

## Lecture 2: Strategic and Extensive form games

Lecturer: Peter Bro Miltersen

Scribe: Maria Elbek Andersen

## 1 Solving Matrix Games using Linear Programming

In the last lecture we saw that the proof of the minimax theorem yields the following corollary.

**Corollary 1** *Maximin/minimax strategies and values can be found in polynomial time (given matrix, with entries of rational numbers, of matrix game as input).*

However, the linear program can also be derived directly.

**Example:** The modified rock, scissors and paper game from last lecture.

	$r$	$s$	$p$	
$R$	0	1	-1	$x_1$
$S$	-1	0	1	$x_2$
$P$	2	-1	0	$x_3$

There are four variables, where  $x_1$ ,  $x_2$  and  $x_3$  are probabilities and  $v$  is the value (remember that  $\underline{v} = \bar{v} = v$ ). Solve with linear programming:

$$\begin{array}{ll}
 \max_{x_1, x_2, x_3, v} & v \\
 \text{s.t.} & \text{r: } 0 \cdot x_1 + (-1) \cdot x_2 + 2 \cdot x_3 \geq v \\
 & \text{s: } 1 \cdot x_1 + 0 \cdot x_2 + (-1) \cdot x_3 \geq v \\
 & \text{p: } (-1) \cdot x_1 + 1 \cdot x_2 + 0 \cdot x_3 \geq v \\
 & x_1 + x_2 + x_3 = 1 \\
 & x_1, x_2, x_3 \geq 0
 \end{array}$$

We now get back to the question we asked in the end of the last lecture: Can matrix games be solved in strongly polynomial time? The answer is yes, if linear programs can be solved in strongly polynomial time. So more precisely, we should ask if it is possible that matrix games can be solved in strongly polynomial time but that linear programs can not. Common wisdom says that this is not possible. In particular, let us have a look of what Dantzig found out in 1948, about a reduction of general linear programming to solving matrix games. Given an LP (in standard form)

$$\begin{array}{l}
 P : \max c^T x \\
 Ax \leq b
 \end{array}$$

Does it have an optimal solution? (so it not infeasible or unbounded) The answer is: Yes if and only if

$$\begin{aligned}
 P' : \\
 Ax &\leq b \\
 c^T x &= w \\
 \text{and} \\
 A^T y &\geq L \\
 b^T y &= w
 \end{aligned}$$

is feasible, by the duality theorem. Dantzig's observation is that  $P'$  is feasible if and only if the following matrix game

$$G = \begin{bmatrix} 0 & -A^T & c \\ A & 0 & -b \\ -c^T & b^T & 0 \end{bmatrix} \begin{matrix} x^* \\ y^* \\ z^* \end{matrix}$$

has some maximin strategy  $(x^*, y^*, z^*)$  that plays the last row with *nonzero probability*  $z^* > 0$ . We shall not give a proof of this statement, but we remark that the  $x$ -part of the feasible solution to  $P'$  is in that case given by  $x = x^*/z^*$  and the  $y$ -part is given by  $y = y^*/z^*$ . Since  $G$  is skew-symmetric, the game appear identical for the two players. We call such a game *symmetric*. It is easy to see that the value of a symmetric game is 0, like in the (unmodified) rock, scissors and paper game. In particular, Dantzig certainly did not reduce the general linear programming problem to computing the value of a matrix game. Thus, it still seem that a proof of the following statement is lacking: *If the value of a matrix game can be found in strongly polynomial time then linear programming has a strongly polynomial time algorithm.* Can a proof of this statement be given? The lecturer doesn't know....

## 2 Maximin strategies and Nash equilibria

**Definition 2** (for general games) Given  $\sigma_i \in \tilde{S}_i$  (mixed strategy for player  $i$ ) and  $\sigma_{-i} \in \times_{j \neq i} \tilde{S}_j$  (mixed strategy for the other players). We say that  $\sigma_i$  is a best reply or best response to  $\sigma_{-i}$ , if  $\sigma_i$  is

$$\arg \max_{\pi_i} \tilde{u}_i(\pi_i, \sigma_{-i})$$

**Definition 3** A strategy profile  $\sigma \in \times_{i=1}^l \tilde{S}_i$  is a Nash equilibrium if  $\sigma_i$  is a best reply to  $\sigma_{-i}$  for all  $i$ .

Nash equilibrium is a central solution concept in game theory. Note that it is conceptually very different from maximin. Nash equilibrium is a descriptive notion capturing *stability* while Maximin is a normative notion capturing *optimal guarantees*. Still, for the case of a two-player, zero-sum game, we can show that the two notions coincide.

**Proposition 4** For a two-player, zero-sum game, we have:  
*Nash equilibria = Maximin strategies  $\times$  Minimax strategies*

**Proof**  $\supseteq$  : We have a strategy profile  $(x^*, y^*)$  where  $x^*$  is maximin and  $y^*$  be minimax. The expected outcome of play must be

$$(x^*)^T A(y^*) = v = \underline{v} = \bar{v},$$

as both players are guaranteed an outcome at least this good. Can any player deviate and get better outcome? No! This would violate guarantee of other players.

