Computing Pure Nash Equilibria in Graphical Games via Markov Random Fields

Constantinos Daskalakis[*]
University of California, Berkeley
costis@cs.berkeley.edu

Christos H. Papadimitriou[†]
University of California, Berkeley
christos@cs.berkeley.edu

ABSTRACT

We present a reduction from graphical games to Markov random fields so that pure Nash equilibria in the former can be found by statistical inference on the latter. Our result, when combined with the junction tree algorithm for statistical inference, yields a unified proof of all previously known tractable cases of the NP-complete problem of finding pure Nash equilibria in graphical games, but also implies efficient algorithms for new classes, such as the games with $O(\log n)$ treewidth. Furthermore, this important problem becomes susceptible to a wealth of sophisticated and empirically successful techniques from Machine Learning.

Categories and Subject Descriptors

F.2.0 [Analysis of Algorithms and Problem Complexity]:General

General Terms

Algorithms, Economics, Theory

Keywords

Nash Equilibrium, Markov Random Fields, Treewidth

1. INTRODUCTION

Several recent results explore the complexity of finding mixed Nash [9, 13, 10, 7, 8] and correlated [29] equilibria in games. Since the description complexity of n-person games is exponential in n, much of this work focuses on succinct representations, of which *graphical games* proposed by Kearns et al. [18] (agents affecting each other's utility along the links of a network) is perhaps the most important.

The main advantage of mixed Nash and correlated equilibria is that they are guaranteed to exist in any game; on the negative side, they are rather unconvincing and fragile as models of behavior and rationality, involving deliberate randomization by players, while their justification is based on the assumptions that all players are risk neutral, and practice the same sophisticated, multi-staged analysis based on common knowledge; these problems are intensified when a large number of players is present. See, e.g., [28, 21] for extensive critical presentation of, and comparison between, these and other concepts of rationality.

In contrast, the *pure Nash equilibrium* is a very primitive, intuitive, straightforward and relatively uncontroversial concept of rationality; on the negative side, it does not exist in all games, and in fact telling if one exists is an NP-complete problem in graphical games [14] (the problem is, of course, computationally trivial in normal form games). *In this paper we present a new class of algorithms for finding pure Nash equilibria in graphical games.*

Besides economy of description, one of the motivations for the introduction of graphical games is their intuitive affinity with graphical statistical models; indeed, several algorithms for graphical games (e.g., [24, 11, 27, 17]) do have the flavor of algorithms for Bayes nets. *Our main result is a mapping from any graphical game to a Markov random field (MRF)* such that finding a maximum a posteriori configuration of the MRF (a central problem in MRF theory [22]) answers the question of whether the graphical game has a pure Nash equilibrium. Furthermore, the marginal probability distributions of the cliques of the MRF constitute a succinct description of all pure Nash equilibria of the graphical game (such description is necessary because there might be exponentially many pure Nash equilibria). In view of the intuitive similarity between graphical games and MRFs, our reduction is not, of course, completely unexpected. However, it has to overcome a number of subtleties and pay much attention to detail, while the proof that it works is quite nontrivial.

Furthermore, by applying the *junction tree algorithm* [23, 16] (see Section 4) to the output of our reduction we show that for large classes of graphical games we can compute in polynomial time a succinct description of the set of pure Nash equilibria; these include all previously known efficient algorithms [14]), as well as a new one for graphical games with $O(\log n)$-treewidth, bounded neighborhood size and bounded cardinality strategy sets — a class which we believe is a plausible model of networked markets. This is the first polynomial algorithm for pure Nash equilibria that is not based on some assumption about the cycle structure of the graph.

But perhaps what is even more interesting about our reduction is that it brings to bear in the computation of pure Nash equilibria all of the large and growing number of statistical inference algorithms (belief propagation [30] and generalized belief propagation [34], variational methods [33], Markov Chain Monte Carlo [26], simulated annealing [19], survey propagation [6] etc.), many of which are empirically very successful albeit without worst-case guarantees; these opportunities are described in Section 4.4. Whether

[*]Research supported by NSF ITR grant CCR-0121555 and a grant from Microsoft Research.

[†]Research supported by NSF ITR grant CCR-0121555 and a grant from Microsoft Research.

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'06, June 11–15, 2006, Ann Arbor, Michigan, USA.
Copyright 2006 ACM 1-59593-236-4/06/0006 ...$5.00.

these lead to experimentally competitive algorithms for Nash equilibria is an interesting research direction. Some initial experimental results are described in Section 5.

Finally, we sketch how the same reduction can be used for the computation of *approximate mixed* Nash equilibria by using ideas similar to [18]. The resulting algorithms run in time exponential in the game description and the desired approximation —which is not surprising given that even the 2-strategies-per-player, 4-sized neighborhood case was shown to be PPAD-complete in [9]. On the other hand, by further restricting the treewidth to $O(\log n)$, the algorithm becomes polynomial in the game description and exponential in the approximation guarantee. These possibilities are explored in Section 4.3

2. PRELIMINARIES

2.1 Games

In a *game* we have *n players*, $1, \ldots, n$. Each player p, $1 \le p \le n$, has a finite set of *strategies* or *choices*, S_p, with $|S_p| \ge 2$, and a *payoff* function $u_p : \prod_{i=1}^n S_i \to \mathbb{N}$. The set $S = \prod_{i=1}^n S_i$ is called *set of strategy profiles* and we denote the set $\prod_{i \neq p} S_i$ by S_{-p}.

In order to specify a game with n players and s strategies each, we need ns^n numbers, an amount of information exponential in the number of players. However, players often interact with a limited number of other players, and this allows for much more succinct representations:

Definition 2.1 A *graphical game* $\mathcal{G} = \langle G, \{S_p\}, \{u_p\} \rangle$ is defined by:

- An undirected graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the set of players.
- For every player $p \in V$:
  - A non-empty finite set of *strategies* S_p
  - A *payoff function* $u_p : \prod_{i \in \mathcal{N}(p)} S_i \to \mathbb{N}$ (where $\mathcal{N}(p) = \{p\} \cup \{v \in V | (p, v) \in E\}$)

A different perspective of graphical games, which will be useful later, is through the *hypergraph* they induce. We can imagine that every game $\mathcal{G}$ defines a hypergraph having as nodes the players and, for every player p, one hyperedge containing p's neighborhood, $\mathcal{N}(p)$; remove duplicate hyperedges if two or more players have the same neighborhood. We denote this hypergraph by $\mathcal{H}(\mathcal{G})$ and we define as *primal graph of the game* the primal graph[*] of $\mathcal{H}(\mathcal{G})$.

Consider a game with n players and strategy sets $S_1, \ldots, S_n$. For every strategy profile $s \in S$, we denote by s_p the strategy of player p in this strategy profile and by s_{-p} the $(n-1)$-tuple of strategies of all players but p. For every $s'_p \in S_p$ and $s_{-p} \in S_{-p}$ we denote by $(s_{-p}; s'_p)$ the strategy profile in which player p plays s'_p and all the other players play according to s_{-p}.

Definition 2.2 (Pure Nash Equilibrium)
A strategy profile s is a *pure Nash equilibrium* if for every player p and strategy $t_p \in S_p$ we have $u_p(s) \ge u_p(s_{-p}; t_p)$.

