Computing Approximate Bayes-Nash Equilibria in Tree-Games of Incomplete Information

Satinder Singh Vishal Soni Michael P. Wellman
University of Michigan
Computer Science and Engineering
Ann Arbor, MI 48109
baveja,soniv,wellman@umich.edu

ABSTRACT

We provide efficient algorithms for finding approximate Bayes-Nash equilibria (BNE) in graphical, specifically tree, games of incomplete information. In such games an agent's payoff depends on its private type as well as on the actions of the agents in its local neighborhood in the graph. We consider two classes of such games: (1) arbitrary tree-games with discrete types, and (2) tree-games with continuous types but with constraints on the effect of type on payoffs. For each class we present a message passing on the game-tree algorithm that computes an ϵ-BNE in time polynomial in the number of agents and the approximation parameter $\frac{1}{\epsilon}$.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms

Keywords

Structured Games, Games of Incomplete Information, Approximate Bayes-Nash Equilibria

1. INTRODUCTION

Graphical representations for games were introduced by Kearns, Littman, and Singh [6] to capture locality of interaction among agents in a static general-sum game. The nodes in a graphical game represent agents. A missing edge between two nodes in the graph implies that the payoff of each of the two agents is not directly dependent on the other agent's actions. Sparse connections in the game can be exploited to define compact payoff functions for the game. In recent work on tree-games of *complete* information, various authors have shown that the locality of interaction can also be exploited to obtain computationally efficient algorithms for computing Nash [8] and correlated equilibria [5]. Locality of interaction can arise from many sources, e.g., the spatial organization of sales agents, the network connectivity of computing devices, the social organization of a population of animals, the work organization of businesses, etc., can all lead to sparsely connected graphical games and so advances in computational methods for such games [14, 7] are of obvious benefit.

Prior work on graphical games has focused on games of complete information. In this paper, we consider graphical games of *incomplete* information. In such games an agent's payoff depends not only on the actions of its neighbors but also on its own private type [3]. Agents have to choose actions without knowledge of the private type of other agents. Because of the incomplete information the relevant equilibrium concept is that of a Bayes-Nash equilibrium (BNE) in which each agent plays a best response to the other agents in expectation over their private types. Current algorithms for computing BNE are impractical for general incomplete information games with many agents. Nevertheless, many important classes of games have incomplete information, e.g., sales agents may have private information about the cost of providing services, computing devices in a network may have private information about the status of local resources, animals in a social network may have private information about their value for relationships, etc.

Our main aim in this work is to find efficient algorithms for computing *approximate* BNE by exploiting locality of interaction in tree-games of incomplete information. Throughout we will assume that the number of actions available to the agents is a constant and our primary concern will be to scale efficiently with respect to the number of agents in the game. Our main contributions in this work are as follows. First, we consider general tree-games of incomplete information with discrete types and provide a message passing algorithm for computing an approximate BNE. Our algorithm is a relatively straightforward adaptation of prior work [6] on tree-games of complete information. Second, we consider tree-games with continuous types which pose a more significant challenge because such games may in general not even be finitely representable. We propose an interesting subclass of such games in which there always exist a compactly representable BNE and provide a different message-passing algorithm for computing an approximation to such a com-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
EC'04, May 17–20, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-711-0/04/0005 ...$5.00.

pact BNE. Finally, we analyze both our algorithms and prove that in their respective classes of games they find ϵ-BNE in time polynomial in the number of agents and in the approximation parameter $\frac{1}{\epsilon}$.

The rest of this paper is organized as follows. We briefly describe general incomplete information games and then describe graphical models for such games. Next, we present an abstract message passing algorithm that computes all BNE in tree-games. This abstract algorithm serves as a template that we specialize in subsequent sections to provide efficient algorithms for games with discrete types and then for games with continuous types. We finish with a small illustrative example and conclusions.

2. PRELIMINARIES

In this section we briefly describe incomplete information games in general and then the straightforward extension of graphical model representations [6] to such games.

2.1 Incomplete Information Games

We restrict attention to simultaneous-move, one-shot, normal form games exactly as in the prior work on graphical models for games with complete information [6, 8]. The main departure in this paper is that each agent also has a *private type* known only to that agent. The payoff to an agent x is not only a function of all the agents' actions (as in the usual complete information game) but also of the *realized* private-type of agent x. The type of an agent may be discrete or continuous. We consider discrete types in the first half of this paper and continuous types in the second half. As in the usual framework for such problems we will assume that each agent's realized type is chosen independently from some commonly *known* distribution over types, and that the payoff matrices for the agents are also common knowledge. These games have incomplete information because each agent must choose its strategy, i.e., its probability distribution over its actions, without knowing the realized types of all the other agents. An agent's strategy, on the other hand, will in general condition on its own realized type.

Notation: Let the number of agents be denoted n. For games with discrete-types, the realized type for each agent is assumed to be chosen independently from some discrete set $T = \{T_1, T_2, \ldots, T_m\}$ according to a distribution P_T. For games with continuous types, the realized type of each agent is assumed to be chosen independently from some interval $T = [t_l\ t_u]$ according to some probability density p_T. For ease of exposition, we assume that all the agents have the same set of actions $A = \{a_1, a_2, \ldots, a_d\}$ available to them, though our results extend trivially to the case where the agents have their own action sets. We use $\overrightarrow{a}$, $\overrightarrow{t}$, and $\overrightarrow{\mu}$ to denote the vector of actions, realized-types and strategies of all the agents. Agent x's strategy, μ_x, is in general a *type-conditional mixed-strategy*, i.e., it is a mapping from types to probability distributions over actions. So, $\mu_x(a|i)$ is the probability of taking action a given that the realized-type is i, and $\mu_x(\cdot|i)$ is the mixed-strategy of agent x given that its realized-type is i. We use the symbol σ to refer to *unconditional* mixed-strategies and reserve the symbol μ for type-conditional strategies. The payoff to agent x, denoted $R_x(\overrightarrow{a}|t_x)$, depends on its realized-type and the actions of all the agents. Specifying an incomplete information game requires defining the payoffs for each agent in the game, i.e., requires specifying $O(nmd^n)$ numbers (recall that n is the number of agents, m is the number of discrete types and d is the number of actions). Finally, $\overrightarrow{t}_x$, $\overrightarrow{a}_x$, and $\overrightarrow{\mu}_x$ will be used to refer to the type, action, and strategy of agent x, while $\overrightarrow{t}_{-x}$, $\overrightarrow{a}_{-x}$, and $\overrightarrow{\mu}_{-x}$ will be used to refer to the types, actions, and strategies of all the agents excluding agent x.

