Local-Effect Games, and an Algorithm for Computing their Equilibria

Navin Bhat
Department of Physics
University of Toronto

Kevin Leyton-Brown
Department of Computer Science
University of British Columbia

1 Introduction

There is much interest in modeling large, real-world problems as games and computing these games' equilibria. This interest has driven recent advances in algorithms for computing Nash equilibria of general games. One recent, general-purpose algorithm is the continuation method of Govindan and Wilson [2003], a gradient-following algorithm which is based on topological insight into the graph of the Nash equilibrium correspondence by Kohlberg and Mertens [1986]. The worst-case complexity of Govindan and Wilson's algorithm is open because the worst-case number of gradient-following steps is not known; however, in practice the algorithm's runtime is dominated by the computation of the gradient (mainly the Jacobian of the payoff function), which is both best- and worst-case exponential in the number of agents. Unfortunately this algorithm, like other state of the art algorithms for computing Nash equilibria, is generally practical only on games having relatively small numbers of players and actions.

Researchers have begun to investigate compact game representations that can be leveraged to yield more efficient computation on large games. One influential class of representations exploits (strict) independencies between players' utility functions; this class includes Graphical Games [Kearns *et al.*, 2001] and Multi-Agent Influence Diagrams [Koller & Milch, 2001]. Recently, Blum *et al.* [2003] introduced the first general algorithm for computing exact equilibria in such games, based on a state-of-the-art algorithm for general games due to Govindan and Wilson (2003). This gradient-following algorithm has as its bottleneck step the exponential-time computation of the Jacobian of the payoff function. Blum *et al.*'s algorithm computes this Jacobian in time no worse than exponential in the tree width of the underlying utility function dependency graph—an exponential improvement—and is consequently able to compute equilibria of considerably larger games.

Although real-world games often have very regular structure, strict independence between agents' utility functions is not a common assumption in the game theory literature. A second approach to compactly representing games aims to be more realistic by focusing on games in which all agents can potentially affect each other's payoffs, but agents' abilities to affect each other depend on the actions they choose. In this paper we describe algorithms for computing the equilibria of local-effect games (LEGs), a class that is able to express any game and is compact for games having such context-specific independencies in agents' utilities.

2 Local-Effect Games

A local-effect game[1] is a tuple $< A, \mathbf{S}, \nu, u >$. Let $A = \{1, \ldots, n\}$ denote the set of agents in the game. $\mathbf{S} =< S_1, \ldots, S_n >$ is a tuple of sets of actions for each agent. Agents may have some action choices in common; let $S = \bigcup_{i \in A} S_i$ denote the set of distinct action choices. Let $\mathcal{D}$ denote the set of possible distributions of agents over actions, so for a given distribution $D \in \mathcal{D}$, denote by $D(s)$ the number of agents who chose action s.

Let G be a graph having one node for every action $s \in S$. The neighbor relation is given by $\nu : S \mapsto 2^S$. Let there be a directed edge from s' to s in G iff $s' \in \nu(s)$. Note that $s \in \nu(s)$ is possible. The utility function $u : S \times \mathcal{D} \mapsto \mathbb{R}$ maps from an action choice s and a distribution of agents D to a real-valued payoff. Observe that all agents have the same utility function. The utility function has the property that given any action s and any pair of distributions D and D',

$$[\forall s' \in \nu(s), D(s') = D'(s')] \Rightarrow u(s, D) = u(s, D').$$

In other words, for every i and j agent i's utility is independent of agent j's action choice conditional on agent j choosing an action which is not in the neighborhood of agent i's action choice. Let the in-degree of the local-effect graph be bounded by I.

LEGs can represent any game. This is easy to see after realizing that in an arbitrary game $\forall i, \forall j \neq i,\ S_i \bigcap S_j = \emptyset$: the agents' sets of action choices S_i are partitions of the nodes in G, and thus $\forall s \in S,\ D(s) \in \{0, 1\}$. The LEG representation becomes more compact than normal form as agents begin to have actions in common, with utility functions depending only on the *number* of agents taking these actions rather than on the *identities* of the agents.

3 Computing Equilibria of LEGs

Our work adapts Govindan and Wilson's algorithm to the special case of computing the Nash equilibria of local-effect games. We prove that considerable computational gains can be realized in the computation of the payoff Jacobian

[1]A more restricted version of LEGs was first introduced in [Leyton-Brown & Tennenholtz, 2003]. This work concentrated on identifying games that are guaranteed to have pure-strategy Nash equilibria and on characterizing the intersection between (restricted) LEGs and Rosenthal's congestion games [Rosenthal, 1973].

when structure inherent in the LEG representation is taken into account. Interestingly, though both the algorithm-construction strategy we employed and the size of our computational gains are similar to what is reported in the work of Blum *et al.* [2003], the details of our algorithm are inherently different, and Blum *et al.*'s algorithm is inapplicable to LEGs.[2]

Our first technical result is an algorithm for computing the payoff Jacobian for general LEGs. We prove that this algorithm's computational complexity is polynomial in $|S|$, in n^I and in $(I+1)^n$. This is an exponential improvement over Govindan and Wilson's method, which requires time polynomial in $|S|^n$.

Next, we consider fully symmetric LEGs in which all agents have the same action choices; as Nash proved in his seminal paper [Nash, 1950], such games always have symmetric equilibria. We describe an algorithm for computing the payoff Jacobian in the case of computing a symmetric equilibrium in this case, and show that it requires time polynomial in $|S|$ and n^I. Thus, in the case where the in-degree of the graph is bounded by a constant, our algorithm is able to compute the payoff Jacobian in polynomial time.

Finally, we show techniques for realizing an additional computational speedup. Specifically, we present dynamic programming techniques to compute successive elements of the Jacobian more quickly once the first element has been computed.

References

Blum, B., Shelton, C., & Koller, D. (2003). A continuation method for Nash equilibria in structured games. *IJCAI*.

Govindan, S., & Wilson, R. (2003). A global Newton method to compute Nash equilibria. *J. Economic Theory*, *110*, 65–86.

Kearns, M., Littman, M., & Singh, S. (2001). Graphical models for game theory. *UAI*.

Kohlberg, E., & Mertens, J. (1986). On the strategic stability of equilibria. *Econometrica*, *54*(5), 1003–1038.

Koller, D., & Milch, B. (2001). Multi-agent influence diagrams for representing and solving games. *IJCAI*.

Leyton-Brown, K., & Tennenholtz, M. (2003). Local-effect games. *IJCAI*.

Nash, J. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, *36*, 48–49.

Rosenthal, R. (1973). A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, *2*, 65–67.

[2]Unfortunately, the description of our algorithm requires the introduction of more notation than space permits. Details are available in the full version of the paper.