[*]The primal graph $G' = (V', E')$ of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ has $V' = \mathcal{V}$ and two nodes $v_1, v_2 \in V'$ are connected iff there is a hyperedge $h \in \mathcal{E}$ such that $v_1, v_2 \in h$.

Intuitively, a strategy profile s is a pure Nash equilibrium if none of the players has a unilateral incentive to deviate: the player cannot increase his/her payoff by deviating to another strategy if the other players continue to play the strategies in s_{-p}.

Definition 2.3 (Best Response Function) The *Best Response Function of player p* is a function $\mathrm{BR}_{u_p} : S_{-p} \to 2^{S_p}$ defined by:

$$\mathrm{BR}_{u_p}(s_{-p}) \triangleq \{s_p | s_p \in S_p \text{ and } \forall s'_p \in S_p : u_p(s_{-p}; s_p) \ge u_p(s_{-p}; s'_p)\}$$

Intuitively, $\mathrm{BR}_{u_p}(s_{-p})$ is the set of strategies in S_p that maximize p's payoff if the other players play s_{-p}.

Thus, a strategy profile s is a pure Nash equilibrium if, for every player p, $s_p \in \mathrm{BR}_{u_p}(s_{-p})$.

2.2 Markov Random Fields

MRFs: We describe informally the notion of an undirected graphical model and we refer the reader to [22] for a more detailed description. An ***undirected graphical model*** or ***Markov random field (MRF)*** over an undirected graph $G = (V, E)$, $|V| = n$, is a probability distribution that factorizes according to functions defined on a set $\mathcal{C}$ of cliques of G. More precisely, associated with every node $v \in V$ is a random variable x_v taking values from a finite set $\mathcal{X}_v$ of values. Also, associated with every clique $c \in \mathcal{C}$ is a potential function $\psi_c : \prod_{v \in c} \mathcal{X}_v \to \mathbb{R}_+$ that depends only on $x_c = \{x_v | v \in c\}$. Using this notation the probability distribution defined on $\mathbf{x} = \{x_v | v \in V\}$ is:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x_c) \qquad (1)$$

where Z is a normalizing constant. We'll refer to $\mathcal{C}$ as the *set of significant cliques of the MRF*.

Inference problems. MRFs enable the formulation and solution of important statistical inference problems:

1. *Maximum-a-posteriori (MAP) estimation* is the problem of finding a configuration that is most likely under the distribution $p(\mathbf{x})$, or, more formally, $\hat{x}_{MAP} \in \arg\max_{x \in \prod_v \mathcal{X}_v} p(\mathbf{x})$.
2. *Computing the marginal probability distribution* of a particular subset of the nodes or some subsets of the nodes simultaneously; usually the marginal probability distributions of the cliques of set $\mathcal{C}$.
3. *Sampling* the distribution $p(\mathbf{x})$.

A crucial observation about the normalizing constant Z: In order to compute Z one has to sum $p(x)$ over all configurations $x \in \mathcal{X} = \prod_{v \in V} \mathcal{X}_v$, a computation that would require exponential time in the number of the nodes. However, this is not really needed for the above inference problems. Indeed, computing a MAP configuration does not change whether we include Z in the computation or not. Also, sampling the distribution is usually based on the ratio of probabilities of configurations (e.g. in the Metropolis-Hastings sampling method) and so Z is canceled. Finally, computing marginal probability distributions involves summing $p(x)$; this can be done without including constant Z and the resulting function can be normalized after the completion of the algorithm; now, since the marginal is usually computed on a subset of few nodes, the time needed to normalize the computed function is much less than that required to find Z. Thus, a very common practice in statistical inference is to assume **Z=1** for all computations.

2.3 The Junction Tree Algorithm

The junction tree algorithm is one of the most celebrated algorithms for statistical inference and is used both for computing marginal distributions and for computing maximum-a-posteriori configurations. It is also in the core of numerous other algorithms for MRFs and Bayesian Networks. For a quick description of the algorithm we refer the reader to [33]; a detailed description can be found in [16, 23]. Here we describe briefly some ingredients of the algorithm. We need a definition:

Definition 2.4 A ***clique tree of a chordal graph*** $G = (V, E)$ is a tree $T = (\mathcal{C}, \mathcal{E})$, where $\mathcal{C}$ is a subset of the set of all cliques of graph G, that has the following properties:

- $\forall$ clique c of G, $\exists c' \in \mathcal{C}$ s.t. $c \subseteq c'$ (thus all maximal cliques of G are nodes of T)
- $\forall c_1, c_2 \in \mathcal{C}, \forall c_3 \in \mathcal{C}$ in the unique path between c_1 and c_2, $c_1 \cap c_2 \subseteq c_3$ (clique intersection property)

The **width** of T is $max_{c \in \mathcal{C}}\{|c|\}$.

Definition 2.5 A ***clique tree of a graph*** $G = (V, E)$ is a clique tree of some triangulation of G.

Lemma 2.6 (e.g. [20]) *Every clique tree of a graph G is a tree decomposition and vice versa. Thus, the treewidth of a graph G is equal to the minimum width of all clique trees of G minus 1.*

Briefly, the junction tree algorithm starts from the graph $G = (V, E)$ of the MRF and triangulates it to get a chordal graph $G' = (V, E')$. Then it builds a clique tree $T = (\mathcal{C}, \mathcal{E})$ of G' and it loads to every node of T a potential function as follows: it assigns the potential function of every significant clique of the MRF to exactly one of the nodes of T that contain it and then for every node of T it takes the product of the potential functions that were assigned to it; if a node of T has no significant cliques assigned to it then its potential function is taken identically equal to one. After doing so, it performs calculations on T and computes by "message passing" the marginal probability distributions of every clique $c \in \mathcal{C}$. The marginal distributions of the significant cliques of the MRF are derived from the marginal distributions of the cliques of set $\mathcal{C}$ by summations. It is well-known that a simple variant of the same algorithm can be used for computing maximum a posteriori configurations (see for example discussion in [33]).

The single non-standard step of the algorithm is the triangulation of G. Different triangulations lead to different clique trees. However, the sizes of the cliques in the clique tree determine the running time of the algorithm, which is proportional to $\sum_{c \in \mathcal{C}} \prod_{v \in c} |\mathcal{X}_v|$. Note that finding the triangulation that leads to the lightest clique tree under this objective function is an NP-hard optimization problem (see e.g. [3] and the references therein) and there are various algorithms for building "good" clique-trees. For an overview of these algorithms see for example [3, 5]. If $|\mathcal{X}_v| = \chi, \forall v \in V$, then the running time of the algorithm is $O(n \cdot \chi^{width(T)})$, which is at least $O(n \cdot \chi^{treewidth(G)+1})$.

3. THE REDUCTION

Given a graphical game $\mathcal{G} = \langle G = (V, E), \{S_p\}, \{u_p\}\rangle$ we construct the following Markov random field:

1. The underlying graph of the MRF is the primal graph $G' = (V', E')$ of $\mathcal{H}(\mathcal{G})$.

2. Associated with every node $p \in V'$ is a random variable with state space $\mathcal{X}_p = S_p$, the strategy set of the corresponding player.

3. For every player $p \in V'$, its neighborhood, $\mathcal{N}(p)$, is a clique c_p in G'. Note that there might be two players $p_1 \neq p_2$ with $c_{p_1} = c_{p_2}$. The set of significant cliques of the MRF is $\mathcal{C} = \bigcup_{p \in V'}\{c_p\}$.

4. We assign to every player p a function $f_p : \prod_{p' \in c_p} \mathcal{X}_{p'} \to \mathbb{R}_+$ that is defined as follows (call $x(c_p)$ the vector of the random variables corresponding to the players of set c_p):
$$f_p\left(x(c_p)\right) = \begin{cases} 1, & \text{if } x(c_p)_p \in \text{BR}_{u_p}\left(x(c_p)_{-p}\right) \\ \epsilon, & \text{otherwise} \end{cases}$$
where $\epsilon < 1$ is a small constant to be fixed later. Intuitively, the function $f_p\left(x(c_p)\right)$ maps a selection of strategies $x(c_p)$ for the players of the set $\mathcal{N}(p)$ to 1 if the strategy $x(c_p)_p$ of player p is a best response to the strategies $x(c_p)_{-p}$ of his neighbors and to a small constant ϵ otherwise.

5. We assign to every clique $c \in \mathcal{C}$ a potential function $\psi_c : \prod_{p \in c} \mathcal{X}_p \to \mathbb{R}_+$ defined as follows:
$$\psi_c\left(x(c)\right) = \prod_{p \in V' : c_p = c} f_p(x(c))$$

6. Since the Markov random field that we defined is parameterized on the choice of ϵ we will refer to it as **MRF**$(\mathcal{G}, \epsilon)$. Also, we will refer to the unnormalized (Z=1) probability distribution defined on MRF$(\mathcal{G}, \epsilon)$ as $p_\epsilon(x)$; i.e. $p_\epsilon(x) = \prod_{c \in \mathcal{C}} \psi_c(x(c))$.

We claim that the construction has many useful properties. First, for any $\epsilon < 1$, finding a MAP configuration, $\hat{x}_{MAP}$, of MRF$(\mathcal{G}, \epsilon)$ answers the question of whether the graphical game $\mathcal{G}$ has a pure Nash equilibrium. This fact is stated by the following lemmas (call $\mathcal{X} = \prod_v \mathcal{X}_v$).

Lemma 3.1
$\forall \epsilon < 1$: *($\mathcal{G}$ has a pure Nash Equilibrium)*
$\Leftrightarrow (\max_{x \in \mathcal{X}}\{p_\epsilon(x)\} = 1)$

PROOF. ($\Rightarrow$) We have:

$$\begin{aligned} & x \text{ is a pure Nash equilibrium of } \mathcal{G} \\ & \Rightarrow \forall p \in V' : x(c_p)_p \in \text{BR}_{u_p}\left(x(c_p)_{-p}\right) \\ & \Rightarrow \forall p \in V' : f_p(x(c_p)) = 1 \\ & \Rightarrow \forall c \in \mathcal{C} : \psi_c(x(c)) = 1 \\ & \Rightarrow p_\epsilon(x) = 1 \\ & \Rightarrow x \in \arg\max_{x \in \mathcal{X}}\{p_\epsilon(x)\} \\ & \qquad (\text{ since by definition } p_\epsilon(x) \leq 1, \forall x) \\ & \Rightarrow \max_{x \in \mathcal{X}} p_\epsilon(x) = 1 \end{aligned}$$

($\Leftarrow$) We have:

$$\begin{aligned}
&\mathcal{G} \text{ has no pure Nash equilibria} \Rightarrow\\
&\nexists x \in \mathcal{X} \text{ s.t. } \forall p \in V' : x(c_p)_p \in \text{BR}_{u_p}\left(x(c_p)_{-p}\right) \Rightarrow\\
&\nexists x \in \mathcal{X} \text{ s.t. } \forall p \in V' : f_p(x(c_p)) = 1 \overset{\epsilon \leq 1}{\Rightarrow}\\
&\nexists x \in \mathcal{X} \text{ s.t. } \forall c \in \mathcal{C} : \psi_c(x(c)) = 1 \Rightarrow\\
&\nexists x \in \mathcal{X} \text{ s.t. } p_\epsilon(x) = 1 \Rightarrow\\
&\max_{x \in \mathcal{X}} p_\epsilon(x) < 1\\
&(\text{ since by definition } p_\epsilon(x) \leq 1, \forall x)
\end{aligned}$$

□

Lemma 3.2
$\forall \epsilon < 1$*: If* $p_\epsilon(\hat{x}) = 1$ *for some* $\hat{x}$*, then* $\hat{x} \in \arg\max_{x \in \mathcal{X}} \{p_\epsilon(x)\}$ *and* $\hat{x}$ *is a pure Nash equilibrium of* $\mathcal{G}$*.*

PROOF. If $p_\epsilon(\hat{x}) = 1$ then $\hat{x} \in \arg\max_{x \in \mathcal{X}} p_\epsilon(x)$ since $p_\epsilon(x) \leq 1$ by definition ($\epsilon < 1$). Also:

$$\begin{aligned}
&p_\epsilon(\hat{x}) = 1 \overset{\epsilon \leq 1}{\Leftrightarrow}\\
&\forall c \in \mathcal{C} : \psi_c(\hat{x}(c)) = 1 \overset{\epsilon \leq 1}{\Leftrightarrow}\\
&\forall p \in V' : f_p(\hat{x}(c_p)) = 1 \Leftrightarrow\\
&\forall p \in V' : \hat{x}(c_p)_p \in \text{BR}_{u_p}\left(\hat{x}(c_p)_{-p}\right) \Leftrightarrow\\
&\hat{x} \text{ is a pure Nash equilibrium}
\end{aligned}$$

□

Moreover, computing the unnormalized marginal distributions of the significant cliques of MRF$(\mathcal{G}, 0)$ answers the question of whether the graphical game has a pure Nash equilibrium and, at the same time, the unnormalized marginal distributions constitute a succinct description of all pure Nash equilibria of the graphical game[†]. This is stated by the following lemmas.

Lemma 3.3 *($\mathcal{G}$ has a pure Nash Equilibrium)* $\Leftrightarrow$
($\forall$ clique c of MRF$(\mathcal{G}, 0)$, $\exists x^*(c)$ s.t. $p_{0,c}(x^*(c)) \neq 0$), where $p_{0,c}(x(c))$ is the unnormalized marginal probability distribution of clique c.

PROOF. The marginalization of $p_0(x)$ with respect to a clique c of MRF$(\mathcal{G}, 0)$ is simply a summation $p_{0,c}(x(c)) = \sum_{x_v, v \notin c} p_0(x)$. It is easy to prove the claim as follows:
($\Rightarrow$) If $\mathcal{G}$ has a pure Nash equilibrium x^* then $p_0(x^*) = 1$ (see proof of Lemma 3.2). It follows that $p_{0,c}(x^*(c)) > 0$ for every clique c of the MRF.
($\Leftarrow$) Proof by contradiction:

$$\begin{aligned}
&\mathcal{G} \text{ does not have a pure Nash equilibrium}\\
&\Rightarrow \forall x, \exists p \text{ s.t. } f_p(x(c_p)) = \epsilon\\
&\overset{\epsilon = 0}{\Rightarrow} \forall x, p_0(x) = 0\\
&\Rightarrow \forall \text{ clique } c, \forall x(c) : p_{0,c}(x(c)) = 0
\end{aligned}$$

□

Lemma 3.4 *If $p_{0,c}(x^*(c)) \neq 0$ for some clique c of MRF$(\mathcal{G}, 0)$ and some $x^*(c)$, then $\exists$ a pure Nash equilibrium x^+ of $\mathcal{G}$ such that $x^+(c) = x^*(c)$.*

[†]A *succinct description* of the set of all pure Nash equilibria (or any other object) x is a string y such that $|y|$ is polynomial in the description of the game and $x = f(y)$ for some function f computable in time polynomial in $|x| + |y|$.

