Understanding Game-Theoretic Algorithms: The Game Matters

Eugene Nudelman Jennifer Wortman Yoav Shoham
{eugnud;jwortman;shoham}@cs.stanford.edu Stanford University, Stanford, CA 94305
Kevin Leyton-Brown
kevinlb@cs.ubc.ca University of British Columbia, Vancouver, BC V6T 1Z4

1. Introduction

There is growing interest in algorithms for solving computational problems associated with game-theoretic domains, and in particular, algorithms for computing Nash equilibria. It is often difficult to offer theoretical guarantees about the performance of such algorithms, particularly in the average case. Even the worst-case computational complexity of computing Nash equilibria remains an open problem; while all evidence suggests that the problem is hard (e.g., [2]), the precise complexity class into which it falls is unknown [7]. In the absence of theoretical guarantees, researchers evaluating algorithms for game-theoretic problems often perform empirical tests.

It has been observed in many problem domains that algorithm performance varies substantially across different "reasonable" distributions of problem instances, even when problem size is held constant. When we examine the empirical tests that have been performed on algorithms that take games as their inputs, we find that they have typically been small-scale and involved very particular choices of games (in many cases, only generic games generated uniformly at random). Such tests can be appropriate for limited proofs-of-concept, but cannot say much about an algorithm's expected performance across a broader range of domains. For this, a comprehensive body of test data is required.

Our work aims to fill this gap in available test data, identifying interesting sets of non-generic games comprehensively and with as little bias as possible. We began with a broad literature survey, examining hundreds of books and papers by game theorists, economists, computer scientists, political scientists and others from the past 50 years. After an analysis of common structures in the games we found—and an extensive software-engineering effort—we built GAMUT, our game generation software. GAMUT generates games from 71 parameterized distributions which range from specific two-by-two matrix games with little variation (e.g., Chicken) to broad classes extensible in both number of players and number of actions (e.g., games that can be encoded compactly in the Graphical Game representation). Our software, along with a database listing properties of and relations between games, is now publicly available. Figure 1 demonstrates a snapshot of some classes of games found in our database, where arrows denote the subset relation. Table 1 lists 35 parameterized generators that support all of the generative sets in our taxonomy.

Arms Race	Grab the Dollar	Polymatrix Game
Battle of the Sexes	Graphical Game	Prisoner's Dilemma
Bertrand Oligopoly	Greedy Game	Random Games
Bidirectional LEG	Guess 2/3 Average	Rapoport's Distribution
Chicken	Hawk and Dove	Rock, Paper, Scissors
Collaboration Game	Local-Effect Game	Shapley's Game
Compound Game	Location Game	Simple Inspection Game
Congestion Game	Majority Voting	Traveler's Dilemma
Coordination Game	Matching Pennies	Uniform LEG
Cournot Duopoly	Minimum Effort Game	War of Attrition
Covariant Game	N-Player Chicken	Zero Sum Game
Dispersion Game	N-Player Pris Dilemma	

Table 1. Games in GAMUT

2. Running the GAMUT

In the remainder of this paper we describe our use of GAMUT to evaluate the empirical properties of three algorithms for computing Nash equilibria. We demonstrate that a broad test suite is needed in this domain by showing that algorithm performance varies dramatically across distributions.[1] For two-player games we used the Gambit [5] implementation of the Lemke-Howson algorithm [4]. For n-player games we used the Gambit implementation of Simplicial Subdivision [9], as well as a recently introduced continuation method by Govindan and Wilson [3] (implemented in the GameTracer package [1]).

One factor that can have a significant effect on an algorithm's runtime is the size of its input. Since our goal was to investigate the extent to which runtimes vary as the result of differences between distributions, we studied fixed-size games. To make sure that our findings were not artifacts

1 In the full version of this paper [6] we present experimental results for another problem—reinforcement learning in repeated games—and also provide more information about the construction and contents of our game generator.

