Multi-Agent Algorithms for Solving Graphical Games

David Vickrey
Computer Science Department
Stanford University
Stanford, CA 94305-9010
dvickrey@cs.stanford.edu

Daphne Koller
Computer Science Department
Stanford University
Stanford, CA 94305-9010
koller@cs.stanford.edu

Abstract

Consider the problem of a group of agents trying to find a stable strategy profile for a joint interaction. A standard approach is to describe the situation as a single multi-player game and find an equilibrium strategy profile of that game. However, most algorithms for finding equilibria are computationally expensive; they are also centralized, requiring that all relevant payoff information be available to a single agent (or computer) who must determine the entire equilibrium profile. In this paper, we exploit two ideas to address these problems. We consider structured game representations, where the interaction between the agents is sparse, an assumption that holds in many real-world situations. We also consider the slightly relaxed task of finding an approximate equilibrium. We present two algorithms for finding approximate equilibria in these games, one based on a hill-climbing approach and one on constraint satisfaction. We show that these algorithms exploit the game structure to achieve faster computation. They are also inherently local, requiring only limited communication between directly interacting agents. They can thus be scaled to games involving large numbers of agents, provided the interaction between the agents is not too dense.

1 Introduction

Consider a system consisting of multiple interacting agents, collaborating to perform a task. The agents have to interact with each other to make sure that the task is completed, but each might still have slightly different preferences, e.g., relating to the amount of resources each expends in completing its part of the task.

The framework of game theory (von Neumann & Morgenstern 1944; Fudenberg & Tirole 1991) tells us that we should represent a multi-agent interaction as a game, and find a strategy profile that forms a *Nash equilibrium* (Nash 1950). We can do so using one of several algorithms for finding equilibria in games. (See (McKelvey & McLennan 1996) for a survey.) Unfortunately, this approach is severely limited in its ability to handle complex multi-agent interactions. First, in most cases, the size of the standard game representations grows exponentially in n. Second, for games involving more than two players, existing solution algorithms scale extremely poorly even in the size of the game representation. Finally, all of the standard algorithms are based on a centralized computation paradigm making them unsuitable for our distributed setting.

We propose an approach that modifies both the representation of the game and the notion of a solution. Following the work of LaMura (2000), Koller and Milch (2001), and Kearns, Littman, and Singh (2001a), we use a structured representations of games, that exploits the locality of interaction that almost always exists in complex multi-agent interactions, and allows games with large numbers of agents to be described compactly. Our representation is based on the *graphical game* framework of Kearns, Littman, and Singh (KLS hereafter), which applies to simultaneous-move games. We wish to find algorithms that can take advantage of this structure to find good strategy profiles effectively, and in a decentralized way.

It turns out that this goal is much easier to achieve when solving a relaxed problem. While philosophically satisfying, the Nash equilibrium requirement is often overly stringent. Although agents arguably strive to maximize their expected utility, in practice inertia or a sense of commitment will cause an agent to abide by an agreed equilibrium even if it is slightly suboptimal for him. Thus, it often suffices to require that the strategy profile form an *approximate equilibrium*, one where each agent's incentive to deviate is no more than some small ϵ.

We present two techniques for finding approximate equilibria in structured games. The first uses a greedy hill-climbing approach to optimize a global score function, whose global optima are precisely equilibria. The second uses a constraint satisfaction approach over a discretized space of agent strategies; somewhat surprisingly, the algorithm of KLS turns out to be a special case of this algorithm. We show that these algorithms allow the agents to determine a joint strategy profile using local communication between agents. We present some preliminary experimental results over randomly generated single-stage games, where we vary the number of agents and the density of the interaction. Our results show that our algorithms can find high-quality approximate equilibria in much larger games than have been previously solved.

2 Graphical games

In this section, we introduce some basic notation and terminology for game theory, and describe the framework of graphical games.

The conceptually simplest and perhaps best-studied representation of game is the *normal form*. In a normal form game, each player (agent) p_i chooses an action a_i from its action set $\{a_{i1}, a_{i2}, ..., a_{ik_i}\}$. For simplicity of notation, we assume that $k_1 = k_2 = ... = k_n = k$. The players are also allowed to play *mixed strategies* $\theta_i = \langle\theta_{i1}, \theta_{i2}, ...\theta_{ik}\rangle$ where θ_{ij} is the probability that p_i plays a_{ij}. If the player assigns probability 1 to one action — $\theta_{ij} = 1$ — and zero to the others, it is said to be playing a *pure strategy*, which we denote as r_{ij}. We use $\boldsymbol{\theta}$ to denote a strategy profile for the set of players, and define $(\boldsymbol{\theta}_{-i}, \theta'_i)$ to be the same as $\boldsymbol{\theta}$ except that p_i plays θ'_i instead of θ_i.

