Geography, Kotzig's Nim, and Variants

Srinivas Arun

Under the direction of

Joshua Messing
MIT Department of Mathematics

Research Science Institute
August 1st, 2023

Abstract

Geography is a combinatorial game in which two players take turns moving a token along edges of a directed graph and deleting the vertex they came from. We study Kotzig's Nim, a special case of Geography where the vertices are labeled and moves correspond to additions by fixed amounts. We prove a conjecture by Fraenkel that Kotzig's Nim is eventually periodic in the size of the graph. We also expand upon work by Fox and Geissler, who classified the computational complexity of determining the winner of various Geography variants given a graph. In particular, we show NP-hardness for undirected partizan Geography with free deletion on bipartite graphs and directed partizan Geography with free deletion on acyclic graphs.

Summary

Geography is a two-player game played on a set of vertices with some directed edges connecting pairs of them. There is a token on one vertex, and players take turns moving the token along an edge. No vertex may be visited twice, and the player who first runs out of moves loses. We start by investigating Kotzig's Nim, which is Geography on a graph of a certain form. We prove a conjectue by Fraenkel that the outcomes of Kotzig's Nim for different sizes of the graph eventually repeat in a predictable way. Next, we investigate algorithms which can be used to calculate the winner in variants of Geography. We show that several variants of Geography are computationally difficult, i.e. probably impossible to solve quickly relative to the input size.

1 Introduction

The game of *Geography* is played as follows. A directed graph on n vertices is given, and the game starts with a token on some vertex. Two players take turns moving the token along edges such that the token never visits the same vertex twice. The first player who cannot move loses.

One special case of Geography is *Kotzig's Nim*. In Kotzig's Nim, the graph consists of n vertices in a circle labeled in clockwise order. Edges are defined by

$$\{u \to u + m | m \in M\},$$

where M is a fixed set of positive integers less than n and labels are taken modulo n. The game is denoted $(M; n)$.

Fraenkel, Jaffray, Kotzig, and Sabidussi [1] analyzed Kotzig's Nim for games with $M = \{a, a + 1\}$. They rely heavily on the "diamond strategy," in which when one player moves by a, the other player moves by $a + 1$ and vice-versa. They almost completely classified who wins the game with move set $\{3, 4\}$ in terms of the remainder when n is divided by 7. Similarly, Tan and Ward [2] completely analyzed the game with move set $\{1, 4\}$ based on the remainder when n is divided by 5.

Fraenkel et al.[1] proposed that the outcomes of all Kotzig Nim games are based on congruences. Using a theorem by Bodlaender [3], we prove Theorem 1 in Section 3.

Theorem 1. *For a fixed move set M, there exist constants n_0, p such that for all $n \geq n_0$ the outcome of the game $(M; n)$ is the same as the outcome of the game $(M; n + p)$.*

We also study the computational complexities of determining the winning player in variants of Geography. We build upon work by Fox and Geissler [4]. For instance, some variants include the following:

- undirected graph.
- partizan play, where each player moves their own token, tokens are not allowed to simultaneously occupy a vertex, and vertices are deleted once players have moved from them.
- free deletion, where players can delete any vertex/edge they could have come from, not necessarily the one they actually came from.
- edge deletion, where players delete edges instead of vertices.

Each of these variants can be represented by a four-letter string, as outlined in Definition 2. A summary of known complexities of variants from various sources, as well as our own results, is shown in Table 1.

Game	Complexity	Analysis
DIRV	PSPACE-complete, even for bipartite with degree ≤ 3	[5]
DIRE	PSPACE-complete, even for bipartite with degree ≤ 3	[6]
DIFV	PSPACE-complete, even for bipartite	[4]
DIFE	PSPACE-complete, even for bipartite	Section 4
DPRV	PSPACE-complete, even for bipartite with degree ≤ 3	[7]
DPFV	PSPACE-complete in general and for bipartite, NP-hard for DAGs	[4], Section 6
DPFE	PSPACE-complete in general, NP-hard for DAGs	Section 4, Section 6
UIRE	PSPACE-complete in general, but P for bipartite	[6]
UPRV	PSPACE-complete, even for bipartite	[4]
UPFV	PSPACE-complete in general, NP-hard for bipartite	[4], Section 5

Table 1: Summary of Geography Variants

We expand on this table. In Section 4, we modify constructions by Fox and Geissler [4] for vertex games with free deletion to edge games with free deletion to show that they are PSPACE-complete. In Section 5, we show NP-hardness for the previously open problem of undirected, partizan, free deletion Geography on bipartite graphs. In Section 6, we investigate directed acyclic graphs (DAGs). We show that directed, partizan games with free deletion are NP-hard even when restricted to DAGs. However, we completely classify such games on complete DAGs. Finally, in Section 7, we give a polynomial-time algorithm to solve undirected impartial Geography with unbounded stack size and no height rule.

2 Definitions and Methods

When discussing a combinatorial game, we refer to the first player by L and the second player by R.

In Sections 4 to 6, we need to refer to specific variants of Geography. We use the notation of Fox and Geissler[4] with a slight modification.

Definition 2. A variant of Geography is denoted by a four letter string. The letters, in order, are described below.

- Directed (D) or Undirected (U)
- Impartial (I) or Partizan (P)

- Restricted Deletion (R) or Free Deletion (F)
- Vertex Deletion (V) or Edge Deletion (E)

In addition, we may include a subscript $k > 1$ indicating the maximum stack size. In these games, each vertex is labeled with a positive integer at most k, and after a player visits a vertex, its label is decreased by 1. A subscript of ∞ indicates unbounded stack size. Vertices with label 0 are deleted. Games may also have a height rule: if a player is currently at a vertex labeled i, they may only move to a vertex of label at least $i - 1$.

We assume readers are familiar with standard computational complexity classes; we refer to P, NP, and PSPACE throughout this paper.

Note the following.

- A problem's computational complexity may be known in general, but not in specific cases (as specific cases may be solvable more quickly). For example, a Geography variant may be PSPACE-complete over the set of all graphs, but in P for bipartite graphs: see UIRE in Table 1.
- All Geography variants, except for those with subscript ∞, last for a polynomial number of moves in terms of the input size. This means that every sequence of moves can be listed in polynomial space, meaning these variants are in PSPACE. In particular, to prove that these Geography variants are PSPACE-complete, it suffices to show that they are PSPACE-hard.
- It is unknown whether P=NP or P=PSPACE, but both are thought to be very unlikely. Thus, proving NP-hardness or PSPACE-completeness does not refute the existence of a polynomial-time algorithm, but it does provide evidence against it.

It will be helpful to define the True Quantified Boolean Formula problem (TQBF), which is known to be PSPACE-Complete.

Definition 3. True Quantified Boolean Formula asks whether the following logical expression is true:

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots \exists x_{n-1} \forall x_n (c_1 \wedge c_2 \wedge \cdots \wedge c_k),$$

where the x_i are booleans, and each c_i is a clause of the form $\ell_1 \vee \ell_2 \vee \ell_3$ where ℓ_1, ℓ_2, ℓ_3 are literals (expressions of the form x_i or $\overline{x_i}$).

Example 4. For $n = k = 2$, an example of an instance of TQBF is:

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 [(x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_4})].$$

In words, this is "There exists x_1 such that for all x_2 there exists x_3 such that for all x_4, both (x_1 or not x_3 or not x_4 is true) and (not x_1 or x_2 or not x_4 is true)."

Alternatively, TQBF can be interpreted as a game in which the first player selects all odd-indexed x_i while the second selects all even-indexed x_i, and the first player wins if the literal formula is true. Determining the outcome of the game is equivalent to solving TQBF.

Similarly, we also define the following NP-hard problems:

Definition 5. The vertex cover problem asks, given an undirected graph and an integer k, whether it is possible to select k vertices such that every edge contains a selected vertex.

Definition 6. The Hamiltonian path problem asks, given an undirected graph, whether there exists a path containing all vertices exactly once.

Finally, we discuss the main method we use to prove complexities. Suppose we want to prove that evaluating a certain game $WXYZ$ on a certain set of graphs S is at least as computationally difficult as some other problem, say $\mathcal{P}$. Then we consider an arbitrary instance of $\mathcal{P}$. We construct a graph in S such that determining the outcome of $WXYZ$ on the graph exactly corresponds to solving $\mathcal{P}$. We call this *reducing* $\mathcal{P}$ to $WXYZ$ on S.

This reduction shows that if a fast algorithm exists for $WXYZ$ on S, then a fast algorithm exists for $\mathcal{P}$. Taking the contrapositive, since no fast algorithm exists for $\mathcal{P}$, no fast algorithm exists for $WXYZ$ on S.

When constructing graphs for this method, we often refer to *gadgets*. A gadget is simply a piece of our full graph which is designed to perform a certain function. Gadgets will usually be labeled by g_i.

3 Periodicity of Kotzig's Nim

We first prove a general result about Geography. In fact, Bodlaender [3] proved a stronger result in a similar way, but we give our own proof as it more readily leads into a proof that Kotzig's Nim is periodic.

Theorem 7 (Bodlaender [3], 1993)**.** *Let c be a fixed constant. Let G be a graph on n vertices labeled $0, 1, \ldots, n-1$ such that any two adjacent vertices have labels with absolute difference at most c. Suppose S, the starting vertex, has label less than c. Then the winner and winning strategy of $DIRV$ on G can be computed in $O(n)$ time.*

Proof. In order to force a recursion, we generalize the game.

We add *powerups* to the game; we define a powerup as a vertex such that the player who moved to it must make another move in the same turn. There are no powerups in the original game.

Define a state as follows:

- An integer $0 \leq a \leq n$; this represents the set of vertices $\{a, a+1, \ldots, n\}$. Call the vertices $\{a, \ldots, a+c-1\}$ *small*.
- An integer $a \leq s \leq a+c$; this represents the starting vertex.
- An **ordered sequence** $e_1, \ldots, e_k$ of distinct ordered pairs of vertices with labeles at most c apart, each of which contains at least one small vertex; this represents all moves going to or from small vertices. Note that these pairs do **not** have to be valid edges in the graph. This is because later in the proof, we will contract paths into edges which may not be present in the graph.

- A subset P of the directed edges above; this represents which edges are powerups and which are not.

- A boolean; this represents the game mode. In normal play, the player who first runs out of moves loses, while in misere play, the player who first runs out of moves wins.

Since c is fixed, the number of states is linear in n. It now suffices to show that we can recurse on these states to determine all of their outcomes.

If $a \geq n - 2c + 1$, simply check all possible sequences of moves in the game, of which there are at most $(2c)!$. This does not affect the time complexity.

Otherwise, call the vertices $\{a + c, \ldots, a + 2c - 1\}$ medium. In our game, we must have the following chains of moves in order, where the m_i are medium vertices, the v_i are small vertices, l is a positive integer, and $1 \leq i_0 < i_2 < \cdots < i_l \leq m$:

$$\text{start} \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \ldots \xrightarrow{e_{i_1}} m_1$$

$$m_2 \xrightarrow{e_{i_1+1}} v_{i_1+1} \xrightarrow{e_{i_1+2}} v_{i_1+2} \ldots \xrightarrow{e_{i_2}} m_3$$

$$m_4 \xrightarrow{e_{i_2+1}} v_{i_1+1} \xrightarrow{e_{i_2+2}} v_{i_2+2} \ldots \xrightarrow{e_{i_3}} m_5$$

$$\vdots$$

$$m_{2l} \xrightarrow{e_{i_{l-1}+1}} v_{i_{l-1}+1} \xrightarrow{e_{i_{l-1}+1}} v_{i_{l-1}+1} \ldots \xrightarrow{e_{i_l}} m_{2l+1}.$$

If the game ends on a small vertex, there will also be another path:

$$m_{2l+2} \xrightarrow{e_{i_l+1}} v_{i_l+1} \xrightarrow{e_{i_l+2}} v_{i_l+2} \ldots \xrightarrow{e_l} v_l.$$

Now consider any valid game in the current state. We claim that we can simplify it to a smaller game of some computed state.

- Set m_1 to be the starting vertex in a game with vertices $\{a + c, \ldots, n - 1\}$.

- Collapse each path from m_{2i} to m_{2i+1} into a directed edge from m_{2i} to m_{2i+1}. To decide whether or not this edge will be a powerup, consider the number of non-powerup edges along the original path. If it is even, make the edge a powerup, and otherwise, make the edge normal.

- Also add the sequence of edges from $\{m_i\}$ to or from the set $\{a + 2c, \ldots, n - 1\}$ (each edge will not be a powerup). These edges must be actual edges on our graph, and they must be consistent with the original game: m_1 must be precede edge m_2m_3, which must precede edge m_4m_5, and so on until m_{2l+2} (if it exists).

- To decide the game mode,

  - If the original game does not end on a small vertex, set the new game mode to be the same as the original game mode.

