CPSC 532
Exploiting k-symmetry for support enumeration on Action-Graph Games

April 26, 2009

Samantha Leung

Abstract

We implement an improvement to support enumeration on Action-Graph games by removing equivalent support profiles under k-symmetry. Our experimental results show small but consistent improvements in the run time of the algorithm. In addition, we propose a small modification to the IRSDS procedure.

1 Introduction

This project complements the author's current work with David Thompson and Kevin Leyton-Brown on an implementation of support enumeration on Action-Graph games (SEM-AGG). In support of SEM-AGG, we implement computational enhancements available from exploiting k-symmetry in games, and provide an empirical evaluation of the effectiveness of these changes. [5].

1.1 Action-Graph Games

We are interested in the computation of Nash equilibria in Action-Graph Games, a game representation introduced by Bhat and Leyton-Brown [1]. Similar to the graphical game[3], the Action-Graph Game (AGG) is a fully-expressive and compact representation that represents agents' actions as nodes of a directed graph. An agent's utility for choosing a certain action can be computed from the number of players that has chosen each of the neighboring nodes. A distribution of players over action nodes is called a *configuration.* One can see that utility functions for nodes with few neighbors can be represented compactly. Indeed, by exploiting utility function structures such as independence, anonmity, and additivity, games that require exponential space in other popular formats can be represented in polynomial space as AGGs.

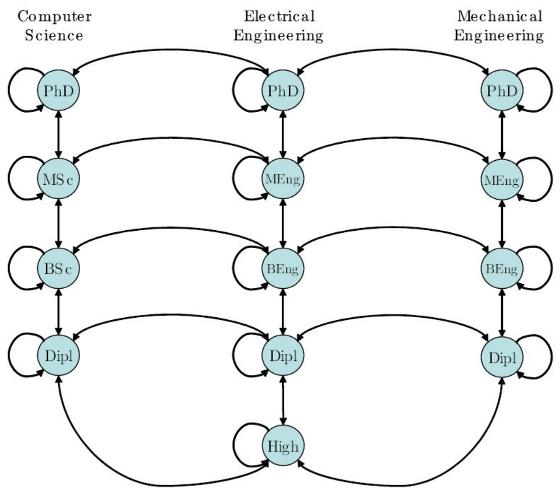

Figure 1: A JobMarketGame in the AGG representation, taken from [2].

In addition to being a compact representation, AGGs also allow for efficient computation on games when compared to the normal form. For example, Jiang et al presented a polynomial-time algorithm to compute an agent's expected utility in AGGs where nodes have bounded in-degrees [2]. This is an exponential speedup over the standard method of computing expected utility, which is polynomial in the size of the normal form and hence potentially exponential in the size of the equivalent AGG. Because the computation of expected utility is a bottle-neck for many equilibrium computation algorithms, the efficient computation of expected utility is very useful.

Thompson leveraged these dramatic efficiencies of AGGs for the computational equilibrium analyses of complex, realistic auction problems [6]. Not only did Thompson's experimental results provide new insight into rules in advertising auction settings; the experiments also shows the power of the AGG representation. These games would have been too big in normal form to even store one on a modern harddrive, let alone computation with the games.

Since it is beyond the scope of this project to give a technical introduction to the AGG representation, the reader is directed to [6] for a concise introduction, and [2] for a more comprehensive discussion of the AGG format.

1.2 Support enumeration

Recall that the *support* of a player's strategy is the set of actions played with positive probability, and a *support profile* specifies a support for each player. Support enumeration is a simple algorithm that searches through the support profile space to find Nash equilibria. Although simple, Porter et al [5] demonstrated that support enumeration for normal form games is empirically faster

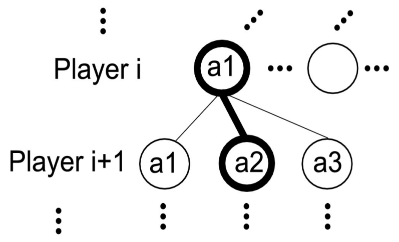

Figure 2: Abstract illustration of support enumeration. The current support profile has player i playing a_1 and player i+1 playing a_2.

than state of the art algorithms, Simplicial Subdivision and Govindan-Wilson.

