Computing equilibria in pure strategies

Uriel Feige

April 16, 2024

1 Classical complexity classes

We shall assume familiarity with basic computational complexity theory, and standard complexity classes. Recall that $P \subset NP \subset PSPACE \subset EXPTIME$. Of these, the only inclusion that is known to be strict is that of P in EXPTIME. Namely, there are problems in EXPTIME that provably do not have polynomial time algorithms.

Recall also the notions of *hardness* and *completeness* with respect to a complexity class. Ignoring some of the finer distinctions of complexity theory, we shall say that a problem (a.k.a. *language*) π (such as 3-colorability) is *hard* for complexity class C (such as NP) if for any problem ψ in C (such as satisfiability), every instance of ψ can be reduced in polynomial time to an instance of π. A problem π is *complete* for C if $\pi \in C$ and π is hard for C.

It follows that every problem complete for EXPTIME has no polynomial time algorithm. It is not known whether the same holds for NP-complete problems (the famous P versus NP question).

2 Dominant strategies

For games in normal form, dominant strategies (when they exist) can be found in polynomial time, by exhaustive search over all strategies. The question of computing dominant strategies becomes more interesting for games that have succinct representations. Here we shall survey some know results concerning two player games, and specifically games in which the payoffs are *win/lose* (and the complement for the other player). In some games, *ties* will be allowed (e.g., if the game does not end).

There is a well developed area of *combinatorial game theory* that we shall not address in this course. See more details in [Conway01, BCG01], for example.

Tasks in program verification are sometimes presented as games. These games often are played on a finite directed graph, and may continue for infinitely many moves. (We use here the word *move* in a sense that is sometimes referred to as *turn*.) Winning conditions for games that continue for infinitely many moves often relate to the set of nodes that are visited infinitely often. See [Thomas02] for example.

Games play an important role in computational complexity as well, because several important complexity classes have complete problems that are games (and hence these games capture the essence of these complexity classes). A well known result in this respect relates to the notion of *alternation* [CKS81]. Alternation may be thought of as a game of perfect information between two players played on a board of size n. An important result in [CKS81] is that alternating PTIME is PSPACE and alternating PSPACE is EXPTIME. This combines two sets of results. One is that computing winning strategies for any such game can be done in polynomial space if the game is limited to a polynomial number of moves, and in exponential time if the game can continue for exponentially many moves. (The games cannot continue for longer without cycling.) The other is that there are such games where PSPACE or EXPTIME (respectively) is necessary. An example of a PSPACE-complete game is *generalized geography*. An example of an EXPTIME-complete game is *generalized chess*. This last result helps explain the apparent intractability of playing chess perfectly even on an 8 by 8 board.

The exponential time algorithm that computes optimal strategies involves constructing an exponential size graph of all possible positions of the game (where the position may include also the move number, bounded by the total number of possible positions, so as to avoid cycling), and labelling positions as win/lose/tie by backward induction.

We mention here an open question related to games of perfect information.

Parity games. The game graph is a directed bipartite graph. Vertices are numbered from 1 to n. A token is placed on a starting vertex. Players alternate in moving the token along an outgoing edge from its current location. If a player cannot move, he looses. If the game continues indefinitely, then the winner is the first player if the highest numbered vertex that is visited infinitely often has an even number, and the other player otherwise. It is known that determining which player has a winning strategy is in $NP \bigcap coNP$, but not known whether it is in P. (See entry in Wikepedia, for example.)

3 Pure Nash

For games in normal form, a pure Nash equilibrium (if it exists) can be computed in polynomial time by trying out all strategy profiles, and for each of them checking whether any player has an incentive to deviate. Hence also here, the main interest is in computing a pure Nash equilibrium for games given under some succinct representation.

A well known example is the *stable matching* (a.k.a. stable marriage) problem. There are n men and n women. Every man has a preference ordering over the women, and every woman has a preference order over the men. The goal is to find a perfect matching (each man matched to exactly one woman) that is *stable* in the following sense: their is no pair of man and woman that are not matched to each other, but prefer each other over the partners matched to them. Gale and Shapley [GaleShapley62] showed that a stable matching always exists, and provided a polynomial time algorithm for finding a stable matching.

First, let us present a multiplayer game that captures the notion of a stable matching. The players are the men. Each man has n possible strategies, where a strategy of a man is to propose to a woman. The payoff of a man is computed as follows. If he is the highest ranked man among the men who proposed to the same woman that he proposed to, the woman accepts him and then he gets a numerical payoff equal to the rank of the woman in his own preference list (n for highest ranked woman, 1 for lowest ranked woman). Else his payoff is 0.