PROOF. (By contradiction)

$$\begin{aligned}
&\nexists \text{ pure Nash equilibrium } x^+ \text{ of } \mathcal{G} \text{ such that } x^+(c) = x^*(c)\\
&\Rightarrow \forall x \text{ with } x(c) = x^*(c) : p_0(x) = 0\\
&\Rightarrow p_{0,c}(x^*(c)) = \sum_{x : x(c) = x^*(c)} p_0(x) = 0
\end{aligned}$$

□

Based on Lemmas 3.3 and 3.4 one can build a dynamic programming algorithm that takes as input the marginal probability distributions of the significant cliques of the MRF and outputs all pure Nash equilibria in output polynomial time.

Also, a useful remark is the following. From the proofs of Lemmas 3.3 and 3.4 it follows that if a game $\mathcal{G}$ does not have a pure Nash equilibrium then in MRF$(\mathcal{G}, 0)$ function $p_0(x)$ is identically zero. In this case, of course, $p_0(x)$ cannot be a probability distribution of a Markov Random Field. Since MRFs are defined in a distributed fashion, i.e. by potential functions on subsets of the nodes, this is an intrinsic problem in statistical inference, and statistical inference algorithms are designed to handle this possibility.

4. ALGORITHMIC CONSEQUENCES

In view of this reduction, one can use all the algorithmic machinery developed for statistical inference in graphical models; any statistical inference algorithm that computes marginal distributions, maximum-a-posteriori configurations or samples distributions defined on Markov random fields works for finding pure Nash equilibria. In this section we combine our reduction with one particular algorithm for statistical inference, the junction tree algorithm explained in Subsection 2.3, to derive polynomial time algorithms for checking the existence of pure Nash equilibria and for computing a succinct description of all pure Nash equilibria for large classes of graphical games.

We note that the algorithms we derive are essentially combinatorial and the following simple observation advocates this. Although the Junction Tree Algorithm performs arithmetic calculations on the potential functions, the only information we really need from the values computed by the Junction Tree algorithm is whether they are zero or positive. Since we start from non-negative entries in our potential functions (the entries are either zeros or ones) and since the Junction Tree algorithm performs no subtractions, one could change the Junction Tree Algorithm arithmetic to account only for whether an entry it computes along the execution is zero or positive. Doing so we need only 1 bit for every stored entry and there are no arithmetic precision issues that we have to address. Thus, our algorithms are essentially combinatorial and our claim that the derived algorithms are polynomial time is valid.

4.1 Trees and Acyclic Hypergraphs

For a formal definition of *hypergraph acyclicity* we refer the reader to [4]. Note that since every tree has an acyclic hypergraph it is enough to state and prove our theorem for acyclic hypergraph games. But first we need a definition:

Definition 4.1 A ***join tree for a hypergraph*** $(\mathcal{P}, \mathcal{H})$ is a tree $T = (V, E)$, where $V \equiv \mathcal{H}$, so that, for all $v_1, v_2 \in V$ and for all $u \in V$ in the unique path between v_1 and v_2, $v_1 \cap v_2 \subseteq u$.

Next we give a high-level description of *Graham's algorithm* adapted from [4].

Algorithm 4.1 *(***Graham's algorithm***)*
On input $\mathcal{G} = (\mathcal{P}, \mathcal{H})$, where $\mathcal{H} = \{h_1, h_2, \dots, h_{|\mathcal{H}|}\}$, Graham's algorithm applies the following two operations to $\mathcal{G}$ repeatedly until neither can be applied:

(a) If a node $n \in \mathcal{P}$ appears in exactly one hyperedge $h_i \in \mathcal{H}$ then delete n from h_i

(b) Delete a hyperedge h_i if there is another hyperedge h_j, $j \neq i$, such that $h_i \subseteq h_j$.

We say that Graham's algorithm succeeds on $\mathcal{G} = (\mathcal{P}, \mathcal{H})$ if it terminates with an empty set of hyperedges.

The above concepts are related by the following important theorem from [4]:

Theorem 4.2 *If $\mathcal{R} = (\mathcal{P}, \mathcal{H})$ is a hypergraph then:*

$$(\mathcal{R} \text{ is acyclic }) \Leftrightarrow (\mathcal{R} \text{ has a join tree }) \Leftrightarrow (\text{Graham's algorithm succeeds on input } \mathcal{R})$$

We can now prove our claim:

Theorem 4.3 *Deciding whether a graphical game has a pure Nash equilibrium is in P when the hypergraph of the game is acyclic. Moreover, computing a succinct description of all pure Nash equilibria can be done in polynomial time.*

PROOF. On input $\mathcal{G} = \langle G = (V, E), S_p, u_p \rangle$, the algorithm proceeds in the following steps:

1. Apply Graham's algorithm on $\mathcal{H}(\mathcal{G})$ to check whether it is acyclic; if so, Graham's algorithm returns a join tree $T = (\mathcal{C}, \mathcal{E})$ for $\mathcal{H}(\mathcal{G})$ ([4]). It is easy to see that T is a clique tree for the primal graph G' of the game.

2. Reduce $\mathcal{G}$ to MRF($\mathcal{G}$,0) as described in Section 3. The graph of the MRF is the primal graph of the game and so T is a clique tree for the graph of MRF($\mathcal{G}$,0).

3. Run the junction tree algorithm on T to compute the marginal probability distributions of the significant cliques of MRF($\mathcal{G}$,0).

4. The marginal probability distributions answer the question of whether $\mathcal{G}$ has a pure Nash equilibrium and also constitute a succinct description of all pure Nash equilibria of $\mathcal{G}$ (see Section 3)

Correctness: The *correctness* of the algorithm follows from the correctness of the reduction and the correctness of Graham's algorithm and the junction tree algorithm. Note, also, that since the Junction Tree algorithm maintains *supportiveness* [23] it won't be affected by any "divisions by zero".
Time Complexity: Graham's algorithm finds the join tree of $\mathcal{H}(\mathcal{G})$ in time polynomial in the size of the hypergraph and thus in the number of players. Reducing $\mathcal{G}$ to MRF($\mathcal{G}$,0) also takes polynomial time. It remains to bound the running time of the message-passing phase of the junction tree algorithm. This phase is executed on the clique tree, which is precisely the join tree that Graham's algorithm returned, and it involves the exchange of as many messages as twice the number of edges of the clique tree, so at most $2n - 2$ messages, where n is the number of players (note that the number of nodes of the clique tree is at most equal to the number of players). Now, the time needed to compute a message that is being sent over an edge of the clique tree is polynomial in the size of the tables (potential functions) that are stored at the endpoints of that edge. However, the clique in every node of the clique tree corresponds to the neighborhood of a player and so its table has the same size as the table describing the utility function of that player. Thus, the complexity of every message is polynomial in the input complexity. It is, thus, obvious that the described algorithm runs in time polynomial in the description of the graphical game. $\square$

4.2 Bounded Treewidth and Hypertreewidth

It is not very hard to extend the results of Section 4.1 for tree games and acyclic hypergraph games to broader classes of graphical games, those of bounded treewidth and bounded hypertree width respectively.

For quick reference, we define here the notion of a *hypertree decomposition* and *hypertree-width* of a hypergraph. For more details on the properties of hypertree-width and its relation to other notions of hypergraph acyclicity we refer the reader to [15].

Definition 4.4 ([15]) Let $\mathcal{R} = (\mathcal{N}, \mathcal{H})$ be a hypergraph. A ***hypertree decomposition*** of $\mathcal{R}$ is a triplet $\langle T, \chi, \lambda \rangle$, where $T = (V, E)$ is a rooted tree and χ, λ are labelling functions associating each vertex $v \in V$ with two sets $\chi(v) \subseteq \mathcal{N}$ and $\lambda(v) \subseteq \mathcal{H}$, so that:

1. $\forall h \in \mathcal{H}, \exists v \in V : h \subseteq \chi(v)$

2. $\forall n \in \mathcal{N}$, the set $\{v \in V | n \in \chi(v)\}$ induces a connected subgraph of T

3. $\forall v \in V, \chi(v) \subseteq \bigcup_{h \in \lambda(v)} h$

4. $\forall v \in V, \chi(T_v) \cap \bigcup_{h \in \lambda(v)} h \subseteq \chi(v)$, where T_v is the subtree of T rooted at v and $\chi(T_v) = \bigcup_{v' \in vert(T_v)} \chi(v')$

The ***width*** of (T, χ, λ) is $max_{v \in V}\{|\lambda(v)|\}$.

The *hypertree width* of a hypergraph $\mathcal{R}$, $hw(\mathcal{R})$, is the minimum width over all its hypertree decompositions.

We can now state our claims. The proof of Theorem 4.6 uses Lemma 4.5. Theorem 4.7 was also proven independently in [14] using different techniques.

Lemma 4.5 *If the graph $G = (V, E)$ of a graphical game $\mathcal{G}$ has treewidth bounded by k then its primal graph has treewidth bounded by $(k+1) \cdot \max_{p \in V} |\mathcal{N}(p)| - 1$. Moreover, given a tree decomposition of G of width k we can compute in polynomial time a clique tree for the primal graph of $\mathcal{G}$ of width $(k+1) \cdot \max_{p \in V} |\mathcal{N}(p)|$.*

PROOF. (sketch) Let $T = (\mathcal{C}, \mathcal{E})$ be a tree decomposition of the game graph $G = (V, E)$, where $\mathcal{C} \subseteq 2^V$, and suppose that $k = \max_{c \in \mathcal{C}} |c| - 1$ is the width of the decomposition. We show how to construct a tree decomposition T' of the primal graph of $\mathcal{G}$ of width at most $(k+1) \cdot \max_{p \in V} |\mathcal{N}(p)| - 1$. T' is isomorphic to T and let σ denote the one-to-one correspondence between vertices of T and T'. Then for all $c \in \mathcal{C}$ we set $\sigma(c) = \bigcup_{p \in c} \mathcal{N}(p)$, i.e. every vertex of T' contains the union of the neighborhoods of all players of the corresponding vertex of T. It is not difficult to see that T' is a tree decomposition of the primal graph $G' = (V, E')$ of the game. Indeed, for every edge $(u, v) \in E'$ there is a vertex of T' that contains both u, v: if (u, v) is an edge in G' then players u, v must belong in the neighborhood of some player p (maybe $p \equiv u$ or $p \equiv v$); but there is at least one vertex of T that contains p and, thus, the corresponding vertex of T' must contain all its neighborhood and, so, u, v as well. Moreover, for every player $p \in V$ the vertices of T' that contain p form a connected subtree of T': since T is a tree decomposition of G, it is easy to see that the vertices of T that contain player p or a neighboring player of p in G form a connected component in T; but, in T', p appears in exactly those vertices whose corresponding vertex in T contains either p or a neighbor of p in G, and, so, the nodes in which p appears must form a connected component in T'. Finally, since $k+1 = \max_{c \in \mathcal{C}} |c|$, every vertex of T' contains at most $(k+1) \cdot \max_{p \in V} |\mathcal{N}(p)|$ vertices. Given T, the construction of T' can be done in polynomial time. □

Theorem 4.6 *Deciding whether a graphical game has a pure Nash equilibrium and computing (a succinct description of) all pure Nash equilibria is in P for all classes of games with bounded treewidth.*

PROOF. For a fixed constant k the algorithm performs the following steps on input $\mathcal{G} = \langle G = (V, E), \{S_p\}, \{u_p\} \rangle$:

1. Check whether G has treewidth bounded by k and, if so, find a tree decomposition of G of width at most k. For details on how to perform this step in polynomial time see for example [2, 20].

2. From the tree decomposition of G get —using Lemma 4.5— a clique tree T' of the primal graph of $\mathcal{G}$ that has width at most $(k+1) \cdot \max_{p \in V} |\mathcal{N}(p)|$.

3. Reduce $\mathcal{G}$ to MRF($\mathcal{G}$,0) as described in Section 3. The graph of the MRF is the primal graph of the game and so T' is a clique tree for the graph of MRF($\mathcal{G}$,0).

4. Run the Junction Tree Algorithm on T' to compute the marginal probability distributions of the significant cliques of MRF($\mathcal{G}$,0). The marginal probability distributions answer the question of whether $\mathcal{G}$ has a pure Nash equilibrium and also constitute a succinct description of all pure Nash equilibria of $\mathcal{G}$.

The *correctness* of the algorithm follows easily from the correctness of every intermediate step. The *running time* of the algorithm is polynomial. This is easily proven using the same rationale as in the proof of Theorem 4.3. However, in this case the cliques contained in the nodes of the clique tree have size that is at most $(k+1) \cdot \max_{p \in V} |\mathcal{N}(p)|$, i.e. at most $k+1$ times the size of the biggest neighborhood of the game. Since the biggest table of the input has dimension $\max_{p \in V} |\mathcal{N}(p)|$ and the biggest table in the clique tree has dimension $k+1$ times $\max_{p \in V} |\mathcal{N}(p)|$, where k is fixed, it follows that the tables of the clique tree have size polynomial in the input complexity. Thus, the algorithm runs in polynomial time. □

Theorem 4.7 *Deciding whether a graphical game has a pure Nash equilibrium and computing (a succinct description of) all pure Nash equilibria is in P for all classes of games with bounded hypertreewidth.*

PROOF. The proof is based on the following graph-theoretic lemma.