Next we define the relevant equilibrium concepts in incomplete information games with discrete types. (Equations 1 and 2 below can be modified for the case of continuous types by simply replacing summations over types with integrals.) The *expected* payoff to agent x if it plays (unconditional) mixed-strategy σ and its type is t_x when the other agents play according to $\overrightarrow{\mu}_{-x}$ is

$$R_x(\sigma, \overrightarrow{\mu}_{-x}|t_x) = \sum_{\overrightarrow{t}_{-x}} P_T(\overrightarrow{t}_{-x})\Big(\sum_{\overrightarrow{a}} \sigma(\overrightarrow{a}_x) \overrightarrow{\mu}_{-x}(\overrightarrow{a}_{-x}|\overrightarrow{t}_{-x}) R_x(\overrightarrow{a}|t_x)\Big) \quad (1)$$

and the expected payoff to agent x when the agents play according to $\overrightarrow{\mu}$ is

$$R_x(\overrightarrow{\mu}) = \sum_{t_x} P_T(t_x) R_x(\overrightarrow{\mu}_x(\cdot|t_x), \overrightarrow{\mu}_{-x}|t_x). \quad (2)$$

For agent x, mixed strategy σ is said to be a *best response* for type t_x to $\overrightarrow{\mu}$, if

$$\forall \sigma' \quad R_x(\sigma, \overrightarrow{\mu}_{-x}|t_x) \geq R_x(\sigma', \overrightarrow{\mu}_{-x}|t_x). \quad (3)$$

A Bayes-Nash equilibrium or *BNE* [4] is a strategy vector $\overrightarrow{\mu}$ such that every agent is playing a best response to the others, i.e.,

$$\forall \text{ agents } x, \text{ and } \forall \mu',\ R_x(\overrightarrow{\mu}) \geq R_x(\mu', \overrightarrow{\mu}_{-x}) \quad (4)$$

We also use the standard definition of approximate Bayes-Nash equilibria. A mixed strategy σ is an ϵ-best-response for agent x with type t_x if for all mixed-strategies σ':

$$R_x(\sigma, \overrightarrow{\mu}_{-x}|t_x) + \epsilon \geq R_x(\sigma', \overrightarrow{\mu}_{-x}|t_x). \quad (5)$$

Similarly, an ϵ-BNE is a strategy vector $\overrightarrow{\mu}$ such that every agent is playing an ϵ-best-response to the other agents, i.e.,

$$\forall \text{ agents } x, \text{ and } \forall \mu'_x,\ R_x(\overrightarrow{\mu}) + \epsilon \geq R_x(\mu'_x, \overrightarrow{\mu}_{-x}). \quad (6)$$

Since BNE are simply Nash equilibria with respect to the expanded strategy space and type distributions [4], it follows from Nash's original result [11] that BNE and hence ϵ-BNE exist for all games of incomplete information [3].

The complexity of computing exact or even approximate BNE in general incomplete information games is still unknown. However a number of hardness results are available for special cases, e.g., recently, Conitzer and Sandholm [2] showed that determining whether a pure-strategy BNE exists is NP-hard. In this paper, we consider the special case of static incomplete information games in which the interaction among the agents can be modeled using graphs that are trees. We provide efficient algorithms for computing approximate BNE in two interesting and large classes of such games.

Next we define the graphical models representation for games with incomplete information.

2.2 Graphical Models for Incomplete Information Games

Graphical models for games have nodes representing agents and edges representing direct interaction between agents (see Figures 1 and 3 for examples). In incomplete information games each node's agent has a private type chosen independently from a known distribution over types. Each node has a payoff matrix that defines the associated agent's payoff as a function of its realized type as well as its neighbours (including itself) actions. Thus the more sparse the graph the more local the interaction among agents. Let the largest number of neighbors in the graph be denoted k. Defining the payoff matrices for a graphical game with discrete-types takes $O(nmd^k)$ numbers. If $k \ll n$ the graphical game formalism can lead to significantly more compact description than the usual game formalism which takes $O(nmd^n)$ numbers. Nevertheless, graphical models for games constitute a completely general representation of static incomplete-information games because a complete graph is identical to the usual static game representation. Accordingly, there always exists a BNE for a graphical game of incomplete information.

The locality of interaction in graphical games can also be exploited to reduce the amount of computation required to compute various payoff related quantities as follows. Let $\mathcal{N}^x$ denote the set of neighbors of x *including* node x itself. The neighbors of node x excluding node X are denoted $\mathcal{N}^x_{-x}$. We can incorporate the neighborhood-based locality into the various payoff definitions for general incomplete-information games of the previous subsection. For example, Equation 1, the expected payoff to agent x when its realized type is t_x and it plays mixed-strategy σ becomes

$$R_x(\sigma, \overrightarrow{\mu}_{\mathcal{N}^x_{-x}} | t_x) = \sum_{\overrightarrow{t}_{\mathcal{N}^x_{-x}}} P_T(\overrightarrow{t}_{\mathcal{N}^x_{-x}}) \Big(\sum_{\overrightarrow{a}_{\mathcal{N}^x}} \Big[\sigma(\overrightarrow{a}_x) \overrightarrow{\mu}_{\mathcal{N}^x_{-x}}(\overrightarrow{a}_{\mathcal{N}^x_{-x}} | \overrightarrow{t}_{\mathcal{N}^x_{-x}}) R_x(\overrightarrow{a}_{\mathcal{N}^x} | t_x) \Big] \Big).$$

The equations (2, 3, 4, 5, and 6) for exact and approximate best-response and BNE can be similarly modified to use $\mathcal{N}^x$.

2.3 Tree-Games of Incomplete Information

In this paper we focus on games in which the neighborhood graph is a tree. Figures 1 and 3 show snippets of such tree-games of incomplete information. Note that, as in Figure 3, we can always select an arbitrary node as the root and orient the tree so that the root node is at the bottom and leaves are at the top. Nodes along the path from a node x to the root are *downstream* from x while nodes that are along a path from x to any leaf node are *upstream* from x. The unique downstream neighbor of a non-root node is its child while the possibly many upstream neighbors of a non-leaf node are its parents.

The key question asked in this paper is: can one exploit the locality structure in tree-games of incomplete information to develop efficient algorithms for computing approximate BNE? In the remainder of this paper, we present algorithms and analyses that answers this question for two classes of finite-agent tree-games: (1) those with discrete types and no other constraints, and (2) those with continuous types and additional constraints on the effect of the type on payoff functions.

3. ABSTRACT-TREEBNE ALGORITHM

All of the algorithms we present in this paper are message-passing algorithms, comprising a downwards pass from the leaves to the root in which the space of possible BNE (ϵ-BNE) are computed, and then an upwards pass from the root to the leaves in which a random BNE (ϵ-BNE) is selected. For ease of exposition, we first present an abstract message-passing algorithm for computing all BNE in arbitrary tree-games of incomplete information. This algorithm, which we call **Abstract-TreeBNE**, will be presented and analyzed for correctness in purely conceptual terms without regard to computational or representational feasibility. Subsequently, we will define feasible algorithms by specializing the Abstract-TreeBNE algorithm by filling in the missing details for the two classes of tree-games of interest in this paper.

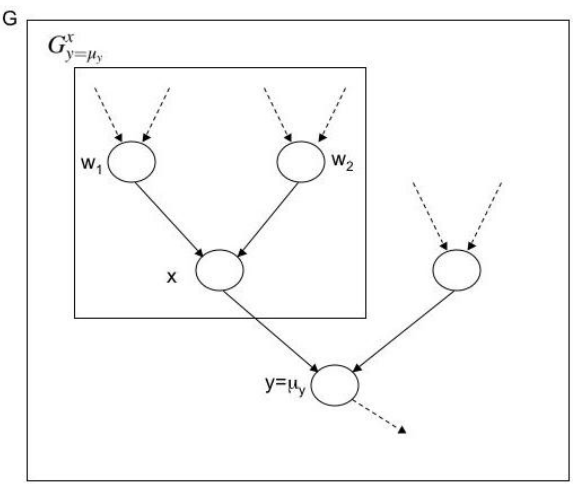

Figure 1: A fragment of a tree-game G that shows the induced game rooted at node x, denoted $G^x_{y=\mu_y}$, obtained by fixing the strategy of the unique child node y to μ_y.

Consider an arbitrary tree-game of incomplete information. We will use the symbol G to denote both the game as well as the oriented tree for that game (Figure 1 shows a fragment of such a tree G). For any node x, let G^x be the sub-tree rooted at node x, i.e., the subgraph induced by the nodes that are upstream from x including node x itself (see Figure 1). Let y be the child of x in the tree G. Note that agent y only interacts directly with node x among the nodes in G^x. If agent y plays according to strategy μ_y, then we can define a **subgame** for the agents in G^x by collapsing the payoff function R_x by one index, that of agent y, and by fixing y's strategy to μ_y. Effectively, fixing the strategy of the unique child of a node in the tree G decouples the sub-tree rooted at that node from the rest of the tree. Let the game defined on G^x obtained by fixing y to μ_y be denoted $G^x_{y=\mu_y}$. A BNE of the game $G^x_{y=\mu_y}$ can be thought of as a BNE *upstream* of x given that y plays μ_y.