Each player also has an associated payoff matrix M_i that specifies the payoff, or utility, for player i under each of the k^n possible combinations of strategies: $M_i(a_1, a_2, ..., a_n)$ is the reward for p_i when for all j, p_j plays a_j. Given a profile $\boldsymbol{\theta}$, we define the *expected utility* (or payoff) for p_i as

$$U_i(\boldsymbol{\theta}) = \sum_{i_1, i_2, \ldots, i_n} \theta_{1i_1} \ldots \theta_{ni_n} M_i(a_{1i_1}, a_{2i_2}, \ldots, a_{ni_n}).$$

Given a set of mixed strategies $\boldsymbol{\theta}$, one strategy per player, we define the *regret of* p_i *with respect to* $\boldsymbol{\theta}$ to be the most p_i can gain (on expectation) by diverging from the strategy profile $\boldsymbol{\theta}$:

$$Reg_i(\boldsymbol{\theta}) = \max_{\theta'_i}(U_i((\theta_{-i}, \theta'_i)) - U_i(\boldsymbol{\theta})).$$

A *Nash equilibrium* is a set of mixed strategies $\boldsymbol{\theta}$ where each player's regret is 0. The Nash equilibrium condition means that no player can increase his expected reward by unilaterally changing his strategy. The seminal result of game theory is that any game has at least one Nash equilibrium (Nash 1950) *in mixed strategies*. An *ϵ-approximate Nash equilibrium* is a strategy profile $\boldsymbol{\theta}$ such that each player's regret is at most ϵ.

A *graphical game* (Kearns, Littman, & Singh 2001a) assumes that each player's reward function depends on the actions of a subset of the players rather than on all other players' actions. Specifically, p_i's utility depends on the actions of some subset $\mathcal{P}_i$ of the other players, as well as on its own action. Thus, each player's payoff matrix M_i depends only on $|\mathcal{P}_i| + 1$ different decision variables, and therefore has $k^{|\mathcal{P}_i|+1}$ entries instead of k^n. We can describe this type of game using a directed graph (V, E). The nodes in V correspond to the n players, and we have a directed edge $e = (p_i, p_j) \in E$ from p_i to p_j if $p_i \in \mathcal{P}_j$, i.e., if j's utility depends on p_i's strategy. Thus, the parents of p_i in the graph are the players on whose action p_i's value depends. We note that our definition is a slight extension of the definition of KLS, as they assumed that the dependency relationship between players was symmetric, so that their graph was undirected.

Example 1: Consider the following example, based on a similar example in (Koller & Milch 2001). Suppose a road is being built from north to south through undeveloped land, and $2n$ agents have purchased plots of land along the road — the agents $w_1, \ldots, w_n$ on the west side and the agents $e_1, \ldots, e_n$ on the east side. Each agent needs to choose what to build on his land — a factory, a shopping mall, or a residential complex. His utility depends on what he builds and on what is built north, south, and across the road from his land. All of the decisions are made simultaneously. In this case, agent w_i's parents are e_i, w_{i-1} and w_{i+1}. Note that the normal form representation consists of $2n$ matrices each of size 3^{2n}, whereas in the graphical game, each matrix has size at most $3^4 = 81$ (agents at the beginning and end of the road have smaller matrices).

If we modify the problem slightly and assume that the prevailing wind is from east to west, so that agents on the east side are not concerned with what is built across the street, then we have an asymmetric graphical game, where agent w_i's parents are e_i, w_{i-1}, w_{i+1}, whereas agent e_i's parents are e_{i-1}, e_{i+1}. ∎

3 Function Minimization

Our first algorithm uses a hill-climbing approach to find an approximate equilibrium. We define a score function that measures the distance of a given strategy profile away from an equilibrium. We then use a greedy local search algorithm that starts from a random initial strategy profile and gradually improves the profile until a local maximum of the score function is reached.

More precisely, for a strategy profile $\boldsymbol{\theta}$, we define $S(\boldsymbol{\theta})$ to be the sum of the regrets of the players:

$$S(\boldsymbol{\theta}) = \sum_i Reg_i(\boldsymbol{\theta}).$$

This function is nonnegative and is equal to 0 exactly when $\boldsymbol{\theta}$ is a Nash equilibrium. It is continuous in each of the separate probabilities θ_{ij} but nondifferentiable.

We can minimize $S(\boldsymbol{\theta})$ using a variety of function minimization techniques that apply to continuous but non-differentiable functions. In the context of unstructured games, this approach has been explored by (McKelvey 1992). More recently, LaMura and Pearson (2001) have applied simulated annealing to this task. We chose to explore greedy hill climbing, as it lends itself particularly well to exploiting the special structure of the graphical game.

Our algorithm repeatedly chooses a player and changes that player's strategy so as to maximally improve the global score. More precisely, we define the *gain* for a player p_i as the amount that global score function would decrease if p_i changed its strategy so as to minimize the score function:

$$G_i(\boldsymbol{\theta}) = \max_{\theta'_i}[S(\boldsymbol{\theta}) - S((\boldsymbol{\theta}_{-i}, \theta'_i))].$$

Note that this is very different from having the player change its strategy to the one that most improves its own utility. Here, the player takes into consideration the effects of its strategy change on the other players.