We adopt the enumeration approach used in [5], that is, to perform a depth first search, with each level of the search tree containing the possible supports for one player. Figure 1.2 provides an illustration.

Because AGGs are compact and have been shown to facilitate efficient game theoretic computation [2], it is anticipated that by using support enumeration on AGGs, appreciable improvements can be achieved comparing to other equilibria computation algorithms on the normal form game.

In addition to being a simple algorithm using shallow heuristics, support enumeration is attractive in that it can find entire sets of equilibria; for example all pure strategy equilibria, all k-symmetric equilibria, or potentially all equilibria. Here we focus on finding one equilibrium, which, due to the order of enumeration, will be the equilibrium with the smallest support.

1.3 K-symmetry in games

A game is *symmetric* if all players are identical and indistinguishable. In a symmetric game, the utility of each player who chooses a certain action depends only on the distribution of the other players among the actions.

An AGG game is called k-symmetric if there are k classes of agents, where agents within each class share the same actions and thus payoffs, and therefore are identical to each other. However, while symmetry information is readily available in the AGG format, extracting player classes from a normal form game would require time polynomial in the size of the game. Therefore, k-symmetry considerations were not applicable for support enumeration algorithm in [5] for the normal form game.

Although support enumeration is commonly appreciated for its lack of deep heuristics, we feel that k-symmetry is a sufficiently shallow and general characteristic for AGGs. Furthermore, since all games are trivially n-symmetric (with n being the number of players), algorithms that assume k-symmetry can handle any AGG.

Because many games of interest are k-symmetric, we would also like to make use of symmetry information to reduce the search space during support enumeration.

[2] considers the role of k-symmetry in equilibrium computation and provides an efficient algorithm for computing expected utility for k-symmetric strategy profiles. However, we do not make use of these algorithms here because we are interested in finding general equilibria. Restricting our search to k-symmetric equilibria would likely mean finding a different equilibrium than that with the smallest support. Because we would like to provide a direct comparision between the performance with and without k-symmetric considerations, it is important that the same equilibria be found under both cases. Furthermore, although we can use the specialized k-symmetric algorithms only when k-symmetric support profiles are encountered, because the number of symmetric support profiles is very small comparing to generic support profiles, we anticipate that this effort will not yield significant improvements to the run time.

2 Observations

2.1 Equivalent support profiles

We call the tuple of support sizes for each player a *support size profile.*

Porter et al [5] showed that the shortest time to find an equilibrium is very often achieved by enumerating support profiles in the order of increasing total support sizes for all players, further ordered by the balance of the support sizes. The rationale is that small support equilibria with balanced supports often exists. Therefore, we enumerate support profiles ordered by their underlying support size profiles. A lexicographical ordering consistent with that of [5] is used for ordering support profiles with the same support sizes.

For simplicity, let us assume each of n agents has the same number of actions in their action sets, and denote this number by A. Since there are A^n distinct support size profiles, but only $\binom{n+a-1}{a}$ after taking k-symmetry into account, the reduction in search space should be noticeable in run time.

Given a fixed support size profile, there are also equivalent support profiles if two or more players in the same player class have the same support size. For example, if player 1 and player 2 belong to the same class and have the same support size 1, then player 1 playing a_1 and player 2 playing a_2 is equivalent to player 1 playing a_2 and player 2 playing a_1.

In our implementation, equivalent support size profiles are eliminated by limiting our search to the set of support size profiles that are weakly increasing within each player class. This removes equivalent support size profiles by making sure we only use one permutation from each equivalent class of support and support size profiles.

Similarly, for a given support size profile, by enumerating only support profiles that have weakly increasing supports, we avoid testing support profiles that are equivalent under k-symmetry.

2.2 IRSDS modification

Porter et al [5] also showed that one of the major helpers of the support enumeration algorithm is iterated removal of strictly dominated strategies (IRSDS). As per [5], instead of being applied only to the original actions, IRSDS is applied on partial support profiles as a means of quickly pruning support profiles that cannot be an equilibrium. Candidate strategies that contain strictly dominated actions cannot be part of an equilibrium and hence are removed. As more players are assigned supports, the outcome space gets smaller and IRSDS can prune more. Because checking for domination by mixed strategies is relatively expensive, we only check for domination by pure strategies.