– If the original game ends on a small vertex and there are an even number of edges on the path from m_{2l+2} to v_l, set the new game mode to be the same as the original game mode.
– If the original game ends on a small vertex and there are an odd number of edges on the path from m_{2l+2} to v_l, set the new game mode to be different from the original game mode.

All of these steps take a constant number of operations in terms of c. Moreover, it is easy to check that the smaller game will be equivalent to the original game.

In this way, we can iterate through all possible games in constant time. Thus we can determine the winner and winning position in constant time. Recursing now gives an $O(n)$ algorithm. □

For any graph G, define $f(G)$ to be the set of pairs (state, outcome of state) over all possible states on the graph G. Notice that in our proof of Theorem 7, the recursive calculation of $f(G)$ only depends on the edges among the first $2c$ vertices of G. This gives the following corollary.

Corollary 8. *Let G_1 and G_2 be arbitrary directed graphs whose vertices are labeled in such a way that any two adjacent vertices have labels with absolute difference at most c. Suppose the first $2c$ vertices of G_1 have the same edges among them as the first $2c$ vertices of G_2. Let G_1' be G_1 with its first c vertices removed, and let G_2' be G_2 with its first c vertices removed. If $f(G_1') = f(G_2')$, then $f(G_1) = f(G_2)$.*

The same is true by symmetry if we define G_1' to be G_1 with its last c vertices removed and G_2' to be G_2 with its last c vertices.

Now we are ready to prove the main result.

Proof of Theorem 1. Let K denote the largest absolute value of a move in M. Consider the Kotzig Nim game for some $n \gg K$. Relabel the vertices clockwise by

$$0, 2, 4, \ldots, n-1, n-2, n-4, \ldots, 3, 1 \text{ if } n \text{ is odd},$$

$$0, 2, 4, \ldots, n-2, n-1, n-3, \ldots, 3, 1 \text{ if } n \text{ is even}.$$

Draw directed edges based on which moves are allowed, and call the resulting graph G_n. Note that any two adjacent vertices in G_n have labels with absolute difference at most $2K$; thus, $2K$ functions as c did before.

Also define a graph H_n which is G_n but with the vertices $0, \ldots, 2K-1$ and the vertices $n-2K, \ldots, n-1$ removed. Note that there is an edge $u \to v$ in H_n if and only if $\frac{v-u}{2} \in M$. In particular, every set of $2K$ consecutive vertices in H_n has the same edges among it.

Since f takes only finitely many values, by Pigeonhole there must exist large $i \neq j$ such that $f(H_i) = f(H_j)$. Thus, by Corollary 8, for all sufficiently large n, we have $f(H_n) = f(H_{n+j-i})$.

But notice that the vertices $0, \ldots, 2K-1$ have the same edges among them for all large enough n, and similarly for the vertices $n-2K, \ldots, n-1$. So by Corollary 8 again, we have that $f(G_n) = f(G_{n+j-i})$. □

4 Free Edge Deletion Variants

We begin by investigating variants with edge deletion, which were not studied by Fox and Geissler.

Theorem 9. *The Geography variant DIFE is PSPACE-Complete, even for bipartite graphs.*

Proof. Let Q be an instance of TQBF, as in Definition 3; we reduce Q to DIFE. We use a similar construction as that of Theorem 1 in [4], as shown in Figure 1.

- The token starts at T and moves through a series of gadgets $g_1, \ldots, g_n$ corresponding to the variables $x_1, \ldots, x_n$ respectively. For all $i \leq n-1$, the bottom vertex of g_i has an outgoing edge (called a joining edge) to the top vertex of g_{i+1}
- The bottom vertex of g_n has an outgoing edge to a vertex B. This vertex has an outgoing edge to vertices $c_1, \ldots, c_m$, each corresponding to a clause in the TQBF expression.
- Each clause vertex c_i has an outgoing edge directly to vertices representing all odd-indexed variable literals in the clause, and an outgoing path of length 2 to vertices representing all even-indexed variable literals in the clause.
- A vertex representing x_i points to
  - The bottom left vertex of g_i, if i is odd.
  - The right vertex of g_i, if i is even.

  A vertex representing $\overline{x_i}$ points to
  - The bottom right vertex of g_i, if i is odd.
  - The left vertex of g_i, if i is even.

Note that all joining edges must be deleted. Also note that L will always be the one moving to the bottom of an odd-indexed variable gadget, while R will always be the one moving to the bottom of an even-indexed variable gadget.

Suppose L ever deletes an edge from a literal vertex to an even-indexed variable gadget. Then there are an even number of vertices on the path to the literal vertex, so R is guaranteed to win if the game moves to the literal vertex (whereas before he wasn't guaranteed to win this branch). Therefore, it is never optimal for L to do this. Similarly, it is never optimal for R to delete an edge from a literal vertex to an odd-indexed variable gadget.

Now assume the expression Q is true. Then, a strategy for L is the following.

- After moving to the bottom of an odd-indexed variable gadget, delete the left edge if the variable should be set to false, and delete the right edge if the variable should be set to true.

