Nash Equilibria in Graphical Games on Trees Revisited *

Edith Elkind
University of Warwick
Coventry, CV4 7AL, U.K.

Leslie Ann Goldberg
University of Warwick
Coventry, CV4 7AL, U.K.

Paul Goldberg
University of Warwick
Coventry, CV4 7AL, U.K.

ABSTRACT

Graphical games have been proposed as a game-theoretic model of large-scale distributed networks of non-cooperative agents. When the number of players is large, and the underlying graph has low degree, they provide a concise way to represent the players' payoffs. It has recently been shown that the problem of finding Nash equilibria in a general degree-3 graphical game with two actions per player is complete for the complexity class PPAD, indicating that it is unlikely that there is any polynomial-time algorithm for this problem. In this paper, we study the complexity of graphical games with two actions per player on bounded-degree trees. This setting was first considered by Kearns, Littman and Singh, who proposed a dynamic programming-based algorithm that computes all Nash equilibria of such games. The running time of their algorithm is exponential, though approximate equilibria can be computed efficiently. Later, Littman, Kearns and Singh proposed a modification to this algorithm that can find a single Nash equilibrium in polynomial time. We show that this modified algorithm is incorrect — the output is not always a Nash equilibrium. We then propose a new algorithm that is based on the ideas of Kearns et al. and computes all Nash equilibria in quadratic time if the input graph is a path, and in polynomial time if it is an arbitrary graph of maximum degree 2. Moreover, our algorithm can be used to compute Nash equilibria of graphical games on arbitrary trees, but the running time can be exponential, even when the tree has bounded degree. We show that this is inevitable — any algorithm of this type will take exponential time, even on bounded-degree trees with pathwidth 2. It is an open question whether our algorithm runs in polynomial time on graphs with pathwidth 1, but we show that finding a Nash equilibrium for a 2-action graphical game in which the underlying graph has maximum degree 3 and constant pathwidth is PPAD-complete (so is unlikely to be tractable).

*This research is supported by the EPSRC research grants "Algorithmics of Network-sharing Games" and "Discontinuous Behaviour in the Complexity of randomized Algorithms".

EC'06, June 11–15, 2006, Ann Arbor, Michigan, USA.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*economics*

General Terms

Algorithms, Economics, Theory

Keywords

Graphical games, Nash equilibrium, PPAD-completeness

1. INTRODUCTION

Graphical games were introduced in the papers of Kearns et al. [8] and Littman et al. [9] as a succinct representation of games with a large number of players. The classical *normal form* (or matrix form) representation has a size that is exponential in the number of players, making it unsuitable for large-scale distributed games. A graphical game associates each player with a vertex of an underlying graph G, and the payoff to that player is a function of the actions chosen by himself and his neighbours in G; if G has low degree, this is a concise way to represent a game with many players.

The papers [8, 9] give a dynamic-programming algorithm for finding Nash equilibria in graphical games where there are two actions per player and G is a tree. The first of these papers describes a generic algorithm for this problem that can be specialized in two ways: as an algorithm that computes approximations to all Nash equilibria in time polynomial in the input size and the approximation quality, or as an exponential-time algorithm that allows the exact computation of all Nash equilibria in G. In [9], the authors propose a modification to the latter algorithm that aims to find a single Nash equilibrium in polynomial time. This does not quite work, as we show in Section 3, though it introduces a useful idea.

1.1 Background

The generic algorithm of [8] consists of two phases which we will refer to as the upstream pass and the downstream pass; [1] the former starts at the leaves of the tree and ends at the root, while the latter starts at the root and ends at the leaves. It is assumed that each player has two pure strategies (actions), which are denoted by 0 and 1; it follows that any mixed strategy can be represented as a single number $x \in [0, 1]$, where x is the probability that the player selects 1. During the upstream pass, each vertex V computes the set of its *potential best responses* to every mixed strategy w of its parent W; a strategy v is a potential best response to w if

[1]Note that the terminology "upstream" and "downstream" are reversed in [8, 9] — our trees are rooted at the top.

there is a Nash equilibrium in the graphical game downstream of V (inclusive) given that W plays w (for a more technical definition, the reader is referred to Section 2). The output of this stage can be viewed as a (continuous) table $T(w, v)$, where $T(w, v) = 1$ if and only if v is a potential best response to w; we refer to this table as the *best response policy* for V. The generic algorithm does not address the problem of representing the best response policy; in fact, the most important difference between the two instantiations of the generic algorithm described in [8] is in their approach to this issue. The computation is performed inductively: the best response policy for V is computed based on the best response policies of V's children $U_1, \ldots, U_k$. By the end of the upstream pass, all children of the root have computed their best response policies.

In the beginning of the downstream pass, the root selects its strategy and informs its children about its choice. It also selects a strategy for each child. A necessary and sufficient condition for the algorithm to proceed is that the strategy of the root is a best response to the strategies of its children and, for each child, the chosen strategy is one of the pre-computed potential best responses to the chosen strategy of the root. The equilibrium then propagates downstream, with each vertex selecting its children's actions. The action of the child is chosen to be any strategy from the pre-computed potential best responses to the chosen strategy of the parent.

To bound the running time of this algorithm, the paper [8] shows that any best response policy can be represented as a union of an exponential number of rectangles; the polynomial time approximation algorithm is obtained by combining this representation with a polynomial-sized grid. The main idea of [9] is that it is not necessary to keep track of all rectangles in the best response policies; rather, at each step of the upstream pass, it is possible to select a polynomial-size subset of the corresponding policy (in [9], this subset is called a *breakpoint policy*), and still ensure that the downstream pass can proceed successfully (a sufficient condition for this is that the subset of the best response policy for V stored by the algorithm contains a continuous path from $w = 0$ to $w = 1$).

1.2 Our Results

One of the main contributions of our paper is to show that the algorithm proposed by [9] is incorrect. In Section 3 we describe a simple example for which the algorithm of [9] outputs a vector of strategies that *does not* constitute a Nash equilibrium of the underlying game.

In Sections 4, 5 and 6 we show how to fix the algorithm of [9] so that it always produces correct output.

Section 4 considers the case in which the underlying graph is a path of length n. For this case, we show that the number of rectangles in each of the best response policies is $O(n^2)$. This gives us an $O(n^3)$ algorithm for finding a Nash equilibrium, and for computing a representation of all Nash equilibria. (This algorithm is a special case of the generic algorithm of [8] — we show that it runs in polynomial time when the underlying graph is a path.)

We can improve the running time of the generic algorithm using the ideas of [9]. In particular, we give an $O(n^2)$ algorithm for finding a Nash equilibrium of a graphical game on a path of length n. Instead of storing best response policies, this algorithm stores appropriately-defined subsets, which, following [9], we call *breakpoint policies* (modifying the definition as necessary). We obtain the following theorem

THEOREM 1. *There is an $O(n^2)$ algorithm that finds a Nash equilibrium of a graphical game with two actions per player on an n-vertex path. There is an $O(n^3)$ algorithm that computes a representation of all Nash equilibria of such a game.*

In Section 5 we extend the results of Section 4 to general degree-2 graphs, obtaining the following theorem.

THEOREM 2. *There is a polynomial-time algorithm that finds a Nash equilibrium of a graphical game with two actions per player on a graph with maximum degree 2.*

In Section 6 we extend our algorithm so that it can be used to find a Nash equilibrium of a graphical game on an arbitrary tree. Even when the tree has bounded degree, the running time can be exponential. We show that this is inevitable by constructing a family of graphical games on bounded-degree trees for which best response policies of some of the vertices have exponential size, and any two-pass algorithm (i.e., an algorithm that is similar in spirit to that of [8]) has to store almost all points of the best response policies. In particular, we show the following.

THEOREM 3. *There is an infinite family of graphical games on bounded-degree trees with pathwidth 2 such that any two-pass algorithm for finding Nash equilibria on these trees requires exponential time and space.*

It is interesting to note that the trees used in the proof of Theorem 3 have pathwidth 2, that is, they are very close to being paths. It is an open question whether our algorithm runs in polynomial time for graphs of pathwidth 1. This question can be viewed as a generalization of a very natural computational geometry problem — we describe it in more detail in Section 8.

In Section 7, we give a complexity-theoretic intractability result for the problem of finding a Nash equilibrium of a graphical game on a graph with small pathwidth. We prove the following theorem.

THEOREM 4. *Consider the problem of finding a Nash equilibrium for a graphical game in which the underlying graph has maximum degree 3 and pathwidth k. There is a constant k such that this problem is PPAD-complete.*

Theorem 4 limits the extent to which we can exploit "path-like" properties of the underlying graph, in order to find Nash equilibria. To prove Theorem 4, we use recent PPAD-completeness results for games, in particular the papers [7, 4] which show that the problem of finding Nash equilibria in graphical games of degree d (for $d \geq 3$) is computationally equivalent to the problem of solving r-player normal-form games (for $r \geq 4$), both of which are PPAD-complete.

2. PRELIMINARIES AND NOTATION

We consider graphical games in which the underlying graph G is an n-vertex tree. Each vertex has two actions, which are denoted by 0 and 1. A mixed strategy is given by a single number $x \in [0, 1]$, which denotes the probability that the player selects action 1.

Fur the purposes of the algorithm, the tree is rooted arbitrarily. For convenience, we assume without loss of generality that the root has a single child, and that its payoff is independent of the action chosen by the child. This can be achieved by first choosing an arbitrary root of the tree, and then adding a dummy "parent" of this root, giving the new parent a constant payoff function.

