Simple Search Methods for Finding a Nash Equilibrium

Ryan Porter and **Eugene Nudelman** and **Yoav Shoham**
Computer Science Department
Stanford University
Stanford, CA 94305
{rwporter,eugnud,shoham}@cs.stanford.edu

Introduction

Nash equilibrium (NE) is arguably the most important concept in game theory, and yet remarkably little is known about the problem of computing a sample NE in a normal-form game. All evidence points to this being a hard problem, but its precise complexity is unknown. In this work (which appears as a full paper in (Porter, Nudelman, & Shoham 2004)), we present a pair of simple search methods (one for 2-player games and the other for n-player games) that perform well in practice.

Both algorithms are based on the same technique of searching through the space of support profiles, which specify the support of each player. While the general problem of computing a NE is a complementarity program, computing whether there exists a NE consistent with a support profile is a relatively easy feasibility program with the following three types of constraints for each player: (1) he must only play actions in his support with positive probability, (2) he must be indifferent between these actions, and (3) he must not strictly prefer an action outside of his support.

The algorithms decompose the search space by separately considering support size profiles (which specify the number of actions in the support of each player), giving precedence to profiles that are small and balanced. Since it turns out that games drawn from classes that researchers have focused on in the past tend to have (at least one) NE with a very small support, our algorithms are often able to find one quickly. Thus, this paper is as much about the properties of NE in games of interest as it is about novel algorithmic insights.

We emphasize, however, that we are not cheating in the selection of games on which we test. Past algorithms were tested almost exclusively on "random" games. We tested on these too (indeed, we will have more to say about how "random" games vary along at least one important dimension), but also on many other distributions (24 in total). To this end we use GAMUT, a recently introduced computational testbed for game theory (Nudelman et al. 2004). Our results are quite robust across all games tested.

For each support size profile, the algorithms use a backtracking procedure to search through the space of support profiles of the specified size. They instantiate each player's support separately, and, after each instantiation, they prune the set of available actions for each uninstantiated support using iterated removal of strictly dominated strategies. If an action in an instantiated support is found to be dominated, then the current set of instantiations cannot be part of an NE, and the algorithm backtracks to the last instantiation.

Experimental Results

Using games generated by GAMUT, we tested our algorithms against the state of the art: Lemke-Howson (Lemke & Howson 1964) for 2-player games, and Simplicial Subdivision (van der Laan, Talman, & van der Heyden 1987) and Govindan-Wilson (which was developed by (Govindan & Wilson 2003) and extended and implemented by (Blum, Shelton, & Koller 2003)) for n-player games.

A distribution of particular importance is the one most commonly tested on in previous work: the "Uniformly Random Game", in which every payoff in the game is drawn independently from an identical uniform distribution. Also important are distributions which fall under a "Covariance Game" model studied by (Rinott & Scarsini 2000), in which the payoffs for the n agents for each action profile are drawn from a multivariate normal distribution in which the covariance ρ between the payoffs of each pair of agents is identical. When $\rho = 1$, the game is common-payoff, while $\rho = \frac{-1}{N-1}$ yields minimal correlation, which occurs in zero-sum games. Thus, by altering ρ, we can smoothly transition between these two extreme classes of games.

Our experiments were executed on a cluster of 12 dual-processor, 2.4GHz Pentium machines, running Linux 2.4.20. We capped runs for all algorithms at 1800 seconds. When describing the statistics used to evaluate the algorithms, we will use "unconditional" to refer to the value of the statistic when timeouts are counted as 1800 seconds, and "conditional" to refer to its value excluding timeouts.

Results for Two-Player Games

In the first set of experiments, we compared the performance of Algorithm 1 (our 2-player algorithm) to that of Lemke-Howson (implemented in Gambit, which added the preprocessing step of iterated removal of weakly dominated strategies) on 2-player 300-action games drawn from 24 different GAMUT distributions. Both algorithms were executed on 100 games drawn from each distribution. The time is measured in seconds and plotted on a logarithmic scale.

Figure 1(a) compares the unconditional median runtimes of the two algorithms, and shows that Algorithm 1 performs

better on all distributions.[1] However, this does not tell the whole story. For many distributions, it simply reflects the fact that there is a greater than 50% chance that the distribution will generate a game with a pure strategy NE, which our algorithm will then find quickly. Two other important statistics are the percentage of instances solved (Figure 1(b)), and the average runtime conditional on solving the instance (Figure 1(c)). Here, we see that Algorithm 1 completes far more instances on several distributions, and solves fewer on just a single distribution (6 fewer, on D23). Additionally, even on distributions for which we solve far more games, our conditional average runtime is 1 to 2 orders of magnitude faster.