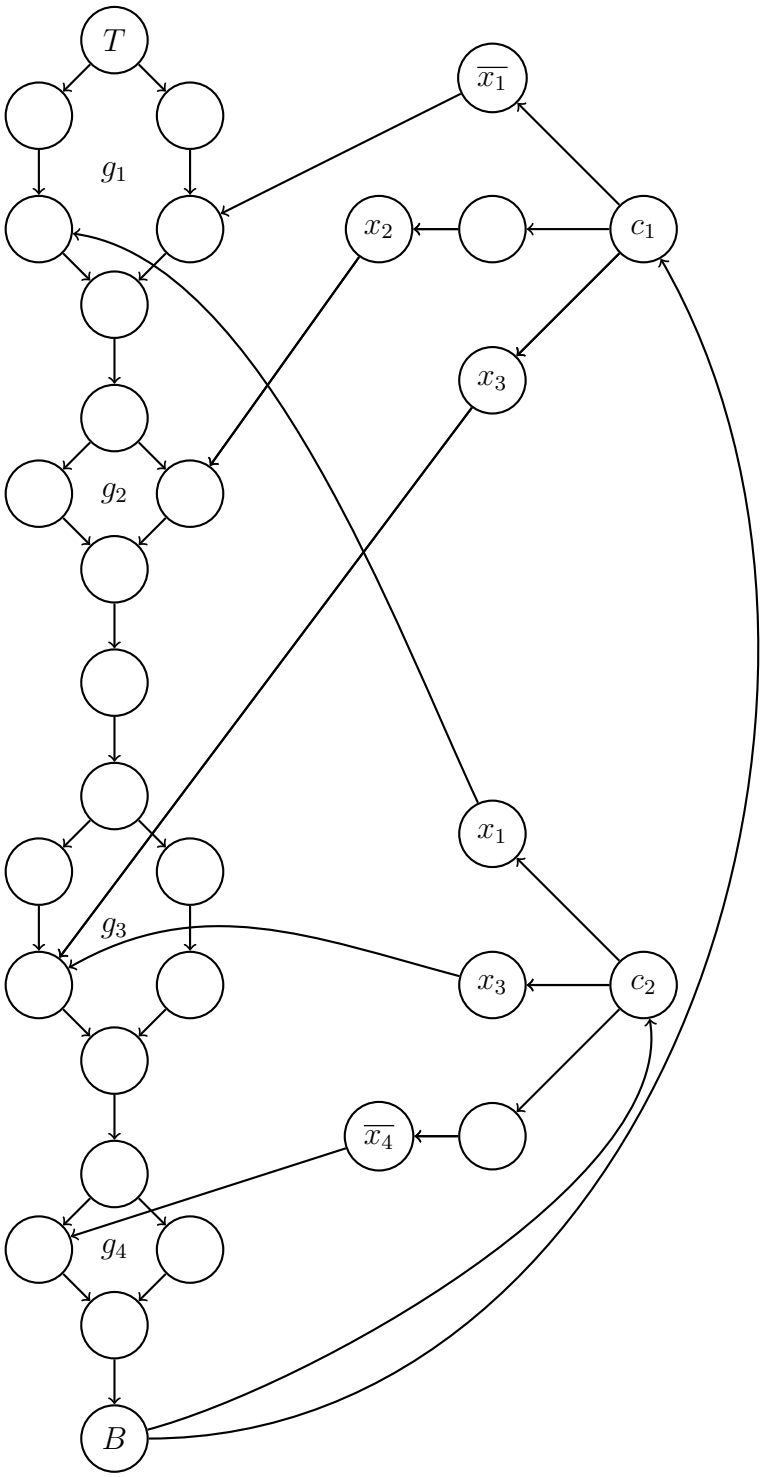

Figure 1: Reduction of $\exists x_1 \forall x_2 \exists x_3 \forall x_4 (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee \overline{x_4})$ to DIFE; modified from Figure 4 of [4]

- Once all variable gadgets have been traversed, no matter which clause vertex R moves to, there will exist some branch corresponding to a true literal u. Let v be the successor of u.

- If u contains an odd-indexed variable, then L can move to u. After this, R is forced to move to v. Next, since the literal is true, there is still an edge from v to the bottom vertex of the variable gadget, so L can take this. Finally, the joining edge must have been deleted, so R has no moves.

- If u contains an even-indexed variable, then L can force R to move to u. After this, L can move to v. Next, since the literal is true, the edge from v to the joining vertex of the variable gadget has been deleted. Thus, R has no moves.

On the other hand, assume Q is false. Then, a strategy for R is the following:

- After moving to the bottom of an even-indexed variable gadget, delete the left edge if the variable should be set to false, and delete the right edge if the variable should be set to true.

- Once all variable gadgets have been traversed, choose to move to a clause vertex leading to no true literals. Suppose L chooses to go on the path to a literal u. Let v be the successor of u.

- If u contains an odd-indexed variable, then R can move to v. Since the literal is false, there is no edge from v to the bottom vertex of the variable gadget, so L has no moves.

- If u contains an even-indexed variable, then R can move to u, from which L will move to v. Since the literal is false, there is an edge from v to the bottom vertex of the variable gadget, so R can take this edge. Finally, the joining edge must have been deleted, so L has no moves.

We have shown that L wins if and only if Q is true. Moreover, our graph is a subgraph of Figure 4 of [4], which was shown to be bipartite; thus, ours is bipartite as well. Hence TQBF can be reduced to an instance of DIFE on a bipartite graph in polynomial time, as desired. □

For the partizan version of this game, we can prove PSPACE-completeness in a very similar way.

Theorem 10. *The Geography variant DPFE is PSPACE-Complete.*

Proof. Let Q be an instance of TQBF, as in Definition 3; we reduce Q to DPFE. We use a similar construction as that of Theorem 2 in [4], as shown in Figure 2.

- Players L and R start by simultaneously going through odd-indexed and even-indexed gadgets g_i respectively. For all $i \leq n-2$, the bottom vertex of g_i has an outgoing edge to g_{i+2}.

- After her variable gadgets, L has a path of $3m$ delay vertices $DLY_1 \to \cdots \to DLY_{3m}$.
- After his variable gadgets, R has set of "delete" vertices $DLT_1, \ldots, DLT_{2m-2}$. Note that these do not form a full path; instead, DLT_{2i} has an outgoing edge to DLT_{2i+1} for all i.
- There are also vertices $c_1, \ldots, c_m$ representing clauses. Each clause vertex has incoming edges from DLY_{3m} as well as all odd-indexed delete vertices, and outgoing edges to all even-indexed delete vertices.
- Each clause vertex has additional outgoing edges to three linking vertices (labeled LNK).
- Each linking vertex has outgoing edges to two intermediate vertices representing the corresponding literal.
- A vertex representing x_i has an outgoing edge to the right vertex of g_i, and a vertex representing $\overline{x_i}$ has an outgoing edge to the left vertex of g_i.
- Finally, DLT_{2m-2} connects to a path of escape vertices $ESC_1 \to \cdots \to ESC_7$.

Clearly, the players will start the game by taking turns choosing whether their variables are true or false. Call this the "selection phase." After this, notice the following:

- Suppose R moves to DLT_{2m-2} before moving through all of $DLT_1, \ldots, DLT_{2m-3}$. Then L has strictly more options than if R had not done this, and R has strictly fewer moves remaining than if he had not done this. Thus, it is never optimal for R to do this.
- If R moves to a linking vertex at any point, then he must have moved from some c_i. However, R can make at most $3(m-2)+1$ moves to and from $c_1, \ldots, c_m$ before being forced to escape. Therefore, it is at most $3m-5$ turns after the selection phase. This means L has at least 5 more delay vertices left to traverse, whereas R has at most 4 moves left from the linking vertex. Again, this is losing for R.

Thus, R should take the following path after the selection phase:

$$DLT_1 \to c_{i_1} \to DLT_2 \to DLT_3 \to c_{i_2} \to \cdots \to DLT_{2m-4} \to DLT_{2m-3} \to c_{i_{m-1}} \to DLT_{2m-2}.$$

During these moves, he has the opportunity to delete the edges from DLY_{2m-3} to all but one of $c_1, \ldots, c_m$. Call this the "deletion phase".