We show that stable matching correspond exactly to the Nash equilibria of the above game. Every stable matching M corresponds to a collection of pure strategies in which each man proposes to the woman that is matched to him under M. No man has incentive to unilaterally deviate from this profile of strategies, because by stability of M whatever other higher ranked woman he will propose to prefers here current partner, and hence his payoff will drop to 0. Similar arguments show that every Nash equilibrium corresponds to a stable matching. (Remark: we are dealing here only with pure Nash equilibrium. Mixed Nash equilibria are not of interest for us here, because the payoff functions do not correspond to utility functions.)

The algorithm of [GaleShapley62] for finding a stable matching proceeds in rounds. At the beginning of every round, some of the men are *engaged* and some are free. Initially, all men are free. The following describes a single round.

1. Every free man proposes to the woman ranked highest on his preference list, and becomes engaged.

2. Every woman rejects all her offers except the one made by the man ranked highest on her list.

3. Any man whose offer got rejected removes the rejecting women from his preference list, and becomes free again.

The algorithm ends when all men become engaged, at which point every man is matched to the woman whom he is engaged to.

Theorem 3.1 *The algorithm described above produces a stable matching.*

Proof: We say that a woman becomes engaged at the first round in which she receives a proposal. Observe that once a woman becomes engaged, she remains engaged throughout the run of the algorithm, though the man to which the woman is engaged may change to a man higher on her preference list. If at some point, all men are engaged, the algorithm ends in a matching. As long as some man is free, the algorithm continues. Every woman is somewhere on every man's preference list, hence if any man ever exhausts his preference list, then all women must be engaged.

As it cannot be that two women are engaged to the same man, all men are engaged as well. This implies that the algorithm must end with a matching.

The matching output by the algorithm is stable because every man already tried proposing to all women higher on his preference list than the woman to which he is matched, and every one of them already had a proposal from a more preferable man at the time, and hence also at the end of the algorithm. □

It is known and not hard to prove that the stable matching produced by the above algorithm is optimal from the point of view of the men. For every man, in every other stable matching, the woman he is matched to is not ranked higher in his preference list than the woman he is matched to under the above algorithm. For women, the opposite holds – there is no worse stable matching.

This stable matching algorithm is used in assigning graduating medical students to hospitals in the United States. Hospitals make offers to the students, and the students reject all but their best offer. Hence this system favors the hospitals' preferences over those of the students.

We remark that given that the algorithm is publicly known, a new game arises. In this game, every man and every woman supplies a preference list, the outcome of the game is the matching produced by the stable matching algorithm, and the payoff for a player is the rank (in the player's original list, which need not necessarily be the supplied list) of the partner assigned to the player. An interesting question is whether the players have incentives to play *truthfully* in this game. Namely, is it always to the benefit of a player to report his or her true preference list, or may the player win a better partner (from the player's point of view) by reporting a different preference list? It can be shown that the 'men proposing" version of the Gale-Shapley algorithm is truthful for men, but not truthful for women. One of the goals of the area of *mechanism design* is to design games in which players have incentives to reveal their true preferences. Note that if players do not report their true preferences for the stable matching algorithm, then the matching produced might not be stable with respect to their true preferences.

References

[BCG01] E. Berlekamp, J. H. Conway, R. Guy. Winning Ways for your Mathematical Plays (2nd ed.). A K Peters Ltd. 2001.

[CKS81] Ashok K. Chandra, Dexter Kozen, Larry J. Stockmeyer: Alternation. J. ACM 28(1): 114-133 (1981)

[Conway01] John Conway. On Numbers And Games (2nd ed.). A K Peters Ltd. 2001.

[FraeLich81] Aviezri Fraenkel and D. Lichtenstein (1981). "Computing a perfect strategy for n by n chess requires time exponential in n". J. Comb. Th. A (31): 199–214.

[GaleShapley62] D. Gale and L. S. Shapley: "College Admissions and the Stability of Marriage", American Mathematical Monthly 69, 9-14, 1962.

[Papadimitriou] Christos Papadimitriou. Computational Complexity. Addison-Wesley. 1994.

[Thomas02] Wolfgang Thomas. Infinite Games and Verification (Extended Abstract of a Tutorial). *CAV 2002: 58–64.*