Our algorithm first chooses an initial random strategy $\boldsymbol{\theta}$ and calculates $G_i(\boldsymbol{\theta})$ for each i. It then iterates over the following steps:

1. Choose the player p_i for which $G_i(\boldsymbol{\theta})$ is largest.
2. If $G_i(\boldsymbol{\theta})$ is positive, update $\theta_i := \text{argmax}_{\theta'_i}[S(\boldsymbol{\theta}) - S((\boldsymbol{\theta}_{-i}, \theta'_i))]$; otherwise, stop.

3. For each player p_j such that $G_j(\boldsymbol{\theta})$ may have changed, recalculate $G_j(\boldsymbol{\theta})$.

Notice that $Reg_i(\boldsymbol{\theta})$ depends only on the strategies of p_i and its parents in $\boldsymbol{\theta}$. Thus changing a player's strategy only affects the terms of the score function corresponding to that player and its children. We can use this to implement steps (2) and (3) efficiently. A somewhat laborious yet straightforward algebraic analysis shows that:

Proposition 2: *The following optimization problem is equivalent to finding $G_i(\boldsymbol{\theta})$ and the maximizing θ'_i:*

$$\begin{array}{ll} \textit{Maximize:} & U_i((\boldsymbol{\theta}_{-i}, \theta'_i)) - \sum_{j:i\in\mathcal{P}_j} (y_j - U_j((\boldsymbol{\theta}_{-i}, \theta'_i))) \\ \textit{Subject to:} & \theta'_{im} \geq 0 \quad \forall m \\ & \sum_m \theta'_{im} = 1 \\ & y_j \geq U_j(((\boldsymbol{\theta}_{-i}, \theta'_i)_{-j}, r_{jl})) \quad \forall j, l \end{array}$$

As the expected utility functions U_j are linear in the θ'_{im}, this optimization problem is simply a linear program whose parameters are the strategy probabilities of player p_i, and whose coefficients involve the utilities only of p_i and its children. Thus, the player p_i can optimize its strategy efficiently, based only on its own utility function and that of its children in the graph. We can therefore execute the optimization in step (2) efficiently. In our asymmetric Road example, an agent w_i could optimize its strategy based only on its children — w_{i-1} and w_{i+1}; similarly, an agent e_i needs to consider its children — e_{i-1}, e_{i+1} and w_i.

To execute step (3), we note that when p_i changes its strategy, the regrets of p_i and its children change; and when the regret of p_j changes, the gains of p_j and its parents change. More formally, when we change the strategy of p_i, the linear program for some other player p_j changes only if one of the expected utility terms changes. Since we only have such terms over p_j and its children, and the payoff of a player is affected only if the strategy at one of its parents changes, then $G_j(\boldsymbol{\theta})$ will change only if the strategy of p_j, or one of its parents, its children, or its spouses (other parents of its children) is changed. (Note the intriguing similarity to the definition of a Markov blanket in Bayesian networks (Pearl 1988).) Thus, in step (3), we only need to update the gain of a limited number of players. In our Road example, if we change the strategy for w_i, we need to update the gain of: w_{i-1} and w_{i+1} (both parents and children); e_i (only a parent); and w_{i-2}, w_{i+2}, e_{i-1}, and e_{i+1} (spouses).

We note that our hill climbing algorithm is not guaranteed to find a global minimum of $S(\boldsymbol{\theta})$. However, we can use a variety of techniques such as random restarts in order to have a better chance of finding a good local minimum. Also, local minima that we find are often fairly good approximate equilibria (since the score function corresponds quite closely to the quality of an approximate equilibrium).

4 CSP algorithms

Our second approach to solving graphical games uses a very different approach, motivated by the recent work of Kearns, Littman, and Singh (2001a; 2001b). They propose a dynamic programming style algorithm for the special case when the graphical game is a symmetric undirected tree.

Their algorithm has several variants. For our purposes, the most relevant (KLS 2001a) discretizes each player's set of mixed strategies, so that the tables represent a discrete grid of the players' strategy profiles. Since this variant does not explore the entire strategy space, it is limited to finding approximate equilibria. (Two other variants (KLS 2001a; 2001b) compute exact equilibria, but only apply in the very limited case of two actions per player.)

It turns out that the KLS algorithm can be viewed as applying *nonserial dynamic programming* or *variable elimination* (Bertele & Brioschi 1972) to a constraint satisfaction problem (CSP) generated by the graphical game. In this section, we present a CSP formulation of the problem of finding Nash equilibria in a general graphical game, and show how variable elimination can be applied to solve it. Unlike the KLS algorithm, our algorithm also applies to asymmetric and non-tree-structured games. We can also solve the problem as a constrained optimization rather than a constraint satisfaction problem, potentially improving the computational performance of the KLS approach.

Constraint Satisfaction There are many ways of formulating the ϵ-equilibrium problem as a CSP. Most simply, each variable V_i corresponds to the player p_i and takes values in the strategy space of p_i. The constraints C_i ensure that each player has regret at most ϵ in response to the strategies of its parents. (Recall that the each player's regret depends only its strategy and those of its parents.) Specifically, the "legal" set for C_i is

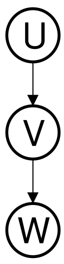
$$\{\langle \theta_i, (\theta_j)_{j\in\mathcal{P}_i} \rangle \mid Reg_i(\theta_i, \theta_{\mathcal{P}_i}) \leq \epsilon\}.$$

This constraint is over all of the variables in $\mathcal{P}_i \cup \{i\}$.

The variables in this CSP have continuous domains, which means that standard techniques for solving CSPs do not directly apply. We adopt the gridding technique proposed by KLS, which defines a discrete value space for each variable. Thus, the size of these constraints is exponential in the maximum family size (number of neighbors of a node), with the base of the exponent growing with the discretization density.

Variable elimination is a general-purpose nonserial dynamic programming algorithm that has been applied to several frameworks, including CSPs. Roughly speaking, we eliminate variables one at a time, combining the constraints relating to that variable into a single constraint, that describes the constraints induced over its neighboring variables. We briefly review the algorithm in the context of the constraints described above.

Example 3 : Consider the three-player graphical game shown in Fig. 1(a), where we have discretized the strategy space of V into three strategies and those of U and W into two strategies. Suppose we have chosen an ϵ such that the constraints for V and W are given by Fig. 1(b),(c) (the constraint for U is not shown). The constraint for V, for example, is indexed by the strategies of U and V; a 'Y' in the table denotes that V's strategy has at most ϵ regret with respect to U's strategy. Eliminating V produces a constraint over U and W as shown in Fig. 1(d). Consider the (u_2, w_1) entry of the resulting constraint. We check each possible strategy for V. If V were playing v_1, then V would not have acceptable regret with respect to u_2, and W's strategy, w_1, would not have acceptable regret with respect to v_1. If V



(a)

	u_1	u_2
v_1		
v_2	Y	Y
v_3		Y

(b)

	v_1	v_2	v_3
w_1		Y	
w_2			Y

(c)

	u_1	u_2
w_1	Y	Y
w_2		Y

(d)

	u_1	u_2
v_1	.6	.4
v_2	.1	.2
v_3	.5	0

(e)

	v_1	v_2	v_3
w_1	.4	.1	.5
w_2	.7	.4	.2

(f)

	u_1	u_2
w_1	.1	.2
w_2	.4	.2

(g)

Figure 1: (a) A simple 3-player graphical game. (b) Constraint table for V. (c) Constraint table for W. (d) Constraint table after elimination of V. (e) Regret table for V. (f) Regret table for W. (g) Regret table after elimination of V.

were playing v_3, V's strategy would be acceptable with respect to U's, but W's would not be acceptable with respect to V's. However, if V were playing v_2, then both V and W would be playing acceptable strategies. As there is a value of V which will produce an acceptable completion, the entry in the corresponding table is 'Y'. The (u_1, w_1) entry is not 'Y' since there is no strategy of V which will ensure that both V and W are playing acceptably. ∎

In general, we can eliminate variables one by one, until we are left with a constraint over a single variable. If the domain of this variable is empty, the CSP is unsatisfiable. Otherwise, we can pick one of its legal values, and execute this process in reverse to gradually extend each partial assignment to a partial assignment involving one additional variable. Note that we can also use this algorithm to find all solutions to the CSP: at every place where we have several legal assignments to a variable, we pursue all of them rather than picking one.

For undirected trees, using an "outside-in" elimination order, variable elimination ends up being very similar to the KLS algorithm. We omit details for lack of space. However, the variable elimination algorithm also applies as is to graphical games that are not trees, and to asymmetric games. Furthermore, the realization that our algorithms are simply solving a CSP opens the door to the application of alternative CSP algorithms, some of which might perform better in certain types of games.

Note that the value of ϵ is used in the CSP algorithm to define the constraints; if we run the algorithm with too coarse a grid, it might return an answer that says that no such equilibrium exists. Thus to be sure of obtaining an ϵ-optimal equilibrium, we must choose the grid according to the bound provided by KLS. Fortunately, the proof given is not specific to undirected trees, and thus we are provided with a gridding density (which is exponential only in the maximum family size) which will guarantee we find a solution. Unfortunately, the bound is usually very pessimistic and leads to unreasonably fine grids. For example, in a 2-action Road game (which is discussed in Section 6), to guarantee a 0.2-approximate equilibrium, the KLS discretization would need to be approximately $.0008$, which means we would need about 1250 grid points per strategy.

Cost Minimization

An alternative to viewing the regret bounds as hard constraints is to try to directly reduce the worst-case regret over the players. This approach, which is a variant of a *cost-minimization problem (CMP)*, allows us to choose an arbitrary grid density and find the best equilibrium for that density. In our CMP algorithm, we replace the constraints with tables which have the same structure but instead of containing 'Y' or being blank, they simply contain the regret of the player under that set of strategies. More precisely, we have one initial factor for each player p_i, which contains one entry for each possible strategy profile $(\theta_i, \theta_{\mathcal{P}_i})$ for p_i and its parents $\mathcal{P}_i$. The value of this entry is simply $Reg_i(\theta_i, \theta_{\mathcal{P}_i})$. (As we discussed, regret only depends on the strategies of the player and his parents.)