The Abstract-TreeBNE algorithm is presented in Figure 2. In addition, Figure 3 shows a pictorial example of the Abstract-TreeBNE algorithm in action for a hypothetical game; it can be used to follow along with the algorithm description. The downwards pass starts at the leaves. Each node x sends to its unique child y a binary-value table D_{yx} indexed by all

Algorithm **Abstract-TreeBNE**
Inputs: Tree game specification.
Output: A BNE for the game.

1. Compute a depth-first ordering for the nodes in the tree.
2. **(Downstream Pass)** For each node x in depth-first ordering (starting at the leaves):
   - **(a)** Let node y be the child of x (or nil if x is the root).
   - **(b)** Initialize $D(\mu_y, \mu_x)$ to be 0 and the witness list for $D(\mu_y, \mu_x)$ to be empty for all μ_y, μ_x.
   - **(c)** If x is a leaf node:
     - **i.** For all μ_y, μ_x, set $D(\mu_y, \mu_x)$ to be 1 if and only if μ_x is a best-response for agent x to agent y playing μ_y (as determined by the payoff matrix R_x).
   - **(d)** Else (x is an internal node):
     - **i.** Let $\overrightarrow{w} = w_1, w_2, \ldots, w_k$ be the parents of node x; let $D(\mu_x, \mu_{w_i})$ be the table passed from w_i to x on the downstream pass.
     - **ii.** For all μ_x, μ_y, and all joint strategies $\overrightarrow{\mu}_{\overrightarrow{w}}$ for agents $\overrightarrow{w}$: If μ_x is a best response to y playing μ_y *and* $\overrightarrow{w}$ playing $\overrightarrow{\mu}_{\overrightarrow{w}}$ (as determined by the local payoff matrix R_x) and $D(\mu_x, \overrightarrow{\mu}_{w_i}) = 1$ for $i = 1, \ldots, k$, set $D(\mu_y, \mu_x)$ to be 1 and add $\overrightarrow{\mu}_{\overrightarrow{w}}$ to the witness list for $D(\mu_y, \mu_x)$.
   - **(e)** Pass the table $D(\mu_y, \mu_x)$ from x to y.
3. **(Upstream Pass)** For each node x in reverse depth-first ordering (starting at the root):
   - **(a)** Let $\overrightarrow{w} = \{w_1, w_2, \ldots, w_k\}$ be the parents of x (or the empty list if x is a leaf); let y be the child of x (or nil if x is root), and (μ_y, μ_x) the values passed from y to x on the upstream pass.
   - **(b)** Label x with the value μ_x.
   - **(c)** (Non-deterministically) Choose any witness $\overrightarrow{\mu}_{\overrightarrow{w}}$ to $D(\mu_y, \mu_x) = 1$.
   - **(d)** For $i = 1, \ldots, k$, pass $(\overrightarrow{\mu}_{w_i}, \mu_x)$ from x to w_i.

Figure 2: The Abstract-TreeBNE algorithm. See text for details.

possible strategies of the agents y and x. The semantics of the table is as follows: for any pair (μ_y, μ_x), $D_{yx}(\mu_y, \mu_x)$ will be 1 if and only if there exists an upstream-BNE in which x plays μ_x for the subgame $G^x_{y=\mu_y}$. Recall that the strategy for an agent is a mapping from types to the simplex of probability distributions over actions and so it may not be possible to represent the table D_{yx} compactly or even finitely for an arbitrary tree-game. As noted already, for now we are simply presenting an abstract algorithm that assumes a finite representation; later we will show how this assumption can be met in two different classes of tree-games.

The downwards pass is initialized at the leaves of the tree G, where the computation of the tables is straightforward. If x is a leaf and if y is its unique child, then $D_{yx}(\mu_y, \mu_x)$ is 1 if and only if μ_x is a best response to the agent y playing μ_y. This is straightforward because x's payoff depends only on its own type and on the actions of agents x and y (step 2c of Figure 2). Next consider a non-leaf and non-root node x with parents $\overrightarrow{w} = \{w_1, w_2, \ldots, w_k\}$ and child y. For induction, assume that each w_i sends the table $D_{x\,w_i}$ to x. For each pair (μ_y, μ_x), $D_{yx}(\mu_y, \mu_x)$ is set to 1 if and only if there exists a vector of strategies $\overrightarrow{\mu}_{\overrightarrow{w}} = \{\mu_{w_1}, \mu_{w_2}, \ldots, \mu_{w_k}\}$ (called a *witness*) for the parents $\overrightarrow{w}$ of x such that

1. $D_{x\,w_i}(\mu_x, \mu_{w_i}) = 1$ for all $1 \leq i \leq k$, and
2. μ_x is a best-response to $\overrightarrow{\mu}_{\overrightarrow{w}}$ and μ_y (recall that $\mathcal{N}^x = \{x, y, \overrightarrow{w}\}$).

Note that there may be more than one witness for $D_{yx}(\mu_y, \mu_x) = 1$. In addition to computing the binary-valued function D_{yx} on the downstream pass of the algorithm, x will also keep a list of witnesses $\overrightarrow{\mu}_{\overrightarrow{w}}$ for each pair (μ_y, μ_x) for which D_{yx} is 1 (step 2(d)ii of Figure 2). These witness lists will be used on the upwards pass. The downstream pass bottoms out at the root node, say z, which computes a special binary-valued table D_z as follows. Let the parents of z be $\overrightarrow{y} = \{y_1, y_2, \ldots, y_k\}$. Given the messages D_{zy_i}, $D_z(\mu_z)$ is set to 1 if and only if there exists a vector of strategies $\overrightarrow{\mu}_{\overrightarrow{y}} = \{\mu_{y_1}, \mu_{y_2}, \ldots, \mu_{y_k}\}$ such that $D_{zy_i}(\mu_z, \mu_{y_i}) = 1$ for all $1 \leq i \leq k$, and μ_z is a best response to $\overrightarrow{\mu}_{\overrightarrow{y}}$ (and as before, $\overrightarrow{\mu}_{\overrightarrow{y}}$ is a witness to $D_z(\mu_z) = 1$). How to compute these downwards pass messages D is not clear; for the abstract algorithm description, we assume that it can be done, and describe two specific implementations later.

To see that the upstream-BNE semantics of the tables are preserved by the abstract computation just described, suppose that for some node x with child y and parents $\overrightarrow{w} = \{w_1, w_2, \ldots, w_k\}$, $D_{yx}(\mu_y, \mu_x)$ was set to 1 for some witness $\overrightarrow{\mu}_{\overrightarrow{w}}$. By construction, for all i, $D_{xw_i}(\mu_x, \mu_{w_i})$ must also be 1, and therefore by induction it must be the case that there exists an upstream BNE in which w_i plays μ_{w_i} provided x plays μ_x. In addition, by construction of D_{yx}, μ_x is a best response of x to the strategies μ_y and $\overrightarrow{\mu}_{\overrightarrow{w}}$ and thus μ_x and $\overrightarrow{\mu}_{\overrightarrow{w}}$ must be part of an upstream BNE provided y plays μ_y, completing the induction argument. Finally, note that the existence of a BNE ensures that none of the tables computed during the downwards pass will be all 0.

The upstream pass starts at the root node z (with parents $\overrightarrow{y}$) which can choose any μ_z for which $D_z(\mu_z) = 1$ and

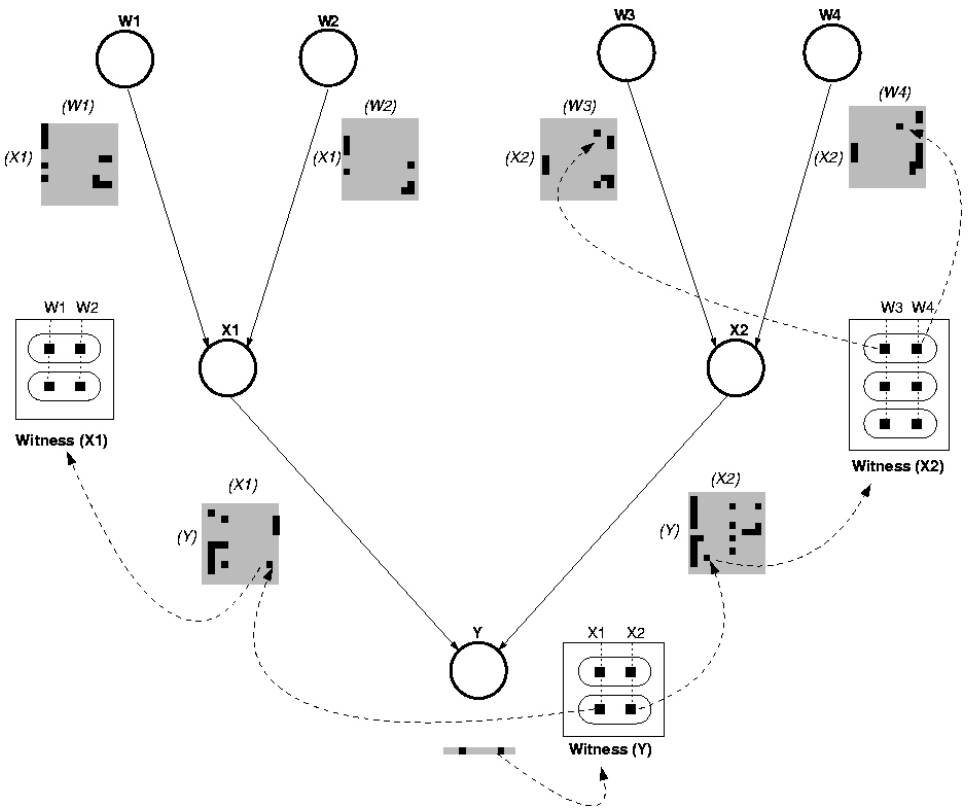

Figure 3: Illustrative view of the downwards and upwards passes of the Abstract-TreeBNE algorithm. In the downwards pass each node sends to its unique child a binary function of its own and child node's strategies. With each entry in the function there is also a witness list of parent-nodes strategies that are part of an upstream BNE. In the upwards pass entries from these witness lists are chosen to compute a BNE.

any witness $\overrightarrow{\mu}_{\overrightarrow{y}}$ from the associated witness list. The node z then passes (μ_z, μ_{y_i}) to parent y_i thereby telling agent y_i that z is going to play μ_z and that y_i should play μ_{y_i}. Inductively if a node x with parents $\overrightarrow{w}$ receives an upstream message (μ_y, μ_x) from its child y then it must be the case that $D_{yx}(\mu_y, \mu_x) = 1$. Node x will then play μ_x and it can choose from any witness $\overrightarrow{\mu}_{\overrightarrow{w}}$ associated with $D_{yx}(\mu_y, \mu_x) = 1$ and pass the message (μ_x, μ_{w_i}) to parent w_i. From the semantics of the downwards pass messages it must be the case that μ_x is a best response to μ_y and $\overrightarrow{\mu}_{\overrightarrow{w}}$. This upstream pass stops at the leaves at which point every agent would have been assigned a strategy that is a best response to the strategies assigned to all the other agents.

In our description of Abstract-TreeBNE above, we have left the choice of which witness to choose in the upstream pass unconstrained. If the witness-choice in the upstream pass is arbitrary, then Abstract-TreeBNE will return an arbitrary BNE. Other choices of witnesses are possible, and we discuss these in Section *4.1.2*. Here, we wish to emphasize that the tables and witnesses computed by Abstract-TreeBNE represent *all* the BNE of the tree-game G. We conclude this section with a theorem the proof of which is in the constructive argument presented above.

Theorem 1 *Algorithm* **Abstract-TreeBNE** *computes a BNE for the tree-game of incomplete information G. Furthermore, the tables and witness lists computed by the algorithm represent all BNE of G.*

4. COMPUTING APPROXIMATE BNE

The algorithm Abstract-TreeBNE presented above is incompletely specified because the representation and computation of the downwards pass tables were left unspecified. In this section we present our main results: efficient algorithms for computing approximate BNE in two classes of incomplete-information tree-games. First, we consider games with discrete types and no other constraints, and then we consider games with continuous types that are guaranteed to have BNE in a restricted class of strategies amenable to efficient computation.

4.1 Discrete Types

Here we adapt the conceptual Abstract-TreeBNE algorithm to compute ϵ-BNE in tree-games with discrete types. The algorithm **Approximate-TreeBNE** takes as input a parameter $\epsilon > 0$ that specifies how good an approximation to BNE we wish to find. As we show, the approximation can be made arbitrarily precise with correspondingly greater computational effort.

4.1.1 Approximate-TreeBNE Algorithm

The central idea in Approximate-TreeBNE(ϵ) is to discretize the unconditional strategy space so that instead of

playing arbitrary mixed strategies over actions, an agent can choose actions only with probabilities that are multiples of τ, for some τ to be determined by analysis. For games in which the agents have only two actions, the probability of playing the first action $\in \{0, \tau, 2\tau, \ldots, 1\}$. For games with d actions, each action's selection-probability is a multiple of τ with the total summing to 1. The discretized type-conditional strategies μ will then map each type to a discretized unconditional strategy. The number of different strategies for any agent x then will be $O\Big(\Big[\frac{1}{\tau}\Big]^{m(d-1)}\Big)$. The number of entries in the binary-valued tables D_{yx} will be determined by the number of joint strategies of agents x and y and thus the downstream-pass D_{yx} tables will be of size $O\Big(\Big[\frac{1}{\tau}\Big]^{2m(d-1)}\Big)$. A witness list at a node can at most contain all the joint strategies of the parent nodes and because the number of parents is at most $k-1$, the witness lists will be of size $O\Big(\Big[\frac{1}{\tau}\Big]^{m(d-1)(k-1)}\Big)$.

Algorithm **Approximate-TreeBNE(ϵ)**

Inputs: Tree game specification, and the approximation parameter ϵ
Outputs: An ϵ-BNE for the game.

Do Abstract-TreeBNE with two changes

1. only consider type-conditional discretized (with parameter τ defined in the text) strategies in both the downstream and upstream passes
2. the requirement of best-response to set a 1 in the downstream tables D is weakened to a requirement of ϵ-best response

Figure 4: The Approximate-TreeBNE algorithm. The two modifications to Abstract-TreeBNE (see Figure 2) required are listed here.

Figure 4 presents the Approximate-TreeBNE algorithm in terms of the two modifications it makes to the Abstract-TreeBNE algorithm. One modification is to only consider τ-discretized type-conditional strategies, thereby making all the tables involved finitely representable. The second modification is in the semantics of the downstream D tables. The new semantics is that $D_{yx}(\mu_y, \mu_x) = 1$ if and only if μ_x is a ϵ-best-response for x to μ_y and μ_x is a part of an upstream ϵ-BNE for the subgame $G^x_{y=\mu_y}$. With these two changes, Approximate-TreeBNE(ϵ) will return an ϵ-BNE. Of course, we have not yet shown that for any given ϵ we can always find a τ for which an ϵ-BNE exists in the space of τ-discretized type-conditional strategies, and that we can do all the computations involved in Approximate-TreeBNE efficiently. To this we turn next.