Exactly 3 moves after the deletion phase, L must be at the only remaining c_i while R is at ESC_3.

- If L moves to DLT_i for even i, then she immediately loses because the edge from DLT_i to DLT_{i+1} has already been deleted by R.
- If L moves to ESC_1, then she also immediately loses because the edge from ESC_1 to ESC_2 has already been deleted by R.

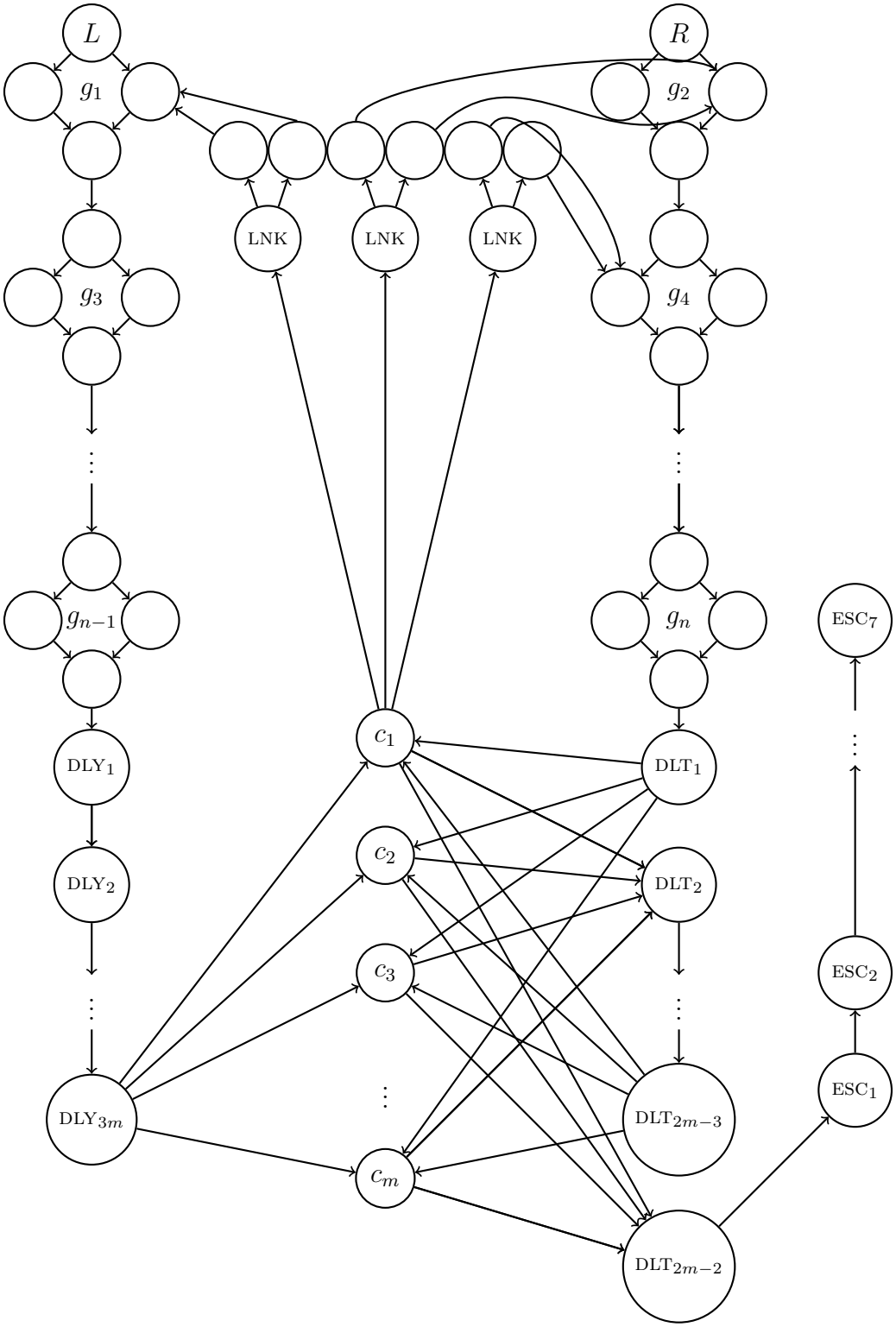

Figure 2: Reduction of $\exists x_1 \forall x_2 \ldots \exists x_{n-1} \forall x_n (x_1 \vee x_2 \vee \overline{x_4}) \wedge c_2 \wedge \cdots \wedge c_m$ to DPFE; modified from Figure 6 of [4]

So L is forced to move into a linking vertex.

Now suppose Q is true. Then, L should be able to choose a linking vertex so that she has 3 moves left, thus outlasting R.

On the other hand, suppose Q is false. Then, R can force all of the linking vertices to represent false literals, meaning L has at most 2 moves left. In this case, R will win.

Thus, L wins if and only if Q is true, so we have reduced TQBF to DPFE. □

5 UPFV On Bipartite Graphs

The game UPFV was proven to be PSPACE-complete for general graphs by Fox and Geissler [4], but the time complexity is unknown for any smaller class of graphs. We show that the problem is NP-hard for bipartite graphs and planar graphs of maximum degree 3, which suggests that a polynomial-time algorithm probably does not exist for these cases.

Theorem 11. *UPFV is NP-Hard for bipartite graphs as well as planar graphs of maximum degree* 3*.*

Proof. Consider the Hamiltonian Path problem (which is known to be NP-hard) on an arbitrary undirected graph G with n vertices, m edges, a given starting vertex s, and a given ending vertex e. Assume WLOG that s and e have degree 1, and that all other vertices have degree greater than 1.

We claim that this problem reduces to the following game of UPFV.

- Draw G, and along each edge segment of G, add five "intermediate" vertices.
- Add a separate path $s - p_1 - p_2 - \cdots - p_{2n+4m-1}$.
- Let L start at p_1 and R start at s.

Player L is forced to start by moving to p_2 and deleting p_1. After this, she is forced to keep moving along the path p_i, so she has exactly $2n + 4m - 3$ more moves.

Meanwhile, R must play in the following way:

- If he is currently at a vertex v of G, he must move to an intermediate vertex along some edge of G. If he chooses to delete v, then he is forced to keep going along the edge. If he chooses instead to delete the next intermediate vertex, then he is forced to go back to v (call these back-and-forth moves a "stall").
- If he ever chooses to go on an edge to a vertex that has already been deleted, then he has at most 5 more moves before he is stuck. Therefore, he will always choose to go on an edge to a non-deleted vertex, which gives him at least 6 more moves.
- If he is entering a vertex v of G, then it is always optimal for him to delete the intermediate vertex he came from (it is a dead end, so he has no reason to ever go back to it).