Example 4: Consider again the three-player graphical game of Fig. 1(a). The regret tables for V, W are shown in Fig. 1(e),(f). Eliminating V produces a table over U and W, shown in Fig. 1(g). Consider the (u_2, w_1) entry of the resulting table. We check each possible strategy for V. If V plays v_1, then V would have regret .4 with respect to u_2, and W's strategy, w_1, would have regret .4 with respect to v_1; thus, we can only obtain a minimal regret of .4 when V plays v_1. If V plays v_3, V would have regret 0, but W would have regret .5, so the minimum regret over all players is .5. Finally, if V plays v_2, then V would have regret .2 and W would have regret .1, for a minimum regret of .2. Thus, the minimum value over all strategies of V of the lowest achievable regret is .2. ∎

More generally, our elimination step in the CMP algorithm is similar to the CSP algorithm, except that now the entry in the table is the minimum achievable value (over strategies of the eliminated player) of the maximum over all tables involving the eliminated player. More precisely, let $\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_k$ be a set of factors each containing p_i, and let $\boldsymbol{N}_j$ be the set of nodes contained in $\boldsymbol{F}_j$. When we eliminate p_i, we generate a new factor $\boldsymbol{F}$ over the variables $\boldsymbol{N} = \cup_{j=1}^{k} \boldsymbol{N}_j - \{p_i\}$ as follows: For a given set of policies $\theta_{\boldsymbol{N}}$, the corresponding entry in $\boldsymbol{F}$ is $\min_{\theta_i} \max_j \boldsymbol{F}_j[(\theta_{\boldsymbol{F}}, \theta_i)_{\boldsymbol{N}_j}]$. Each entry in a factor $\boldsymbol{F}_j$ corresponds to some strategy profile for the players in $\boldsymbol{N}_j$. Intuitively, it represents an upper bound on the regret of some of these players, assuming this strategy profile is played. To eliminate p_i, we consider all of his strategies, and choose the one that guarantees us the lowest regret.

After eliminating all of the players, the result is the best achievable worst-case regret — the one that achieves the minimal regret for the player whose regret is largest. The associated completion is precisely the approximate equilibrium that achieves the best possible ϵ. We note that the CSP algorithm essentially corresponds to first rounding the entries in the CMP tables to either 0 or 1, using ϵ as the rounding cutoff, and then running CMP; an assignment is a solution to the CSP iff it has value 0 in the CMP.

Finally, note that all of the variable elimination algorithms naturally use local message passing between players in the game. In the tree-structured games, the communication directly follows the structure of the graphical game. In more complex games, the variable elimination process might lead to interactions between players that are not a priori directly related to each other. In general, the communication will be along edges in the *triangulated graph* of the graphical game (Lauritzen & Spiegelhalter 1988). However, the communication tends to stay localized to "regions" in the graph, except for graphs with many direct interactions between "remote" players.

5 Hybrid algorithms

We now present two algorithms that combine ideas from the two techniques presented above, and which have some of the advantages of both.

Approximate equilibrium refinement

One problem with the CSP algorithm is the rapid growth of the tables as the grid resolution increases. One solution is to find an approximate equilibrium using some method, construct a fine grid around the region of the approximate equilibrium strategy profile, and use the CMP or CSP algorithms to find a better equilibrium over that grid. If we find a better equilibrium in this finer grid, we recenter our grid around this point, shifting our search to a slightly different part of the space. If we do not find a better equilibrium with the specified grid granularity, we restrict our search to a smaller part of the space but use a finer grid. This process is repeated until some threshold is reached.

Note that this strategy does not guarantee that we will eventually get to an exact equilibrium. In some cases, our first equilibrium might be at a region where there is a local minimum of the cost function, but no equilibrium. In this case, the more refined search may improve the quality of the approximate equilibrium, but will not lead to finding an exact equilibrium.

Subgame decomposition

A second approach is based on the idea that we can decompose a single large game into several subgames, solve each separately, and then combine the results to get an equilibrium for the entire game. We can implement this general scheme using an approach that is motivated by the clique tree algorithm for Bayesian network inference (Lauritzen & Spiegelhalter 1988).

To understand the intuition, consider a game that is composed of two almost independent subgames. Specifically, we can divide the players into two groups $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ whose only overlap is the single player p_i. We assume that the games are independent given p_i, in other words, for any $j \neq i$, if $p_j \in \boldsymbol{C}_k$, then $\mathcal{P}_j \subseteq \boldsymbol{C}_k$. If we fix a strategy θ_i of p_i, then the two halves of the game no longer interact. Specifically, we can find an equilibrium for the players in $\boldsymbol{C}_k$, ensuring that the players' strategies are a best response both to each other's strategies and to θ_i, without considering the strategies of players in the other cluster. However, we must make sure that these strategy profiles will combine to form an equilibrium for the entire game. In particular, all of the players' strategies must be a best response to the strategy profiles of their parents. Our decomposition guarantees this property for all the players besides p_i. To satisfy the best-response requirement for p_i we must address two issues. First, it may be the case that for a particular strategy choice of p_i, there is no total equilibrium, and thus we may have to try several (or all) of his strategies in order to find an equilibrium. Second, if p_i has parents in both subgames, we must consider both subgames when reasoning about p_i, eliminating our ability to decouple them. Our algorithm below addresses both of these difficulties.