Given an edge (V, W) of the tree G, and a mixed strategy w for W, let $G_{(V,W),W=w}$ be the instance obtained from G by (1) deleting all nodes Z which are separated from V by W (i.e., all nodes Z such that the path from Z to V passes through W), and (2) restricting the instance so that W is required to play mixed strategy w.

Definition 1. Suppose that (V, W) is an edge of the tree, that v is a mixed strategy for V and that w is a mixed strategy for W.

We say that v is a *potential best response* to w (denoted by $v \in \mathrm{pbr}_V(w)$) if there is an equilibrium in the instance $G_{(V,W),W=w}$ in which V has mixed strategy v. We define the *best response policy* for V, given W, as $\mathcal{B}(W,V) = \{(w,v) \mid v \in \mathrm{pbr}_V(w), w \in [0,1]\}$. Typically, W is the parent of V, and this is just referred to as "the best response policy for V". The expression $\mathcal{B}(W,V)|_{V=v}$ is used to denote the set $\mathcal{B}(W,V) \cap [0,1]\times\{v\}$.

The upstream pass of the generic algorithm of [8] computes the best response policy for V for every node V other than the root. With the above assumptions about the root, the downstream pass is straightforward: Let W denote the root and V denote its child. The root selects any pair (w,v) from $\mathcal{B}(W,V)$. It decides to play mixed strategy w and it instructs V to play mixed strategy v. The remainder of the downward pass is recursive. When a node V is instructed by its parent to adopt mixed strategy v, it does the following for each child U — It finds a pair $(v,u) \in \mathcal{B}(V,U)$ (with the same v value that it was given by its parent) and instructs U to play u.

3. ALGORITHM OF LITTMAN ET AL.

The algorithm of [9] is based on the following observation: to compute a single Nash equilibrium by a two-pass algorithm, it is not necessary to construct the entire best response policy for each vertex. As long as, at each step of the downstream pass, the vertex under consideration can select a vector of strategies for all its children so that each child's strategy is a potential best response to the parent's strategy, the algorithm succeeds in producing a Nash equilibrium. This can be achieved if, at the beginning of the downstream pass, we have a data structure in which each vertex V with parent W stores a set $\hat{\mathcal{B}}(W,V) \subseteq \mathcal{B}(W,V)$ (called a *breakpoint policy*) which covers every possible $w \in [0,1]$. We will show later that a sufficient condition for the construction of such a data structure is the invariant that, at every level of the upstream pass, $\hat{\mathcal{B}}(W,V)$ contains a continuous path from $w=0$ to $w=1$.

In [9], it is suggested that we can select the breakpoint policy in a particular way. Namely, the paper uses the following definition:

Definition 2. (cf. [9]) A breakpoint policy for a node V with parent W consists of an ordered set of W-breakpoints $w_0 = 0 < w_1 < w_2 < \cdots < w_{t-1} < w_t = 1$ and an associated set of V-values $v_1, \ldots, v_t$. The interpretation is that for any $w \in [0,1]$, if $w_{i-1} < w < w_i$ for some index i and W plays w, then V shall play v_i; and if $w = w_i$ for some index i, then V shall play any value between v_i and v_{i+1}. We say such a breakpoint policy has $t-1$ breakpoints.

The paper then claims that any vertex V can compute its breakpoint policy with respect to its parent W given the breakpoint policies of its children $U_1, \ldots, U_k$. The proof proceeds by ordering the children's breakpoints (i.e., the respective values of v) from left to right (it can be assumed without loss of generality that all these breakpoints are distinct) and considering them in turn; each such point $v_l \in \{v_1, \ldots, v_L\}$ corresponds to a fixed choice of strategies for $k-1$ children and an interval of admissible strategies for one child. Assume for convenience that this child is U_1 and its interval of admissible strategies at v_l is $[a,b]$; assume also that for U_j, $j = 2, \ldots, k$, their respective breakpoint policies prescribe them to play u_j in response to v_l. Let $P^i(u,w)$, $i = 0, 1$, be the expected payoff for V when V plays i, U_1 plays u, each U_j, $j = 2, \ldots, k$, plays u_j, and W plays w, and consider the set

$$\mathcal{W}_l = \{w \in [0,1] \mid \exists u \in [a,b] \text{ s.t. } P^0(u,w) = P^1(u,w)\};$$

note that for any $w \in \mathcal{W}_l$ we have $v_l \in \mathrm{pbr}_V(w)$.

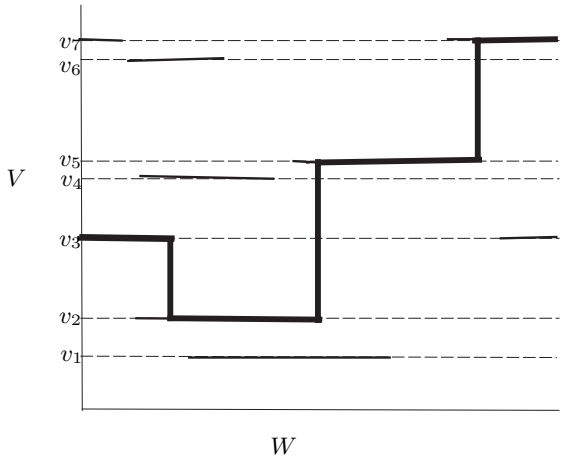

Figure 1: LKS: Trimming to find breakpoint policies.

The authors show that for any breakpoint v_l, the set $\mathcal{W}_l$ is either empty, a single interval, or a union of two non-floating intervals (an interval is *non-floating* if one of its endpoints is 0 or 1); moreover, the union of all sets $\mathcal{W}_l$, $l = 1, \ldots, L$, covers the interval $[0,1]$. It follows easily that one can cover $[0,1]$ with at most $L+2$ intervals, each of which is a subset of some $\mathcal{W}_l$. The authors then claim that any such cover can be transformed into a breakpoint policy for V. Namely, they say that for any two intervals $\mathcal{W}_{l_1}$ and $\mathcal{W}_{l_2}$ in the cover, "Any overlap between $\mathcal{W}_{l_1}$ and $\mathcal{W}_{l_2}$ can be arbitrarily assigned coverage by $\mathcal{W}_{l_1}$ and $\mathcal{W}_{l_2}$ "trimmed" accordingly (cf. [9], p. 5). They illustrate their approach in a figure, which is reproduced as Figure 1 here. In the figure, the dashed horizontal lines represent the breakpoints $v_1, v_2, \ldots, v_7$ and the solid intervals along these breakpoints are the sets $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_7$. The thick connected path is the corresponding breakpoint policy. It is chosen as follows: begin on the left, and always "jump" to the interval allowing greatest progress to the right.

To see why this approach does not work in general, consider a path of length 4 consisting of an indifferent root R, its child W, W's child V, and V's child U. Suppose that U receives a payoff of 1 if it plays differently to V and 0 otherwise. Thus, if v denotes the mixed strategy of V (i.e., V plays 1 with probability v), then the expected payoff that U derives from playing 0 is given by $P^0(U) = v$ and the expected payoff that U derives from playing 1 is given by $P^1(U) = 1 - v$. Suppose that V derives no payoff from playing 1 (so $P^1(V) = 0$) and that its payoff matrix for playing 0 is $\begin{pmatrix} 1 & -9 \\ 9 & -1 \end{pmatrix}$, so if u denotes the mixed strategy of U and w denotes the mixed strategy of W, the expected payoff that V derives from playing 0 is given by $P^0(V) = (1-u)(1-w) + (1-u)w(-9) + u(1-w)9 + uw(-1)$.

Using the techniques of [8] (or, alternatively, those of Section 4), it is not hard to verify that the best response policies for U and V (as in Definition 1) are given by the graphs in Figure 2. The best response policy for U is a breakpoint policy for U (as in Definition 2) with V-breakpoints $v_0 = 0$, $v_1 = 1/2$ and $v_2 = 1$ with associated values $u_1 = 1$ and $u_2 = 0$. The best response policy for V is not a breakpoint policy (because of how the curve from $w = 0$ to $w = 1$ "doubles back").

The LKS algorithm would "trim" to get a breakpoint policy such as the one in Figure 3. Note that this breakpoint policy $\hat{\mathcal{B}}(W,V)$ is invalid in the sense that it does not satisfy $\hat{\mathcal{B}}(W,V) \subseteq \mathcal{B}(W,V)$.

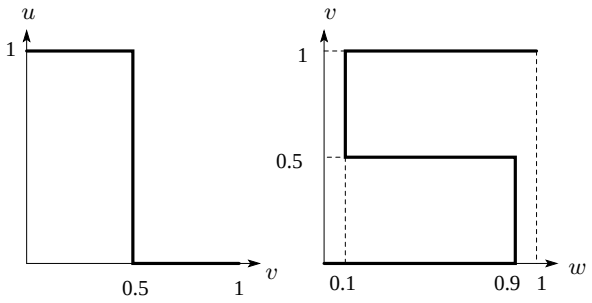

Figure 2: Best response policies for U and V.