$\subseteq$  : Let  $(x^*, y^*)$  be a Nash equilibrium and let

$$v(x^*) = \min_{y \in \Delta_m} (x^*)^T A y, \quad v(y^*) = \max_{x \in \Delta_n} x^T A y^*$$

(we may call  $v(x^*)$  "the value of strategy  $x^*$ "). We should prove that  $x^*$  is maximin and that  $y^*$  is minimax. We shall just show that  $x^*$  is maximin; the other proof is similar. So suppose, to the contrary, that  $x^*$  is not maximin. That is,  $v(x^*) \neq \underline{v}$ . Then  $v(x^*) < \underline{v}$ . If  $(x^*)^T A y > v(x^*)$ , then player 2 can deviate and achieve  $v(x^*)$ , contradicting the Nash equilibrium property  $\frac{1}{2}$ . On the other hand, if  $(x^*)^T A y \leq v(x^*)$ , then player 1 can deviate to his maximin strategy and achieve  $\underline{v}$ , contradicting the Nash equilibrium property  $\frac{1}{2}$ .  $\square$

### 3 Extensive form games (Game Trees, Kuhn Trees)

The strategic form of a game is a very general and clean way of representing a game, but it is not a very convenient one. Suppose for instance, that we want to represent the game tic-tac-toe in the strategic form. We need a matrix with a row for every possible way of playing tic-tac-toe as X and a column for every possible way of player tic-tac-toe as O. It may not be very obvious how to enumerate the possible ways of playing tic-tac-toe. We shall define an alternative representation of a game, the *extensive form* or *game tree* or *Kuhn tree* representation which will make it more explicit what those possible ways of playing a game is. The extensive form of a game thereby gives a better visualization of the game. Also, as we shall see, it provides a much more compact representation of games such as, say, tic-tac-toe, than the strategic form. This compactness is clearly important for computational purposes.

A game in extensive form is a rooted tree. Each node of the tree is also called a *position*. Each position belongs to exactly one player, or to *nature*. If a position belongs to nature, a fixed probability distribution is associated to its outgoing arcs. For each player, a partition (i.e., an equivalence relation) is given on the set positions belonging to that player. The equivalence classes are called *information sets*. Intuitively, the player cannot distinguish between the nodes in an information set. In each position of the game belonging to a player, the outgoing arcs are also known as the *actions* of that position. Intuitively, a player must choose between one of these actions whenever he finds himself in that position. Each action has a name. If two positions are in the same information set, the set of action names in those positions should coincide. On the other hand, if two positions are *not* in the same information sets, we require that the actions names in those positions are disjoint sets (this will be convenient later on).

**Example:** Basic Endgame in Poker The rules of the game are:

- (1) Both players put \$1 into the pot.
- (2) Player 1 is dealt a card (he inspects it, but keeps it hidden from player 2). A heart a winning card for Player 1. So it is a winning card with probability  $\frac{1}{4}$  and a losing card with probability  $\frac{3}{4}$ .

(3) Player 1 either bets or he checks. If he bets, he puts \$2 more into the pot.

If player 1 bets:

(4) Player 2 either calls or folds. If he folds, he loses \$1, no matter what card player 1 has. If he calls, he adds \$2 to the pot.

(5) If player 1 has hearts, he wins the pot. He also wins if he bets and player 2 folds. Otherwise player 2 wins.

We draw a Kuhn tree for this game (Figure 1). To each vertex of the game tree, we attach a label indicating which payer is to move from that position. The random card that is deal in the beginning, we generally refer to as moves by nature and use the label N. At each terminal vertex, we write the numerical value of player 1’s winnings (= player 2’s losses, because we are in a zero-sum game). Player 2 does not know player 1’s card, that is when it is his turn to move, he does not know at which of his two possible positions he is. To indicate this on the diagram we are encircling the two positions in a closed curve to indicate that these two vertices constitute an infomation set. The two vertices at which player 1 is to move constitute two separate infomation sets since he has inspected the card, and know at which positions he is.

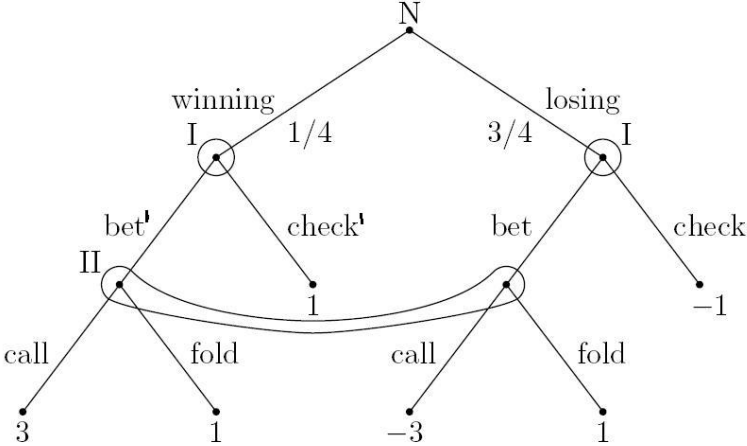


Figure 1: Game tree

In general, information sets could describe situations in which one player has forgotten a move he has made earlier in the game or some information he once knew (see Player 1’s information set in figure 2). However, we do not allow this in our course. We only deal with games of **perfect recall**, which are games in which players remember all past information they once knew and all past moves they made. Formally, a game tree satisfies the perfect recall condition if, for all nodes  $x$  and  $y$  belonging to the same information  $h$  set belonging to Player  $j$  the following is true: The sequence of action names performed by Player  $j$  on the path from the root of the tree to  $x$  is identical to the sequence of action names performed by Player  $j$  from the root of the tree to  $y$ . Note that this also

