

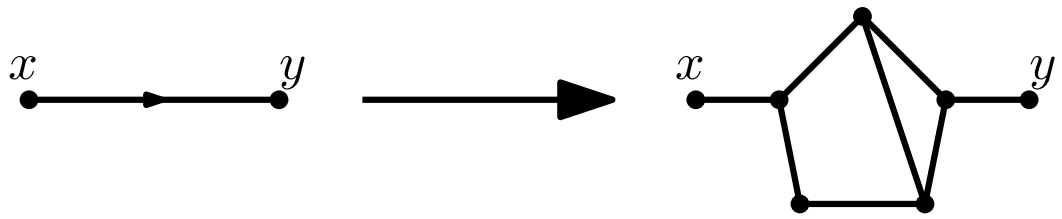

Figure 7 Pseudoarc used in reduction. Left: a directed edge (x, y). Right: a pseudoarc simulating the directed edge on the left. We observe that entry from x yields safe exit at y under optimal play, while entry the other way results in a loss for the entrant.

▶ **Corollary 11.** *Undirected Edge Geography is XNLP-hard when parameterized by pathwidth.*

Proof. By the previous theorem, Directed Edge Geography is XNLP-hard when parameterized by pathwidth. By Lemma 10, there is a parameter-preserving reduction from the directed game to the undirected one. Therefore Undirected Edge Geography is XNLP-hard when parameterized by pathwidth. ◀

▶ **Corollary 12.** *Directed Edge Geography can be solved in time $f'(k,d) \cdot n$ on directed graphs of pathwidth at most k and maximum degree at most d, provided with a path decomposition of width at most k.*

Moreover, the same holds on directed graphs of treewidth at most k and maximum degree at most d, provided with a tree decomposition of width at most k.

Proof. Let G be an instance of Directed Edge Geography on a directed graph of pathwidth at most k and maximum degree at most d, together with a path decomposition of width at most k.

Apply Lemma 10 to obtain an equivalent instance G' of Undirected Edge Geography. By the lemma, each directed edge of G is replaced by a constant-size gadget, so $|V(G')| = O(|V(G)| + |E(G)|)$, the maximum degree of G' is bounded by a function of d, and $\mathrm{pw}(G') \leq g(k)$ for some function g.

Now Undirected Edge Geography is a special case of Edge Generalized Geography, and pathwidth bounds treewidth from above. Hence Theorem 5 implies that G' can be solved in time $f(g(k), d') \cdot |V(G')|$, where d' is the maximum degree of G'. Since d' depends only on d and $|V(G')|$ is linear in the size of G, this running time is of the form $f'(k,d) \cdot n$.

Because the reduction preserves the winner, the same bound holds for Directed Edge Geography.

For the treewidth statement, one argues in the same way starting from a tree decomposition. Using a nice tree decomposition with introduce-edge bags, one inserts the constant-size gadget when the corresponding edge is introduced, places the new gadget vertices in that bag, and forgets them immediately afterwards. This yields an equivalent undirected instance whose treewidth and maximum degree are bounded by functions of the original parameters, so Bodlaender's theorem applies exactly as above. ◀

5 An XP Algorithm on Given Tree Partitions

In this section we show that Undirected Edge Geography is in XP on simple graphs when the input is given together with a rooted tree partition of bounded width. By the standard reduction from the previous section, the same will then hold for Directed Edge Geography.

The key idea is to summarize each child subtree by a finite *interface type* describing how play can enter the subtree from its parent bag, how it may later return to the parent bag, and what smaller residual child subinstance remains afterwards. For fixed width k, only finitely many such types can occur. Moreover, children of the same type are interchangeable from the perspective of the parent bag. This allows us to compress the state of a bag to the multiplicities of the child types that are still present.

A complete formal development of the type universe, the replacement lemma, the local semantics, and a worked example are given in the full version. Implementation details are also given in the full version, and a proof-of-concept implementation can be found at `https://github.com/ThobiasKH/GeographyXPAlgorithm`.

5.1 Interfaces and types

We use a rooted version of the notion of a tree partition.

