Algorithms for Computing Solution Concepts in Game Theory

Uriel Feige

December 21, 2008

Preface

These are notes for the first five lectures of the course on Algorithmic Game Theory, given (starting November 2008) in the Weizmann Institute jointly by Uriel Feige, Robert Krauthgamer and Moni Naor. The lecture notes are not intended to provide a comprehensive view of solution concepts in game theory, but rather discuss some of the algorithmic aspects involved. Hence some of the definitions will not be given here in their full generality. More information can be found in the first three chapters of [NRTV07] and references therein.

Each lecture in the course was roughly two hours, with a break in the middle. The experience of the author when teaching this material was as follows.

The first lecture was intended to cover Section 1, but this turned out to be too optimistic. The notion of a mixed Nash was only touched upon briefly at the end of the lecture, and there was no time for the notion of a correlated equilibrium (which was deferred to lecture 5).

The second lecture covered Section 2, except that the open questions mentioned in Section 2.2 were not discussed in class for lack of time.

The third lecture covered sections 3.2 (an introduction to linear programming) and 3.3. Initially, I was planning to precede this by discussing known results on two-player win-lose games of partial information (see Section 3.1) but decided to drop this subject due to lack of time. The associated homework assignment partially touches upon this subject.

The fourth lecture discussed mixed Nash equilibria. Towards the end of the lecture I started describing the Lemke-Howson algorithm, but did not finish in time.

The fifth lecture started with the Lemke-Howson algorithm, which took roughly an hour to describe (longer than I expected). Then the class PPAD was discussed (including showing the proof of Sperner's Lemma in two dimensions, not appearing in the lecture notes), finishing with correlated equilibria (the end of Section 1), and a short mention that it can be computed using linear programming. Theorem 3.7 was omitted due to lack of time.

I would have liked to have an additional lecture covering parts of Chapter 4 in [NRTV07] (iterative algorithms for regret minimization), which among other things explains how to (approximately) solve two-player zero-sum games in certain situations in which the game is not in standard form, and provides an alternative proof to the minimax theorem. However, I decided to drop this subject since the other subjects already took up the five lectures devoted to this part of the course.

Homework assignments as given in the course are also included. They are intended to be an integral part of the course. The first question in assignment 2 turned out to be too difficult, especially as it was given with an incorrect hint. The first question in assignment 5 could have been given after lecture 4.

The notes are made accessible on the web for the use of students in the course, and for non-commercial use of others who may be interested. To help make the notes a more reliable source of information for future courses, please send comments to uriel.feige@weizmann.ac.il.

1 Introduction

1.1 What is a game?

A game has *players*. We shall assume here that the number of players in a game is at least two and finite. (For simplicity, we shall assume players of male gender, though the gender of the players is irrelevant, and the players may also be genderless software programs.)

Every player i has a set S_i of *strategies* available to him. We shall assume here that the sets S_i are finite, though game theory also addresses games with infinite strategy sets. In a game, every player selects one strategy from his respective set of strategies.

The *outcome* of the game is the profile of strategies selected by the players, one strategy for each player. Hence it is a vector of strategies.

Every player is assumed to have his own *preference relation* over outcomes. This preference relation induces a partial order over outcomes. For simplicity, let us assume here that the partial order is given in form a numerical value for each outcome, and the player prefers those outcomes with higher numerical value over those with lower numerical value. These values are often referred to as *payoffs* for the players. It is often the case that the value of the payoff is meant to have quantitative significance beyond the preference of order over outcomes. For example, a payoff 2 would be considered twice as good as a payoff of 1. In this case, the payoff function would typically be called a *utility* function for the player.

A game is represented in *standard* form (a.k.a. strategic form) if it explicitly list for each player and every outcome the value of the outcome for that player. For two player games, a game in standard form is often depicted as a pair of payoff matrices, one for the row player and the other for the column player, or even as a single matrix with both payoffs written in each entry.

Often, standard form representation of games is prohibitively large. There are many other more succinct representations of games. Many games (such as the game of chess) are represented implicitly. Two common forms of succinct representations for general classes of games are *extensive form* (typically, a game tree for two person games), and *graphical games* (a graphical representation of multiplayer games in which the payoff of a player is affected only by actions of players in his immediate neighborhood).

1.2 Solution concepts

A description of a game does not say how the players actually choose their strategies. Game theory tries to answer this question. There are two different aspects to this question.

- *Descriptive.* Given a game, how will people actually play it? What strategies will they choose? These questions involve considerations from social science and psychology, and their study may well require experimentation.
- *Prescriptive.* Given a game, how should players play it? Recommend strategies for the players. This is the more theoretical part of game theory, and here mathematics and computer science have a more prominent role.

Though the descriptive and prescriptive aspects are related (e.g., if a strategy is recommended, will the players actually play it?), the emphasize here will be on the prescriptive aspect of game theory.

We will try to associate *solution concepts* with a game. The solution concept will offer to players a recommendation of which strategy to play. The recommended strategy will satisfy some optimality conditions (that depends on the solution concept).

Before presenting the solution concepts that will be discussed here, let us explicitly list some of the assumptions that we shall make here.

1. *The game is given.* A game is sometimes an abstraction of a real life situation. Reaching the right abstraction (who the players are, what actions are available to them, what are their preferences) might be a very difficult task and will not be addressed here. We assume that the game is given.

2. *Self awareness.* The player is aware of those aspects of the game that are under his control. He knows which strategies are available to him, and his own preference relation over outcomes. (Indirectly, this requires awareness of strategies available to the other players, as they define the possible outcomes.)

3. *Full information.* For games in standard form, this is essentially the same as self awareness. However, for games given in some succinct representation, full information is different from self awareness. For example, for games in extensive form, in means that after every move the player knows exactly at what node of the game tree the game is. Chess is an example of a game of full information. Poker is an example of a game of partial information.

4. *No computational restrictions.* When suggesting a solution concept, we shall ignore the question of whether finding a recommended strategy under this concept is computationally tractable. Of course, in many games (chess being one example) the issue of computational limitations (a.k.a. *bounded rationality*) is an important aspect of the game. We remark that even though the solution concepts themselves will not a-priori be required to be computationally efficient, we shall eventually be interested in their computational complexity.

We now present some of the solution concepts that will be discussed more thoroughly later. The underlying assumption is that players are *rational*. Here the word rational is meant to convey that a player attempts to maximize his own payoff. It is worth pointing out that a player may appear to behave irrationally from this respect. This can often be attributed to the failure of one or more of the assumptions listed above.

1.3 Solutions in pure strategies

Dominant strategies. A strategy s is *dominant* for player i if regardless of the strategies played by other players, the payoff of player i is strictly maximized by playing s_i.

Formally, for every set of strategies s_{-i} for all players but i and every strategy $s' \neq s$,

$$u_i(s, s_{-i}) \geq u_i(s', s_{-i})$$

Dominant strategies do not always exist (e.g., for a payoff matrix that is the identity matrix). However, when they do exist, they are a very favorable solution concept. For games in standard form, they can be computed efficiently.

An interesting example is the game *prisoner's dilemma*, which has the following game matrix.

```
            Cooperate     Defect
          |------------|------------|
Cooperate |   -2; -2   |   -9; -1   |
          |------------|------------|
Defect    |   -1; -9   |   -8; -8   |
          |------------|------------|
```

Both players have dominant strategies, but the outcome of playing them is inferior for both players than the outcome if alternative strategies are played. The source of the problem is that a

player can slightly improve his own payoff at the cost of other players loosing a lot. If each player selfishly maximizes his payoff, everyone looses.

Examples such as prisoner's dilemma point to a shortcoming of the concept of rational behavior - it might lead to undesirable outcomes.

It is worth mentioning in this context an area of game theory that will be addressed in other parts of the course, that of *mechanism design*. At a high level, the purpose of this area is to set up games in such a way that rational behavior will always lead to desirable outcomes, avoiding situations such as the prisoner's dilemma.