When checking whether an action a_i dominates another action a_j, we check, given the current partial support profile, whether there are any possible configurations where a_j yields at least as much utility as a_i. As noted before, as the partial support profile becomes more restricted, the number of possible configurations decrease, and the number of dominated actions weakly increases.

One observation pertaining to k-symmetry is that, if we know that given a partial support profile, action a_i dominates action a_j for player p, then for any player p', if their domain is a superset of p's, then a_i must also dominate action a_j for p'.

The reason is that, when determining whether a_i dominates a_j for p, we compute the set of possible configurations that can result from a given partial support profile. If we had done the computation for p' instead of p, then p would have contributed to the possible configurations instead of p'. However, this set of possible configurations must be a subset of that produced by including p'. As a result, if there were no configurations where a_j does as least as well as a_i when p' is one of the other players, then there will still be no such configurations when p is part of the players determining the configurations.

Because removing an action from p' can only reduce the number of possible configurations, we know that we can proceed to remove a_j from all players p' whose action domain is a superset of p, in the same step.

Due to time constraints and technical malfunctions, we are unable to execute experiments with this IRSDS modification at this time. [1]

3 Experimental Setup

3.1 Game Distributions

We attempt to select a representative set of game distributions from GAMUT [4], subject to the constraint that only a limited number of games are currently available in the AGG format. In addition to GAMUT distributions, several

[1]For the sake of academic honesty the writer is compelled to elaborate on the situation before hardware malfunctions occured. A limited set of data has indeed been collected for the IRSDS modification. However, in over half of the instances, it appears that the algorithm has missed support profiles that could be equilibria. The writer suspects that this is either due to buggy implementation or flaws in the stated observations about the IRSDS procedure.

Ad-Auction variants from [6] have been added as representatives of 'real-life' games.

We have used a smaller number of distributions than in Porter's experiments [5], because the number of GAMUT distributions available in the AGG format is limited.

Our test distributions consists of 860 game instances in total, varying from 6 to 10 players and 5 to 12 actions, with 5 randomized instances per distribution, player and action combination. Table 3.1 show the GAMUT game distributions used.

Distribution
1 player class (symmetric game)
JobMarketGame
RandomSymmetricAGG-RandomGraph (3A edges)
RandomSymmetricAGG-RandomGraph (A edges)
RandomSymmetricAGG-SmallWorldGraph (each edge is present with 0.5 probability)
RandomSymmetricAGG-StarGraph
RandomSymmetricAGG-RandomGraph (A/2 edges)
CoffeeShopGame (With DecreasingFunction for home, near, far; SumFunction for combine)
3 player classes (3-symmetric)
IceCreamGame
n player classes
AdAuction (Weighted, PPC, GFP)
AdAuctionVarian (Unweighted, GSP)
AdAuctionYahoo (Unweighted, PPC, GSP)

The interested reader is directed to [6] for discussion on the different AdAuction settings, [2] for the CoffeeShopGame, IceCreamGame, and JobMarketGame settings, as well as GAMUT documentation for more information about the distributions.

3.2 Hardware

Our experiments were executed on the Arrow cluster with 50 machines, two 3.2 GHz CPUs and 2GB memory each, running Linux.

3.3 Algorithms

As stated earlier, we seek to evaluate the isolated performance difference provided by consideration of k-symmetry in the games. As a result, we ran two versions of SEM-AGG: one without any optimizations, and the other with consideration of equivalent support profiles. No other optimizations have been employed.

4 Experimental Observations

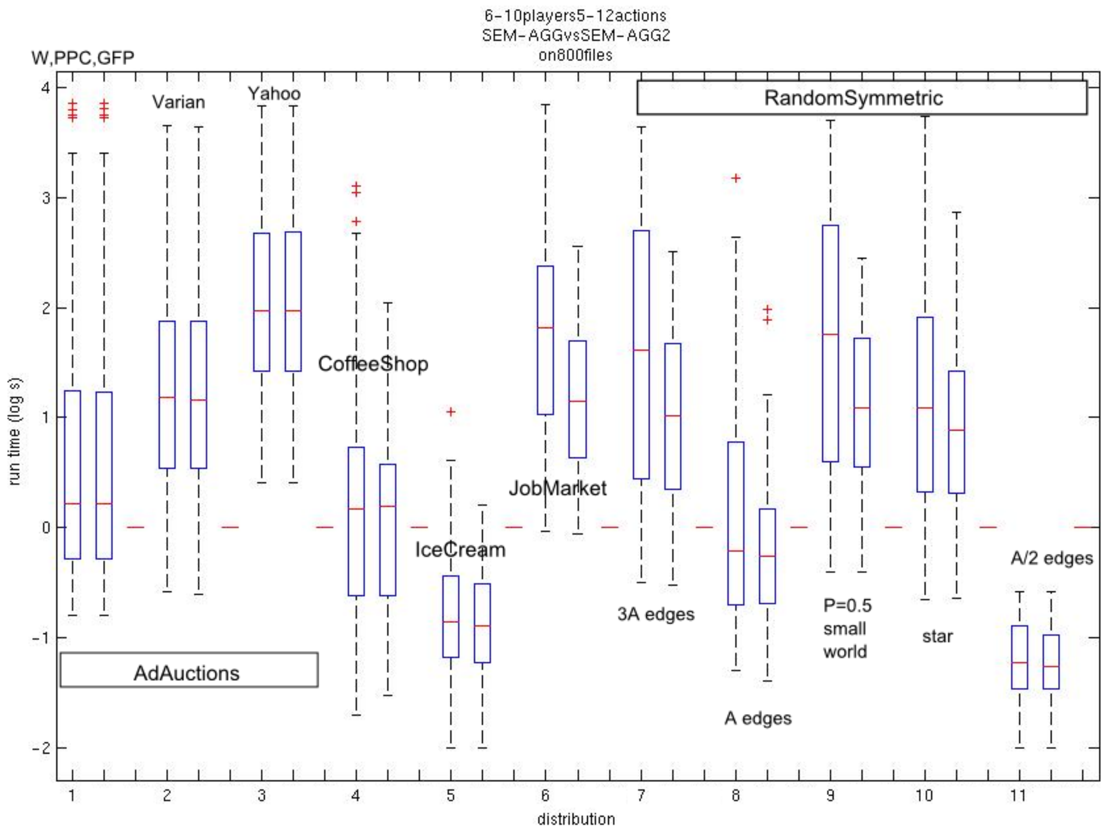

Figure 3: Boxplot comparing basic SEM-AGG (left columns) with SEM-AGG search on reduced support profile space

Figure 3 shows the run times for 800 game instances where both implementations finished under 2 hours. We see that for the AdAuction games, which does not have player symmetry, there are no discernable differences in run time. Small improvements are noticed in games with symmetry.

It appears that more improvements are observed in instances with longer run times. For example a larger difference is seen for JobMarketGame distribution, which has a higher median run time, than the IceCreamGame distribution. This trend is consistent with the intuition that instances that tend to take longer to solve have larger, more complex equilibria, and the benefits of reducing the support profile space becomes more prominent as the support sizes increase.

The cumulative percentages in Figure 4 also suggests modest improvements in run time from eliminating equivalent support profiles. It appears that avoiding equivalent support profiles may be worth implementing if one wants to minimize run time.

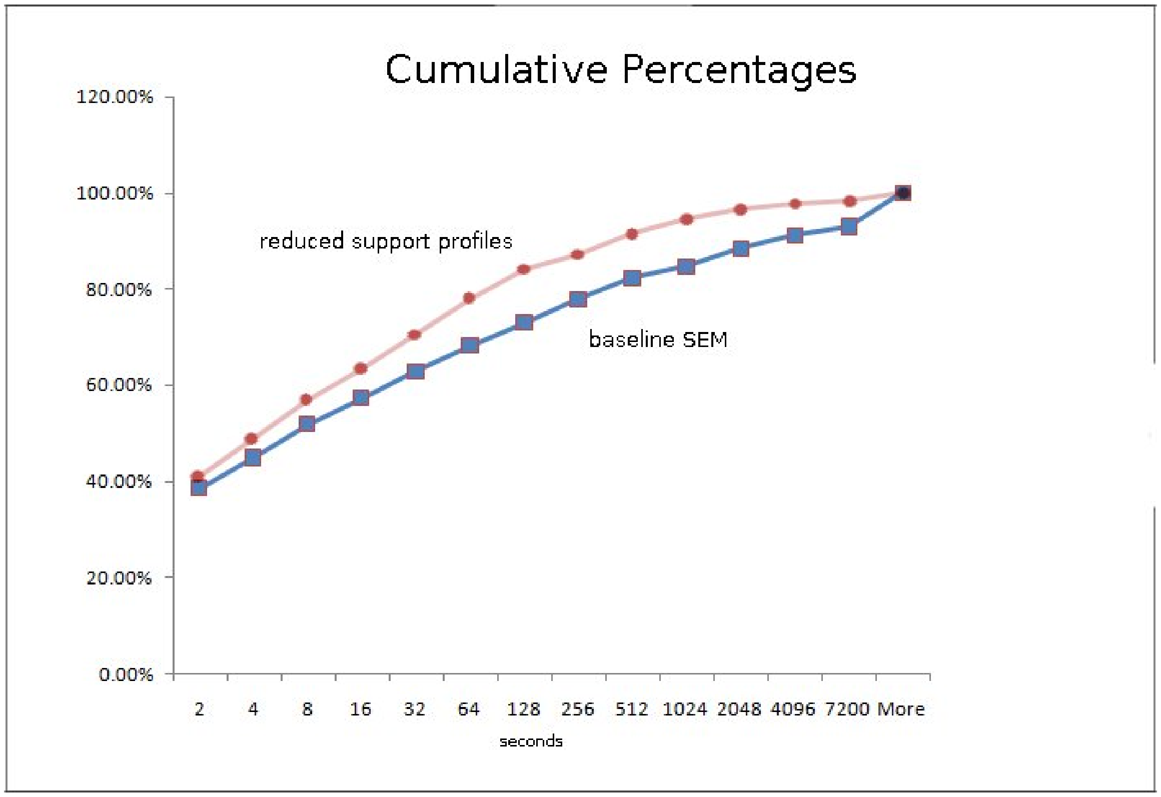

Figure 4: Cumulative distributions. Trials are terminated at 7200 seconds.

5 Limitations and future work

In addition to the basic AGG model with actions as nodes and dependences as vertices, Jiang et al have also introduced, and continue to introduce, constructs such as function nodes into the AGG model [2]. Due to time constraints, we have not made considerations for such constructs for the proposed IRSDS modification. We intend to take function nodes into consideration prior to implementing and testing the proposed IRSDS modification.

As mentioned, the set of games used in our experiments is limited. Therefore a more comprehensive set of distributions may provide more confidence in our claims. For example, another set of game distributions of interest are games that are compact in the Graphical Game format.

The inquiring reader may be wondering what is the performance of the SEM-AGG algorithm when compared to support enumeration on normal form games, and why no experimental data was provided. Although we predict that SEM-AGG would enjoy reliable improvements in run time by relying on the convenience that AGGs provide, currently available experimental data does not support this claim. The reasons for this unexpected behaviour is still under investigation. Possible problems may involve mistakes in the SEM-AGG implementation, mistakes in data analysis, or algorithmic properties of SEM-AGG.

References

[1] N. Bhat and K. Leyton-Brown. Computing nash equilibria of action-graph games. *Uncertainty in Artificial Intelligence (UAI-2004)*, 2004.

[2] A. Jiang, K. Leyton-Brown, and N. Bhat. Action-graph games. *University of British Columbia Technical Report TR-2008-13*, 2008.

[3] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI2001)*, 2001.

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

[5] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a nash equilibrium. *Games and Economic Behavior*, 63(2):642–662, 2008.

[6] D. R. M. Thompson and K. Leyton-Brown. Tractable computational methods for finding nash equilibria of perfect-information position auctions. *Fourth Workshop on Ad Auctions*, 2008.