▶ **Definition 13** (Rooted tree partition)**.** *A* rooted tree partition *of a graph G is a pair $(\{X_i \mid i \in I\}, T)$, where $\{X_i \mid i \in I\}$ is a partition of $V(G)$ and T is a rooted tree with node set I, such that every edge of G is either contained in a bag X_i or has its endpoints in two bags whose nodes are adjacent in T. The* width *of the tree partition is* $\max_{i \in I} |X_i|$.

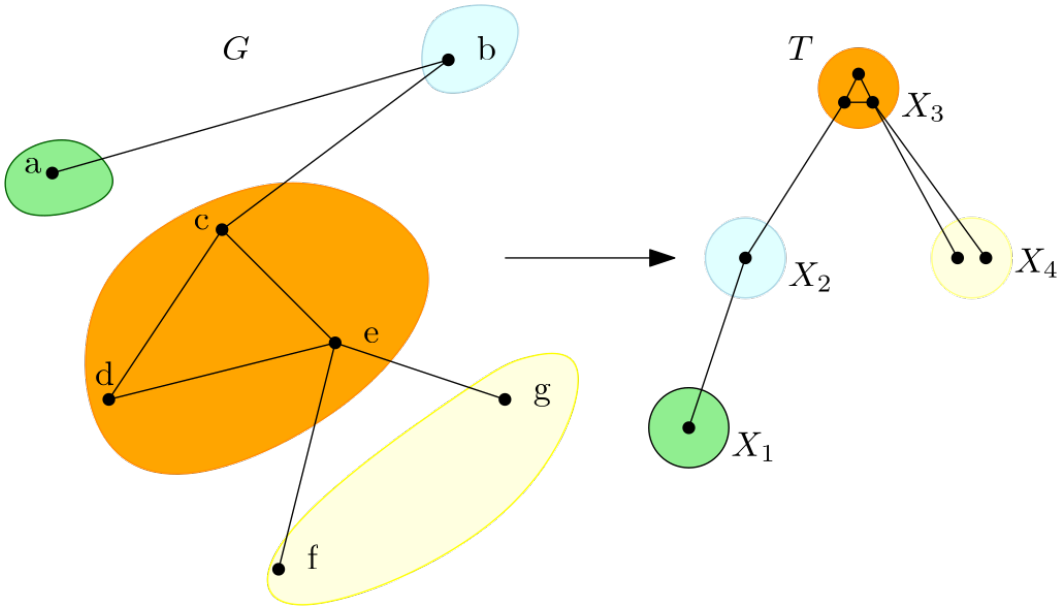

Figure 8 A tree partition of a graph G with width $k = 3$.

Throughout this section, let $(\{X_i \mid i \in I\}, T)$ be a rooted tree partition of width k of the input graph G. If i is a non-root node, let $p(i)$ denote its parent in T. The *parent cut* of X_i is the set of edges with one endpoint in X_i and the other endpoint in $X_{p(i)}$. We call each edge in this cut a *port* of X_i. The endpoint in $X_{p(i)}$ is the *parent-side endpoint* of the port, and the endpoint in X_i is the *child-side endpoint*.

Since G is simple and both bags have size at most k, the parent cut has size at most k^2. After fixing an injective labeling $\iota_i : X_i \hookrightarrow [k]$ for each bag X_i, every port receives two labels: the label of its parent-side endpoint and the label of its child-side endpoint.

Interface type, abbreviated. Let (P, λ) be a labeled interface, where P is a set of ports and $\lambda : P \to [k]$ records the parent-side labels. An interface type on (P, λ) consists, for each entry port $a \in P$, of:

1. a set of possible *exit labels* (q, τ, b), where $q \in P \setminus \{a\}$ is the return port, τ is the type of the smaller residual child subinstance that remains after the excursion, and $b \in \{\mathsf{same}, \mathsf{opp}\}$ records the parity of the return;
2. a Boolean function Φ_a telling whether the player entering through a can force a win, given the truth values of the attainable returns.

The fully rigorous recursive definition is given in the full version.

Intuitively, an interface type is a finite summary of the boundary behavior of a child subtree: it records how play may enter the subtree, return to the parent bag, and leave behind a smaller residual subinstance. The set of attainable returns alone is not enough, since the value of entering the child also depends on which of those returns are winning in the surrounding parent-side game. The Boolean functions Φ_a encode exactly this dependence. A useful mental model is thinking of any subtree of a bag X_i as a gadget that may only be entered (and possibly exited) a certain number of times, and the parent does not care about the exact inner workings of this gadget.