We decompose the graph into a set of overlapping clusters $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_\ell$, where each $\boldsymbol{C}_l \subseteq \{p_1, \ldots, p_n\}$. These clusters are organized into a tree $\mathcal{T}$. If $\boldsymbol{C}_l$ and $\boldsymbol{C}_m$ are two neighboring clusters, we define $\boldsymbol{S}_{lm}$ to be the intersection $\boldsymbol{C}_l \cap \boldsymbol{C}_m$. If $p_i \in \boldsymbol{C}_m$ is such that $\mathcal{P}_i \subseteq \boldsymbol{C}_m$, then we say that p_i is *associated* with $\boldsymbol{C}_m$. If all of a node's parents are contained in two clusters (and are therefore in the separator between them), we associate it arbitrarily with one cluster or the other.

Definition 5: We say that $\mathcal{T}$ is a *cluster tree* for a graphical game if the following conditions hold:

- **Running intersection:** If $p_i \in \boldsymbol{C}_l$ and $p_i \in \boldsymbol{C}_m$ then p_i is also in every $\boldsymbol{C}_o$ that is on the (unique) path in $\mathcal{T}$ between $\boldsymbol{C}_l$ and $\boldsymbol{C}_m$.
- **No interaction:** All p_i are associated with a cluster.

The *no interaction* condition implies that the best response criterion for players in the separator involves at most one of the two neighboring clusters, thereby eliminating the interaction with both subgames.

We now use a CSP to find an assignment to the separators that is consistent with some global equilibrium. We have one CSP variable for each separator $\boldsymbol{S}_{lm}$, whose value space are joint strategies $\theta_{\boldsymbol{S}_{lm}}$ for the players in the separator. We have a binary constraint for every pair of neighboring separators $\boldsymbol{S}_{lm}$ and $\boldsymbol{S}_{mo}$ that is satisfied iff there exists a strategy profile $\boldsymbol{\theta}$ for $\boldsymbol{C}_m$ for which the following conditions hold:

1. $\boldsymbol{\theta}$ is consistent with the separators $\theta_{\boldsymbol{S}_{lm}}$ and $\theta_{\boldsymbol{S}_{mo}}$.
2. For each p_i associated with $\boldsymbol{C}_m$, the strategy θ_i is an ϵ-best response to $\theta_{\mathcal{P}_i}$; note that all of p_i's parents are in $\boldsymbol{C}_m$, so their strategies are specified.

It is not hard to show that an assignment $\theta_{\boldsymbol{S}_{lm}}$ for the separators that satisfies all these constraints is consistent with an approximate global equilibrium. First, the constraints assert that there is a way of completing the partial strategy profile with a strategy profile for the players in the clusters. Second, the running intersection property implies that if a player appears in two clusters, it appears in every separator along the way; condition (1) then implies that the same strategy is assigned to that player in all the clusters where it appears. Finally, according to the *no interaction* condition, each player is associated with some cluster, and that cluster specifies the strategies of its parents. Condition (2) then tells us that this player's strategy is an ϵ-best response to its parents. As all players are playing ϵ-best responses, the overall strategy profile is an equilibrium.

There remains the question of how we determine the existence of an approximate equilibrium within a cluster given strategy profiles for the separators. If we use the CSP algorithm, we have gained nothing: using variable elimination within each cluster is equivalent to using variable elimination (using some particular ordering) over the entire CSP. However, we can solve each subgame using our hill climbing approach, giving us yet another hybrid algorithm — one where a CSP approach is used to combine the answers obtained by the hill-climbing algorithm in different clusters.

6 Experimental Results

We tested hill climbing, cost minimization, and the approximate equilibrium refinement hybrid on two types of games.

The first was the Road game described earlier. We tested two different types of payoffs. One set of payoffs corresponded to a situation where each developer can choose to build a park, a store, or a housing complex; stores want to be next to houses but next to few other stores; parks want to be next to houses; and houses want to be next to exactly one store and as many parks as possible. This game has pure strategy equilibria for all road lengths; thus, it is quite easy to solve using cost minimization where only the pure strategies of each developer are considered. A 200 player game can be solved in about 1 second. For the same 200 player game, hill climbing took between 10 and 15 seconds to find an approximate equilibrium with ϵ between .01 and .04 (the payoffs range from 0 to 2).

In the other payoff structure, each land developer plays a game of paper, rock, scissors against each of his neighbors; his total payoff is the sum of the payoffs in these separate games, so that the maximum payoff per player is 3. This game has no pure strategy equilibria; thus, we need to choose a finer discretization in order to achieve reasonable results. Fig. 2(a),(b) shows the running times and equilibria quality for each of the three algorithms. Cost minimization was run with a grid density of 0.2 (i.e., the allowable strategies all have components that are multiples of 0.2). Since each player has three possible actions, the resulting grid has 21 strategies per player. The hybrid algorithm was run starting from the strategy computed by hill-climbing. The nearby area was then discretized so as to have 6 strategies per player within a region of size roughly .05 around the current equilibrium. We ran the hybrid as described above until the total size is less than .00001.