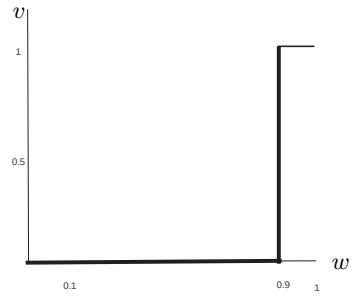

Figure 3: A "trimmed" policy for V

The point is that the payoff matrix of W can now be chosen to prevent the LKS algorithm from finding a Nash equilibrium. For example, suppose the payoffs are given so that $P^0(W) = v$ and $P^1(W) = (1-v)2$. The best response policy for W is a horizontal line at $w = .1$ (This is the value of w that allows $v = 2/3$ — see Figure 2, which makes $P^0(W) = P^1(W)$.) In the downward pass, the chosen values are $w = .1$, then, from the trimming, $v = 0$ and $u = 1$, which is not a Nash equilibrium since W prefers action 1.

The failure of the algorithm is not caused by the fact that the trimming policy goes as far to the right as possible. Any other "trimming" would be just as bad. For example, suppose the breakpoint policy for V has $v = 0$ until some point $w^* < .9$ and then jumps to $v = 1$. The algorithm is then defeated by the payoff matrix with $P^0(W) = 2v$ and $P^1(W) = (1 - v)$ in which the best response policy for W is a horizontal line at $w = .9$. The algorithm then gives $w = .9$, $v = 1$, and $u = 0$, which is not a Nash equilibrium since W prefers action 0.

We conclude that the LKS algorithm does not always find a Nash equilibrium. In Sections 4 and 6 we show how to modify the algorithm so that it always finds a Nash equilibrium. For the modified algorithm, we have to extend the definition of "breakpoint policy" (see Definition 3) so that it includes breakpoint policies such as the best response policy for V in Figure 2. Unfortunately, such a breakpoint policy may be exponential in size (see Figure 7) so the corrected algorithm does not run in polynomial time on all trees. In the next section, we show that it runs in polynomial time on a path.

4. FINDING EQUILIBRIA ON A PATH

In this section, we focus on the case when the underlying graph is a path, i.e., its vertex set is $\{V_1, \ldots, V_n\}$, and its edge set is $\{(V_j, V_{j+1}) \mid j = 1, \ldots, n-1\}$. We show that in this case the best response policy for each vertex can be represented as a union of a polynomial number of rectangles, where a *rectangle* is defined by a pair of closed intervals (I_V, I_U) and consists of all points in $I_V \times I_U$; it may be the case that one or both of the intervals I_V and I_U consists of a single point.

THEOREM 5. *For any $j = 1, \ldots, n$, the set $\mathcal{B}(V_j, V_{j-1})$ can be represented as a disjoint union of at most $(j+4)^2$ rectangles. Moreover, given such representation of $\mathcal{B}(V_j, V_{j-1})$, one can compute a representation of $\mathcal{B}(V_{j+1}, V_j)$ in time $O(j^2)$.*

PROOF. For any set $A \subseteq [0,1]^2$ that is represented as a union of a finite number of rectangles, we say that a point $u \in [0,1]$ on the U-axis is a *U-event point* of A if $u = 0$ or $u = 1$ or A contains a rectangle of the form $I_V \times I_U$ and u is an endpoint of I_U; V-event points are defined similarly. Observe that for any $u \in [0,1]$, the number of connected components of $[0,1] \times \{u\} \cap A$ is at most the number of V-event points of A.

We use induction on j to show that for each V_j the statement of the theorem holds and, additionally, each $\mathcal{B}(V_j, V_{j-1})$ has at most $2j + 4$ event points.

To simplify the base case, we modify the graphical game by appending a dummy vertex V_0 to the beginning of the path: the only neighbour of V_0 is V_1, the payoffs of V_0 are always equal to 0, and the payoffs of all other vertices (including V_1) are the same as in the original game.

For $j = 0$, we have $\mathcal{B}(V_1, V_0) = [0,1]^2$, so the statement of the theorem is trivially true.

Now, suppose that $j > 0$, set $V = V_j$ and let $U = V_{j-1}$ and $W = V_{j+1}$ be the vertices that precede and follow V, respectively. The payoffs to V are described by a $2 \times 2 \times 2$ matrix P: P_{xyz} is the payoff that V receives when U plays x, V plays y, and W plays z, where $x, y, z \in \{0, 1\}$. Suppose that U plays 1 with probability u and W plays 1 with probability w. Then V's expected payoff from playing 0 is

$$P^0 = (1-u)(1-w)P_{000} + (1-u)wP_{001} + u(1-w)P_{100} + uwP_{101},$$

while its expected payoff from playing 1 is

$$P^1 = (1-u)(1-w)P_{010} + (1-u)wP_{011} + u(1-w)P_{110} + uwP_{111}.$$

If $P^0 > P^1$, V strictly prefers to play 0, if $P^0 < P^1$, V strictly prefers to play 1, and if $P^0 = P^1$, V is indifferent, i.e., can play any (mixed) strategy. Since P^0 and P^1 are linear in w and u, there exist some constants A_1, A_0, B_1, and B_0 that depend on the matrix P, but not on u and w, such that

$$P^0 - P^1 = w(B_1 u + B_0) - (A_1 u + A_0). \tag{1}$$

Depending on the values of A_1, A_0, B_1, and B_0, we subdivide the rest of the proof into the following cases.

- $B_1 = 0, B_0 = 0$.

In this case, $P^0 > P^1$ if and only if $A_1 u + A_0 < 0$.

If also $A_1 = 0$, $A_0 = 0$, clearly, $\mathcal{B}(W, V) = [0,1]^2$, and the statement of the theorem is trivially true.

Otherwise, the vertex V is indifferent between 0 and 1 if and only if $A_1 \neq 0$ and $u = -A_0/A_1$. Let $\mathcal{V} = \{v \mid v \in (0,1), -A_0/A_1 \in \mathrm{pbr}_U(v)\}$. By the inductive hypothesis, $\mathcal{V}$ consists of at most $2(j-1)+4$ segments and isolated points.

For any $v \in \mathcal{V}$, we have $\mathcal{B}(W, V)|_{V=v} = [0,1]$: no matter what W plays, as long as U is playing $-A_0/A_1$, V is content to play v. On the other hand, for any $v \in (0,1) \setminus \mathcal{V}$ we have $\mathcal{B}(W, V)|_{V=v} = \emptyset$: when V plays v, U can only respond with $u \neq -A_0/A_1$, in which case V can benefit from switching to one of the pure strategies.

To complete the description of $\mathcal{B}(W, V)$, it remains to analyze the cases $v = 0$ and $v = 1$. The vertex V prefers to play 0 if $A_1 > 0$ and $u \leq -A_0/A_1$, or $A_1 < 0$ and $u \geq -A_0/A_1$, or

$A_1 = 0$ and $A_0 < 0$. Assume for now that $A_1 > 0$; the other two cases can be treated similarly. In this case $0 \in \mathrm{pbr}_V(w)$ for some $w \in [0,1]$ if and only if there exists a $u \in \mathrm{pbr}_U(0)$ such that $u \le -A_0/A_1$: if no such u exists, whenever V plays 0 either U's response is not in $\mathrm{pbr}_U(0)$ or V can improve its payoff by playing 1. Therefore, either $\mathcal{B}(W,V)|_{V=0} = [0,1]$ or $\mathcal{B}(W,V)|_{V=0} = \emptyset$. Similarly, $\mathcal{B}(W,V)|_{V=1}$ is equal to either $[0,1]$ or $\emptyset$, depending on $\mathrm{pbr}_U(1)$.

Therefore, the set $\mathcal{B}(W,V)$ consists of at most $2j+4 \le (j+4)^2$ rectangles: $\mathcal{B}(W,V) \cap [0,1]\times(0,1) = [0,1]\times\mathcal{V}$ contributes at most $2j+2$ rectangles, and each of the sets $\mathcal{B}(W,V)|_{V=0}$ and $\mathcal{B}(W,V)|_{V=1}$ contributes at most one rectangle. Similarly, its total number of event points is at most $2j+4$: the only W-event points are 0 and 1, each V-event point of $\mathcal{B}(W,V)$ is a V-event point of $\mathcal{B}(V,U)$, and there are at most $2j+2$ of them.

- $B_1 u + B_0 \not\equiv 0$, $A_1 = \alpha B_1$, $A_0 = \alpha B_0$ for some $\alpha \in \mathbf{R}$.

In this case, V is indifferent between 0 and 1 if and only if $w = \alpha$, or $B_1 \ne 0$ and $u = -B_0/B_1 = -A_0/A_1$. Similarly to the previous case, we can show that $\mathcal{B}(W,V) \cap [0,1]\times(0,1)$ consists of the rectangle $\{\alpha\}\times[0,1]$ and at most $2j+2$ rectangles of the form $[0,1]\times I_V$, where each I_V corresponds to a connected component of $\mathcal{B}(V,U)|_{U=-B_0/B_1}$.

Furthermore, V prefers to play 0 if $B_1 u + B_0 > 0$ and $w \ge \alpha$ or $B_1 u + B_0 < 0$ and $w \le \alpha$. Therefore, if $B_1 u^* + B_0 > 0$ for some $u^* \in \mathrm{pbr}_U(0)$, then $\mathcal{B}(W,V)|_{V=0}$ contains $[\alpha, +\infty) \cap [0,1]$ and if $B_1 u^{**} + B_0 < 0$ for some $u^{**} \in \mathrm{pbr}_U(0)$, then $\mathcal{B}(W,V)|_{V=0}$ contains $[-\infty, \alpha] \cap [0,1]$; if both u^* and u^{**} exist, $\mathcal{B}(W,V)|_{V=0} = [0,1]$. The set $\mathcal{B}(W,V)|_{V=1}$ can be described in a similar manner.

By the inductive hypothesis, $\mathcal{B}(V,U)$ has at most $2j+2$ event points; as at least two of these are U-event points, it has at most $2j$ V-event points. Since each V-event point of $\mathcal{B}(W,V)$ is a V-event point of $\mathcal{B}(V,U)$ and $\mathcal{B}(W,V)$ has at most 3 W-event points (0, 1, and α), its total number of event points is at most $2j+3 < 2j+4$. Also, similarly to the previous case it follows that $\mathcal{B}(W,V)$ consists of at most $2j+4 < (j+4)^2$ rectangles.

- $B_1 u + B_0 \not\equiv 0$, $\alpha(B_1 u + B_0) \not\equiv A_1 u + A_0$.

In this case, one can define the *indifference function* $f(\cdot)$ as $f(u) = \frac{A(u)}{B(u)} = \frac{A_1 u + A_0}{B_1 u + B_0}$, where $A(u)$ and $B(u)$ never turn into zero simultaneously. Observe that whenever $w = f(u)$ and $u, w \in [0,1]$, V is indifferent between playing 0 and 1. For any $A \subseteq [0,1]^2$, we define a function $\hat{f}_V$ by $\hat{f}_V(A) = \{(f(u), v) \mid (v,u) \in A\}$; note that $\hat{f}_V$ maps subsets of $[0,1]^2$ to subsets of $\mathbf{R}\times[0,1]$. Sometimes we drop the subscript V when it is clear from the context.

LEMMA 1. *For any $(w,v) \in [0,1]\times(0,1)$ we have $(w,v) \in \mathcal{B}(W,V)$ if and only if there exists a $u \in [0,1]$ such that $(v,u) \in \mathcal{B}(V,U)$ and $w = f(u)$.*

PROOF. Fix an arbitrary $v \in (0,1)$. Suppose that U plays some $u \in \mathrm{pbr}_U(v)$, $w = f(u)$ satisfies $w \in [0,1]$, and W plays w. There exists a vector of strategies $v_1, \ldots, v_{j-1} = u, v_j = v$ such that for each V_k, $k < j$, its strategy is a best response to its neighbours' strategies. Since $w = f(u)$, V is indifferent between playing 0 and 1; in particular, it can play v. Therefore, if we define $v_{j+1} = w$, the vector of strategies $(v_1, \ldots, v_{j+1})$ will satisfy the conditions in the definition of potential best response, i.e., we have $v \in \mathrm{pbr}_V(w)$.

Conversely, suppose $v \in \mathrm{pbr}_V(w)$ for some $w \in [0,1]$, $v \ne 0, 1$. Then there exists a vector of strategies $v_1, \ldots, v_{j-1}, v_j = v, v_{j+1} = w$ such that for each V_k, $k \le j$, its strategy is a best response to its neighbours' strategies. As $v \ne 0, 1$, V is, in fact, indifferent between playing 0 and 1, which is only possible if $w = f(v_{j-1})$. Choose $u = v_{j-1}$; by construction, $u \in \mathrm{pbr}_U(v)$. □

Lemma 1 describes the situations when V is indifferent between playing 0 and playing 1. However, to fully characterize $\mathcal{B}(W,V)$, we also need to know when V prefers a pure strategy.

Define $\hat{f}(0) = \cup_{u \in \mathrm{pbr}_U(0)} R_u$, where

$$R_u = \begin{cases} [f(u), +\infty)\times\{0\} & \text{if } B(u) > 0, \\ (-\infty, f(u)]\times\{0\} & \text{if } B(u) < 0. \end{cases}$$

and $\hat{f}(1) = \cup_{u \in \mathrm{pbr}_U(1)} R'_u$, where

$$R'_u = \begin{cases} [f(u), +\infty)\times\{1\} & \text{if } B(u) < 0, \\ (-\infty, f(u)]\times\{1\} & \text{if } B(u) > 0. \end{cases}$$

LEMMA 2. *For any $w \in [0,1]$, we have $(w,0) \in \hat{f}(0)$ if and only if $0 \in \mathrm{pbr}_V(w)$ and $(w,1) \in \hat{f}(1)$ if and only if $1 \in \mathrm{pbr}_V(w)$.*

PROOF. Consider an arbitrary $u_0 \in \mathrm{pbr}_U(0)$. If $B(u_0) > 0$, for $u = u_0$ the inequality $P^0 \ge P^1$ is equivalent to $w \ge f(u_0)$. Therefore, when U plays u_0 and W plays w, $w \ge f(u_0)$, V prefers to play 0; as $u_0 \in \mathrm{pbr}_U(u)$, it follows that $0 \in \mathrm{pbr}_V(w)$. The argument for the case $B(u_0) < 0$ is similar.

Conversely, if $0 \in \mathrm{pbr}_V(w)$ for some $w \in [0,1]$, there exists a vector $(v_1, \ldots, v_{j-1}, v_j = 0, v_{j+1} = w)$ such that for each V_k, $k \le j$, V_k plays v_k, and this strategy is a best response to the strategies of V_k's neighbours. Note that for any such vector we have $v_{j-1} \in \mathrm{pbr}_U(0)$. By way of contradiction, assume $(w,0) \notin \bigcup_{u \in \mathrm{pbr}_U(0)} R_u$. Then it must be the case that for any $u_0 \in \mathrm{pbr}_U(0)$ either $f(u_0) < w$ and $R_{u_0} = (-\infty, f(u_0)]\times\{0\}$ or $f(u_0) > w$ and $R_{u_0} = [f(u_0), +\infty)\times\{0\}$. In both cases, when V plays 0, U plays u_0, and V plays w, the inequality between $f(u_0)$ and w is equivalent to $P^0 < P^1$, i.e., V would benefit from switching to 1.

The argument for $\hat{f}(1)$ is similar. □

Together, Lemma 1 and Lemma 2 completely describe the set $\mathcal{B}(W,V)$: we have

$$\mathcal{B}(W,V) = \left(\hat{f}(0) \cup \hat{f}(\mathcal{B}(V,U)) \cup \hat{f}(1)\right) \bigcap [0,1]^2.$$

It remains to show that $\mathcal{B}(W,V)$ can be represented as a union of at most $(j+4)^2$ rectangles, has at most $2j+4$ event points, and can be computed in $O(j^2)$ time.

Set $u^* = -B_0/B_1$. [2] Consider an arbitrary rectangle $R = [v_1, v_2]\times[u_1, u_2] \subseteq \mathcal{B}(V,U)$. If $u^* \notin [u_1, u_2]$, the function $f(\cdot)$ is continuous on $[u_1, u_2]$ and hence $\hat{f}(R) = [f_{\min}, f_{\max}]\times[v_1, v_2]$, where

$$f_{\min} = \min\{f(u_1), f(u_2)\}, f_{\max} = \max\{f(u_1), f(u_2)\},$$

i.e., in this case $\hat{f}(R) \cap [0,1]^2$ consists of a single rectangle.

Now, suppose that R is intersected by the line $[0,1]\times\{u^*\}$; as was noted earlier, there are at most $2j+2$ such rectangles. Suppose that $\lim_{u \to u^{*-}} f(u) = +\infty$; as $f(\cdot)$ is a fractional linear function, this implies that $\lim_{u \to u^{*+}} f(u) = -\infty$ and also $f(u_1) > f(u_2)$. Since $f(\cdot)$ is continuous on $[u_1, u^*)$ and $(u^*, u_2]$, it is easy to see that

$$\hat{f}([v_1, v_2]\times[u_1, u^*)) = [f(u_1), +\infty)\times[v_1, v_2]$$

[2] The case $B_1 = 0$ causes no special problems. For completeness, set u^* to be any value outside of $[0,1]$ in this case.

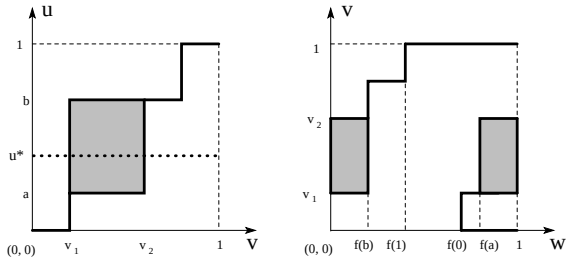

Figure 4: f is increasing on $(-\infty, u^*)$ and $(u^*, +\infty)$.

and

$$\hat{f}([v_1, v_2]\times(u^*, u_2]) = (-\infty, f(u_2)]\times[v_1, v_2],$$

i.e., in this case $\hat{f}(R) \cap [0,1]^2$ consists of at most two rectangles. The case $\lim_{u\to u^*-} f(u) = -\infty$ is similar.

As $\hat{f}(\mathcal{B}(V,U)) = \bigcup_{R\subset\mathcal{B}(V,U)} \hat{f}(R)$, it follows that $\hat{f}(\mathcal{B}(V,U))$ consists of at most $(j+3)^2 + 2j + 2$ rectangles. Also, it is easy to see that both $\hat{f}(0)$ and $\hat{f}(1)$ consist of at most 2 line segments each. We conclude that $\mathcal{B}(W,V)$ can be represented as a union of at most $(j+3)^2 + 2j + 6 < (j+4)^2$ rectangles.

Moreover, if v is a V-event point of $\mathcal{B}(W,V)$, then v is a V-event point of $\mathcal{B}(V,U)$ (this includes the cases $v = 0$ and $v = 1$, as 0 and 1 are V-event points of $\mathcal{B}(V,U)$) and if w is a W-event point of $\mathcal{B}(W,V)$, then either $w = 0$ or $w = 1$ or there exists some $u \in [0,1]$ such that $w = f(u)$ and u is a U-event point of $\mathcal{B}(V,U)$. Hence, $\mathcal{B}(W,V)$ has at most $2j + 4$ event points.

The $O(j^2)$ bound on the running time in Theorem 5 follows from our description of the algorithm. The $O(n^3)$ bound on the overall running time for finding a Nash equilibrium (and a representation of all Nash equilibria) follows.

4.1 Finding a Single Nash Equilibrium in $O(n^2)$ Time

The upper bound on the running time of our algorithm is tight, at least assuming the straightforward implementation, in which each $\mathcal{B}(V_{j+1}, V_j)$ is stored as a union of rectangles: it is not hard to construct an example in which the size of $\mathcal{B}(V_{j+1}, V_j)$ is $\Omega(j^2)$.

However, in some cases it is not necessary to represent *all* Nash equilibria; rather, the goal is to find an arbitrary equilibrium of the game. In this section, we show that this problem can be solved in quadratic time, thus obtaining a proof of Theorem 1. Our solution is based on the idea of [9], i.e., working with subsets of the best response policies rather than the best response policies themselves; following [9], we will refer to such subsets as *breakpoint policies*. While it is not always possible to construct a breakpoint policy as defined in [9], we show how to modify this definition so as to ensure that a breakpoint policy always exists; moreover, we prove that for a path graph, the breakpoint policy of any vertex can be stored in a data structure whose size is linear in the number of descendants this vertex has.

Definition 3. A *breakpoint policy* $\hat{\mathcal{B}}(V,U)$ for a vertex U whose parent is V is a non-self-intersecting curve of the form

$$X_1 \cup Y_1 \cup \cdots \cup Y_{m-1} \cup X_m,$$

where $X_i = [v_{i-1}, v_i]\times\{u_i\}$, $Y_i = \{v_i\}\times[u_i, u_{i+1}]$ and $u_i, v_i \in [0,1]$ for $i = 0, \ldots, m$. We say that a breakpoint policy is *valid* if $v_0 = 0$, $v_m = 1$, and $\hat{\mathcal{B}}(V,U) \subseteq \mathcal{B}(V,U)$.

We will sometimes abuse notation by referring to $\hat{\mathcal{B}}(V,U)$ as a collection of segments X_i, Y_i rather than their union. Note that we do not require that $v_i \le v_{i+1}$ or $u_i \le u_{i+1}$; consequently, in any argument involving breakpoint policies, all segments are to be treated as directed segments. Observe that any valid breakpoint policy $\hat{\mathcal{B}}(V,U)$ can be viewed as a continuous 1–1 mapping $\gamma(t) = (\gamma_v(t), \gamma_u(t))$, $\gamma : [0,1] \mapsto [0,1]^2$, where $\gamma(0) = (0, u_1)$, $\gamma(1) = (1, u_m)$ and there exist some $t_0 = 0, t_1, \ldots, t_{2m-2} = 1$ such that $\{\gamma(t) \mid t_{2k} \le t \le t_{2k+1}\} = X_{k+1}$, $\{\gamma(t) \mid t_{2k+1} \le t \le t_{2k+2}\} = Y_{k+1}$.

As explained in Section 3, we can use a valid breakpoint policy instead of the best response policy during the downstream pass, and still guarantee that in the end, we will output a Nash equilibrium. Theorem 6 shows that one can inductively compute valid breakpoint policies for all vertices on the path; the proof of this theorem can be found in the full version of this paper [6].

THEOREM 6. *For any $V = V_j$, one can find in polynomial time a valid breakpoint policy $\hat{\mathcal{B}}(W,V)$ that consists of at most $2j+1$ segments.*

5. NASH EQUILIBRIA ON GRAPHS WITH MAXIMUM DEGREE 2

In this section we show how the algorithm for paths can be applied to solve a game on any graph whose vertices have degree at most 2. A graph having maximum degree 2 is, of course, a union of paths and cycles. Since each connected component can be handled independently, to obtain a proof of Theorem 2, we only need to show how to deal with cycles.

Given a cycle with vertices $V_1, \ldots, V_k$ (in cyclic order), we make two separate searches for a Nash equilibrium: first we search for a Nash equilibrium where some vertex plays a pure strategy, then we search for a *fully mixed* Nash equilibrium, where all vertices play mixed strategies. For $i \le k$ let v_i denote the probability that V_i plays 1.

The first search can be done as follows. For each $i \in \{1, \ldots, k\}$ and each $b \in \{0,1\}$, do the following.

1. Let P be the path $(V_{i+1}, V_{i+2} \ldots, V_k, V_1, \ldots, V_{i-1}, V_i)$
2. Let payoff to V_{i+1} be based on putting $v_i = b$ (so it depends only on v_{i+1} and v_{i+2}.)
3. Apply the upstream pass to P
4. Put $v_i = b$; apply the downstream pass For each vertex, V_j, keep track of all possible mixed strategies v_j
5. Check whether V_{i+1} has any responses that are consistent with $v_i = b$; if so we have a Nash equilibrium. (Otherwise, there is no Nash equilibrium of the desired form.)

For the second search, note that if V_i plays a mixed strategy, then v_{i+1} and v_{i-1} satisfy an equation of the form $v_{i+1} = (A_0 + A_1 v_{i-1})/(B_0 + B_1 v_{i-1})$. Since all vertices in the cycle play mixed strategies, we have $v_{i+3} = (A_0' + A_1' v_{i+1})/(B_0' + B_1' v_{i+1})$. Composing the two linear fractional transforms, we obtain $v_{i+3} = (A_0'' + A_1'' v_{i-1})/(B_0'' + B_1'' v_{i-1})$. for some new constants A_0'', A_1'', B_0'', B_1''.

Choose any vertex V_i. We can express v_i in terms of v_{i+2}, then v_{i+4}, v_{i+6} etc. and ultimately v_i itself to obtain a quadratic equation (for v_i) that is simple to derive from the payoffs in the game. If the equation is non-trivial it has at most 2 solutions in $(0,1)$. For an odd-length cycle all other v_j's are derivable from those solutions, and if a fully mixed Nash equilibrium exists, all the v_j should turn out to be real numbers in the range $(0,1)$. For an even-length cycle, we obtain two quadratic equations, one for v_i and another for

v_{i+1}, and we can in the same way test whether any solutions to these yield values for the other v_j, all of which lie in $(0, 1)$.

If the quadratic equation is trivial, there is potentially a continuum of fully-mixed equilibria. The values for v_i that may occur in a Nash equilibrium are those for which all dependent v_j values lie in $(0, 1)$; the latter condition is easy to check by computing the image of the interval $(0, 1)$ under respective fractional linear transforms.

6. FINDING EQUILIBRIA ON AN (ARBITRARY) TREE

For arbitrary trees, the general structure of the algorithm remains the same, i.e., one can construct a best response policy (or, alternatively, a breakpoint policy) for any vertex based on the best response policies of its children. We assume that the degree of each vertex is bounded by a constant K, i.e., the payoff matrix for each vertex is of size $O(2^K)$.

Consider a vertex V whose children are $U_1, \ldots, U_k$ and whose parent is W; the best response policy of each U_j is $\mathcal{B}(V, U_j)$. Similarly to the previous section, we can compute V's expected payoffs P^0 and P^1 from playing 0 or 1, respectively. Namely, when each of the U_j plays u_j and W plays w, we have $P^0 = L^0(u_1, \ldots, u_k, w)$, $P^1 = L^1(u_1, \ldots, u_k, w)$, where the functions $L^0(\cdot, \ldots, \cdot), L^1(\cdot, \ldots, \cdot)$ are linear in all of their arguments. Hence, the inequality $P^0 > P^1$ can be rewritten as

$$wB(u_1, \ldots, u_k) > A(u_1, \ldots, u_k),$$

where both $A(\cdot, \ldots, \cdot)$ and $B(\cdot, \ldots, \cdot)$ are linear in all of their arguments. Set $\vec{u} = (u_1, \ldots, u_k)$ and define the indifference function $f : [0, 1]^k \mapsto [0, 1]$ as $f(\vec{u}) = A(\vec{u})/B(\vec{u})$; clearly, if each U_j plays u_j, W plays w and $w = f(\vec{u})$, V is indifferent between playing 0 and 1. For any $X = X_1 \times \cdots \times X_k$, where $X_i \subseteq [0, 1]^2$ define

$$\hat{f}(X) = \{(f(\vec{u}), v) \mid (v, u_i) \in X_i, i = 1, \ldots, k\}$$

Also, set

$$\hat{f}(0) = \{(w, 0) \mid \exists \vec{u} \text{ s.t. } u_i \in \mathrm{pbr}_{U_i}(0) \text{ and } wB(\vec{u}) \geq A(\vec{u})\}$$

and

$$\hat{f}(1) = \{(w, 1) \mid \exists \vec{u} \text{ s.t. } u_i \in \mathrm{pbr}_{U_i}(1) \text{ and } wB(\vec{u}) \leq A(\vec{u})\}.$$

As in previous section, we can show that $\mathcal{B}(W, V)$ is equal to

$$\Big(\hat{f}(0) \cup \hat{f}(\mathcal{B}(V, U_1) \times \cdots \times \mathcal{B}(V, U_k)) \cup \hat{f}(1)\Big) \bigcap [0, 1]^2;$$

also, any path from $w = 0$ to $w = 1$ that is a subset of $\mathcal{B}(W, V)$ constitutes a valid breakpoint policy.

6.1 Exponential Size Breakpoint Policy

While the algorithm of Section 4 can be generalized for bounded-degree trees, its running time is no longer polynomial. In fact, the converse is true: we can construct a family of trees and payoff matrices for all players so that the best response policies for some of the players consist of an exponential number of segments. Moreover, in our example the breakpoint policies coincide with the best response policies, which means that even finding a single Nash equilibrium using the approach of [8, 9] is going to take exponentially long time. In fact, a stronger statement is true: for any polynomial-time two-pass algorithm (defined later) that works with subsets of best response policies for this graph, we can choose the payoffs of the vertices so that the downstream pass of this algorithm will fail.

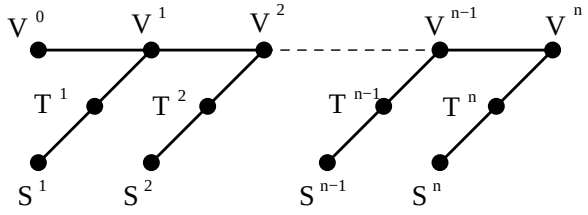

Figure 5: The tree $\mathcal{T}_n$ that corresponds to exponential-size breakpoint policy.

In the rest of this subsection, we describe this construction. Consider the tree $\mathcal{T}_n$ given by Figure 5; let V_n be the root of this tree. For every $k = 1, \ldots, n$, let the payoffs of S_k and T_k be the same as those for the U and V described in Section 3; recall that the breakpoint policies for U and V are shown in Figure 2. It is not hard to see that the indifference function for T_k is given by $f(s) = .8s + .1$.

The payoff of V_0 is 1 if V_1 selects the same action as V_0 and 0 otherwise; V_0's best response policy is given by Figure 6.

LEMMA 3. *Fix $k < n$, and let u, t, v, and w denote the strategies of V_{k-1}, T_k, V_k, and V_{k+1}, respectively. Suppose that V_k prefers playing 0 to playing 1 if and only if $.5t + .1u + .2 > w$. Then $\mathcal{B}(V_{k+1}, V_k)$ consists of at least 3^k segments. Moreover,*

$$\{(v, w) \mid (v, w) \in \mathcal{B}(V_{k+1}, V_k), 0 \leq w \leq .2\} = [0, .2] \times \{0\}$$

and

$$\{(v, w) \mid (v, w) \in \mathcal{B}(V_{k+1}, V_k), .8 \leq w \leq 1\} = [.8, 1] \times \{1\}.$$

PROOF. The proof proceeds by induction on k. For $k = 0$, the statement is obvious. Now, suppose it is true for $\mathcal{B}(V_k, V_{k-1})$.

One can view $\mathcal{B}(V_{k+1}, V_k)$ as a union of seven components: $\hat{f}(0) \cap [0, 1] \times \{0\}$, $\hat{f}(1) \cap [0, 1] \times \{1\}$, and five components that correspond to the segments of $\mathcal{B}(V_k, T_k)$. Let us examine them in turn.

To describe $\hat{f}(0) \cap [0, 1] \times \{0\}$, note that $f(u, t) = .5t + .1u + .2$ is monotone in t and u and satisfies $f(0, 0) = .2$. Also, we have $\mathrm{pbr}_{V_{k-1}}(0) = \{0\}$ and $\mathrm{pbr}_{T_k}(0) = \{0\}$. For any $w \in [0, 1]$ we have $f(0, 0) \geq w$ if and only if $w \in [0, .2]$. We conclude that $\hat{f}(0) \cap [0, 1] \times \{0\} = [0, .2] \times \{0\}$. Similarly, it follows that $\hat{f}(1) \cap [0, 1] \times \{1\} = [.8, 1] \times \{1\}$.

Define

$$S_1 = \{(f(u, 0), v) \mid (v, u) \in \mathcal{B}(V_k, V_{k-1}) \cap [0, .9] \times [0, 1]\},$$
$$S_2 = \{(f(u, .5), v) \mid (v, u) \in \mathcal{B}(V_k, V_{k-1}) \cap [.1, .9] \times [0, 1]\},$$
$$S_3 = \{(f(u, 1), v) \mid (v, u) \in \mathcal{B}(V_k, V_{k-1}) \cap [.1, 1] \times [0, 1]\};$$

these sets correspond to horizontal segments of $\mathcal{B}(V_k, T_k)$.

It is easy to see that $S_1, S_2, S_3 \subset \mathcal{B}(V_{k+1}, V_k)$. Since f is a continuous function, the number of segments in each S_i is at least the number of segments in $\mathcal{B}(V_k, V_{k-1}) \cap [.1, .9] \times [0, 1]$, which is at least 3^{k-1} by induction hypothesis. Moreover, as f is monotone in u and $f(1, 0) < f(0, .5) < f(1, .5) < f(0, 1)$, all S_i, $i = 1, 2, 3$, are disjoint.

Finally, the set $\mathcal{B}(V_{k+1}, V_k)$ contains two segments that correspond to the vertical segments of $\mathcal{B}(V_k, T_k)$, i.e.,

$$S_4 = \{(f(0, t), .1) \mid t \in [.5, 1]) = [.45, .7] \times \{.1\} \text{ and}$$
$$S_5 = \{(f(1, t), .9) \mid t \in [0, .5]) = [.3, .55] \times \{.9\}.$$

Clearly, S_4 connects S_2 and S_3, S_5 connects S_1 and S_2, and S_4 and S_5 do not intersect each other. We conclude that $\mathcal{B}(V_{k+1}, V_k)$

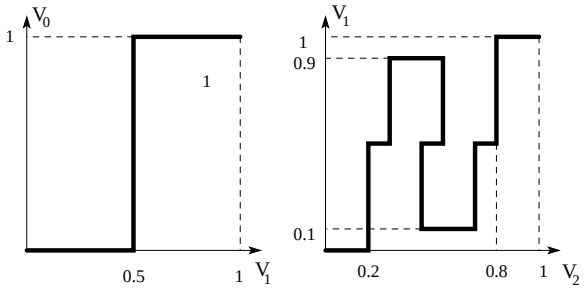

Figure 6: Breakpoint policies for V_0 and V_1.

is a continuous line that consist of at least 3^k segments and satisfies the condition of the lemma. □

To complete the construction, we need to show that we can design the payoff matrix for V_k so that it prefers playing 0 to playing 1 if and only if $.5t + .1u + .2 > w$. To this end, we prove a more general statement, namely, that the indifference function of a vertex can be an arbitrary fractional multilinear function of its descendants' strategies.

We say that a function of k variables is *multilinear* if it can be represented as a sum of monomials and each of these monomials is linear in all of its variables. Note that this definition is different from a more standard one in that we do not require that all of the monomials have the same degree. Recall that the payoffs of a vertex with $k+1$ neighbours are described by matrices P^0 and P^1, where $P^j_{i_0 i_1 \dots i_k}$ is the payoff that V gets when it plays j, and its neighbours play $i_0, \dots, i_k$, and $j, i_0, \dots, i_k \in \{0,1\}$. Let $P[j] = P[j](w, u_1, \dots, u_k)$ be the expected payoff obtained by this vertex when it plays j and the (mixed) strategies of its neighbours are given by a vector $(w, u_1, \dots, u_k)$, i.e., $P[j] = \mathbf{E}[P^j_{i_0 i_1 \dots i_k}]$ where $i_0, \dots, i_k$ are independent Bernoulli random variables, each of which is 1 with the respective probabilities $w, u_1, \dots, u_k$.

LEMMA 4. *Given a tree vertex V whose parent is W and whose children are $U_1, \dots, U_k$, for any function $f = f(u_1, \dots, u_k)$ that can be represented as a ratio of two multilinear functions f_1, f_2, i.e., $f = \frac{f_1(u_1,\dots,u_k)}{f_2(u_1,\dots,u_k)}$, there exist payoff matrices P^0 and P^1 for V such that*

$$P[0] - P[1] = w f_2(u_1, \dots, u_k) - f_1(u_1, \dots, u_k).$$

The proof of this lemma is based on the fact that every monomial of the form $a_s (u_0)^{s_0} \dots (u_k)^{s_k}$, $s_1, \dots, s_k \in \{0,1\}$, can be represented as

$$\sum_{t = t_0 \dots t_k \in \Sigma^{k+1}} C_t (u_0)^{t_0} (1-u_0)^{1-t_0} \dots (u_k)^{t_k} (1-u_k)^{1-t_k}$$

for some C_t, $t \in \{0,1\}^{k+1}$. The details can be found in the full version of this paper [6].