Clearly, the hardest distribution for our algorithm is D6, the "Covariance Game" distribution in which the payoffs in which the covariance ρ is drawn uniformly at random from the range $[-1, 1]$. To further investigate this continuum, we sampled 300 values for ρ in the range $[-1, 1]$, with heavier sampling in the transition region and at zero. For each such game, we plotted a point for the runtime of both Algorithm 1 and Lemke-Howson in Figure 1(d).[2] The theoretical results of (Rinott & Scarsini 2000) suggest that the games with lower covariance should be more difficult for Algorithm 1, because they are less likely to have a pure strategy Nash equilibrium. Nevertheless, it is interesting to note the sharpness of the transition that occurs in the $[-0.3, 0]$ interval. More surprisingly, a similarly sharp transition also occurs for Lemke-Howson, despite the fact that the two searches operate in unrelated ways. Finally, it is important to note that the transition region for Lemke-Howson is shifted to the right by approximately 0.3, and that, on instances in the easy region for both algorithms, Algorithm 1 is still an order of magnitude faster.

In the third set of experiments we explore the scaling behavior of both algorithms on the "Uniformly Random Game" distribution (D18), as the number of actions increases from 100 to 1000. For each multiple of 100, we generated 20 games. Because space constraints preclude an analysis similar to that of Figures 1(a) through 1(c), we instead plot in Figure 1(e) the *unconditional* average runtime over 20 instances for each data size, with a timeout counted as 1800s. While Lemke-Howson failed to solve any game with more than 600 actions and timed out on some 100-action games, Algorithm 1 solved all instances, and, without the help of cutoff times, still had an advantage of 2 orders of magnitude at 1000 actions.

Results for N-Player Games

Next, we compared Algorithm 2 (our n-player algorithm) to Govindan-Wilson and Simplicial Subdivision (also implemented in Gambit, and thus combined with iterated removal of weakly dominated strategies). First, we tested the algorithms on 100 6-player, 5-action games from each of 22 of GAMUT's n-player distributions.

[1] Obviously, the lines connecting data points across distributions for a particular algorithm are meaningless– they were only added to make the graph easier to read.

[2] The capped instances for Algorithm 1 were perturbed slightly upward on the graph for clarity.

Once again, Figures 2(a), 2(b), and 2(c) show unconditional median runtime, percentage of instances solved, and conditional average runtime, respectively. Algorithm 2 has a very low unconditional median runtime, for the same reason that Algorithm 1 did for two-player games, and outperforms both other algorithms on all distributions. While this dominance does not extend to the other two metrics, the comparison still favors Algorithm 2.

We again investigate the relationship between ρ and the hardness of games under the "Covariance Game" model. For general n-player games, minimal correlation under this model occurs when $\rho = -\frac{1}{n-1}$. Thus, we can only study the range $[-0.2, 1]$ for 6-player games. Figure 2(d) shows the results for 6-player 5-action games. Algorithm 2, over the range $[-0.1, 0]$, experiences a transition in hardness that is even sharper than that of Algorithm 1. Simplicial Subdivision also undergoes a transition, which is not as sharp, that begins at a much larger value of ρ (around 0.4). However, the running time of Govindan-Wilson is only slightly affected by the covariance, as it neither suffers as much for small values of ρ nor benefits as much from large values.

Finally, Figures 2(e) and 2(f) compare the scaling behavior (measured by the unconditional average runtimes): the former holds the number of players constant at 6 and varies the number of actions from 3 to 8, while the latter holds the number of actions constant at 5, and varies the number of players from 3 to 8. In both experiments, both Simplicial Subdivision and Govindan-Wilson solve no instances for the largest two sizes, while Algorithm 2 solves most games.

References

Blum, B.; Shelton, C. R.; and Koller, D. 2003. A continuation method for Nash equilibria in structured games. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence.

Govindan, S., and Wilson, R. 2003. A global newton method to compute Nash equilibria. In Journal of Economic Theory.

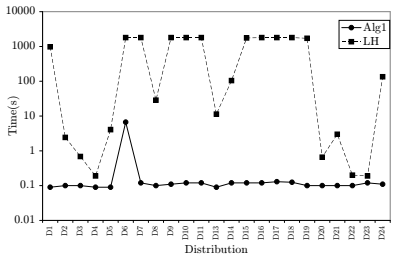
Lemke, C., and Howson, J. 1964. Equilibrium points of bimatrix games. Journal of the Society for Industrial and Applied Mathematics 12:413–423.

Nudelman, E.; Wortman, J.; Shoham, Y.; and Leyton-Brown, K. 2004. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In AAMAS-04.