Thus, if there is a Hamiltonian path, R can make at most the following moves.

- $6(n-1)$ moves, for the actual edges on the path
- $2(\deg v - 2)$ stalls at each vertex $v \neq s, e$

This gives a total of

$$6(n-1) + \left(\sum_{v \neq s,e} 2(\deg v - 2) \right) = 6n - 6 + 2(2m-2) - 4(n-2) = 2n + 4m - 2.$$

Therefore, R will have more moves than L if and only if a Hamiltonian path exists, so we have shown that the Hamiltonian path problem reduces to UPFV.

It is easy to see that the UPFV graph we constructed has no odd cycles, so it is bipartite. This shows that UPFV is NP-hard on bipartite graphs.

Moreover, if G is planar with maximum degree 3, then so is the graph we constructed. Since it is known that the Hamiltonian path problem is NP-hard even with this restriction, UPFV is as well. □

6 Directed Acyclic Graphs

We next consider partizan games on directed acyclic graphs (DAGs). Fraenkel and Simonson [7] showed that DPRV and DPRE are NP-hard on DAGs by reducing from the vertex cover problem. We prove the same for the case of free deletion.

Theorem 12. *DPFV is NP-Hard for DAGs.*

Proof. Consider the problem of determining whether a vertex cover of size k exists in a given undirected graph G on vertices $v_1, \ldots, v_n$ and m edges. We claim that it reduces to DPFV on the following DAG. An example of a reduction from the vertex cover problem in Figure 3 to DPFV on a DAG is shown in Figure 4.

- A grid of vertices $\{u_{i,j}\}_{1 \leq i,j \leq n}$ such that there is a directed edge from u_{ij} to $u_{i,j+1}$ for all $j < n$, and there is a directed edge from $u_{a,n}$ to $u_{b,1}$ for all $a < b$. Each row of this grid represents a vertex of G.
- A set of m vertices $e_1, \ldots, e_m$, each representing an edge of G, where the vertex representing an edge $v_i v_j$ has an outgoing edge to both $u_{i,j}$ and $u_{j,i}$.
- A path of delay vertices $DLY_1 \to DLY_2 \to \cdots \to DLY_{nk}$, where DLY_{nk} has an outgoing edge to each of $e_1, \ldots, e_m$.
- A path of escape vertices $ESC_1 \to ESC_2 \to \cdots \to ESC_{n^2}$, where there are outgoing edges from each of $e_1, \ldots, e_m$ to ESC_1.
- A vertex x, which is L's starting point, whose only edge is an outgoing edge to DLY_1.

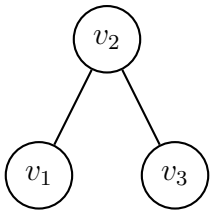

Figure 3: Example of vertex cover problem with $n = 3$, $m = 2$, $k = 1$

- A vertex y, which is R's starting point, whose only edges are outgoing edges to each of $u_{1,1}, u_{2,1}, \ldots, u_{n,1}$.

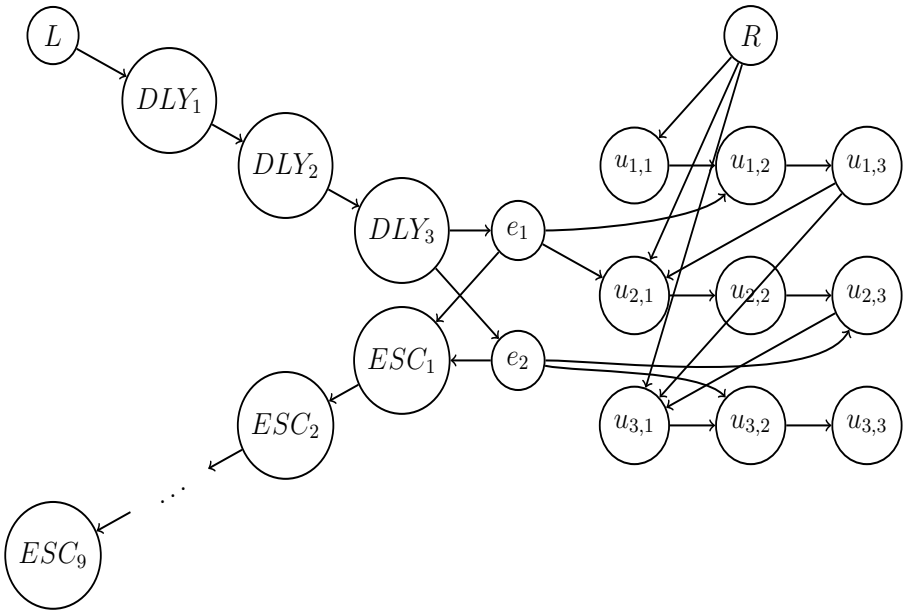

Figure 4: Reduction of vertex cover problem in Figure 3 to DPFV on a DAG

After nk turns, player L will be at DLY_{nk}, while player R will have fully traversed some k rows of the grid.

- If a vertex cover of size k exists, then R could have deleted $e_1, \ldots, e_m$ by this time, thus trapping player L.
- Otherwise, at least one of $e_1, \ldots, e_m$ remains, so L can go to it and reach the escape path. This gives her at least n^2 moves, which is more than R has.

Thus, determining the winner of this game is equivalent to finding a vertex cover, so DPFV is NP-Hard for DAGs. □

The game DPFE seems easier to analyze, because each deletion does not change the structure of the graph as much. However, a similar construction shows that it, too, is NP-hard for DAGs.

Theorem 13. *DPFE is NP-Hard for DAGs.*

Proof. Again, consider the problem of determining whether a vertex cover of size k exists in a given undirected graph G on vertices $v_1, \ldots, v_n$ and m edges. We construct a DPFE game on a DAG which is equivalent to this problem. An example of reducing the vertex cover problem in Figure 3 to DPFE on a DAG is shown in Figure 5.