A well known example for mechanism design is that of a Vickery auction. Consider an auctioneer who has one item for sale, and k bidders (players), where bidder i has value u_i for the item (known only to bidder i). The process of the auction is a sealed bid auction, in which first all players submit their bids in sealed envelopes, and then the envelopes are opened and the highest bidder wins. In a *first price auction*, the winner pays his bid. In general, there are no dominant strategies in first price auctions, and the bids of the bidders depend on their beliefs regarding what other bidders will bid. In a *second price (Vickery) auction*, the winner pays the second highest bid. This has the desirable outcome that players have dominant strategies - to bid their true value. Moreover, in this case the outcome is that the item is allocated to the player who desires it most, which promotes economic efficiency. Hence the second price auction is an example of a game that is designed in such a way that its solution under a standard solution concept optimizes some economic goal: maximizing total economic welfare. (Note that here money paid by the bidder to the seller is assumed to have no effect on the total economic welfare, because the total sum of money remains constant.)

Nash equilibrium. In this solution concept, one recommends strategies to all players in the game, with the property that given that all other players stick to their recommendation, the strategy recommended to a player is strictly better than any other strategy.

Formally, a Nash equilibrium in a k player game is a vector $\bar{s} = (s_1, \ldots, s_k)$ such that for every player i and any strategy $s_i' \neq s_i$

$$u_i(s_i, \bar{s}_{-i}) > u_i(s_i', \bar{s}_{-i})$$

For weak Nash equilibrium the inequality need not be strict.

An example of a two player game with a Nash equilibrium is the *battle of sexes*, with male and female players who want to go some movie together, but have different tastes in movies.

	Action	Romantic
Action	5; 4	2; 2
Romantic	1; 1	4; 5

There are no dominant strategies, but two Nash equilibria: the top left corner and the bottom right corner. Each player prefers a different Nash equilibrium.

A famous multi-player example is the stable marriage problem. See Section 2.3.

An example of a game that does not have a Nash Equilibrium is that of *matching pennies*.

1; 0	0; 1
0; 1	1; 0

An issue that comes up with Nash equilibrium and not in dominant strategies is that games may have multiple Nash equilibria. Moreover, different players may prefer different Nash equilibria. This might make Nash equilibria unstable in practice. (A player may deviate from the recommended strategy, suffering some loss, but also inflicting loss to others, in the hope that other players deviate as well, and that a new Nash equilibrium that is more favorable to the player is reached. A strike by a worker's union may be explained as an attempt by the workers to switch to a different equilibrium point between the workers and the employers.)

Subgame optimality. This notion addresses to some extent the issue of multiple solutions of a game under a given solution concept. It applies to games that take multiple rounds. As players make moves and the game progresses, the portion of the game that remains is called a *subgame*. It is required that the strategies of the players be a solution to these subgames as well.

There are two different motivations for this notion, that we illustrate by examples.

Chess. There is a difference between having a strategy that plays optimally only from the initial position, and a strategy that plays optimally from any position. For example, assume that it is true that white has a winning strategy in chess. Then the strategy of playing arbitrarily is optimal for black (in the sense that no other strategy guarantees a higher payoff), but not subgame optimal (it does not take advantage of situations in which white has previously blundered.

Ultimatum game. This example illustrates well several aspects of solution concepts. It is a game for splitting 10 dollars among two players. The column player offers how to split the money, and the row player may either accept the split, or reject it, in which case neither player gets anything. For simplicity of the presentation, assume that the first player is allowed to suggest only one of the following three options (1,9),(5,5),(9,1). In extensive form, the game tree then has six leaves, whereas in standard form it has three columns and eight rows. The following matrix illustrates the payoffs for the row player. The payoffs for the column player are 0 or $10 - x$, depending on whether the payoff x for the row player is 0 or more.

0	0	0
0	0	9
0	5	0
0	5	9
1	0	0
1	0	9
1	5	0
1	5	9

The row player has exactly one dominant strategy - to always accept the offer (the last row). The column player does not have any dominant strategy.

The game has several Nash equilibria. For example, one of them is that the row player accepts only (5,5) splits, and the column player offers a (5,5) split. Clearly, the row player prefers this Nash equilibrium to playing his dominant strategy (which leads to the Nash equilibrium of the column player offering a (1,9) split). Hence the row player may appreciate a possibility to manoeuver

the game towards one of those Nash equilibria that is better from his perspective than the Nash equilibrium that involves him playing his dominant strategy. However, the dynamics of the game do not allow this. The column player plays first, and then can play no more. If the column player plays (1,9), this results in a subgame in which the row player has only two possible strategies, either to accept or reject. The only subgame optimal decision is to accept. Hence the only *subgame perfect equilibrium* is the one in which the row player always accepts. In the case of the ultimatum game, the notion of subgame perfect equilibrium allows us to select one out of the many Nash equilibria.

It turns out that when the ultimatum game is played in real life (experimental economists have actually experimented with this game), the row players often do not play their dominant strategy. Likewise, the column players do not always offer the (1,9) split. An explanation for this is that in real life scenarios, the payoff matrix does not really represent the true payoffs for the players. Besides the monitory payoffs represented in the payoff matrix, there are other forms payoffs (feeling of pride, feeling of fairness, feeling of creating a reputation) that if properly represented in the description of the game would explain this "irrational" sort of behavior.

1.4 Mixed strategies

In certain games, it is desirable for one of the players to play in a nondeterministic way. Matching pennies is one such example. Given its zero sum game payoff functions, it is clear that if a player's move can be predicted by the other player, he will lose. Another example would be a game of partial information such as poker. If a player plays deterministically (his moves are dictated only by the cards that he holds and previous moves that he has seen), then other players may be able to infer (something about) his cards, and use this information to improve their chance of winning (and hence the player suffers, the game being a zero sum game). Experienced poker players base their play not only on information directly available from the play of the hands, but also on other factors, and hence from the point of view of game theory (that fails to completely model all factors involved), their play is nondeterministic.

A way game theory models the issue of playing in a nondeterministic way is through the concept of a *mixed strategy*, which is a probability distribution over strategies. For example, one mixed strategy for matching pennies is to play each option with probability 1/2. The point is that a-priori, other players may be aware of the probability distribution that a player is using in his mixed strategy (it might be announced, or inferred), but they do not know the actual strategy that is selected until after they select their own strategies. (There are obvious modifications to this last statement when one deals with multi-round games.)

It turns out that the notion of mixed strategies opens up the possibility for more solution concepts. To realize this potential, one assumes that payoff functions are actually utility functions (numerical values are a linear scale on which preferences have exact values), and moreover that player's are concerned only with expected payoff (and not the distribution of payoffs). For example, getting a payoff of either 3 or 5, each with probability 1/2, is assumed to be equivalent to getting a payoff of 4 with probability 1. This assumption is often referred to as the players being *risk neutral* (with other options being *risk seeking* or *risk averse*). Risk neutrality is a natural assumption in certain situations. For example, if we assume that a player is involved in many independent games throughout his lifetime each involving a relatively small payoff, and the payoffs from these games add up, then the law of large numbers shows that the total payoff converges to the sum of expectations, and hence the function to optimize per game is indeed the expected payoff.

Another situation in which maximizing the expected payoff is natural (regardless of risk neutrality) is if a game has only two possible payoffs for a player (say, either *win* or *lose*). In this case, maximizing the expected payoff is equivalent to maximizing the probability of winning.

Mixed strategies are sometimes criticized as not being realistic (though they become more realistic when the players are computer programs). The argument is that human players do not really choose randomly among strategies. This relates more to the descriptive aspects of game theory than the prescriptive aspects, and hence is not of major concern to us. However, let us point out that this

issue is discussed extensively in game theory literature, and various justifications are offered for mixed strategy. An interesting one is to consider a two player game in which each player is really a "super-player" composed of a population of individuals. Each individual plays a pure strategy. Every individual has random encounters with individuals of the other population. Hence here the randomness is not in the choice of strategies for each individual, but in the choice of encounters, which effectively corresponds to the super-player corresponding to the other population having a mixed strategy. In a Nash equilibrium (over the populations), every individual is playing optimally (in expectation) against the other population. One may say that its strategy is "fitted" to the environment (the environment for an individual is the other population). If the distribution of strategies over populations is not at a Nash equilibrium, then there may be some strategy not currently used by any individual, which is better fitted to the environment. Such a strategy may then "invade" the space of strategies, and be adopted by many individuals ("survival of the fittest"), changing the mixed strategy of the respective super-player.