4.1.2 Analysis

Our main result is that Approximate-TreeBNE(ϵ) will return an ϵ-BNE with a running time that scales polynomially in the total number of agents n as well as the approximation parameter $\frac{1}{\epsilon}$.

Theorem 2 *For any $\epsilon > 0$, let*

$$\tau \leq \min\Big(\epsilon/(d^k(4k\log k)), 2/(k\log^2(k/2))\Big).$$

Then Approximate-TreeBNE computes an ϵ-BNE for the tree-game G. Furthermore, for every exact BNE, the tables and witness lists computed by the algorithm contain an ϵ-BNE that is within τ (in L_1 norm) of this exact BNE. The running time of the algorithm scales polynomially in the total number of agents and the accuracy parameter.

Proof Lemma 9 in the Appendix shows that if $\tau \leq 2/(k\log^2(k/2)$ then for every BNE in the tree-game G, the nearest strategy on the τ-grid will be a $d^k(4k\log k)\tau$-BNE. We want the algorithm to find ϵ-BNE and so if we set $\tau \leq \min\Big(\epsilon/(d^k(4k\log k)), 2/(k\log^2(k/2))\Big)$, then for every BNE in the tree-game G, there will exist a corresponding ϵ-BNE in the τ-discretized strategy space. The tables and witness lists of Approximate-TreeBNE will represent all the ϵ-BNE as this property is directly inherited from the original Abstract-TreeBNE algorithm. Thus, Approximate-TreeBNE will return an ϵ-BNE.

The running time of Approximate-TreeBNE is dominated by the time to compute the D tables for the downwards pass. Consider a node x with child node y and parent nodes $\overrightarrow{w}$. By assumption, the maximum number of neighbors of any node in G is k. Computing table D_{yx} requires filling out $O\Big(\Big[\frac{1}{\tau}\Big]^{2m(d-1)}\Big)$ entries. Computing the binary value of each entry involves considering all possible τ-discretized strategies of the parents of x (the nodes in $\overrightarrow{w}$) to see if there is any setting of the $\overrightarrow{w}$ strategies which make μ_x part of an upstream ϵ-BNE if y plays μ_y. The maximum number of strategies of the parents of x, $\overrightarrow{w}$, is $O\Big(\Big[\frac{1}{\tau}\Big]^{m(d-1)(k-1)}\Big)$. Thus the computation required to fill D_{xy} is of $O\Big(\Big[\frac{1}{\tau}\Big]^{m(d-1)(k+1)}\Big)$. There are n such tables to fill out in the downwards pass and so the total running time is $O\Big(n\Big[\frac{1}{\tau}\Big]^{m(d-1)(k+1)}\Big)$. The upstream pass does not add to the order of the running time. □

Corollary 3 *For discrete-type tree-games of incomplete information if the number of actions and discrete types is held constant, then Approximate-TreeBNE(ϵ) computes an ϵ-BNE in time polynomial in the size of the representation and $\frac{1}{\epsilon}$.*

The running time of Approximate-TreeNash scales polynomially with the accuracy parameter $\frac{1}{\epsilon}$ and the total number of agents n, but scales exponentially with the number of actions d, the number of discrete types m and the maximum number of neighbors k. Of course, the size of the representation itself scales exponentially with k (recall that the number of entries in the payoff functions is $O(nmd^k)$. Thus, if we can treat the number of actions and discrete types to be a constant, Approximate-TreeBNE is polynomial in the size of the tree-game representation and $\frac{1}{\epsilon}$. We note that a similar assumption about a constant number of actions was needed for the similar result in the original paper on complete information tree-games [6]. At the same time, the exponential dependence on the number of types is less than satisfactory and is in part the motivation for the second half of the paper where arbitrary effects of types on payoffs are disallowed.

Extensions: The results presented in this section assumed that the upstream-pass of Approximate-TreeBNE chose randomly from among the feasible witnesses at each step resulting in the output of a randomly selected ϵ-BNE. By appropriately augmenting the tables computed in the downstream pass of our algorithm, it is possible to identify other kinds of approximate BNE, e.g., those that maximize the expected summed payoff to all agents, or those that maximize the least expected payoff among all the agents. We leave the details of these extensions to future work.

4.2 Continuous Types

Here we consider games where the real-valued types of each player are chosen independently from some interval $T = [t_l\ t_u]$. For such games it is reasonable to require that types have some (usually compactly parameterized) structured effect on the payoffs of agents, for else just the definition of the game could become unbounded. The main question of interest in this section is whether there exist interesting kinds of structure in the effect of types on payoffs that can be exploited for computational efficiency (in addition to exploiting the locality structure already present in tree-games). The hope is that the structured effect of types on payoffs will allow us to search efficiently over some small space of restricted strategies to find approximate BNE.

In particular, the class of payoff functions we consider are those satisfying the *single-crossing condition* (SCC), as defined by Athey [1], based on a relation studied by Milgrom and Shannon [9]. Technically, SCC holds[1] if for all $\overrightarrow{a}_{-x}$, $a_x > a'_x$, and $t_x > t'_x$,

$$R_x(a_x, \overrightarrow{a}_{-x}|t'_x) - R_x(a'_x, \overrightarrow{a}_{-x}|t'_x) \geq (>)0$$
$$\text{implies } R_x(a_x, \overrightarrow{a}_{-x}|t_x) - R_x(a'_x, \overrightarrow{a}_{-x}|t_x) \geq (>)0.$$

In words, the SCC condition requires that the incremental payoff to x from raising its action crosses zero at most once, from below, as a function of type. The key implication of SCC, from our perspective, is that x must have a best-response strategy that is increasing in type. If all agents' payoffs satisfy SCC, moreover, and type distributions are atomless, then the game must have a pure-strategy BNE consisting of these monotone best-response strategies [1].

It follows that for games satisfying SCC, we can restrict attention to what we call *monotone threshold pure strategies*. In a threshold strategy, the agent plays pure actions for all possible types, and each action is assigned to at most one interval in the type space T. Equivalently, in such a strategy each pure action is either not played at all or is played deterministically in one contiguous region of type space and nowhere else. We refer to these as *threshold* strategies because the points in type-space at which the strategy switches to a new pure action constitute thresholds. In a monotone threshold strategy, we further have that $a_x \geq a'_x$, where a_x is the pure action assigned to type t_x and a'_x the action assigned to $t'_x \leq t_x$.

Many different classes of interesting games satisfy SCC, and therefore have BNE in which each agent plays monotone threshold pure strategies. Examples include certain kinds of first-price auctions, all-pay auctions, noisy signaling games, oligopoly games with incomplete information about costs or demand, and several other games of economic interest [1, 10, 13]. In Section *4.2.3*, we illustrate our algorithm with a simple example that has payoffs additive in action and type, directly satisfying SCC.

[1]Athey presents a more general definition that allows that types be affiliated. The version here is a special case for our assumption of independent type distributions.

4.2.1 Threshold-TreeBNE Algorithm

As discussed in Section 4.1, we can specialize the Abstract-TreeBNE algorithm to cover mixed strategies and discrete types by discretizing the mixed-strategy space. Here we specialize the Abstract-TreeBNE algorithm for continuous types by discretizing the type space. Since we need consider only pure actions, no further discretization is required for the strategy space. Our Threshold-TreeBNE algorithm, specified briefly in Figure 5, takes as input an accuracy parameter ϵ that specifies how good an approximation to a BNE we wish to find. As for the case of discrete types we will map ϵ to a discretization parameter τ and show that the resulting algorithm returns as output an ϵ-BNE. As before, the approximation can be made arbitrarily precise with correspondingly greater effort.

The central idea in Threshold-TreeBNE(ϵ) is to discretize the type-interval $T = [t_l\ t_u]$ so that agents are restricted to playing threshold pure strategies in which the thresholds are constrained to take on values of the form $t_l + c\tau \leq t_u$ (where c is some integer and τ is the discretization parameter to be determined by analysis). With d actions, the maximum number of possible thresholds is $d-1$ and therefore with τ-discretization the maximum number of different possible strategies available to an agent is of size $O\left(\left[\frac{1}{\tau}\right]^{d-1}\right)$. This allows the binary valued tables D_{yx} of the Abstract-TreeBNE algorithm to have finite representations of size $O\left(\left[\frac{1}{\tau}\right]^{2(d-1)}\right)$ and the witness lists to have a finite representation of size $O\left(\left[\frac{1}{\tau}\right]^{(d-1)(k-1)}\right)$

Algorithm **Threshold-TreeBNE(ϵ)**
Inputs: Specification of threshold pure-strategy tree-game, and the approximation parameter ϵ
Output: A threshold pure-strategy ϵ-BNE for the game.

Do Abstract-TreeBNE with two change

1. only consider discretized-type (with parameter τ defined in the text) threshold pure strategies in both the downstream and upstream passes
2. the requirement of the best-response to set a 1 in the downstream tables D is weakened to a requirement of ϵ-best response.

Figure 5: The Threshold-TreeBNE algorithm. The two modifications to Abstract-TreeBNE (see Figure 2) required are listed here.

As in the case of Approximate-TreeBNE, the Threshold-TreeBNE makes two modifications to the Abstract-TreeBNE. One modification is to only consider τ-discretized threshold pure strategies in all the steps of Figure 2. This makes all the tables involved in the downstream and upstream passes finitely representable. The second modification is to replace the upstream-BNE semantics of the D tables with

an upstream-ϵ-BNE semantics. This change replaces the requirement of best-response in steps 2(c)i and 2(d)ii of Figure 2 with a requirement of ϵ-best-response. With these two modifications, Threshold-TreeBNE will return an ϵ-BNE. In the next section we show how to compute τ from ϵ and the game parameters, as well as how all the computations involved in our algorithm can be done efficiently.

4.2.2 Analysis

Our main result is that Threshold-TreeBNE will return an ϵ-BNE in threshold pure-strategy tree-games with a running time that scales polynomially in the total number of agents n as well as the approximation parameter $\frac{1}{\epsilon}$. Recalling our assumption of a constant number of actions we will also show that Threshold-TreeBNE is polynomial in the size of the representation.

Theorem 4 *For any $\epsilon > 0$, let*

$$\tau < \frac{\epsilon}{4dk\rho_t},$$

where ρ_t is an upper-bound on the type-density p_T. Then Threshold-TreeBNE will compute an ϵ-BNE for any tree-game of incomplete information that has a threshold pure-strategy BNE. Furthermore, for every exact threshold pure-strategy BNE, the tables and witness lists computed by the algorithm contain an ϵ-BNE that is within τ (in L_1 norm) of this exact BNE. The running time of the algorithm is polynomial in the total number of agents and the accuracy parameter $\frac{1}{\epsilon}$.

Proof Lemma 10 in the Appendix proves that for any $\epsilon > 0$, if $\tau < \frac{\epsilon}{4dk\rho_t}$, then there exists a threshold pure-strategy ϵ-BNE in the search space of the algorithm. The tables and witness lists of Threshold-TreeBNE will represent all the ϵ-BNE that are in the space searched by the algorithm as this property is directly inherited from the original Abstract-TreeBNE algorithm. Thus, Threshold-TreeBNE will return an ϵ-BNE.

The running time of the algorithm is dominated by the time to compute the type-discretized D tables for the downwards pass. Each D table will have at most $O\left(\left[\frac{1}{\tau}\right]^{2(d-1)}\right)$ entries to fill. Consider a node x with child node y and parent nodes $\overrightarrow{w}$ in the tree. By assumption, the maximum number of parents of any node is $k-1$. Computing each entry in the D_{yx} table potentially requires considering all possible joint strategies of $\overrightarrow{w}$. Therefore, the running time of Treshold-TreeBNE is of $O\left(n\left[\frac{1}{\tau}\right]^{(d-1)(k+1)}\right)$ □

So far we have successfully avoided having to describe the size of the representation of a threshold pure-strategy game. Given that the main constraint in such games is in the effect of types on payoffs, we will continue to assume that the payoff for a node must at least consider all possible actions for the neighbors of that node, i.e., the size of the representation would at least be of $O(nd^k)$. Therefore, recalling our assumption of a constant number of actions, we get the following corollary.

Corollary 5 *Threshold-TreeBNE computes an ϵ-BNE for threshold pure-strategy games in time polynomial in the size of the representation and $\frac{1}{\epsilon}$.*

The value of τ specified in the condition of Theorem 4 is required if one wants to guarantee that for every exact threshold pure-strategy BNE, Threshold-TreeBNE computes a threshold pure-strategy ϵ-BNE that is close to it. If the goal is instead to just compute any one ϵ-BNE, then one may be able to use a much larger value of τ with a resulting reduction in the running time of the algorithm. Of course, if one uses too large a value of τ, then Threshold-TreeBNE may return nothing. The following corollary suggests that one can search for an appropriate value of τ by starting with a large value and decreasing it until an ϵ-BNE is found.

Corollary 6 *For all $\epsilon, \tau > 0$ and for any threshold pure-strategy tree-game, Threshold-TreeBNE(ϵ, τ) runs in time polynomial in the number of agents, $\frac{1}{\epsilon}$ and $\frac{1}{\tau}$ and either computes an ϵ-BNE or returns with no answer.*

4.2.3 An Example

We illustrate Threshold-TreeBNE by defining a simple class of games satisfying SCC, and presenting numeric results for a specific instance. Consider n-agent 2-action games in which the actions can intuitively be thought of as "engage" and "null". If the agent chooses to engage it incurs a private cost that subtracts from the payoff obtained as a function of the joint action of all the agents. If instead it chooses the null action, the private cost has no effect. Finally, we assume the types are real-valued and chosen independently for each agent from an atomless distribution over a finite interval $[t_l\ t_u]$. We denote these games as *additive-type* games. An example of such a game would be where the agents can accept or reject a job in a marketplace for jobs and where their type is their private cost for fulfilling a job; they pay the cost only if they accept the job. The expected payoff to an agent x with continuous type t_x when the other agents are playing according to strategy $\overrightarrow{\mu}$ is defined as follows:

$$\begin{aligned} R_x(\text{engage}, \overrightarrow{\mu}_{-x}|t_x) &= R_x(\text{engage}, \overrightarrow{\mu}_{-x}) - t_x \\ R_x(\text{null}, \overrightarrow{\mu}_{-x}|t_x) &= R_x(\text{null}, \overrightarrow{\mu}_{-x}) \end{aligned}$$

It is easy to see that additive-type games respect SCC, and thus the Threshold-TreeBNE algorithm can be used to compute their ϵ-BNE.

Figure 6 shows a specific illustrative example of an additive-type game. Seven agents have been invited to a party and must choose to attend or not. Each agent either likes or dislikes its neighbors in the graph and has no opinion about non-neighbors. The payoff to each agent increases with the number of agents it likes attending and decreases with the number of agents it dislikes attending. There is a private cost for each agent that attends in the range $[-1\ 1]$.