Lemma 4.8 *If a graphical game $\mathcal{G} = \langle G = (V, E), \{S_p\}, \{u_p\} \rangle$ has hypertree width bounded by k then its primal graph has treewidth bounded by $k \cdot \max_{p \in V} |\mathcal{N}(p)| - 1$. Moreover, given a hypertree decomposition (T, χ, λ) of $\mathcal{H}(\mathcal{G})$ we can compute in polynomial time a clique tree for the primal graph of $\mathcal{G}$ of width $k \cdot \max_{p \in V} |\mathcal{N}(p)|$, where k is the width of (T, χ, λ).*

PROOF. Given a hypertree decomposition (T, χ, λ) of $\mathcal{H}(\mathcal{G})$, take T' to be a tree isomorphic to T after removing directions from the edges and let σ denote the one-to-one correspondence between vertices of T and T'. Then for all $v \in vert(T)$ we set $\sigma(v) = \chi(v)$. It is easy to see that T' is a clique tree for the primal graph of $\mathcal{G}$, using properties 1 and 2 of the hypertree decomposition (see Definition 4.4). Moreover, from property 3 it follows that for every node v of T: $\chi(v) \subseteq \bigcup_{h \in \lambda(v)} h$. Thus, for every node v of T: $|\chi(v)| \le k \cdot \max_{p \in V} |\mathcal{N}(p)|$, where k is the width of the hypertree decomposition (T, χ, λ). Thus, the clique tree T' has width at most $k \cdot \max_{p \in V} |\mathcal{N}(p)|$. The construction can obviously be done in polynomial time. □

For a fixed constant k the algorithm performs the following steps on input $\mathcal{G} = \langle G = (V, E), \{S_p\}, \{u_p\} \rangle$:

1. Check whether $\mathcal{H}(\mathcal{G})$ has hypertree width bounded by k; if so find a hypertree decomposition (T, χ, λ) of $\mathcal{H}(\mathcal{G})$ of width at most k. For details on how to perform this step in polynomial time see [15].

2. From the hypertree decomposition (T, χ, λ), get —using Lemma 4.8— a clique tree T' of the primal graph of $\mathcal{G}$ that has width at most $k \cdot \max_{p \in V} |\mathcal{N}(p)|$.

3. run algorithm of Theorem 4.6 from step 3;

The *correctness* and the *time complexity* of the algorithm are analyzed in the same way as in the proof of Theorem 4.6.□

4.3 Games of $\mathbf{O}(\log \mathbf{n})$-Treewidth

In this section we go one step further in our study of graphical games and study games with $O(\log n)$ treewidth. For these games we derive polynomial time algorithms for pure Nash equilibria under the assumption that the degree of the graph is bounded (notice that this is necessary for the description of the problem to be polynomial in the number of players) and that the number of strategies available to each player is also bounded. Given that NP-completeness of computing pure Nash equilibria holds even when restricted to the class of graphical games with bipartite graphs of degree at most 3 and strategy sets of cardinality at most 3 [14], our result is tight. Moreover, it is the first positive result for computing pure Nash equilibria that is not based on some assumption about the cyclic structure of the graph.

Theorem 4.9 *Deciding whether a graphical game has a pure Nash equilibrium is in P for all classes of games with $O(\log n)$ treewidth, bounded number of strategies, and bounded neighborhood size. Moreover, computing a succinct description of all pure Nash equilibria can be done in polynomial time.*

PROOF. The algorithm is similar in spirit to the ones presented so far, but differs in the construction of the clique tree on which the Junction Tree algorithm performs. This task is somewhat involved and is described here. Suppose $k = k(n)$; by slightly modifying the algorithm presented by Becker and Geiger [3], we get an algorithm that runs in time $poly(n) \cdot 2^{4.67 \cdot k}$, on input a graph G of n nodes, and either outputs a tree decomposition of G of width at most $3.67k$ or outputs that the treewidth of G is larger than k[‡]. For $k = c \log n$, where c is a fixed constant, we get an algorithm that runs in time polynomial in n and either returns a tree decomposition of the input graph G of width at most $3.67 c \log n$ or outputs that the treewidth of G is larger than $c \log n$. Now, suppose

[‡]Alternatively we could use Reed's approximation algorithm [31] for treewidth or other approximation algorithms.

$\mathcal{G} = \langle G, \{S_p\}, \{u_p\}\rangle$ is a game drawn from a family of games with treewidth at most $c \log n$. Applied to G, the algorithm returns a tree decomposition of G of width at most $3.67 \cdot c \cdot \log n$. Given this tree decomposition, from Lemma 4.5, we can construct in polynomial time a clique tree for the primal graph of $\mathcal{G}$, which is the graph of the MRF, of width $w = (3.67 \cdot c \cdot \log n + 1) \cdot \max_{p \in V} |\mathcal{N}(p)|$. Now, if we assume bounded cardinality strategy sets, it follows that the sizes of the tables (potential functions) that will be stored in the clique tree before the execution of the junction tree algorithm and, thus, all the messages exchanged during the execution have size $O\left(n^{O(\max |\mathcal{N}(p)|)}\right)$. If, moreover, we assume bounded neighborhood size they are polynomial in the number of players. So all the computation takes polynomial time. $\square$

We can get rid of the bounded neighborhood requirement by pushing the $O(\log n)$-treewidth requirement to the primal graph of the game as stated by the following theorem. This way we can in some cases accommodate neighborhoods of size up to $O(\log n)$ which might be helpful in some applications.

Theorem 4.10 *Deciding whether a graphical game has a pure Nash equilibrium is in P for all classes of games with primal graphs of treewidth $O(\log n)$ and bounded cardinality strategy sets. Moreover, computing a succinct description of all pure Nash equilibria can be done in polynomial time.*

PROOF. The algorithm is similar in spirit to the one presented in the proof of Theorem 4.9. Again, we use the modified -as described in the proof of Theorem 4.9- algorithm of Becker and Geiger for $k = c \log n$, where c is a fixed constant, but we apply it directly on the graph G of the MRF. The algorithm runs in polynomial time and if G has treewidth bounded by $c \log n$, it returns a tree decomposition of G of width at most $3.67c \log n$. So the biggest table in the clique tree will be of dimension $3.67c \log n$. Thus, assuming bounded cardinality strategy sets, the sizes of the tables (potential functions) that we store at the nodes of the clique tree are polynomial in the number of players so all the computation takes polynomial time. $\square$

Let us also note that, by combining our results for finding pure Nash equilibria with a reduction, implicit in [18], of the problem of finding approximate mixed Nash equilibria (definition omitted, see e.g. [9]) to the problem of finding pure Nash equilibria, we obtain the following result:

Theorem 4.11 *An ϵ-approximate mixed Nash equilibrium of any graphical game with $O(\log n)$ treewidth, bounded neighborhood size and bounded number of strategies per player can be found in time polynomial in n and $1/\epsilon$.*

One final remark: Our pure Nash equilibrium algorithms try to reconcile two things. On the one hand, the running time of the junction tree algorithm crucially depends on the width of the clique tree compared to the size of the largest neighborhood of the game. On the other hand, computing the optimal clique tree is an NP-hard problem. To circumvent this intrinsic difficulty, notice that we do not really need to find the optimal clique tree, but only a constant factor approximation of the optimal. Indeed, the running time of the junction tree algorithm is polynomial in $n \cdot s^w$, where s is the number of strategies and w is the width of the clique tree the algorithm runs on. Therefore, the running time on a constant factor approximation of the optimal clique tree is polynomial in the running time on the optimal clique tree — and the algorithm for computing the constant factor approximation to the treewidth is sufficiently fast if it runs in time polynomial in $n \cdot 2^w$, where w is the achieved treewidth.