Two player zero sum games and the minimax theorem.

A solution concept for two player zero sum games is offered by the minimax player. It assumes risk neutral players that have the conservative goal of securing at least some minimum payoff. For the maximizing player (player 1), this amounts to finding the highest possible value t^+ and a mixed strategy s_1 such that $\min_{s_2} E[u(s_1, s_2)] \geq t^+$. Here u is the payoff function for the maximizing player, and $-u$ is the payoff function for the minimizing player. The strategy s_2 in the above expression ranges over pure strategies. The value t^+ is a max-min value $\max_{s_1} \min_{s_2} E[u(s_1, s_2)]$.

Likewise, the minimizing player seeks a smallest possible value t^- and a strategy s_2 satisfying $\max_{s_1} E[u(s_1, s_2)] \leq t^-$. The value t^- is a min-max value $\min_{s_2} \max_{s_1} E[u(s_m, s_2)]$, where s_2 ranges over mixed strategies, and s_1 ranges over pure strategies.

The famous minimax theory of Von-Neumann says that $t^+ = t^-$. Namely, for every finite two player game, for mixed strategies the following equality holds:

$$\max_{s_1} \min_{s_2} E[u(s_1, s_2)] = \min_{s_2} \max_{s_1} E[u(s_m, s_2)]$$

The expected payoff at this mutually optimal point is referred to as the *value* of the game. It is essential that strategies are allowed to be mixed in the minimax theorem, as the game of matching pennies illustrates.

Observe that the minimax theorem implies that a player may announce his mixed strategy upfront, and still get the expected payoff guaranteed by the minimax theorem.

A modern proof of the minimax theorem is based on linear programming duality (we will review this proof later), and implies also a polynomial time algorithm for computing the minimax value. von-Neumann's original proof (from 1928) predated the concept of linear programming duality, and applies also to some classes of games with infinite strategy space.

For matching pennies, the value of the game is 0. The unique optimal mixed strategies for the players are to play the two options with equal probability. Note however that in this case, if one of the players plays his optimal mix strategy, then the other player can play arbitrarily without changing the value of the game. In fact, this is a rather typical situations in many games: when one player chooses an optimal mixed strategy, the other player has a choice of several pure strategies, each of which is optimal (in expectation).

Mixed Nash. A mixed Nash equilibrium (defined and proven to exist by John Nash) is a profile of mixed strategies, one mixed strategy for each player. If has the property that given that the other players follow this profile, no player has an incentive for deviating from his own mixed strategy. That is, every strategy in the support of his mixed strategy is a best response to the mixed strategies of the other players (gives maximum expected payoff among all pure strategies).

Let $\bar{s}$ denote a profile of mixed strategies, s_i denotes the mixed strategy that it associates with player i, and $\bar{S}_{-i}$ denotes the mixed strategies that it associates with the other players. For $\bar{s}$ to be a mixed Nash equilibrium it is required that for every player i and for every pure strategy s'_i for player i,

$$E[u_i(s_i, \bar{s}_{-i})] \geq E[u_i(s'_i, \bar{s}_{-i})]$$

Mixed Nash equilibria exist for every finite game, with any number of players. Nash proved this as a consequence of certain nonconstructive fixed point theorems. Later we shall present an algorithmic proof for the case of two players (though the algorithm is not a polynomial time algorithm). The question of whether there exists a polynomial time algorithm for computing a mixed Nash equilibrium will be discussed as well.

It is common to refer to mixed Nash equilibrium as Nash equilibrium and to what we previously referred to as Nash equilibrium as pure Nash. We shall follow this terminology from now on.

Correlated equilibrium. A mixed Nash is a product distribution over profiles. To realize a mixed Nash, it suffices that each player has a private source of randomness. In contrast, a correlated equilibrium will be an arbitrary distribution over strategy profiles. To realize it, some coordination mechanism involving a common source of randomness is needed. We shall not discuss here in detail what may serve as a coordination mechanism, but rather mention one common example - that of the traffic light. In the absence of any prior agreed upon traffic rules, the game played by two drivers who arrive from perpendicular directions to a traffic junction at the same time resembles the game of *chicken*.

	Stop	Go
Stop	0; 0	0; 2
Go	2; 0	-9; -9

If both players observe a common signal (the traffic light), they can use it to reach a correlated equilibrium (if you see green, go, and if you see red, stop). Once the correlated equilibrium strategy is announced, then whenever two drivers arrive at a junction with a traffic light, it is in the interest of the players to follow its recommendation (if they trust that the other player also follows it).

To be a correlated equilibrium, a distribution over a profile of strategies has to obey the following condition. First, observe that for every player, given a recommendation by the coordinating device, there is some marginal probability distribution over the strategy profile of the other players. It is required that the strategy recommended to the player is a best response strategy with respect to this marginal profile.

The notion of a correlated equilibrium was first suggested by Aumann. Like Nash equilibrium, it exists for every finite game (simply because Nash is a special case). It answers two concerns of game theory. One is that it often can offer an equilibrium of higher expected payoff than any Nash equilibrium. The other is that there are polynomial time algorithms for computing it (for games in standard form). We shall discuss the algorithmic aspects later, and here we shall just present an example showing a correlated equilibrium better than any Nash.

2; 1	1; 2	0; 0
0; 0	2; 1	1; 2
1; 2	0; 0	2; 1

A correlated equilibrium that picks each nonzero cell with probability 1/6 has expected payoff 3/2 for each player. Given a recommendation, a player cannot improve the expected payoff by deviating from it. For example, given the recommendation to play the first row, the row player knows that the column player must have received a recommendation to play one of the first two columns, and assuming that the column player follows the recommendation, the first row indeed gives the highest expected payoff. The game has a unique Nash equilibrium, namely, each player chooses a strategy uniformly at random, and if gives an expected payoff of only 1 to each player.

1.5 Summary of introduction

We defined the notion of a game, and presented some solution concepts for games. These include the notions of dominant strategies, pure Nash, subgame perfect equilibrium, minimax value, mixed Nash and correlated equilibrium. We discussed some of the assumptions that are used in making these definitions, and some of the shortcoming of these definitions.

In later parts, we shall accept the definitions of the solution concepts as given, and discuss algorithms for computing them.

2 Computing equilibria in pure strategies

2.1 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).

2.2 Dominant strategies

For games in standard 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). We remark that the first homework assignment (handout 1) relates to such games.

There is a well developed area of *combinatorial game theory* that we shall not address in this course. A course on this subject is given in the Weizmann Institute by Aviezri Fraenkel. 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 two open questions 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.)

Reward collecting games. The game graph is a directed bipartite graph. Each vertex has a nonnegative reward, with all rewards summing up to 1 (this is simply a normalization condition). There is a token placed on some starting vertex, and the two players, *collector* and *defender* alternate in moving the token along an outgoing edge from its current location. For every vertex visited for the first time, the collector gets to collect the reward associated with this vertex. Can the defender prevent the collector from accumulating a reward of at least 1/2? This problem is NP-hard and in PSPACE, but not much is known beyond this.

2.3 Pure Nash

For games in standard 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 strategy, 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 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 2.1 *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 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? This is the subject of the homework assignment. 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.

3 Computing equilibria in mixed strategies

3.1 Two-player win-lose games of partial information

Games of partial information are considerably more complex to play than games of full information. The optimal strategies are not necessarily deterministic – one needs to consider mixed strategies. For win-lose games, there is a natural quantity for a player to optimize, namely, the probability of winning. It might have any value between 0 and 1. One source of difficulty is that in games of partial information, the effective length of a game (number of moves, or turns) cannot be bounded simply by the number of possible positions, because the players do not necessarily know in which position the game is. As a consequence, the number of moves may even be infinite. (This subject is touched upon in a homework assignment.)

Among the interesting results concerning two player games of partial information is that the question of which player has a higher probability of winning is undecidable if the number of moves is unbounded, and EXPTIME-complete if the number of moves is polynomial. Moreover, these results hold also with respect to approximation of the winning probability. See [Reif84, FeigeShamir92, Shamir92, FeigeKilian97] for some results on this topic.