- A grid of vertices $\{u_{i,j}\}_{1 \le i \le n, 1 \le j \le n^2}$ such that there is a directed edge from u_{ij} to $u_{i,j+1}$ for all $j < n$, and there is a directed edge from u_{a,n^2} to $u_{b,1}$ for all $a < b$. Again, each row of this grid represents a vertex of G.
- A set of m vertices $e_1, \ldots, e_m$, each representing an edge of G, where the vertex representing an edge $v_i v_j$ has an outgoing edge to both u_{i,n^2-j} and u_{j,n^2-i}.
- A path of delay vertices $DLY_1 \to DLY_2 \to \cdots \to DLY_{n^2k+m}$, where DLY_{n^2k+m} has an outgoing edge to each of $e_1, \ldots, e_m$.
- A set of m mirror vertices $e'_1, \ldots, e'_m$, each representing an edge, where the vertex representing an edge $v_i v_j$ has an *incoming* edge from both u_{i,n^2-j} and u_{j,n^2-i}. Also, e'_1 has an incoming edge from each of $u_{1,n^2}, \ldots, u_{n,n^2}$, and there is a path $e'_1 \to e'_2 \to \cdots \to e'_m$.
- For each e'_i, two disjoint escape paths of $2n^3$ vertices starting at e'_i.
- A vertex x, which is L's starting point, whose only edge is an outgoing edge to DLY_1.
- A vertex y, which is R's starting point, whose only edges are outgoing edges to each of $u_{1,1}, \ldots, u_{n,1}$.

Suppose a vertex cover of size k exists. Let S be the set of $u_{i,j}$ such that v_i is in the vertex cover, and let S' be the set of $u_{i,j}$ such that v_i is not in the vertex cover. Then, R should do the following:

- Traverse S, and delete the respective edges from e_i.
- Traverse the path $e'_1 \to e'_2 \to \cdots \to e'_m$, and at each step, delete the edge (if any) from S' to e'_i.

This takes $n^2k + m$ turns, at which point L will be at DLY_{n^2k+m}. Now L will move to some e_i, from which she may be able to move to a vertex in S'. When she is in S', she cannot move to any e'_i, and if she ever moves to the first vertex of a row in S, she will be immediately stuck. In particular, L has at most n^3 more moves, which is less than R has from his escape path.

On the other hand, suppose there is no vertex cover of size k. We have two cases.

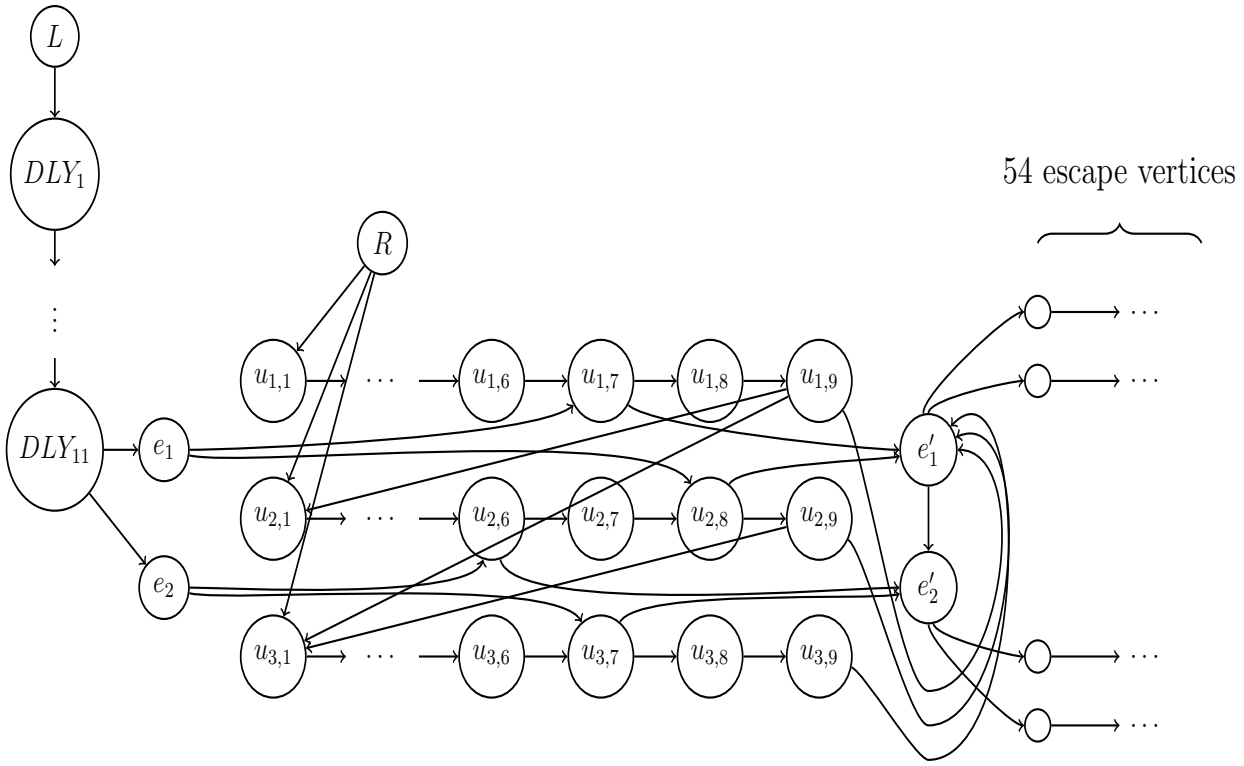

Figure 5: Reduction of vertex cover problem in Figure 3 to DPFE on a DAG

- If R traverses k or fewer rows before moving to e'_1, he will reach e'_m in at most n^2k+m turns and be forced to take one of the escape paths. Meanwhile, L will reach DLY_{n^2k+m} after n^2k+m turns. Since R did not make a vertex cover, L will now have a path to some e'_i, from which she can take an escape path. Since L starts her escape path after R starts his, she wins.

- If R traverses k rows and then chooses to enter another row, then after n^2k+m turns, he will still be traversing the row (since $m < n^2$). Meanwhile, L can simply go to an edge that connects to near the end of the row, blocking R's path. Now R has at most n^2 moves left, while L can eventually reach an escape path, so L wins.

We have shown that R wins if and only if there is a vertex cover of size k in the original graph, so we are done. □

As suggested by Fraenkel and Simonson [7], we fully classify DPFV and DPFE on complete DAGs.

Theorem 14. *Suppose L and R are playing DPFV on a complete DAG on n vertices, labeled $1, 2, \ldots, n$ in topological order. If A starts on i and B starts on j, then A wins if and only if one of the following is true:*

- $i \equiv n-1 \pmod 3$ *and* $j = i-1$

- $i \equiv n-1 \pmod 3$ *and* $j \geq i+2$
- $i \not\equiv n-1 \pmod 3$ *and* $j > i$.

Proof. Induct on n; the base cases $n = 2, 3, 4$ can be checked manually.