There are various approximation algorithms for treewidth that have this property. The one given in [3] runs in time proportional to $poly(n) \cdot 2^{4.67 \cdot k}$, where k is the treewidth, and returns a clique tree of width at most $3.67 \cdot k$; similarly for Reed's algorithm [31], Robertson and Seymour's algorithm [32] and other algorithms [5]. Interestingly, it turns out *all algorithms presented in this section can be derived as special cases of this scheme*. Details are deferred to the full paper.

4.4 Heuristics

Our definition of MRF($\mathcal{G}, \epsilon$) ensures that for some small $\epsilon(\mathcal{G}) < 1$ most of the probability mass is concentrated on the set of pure Nash equilibria of $\mathcal{G}$, if such equilibria exist. More generally, the probability of a configuration decays exponentially in the number of unsatisfied players. This observation suggests quite naturally the use of sampling techniques to find pure Nash equilibria of games or approximations to pure Nash equilibria, i.e. configurations with as many satisfied players as possible. Markov Chain Monte Carlo methods and Simulated Annealing (see e.g. [12]) can be easily applied for computing pure Nash equilibria under our reduction.

Survey propagation is an algorithm originating in statistical inference that appears to be very effective in solving random k-SAT instances and other constraint satisfaction problems (see, e.g. [6]). Survey propagation performs better than belief propagation for k-SAT instances near the SAT/UNSAT threshold (see e.g. [1] for the later). In [25] survey propagation is extended to a family of survey propagation algorithms (parameterized by a real number $\rho \in [0, 1]$) that has at one extreme ($\rho = 0$) belief propagation and on the other extreme ($\rho = 1$) survey propagation. We believe that survey propagation algorithms for pure Nash equilibria will be very useful for solving hard instances of graphical games. The effectiveness of all these schemes must be determined experimentally. Some experimental results are presented in the next section.

5. EXPERIMENTS

The following algorithms for solving graphical games were implemented: the Junction Tree Algorithm of Section 4, a Loopy Belief Propagation Algorithm [30] and a Markov Chain Monte Carlo, all based on our reduction. The Markov Chain Monte Carlo is a Metropolis-Hastings algorithm with the naive proposal distribution. The test cases were random graphical games constructed as follows: the graphs of the games were drawn from the family $\mathcal{G}(n, p)$ of random graphs, for different values of n -number of players- and p -densities of the games; the payoff tables were then populated with random values from $[0, 1]$.

The Junction Tree Algorithm is exact; it was, therefore, only meaningful to measure the dependence of its running time on the density of the game-graph. Figure 1 shows the running time of the junction tree algorithm on 15-player games for different densities, where for each density we considered 100 test cases. The exponential dependence of the running time on the density of the game is apparent. This was expected since the running time is exponential in the treewidth.

On the other hand, the Loopy Belief Propagation and the MCMC algorithms are not exact, so their output on the test cases was compared to that of the exact algorithm. In all our experiments and regardless of the number of players, the Belief Propagation Algorithm and the MCMC were let run for only a few iterations. Note,

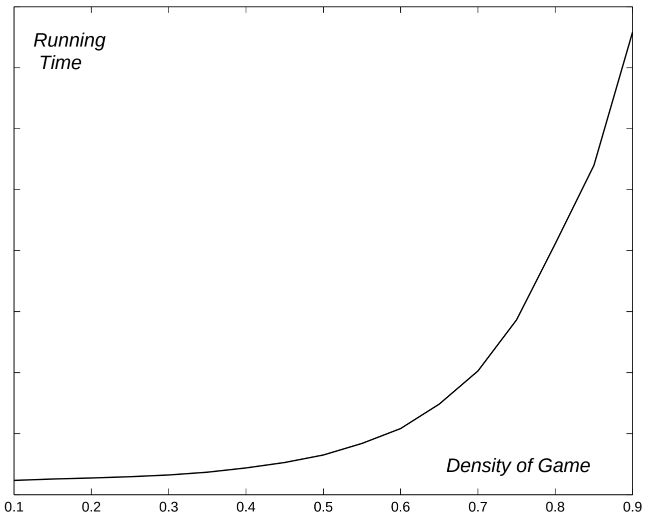

Figure 1: The running time of the Junction Tree algorithm for different densities of 15 player games.

however, that the search space increases exponentially with the number of players; this explains the divergence of the two curves of Figure 2, in which we present the performance of the MCMC algorithm for values of n ranging from 5 to 20 and for graphical games of density 0.5 —for every value of n we considered 100 test cases and we present the average of the results. The output of the MCMC algorithm is comparable to that of the exact algorithm, although the MCMC run in time linear in the number of players whereas the exact algorithm in time exponential. Note also that for the value of the density that we considered, the graphs of the games are dense and we are departing from the regime where pure Nash equilibria can be computed efficiently (see e.g. Figure 1).

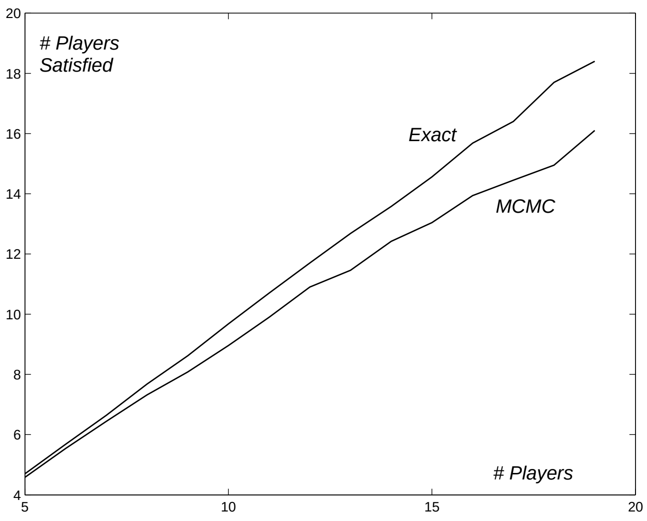

Figure 2: The performance of the MCMC algorithm for different game sizes and density 0.5. The performance is measured in terms of the number of players that are satisfied, i.e. play a strategy that is a best response to the strategies of their neighbors.

In Figure 3 we present how the performance of the MCMC algorithm scales with the density of the game. We observe that, for low game densities, which result in games with small treewidth, the MCMC algorithm is very efficient; as density increases the performance diverges from the optimal value.

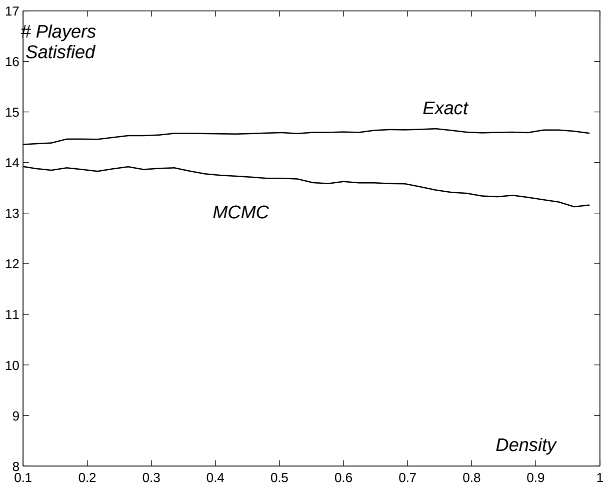

Figure 3: The performance of the MCMC algorithm for different densities of 15 player games.