3.2 Linear programming

Many optimization problems can be formulated as linear programs. The main features of a linear program are the following:

- Variables are *real numbers*. That is, they are continuous rather than discrete.
- The objective function is a linear function of the variables. (Each variable effects the object function linearly, at a slope independent of the values of the other variables.)
- Constraints on the variables are linear.

A solution satisfying all constraints is *feasible*. A feasible solution that also optimizes the objective function is *optimal*.

Linear programs are often represented using matrix and vector representation. For example, the following is a representation of a linear program in *canonical form*. x,b,and c are column vectors, whereas A is a matrix.

minimize $c^T x$
subject to
$Ax \geq b$
$x \geq 0$

In a linear program in *general form*, the constraints are linear but may involve inequalities of both types ($\leq$ and $\geq$), as well as equalities ($=$). Variables may be required to be nonnegative ≥ 0, or else be unconstrained. Another useful form of a linear program is the *standard form*:

minimize $c^T x$
subject to
$Ax = b$
$x \geq 0$

All forms are equivalent in terms of their expressive power, and it is simple to transform a linear program in general form to standard form and to canonical form.

For linear programs in standard form, it is convenient to assume that the constraints (rows of the matrix A) are linearly independent. If the rows are not linearly independent, then it suffices to consider rows of A that constitute a basis for the row space (a maximal linearly independent set of row vectors). Either every solution that satisfies the constraints that correspond to the basis satisfies all constraints, or the LP is infeasible.

Consider an LP in standard form, with m linearly independent constraints and n variables. Let B be a submatrix of A containing exactly m linearly independent columns. This is a *basis* of the column space of A. Let x_B be the set of basic variables corresponding to the columns of B. If $B^{-1}b \geq 0$, then the following is a basic feasible solution: the basic variables are set to $B^{-1}b$, and the nonbasic variables are set to 0. Clearly this solution is feasible. Note that it satisfies n linearly independent constraints with equality: the m constrains of $Ax = b$, and $n - m$ of the nonnegativity constraints. The other (nonnegativity) constraints are also satisfied, though not necessarily with equality.

Each basis gives at most one basic feasible solution. (It gives none if the condition $B^{-1}b \geq 0$ fails to hold.) Two different bases may give the same basic feasible solution, in which case the basic feasible solution is degenerate (more than $n - m$ variables are set to 0).

The following lemma is well known and we omit its proof.

Lemma 3.1 *Every LP in standard form is either infeasible, or the optimal value is unbounded, or it has a basic feasible solution that is optimal.*

Lemma 3.1 implies that in order to solve an LP optimally, it suffices to consider only basic feasible solutions. As there are at most $\binom{n}{m}$ basic feasible solutions, we can solve LPs optimally in this time.

Recall Cramer's rule for solving $Bx = b$, where B is an invertible order n matrix. The solution is

$$x_j = \frac{\det B^j}{\det B}$$

for $1 \leq j \leq n$, where here B^j is the matrix B with column j replaced by b. If each entry in B and b is an integer with absolute value at most M, then each x_j is a rational number with numerator and denominator bounded by at most $M^n n!$. This can be used to show that the length of numbers involved in a basic feasible solution are polynomially related to the input size. (Moreover, it can be shown that when a system of linear equations is solved by Gaussian elimination, the length of intermediate numbers produced by the algorithm is also polynomially related to the input size.)

The notion of a BFS can be extended to LPs in general form. Ignoring nonnegativity constraints, if a feasible and bounded LP has m linearly independent constraints, that it always has a BFS with at most m nonzero variables.

A well known algorithm to solve linear programs is the *simplex* algorithm. It is not a polynomial time algorithm, though it appears to be pretty fast in practice. (Technically, simplex is a family of algorithms that differ by the *pivot* rule that they use. It is still open whether there is some clever choice of pivot rule that would make the algorithm polynomial. The *ellipsoid* algorithm does solve linear programs in polynomial time, though its running time in practice is quite slow. (The simplex algorithm is slower than the ellipsoid algorithm on worst case instances, but appears to be faster on average.) There are *interior point methods* that are both polynomial time in the worst case and pretty fast on average. (It is still not known whether there are *strongly polynomial time* algorithms for linear programming, whose number of operations depend only on n and m but not on the precision of the numbers involved.)

An important concept related to linear programming is the notion of *duality*. Let us first illustrate it on an example.

There are n foods, m nutrients, and a person (the *buyer*) is required to consume at least b_i units of nutrient i (for $1 \leq i \leq m$). Let a_{ij} denote the amount of nutrient i present in one unit of food j. Let c_i denote the cost of one unit of food item i. One needs to design a diet of minimal cost that supplies at least the required amount of nutrients. This gives the following linear program.

minimize $c^T x$
subject to
$Ax \geq b$
$x \geq 0$

Now assume that some other person (the *seller*) has a way of supplying the nutrients directly, not through food. (For example, the nutrients may be vitamins, and the seller may sell vitamin pills.) The seller wants to charge as much as he can for the nutrients, but still have the buyer come to him to buy nutrients. A plausible constraint in this case is that the price of nutrients is such that it is never cheaper to buy a food in order to get the nutrients in it rather than buy the nutrients directly. If y is the vector of nutrient prices, this gives the constraints $A^T y \leq c$. In addition, we have the nonnegativity constrain $y \geq 0$. Under these constraints the seller wants to set the prices of the nutrients in a way that would maximize the sellers profit (assuming that the buyer does indeed by all his nutrients from the seller). This gives the the following *dual* LP:

maximize $b^T y$
Subject to
$A^T y \leq c$
$y \geq 0$

As one can replace any food by its nutrients and not pay more, one gets *weak duality*, namely, the dual provides a lower bound for the primal. Weak duality goes beyond the diet problem and holds even if A, b, c have some entries that are negative. That is, for every pair of feasible solutions to the primal and dual LPs we have:

$$b^T y \leq (Ax)^T y = x^T A^T y \leq x^T c = c^T x \tag{1}$$

In particular, weak duality implies that if the optimal value of the primal is unbounded then the dual is infeasible, and if the optimal value of the dual is unbounded, then the primal is infeasible.

Assume that there is a pair of solutions x^* and y^* for which the values of the primal and dual LPs are equal, namely $c^T x^* = b^T y^*$. Then necessarily both x^* and y^* are optimal solutions to their respective LPs. In economics, the vector y^* is refereed to as *shadow prices*. These optimal solutions need to satisfy the inequalities of (1) with equality. This gives the following *complementary slackness* conditions:

$$(Ax^* - b)^T y^* = 0 \tag{2}$$

$$(c - A^T y^*)^T x^* = 0 \tag{3}$$

Condition (2) has the following economic interpretation. If a certain nutrient is in surplus in the optimal diet, then its shadow price is free (a *free good*). Condition (3) can be interpreted to say that if a food is overpriced (more expensive than the shadow price of its nutrients) then this food does not appear in the optimal diet.

The following table explains how to obtain the dual of a primal LP that is in general form. Here A_j denotes a row of matrix A and A^j denotes a column.

$\min c^T x$		$\max b^T y$
$A_i x \geq b_i$	$i \in I^+$	$y_i \geq 0$
$A_i x = b_i$	$i \in I^=$	y_i free
$x_j \geq 0$	$j \in J^+$	$y^T A^j \leq c_j$
x_j free	$j \in J^=$	$y^T A^j = c_j$

Note that the dual of the dual is the primal.

Weak and strong duality apply also in this case. More specifically, if the optimum to the primal is bounded, then so is the optimum to the dual, and vice versa. If the optimum to one of the LPs is unbounded, then the other is not feasible. It may also happen that neither one of them is feasible.

3.3 Max-min mixed strategies

A fairly pessimistic solution concept for games is that of a max-min strategy. This is a choice of strategy that would maximize the payoff in the "worst case" – no matter what strategies the other players use, a certain minimum payoff is guaranteed. This notion becomes more interesting when mixed strategies are involved, and the player wishes to guarantee a minimum expected payoff. (Players have risk neutral utility functions.)

Proposition 3.2 *For any game in standard form and any player:*

1. *If the payoffs are rational numbers, then the probabilities involved in a max-min strategy are rational.*
2. *A max-min strategy can be computed in polynomial time.*

Proof: W.l.o.g., let player 1 be the player for which we need to compute a max-min strategy. Let A be a payoff matrix for player 1, where its columns are indexed by the strategies of player 1, whereas its rows are indexed by profiles of strategies for the other players. We let A_j denote the jth row of A. Let x_i be the probability with which player 1 plays strategy i. Then a max-min strategy for player 1 is the solution to the following linear program.

maximize t
subject to
$A_j x - t \geq 0$ (for every j)
$\sum x_i = 1$
$x \geq 0$

The proposition follows as an immediate corollary to the theory of linear programming. □

The solution concept of a max-min strategy is often too pessimistic to be of interest. However, for one important class of games, that of zero sum (or constant sum) two person games, it is a very useful solution concept. The reason is the celebrated minimax theorem of Von-Neumann.

Theorem 3.3 *For every (finite) two person zero sum game, the payoff guaranteed to a player under his mixed max-min strategy is equal to the maximum payoff that the player can get by playing a (pure) strategy against the mixed max-min strategy of the other player.*

Proof: Let A be the payoff matrix for the column player, and $-A$ be the payoff matrix for the row player. Then the LP for a max-min strategy for the column player was given in the proof of proposition 3.2. For the row player, let y_j be the probability with which he plays strategy j. Then the LP for the max-min value for the row player is the following.

maximize t
subject to
$-A_i^T y - t \geq 0$ (for every i)
$\sum y_j = 1$
$y \geq 0$

Simple manipulations show that the LP for the row player is equivalent to the dual of the LP for the column player. To construct this dual, associated a variable y_j with every row A_j and a variable z with the constraint $\sum x_i = 1$. Also, make a change of variable of $s = -t$, changing the objective function to the equivalent **minimize** s, and changing the constraints to $A_j x + s \geq 0$. Then the dual becomes:

maximize z
subject to
$A_i^T y + z \leq 0$ (for every i)
$\sum y_j = 1$
$y \geq 0$

Renaming z as t, one gets the LP for the row player.
Both LPs are feasible and bounded. Strong duality now implies the minimax theorem □
Another useful observation concerning two player games is the following.

Proposition 3.4 *In a two player game, the support of a max-min strategy need not be larger than the number of strategies available to the other player.*

Proof: This follows by taking a basic feasible solution for the max-min LP. □

The minimax theorem plays an important role in connecting between two notions of randomness in algorithms. A *randomized algorithm* can be viewed as a probability distribution over deterministic algorithms. A worst case performance measure for it is the probability that it outputs the correct answer on the worst possible input. A *distributional algorithm* is a deterministic algorithm that is used in a case that the inputs are drawn at random from some known distribution. An average case performance measure for the algorithm is the probability (over choice of input) that it answers correctly. One can set up a zero sum game in which the row player chooses an algorithm and the column player chooses an input. The row player wins if the algorithm gives the correct answer on the chosen input.

Fixing a finite collection of algorithms and a finite collection of possible inputs, Yao's minimax principle says that the worst case performance of the optimal randomized algorithm (success probability on worst input) is exactly equal to the best average case performance against the worst possible distribution over inputs.

Another useful observation is that if the collection of algorithms is small, then the support of the difficult distribution may be small as well, and vice versa.

3.4 Multiplayer Nash

Recall that a (mixed) Nash equilibrium is a profile of mixed strategies, such that for every player, every strategy in the support of his mixed strategy is a best response (in expectation) to the profile of mixed strategies played by the other players.

Theorem 3.5 *Every finite game has a Nash equilibrium.*

A Nash equilibrium can be thought of as a "fixed point" of a game, in the sense that no player has an incentive to unilaterally deviate from it. (Note however that a player may be indifferent to deviating from a Nash equilibrium, in the sense of not caring how exactly to mix over his best response strategies.) Nash's proof of Theorem 3.5 was based on applying a different fixed point theorem, namely Brouwer's fixed point theorem that states that every continuous function from a compact convex body to itself has a fixed point.

Here is a sketch of how Nash's theorem can be proved using Brouwer's fixed point theorem. Let t be the number of players, and let n_i be the number of strategies available to player i. The set of all profiles of mixed strategies is a compact convex body G in R^d, where $d = \sum_{i=1}^{t} n_i$. One now defines a continuous function F from G to itself in a way that any fixpoint of F corresponds to a Nash equilibrium of the game. One such function is the following. Let $g_{i,j}(x)$ be the "gain" for player i with respect to profile x if he switches to pure strategy j. Observe that g is a continuous function (it is a polynomial in x). Define:

$$F(x)_{i,j} = \frac{x_{i,j} + \max[0, g_{i,j}(x)]}{1 + \sum_{k=1}^{n_i} \max[0, g_{i,k}(x)]}$$

Observe that in every point x of F, for every player i, at least for one value of j it must hold that $\max[0, g_{i,j}(x)] = 0$, because it could not be that every pure strategy is strictly better than the mixed strategy. Hence in a fixed point x of F, for all i and j it must hold that $\max[0, g_{i,j}(x)] = 0$. If j is in the support of the fixed point than it must be that $g_{i,j}(x) = 0$, as otherwise averaging implies

that there must be some other strategy j' with $g_{i,j'}(x) = 0$. This means that the mixed strategy is composed only of best responses, and hence is a Nash equilibrium.

The reader may observe that the function F above can be thought of as a way of imposing dynamics on the space of profiles, where at a step one moves from profile x to profile $F(x)$. Though this dynamical system is guaranteed to have a fixed point, the proof does not guarantee that starting from an arbitrary point, a fixed point is ever reached (or that the dynamics eventually get arbitrarily close to a fixed point).

The proof of Nash's theorem is nonconstructive, due to the fact that Brouwer's fixed point theorem is nonconstructive. There is no polynomial time algorithm known for computing a Nash equilibrium. Part of the difficulty is that there are games (Nash shows such an example with three players) in which in every Nash equilibrium, the strategies in the support are played with nonrational probabilities (even though all payoffs have integer values). In a homework assignment we will see a three player game with an irrational Nash (though this game also has some rational Nash equilibria). Hence (at least in the case where there are more than two players), one would need to represent a Nash equilibrium either symbolically (for example, allow square root notation), or approximately using rational numbers. For standard models of computation, the latter option may be more convenient. Hence one may consider a notion of an ϵ-Nash, where $\epsilon > 0$ is a parameter that measures how close a profile of strategies is to being a true Nash. There are several notions of distance that one may consider, and we list here the three main ones.

1. ϵ-close. Here one seeks a profile of mixed strategies that is close in some metric to a profile of strategies that represents some true Nash. One possible metric to use here is the ℓ_∞ norm. Namely, x is ϵ-close to a true Nash if there is a true Nash x' such that for every strategy, the probability that it is played in x differs from that in x' by at most ϵ.

2. ϵ-well supported. Here one normalizes all payoffs so that they are between 0 and 1. Then one seeks a profile x in which for every player, every strategy with nonzero probability gives expected payoff within ϵ of the expected payoff of the best response.

3. ϵ-Nash. Again, one first normalizes the payoffs. Then one seeks a profile x in which for every player, his mixed strategy gives expected payoff within ϵ of the expected payoff of the best response.

Of all three notions, the one that we call an ϵ-Nash is in a sense the weakest. Clearly, every ϵ-well supported Nash is also an ϵ Nash. It is also not difficult to see that for every game, there is a constant N that depends only on the game (for example, the total number of strategies) such that every ϵ/N-close Nash is an ϵ-Nash. The notion of ϵ-well supported Nash is not much stronger than ϵ-Nash. Every ϵ^2-Nash can be transformed into an $O(\epsilon)$-Nash. (This is left as homework.)

The notion of an ϵ-close Nash is perhaps too strong as the following argument shows (taken from [EtessamiYannakakis07]).

One can design three person games with integer payoffs that have only one Nash, and in this Nash probabilities are irrational. Now add a fourth player to the game whose payoffs depends on the strategies played by only one of the other players, and the payoffs of the first three players do not depend on the strategy played by the fourth player. Hence the best response strategy of the fourth player is determined by comparing sums of irrational numbers and determining which is larger. An ϵ-close Nash would tell us which of these irrational numbers is larger. This connects to the following well known open question. Is there a polynomial time algorithm that given integers $a_1, \ldots, a_n$ and b, can tell whether $\sum \sqrt{a_i} > b$? This problem is even not known to be in NP.