For fixed k, there are only finitely many realizable interface types. We denote this finite set by $\mathcal{T}_k$, and note that $|\mathcal{T}_k|$ is entirely dependent on k.

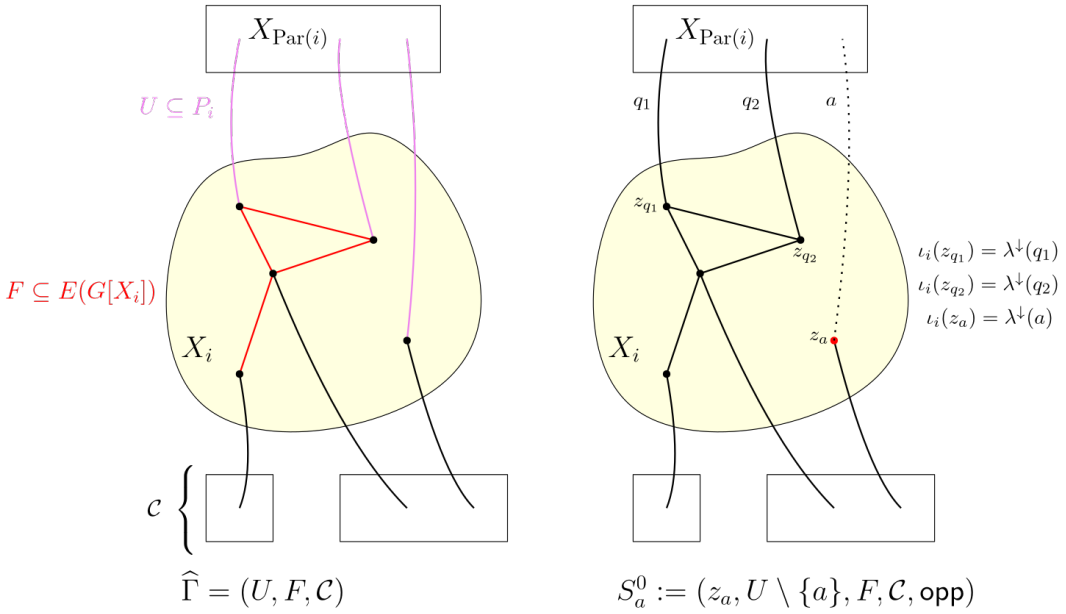

Figure 9 A residual configuration and entry state at a bag X_i. Left: an expanded residual configuration $\widehat{\Gamma} = (U, F, \mathcal{C})$ consisting of unused parent ports U, unused internal edges F, and multiset of residual child subinstances $\mathcal{C}$ (In the full technical development of this approach we consider multisets of child subinstances before showing that this multiset can be compressed to a multiplicity vector). Right: after entering through $a \in U$, the local game starts in the entry state $S_a^0 = (z_a, U \setminus \{a\}, F, \mathcal{C}, \mathsf{opp})$, where $z_a \in X_i$ is the child-side endpoint of a. From this entry state, play may only proceed by a child-side excursion.

5.2 Compressed residual configurations

Fix a non-root bag X_i. A *compressed residual configuration* at i is a triple $\Gamma = (U, F, \mathbf{m})$, where

- U is the set of unused parent ports of i,
- $F \subseteq E(G[X_i])$ is the set of unused internal edges of X_i,
- $\mathbf{m} = (m_\sigma)_{\sigma \in \mathcal{T}_k}$ is a multiplicity vector recording how many residual child subinstances of each realizable type are still present below i.

The point of compression is that the identities of the child subinstances do not matter: only their types do. The full version proves a replacement lemma showing that if one residual child subinstance is replaced by another of the same type, then the parent-side outcome does not change. Consequently, the type of a bag configuration is determined entirely by $(U, F, \mathbf{m})$.

5.3 Local games inside a bag

Let $\Gamma = (U, F, \mathbf{m})$ be a compressed residual configuration at a non-root node i, and let $a \in U$ be an entry port. We consider the local game that starts when play enters the current subtree through a.

A *compressed local state* has the form $S = (z, U', F', \mathbf{m}', \pi)$, where

- $z \in X_i$ is the current token position,
- $U' \subseteq U \setminus \{a\}$ is the set of still unused parent ports,
- $F' \subseteq F$ is the set of still unused internal bag edges,
- $\mathbf{m}' \leq \mathbf{m}$ is the current multiplicity vector of residual child types,
- $\pi \in \{\mathsf{same}, \mathsf{opp}\}$ records whether the player to move is the same as or opposite to the player who originally entered through a.

From such a state, play may proceed in one of three ways:

1. *Internal move:* traverse an unused edge of $G[X_i]$ incident with z;
2. *Exit move:* leave the current subtree through a port $q \in U'$ whose child-side label matches z;
3. *Child excursion:* enter a residual child subinstance of some type σ with $m'_\sigma > 0$ through a port whose label matches z. If the child later returns with exit label (q, τ, b), then the successor state replaces one copy of σ in the multiplicity vector by one copy of τ, moves the token to the port q, and updates the parity by b.

The full formal statement of these transitions, together with the proof that they faithfully capture the underlying expanded game, is given in the full version. The crucial point is that this local game is acyclic. Indeed, if we define the measure

$$\mu(S) := |U'| + |F'| + \sum_{\sigma \in \mathcal{T}_k} \mathrm{rk}(\sigma) m'_\sigma,$$

then every internal move strictly decreases $|F'|$, and every returning child excursion replaces a type by a strictly smaller residual type, thereby decreasing the rank contribution. Hence every transition strictly decreases $\mu(S)$.

It follows that for every pair (Γ, a), the local game can be solved by reverse dynamic programming on a finite acyclic graph. From these values we can compute:

- the set A_a^Γ of attainable exit labels from entry through a,
- the Boolean function Φ_a^Γ describing whether the entrant can force a win as a function of the values of those returns.

Together, these determine the interface type $\tau(\Gamma)$ of the compressed residual configuration.

5.4 Bottom-up computation and the root bag

We now compute all relevant types bottom-up over the rooted tree partition.

Fix a node i. Since the children of i have already been processed, the set of realizable child types that may appear below i is known. We consider all compressed residual configurations arising at i, ordered by measure. For each such Γ and each entry port $a \in U$, we solve the associated local game as above and thereby compute $\tau(\Gamma)$.

For fixed k, let $t_k := |\mathcal{T}_k|$. Since the parent cut of a bag has size at most k^2, there are at most 2^{k^2} choices for the unused port set U. Since a bag contains at most $\binom{k}{2}$ internal edges, there are at most $2^{\binom{k}{2}}$ choices for F. Finally, each multiplicity vector $\mathbf{m} = (m_\sigma)_{\sigma \in \mathcal{T}_k}$ has t_k coordinates, each lying in $\{0, \ldots, n\}$, and hence there are at most $(n+1)^{t_k}$ possibilities for $\mathbf{m}$. Thus the number of compressed residual configurations at a bag is at most $2^{k^2 + \binom{k}{2}}(n+1)^{t_k} = n^{f_1(k)}$.

For a fixed pair (Γ, a), a compressed local state is determined by $z \in X_i$, a subset $U' \subseteq U \setminus \{a\}$, a subset $F' \subseteq F$, a multiplicity vector $\mathbf{m}' \leq \mathbf{m}$, and a parity bit. Therefore the number of compressed local states is at most $2k \cdot 2^{k^2 + \binom{k}{2}}(n+1)^{t_k} = n^{f_2(k)}$. Moreover, each such state has only $f_3(k)$ outgoing transitions: the number of internal moves is bounded

by $\binom{k}{2}$, the number of exits by k^2, and the number of child excursions by a function of k, since $\mathcal{T}_k$ is finite and each type has rank at most k^2. Since the local game is acyclic, its values can therefore be computed in time $n^{f_4(k)}$. Summing over all bags still yields total running time $n^{f(k)}$.

At the root bag X_r, there is no parent interface. After all child types have been computed, we solve an analogous compressed root game whose states have the form $R = (z, F, \mathbf{m}, \pi)$, where $z \in X_r$, $F \subseteq E(G[X_r])$, $\mathbf{m}$ is the multiplicity vector of the residual child types of the root's children, and $\pi \in \{\mathsf{same}, \mathsf{opp}\}$ records the player to move relative to the initial player.

As before, the root game is acyclic, since every internal move consumes an internal root edge and every returning child excursion strictly decreases the total rank contribution of the child types. Thus it can again be solved by reverse dynamic programming. The initial root state is $R_0 = (s, E(G[X_r]), \mathbf{m}_r, \mathsf{same})$, and Player 1 has a winning strategy in the original instance if and only if R_0 is winning in this compressed root game.

▶ **Theorem 14.** *Undirected Edge Geography on simple graphs, given together with a rooted tree partition of width k, is solvable in XP time parameterized by k.*

Proof sketch. For fixed k, only finitely many realizable interface types can occur. At a bag X_i, a compressed residual configuration is determined by the unused parent ports, the unused internal bag edges, and a multiplicity vector over the finite type universe $\mathcal{T}_k$. Hence the number of such configurations is bounded by $n^{f_1(k)}$. For each configuration and entry port, the associated compressed local game has at most $n^{f_2(k)}$ states and only $f_3(k)$ outgoing transitions per state, and is acyclic. Its values can therefore be computed by reverse dynamic programming in time $n^{f_4(k)}$. Processing all bags bottom-up and finally solving the analogous root game yields the winner of the original instance in time $n^{f(k)}$. Thus the problem belongs to XP. ◀

▶ **Corollary 15.** *Directed Edge Geography on simple graphs, given together with a rooted tree partition of width k, is solvable in XP time parameterized by k.*

Proof sketch. Reduce the directed instance to an equivalent undirected instance by replacing each directed edge by the standard constant-size gadget used in the reduction earlier. As shown in the full version, the given rooted tree partition can be transformed into one of width bounded by a function of k for the resulting undirected graph. The theorem above then applies. ◀

▶ Remark 16. By a result of Bodlaender, Groenland, and Jacob [5], given an n-vertex graph G and an integer k, one can in time $k^{O(1)}n^2$ either compute a tree partition of width $O(k^7)$ or correctly conclude that the tree-partition-width of G is greater than k. Since a tree partition can be rooted arbitrarily without changing its width, Theorem 14 also yields XP algorithms for Undirected Edge Geography and Directed Edge Geography parameterized by tree-partition width.

6 Conclusion

We proved that both Directed Edge Geography and Undirected Edge Geography are XNLP-hard when parameterized by pathwidth. Bodlaender left open what width bounds alone imply for Edge Generalized Geography. Our XNLP-hardness result for pathwidth does not settle the possibility of polynomial-time solvability for each fixed width bound, but it shows that width alone is unlikely to yield an FPT-type parameterized algorithm, unless FPT=XNLP. On the positive side, we showed that both variants are in XP on simple

graphs when parameterized by tree-partition width. We also showed that Directed Edge Geography is fixed-parameter tractable when parameterized by treewidth and maximum degree. Together, these results clarify that width-based tractability for edge geography depends strongly on the decomposition model: pathwidth already captures enough structure for hardness, while tree partitions still admit a decomposition-based dynamic program.

6.1 Discussion

Our hardness result suggests that the difficulty of the pathwidth case is not merely technical. In edge-deletion geography, a small separator does not localize play in the same way as for many vertex-based problems. Separator vertices may be revisited many times, while the incident edges are consumed one by one, so the interaction across the separator depends on a history that is not naturally summarized by a small local state. From this perspective, the main obstacle to upper bounds for pathwidth is not simply the lack of the right dynamic program, but the apparent absence of a compact interface that captures the residual effect of partial play.

At the same time, the XP algorithm on rooted tree partitions shows that decomposition-based methods do become viable once the interaction between pieces is sufficiently restricted. In that setting, one can push richer interface information through the decomposition and still retain enough structure for a finite-type compression argument. This suggests that the right structural boundary for geography-type games may lie not between hard and easy graph classes in the ordinary sense, but between decomposition models that do or do not support a controlled summary of residual play.

More broadly, edge geography may be viewed as a canonical token-moving model for alternating play on explicit residual-state graphs. For example, it can serve as an intermediate model for reductions from games or reconfiguration processes in which moves irreversibly consume resources, force local responses, or gradually restrict the future state space. Whenever the evolution of a game can be represented by an acyclic residual-state graph of manageable size, winner determination reduces to a geography-type reachability problem on that graph. Even in settings where the residual-state graph contains cycles, a similar perspective may still apply after augmenting states with bounded progress information that restores acyclicity. In this sense, the present work suggests a broader program of studying which decomposition parameters can support structural algorithms for geography-type games and related PSPACE-complete problems.