implies that the sequences of information sets belonging to that player encountered on those two paths coincide, as actions in different information sets are required to have different names. There is a conceptual reason for demanding the perfect recall condition: Arguably, if a player forgets information, this should not be part of a model of the *game*, it should be part of a model of the *player*. There is also a computational reason: Most computational problems associated with game without perfect recall, such as value computation, turns out to be **NP**-hard.

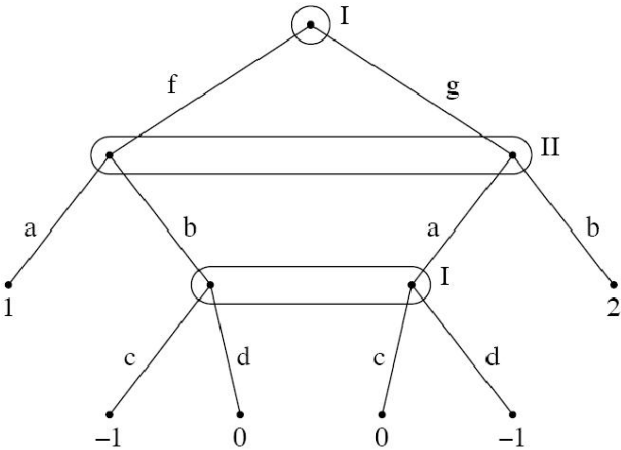


Figure 2: Game tree, not perfect recall

### 4 Converting from the extensive form to the strategic form

We can convert a game from extensive form to strategic form. This conversion procedure can be regarded as the definition of the “semantics” of the notion of an extensive form game.

**Definition 5** The strategic form game corresponding to an extensive form game is the following. Let  $K_i$  be the set of information sets  $h$  belonging to player  $i$  in the extensive form game. Then we let the strategy space for player  $i$  in the strategic form game be the set

$$S_i = \times_{h \in K_i} \text{ set of action names in } h$$

We also need to define the payoff functions. Note that a strategy profile  $(x_1, x_2, \dots, x_l)$  with  $x_i \in S_i$  can be viewed as a set of selected actions, one for each information set of the game. Given a strategy profile, we now consider the following random process: We put a pebble in the root of the game. If the pebble is in a position belonging to nature, we take a random sample from the probability distribution on the outgoing arcs indicated at the position, and move the pebble along the randomly chosen arc. If the pebble is in a position belonging to a player, we take the outgoing arc corresponding to the action chosen by the strategy profile. The payoff  $u_i(x_1, x_2, \dots, x_l)$  for Player  $i$  is defined to be the expected value of the payoff of Player  $i$  found in the leaf of the tree where the pebble ends up.

We can convert our Basic Endgame in Poker from extensive form to strategic form. Player 1 has two information sets, in each set he must make choice from among two options. He therefore has  $2 \cdot 2 = 4$  pure strategies. We may denote them by

- (b',b): bet with a winning card or a losing card.
- (b',c): bet with a winning card, check with a losing card.
- (c',b): check with a winning card, bet with a losing card.
- (c',c): check with a winning card or a losing card.

Therefore,  $S_1 = \{(b', b), (b', c), (c', b), (c', c)\}$ . Player 2 has only one information set.

C: if player 1 bets, call.

F: if player 1 bets, fold.

Therefore,  $S_2 = \{(C, F)\}$ . The payoff function on two strategies is the expected payoff when strategies are played against each other in the tree, as explained formally in the definition. Suppose Player 1 uses (b',b) and Player 2 uses C. Then the expected payoff is

$$u_1((b, b), C) = \frac{1}{4}(3) + \frac{3}{4}(-3) = -\frac{3}{2}$$

This gives the upper left entry in the following matrix. The other entries may be computed similarly.

	<i>C</i>	<i>F</i>
<i>(b, b)</i>	$-\frac{3}{2}$	1
<i>(b, c)</i>	0	$-\frac{1}{2}$
<i>(c, b)</i>	-2	1
<i>(c, c)</i>	$-\frac{1}{2}$	$-\frac{1}{2}$

In this example the payoff matrix is manageable. But in general, the blowup in size when going from extensive form to strategic form is exponential. Say, suppose Player 1 has 100 information sets, each with a choice between two actions. Then the number of rows in the matrix of the corresponding matrix game is  $2^{100}$ . So, we often like to represent a game in extensive form.