The argument above refers to games with four players. However, one can extend it to games with three players using the following theorem of Bubelis [Bubelis79].

Theorem 3.6 *Every game G with $d > 3$ players and rational payoffs can be reduced in polynomial time to a game G' with three players and rational payoffs in which player 1 in G' simulates all players*

in G in the following sense. In every Nash for G', the mixed strategy x for player 1 corresponds (after scaling by d) to a Nash of G. Every Nash profile x of G corresponds (after scaling by $1/d$) to a mixed strategy of player 1 in G' that is part of a Nash for G'.

(In fact, in Theorem 3.6 the scaling factor can be made nonuniform: $1/d_i$ for player i with $\sum 1/d_i = 1$. By setting d_4 close to 1, there is almost no loss in ϵ with respect to ϵ-closeness when moving from the four player game to the three player game.)

Given the above discussion, the interesting challenges are computation of ϵ-Nash of three player games, and Nash for two player games.

3.5 Two player Nash

In some senses, computing Nash for two player games is an easier task than for three player games.

Given the support S^1 for player 1 and S^2 for player 2 of a Nash, the mixed strategies x and y associated with them need to satisfy the following linear constraints:

- Nonnegativity: $x, y \geq 0$.
- Probability distribution: $\sum x_i = \sum y_j = 1$.
- Best response for player 1: $u_1(x', y) \geq u_2(x", y)$ for every pure strategies $x' \in S^1$ and $x"$.
- Best response for player 2: $u_2(x, y') \geq u_2(x, y")$ for every pure strategies $y' \in S^2$ and $y"$.

The probabilities involved in the mixed strategies are solutions to a linear program and hence (when taking a basic feasible solution) are rational and can be represented using polynomially many bits.

We now proceed to describe the Lemke-Howson algorithm that can be used for finding a Nash equilibrium in two player games. It also provides a proof that in two player games, Nash equilibria must exist (and hence also proves the minimax theorem).

Consider a two player game (given by payoff matrices A and B) and scale all payoffs so that they are between 1 and 2. We shall assume first that the game is non-degenerate. The non-degeneracy condition can be stated formally (includes conditions such as that every k by k submatrix has full rank), but may be informally thought of as saying that there are no "numerical coincidences". This assumption will imply that there are no ties in the Lemke-Howson algorithm (new strategies to be picked up or dropped are uniquely defined).

The Lemke-Howson algorithm maintains for each player a vector of nonnegative weights associated with pure strategies, a vector x for player 1 and a vector y for player 2. These weights can be viewed as probabilities in a mixed strategy, though they need not sum up to 1. The invariant that is maintained throughout the algorithm is that all but one of the strategies in the support of these vectors are best responses with respect to the vector of the other player, and have payoffs of 1. Here, the notion of a payoff of strategy i for player 1 is taken in the sense of $\sum_j a_{ij} y_j$ and the notion of a payoff of strategy j for player 2 is taken in the sense of $\sum_i b_{ij} x_i$. The one exceptional strategy is strategy 1 for player 1. The initialization phase proceeds as follows.

1. Set all weights to 0.
2. Arbitrarily, choose one strategy for one of the players. W.l.o.g., let it be strategy s_1 for player 1. Raise its weight until player 2 has a response t_1 with payoff 1. (Our non-degeneracy assumptions implies that this t_1 is unique.)
3. Raise the weight of t_1 until player 1 has a response s_2 with payoff 1. (Again, the non-degeneracy assumption implies that s_2 is uniquely determined.)

If after the initialization phase $s_1 = s_2$ then s_1 is a best response to t_1 and vice versa, and we are done. If $s_1 \neq s_2$ then the main body of the algorithm takes place. The main body of the algorithm maintains after each step one of the following two types of configurations:

1. The support $s(x)$ of x is equal in size to the support $s(y)$ of y. In addition there is one more strategy p (for *pivot*) for player 1 that is a best response to y and currently has a weight of 0.
2. $|s(x)| = |s(y)| + 1$. In addition there is one more strategy p for player 2 that is a best response to x and currently has a weight of 0.

After the initialization phase the algorithm is in the first of the two configurations.
Each of the two configurations offers a possible move to each one of the players as follows.

1. $|s(x)| = |s(y)|$.
   (a) Move for player 1. Pick up p into the support of x. Now x contains one more free variable than the number of constraints (best responses) in y. Hence raising x_i that corresponds to p can be offset by changing the other values in $s(x)$ while still maintaining that all strategies in $s(y)$ have payoff 1. A step for player 1 is raising x_i until either some other x_j reaches 0 (and then we remain in configuration 1 and s_j becomes the new pivot strategy), or some new y_j becomes a best response (and then we move to configuration 2 with s_j as the new pivot strategy). (One of the two must happen since payoffs are strictly positive.)
   (b) Move for player 2. Ignore p. Ignoring also s_1, there is one more free variable in the support of y then constraints (best responses) in x. Maintaining the constraints, move along the line implied in y in the direction that lowers the payoff of p_i. Stop when either some y_j reaches 0 (and then we move to configuration 2 and s_j becomes the new pivot strategy), or some new x_j becomes a best response (and then we remain in configuration 1 and s_j becomes the new pivot strategy).
2. $|s(x)| = |s(y)| + 1$.
   (a) Move for player 1. Ignore p. Now x contains one more free variable than the number of constraints (best responses) in y. Maintaining the constraints, move along the line implied in x in the direction that lowers the payoff of p. Stop when either some x_j reaches 0 (and then move to configuration 1 and s_j becomes the new pivot strategy), or some new y_j becomes a best response (and then remain in configuration 2 and s_j becomes the new pivot strategy).
   (b) Move for player 2. Pick up p into the support of y. Ignoring s_1, there is one more free variable in the support of y then constraints (best responses) in x. Hence raising y_i that corresponds to p can be offset by changing the other values in $s(y)$ while still maintaining that all strategies in $s(x) - s_1$ have payoff 1. A step for player 2 is raising y_i until either some other y_j reaches 0 (and then we remain in configuration 2 and s_j becomes the new pivot strategy), or some new x_j becomes a best response (and then we move to configuration 1 with s_j as the new pivot strategy).

We may describe the Lemke-Howson algorithm in terms of a directed state graph. Each state in the graph is a configuration given by the sets $s(x)$, $s(y)$ and the pivot strategy p. The non-degeneracy condition implies that given a configuration, the probability with which each strategy is played in this configuration is determined uniquely, and hence no two states share the same configuration. It follows that the number of possible states is exponential in the number of pure strategies available to the players (and not larger). The state graph for the Lemke-Howson algorithm has out-degree 2. A crucial observation is that the pivoting process is *reversible*. Hence edges of the state graph are bidirectional. There are certain special states of degree 1. One is the configuration reached after

the initialization phase. The other states of degree 1 are those in which either the strategy to be dropped is s_1 (and then player 2 has no move) or the strategy to become a best response is s_1 (and then player 1 has no move). In either case, these are Nash equilibria. As a graph has an even number of vertices of odd degree, it follows that there must be an odd number of Nash equilibria, and in particular, at least one. To find one Nash equilibrium, start from the trivial state and let the players alternate in turns. This process is guaranteed to reach a Nash equilibrium.

The Lemke-Howson algorithm is not a polynomial time algorithm. There are known families of examples that require an exponential number of steps to reach a fixed point [SavaniStengel04].

Many games are degenerate, in which case the Lemke-Howson algorithm cannot be run as described. A standard approach of dealing with degeneracies is by introducing small perturbations to the payoffs (e.g., for a very small ϵ, reward player 1 by ϵ^i for playing strategy i). The new game will not be degenerate, and its Nash will be ϵ-Nash of the original game (and vice versa). In fact, when ϵ is sufficiently small, the support of every Nash in the new game will be a support of a Nash in the original game. There are versions of the Lemke-Howson algorithm that do not introduce an explicit perturbation but instead follow some pivot rule that is equivalent to introducing a small perturbation.

It is not known whether there is a polynomial time algorithm for computing a two-player Nash. It is know that many variations of this question are NP-hard. For example, consider the following two player game that we call here the k-clique game.