We have four cases:

- If both i and j are in $\{n-2, n-1, n\}$, then the condition can be checked manually.
- If i is in $\{n-2, n-1, n\}$ but j isn't, then
  - If L starts by moving to $n-1$, then R moves to n and wins.
  - If L starts by moving to n, then R moves to either $n-2$ or $n-1$ (it is impossible that both have been deleted) and wins.
- If j is $\{n-2, n-1, n\}$ but i isn't, then
  - If $j = n-2$, then L moves to n and deletes $n-1$ to win.
  - If $j = n-1$, then L moves to n and deletes $n-2$ to win.
  - If $j = n$, then L moves to $n-1$ and wins.
- If neither i nor j is in $\{n-2, n-1, n\}$, then the player who wins the game on $n-3$ vertices should adopt the same strategy as for that game. This will force the other player to move to $\{n-2, n-1, n\}$, from which they will lose according to the analysis in the previous case.

All these cases are consistent with our claimed set of positions, so we are done. □

Theorem 15. *Suppose L and R are playing DPFE on a complete DAG on n vertices, labeled $1, 2, \ldots, n$ in topological order. If L starts on i and R starts on j, then L wins if and only if one of the following is true:*

- $i \equiv n \pmod 2$ *and* $j = i+1$
- $j \geq i+2$

Proof. Induct on n; the base cases $n = 2, 3$ can be checked manually.

We have four cases:

- If both i and j are in $\{n-1, n\}$, then L doesn't have any moves, so R wins.
- If i is in $\{n-1, n\}$ but j isn't, then L has at most one more move, while R has at least one more move (to $n-1$). So R wins.
- If j is in $\{n-1, n\}$ but i isn't, then L moves to whichever of $n-1$ and n is unoccupied. Now R has no moves, so L wins.

- If neither i nor j is in $\{n-1, n\}$, then the player who wins the game on $n-2$ vertices should adopt the same strategy as for that game. This will force the other player to move to one of $n-1$ and n, from which they will lose according to the analysis in the previous case.

All these cases are consistent with our claimed set of positions, as desired. □

7 Stacks

Finally, we study stacking variants. Without a height rule, when stack size is bounded by a constant, Fox and Geissler [4] showed that we can simply replace each vertex of height k with k vertices of height 1, and the game now has no stacks.

When stack size is unbounded, this cannot be done, because it takes exponential time in the number of bits necessary to represent stack sizes to create the required number of vertices. However, we show that for $UIRV_\infty$ on trees, the decision problem is still solvable in polynomial time.

We need the following lemma.

Lemma 16. *Consider the game $UIRV_\infty$ without a height rule. For any given tree, there exists a positive integer threshold c such that*

- *If the starting vertex's height is replaced by any integer at least c, then the game is a P-position.*
- *If the starting vertex's height is replaced by any integer less than c, then the game is an N-position.*

Proof. Root the tree at the starting vertex, and induct from the bottom layer upwards. For the base cases (leaves), just take $c = 0$.

Now consider an arbitrary subtree. Let $h_1, \ldots, h_d$ denote the actual heights of the children of the root, and let $c_1, \ldots, c_d$ be their respective thresholds given by the inductive hypothesis. We claim that the following threshold works:

$$c = 1 + \sum_{i=1}^{d} \max(h_i - c_i + 1, 0).$$

Suppose the root's height is replaced by $h \geq c$. Player R should use the following strategy:

- If L moved from the root to a child with $h_i < c_i$, player R can play the game as if it is the subtree of the child to win.
- If L moved from the root to a child with $h_i \geq c_i$, player R can move to the root.

Since the second scenario can occur at most $c - 1$ times, the first scenario must occur eventually, so R will win.

On the other hand, suppose the root's height is replaced by $h < c$. Suppose that whenever player L is in a subtree of a child of the root, she follows the same strategy as he would if only that subtree existed. Then, we may assume WLOG that all moves to or from the root happened at the beginning of the game; all other moves can be viewed as part of games within subtrees of children of the root. Because of the height condition on the root, L can ensure that whenever she is on the root, she moves to a child with $h_i \geq c_i$. Thus L will always end up in a subtree where she wins. □

Theorem 17. *The Geography variant $UIRV_\infty$ without a height rule is in the complexity class P for trees.*

Proof. Find the threshold value of the root by recursing upwards as described in the above proof. Each subtree calculation takes $O(n)$ time, for a total of $O(n^2)$ time. Once the threshold is known, just compare it to the actual height of the root to determine the outcome of the game. □

8 Conclusion

We resolved a long-standing open problem by showing that Kotzig's Nim is eventually periodic. Moreover, we furthered the analysis of computational complexities of variants of Geography with free deletion, edge deletion, and stacks. We found that even in graphs with very restricted structures, like bipartite graphs, planar graphs, and directed acyclic graphs, determining the winner of some Geography variants is computationally difficult.

Many problems remain open. In particular, the complexities of UIR_∞ with a height rule and UIR_3, as posed by Fox and Geissler [4], are still unknown. Furthermore, any of the variants we studied can be analyzed on even more narrow classes of graphs such as grid graphs and k-partite graphs for $k > 2$.

9 Acknowledgements

Thank you to my mentor Joshua Messing for suggesting directions for research, evaluating my proofs, and providing advice on writing this paper. Thank you to head math coordinator Tanya Khovanova and my tutor Peter Gaydarov for giving feedback on my work throughout the research process. Thank you to Professor David Jerison and my TAs Alec DeWulf and Allen Lin for making suggestions to improve my final paper and presentation. Thanks also to RSI, CEE, and MIT for organizing and funding this invaluable experience.

References

[1] A. S. Fraenkel, A. Jaffray, A. Kotzig, and G. Sabidussi. Modular nim. *Theoretical computer science*, 143(2):319–333, 1995.

[2] X. L. Tan and M. D. Ward. On kotzig's nim. *Integers*, 14:G6, 2014.

[3] H. L. Bodlaender. Complexity of path-forming games. *Theoretical Computer Science*, 110(1):215–245, 1993.

[4] N. Fox and C. Geissler. On the computational complexities of various geography variants. *CoRR*, abs/2108.09367, 2021.

[5] D. Lichtenstein and M. Sipser. Go is polynomial-space hard. *J. ACM*, 27(2):393–401, apr 1980.

[6] A. S. Fraenkel, E. R. Scheinerman, and D. Ullman. Undirected edge geography. *Theoretical Computer Science*, 112(2):371–381, 1993.

[7] A. Fraenkel and S. Simonson. Geography. *Theor. Comput. Sci.*, 110:197–214, 03 1993.