We ran Threshold-TreeBNE for an ϵ value of 0.03 for various values of τ as discussed at the end of Section 4.2.2. For τ equal to 0.5 the algorithm found two equilibria (shown in Table 1 as Equilibrium 1 and Equilibrium 2). For τ equal to 0.25 the algorithm found all three equilibria shown in Table 1. The payoff matrices for our game were chosen to set up an interesting dynamic between agents 2 and 4. Agent 2 has a positive reward for not going but it really likes agent 4 and so if agent 4 goes it makes it more likely that agent 2 will also go (depending on what its other neighbors do). This dynamic is reflected in the threshold pure-strategy equilibria of Table 1.

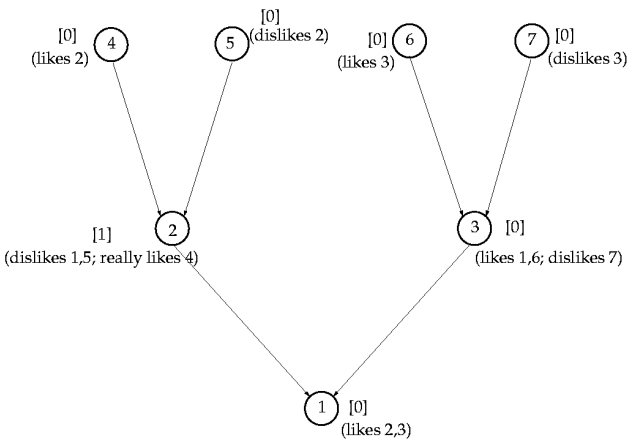

Figure 6: The above graph details the "party" game. Seven players have been invited to a party and must decide whether to attend. Each player has a like or dislike of their neighbors on the graph (written next to the node, e.g., agent 2 dislikes agents 1 and 5 but likes 4). Attending the party incurs a private cost to an agent chosen uniformly at random from the interval $[-1\ 1]$. The payoff to a player for not attending is independent of the actions of the other players and is shown next to the nodes in square brackets (e.g., agent 2 has payoff of 1 for not attending). An attendee's payoff is incremented for every agent she likes at the party and is decremented for every agent she dislikes at the party.

5. EXTENSIONS

There are a number of extensions to this paper, briefly itemized here, that we will explore as future work.

- Arbitrary graphical games can be converted into tree games by merging nodes into cluster-nodes. We will extend both Approximate-TreeBNE and Threshold-TreeBNE to sparsely connected graphical games albeit at the expense of exponential computation in the maximal size of the cluster-nodes.
- Ortiz and Kearns [12] showed that an *iterative* version of the message passing algorithm for tree-games of complete information worked in practice on complete-information loopy graphical games. A similar extension of our algorithms may generalize them to graphical games of incomplete information.

$\epsilon = 0.03$	Equilibrium 1	Equilibrium 2	Equilibrium 3
Agent 1	go	go	go
Agent 2	not go	$t < 0.5$, go	$t < 0.25$, go
Agent 3	go	go	go
Agent 4	$t < 0.5$, go	go	go
Agent 5	go	$t < 0.5$ go	$t < 0.75$, go
Agent 6	go	go	go
Agent 7	$t < 0.5$, go	$t < 0.5$, go	$t < 0.5$, go

Table 1: Results for party game of Figure 6.

- Dynamic graphical games are extensions of static graphical games in which the agents actions influence not just the payoffs but also the stochastic evolution of the dynamics of a system. Extending our algorithms to such games poses a significant challenge and is the main focus of our current research on algorithms for graphical games.

6. CONCLUSIONS

In this paper we considered two interesting and large classes of incomplete information games: tree-games with discrete types, and tree-games with continuous types that satisfy the single crossing condition and are thus guaranteed to have threshold pure-strategy BNE. Both of these classes contain many interesting kinds of games. We presented analyses that showed that our algorithms for the two cases compute ϵ-BNE in time polynomial in the number of agents and the approximation parameter $\frac{1}{\epsilon}$.

7. REFERENCES

[1] S. Athey. Single crossing properties and the existence of pure strategy equilibria in games of incomplete information. *Econometrica*, 69:861–869, 2001.

[2] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *Eighteenth International Joint Conference on Artificial Intelligence*, pages 765–771, Acapulco, 2003.

[3] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[4] J. Harsanyi. Games of incomplete information played by Bayesian players. *Management Science*, 14:159–182, 320–334, 486–502, 1967.

[5] S. Kakade, M. Kearns, J. Langford, and L. Ortiz. Correlated equilibria in graphical games. In *Fourth ACM Conference on Electronic Commerce*, pages 42–47, San Diego, 2003.

[6] M. Kearns, M. L. Littman, and S. Singh. Graphical models for game theory. In *Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 253–260, Seattle, 2001.

[7] K. Leyton-Brown and M. Tennenholtz. Local-effect games. In *Eighteenth International Joint Conference on Artificial Intelligence*, pages 772–777, Acapulco, 2003.

[8] M. L. Littman, M. Kearns, and S. Singh. An efficient, exact algorithm for solving tree-structured graphical games. In *Advances in Neural Information Processing Systems 14*, pages 817–823, 2001.

[9] P. Milgrom and C. Shannon. Monotone comparative statics. *Econometrica*, 62:157–180, 1994.

[10] P. Milgrom and R. Weber. Distributional strategies for games with incomplete information. *Mathematics of Operations Research*, 10:619–632, 1985.

[11] J. F. Nash. Non cooperative games. *Annals of Mathematics*, 54:286–295, 1951.

[12] L. E. Ortiz and M. Kearns. Nash propagation for loopy graphical games. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 793–800. MIT Press, Cambridge, MA, 2003.

[13] D. M. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 2002.

[14] D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *Eighteenth National Conference on Artificial Intelligence*, pages 345–351, Edmonton, 2002.

APPENDIX

Lemma 7 (From Kearns et.al. [6]) *Let $\overrightarrow{p}, \overrightarrow{q} \in [0,1]^k$ satisfy $|p_i - q_i| < \tau$ for all $1 \le i \le k$, and let $\tau \le 2/(k \log^2(k/2))$. Then $|\Pi_{i=1}^k p_i - \Pi_{i=1}^k q_k| \le (2k \log k)\tau$.*

Lemma 8 *If $\overrightarrow{\mu}$ and $\overrightarrow{\mu'}$ are such that for all agents x and all types t_x, $|\mu_x(\cdot|t_x) - \mu'_x(\cdot|t_x)| \le \tau$, then $|R_x(\overrightarrow{\mu}_{\mathcal{N}^x}) - R_x(\overrightarrow{\mu'}_{\mathcal{N}^x})| \le d^k(2k \log k)\tau$.*

Proof Recall that

$$R_x(\overrightarrow{\mu}_{\mathcal{N}^x}) = \sum_{\overrightarrow{t}_{\mathcal{N}^x}} P_T(\overrightarrow{t}_{\mathcal{N}^x}) \sum_{\overrightarrow{a}_{\mathcal{N}^x}} \overrightarrow{\mu}_{\mathcal{N}^x}(\overrightarrow{a}_{\mathcal{N}^x}|\overrightarrow{t}_{\mathcal{N}^x}) R_x(\overrightarrow{a}_{\mathcal{N}^x})$$

and therefore

$$\begin{aligned} |R_x(\overrightarrow{\mu}_{\mathcal{N}^x}) - R_x(\overrightarrow{\mu'}_{\mathcal{N}^x})| &= \sum_{\overrightarrow{t}_{\mathcal{N}^x}} P_T(\overrightarrow{t}_{\mathcal{N}^x}) \sum_{\overrightarrow{a}_{\mathcal{N}^x}} |R_x(\overrightarrow{a}_{\mathcal{N}^x})| \\ &\quad \left| \left[\overrightarrow{\mu}_{\mathcal{N}^x}(\overrightarrow{a}_{\mathcal{N}^x}|\overrightarrow{t}_{\mathcal{N}^x}) - \overrightarrow{\mu}_{\mathcal{N}^x}(\overrightarrow{a}_{\mathcal{N}^x}|\overrightarrow{t}_{\mathcal{N}^x}) \right] \right| \\ &\le \sum_{\overrightarrow{t}_{\mathcal{N}^x}} P_T(\overrightarrow{t}_{\mathcal{N}^x}) \sum_{\overrightarrow{a}_{\mathcal{N}^x}} |R_x(\overrightarrow{a}_{\mathcal{N}^x})| (2k \log k)\tau \\ &\le d^k(2k \log k)\tau \end{aligned}$$

where we have used the assumption that the payoff function is bounded in absolute value by 1. □

Lemma 9 *Let $\overrightarrow{\mu}$ be a BNE for the tree-game G, and let $\overrightarrow{\mu'}$ be the nearest (in L_1 metric) strategy on the τ-grid. Then provided $\tau \le 2/(k \log^2(k/2))$, $\overrightarrow{\mu'}$ is a $d^k(4k \log k)\tau$-BNE of the tree-game G.*

Proof Let strategy η be a best-response for agent x to $\overrightarrow{\mu'}$. To obtain our result we need to bound $R_x(\eta, \overrightarrow{\mu'}_{\mathcal{N}^x_{-x}}) - R_x(\overrightarrow{\mu'}_{\mathcal{N}^x}) > 0$. By Lemma 8, we have that

$$|R_x(\eta, \overrightarrow{\mu'}_{\mathcal{N}^x_{-x}}) - R_x(\eta, \overrightarrow{\mu}_{\mathcal{N}^x_{-x}})| \le d^k(2k \log k)\tau.$$

Since $\overrightarrow{\mu}$ is a BNE, $R_x(\overrightarrow{\mu}_{\mathcal{N}^x}) \ge R_x(\eta, \overrightarrow{\mu}_{\mathcal{N}^x_{-x}})$. Thus,

$$R_x(\eta, \overrightarrow{\mu'}_{\mathcal{N}^x_{-x}}) \le R_x(\overrightarrow{\mu}_{\mathcal{N}^x}) + d^k(2k \log k)\tau.$$

At the same time, again by Lemma 8,

$$R_x(\overrightarrow{\mu'}_{\mathcal{N}^x}) \ge R_x(\overrightarrow{\mu}_{\mathcal{N}^x}) - d^k(2k \log k)\tau,$$

and thus $R_x(\eta, \overrightarrow{\mu'}_{\mathcal{N}^x_{-x}}) - R_x(\overrightarrow{\mu'}_{\mathcal{N}^x}) \le d^k(4k \log k)\tau$. □

Lemma 10 *If a threshold pure-strategy BNE exists for a tree-game, G, of incomplete information in which each agent's real-valued type is chosen independently from the interval $T = [t_l\ t_u]$ according to some probability density p_T, then there exists a threshold pure-strategy ϵ-BNE if we discretize the types with grid-spacing $\tau \le \frac{\epsilon}{4dk\rho_t}$ (where k is the maximum neighborhood size, and ρ_t is an upper-bound on the density p_T).*

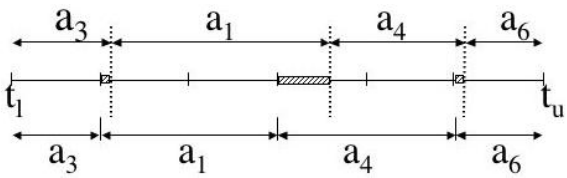

Figure 7: An example of the effect of discretization of types on a threshold pure strategy. The strategy on top is an undiscretized strategy while the strategy at the bottom is its discretized version. The two strategies differ on the intervals shown as hashed-rectangles.

Proof Let $\overrightarrow{\mu}$ be a threshold pure-strategy BNE for the game G in the statement of the lemma. Let $\overrightarrow{\mu}'$ be the nearest (in L_1 metric) strategy on the τ-grid over types; this is defined by mapping each threshold to the nearest grid point in types and preserving the pure-actions in the associated τ-perturbed intervals. Figure 7 shows a pictorial example of mapping $\overrightarrow{\mu}$ to $\overrightarrow{\mu'}$ — the strategy shown on top is $\overrightarrow{\mu}$ and the one at the bottom is $\overrightarrow{\mu'}$. We will show that if τ satisfies the condition in the lemma, $\overrightarrow{\mu}'$ is a ϵ-BNE.

For any threshold pure strategy $\overrightarrow{\theta}$, let $\overrightarrow{\theta}'$ be its nearest pure strategy on the τ-grid over types. Then, for any agent x,

$$\begin{aligned} |R_x(\overrightarrow{\theta}_{\mathcal{N}^x}) - R_x(\overrightarrow{\theta}'_{\mathcal{N}^x})| &\le \\ E_{\overrightarrow{t}_{\mathcal{N}^x}} |R_x(\overrightarrow{a}_{\mathcal{N}^x}|\overrightarrow{\theta}_{\mathcal{N}^x}, \overrightarrow{t}_{\mathcal{N}^x}) - R_x(\overrightarrow{a}_{\mathcal{N}^x}|\overrightarrow{\theta}'_{\mathcal{N}^x}, \overrightarrow{t}_{\mathcal{N}^x})| & \\ \le \quad 2d\tau\rho_t k & \qquad (7) \end{aligned}$$

where ρ_t is the maximum density at any type, d is the number of actions, and k is the size of the largest neighborhood in tree-game G. The last step of Equation 7 follows from the fact that there is no difference in the joint actions for the two strategies except for joint types in which at least one agent falls into a region resulting from the perturbed thresholds, and that the probability that at least one of the k neighboring-agents falls into such a region is upper-bounded by $d\tau\rho_t k$ (we also assume, w.l.o.g., that the maximum absolute payoff is bounded by 1). The intervals in type-space in which the discretized version of a strategy differs from the original strategy are shown with hashed-rectangles in Figure 7.

To obtain the result, we need to bound $|R_x(\eta_x, \overrightarrow{\mu}'_{\mathcal{N}^x_{-x}}) - R_x(\overrightarrow{\mu}'_{\mathcal{N}^x})|$ for any strategy η_x for agent x. We know from Equation 7 that

$$|R_x(\eta_x, \overrightarrow{\mu}'_{\mathcal{N}^x_{-x}}) - R_x(\eta_x, \overrightarrow{\mu}_{\mathcal{N}^x_{-x}})| \le 2d\tau\rho_t k$$

which implies that

$$\begin{aligned} R_x(\eta_x, \overrightarrow{\mu}'_{\mathcal{N}^x_{-x}}) &\le R_x(\eta_x, \overrightarrow{\mu}_{\mathcal{N}^x_{-x}}) + 2d\tau\rho_t k \\ \text{which because } \overrightarrow{\mu} \text{ is a BNE} &\implies \\ R_x(\eta_x, \overrightarrow{\mu}'_{\mathcal{N}^x_{-x}}) &\le R_x(\overrightarrow{\mu}_{\mathcal{N}^x}) + 2d\tau\rho_t k \\ \text{which from equation 7} &\implies \\ R_x(\eta_x, \overrightarrow{\mu}'_{\mathcal{N}^x_{-x}}) &\le R_x(\overrightarrow{\mu}'_{\mathcal{N}^x}) + 4d\tau\rho_t k \end{aligned}$$

We get τ by setting $4d\tau\rho_t k \le \epsilon$. □