The payoff matrix A for the row player has $n+1$ rows and n columns. The top n by n submatrix is the adjacency matrix of a graph G. All the entries of the bottom row are $\frac{k-1}{k}$. For the column player, the top n by n submatrix is the identity matrix, and the bottom row is all 0.

Theorem 3.7 *If G does not have a clique of size k, then in all Nash equilibria of the k-clique game the support for the row player is the last row, and the payoff for the column player is 0. If G has a clique of size k, then there are Nash equilibria in which the support for the row player has at least k rows, and the expected payoff for the column player is positive.*

Proof: Consider an arbitrary Nash equilibrium, and let j the column played by the column player with highest probability. If this probability $y_j > 1/k$ then in the Nash, the row player cannot be playing row j (because the expected payoff of row j is strictly smaller than $1-1/k$, whereas the payoff of row $n+1$ is always $1-1/k$). Hence the expected payoff for the column player when playing column j is 0, implying that he has no strategy with positive payoff. This in turn implies that the row player is playing row $n+1$ at this Nash.

Assume now that the row player is playing some strategy $i < n+1$ with positive probability. This implies that the column player is playing each strategy with probability at most $1/k$. Let S be the set of columns played by the column player with positive probability, and observe that $|S| \geq k$. Then to be best responses, all rows corresponding to S must be played by the row player. Moreover, to be best responses, the expected payoff for each such row is at least $1-1/k$ (which is the payoff guaranteed by row $n+1$). We now show that the subgraph of G induced on S must have a k-clique.

By averaging, there must be a column in S for which at least a $\frac{k-1}{k}$ fraction of the rows of S have payoff 1. Put the corresponding vertex in the clique, and continue with a set $S' \subset S$ of those vertices that had payoff 1. For S', using the fact that no column has probability greater than $1/k$, averaging shows that there must be a column in S' for which at least a $\frac{k-2}{k-1}$ fraction of the rows of S' have payoff 1. Continuing in this fashion, we can extract a clique of size $k-1$ and still have some row left in S', which can serve as the kth vertex in the clique.

Finally, observe that if S is any *maximal* clique of size at least k, then both players playing uniformly over S gives a Nash with payoff $1-1/|S'|$ for the row player and $1/|S'|$ for the column player. □

Theorem 3.7 shows (among other things) that in a game with non-negative payoffs, it is NP-hard to determine whether there is a Nash equilibrium in which the column player has expected positive

payoff, and to determine whether there is a Nash equilibrium in which the row player has a mixed strategy (rather than a pure strategy).

The Lemke-Howson algorithm places Nash in the class PPAD [Papadimitriou94], standing for *Polynomial Parity Argument Directed*. This class assumes a state graph of degree two (similar to the state graph of the non-degenerate Lemke-Howson algorithm), but these graphs are directed in a consistent way (a vertex has at most one incoming edge and one outgoing edge). There is a way of assigning a consistent set of directions to the Lemke-Howson graph, though this is not easy to state. Assigning a direction is easier for the directed graph of a related problem, that associated with the proof of the two-dimensional Sperner lemma. (The proof of Sperner's lemma was presented in class, and is not presented here because it will be a pity to present it without an accompanying drawing. The direction there is determined by whether the unique color is to the left or to the right of the movement.) Recently, it was shown that two player Nash is complete for PPAD [ChenDeng06] (even for ϵ-Nash for polynomially small ϵ [ChenDengTeng06]). As computing an ϵ-Nash for multi-player games is in PPAD (this is not easy to see, but will appear in the full version of [DGP06]), this implies a reduction from multi player games to two player games, preserving the concept of ϵ-Nash (though ϵ needs to be very small for the two player game). For other problems in PPAD, see [Papadimitriou94]. Some problems related to PPAD computations are known to be very difficult. This is touched upon in the homework assignment. Finding the other end of a PPAD computation is PSPACE complete (since reversible computation is universal [Bennett73]). Reversible Turing machines are not in PPAD because in a PPAD problem one is allowed to output any sink (or any nonstandard source) and in reversible computation one seeks a particular sink.

Research questions.

1. Is there a direct (relatively simple) reduction from finding an ϵ-Nash in a 3-player game to finding a Nash in a two player game?

2. Is there a direct (relatively simple) reduction from finding a Nash in a two player game to finding an ϵ-Nash (with polynomially small ϵ) in a two player game?

The question of finding ϵ-Nash in two player games has received much attention. For sufficiently large ϵ (e.g., $\epsilon > 0.34$) there are known polynomial time algorithms. For every Nash and any ϵ, there is a nearby ϵ Nash with support size $O(\epsilon^{-2} \log n)$. This implies that optimization over ϵ Nash can be done by enumeration in time $n^{O(\epsilon^{-2} \log n)}$. See [LMM03].

3.6 Correlated equilibrium

Consider a multi-player game and let $s_1, s_2, \ldots, s_N$ be all its possible strategy profiles. Let $p^i(s_j)$ denote the payoff to player i if strategy profile s_j is played. Let $s_j \oplus_i s'$ be the strategy profile that results from s_j by replacing the strategy for player i by strategy s' (where s' is a strategy available to player i).

Recall that a *correlated equilibrium* is a probability distribution over strategy profiles that no player has an incentive to deviate from. Formally, let x_j be the probability associated with strategy profile s_j. Then a correlated equilibrium needs to satisfy the following set of constraints. For every player i, for every strategy s for player i and every strategy s' for player i:

$$\sum_{s \in s_j} p^i(s_j) x_j \geq \sum_{s \in s_j} p^i(s_j \oplus_i s') x_j$$

where the sums are taken only over those strategy profiles in which player i plays strategy s. Adding the nonnegativity condition $x \geq 0$ and the requirement $\sum x_j = 1$ that probabilities sum up to 1, we see that a correlated strategy is a feasible solution to a polynomial size (if the game is given in standard form) linear program. This linear program is indeed feasible, because a mixed Nash equilibrium is a correlated equilibrium. The discussion above provides the proof to the following proposition.

Proposition 3.8 *Given a game in standard form in which all payoffs are rational, a correlated equilibrium can be computed in polynomial time. Furthermore, for any linear objective function (with rational coefficients) over the payoffs to the players, one can find in polynomial time a correlated equilibrium maximizing the objective function, and moreover, the probabilities involved will all be rational.*

The above proposition helps make correlated equilibria a desirable solution concept, when it applies (when there is some trusted party that draws a random profile from the distribution implied by the correlated equilibrium, and presents to every player his recommended strategy and no additional information about the strategies recommended to other players).

Homework assignments

Please keep the answers short and easy to read.

5 November 2008

Read chapter 1 in [NRTV07].

The family of *and-or* games is a family of two-player constant-sum games given in extensive form (as a game tree). There are two players, *min* and *max*. The game tree is a full binary tree of depth n and $N = 2^n$ leaves. Every leaf has a value, which is the payoff given to *max*, which can be either 1 (win) or 0 (loose). The payoff to *min* is the complement of the payoff to *max* (when *max* looses *min* wins). The game starts at the root of the tree. Number the layers of the tree from 0 at the root to n at the leaves. At even numbered layers, *min* chooses a move (an edge leading from the current node to the next layer), and at odd layers *max* chooses a move. After n moves a leaf is reached, each player gets his respective payoff, and the game ends.

1. How would an *and-or* game be represented in standard form? In particular, how many rows and how many columns will the game matrix have?
2. Prove that in every *and-or* game one of the players has a strategy that forces a win for that player regardless of the strategy of the other player. (Hint: you may use induction.) Show an algorithm of time complexity poly(N) for computing such a strategy. (Space complexity is also important in practice. Space poly(n) suffices for computing an optimal move.)
3. What is the smallest number of leaves that can have value +1 and still *max* will have a winning strategy? Explain.
4. Prove that in every *and-or* game, at least one of the players has a dominant strategy.
5. Show an example of an *and-or* game in which *min* does not have a dominant strategy.
6. What is the largest number of leaves that can have value +1 and still *max* will have a dominant strategy but no winning strategy? Explain.
7. Show an algorithm of time complexity poly(N) for computing a subgame perfect Nash equilibrium for an *and-or* game.