Methodologically, Edge Geography is also a useful intermediate problem. Reductions are often more natural to design in the directed setting, where one can encode asymmetric choices and forced traversals more explicitly. On the other hand, undirected formulations are often more intuitive from an algorithmic point of view. In our setting, this asymmetry is especially helpful because Directed Edge Geography admits a simple constant-size gadget reduction to Undirected Edge Geography that preserves the winner and controls the structural parameters typically of interest. Thus one may often work in the directed variant when designing reductions, and then transfer the result to the undirected variant essentially for free.

6.2 Future Work

The main open question is the parameterized complexity of Directed Edge Geography and Undirected Edge Geography with respect to pathwidth alone. In particular, it remains open whether either problem admits an XP algorithm, or belongs to some other natural class under this parameterization, or whether these problems could be complete for a larger parameterized space class, such as para-PSPACE.

Furthermore, it would be interesting to test whether the interface-based viewpoint developed here extends to other PSPACE-complete graph games and residual-state models. This may help identify which kinds of boundary information are sufficient for decomposition-based algorithms beyond the specific case of edge geography.

Lastly, from the reviewers' feedback we posit the following conjectures and open questions.

▶ **Conjecture 17** (Treewidth XALP lower bound). *Directed Edge Geography is XALP-hard parameterized by treewidth.*

If the above conjecture holds then, by the transfer argument from earlier, XALP-hardness holds for the undirected version also.

▶ **Conjecture 18** (Tree-partition width XALP lower bound). *Undirected Edge Geography is XALP-hard parameterized by tree-partition width.*

If the above conjecture holds then it is natural to ask whether there exists an XALP algorithm for Undirected Edge Geography parameterized by tree-partition width, and whether the same lower bound holds for the directed variant as well.

References

1. Albert Atserias and Sergi Oliva. Bounded-width QBF is PSPACE-complete. *Journal of Computer and System Sciences*, 80(7):1415–1429, 2014. doi:10.1016/j.jcss.2014.04.014.
2. Christer Bäckström and Peter Jonsson. All PSPACE-complete planning problems are equal but some are more equal than others. In *Proceedings of the Fourth Annual Symposium on Combinatorial Search (SOCS-11)*, pages 10–17. AAAI Press, 2011. doi:10.1609/socs.v2i1.18186.
3. Hans L. Bodlaender. Complexity of path-forming games. *Theoretical Computer Science*, 110(1):215–245, 1993. doi:10.1016/0304-3975(93)90357-Y.
4. Hans L. Bodlaender, Gunther Cornelissen, and Marieke van der Wegen. Problems hard for treewidth but easy for stable gonality. In Michael A. Bekos and Michael Kaufmann, editors, *Graph-Theoretic Concepts in Computer Science*, volume 13453 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2022. doi:10.1007/978-3-031-15914-5_7.
5. Hans L. Bodlaender, Carla Groenland, and Hugo Jacob. On the parameterized complexity of computing tree-partitions. In *17th International Symposium on Parameterized and Exact Computation (IPEC 2022)*, volume 249 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:20, 2022. doi:10.4230/LIPIcs.IPEC.2022.7.
6. Hans L. Bodlaender, Carla Groenland, Hugo Jacob, Lars Jaffke, and Paloma T. Lima. XNLP-completeness for parameterized problems on graphs with a linear structure. *Algorithmica*, 87(4):465–506, 2025. doi:10.1007/s00453-024-01274-9.
7. Hans L. Bodlaender, Carla Groenland, Jesper Nederlof, and Céline M. F. Swennenhuis. Parameterized problems complete for nondeterministic FPT time and logarithmic space. *Information and Computation*, 300:105195, 2024. doi:10.1016/j.ic.2024.105195.
8. Édouard Bonnet, Serge Gaspers, Antonin Lambilliotte, Stefan Rümmele, and Abdallah Saffidine. The parameterized complexity of positional games. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 90:1–90:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2017.90.
9. Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994. doi:10.1016/0004-3702(94)90081-7.

