Local Search Techniques for Computing Equilibria in Two-Player General-Sum Strategic-Form Games

(Extended Abstract)

Sofia Ceppi
DEI, Politecnico di Milano, Milano, Italy
ceppi@elet.polimi.it

Nicola Gatti
DEI, Politecnico di Milano, Milano, Italy
ngatti@elet.polimi.it

Giorgio Patrini
DEI, Politecnico di Milano, Milano, Italy
giorgio.patrini@mail.polimi.it

Marco Rocco
DEI, Politecnico di Milano, Milano, Italy
marco1.rocco@mail.polimi.it

ABSTRACT

The computation of a Nash equilibrium in a game is a challenging problem in artificial intelligence. This is because the computational time of the algorithms provided by the literature is, in the worst case, exponential in the size of the game. To deal with this problem, it is common the resort to concepts of approximate equilibrium. In this paper, we follow a different route, presenting, to the best of our knowledge, the first algorithm based on the combination of support enumeration methods and local search techniques to find an exact Nash equilibrium in two-player general-sum games and, in the case no equilibrium is found within a given deadline, to provide an approximate equilibrium. We design some dimensions for our algorithm and we experimentally evaluate them with games that are unsolvable with the algorithms known in the literature within a reasonable time. Our preliminary results are promising, showing that our techniques can allow one to solve hard games in a short time.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents*

General Terms

Algorithms

Keywords

Game theory (cooperative and non-cooperative)

1. INTRODUCTION

Non-cooperative game theory provides elegant models and solution concepts for capturing settings in which rational agents strategically interact [9]. Technically speaking, a game is a pair: the *mechanism* defines the rules (e.g., number of agents, available actions, outcomes), and the *strategies* define the behavior of the agents. The central solution concept is Nash equilibrium [3]. It prescribes strategies such that no agent can gain more by deviating unilaterally from them. Any game is proved to admit at least a Nash equilibrium, however its computation is a challenging problem also with two agents. In [2] the authors show that computing a Nash equilibrium in an n-player game is PPAD-complete. It is not known whether or not P=PPAD. However, it is generally believed that the two classes are not equivalent and that, in the worst case, computing a Nash equilibrium will take time that is exponential in the size of the game [9]. In this paper we focus on the problem of designing efficient algorithms for solving two-player general-sum strategic-form games with complete information.

Cite as: Local Search Techniques for Computing Equilibria in Two-Player General-Sum Strategic-Form Games (Extended Abstract), S. Ceppi and N. Gatti and G. Patrini and M. Rocco, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

2. STATE OF THE ART

Searching for a Nash equilibrium essentially requires the resolution of a feasibility mathematical programming problem. The literature provides three solving algorithms for two-player general-sum games: Lemke-Howson (LH) [4], Porter-Nudelman-Shoham (PNS) [7], and Sandholm-Gilpin-Conitzer (SGC) [8]. LH provides a linear complementarity mathematical programming formulation and an algorithm based on pivoting techniques [4]. PNS provides an algorithm that enumerates all the agents' supports and for each joint support checks the existence of a Nash equilibrium by solving a linear feasibility problem [7]. SGC provides a mixed integer linear mathematical programming formulation and several methods to improve the computational efficiency [8]. Each of the above three algorithms outperforms the others in some specific settings: PNS outperforms SGC and LH for almost all the games generated by GAMUT [6]; LH outperforms PNS and SGC for games with medium-large support equilibria (this class of games is developed in [8]); SGC outperforms PNS and LH when one searches for an optimum equilibrium. As shown in [1, 8], the instances of the most game classes (with 150 actions per agent) are solved in a negligible time (< 1 s). However, there are some classes (e.g., Covariant, Graphical, and Polymatrix) whose instances are hard to be solved with all the three algorithms. This is due to three reasons: the algorithms search for an equilibrium by enumerating all the possible solutions in a static way, the number of these rises exponentially in the number of agents' actions, and, in the worst case, they must explore the whole solution space.

3. OUR PROPOSAL

We propose, to the best of our knowledge, the first algorithm based on the combination of support enumeration methods and local search techniques [5]. We formulate the problem of finding a Nash equilibrium as a combinatorial optimization problem where the search space is the support space and the function to be minimized is designed such that its global minima correspond to Nash equilibria. Basically, our algorithm works iteratively generating new solutions (according to a topological representation of the support space) and accepting those that improve the value of a given objective function. Since the objective function is generally non-convex and presents multiple local optima, metaheuristics are employed to escape from them and reach a global optimum. We design the following dimensions for our algorithm.