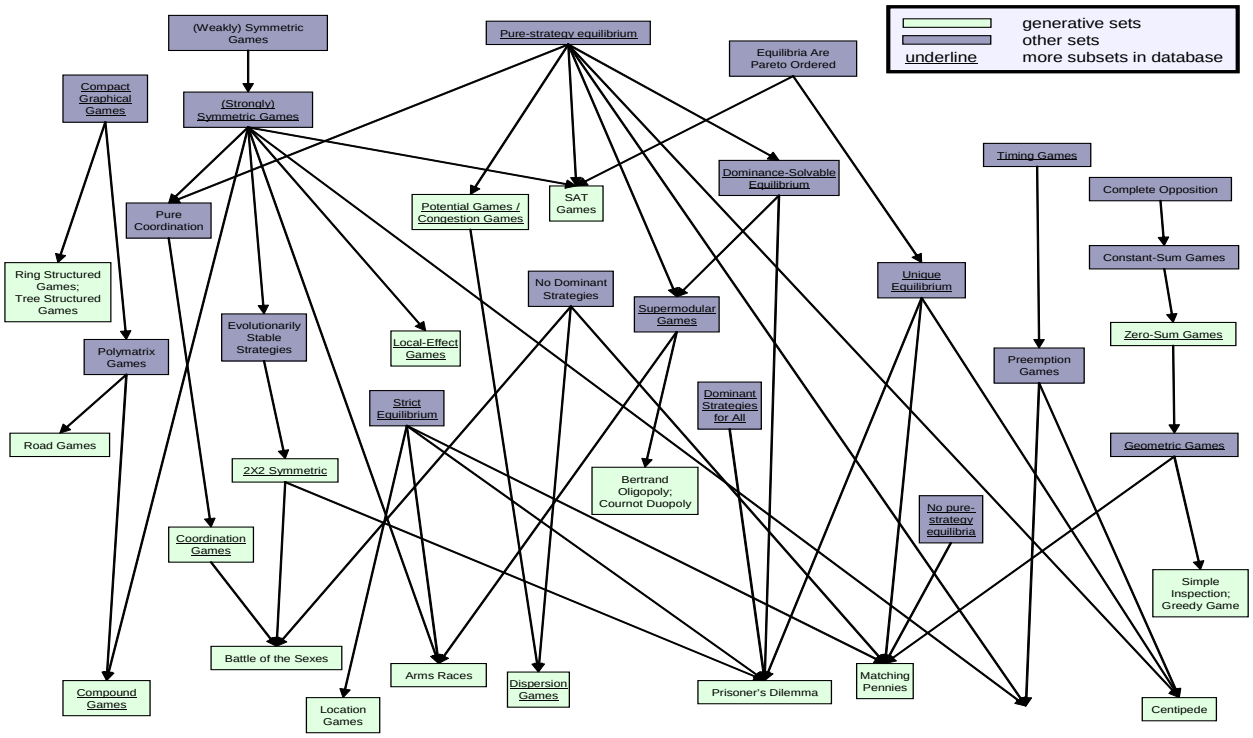

Figure 1. GAMUT Taxonomy (Partial)

of any particular problem size we compared results across several fixed problem sizes. We ran the Lemke-Howson algorithm on games with 2 players, 150 actions and 2 players, 300 actions. Because Govindan-Wilson is very similar to Lemke-Howson on two-player games and is not optimized for this case [1], we did not run it on these games. We ran Govindan-Wilson and Simplicial Subdivision on games with 6 players, 5 actions and 18 players, 2 actions. For each problem size and distribution, we generated 100 games.

Both to keep our machine-time demands manageable and to keep the graphs in this paper from getting too cluttered, we chose not to use *all* of the GAMUT generators. Instead, we chose a broad and representative slate of 22 distributions from GAMUT. Some of our generators (e.g., Graphical Games, Polymatrix games, and Local Effect Games–LEGs) are parameterized by graph structure; we split these into several sub-distributions based on the kind of graph used. Suffixes "-CG", "-RG", "-SG", "-SW" and "-Road" indicate, respectively, complete, random, star-shaped, small-world, and road-shaped graphs. Another distribution that we decided to split was the Covariant Game distribution, which implements the random game model of [8]. In this distribution, payoffs for each outcome are generated from a multivariate normal distribution, with correlation between all pairs of players held at some constant ρ. With $\rho = 1$ these games are common-payoff, while $\rho = \frac{-1}{n-1}$ yields minimum correlation and leads to zero-sum games in the two-player case. Rinott and Scarsini show that the probability of the existence of a pure strategy Nash equilibrium in these games varies as a monotonic function of ρ, which makes the games computationally interesting. Suffixes "-Pos", "-Zero", and "-Rand" indicate whether ρ was held at 0.9, 0, or drawn uniformly at random from $[\frac{-1}{n-1}, 1]$.

All our experiments were performed using a cluster of 12 dual-CPU 2.4GHz Xeon machines running Linux 2.4.20, and took about 120 CPU-days to run. We capped runs for all algorithms at 30 minutes (1800 seconds).

2.1. Experimental Results

Lemke-Howson, Simplicial Subdivision and Govindan-Wilson are all very complicated path-following numerical algorithms that offer virtually no theoretical guarantees. They all have worst-case running times that are at least exponential, but it is not known whether this bound is tight. On the empirical side, very little previous work has attempted to evaluate these algorithms. The best-known empirical results were obtained for generic games with payoffs drawn independently uniformly at random (in GAMUT, this would be the `random` generator). Our work may therefore represent the first systematic attempt to understand the empirical behavior of these algorithms on non-generic games.