10 Hubie Chen. Quantified constraint satisfaction and bounded treewidth. In Ramón López de Mántaras and Lorenza Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 161–165. IOS Press, 2004. URL: `https://dblp.org/rec/conf/ecai/Chen04.html`.

11 Joseph C. Culberson. Sokoban is PSPACE-complete. In Elena Lodi, Linda Pagli, and Nicola Santoro, editors, *Proceedings of the International Conference on Fun with Algorithms (FUN '98)*, pages 65–76. Carleton Scientific, 1998. Conference held in Elba, Italy, June 18–20, 1998.

12 Ronald de Haan and Stefan Szeider. Parameterized complexity classes beyond para-NP. *Journal of Computer and System Sciences*, 87:16–57, 2017. `doi:10.1016/j.jcss.2017.02.002`.

13 Eduard Eiben, Robert Ganian, and Sebastian Ordyniak. Using decomposition-parameters for QBF: Mind the prefix! *Journal of Computer and System Sciences*, 110:1–21, 2020. `doi:10.1016/j.jcss.2019.12.005`.

14 Michael Elberfeld, Christoph Stockhusen, and Till Tantau. On the space complexity of parameterized problems. In Dimitrios M. Thilikos and Gerhard J. Woeginger, editors, *Parameterized and Exact Computation*, volume 7535 of *Lecture Notes in Computer Science*, pages 206–217. Springer, 2012. `doi:10.1007/978-3-642-33293-7_20`.

15 Michael Elberfeld, Christoph Stockhusen, and Till Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015. `doi:10.1007/s00453-014-9944-y`.

16 Gary William Flake and Eric B. Baum. Rush hour is PSPACE-complete, or "why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1–2):895–911, 2002. `doi:10.1016/S0304-3975(01)00173-6`.

17 Aviezri S. Fraenkel and David Lichtenstein. Computing a perfect strategy for n x n chess requires time exponential in n. *Journal of Combinatorial Theory, Series A*, 31(2):199–214, 1981. `doi:10.1016/0097-3165(81)90016-9`.

18 Aviezri S. Fraenkel, Edward R. Scheinerman, and Daniel Ullman. Undirected edge geography. *Theoretical Computer Science*, 112(2):371–381, 1993. `doi:10.1016/0304-3975(93)90026-P`.

19 Aviezri S. Fraenkel and Shai Simonson. Geography. *Theoretical Computer Science*, 110(1):197–214, 1993. `doi:10.1016/0304-3975(93)90356-X`.

20 Thobias Kvalvik Høivik. GeographyXPAlgorithm. Software, swhId: `swh:1:dir:138f852ee494fb59e4a77b073493b0e6a1788c53` (visited on 2026-08-11). URL: `https://github.com/ThobiasKH/GeographyXPAlgorithm`, `doi:10.4230/artifacts.27616`.

21 Thobias Kvalvik Høivik and Erlend Raa Vågset. Edge geography is XNLP-hard for pathwidth and in XP for tree-partition width, 2026. `arXiv:2607.03189`.

22 John E. Hopcroft, Jacob T. Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the Warehouseman's Problem. *The International Journal of Robotics Research*, 3(4):76–88, 1984. `doi:10.1177/027836498400300405`.

23 David Lichtenstein and Michael Sipser. GO is polynomial-space hard. *Journal of the ACM*, 27(2):393–401, 1980. `doi:10.1145/322186.322201`.

24 Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, Narges Simjour, and Akira Suzuki. On the parameterized complexity of reconfiguration problems. *Algorithmica*, 78(1):274–297, 2017. `doi:10.1007/s00453-016-0159-2`.

25 Guoqiang Pan and Moshe Y. Vardi. Fixed-parameter hierarchies inside PSPACE. In *21st Annual IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 27–36. IEEE Computer Society, 2006. `doi:10.1109/LICS.2006.25`.

26 A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985. `doi:10.1145/3828.3837`.

27 Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015. `doi:10.15607/RSS.2015.XI.046`.

28 William R. Vega-Brown and Nicholas Roy. Task and motion planning is PSPACE-complete. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10385–10392, 2020. doi:10.1609/aaai.v34i06.6607.

29 David R. Wood. On tree-partition-width. *European Journal of Combinatorics*, 30(5):1245–1253, 2009. doi:10.1016/j.ejc.2008.11.010.