Objective functions. We design four different objective functions $f(\cdot)$s.

Irreducible infeasible set size. For a linear programming problem, an irreducible infeasible set is an infeasible subset of constraints and variable bounds that becomes feasible if any single constraint or variable bound is removed. We define f as the size of the irreducible infeasible set The idea is simple, the larger the IIS the lower the infeasibility measure of the problem.

Inequality constraint violations. In the previous case, we gave the same importance to equality and inequality constraints. Instead, in this case, we force the equality constraints to be satisfied and we measure the violations only of the inequality constraints. The basic idea is that with non-degenerate games the equality constraints constitute a non singular linear equation set that, by definition, admits a unique solution and can be easily solved. We define f as the number of the violated inequality constraints.

Best well-supported ϵ-Nash. Given the agents' joint support, we formulate the problem of computing the best well-supported ϵ-Nash equilibrium as a linear programming problem. We define f as the value of ϵ.

Minimal regret. Given the agents' joint support, we formulate the problem of computing the strategy profile with the minimal agents' regret r as a linear programming problem. We define f as the value of r.

Heuristics. We design some heuristics.

Iterative improvement. Given a solution, neighbors are repeatedly generated until a better solution is not found. Then, the algorithm moves on this last solution. The generation of the neighbors can be accomplished in different ways: *best improvement* (all the neighbors are generated in lexicographic order and the best one, if better than the current solution, is chosen as next solution), *first improvement with lexicographic generation* (the neighbors are generated in lexicographic order and the first generated solution, that is better than the current one, is chosen as next solution), *first improvement with random generation* (the neighbors are generated randomly and the first generated solution, that is better than the current one, is chosen as next solution).

Metropolis. Given a solution s, its neighbors are explored randomly and a solution s' is always accepted if $f(s') < f(s)$ and is accepted with a probability of $\exp\left(\frac{f(s)-f(s')}{t}\right)$, where t is a parameter called temperature, if $f(s) \leq f(s')$.

Metaheuristics. We design some metaheuristics. They are used every time a local minimum is found until a given temporal deadline is not expired.

Random restart. Every time a local minimum is reached, the algorithm starts from a solution generated randomly.

Simulated annealing. It uses Metropolis forcing the temperature to be a function of the iteration number.

Tabu search. We introduce a circular list containing the last visited solutions. Whenever a solution is generated, we check whether or not it is in the list. In the former case we discard it.

4. EXPERIMENTAL EVALUATION

In our experimental analysis, we isolate hard instances produced with GAMUT that are unsolvable with the above three algorithms within a reasonable time (i.e., several hours) and then we apply our algorithm to such instances evaluating the time needed for finding an equilibrium and, in the cases no equilibrium is found within a given deadline, the quality of the best solution found so far. Our preliminary experimental results are promising: with the best configuration of our algorithm, hard game instances are solved in short time with high probability. In particular, the use of regret based objective function allows one to solve games with 50 actions per player within 10 minutes with a probability of ~ 95%.

5. REFERENCES

[1] S. Ceppi, N. Gatti, and N. Basilico. Computing Bayes-Nash equilibria through support enumeration methods in Bayesian two-player strategic-form games. In *IAT*, pages 541–548, Milan, Italy, 2009.

[2] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. In *STOC*, pages 71–78, Seattle, USA, 2006.

[3] D. Fudenberg and J. Tirole. *Game Theory*. The MIT Press, Cambridge, USA, 1991.

[4] C. Lemke and J. Howson. Equilibrium points of bimatrix games. *SIAM J APPL MATH*, 12(2):413–423, 1964.

[5] W. Michiels, E. Aarts, and J. Korst. *Theoretical Aspects of Local Search*. Springer, Berlin, Germay, 2007.

[6] E. Nudelman, J. Wortman, K. Leyton-Brown, and Y. Shoham. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *AAMAS*, pages 880–887, New York, USA, 2004.

[7] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. In *AAAI*, pages 664–669, 2004.

[8] T. Sandholm, A. Gilpin, and V. Conitzer. Mixed-integer programming methods for finding Nash equilibria. In *AAAI*, pages 495–501, Pittsburgh, USA, 2005.

[9] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations*. Cambridge University Press, Cambridge, USA, 2008.