6.2 Irreducibility of the Best Response Policy for $\mathcal{T}_n$

While the best response policy constructed in the previous subsection has exponential size, it is not clear *à priori* that it is necessary to keep track of all of its line segments rather than to focus on a small subset of these segments. However, it turns out that for two-pass algorithms such as the algorithm of [8], the best response policy cannot be simplified. More precisely, we say that an algorithm $\mathcal{A}$ is a *two-pass* algorithm if

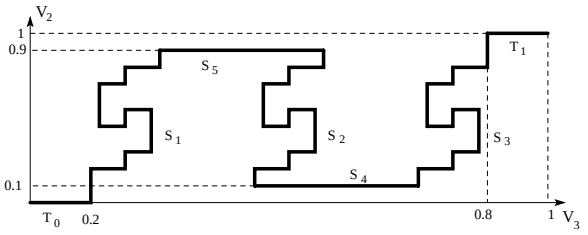

Figure 7: Breakpoint policy for V_2.

- $\mathcal{A}$ consists of an upstream pass and a downstream pass.
- During the upstream pass, for each vertex V with parent W, $\mathcal{A}$ constructs a set $\mathcal{BB}(W,V) \subseteq \mathcal{B}(W,V)$. This set is produced from the sets $\{\mathcal{BB}(V,U) \mid U \text{ is a child of } V\}$ by applying the procedure from the beginning of Section 6 (substituting $\mathcal{BB}(V,U_j)$ for $\mathcal{B}(V,U_j)$ for all children U_j of V) , and then possibly omitting some of the points of the resulting set (which is then stored explicitly).
- The downstream pass is identical to the downstream pass of [8] as described in Section 2 except that it operates on the sets $\mathcal{BB}(W,V)$ rather than on the sets $\mathcal{B}(W,V)$.

Theorem 7 demonstrates that any two-pass algorithm will fail during the downstream pass on $\mathcal{T}_n$ if there is an index j such that the set $\mathcal{BB}(V_{j+1}, V_j)$ omits any interior point of any of the (at least 3^j) segments of $\mathcal{B}(V_{j+1}, V_j)$. This implies Theorem 3.

THEOREM 7. *For any two-pass algorithm $\mathcal{A}$ for which there exists an index j, $j \in [1, n/4]$, a segment S of $\mathcal{B}(V_j, V_{j-1})$, and an interior point (x,y) of S such that $\mathcal{BB}(V_j, V_{j-1})$ does not contain (x,y), we can choose payoff matrices of the vertices $V_j, \dots, V_n$ so that the downstream pass of $\mathcal{A}$ will fail, and, additionally, payoffs to $V_{4j}, \dots, V_n$ are identically 0.*

We sketch the proof of Theorem 7; the details can be found in the full version of this paper [6]. We proceed by induction. For $j = 1$, the argument is similar to that in Section 3. For the inductive step, the main idea is that we can "zoom in" on any part of a best response policy (including the part that was omitted!) by using an appropriate indifference function; this allows us to reduce the case $j = j_0$ to $j = j_0 - 1$.

7. PPAD-COMPLETENESS OF BOUNDED PATHWIDTH GRAPHICAL GAMES

In the previous section, we showed that for graphical games on trees that are almost but not quite paths, two-pass algorithms fail to find Nash equilibria in polynomial time. We next show that a milder path-like graph property allows us to construct graphical games for which it is unlikely that *any* polynomial-time algorithm will find Nash equilibria.

7.1 Pathwidth

A *path decomposition* of a graph $G = (V, E)$ is a sequence of subset $S_i(V) \subseteq V$ such that for each edge $(v, v') \in E$, $v, v' \in S_i(V)$ for some i, and furthermore, for each $v \in V$, if $v \in S_i(V)$ and $v \in S_j(V)$ for $j > i$, then $v \in S_k(V)$ for all $i \le k \le j$. The path decomposition has width k if all sets $S_i(V)$ have cardinality at most $k+1$. The *pathwidth* of G is the minimum width of any path decomposition of G.

Pathwidth is a restriction of *treewidth* (in which one would seek a tree whose vertices were the sets $S_i(V)$, and the sets containing some vertex would have to form a subtree). For any constant k it can be decided in polynomial time whether a graph has pathwidth (or treewidth) k. Furthermore many graph-theoretic problems seem easier to solve in polynomial time, when restricted to fixed treewidth, or pathwidth, graphs, see [1] for an overview. Note that a path has pathwidth 1 and a cycle has pathwidth 2.

7.2 PPAD-completeness

We review some basic definitions from the computational complexity theory of search problems. A search problem associates any input (here, a graphical game) with a set of solutions (here, the Nash equilibria of the input game), where the description length of any solution should be polynomially bounded as a function of the description length of its input. In a *total* search problem, there is a guarantee that at least one solution exists for any input. Nash's theorem assures us that the problem of finding Nash equilibria is total.

A *reduction* from search problem $\mathcal{S}$ to problem $\mathcal{S}'$ is a mechanism that shows that any polynomial-time algorithm for $\mathcal{S}'$ implies a polynomial-time algorithm for $\mathcal{S}$. It consists of functions f and g, computable in polynomial time, where f maps inputs of $\mathcal{S}$ to inputs of $\mathcal{S}'$, and g maps solutions of $\mathcal{S}'$ to solutions of $\mathcal{S}$, in such a way that if $I_{\mathcal{S}}$ is an input to $\mathcal{S}$, and $S_{\mathcal{S}'}$ is a solution to $f(I_{\mathcal{S}})$, then $g(S_{\mathcal{S}'})$ is a solution to $I_{\mathcal{S}}$.

Observe that total search problems do not allow the above reductions from problems such as CIRCUIT SAT (where the input is a boolean circuit, and solutions are input vectors that make the output *true*) due to the fact that CIRCUIT SAT and other NP-complete problems have inputs with empty solution sets. Instead, recent work on the computational complexity of finding a Nash equilibrium [7, 4, 5, 2, 3] has related it to the following problem.

Definition 4. END OF THE LINE. Input: boolean circuits S and P, each having n input and n output bits, where $P(0^n) = 0^n$ and $S(0^n) \neq 0^n$. Solution: $x \in \{0,1\}^n$ such that $S(x) = x$, or alternatively $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$.

S and P can be thought of as standing for "successor" and "predecessor". Observe that by computing $S^i(0^n)$ (for $i = 0, 1, 2, \ldots$) and comparing with $P(S^{i+1}(0^n))$, we must eventually find a solution to END OF THE LINE. END OF THE LINE characterizes the complexity class PPAD (standing for parity argument on a graph, directed version), introduced in Papadimitriou [11], and any search problem $\mathcal{S}$ is PPAD-complete if END OF THE LINE reduces to $\mathcal{S}$. Other PPAD-complete problems include the search for a ham sandwich hyperplane, and finding market equilibria in an exchange economy (see [11] for more detailed descriptions of these problems).

3-GRAPHICAL NASH is the problem of finding a Nash equilibrium for a graphical game whose graph has degree 3. Daskalakis et al. [4] show PPAD-completeness of 3-GRAPHICAL NASH by a reduction from 3-DIMENSIONAL BROUWER, introduced in [4] and defined as follows.

Definition 5. 3-DIMENSIONAL BROUWER. Input: a circuit C having $3n$ input bits and 2 output bits. The input bits define a "cubelet" of the unit cube, consisting of the 3 coordinates of its points, given to n bits of precision. The output represents one of four colours assigned by C to a cubelet. C is restricted so as to assign colour 1 to cubelets adjacent to the (y,z)-plane, colour 2 to remaining cubelets adjacent to the (x,z)-plane, colour 3 to remaining cubelets on the (x,y)-plane, and colour 0 to all other cubelets on the surface of the unit cube.

A solution is a *panchromatic vertex*, a vertex adjacent to cubelets that have 4 distinct colours.

The reason why a solution is guaranteed to exist, is that an associated *Brouwer function* ϕ can be constructed, i.e. a continuous function from the unit cube to itself, such that panchromatic vertices correspond to fixpoints of ϕ. Brouwer's Fixpoint Theorem promises the existence of a fixpoint.

The proof of Theorem 4 uses a modification of the reduction of [4] from 3-DIMENSIONAL BROUWER to 3-GRAPHICAL NASH. To prove the theorem, we begin with some preliminary results as follows. Each player has 2 actions, denoted 0 and 1. For a player at vertex V let $\mathbf{p}[V]$ denote the probability that the player plays 1.

LEMMA 5. *[7] There exists a graphical game $\mathcal{G}_{shift}$ of fixed size having vertices V, V' where $\mathbf{p}[V']$ is the fractional part of $2\mathbf{p}[V]$.*

COROLLARY 1. *There exists a graphical game $\mathcal{G}_{n-shift}$ of size $\Theta(n)$ of constant pathwidth, having vertices V, V_n where $\mathbf{p}[V_n]$ is the fractional part of $2^n.\mathbf{p}[V]$.*

PROOF. Make a chain of n copies of $\mathcal{G}_{shift}$ in Lemma 5. Each subset of vertices in the path decomposition is the vertices in a copy of $\mathcal{G}_{shift}$. □

Let $I_n(x)$ denote the n-th bit of the binary expansion of x, where we interpret 1 as *true* and 0 as *false*. The following uses gadgets from [7, 4].