Figure 2 shows each algorithm's performance across distributions for two different input sizes. The Y-axis shows

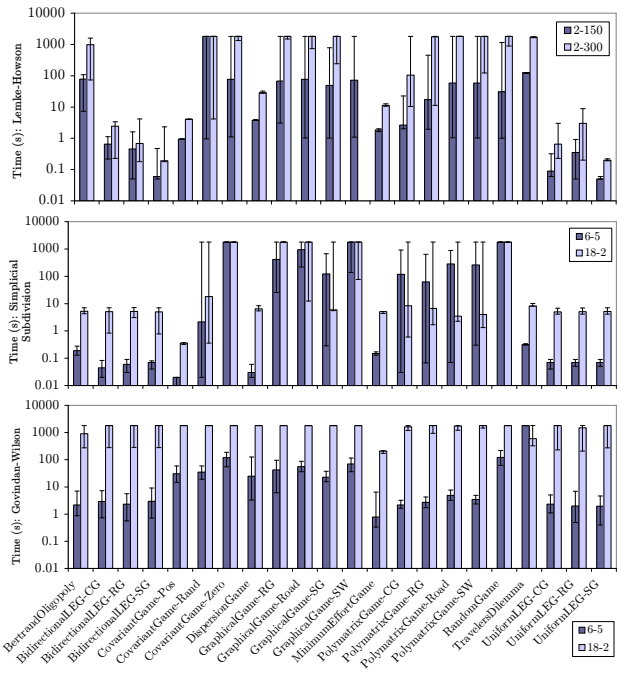

Figure 2. Solver Performance on Problems of Different Sizes

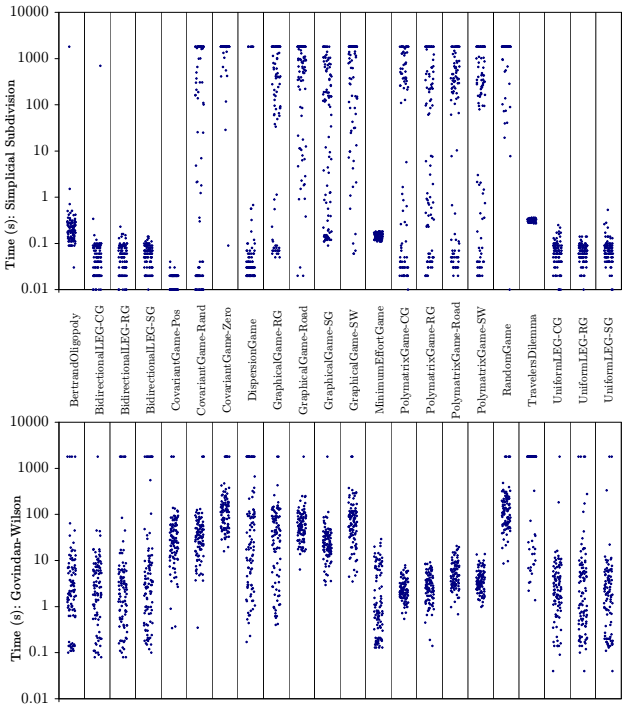

Figure 3. Runtime Distributions for 6-player, 5-action Games

CPU time measured in seconds and plotted on a log scale. Column height indicates median runtime over 100 instances, with the error bars showing the 25th and 75th percentiles. The most important thing to note about this graph is that each algorithm exhibits highly variable behavior across our distributions. This is less obvious in the case of the Govindan-Wilson algorithm on 18-player games, because this algorithm's runtime exceeds our cap for a majority of the problems. However, even on this dataset the error bars demonstrate that the distribution of runtimes varies substantially with the distribution. Moreover, for all three algorithms, we observe that this variation is not an artifact of one particular problem size.

Figure 3 illustrates runtime differences both across and among distributions for 6-player 5-action games. Each dot on the graph corresponds to a single run of an algorithm on a game. This graph shows that the distribution of algorithm runtimes varies substantially from one distribution to another, and cannot easily be inferred from 25th/50th/75th quartile figures such as Figure 2. The highly similar Simplicial Subdivision runtimes for Traveler's Dilemma and Minimum Effort Games are explained by the fact that these games can be solved by iterated elimination of dominated strategies—a step not performed by the GameTracer implementation of Govindan-Wilson. We note that distributions that are related to each other in our taxonomy usually give rise to similar—but not identical—algorithmic behavior.

References

[1] B. Blum, C. Shelton, and D. Koller. A continuation method for Nash equilibria in structured games. In *IJCAI*, 2003.

[2] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *IJCAI*, 2003.

[3] S. Govindan and R. Wilson. A global Newton method to compute Nash equilibria. In *Journal of Economic Theory*, 2003.

[4] C. Lemke and J. Howson. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12, 1964.

[5] R. D. McKelvey, A. McLennan, and T. Turocy. Gambit: game theory analysis software and tools, 1992. http://econweb.tamu.edu/gambit.

[6] E. Nudelman, J. Wortman, Y. Shoham, and K. Leyton-Brown. Run the gamut: A comprehensive approach to evaluating game-theoretic algorithms. In *AAMAS*, 2004.

[7] C. Papadimitriou. Algorithms, games, and the internet. In *STOC*, 2001.

[8] Y. Rinott and M. Scarsini. On the number of pure strategy Nash equilibria in random games. *Games and Economic Behavior*, 33, 2000.

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