Each algorithm appears to scale approximately linearly with the number of nodes, as expected. Given that the number of strategies used for the hybrid is less than that used for the actual variable elimination, it is not surprising that cost minimization takes considerably longer than the hybrid. The equilibrium error is uniformly low for cost minimization; this is not surprising as, in this game, the uniform strategy $(1/3, 1/3, 1/3)$ is always an equilibrium. The quality of the equilibria produced by all three algorithms is fairly good, with a worst ϵ value of about 10% of the maximum payoffs in the game. The error of the equilibria produced by hill climbing grows with the game size, a consequence of the fact that the hill-climbing search is over a higher-dimensional space. Somewhat surprising is the extent to which the hybrid approach improves the quality of the equilibria, at least for this type of game.

We also tested the algorithms on symmetric 3-action games structured as a ring of rings, with payoffs chosen at random from $[0, 1]$. The results are shown in Fig. 2(c),(d). For the graph shown, we varied the number of nodes on the internal ring; each node on the inner ring is also part of an outer ring of size 20. Thus, the games contain as many as 400 nodes. For this set of results, we set the gridding density for cost minimization to 0.5, so there were 6 strategies per node. The reduced strategy space explains why the algorithm is so much faster than the refinement hybrid: each step of the hybrid is similar to an entire run of cost minimization (for these graphs, the hybrid is run approximately 40 times).

The errors obtained by the different algorithms are somewhat different in the case of rings of rings. Here, refinement only improves accuracy by about a factor of 2, while cost minimization is quite accurate. In order to explain this, we tested simple rings, using cost minimization over only pure strategies. Based on 1000 trial runs, for 20 player rings, the best pure strategy equilibria has $\epsilon = 0$ 23.9% of the time; between $\epsilon \in [0, .05]$ 45.8% of the time; $\epsilon \in [.05, .1]$ 25.7%; and $\epsilon > .1$ 4.6%. We also tested (but did not include results for) undirected trees with random payoffs. Again, using a low gridding density for variable elimination, we obtained results similar to those for rings of rings. Thus, it appears that, with random payoffs, fairly good equilibria often exist in pure strategies.

Clearly the discretization density of cost minimization has a huge effect on the speed of the algorithm. Fig. 2(e)&(f) shows the results for CMP using different discretization levels as well as for hill climbing, over simple rings of various sizes with random payoffs in [0,1]. The level of discretization impacts performance a great deal, and also noticeably affects solution quality. Somewhat surprisingly, even the lowest level of discretization performs better than hill climbing. This is not in general the case, as variable elimination may be intractable for games with high graph width.

In order to get an idea of the extent of the improvement relative to standard, unstructured approaches, we converted each graphical game into a corresponding strategic form game (by duplicating entries), which expands the size of the game exponentially. We then attempted to find equilibria using the available game solving package Gambit[1] specifically using the QRE algorithm with default settings. (QRE seems to be the fastest among the algorithms implemented in Gambit). For a road length of 1 (a 2-player game) QRE finds an equilibrium in 20 seconds; for a road of length 2, QRE takes 7min56sec; and for a road of length 3, about 2h30min.

Overall, the results indicate that these algorithms can find good approximate equilibria in a reasonable amount of time. Cost minimization has a much lower variance in running time, but can get expensive when the grid size is large. The quality of the answers obtained even with coarse grids are often surprisingly good, particularly when random payoffs are used so that there are pure strategy profiles that are almost equilibria. Our algorithms provide us with a criterion for evaluating the error of a candidate solution, allowing us to refine our answer when the error is too large. In such cases, the hybrid algorithm is often a good approach.

[1] http://www.hss.caltech.edu/gambit/Gambit.html.

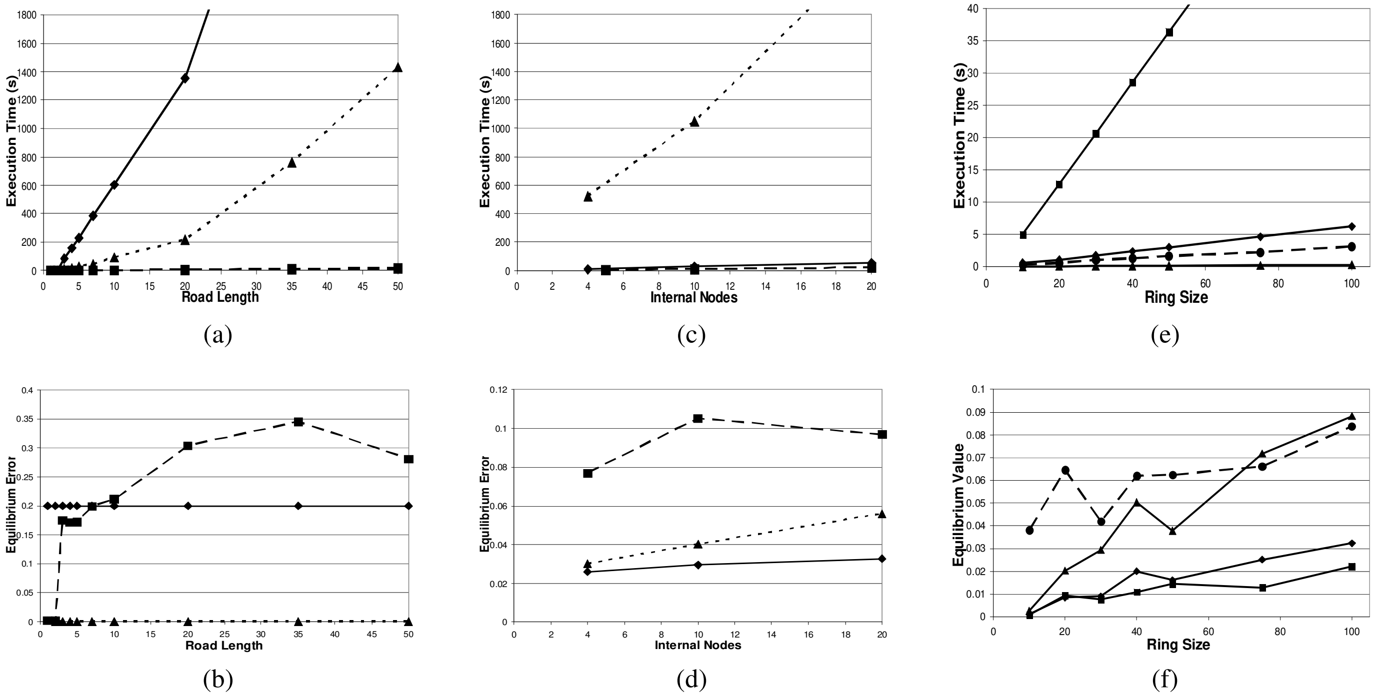


Figure 2: Comparison of Algorithms as number of players varies: Dashed for hill climbing, solid for cost minimization, dotted for refinement. Road games: (a) running time; (b) equilibrium error. Ring of rings: (c) running time; (d) equilibrium error. CMP on single ring with different grid density and hill climbing in simple ring. Dashed line indicates hill climbing, solid lines with squares, diamonds, triangles correspond to grid densities of 1.0 (3 strategies), 0.5 (6 strategies), and 0.333 (10 strategies) respectively. (e) running time; (f) equilibrium error.

7 Conclusions

In this paper, we considered the problem of collaboratively finding approximate equilibria in a situation involving multiple interacting agents. We focused on the idea of exploiting the locality of interaction between agents, using graphical games as an explicit representation of this structure. We provided two algorithms that exploit this structure to support solution algorithms that are both computationally efficient and utilize distributed collaborative computation that respects the "lines of communication" between the agents. Both strongly use the locality of regret: hill climbing in the score function, and CSP in the formulation of the constraints. We showed that our techniques provide good solutions for games with a very large number of agents.

We believe that our techniques can be applied much more broadly; in particular, we plan to apply them in the much richer *multi-agent influence diagram* framework of (Koller & Milch 2001), which provides a structured representation, similar to graphical games, but for substantially more complex situations involving time and information.

Acknowledgments. We are very grateful to Ronald Parr for many useful discussions. This work was supported by the DoD MURI program administered by the Office of Naval Research under Grant N00014-00-1-0637, and by Air Force contract F30602-00-2-0598 under DARPA's TASK program.

References

Bertele, U., and Brioschi, F. 1972. *Nonserial Dynamic Programming*. New York: Academic Press.

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

Kearns, M.; Littman, M.; and Singh, S. 2001a. Graphical models for game theory. In *Proc. UAI*.

Kearns, M.; Littman, M.; and Singh, S. 2001b. An efficient exact algorithm for singly connected graphical games. In *Proc. 14th NIPS*.

Koller, D., and Milch, B. 2001. Multi-agent influence diagrams for representing and solving games. In *Proc. IJCAI*.

LaMura, P. 2000. Game networks. In *Proc. UAI*, 335–342.

Lauritzen, S. L., and Spiegelhalter, D. J. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Stat. Soc. B* 50(2):157–224.

McKelvey, R., and McLennan, A. 1996. Computation of equilibria in finite games. In *Handbook of Computational Economics*, volume 1. Elsevier Science. 87–142.

McKelvey, R. 1992. A Liapunov function for Nash equilibria. unpublished.

Nash, J. 1950. Equilibrium points in n-person games. *PNAS* 36:48–49.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Pearson, M., and La Mura, P. 2001. Simulated annealing of game equilibria: A simple adaptive procedure leading to nash equilibrium. Unpublished manuscript.

von Neumann, J., and Morgenstern, O. 1944. *Theory of games and economic behavior*. Princeton Univ. Press.