COROLLARY 2. *There exists k such that for all n, and for all n_1, n_2, $n_3 \leq n$, there exists a graphical game of size $O(n)$ with pathwidth k, having vertices V_1, V_2, V_3 where $\mathbf{p}[V_3] = \mathbf{p}[V_1] + 2^{-n_3}(I_{n_1}\mathbf{p}[V_1] \wedge I_{n_2}\mathbf{p}[V_2])$.*

PROOF OF THEOREM 4. Let C be the boolean circuit describing an instance of 3-DIMENSIONAL BROUWER. Let $g_1, \ldots, g_{p(n)}$ be the gates of C indexed in such a way that the input(s) to any gate are the output(s) of lower-indexed gates. $g_1, \ldots, g_{3n}$ will be the $3n$ inputs to C.

All players in the graphical game $\mathcal{G}$ constructed in [4] have 2 actions denoted 0 and 1. The probability that V plays 1 is denoted $\mathbf{p}[V]$. $\mathcal{G}$ has 3 players V_x, V_y and V_z for which $\mathbf{p}[V_x]$, $\mathbf{p}[V_y]$ and $\mathbf{p}[V_z]$ represent the coordinates of a point in the unit cube. $\mathcal{G}$ is designed to incentivize V_x, V_y and V_z to adjust their probabilities in directions given by a Brouwer function which is itself specified by the circuit C. In a Nash equilibrium, $\mathbf{p}[V_x]$, $\mathbf{p}[V_y]$ and $\mathbf{p}[V_z]$ represent coordinates of a fixpoint of a function that belongs to the class of functions represented by 3-DIMENSIONAL BROUWER.

For $1 \leq i \leq p(n)$ we introduce a vertex $V_C^{(i)}$ such that for $1 \leq j \leq i$, $I_j(\mathbf{p}[V_C^{(i)}])$ is the output of gate g_j; for $i < j \leq p(n)$, $I_j(\mathbf{p}[V_C^{(i)}])$ is 0.

Construct $V_C^{(i)}$ from $V_C^{(i-1)}$ using Corollary 2. Let $\mathcal{G}^{(i)}$ be the graphical game that does this. Let $S_1(\mathcal{G}^{(i)}), \ldots, S_n(\mathcal{G}^{(i)})$ be a length n path decomposition of $\mathcal{G}^{(i)}$, where $V_C^{(i-1)} \in S_1(\mathcal{G}^{(i)})$ and $V_C^{(i)} \in S_n(\mathcal{G}^{(i)})$.

Then, a path decomposition of $\cup_{1 \leq i \leq p(n)} \mathcal{G}^{(i)}$ is obtained by taking the union of the separate path decompositions, together with $S_n(\mathcal{G}^{(i-1)}) \cup S_1(\mathcal{G}^{(i)})$ for $2 \leq i \leq p(n)$.

Let $\mathcal{G}_C$ be the above graphical game that simulates C. $\mathcal{G}_C$ has $3n$ inputs, consisting of the first n bits of the binary expansions of $\mathbf{p}[V_x]$, $\mathbf{p}[V_y]$ and $\mathbf{p}[V_z]$. Similarly to [4], the output of $\mathcal{G}_C$ affects V_x, V_y and V_z as follows. Colour 0 incentivizes V_x, V_y and V_z

to adjust their probabilities $\mathbf{p}[V_x]$, $\mathbf{p}[V_y]$ and $\mathbf{p}[V_z]$ in the direction $(-1,-1,-1)$; colour 2 incentivizes them to move in direction $(1,0,0)$; colour 2, direction $(0,1,0)$; colour 3, direction $(0,0,1)$.

We need to ensure that at points at the boundaries of adjacent cubelets, the change of direction will be approximately the average of directions of surrounding points. That way, all four colors/directions must be nearby so that they can cancel each other out (and we are at a panchromatic vertex). This is achieved using the same trick as [4], in which we make a constant number M of copies of $\mathcal{G}_C$, which differ in that each copy adds a tiny displacement vector to its copies of $\mathbf{p}[V_x]$, $\mathbf{p}[V_y]$ and $\mathbf{p}[V_z]$ (which are derived from the original using the addition gadget of [7]). Using the addition and multiplication gadgets of [7] we average the directions and add a small multiple of this average to $(\mathbf{p}[V_x], \mathbf{p}[V_y], \mathbf{p}[V_z])$.

At a Nash equilibrium the outputs of each copy will cancel each other out. The pathwidth of the whole game is at most M times the pathwidth $\mathcal{G}_C$. □

8. OPEN PROBLEMS

The most important problem left open by this paper is whether it is possible to find a Nash equilibrium of a graphical game on a bounded-degree tree in polynomial time. Our construction shows that any two-pass algorithm that explicitly stores breakpoint policies needs exponential time and space. However, it does not preclude the existence of an algorithm that is based on a similar idea, but, instead of computing the entire breakpoint policy for each vertex, uses a small number of additional passes through the graph to decide which (polynomial-sized) parts of each breakpoint policy should be computed. In particular, such an algorithm may be based on the approximation algorithm of [8], where the value of ϵ is chosen adaptively.

Another intriguing question is related to the fact that the graph for which we constructed an exponential-sized breakpoint policy has pathwidth 2, while our positive results are for a path, i.e., a graph of pathwidth 1. It is not clear if for any bounded-degree graph of pathwidth 1 the running time of (the breakpoint policy-based version of) our algorithm will be polynomial. In particular, it is instructive to consider a "caterpillar" graph, i.e., the graph that can be obtained from $\mathcal{T}_n$ by deleting the vertices $S_1, \ldots, S_n$. For this graph, the best response policy of a vertex V_k in the "spine" of the caterpillar is obtained by combining the best response policy of its predecessor on the spine V_{k-1} and its other child T_k; since the latter is a leaf, its best response policy is either trivial (i.e., $[0,1]^2$, $[0,1]\times\{0\}$, or $[0,1]\times\{1\}$) or consists of two horizontal segments and one vertical segment of the form $\{\alpha\}\times[0,1]$ that connects them. Assuming for convenience that

$$\mathcal{B}(V_k, T_k) = [0,\alpha]\times\{0\} \cup \{\alpha\}\times[0,1] \cup [\alpha,1]\times\{1\},$$

and f is the indifference function for V_k, we observe that the best response policy for V_k consists of 5 components: $\hat{f}(0)$, $\hat{f}(1)$, and three components that correspond to $[0,\alpha]\times\{0\}$, $\{\alpha\}\times[0,1]$, and $[\alpha,1]\times\{1\}$.

Hence, one can think of constructing $\mathcal{B}(V_{k+1}, V_k)$ as the following process: turn $\mathcal{B}(V_k, V_{k-1})$ by $\pi/2$, cut it along the (now horizontal) line $v_k = \alpha$, apply a fractional linear transform to the horizontal coordinate of both parts, and reconnect them using the image of the segment $\{\alpha\}\times[0,1]$ under f. This implies that the problem of bounding the size of the best response policy (or, alternatively, the breakpoint policy), can be viewed as a generalization of the following computational geometry problem, which we believe may be of independent interest:

PROBLEM 1. *Given a collection of axis-parallel segments in $\mathbf{R}^2$, consider the following operation: pick an axis-parallel line l_i (either vertical or horizontal), cut the plane along this line, and shift one of the resulting two parts by an arbitrary amount δ_i; as a result, some segments will be split into two parts. Reconnect these parts, i.e., for each segment of the form $[a,b]\times\{c\}$ that was transformed into $[a,t]\times\{c+\delta_i\}$ and $[t,b]\times\{c\}$, introduce a segment $\{t\}\times[c,c+\delta_i]$. Is it possible to start with the segment $[0,1]$ and after n operations obtain a set that* cannot *be represented as a union of $\mathrm{poly}(n)$ line segments? If yes, can it be the case that in this set, there is no path with a polynomial number of turns that connects the endpoints of the original segment?*

It turns out that in general, the answer to the first question is positive, i.e., after n steps, it is possible to obtain a set that consists of $\Theta(c^n)$ segments for some $c > 0$. This implies that even for a caterpillar, the best response policy can be exponentially large. However, in our example (which is omitted from this version of the paper due to space constraints), there exists a polynomial-size path through the best response policy, i.e., it does not prove that the breakpoint policy is necessarily exponential in size. If one can prove that this is always the case, it may be possible to adapt this proof to show that there can be an exponential gap between the sizes of best response policies and breakpoint policies.

9. REFERENCES

[1] H. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21:358–402, 1996.

[2] X. Chen and X. Deng. 3-NASH is PPAD-complete. Technical Report TR-05-134, Electronic Colloquium in Computational Complexity, 2005.

[3] X. Chen and X. Deng. Settling the complexity of 2-player Nash equilibrium. Technical Report TR-05-140, Electronic Colloquium in Computational Complexity, 2005.

[4] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, 2006.

[5] C. Daskalakis and C. Papadimitriou. Three-player games are hard. Technical Report TR-05-139, Electronic Colloquium in Computational Complexity, 2005.

[6] E. Elkind, L. Goldberg, and P. Goldberg. Nash equilibria in graphical games on trees revisited. Technical Report TR-06-005, Electronic Colloquium in Computational Complexity, 2006.

[7] P. Goldberg and C. Papadimitriou. Reducibility among equilibrium problems. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, 2006.

[8] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, 2001.

[9] M. Littman, M. Kearns, and S. Singh. An efficient exact algorithm for singly connected graphical games. In *Proceedings of the 15th Annual Conference on Neural Information Processing Systems*, 2001.

[10] L. Ortiz and M. Kearns. Nash propagation for loopy graphical games. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems*, 2003.

[11] C. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.