Finally, in Figure 4 we present the performance of Loopy Belief Propagation on a test set of 15-player games with densities varying from 0.1 to 0.9. Belief Propagation was set to compute the maximum-a-posteriori configuration and was let run for only a few iterations. The result was examined to find how many players of the game are satisfied, i.e. how many players play a strategy that is a best response to the strategies of their neighbors, and was compared to the optimal value as in Figures 2, 3. We observe that as the density increases and the primal graph of the game becomes rich in cycles the performance of loopy belief propagation deteriorates significantly. Even for small density values, the performance is seriously affected by the small cycles introduced when making a clique out of every neighborhood of the game (recall that the graph of the MRF is the primal graph of the game). We believe that generalized belief propagation [34] —if set appropriately— will not be affected by these small cycles introduced by the reduction and will perform significantly better.

Acknowledgements

We would like to thank Elchanan Mossel for interesting discussions.

6. REFERENCES

[1] D. Achlioptas and Y. Peres. The Threshold for Random k-Sat is 2^k (ln 2 - o(k)). In *STOC*, 2003.

[2] S. Arnborg, D. G. Corneil and A. Proskurowski. Complexity of Finding Embeddings in a k-Tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.

[3] A. Becker and D. Geiger. A Sufficiently Fast Algorithm for Finding Close to Optimal Clique Trees. *Artif. Intell.*, 125(1-2):3–17, 2001.

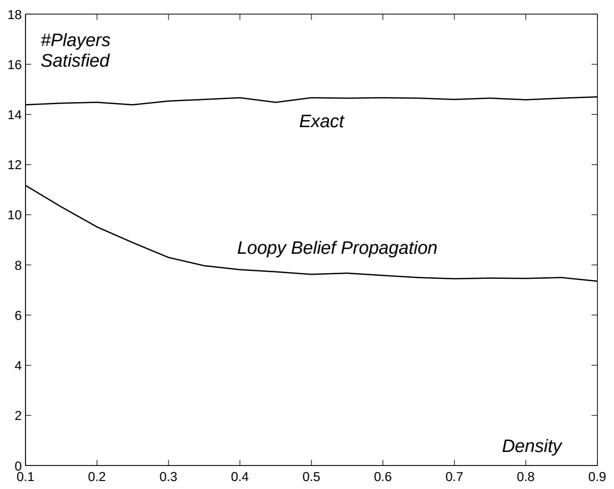

Figure 4: The performance of loopy belief propagation for different densities of 15 player games. When the primal graph of the game becomes rich in cycles the performance deteriorates significantly.

[4] C. Beeri, R. Fagin, D. Maier and M. Yannakakis. On the Desirability of Acyclic Database Schemes. *J. ACM*, 30(3):479–513, 1983.

[5] H. L. Bodlaender. Discovering Treewidth. In *SOFSEM*, 2005.

[6] A. Braunstein, M. Mezard and R. Zecchina. Survey Propagation: an Algorithm for Satisfiability. *CoRR*, cs.CC/0212002, 2002.

[7] X. Chen and X. Deng. 3-NASH is PPAD-Complete. *Electronic Colloquium in Computational Complexity TR05-134*, 2005.

[8] X. Chen and X. Deng. Settling the Complexity of 2-Player Nash-Equilibrium. *Electronic Colloquium in Computational Complexity TR05-140*, 2005.

[9] C. Daskalakis, P. W. Goldberg and C. H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. In *STOC*, 2006.

[10] C. Daskalakis and C. H. Papadimitriou. Three-Player Games Are Hard. *Electronic Colloquium in Computational Complexity TR05-139*, 2005.

[11] E. Elkind, L. A. Goldberg and P. W. Goldberg. Nash Equilibria in Graphical Games on Trees Revisited. *ACM Conference on Electronic Commerce*, 2006.

[12] W. Gilks, S. Richardson and D. Spiegelhalter. *Markov Chain Monte Carlo in practice.* Chapman and Hall, London, 1996.

[13] P. W. Goldberg and C. H. Papadimitriou. Reducibility Among Equilibrium Problems. In *STOC*, 2006.

[14] G. Gottlob, G. Greco and F. Scarcello. Pure Nash Equilibria: Hard and Easy Games. In *TARK*, 2003.

[15] G. Gottlob, N. Leone and F. Scarcello. Hypertree Decompositions and Tractable Queries. *J. Comput. Syst. Sci.*, 64(3):579–627, 2002.

[16] F. Jensen, S. Lauritzen and K. Olesen. Bayesian Updating in Causal Probabilistic Networks by Local Computations. *Computational Statistics Quarterly*, 4:269–282, 1990.

[17] S. Kakade, M. J. Kearns, J. Langford and L. E. Ortiz. Correlated Equilibria in Graphical Games. In *ACM Conference on Electronic Commerce*, 2003.

[18] M. J. Kearns, M. L. Littman and S. P. Singh. Graphical Models for Game Theory. In *UAI*, 2001.

[19] S. Kirkpatrick, J. C. D. Gelatt and M. P. Vecchi. Optimization by Simulated Annealing. *Readings in computer vision: issues, problems, principles, and paradigms*, pages 606–615, 1987.

[20] T. Kloks. *Treewidth: Computations and Approximations*. Springer-Verlag, 1994.

[21] D. M. Kreps. *Game Theory and Economic Modelling.* Oxford University Press, 1990.

[22] S. L. Lauritzen. *Graphical Models.* Clarendon Press, Oxford, 1996.

[23] S. L. Lauritzen and D. J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. *Readings in uncertain reasoning*, pages 415–448, 1990.

[24] M. L. Littman, M. J. Kearns and S. P. Singh. An Efficient, Exact Algorithm for Solving Tree-Structured Graphical Games. In *NIPS*, 2001.

[25] E. Maneva, E. Mossel and M. J. Wainwright. A New Look at Survey Propagation and its Generalizations. *SODA*, 2005.

[26] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21, 1953.

[27] L. E. Ortiz and M. J. Kearns. Nash Propagation for Loopy Graphical Games. In *NIPS*, 2002.

[28] M. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

[29] C. H. Papadimitriou. Computing Correlated Equilibria in Multi-Player Games. In *STOC*, 2005.

[30] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, 1988.

[31] B. A. Reed. Finding Approximate Separators and Computing Tree Width Quickly. In *STOC*, 1992.

[32] N. Robertson and P. D. Seymour. Graph Minors. xiii: the Disjoint Paths Problem. *J. Comb. Theory Ser. B*, 63(1):65–110, 1995.

[33] M. J. Wainwright and M. I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Technical Report*, 64, Department of Statistics, University of California, Berkeley, 2003.

[34] J. S. Yedidia, W. T. Freeman and Y. Weiss. Generalized Belief Propagation. In *NIPS*, 2000.