12 November 2008

Reading: you are assumed to know basic notions in computational complexity (definitions of classical complexity classes such as P, NP, PSPACE and EXPTIME, the notions of *hardness* and *completeness*). If needed, refresh your memory on these concepts. See for example [Papadimitriou].

In class we showed the well known algorithm for finding a stable matching (a.k.a. stable marriage) of Gale and Shapely [GaleShapley62]. This algorithm is also presented in Chapter 10 in [NRTV07], in Wikipedia, and elsewhere.

Consider the following game with $2n$ players, n men and n women, each having his/her own preference list over partners of the other sex. In this game, every man and every woman supplies a preference list (either their true preference list, or some other preference list), the outcome of the game is the matching produced by the stable matching algorithm when run on the supplied preference lists (the algorithm where the unengaged men propose), and the payoff for a player is the rank (in the player's 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?

1. We remarked (without proof) in class that the algorithm is optimal for the men. Prove that for every man, reporting his true preference list is a dominant strategy. (Remark: this homework turned out to be too difficult, especially as originally I gave a misleading hint, and the proof in [NRTV07] has gaps in it.)

2. Show that all players following the strategy of reporting their true preference lists is not necessarily a Nash equilibrium of the game. Namely, show an example ($n = 3$ suffices for this purpose), where a woman can benefit (eventually be matched by the algorithm to a man that she prefers more) by reporting a preference list that is different from her true preference list.

3. Prove that this game always has some pure Nash equilibrium (though as question 2 shows, in this Nash equilibrium some players might not be reporting their true preferences).

19 November 2008

Reading: linear programming is a very useful algorithmic technique that has a well developed theory behind it. If this subject is new to you, it is recommended that you read more about it.

The surprise examination paradox.

A teacher announces that in the following week there will be a surprise examination. A clever student argues by backward induction that having a surprise examination is impossible. (The exam cannot be on the last day because by then the students will not be surprised by it. Having agreed that it cannot be on the last day, by the time the day before last arrives, the students expect it to be given on that day. And so on.) Here we consider a multi-round two player zero sum game between a teacher and a student. In every round the teacher has two possible actions: either A (to give an exam) or B (not to give an exam). In every round, the student has two possible actions: either A (to study towards an exam) or B (not to study). In every round, both players play simultaneously. In the unbounded version of the game, the game ends on the first round on which at least one of the players plays A. If on that round both play A the student wins, and if only one of them played A the teacher wins. If the game never ends, no player wins (it is a tie). In the bounded version of the game, the game is known to last for at most 4 rounds. If no player wins by the 4th round (no player played A in any of the first 4 rounds), then the student wins.
Please keep the answers to the following questions short and easy to read.

1. In the bounded game, what is a max-min strategy for the teacher? What is a max-min strategy for the student? How would you provide a simple proof that each of the strategies that you present is indeed a max-min strategy?

2. In the unbounded game, is there a max-min strategy for the teacher? For the student? Explain.

Food for thought (no need to hand in):

1. We said in class that a two player game of perfect information can be assumed not to repeat the same position twice (when it does the game can be declared a tie). Does this observation apply to the surprise examination game?

2. Does the surprise examination game have a mixed Nash equilibrium? If not, then what solution concept would you suggest for this game?

26 November 2008

Reading: More information on the computation of of Nash equilibria can be found in chapters 2 and 3 of [NRTV07].

1. Prove that for every finite two-player zero-sum game, in every Nash equilibrium every player is playing a max-min (mixed) strategy.

2. A pure strategy s in a two player game is said to be *dominated* if for every mixed strategy t of the other player, strategy s is not a best response with respect to t. Clearly, a dominated strategy cannot be part of a Nash equilibrium. Show that there is a polynomial time algorithm for detecting whether a two player game (given in standard form) has a dominated strategy. (Hence such strategy can be removed prior to attempting to find a Nash equilibrium.)

3. Show that there is a universal constant c (say, $c = 4$) such that in every two person game with payoffs between 0 and 1, every ϵ-Nash can be changed into a $c\sqrt{\epsilon}$-well supported Nash that is supported only on strategies that appear in the support of the given ϵ-Nash.

3 December 2008

1) Consider the following three player game. Player A has strategies a1 and a2, player B has strategies b1 and b2, and player C has strategies c1 and c2. The payoffs are described below. The name of a player appearing in a strategy profile means that the player gets a payoff of 1. Otherwise the payoff is 0. For example, on profile (a1,b2,c2) players A and B each gets a payoff of 1 and player C gets a payoff of 0.

```
          b1          b2                      b1          b2
     |----------|----------|            |----------|----------|
a1   |          |    B     |       a1   |    C     |   A; B   |
     |----------|----------|            |----------|----------|
a2   |    A     |   A; C   |       a2   |   B; C   |          |
     |----------|----------|            |----------|----------|

                c1                                  c2
```

Equivalently, the payoff for each player can be described as follows: if a player plays his first strategy he gets a payoff of 1 iff the two other players play their second strategy. If a player plays his second strategy, he gets a payoff of 1 iff the player preceding him (in the cyclic order A-B-C-A) plays his first strategy.

Find *all* Nash equilibria of this game, and prove that no other Nash equilibrium exists. (For the proof, you may need to solve a system of algebraic equations that expresses the conditions for a profile of strategies being a Nash equilibrium.)

2) Recall that problems in PPAD are problems whose input includes an implicit description of a directed graph with at most exponentially many nodes. There is a polynomial time algorithm that given the name of a node figures out from the implicit description the edges incident with the node. Every node has at most one incoming edge and at most one outgoing edge. One is given a source node (has no incoming edge), and the goal is to find any sink node (has no outgoing edge). The *matching-sink* problem is more specific and requires one to output the sink node that lies on the end of the path of the given source node. Prove that *matching-sink* is NP-hard. (Hint: related to exhaustive search.) Remark: *matching-sink* is in fact PSPACE-complete.

References

[Bennett73] C. H. Bennett, "Logical reversibility of computation," IBM Journal of Research and Development, vol. 17, no. 6, pp. 525-532, 1973.

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

[Bubelis79] V. Bubelis. On Equilibria in Finite Games. *Int. Journal of Game Theory, Vol 8, Issue 2, page 65- 79*, 1979.

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

[ChenDeng06] Xi Chen, Xiaotie Deng: Settling the Complexity of Two-Player Nash Equilibrium. FOCS 2006: 261-272

[ChenDengTeng06] Xi Chen, Xiaotie Deng, Shang-Hua Teng: Computing Nash Equilibria: Approximation and Smoothed Complexity. FOCS 2006: 603-612

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

[DGP06] Constantinos Daskalakis, Paul W. Goldberg, Christos H. Papadimitriou: The complexity of computing a Nash equilibrium. STOC 2006: 71-78

[EtessamiYannakakis07] Kousha Etessami, Mihalis Yannakakis: On the Complexity of Nash Equilibria and Other Fixed Points. FOCS 2007: 113-123

[FeigeKilian97] Uriel Feige, Joe Kilian: Making Games Short. STOC 1997: 506-516.

[FeigeShamir92] Uriel Feige, Adi Shamir: Multi-Oracle Interactive Protocols with Constant Space Verifiers. J. Comput. Syst. Sci. 44(2): 259-271 (1992).

[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): 199214.

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

[LMM03] Richard J. Lipton, Evangelos Markakis, Aranyak Mehta: Playing large games using simple strategies. ACM Conference on Electronic Commerce 2003: 36-41

[NRTV07] Noam Nisan, Tim Roughgarden, Eva Tardos and Vijay V. Vazirani (Editors), Algorithmic Game Theory, Cambridge University Press, 2007.

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

[Papadimitriou94] Christos H. Papadimitriou: On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. J. Comput. Syst. Sci. 48(3): 498-532 (1994).

[Reif84] John H. Reif: The Complexity of Two-Player Games of Incomplete Information. J. Comput. Syst. Sci. 29(2): 274-301 (1984).

[SavaniStengel04] Rahul Savani, Bernhard von Stengel: Exponentially Many Steps for Finding a Nash Equilibrium in a Bimatrix Game. FOCS 2004: 258-267

[Shamir92] Adi Shamir: IP = PSPACE. J. ACM 39(4): 869-877 (1992).

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