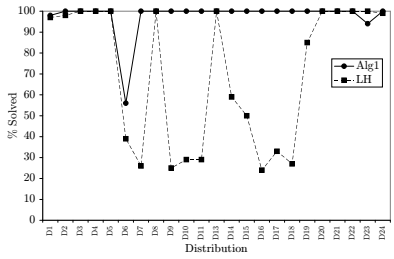
Porter, R.; Nudelman, E.; and Shoham, Y. 2004. Simple search methods for finding a nash equilibrium. In Proceedings of the Nineteenth National Conference on Artificial Intelligence.

Rinott, Y., and Scarsini, M. 2000. On the number of pure strategy Nash equilibria in random games. Games and Economic Behavior 33:274–293.

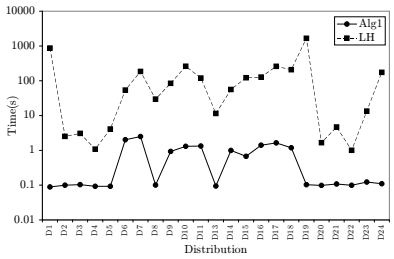
van der Laan, G.; Talman, A.; and van der Heyden, L. 1987. Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling. Mathematics of Operations Research.



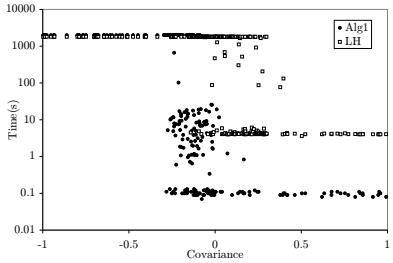
(a) Unconditional median runtime



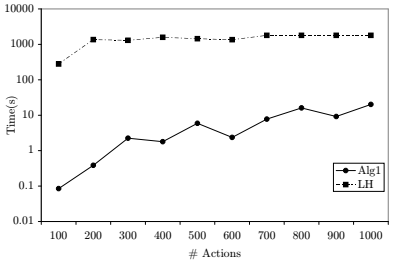
(b) Percentage solved



(c) Average time on solved instances

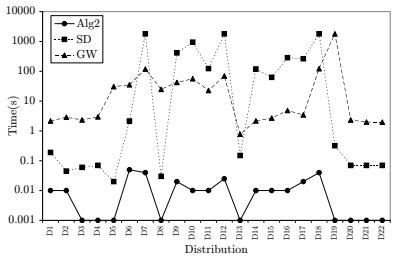


(d) Runtime vs. Covariance

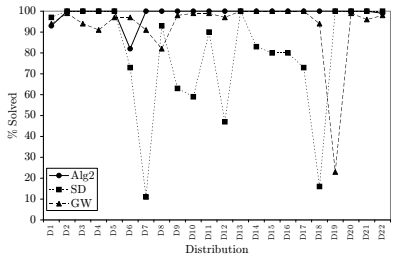


(e) Unconditional average vs. Actions

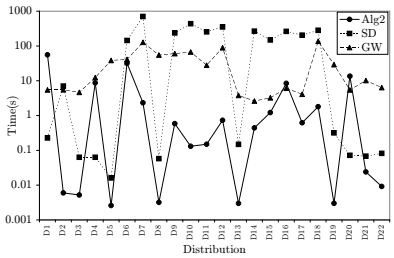
Figure 1: Comparison of Algorithm 1 and Lemke-Howson on 2-player games. Subfigures (a)-(d) are for 300-action games.



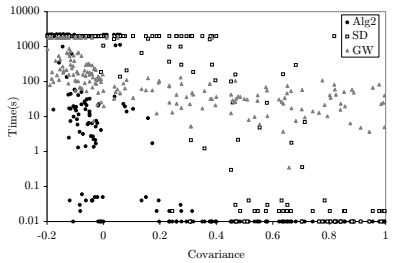
(a) Unconditional median runtimes



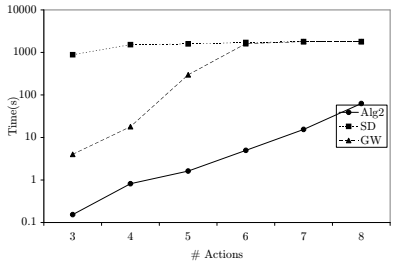
(b) Percentage solved



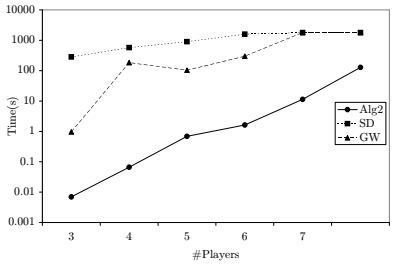
(c) Average time on solved instances



(d) Runtime vs. Covariance



(e) Unconditional average runtime vs. Actions, on 6-player games



(f) Unconditional average runtime vs. Players, on 5-action games

Figure 2: Comparison of Algorithm 2, Simplicial Subdivision, and Govindan-Wilson. Subfigures (a)-(d) are for 6-player, 5-action games.