Algorithmic Game Theory*

Paul W. Goldberg

Department of Computer Science
Oxford University, United Kingdom
Paul.Goldberg@cs.ox.ac.uk

Abstract

Game theory studies mathematical models of interactions amongst self-interested entities. A "solution concept" means a description of the outcome of a game, and it is important that it should be defined in such a way that a solution always exists (every game should have an outcome). Nash's famous theorem that mixed-strategy equilibria are guaranteed to exist, resulted in *Nash equilibrium* being the most prominent solution concept in game theory.

As a result, computational challenges of the form "given a game, find a solution", have the property that we are searching for something whose existence is guaranteed (they are *total search problems*). Moreover, these solutions belong to the complexity class NP, since it is usually straightforward to check whether a proposed solution is correct (an incorrect one will admit a profitable deviation by one or more of the players, and this is usually easy to find). However, in versions of the problem that appear to be computationally hard, we cannot apply NP-completeness, due to a result of Megiddo saying that total search problems cannot be NP-complete unless NP is equal to co-NP.

In this tutorial, which is intended for people familiar with NP-completeness, I give an overview of the alternative notions of computational hardness that apply to game-theoretic solution concepts. I discuss the complexity class PPAD (introduced by Papadimitriou) which captures the computational complexity of various classes of games that don't seem to be solvable in polynomial time. I also mention the complexity classes PLS and FIXP, and the kinds of games that they apply to.

Suppose, alternatively, that we have a polynomial-time algorithm that applies to some given class of games. A follow-up question is whether there exist algorithms that find a solution via processes that reflect decentralised selfish behaviour. This is because a solution concept arguably remains unrealistic if it can be efficiently computed, but only using a highly centralised algorithm. In the second half of the tutorial I present some results on learning dynamics for equilibrium computation, and mention recent work on communication complexity and query complexity.

I discuss some research directions and open problems, such as the following. What are the prospects for proving that PPAD is as hard as NP? How about algorithms that find improved *approximate* Nash equilibria? 2-player games are easy to solve in practice, using the Lemke-Howson algorithm, so is there a satisfying mathematical sense in which 2-player games are easy to solve? (For example, a sense in which Lemke-Howson works "most of the time"?)

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, J.4 Economics

Keywords and phrases equilibrium, non-cooperative games, computational complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2015.20

Category Invited Talk

* This work was partially supported by EPSRC under grant EP/K01000X/1.

© Paul W. Goldberg;
licensed under Creative Commons License CC-BY
32nd Symposium on Theoretical Aspects of Computer Science (STACS 2015).
Editors: Ernst W. Mayr and Nicolas Ollinger; pp. 20–20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE