Available online at www.sciencedirect.com
ScienceDirect
Games and Economic Behavior 63 (2008) 621–641
GAMES and Economic Behavior
www.elsevier.com/locate/geb

New complexity results about Nash equilibria ☆

Vincent Conitzer [a,*], Tuomas Sandholm [b]

[a] *Department of Computer Science and Department of Economics, Duke University, Durham, NC 27708, USA*
[b] *Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

Received 10 January 2005

Available online 1 May 2008

Abstract

We provide a single reduction that demonstrates that in normal-form games: (1) it is $\mathcal{NP}$-complete to determine whether Nash equilibria with certain natural properties exist (these results are similar to those obtained by Gilboa and Zemel [Gilboa, I., Zemel, E., 1989. Nash and correlated equilibria: Some complexity considerations. Games Econ. Behav. 1, 80–93]), (2) more significantly, the problems of maximizing certain properties of a Nash equilibrium are inapproximable (unless $\mathcal{P} = \mathcal{NP}$), and (3) it is $\#\mathcal{P}$-hard to count the Nash equilibria. We also show that determining whether a pure-strategy Bayes–Nash equilibrium exists in a Bayesian game is $\mathcal{NP}$-complete, and that determining whether a pure-strategy Nash equilibrium exists in a Markov (stochastic) game is $\mathcal{PSPACE}$-hard even if the game is unobserved (and that this remains $\mathcal{NP}$-hard if the game has finite length). All of our hardness results hold even if there are only two players and the game is symmetric.
© 2008 Elsevier Inc. All rights reserved.

JEL classification: C63; C70; C72; C73

☆ This work appeared as an oral presentation at the Second World Congress of the Game Theory Society (GAMES-04), and a short, early version was also presented at the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03). The material in this paper is based upon work supported by the National Science Foundation under grants IIS-0234694, IIS-0427858, IIS-0234695, and IIS-0121678, as well as two Sloan Fellowships and an IBM PhD Fellowship.

* Corresponding author at: Duke University, Department of Computer Science and Department of Economics, Box 90129, LRSC, Durham, NC 27708, USA.
E-mail addresses: conitzer@cs.duke.edu (V. Conitzer), sandholm@cs.cmu.edu (T. Sandholm).

0899-8256/$ – see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.geb.2008.02.015

1. Introduction

Game theory provides a normative framework for analyzing strategic interactions. However, in order for anyone to play according to the solutions that it prescribes, these solutions must be *computed*. There are many different ways in which this can happen: a player can consciously solve the game (possibly with the help of a computer[1]); some players can perhaps eyeball the game and find the solution by intuition, even without being aware of the general solution concept; and in some cases, the players can converge to the solution by following simple learning rules. In each case, some computational machinery (respectively, one player's conscious brain, a computer, one player's subconscious brain, or the system consisting of all players together) arrives at the solution using some procedure, or *algorithm*.

Some of the most basic computational problems in game theory concern the computation of Nash equilibria of a finite normal-form game. An example problem is to compute *one* Nash equilibrium—any equilibrium will do. What are good algorithms for solving such a problem? Certainly, we want the algorithm to always return a correct solution. Moreover, we are interested in how *fast* the algorithm returns a solution. Generally, as the size of the game (more generally, the *problem instance*) increases, so does the running time of the algorithm. Whether the algorithm is practical for solving larger instances depends on how rapidly its running time increases. An algorithm is generally considered *efficient* if its running time is at most a polynomial function of the size of the instance (game). There are certainly other properties that one may want the algorithm to have—for example, one may be interested in learning algorithms that are simple enough for people to use—but the algorithm should *at least* be correct and computationally efficient.

The same computational problem may admit both efficient and inefficient algorithms. The theory of *computational complexity* aims to analyze the inherent complexity of the problem itself: how fast is the *fastest* (correct) algorithm for a given problem? $\mathcal{P}$ is the class of problems that admit at least one efficient (polynomial-time) algorithm.[2] While many problems have been proved to be in $\mathcal{P}$ (generally by explicitly giving an algorithm and proving a bound on its running time), it is extremely rare that someone proves that a problem is *not* in $\mathcal{P}$. Instead, to show that a problem is hard, computer scientists generally prove results of the form: "If this problem can be solved efficiently, then so can every member of the class $\mathcal{X}$ of problems." This is usually shown using a *reduction* from one problem to another (we will give more detail on reductions in Section 2). If this has been proven, the problem is said to be $\mathcal{X}$*-hard* (and $\mathcal{X}$*-complete* if, additionally, the problem has also been shown to lie in $\mathcal{X}$). The strength of such a *hardness result* depends on the class $\mathcal{X}$ used. Usually, the class $\mathcal{NP}$ is used (we will describe $\mathcal{NP}$ in more detail in Section 2), and most problems of interest turn out to be either in $\mathcal{P}$ or $\mathcal{NP}$-hard. $\mathcal{NP}$ contains $\mathcal{P}$, and it is generally considered unlikely that $\mathcal{P} = \mathcal{NP}$. Exhibiting a polynomial-time algorithm for an $\mathcal{NP}$-hard problem (thereby showing $\mathcal{P} = \mathcal{NP}$) would constitute a truly major upset: among other things, it would (at least in a theoretical sense, and possibly in a practical

[1] The player might also *be* a computer, for example, a poker-playing computer program. Indeed, at least for some variants of poker, the top computer programs are based around computing a game-theoretic solution (usually, a minimax strategy).

[2] To define $\mathcal{P}$ formally (which we will not do here), one must also formally define a model of computation. Fortunately, the class of polynomial-time solvable problems is quite robust to changes in the model of computation. Nevertheless, it is in principle possible that humans have a more powerful computational architecture, and hence that they can solve problems outside $\mathcal{P}$ efficiently.

دریافت فوری ← متن کامل مقاله

ISIArticles
مرجع مقالات تخصصی ایران

✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
✓ امکان دانلود نسخه ترجمه شده مقالات
✓ پذیرش سفارش ترجمه تخصصی
✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
✓ دانلود فوری مقاله پس از پرداخت آنلاین
✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات