CHAPTER 1

Basic Solution Concepts and Computational Issues

Éva Tardos and Vijay V. Vazirani

Abstract

We consider some classical games and show how they can arise in the context of the Internet. We also introduce some of the basic solution concepts of game theory for studying such games, and some computational issues that arise for these concepts.

1.1 Games, Old and New

The Foreword talks about the usefulness of game theory in situations arising on the Internet. We start the present chapter by giving some classical games and showing how they can arise in the context of the Internet. At first, we appeal to the reader's intuitive notion of a "game"; this notion is formally defined in Section 1.2. For a more in-depth discussion of game theory we refer the readers to books on game theory such as Fudenberg and Tirole (1991), Mas-Colell, Whinston, and Green (1995), or Osborne and Rubinstein (1994).

1.1.1 The Prisoner's Dilemma

Game theory aims to model situations in which multiple participants interact or affect each other's outcomes. We start by describing what is perhaps the most well-known and well-studied game.

Example 1.1 (Prisoners' dilemma) Two prisoners are on trial for a crime and each one faces a choice of confessing to the crime or remaining silent. If they both remain silent, the authorities will not be able to prove charges against them and they will both serve a short prison term, say 2 years, for minor offenses. If only one of them confesses, his term will be reduced to 1 year and he will be used as a witness against the other, who in turn will get a sentence of 5 years. Finally

if they both confess, they both will get a small break for cooperating with the authorities and will have to serve prison sentences of 4 years each (rather than 5).

Clearly, there are four total outcomes depending on the choices made by each of the two prisoners. We can succinctly summarize the costs incurred in these four outcomes via the following two-by-two matrix.

P1 \ P2	Confess	Silent
Confess	4, 4	5, 1
Silent	1, 5	2, 2

Each of the two prisoners "P1" and "P2" has two possible strategies (choices) to "confess" or to remain "silent." The two strategies of prisoner P1 correspond to the two rows and the two strategies of prisoner P2 correspond to the two columns of the matrix. The entries of the matrix are the costs incurred by the players in each situation (left entry for the row player and the right entry for the column player). Such a matrix is called a *cost matrix* because it contains the cost incurred by the players for each choice of their strategies.

The only stable solution in this game is that both prisoners confess; in each of the other three cases, at least one of the players can switch from "silent" to "confess" and improve his own payoff. On the other hand, a much better outcome for both players happens when neither of them confesses. However, this is not a stable solution – even if it is carefully planned out – since each of the players would be tempted to defect and thereby serve less time.

The situation modeled by the Prisoner's Dilemma arises naturally in a lot of different situations; we give below an ISP routing context.

Example 1.2 (ISP routing game) Consider Internet Service Providers (ISPs) that need to send traffic to each other. In routing traffic that originates in one ISP with destination in a different ISP, the routing choice made by the originating ISP also affects the load at the destination ISP. We will see here how this situation gives rise to exactly the Prisoner's dilemma described above.

Consider two ISPs (Internet Service Providers), as depicted in Figure 1.1, each having its own separate network. The two networks can exchange traffic via two transit points, called peering points, which we will call C and S.

In the figure we also have two origin–destination pairs s_i and t_i each crossing between the domains. Suppose that ISP 1 needs to send traffic from point s_1 in his own domain to point t_1 in 2nd ISP's domain. ISP 1 has two choices for sending its traffic, corresponding to the two peering points. ISPs typically behave selfishly and try to minimize their own costs, and send traffic to the closest peering point,

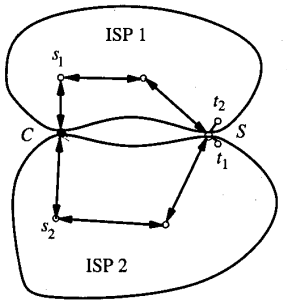


Figure 1.1. The ISP routing problem.

as the ISP with the destination node must route the traffic, no matter where it enters its domain. Peering point C is closer, using this peering point ISP 1 incurs a cost of 1 unit (in sending traffic along 1 edge), whereas if it uses the farther peering point S, it incurs a cost of 2.

Note that the farther peering point S is more directly on route to the destination t_1, and hence routing through S results in shorter overall path. The length of the path through C is 4 while through S is 2, as the destination is very close to S.

The situation described for ISP 1 routing traffic from s_1 to t_1 is in a way analogous to a prisoner's choices in the Prisoner's Dilemma: there are two choices, one is better from a selfish perspective ("confess" or route through peering point C), but hurts the other player. To make our routing game identical to the Prisoner's Dilemma, assume that symmetrically the 2nd ISP needs to send traffic from point s_2 in his domain to point t_2 in the 1st ISP's domain. The two choices of the two ISPs lead to a game with cost matrix identical to the matrix above with C corresponding to "confess" and S corresponding to remaining "silent."

1.1.2 The Tragedy of the Commons

In this book we will be most concerned with situations where many participants interact, and such situations are naturally modeled by games that involve many players: there are thousands of ISPs, and many millions of traffic streams to be routed. We will give two examples of such games, first a multiplayer version of the Prisoner's Dilemma that we will phrase in terms of a pollution game. Then we will discuss the well-known game of Tragedy of the Commons.

Example 1.3 (Pollution game) This game is the extension of Prisoner's Dilemma to the case of many players. The issues modeled by this game arise in many contexts; here we will discuss it in the context of pollution control. Assume that there are n countries in this game. For a simple model of this situation, assume that each country faces the choice of either passing legislation to control pollution or not. Assume that pollution control has a cost of 3 for the country, but each country that pollutes adds 1 to the cost of all countries (in terms of added

health costs, etc.). The cost of controlling pollution (which is 3) is considerably larger than the cost of 1 a country pays for being socially irresponsible.

Suppose that k countries choose not to control pollution. Clearly, the cost incurred by each of these countries is k. On the other hand, the cost incurred by the remaining $n-k$ countries is $k+3$ each, since they have to pay the added cost for their own pollution control. The only stable solution is the one in which no country controls pollution, having a cost of n for each country. In contrast, if they all had controlled pollution, the cost would have been only 3 for each country.

The games we have seen so far share the feature that there is a unique optimal "selfish" strategy for each player, independent of what other players do. No matter what strategy the opponent plays, each player is better off playing his or her selfish strategy. Next, we will see a game where the players' optimal selfish strategies depend on what the other players play.

Example 1.4 (Tragedy of the commons) We will describe this game in the context of sharing bandwidth. Suppose that n players each would like to have part of a shared resource. For example, each player wants to send information along a shared channel of known maximum capacity, say 1. In this game each player will have an infinite set of strategies, player i's strategy is to send x_i units of flow along the channel for some value $x_i \in [0, 1]$.

Assume that each player would like to have a large fraction of the bandwidth, but assume also that the quality of the channel deteriorates with the total bandwidth used. We will describe this game by a simple model, using a benefit or payoff function for each set of strategies. If the total bandwidth $\sum_j x_j$ exceeds the channel capacity, no player gets any benefit. If $\sum_j x_j < 1$ then the value for player i is $x_i(1 - \sum_j x_j)$. This models exactly the kind of trade-off we had in mind: the benefit for a player deteriorates as the total assigned bandwidth increases, but it increases with his own share (up to a point).

To understand what stable strategies are for a player, let us concentrate on player i, and assume that $t = \sum_{j \neq i} x_j < 1$ flow is sent by all other players. Now player i faces a simple optimization problem for selecting his flow amount: sending x flow results in a benefit of $x(1 - t - x)$. Using elementary calculus, we get that the optimal solution for player i is $x = (1 - t)/2$. A set of strategies is stable if all players are playing their optimal selfish strategy, given the strategies of all other players. For this case, this means that $x_i = (1 - \sum_{j \neq i} x_j)/2$ for all i, which has a unique solution in $x_i = 1/(n+1)$ for all i.

Why is this solution a tragedy? The total value of the solution is extremely low. The value for player i is $x_i(1 - \sum_{j \neq i} x_j) = 1/(n+1)^2$, and the sum of the values over all payers is then $n/(n+1)^2 \approx 1/n$. In contrast, if the total bandwidth used is $\sum_i x_i = 1/2$ then the total value is 1/4, approximately $n/4$ times bigger. In this game the n users sharing the common resource overuse it so that the total value of the shared resource decreases quite dramatically. The pollution game above has a similar effect,

where the common resource of the environment is overused by the n players increasing the cost from 3 to n for each players.

1.1.3 Coordination Games

In our next example, there will be multiple outcomes that can be stable. This game is an example of a so-called "coordination game." A simple coordination game involves two players choosing between two options, wanting to choose the same.

Example 1.5 (Battle of the sexes) Consider that two players, a boy and a girl, are deciding on how to spend their evening. They both consider two possibilities: going to a baseball game or going to a softball game. The boy prefers baseball and the girl prefers softball, but they both would like to spend the evening together rather than separately. Here we express the players' preferences again via payoffs (benefits) as follows.

Girl \ Boy	B	S
B	6, 5	1, 1
S	2, 2	5, 6

Clearly, the two solutions where the two players choose different games are not stable – in each case, either of the two players can improve their payoff by switching their action. On the other hand, the two remaining options, both attending the same game, whether it is softball or baseball, are both stable solutions; the girl prefers the first and the boy prefers the second.

Coordination games also arise naturally in many contexts. Here we give an example of a coordination game in the context of routing to avoid congestion. The good outcomes in the Battle of the Sexes were to attend the same game. In contrast, in the routing game, good outcomes will require routing on different paths to avoid congestion. Hence, this will be an "anticoordination" game.

Example 1.6 (Routing congestion game) Suppose that two traffic streams originate at proxy node O, and need to be routed to the rest of the network, as shown in Figure 1.2. Suppose that node O is connected to the rest of the network via connection points A and B, where A is a little closer than B. However, both connection points get easily congested, so sending both streams through the same connection point causes extra delay. Good outcomes in this game will be for the two players to "coordinate" and send their traffic through different connection points.

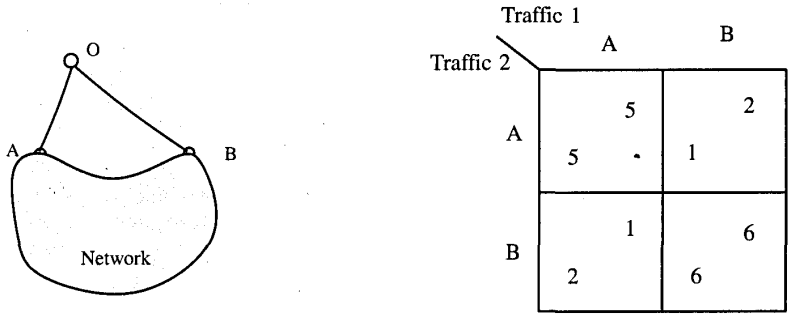


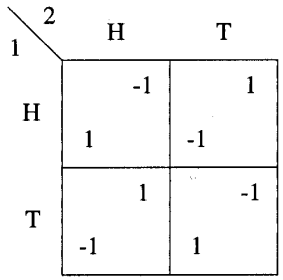
Figure 1.2. Routing to avoid congestion and the corresponding cost matrix.

We model this situation via a game with the two streams as players. Each player has two available strategies – routing through A or routing through B – leading to four total possibilities. The matrix of Figure 1.2 expresses the costs to the players in terms of delays depending on their routing choices.

1.1.4 Randomized (Mixed) Strategies

In the games we considered so far, there were outcomes that were stable in the sense that none of players would want to individually deviate from such an outcome. Not all games have such stable solutions, as illustrated by the following example.

Example 1.7 (Matching pennies) Two payers, each having a penny, are asked to choose from among two strategies – heads (H) and tails (T). The row player wins if the two pennies match, while the column player wins if they do not match, as shown by the following payoff matrix, where 1 indicates win and -1 indicated loss.



One can view this game as a variant of the routing congestion game in which the column player is interested in getting good service, hence would like the two players to choose different routes, while the row player is interested only in disrupting the column player's service by trying to choose the same route. It is easy to see that this game has

no stable solution. Instead, it seems best for the players to randomize in order to thwart the strategy of the other player.

1.2 Games, Strategies, Costs, and Payoffs

We have given examples of games and discussed costs, payoffs, and strategies in an informal way. Next we will define such a game more formally. The games we considered above were all *one-shot simultaneous move games*, in that all players simultaneously chose an action from their set of possible strategies.

1.2.1 Defining a Simultaneous Move Game

Formally, such a game consists of a set n of *players*, $\{1, 2, \ldots, n\}$. Each player i has his own *set of possible strategies*, say S_i. To play the game, each player i selects a strategy $s_i \in S_i$. We will use $s = (s_1, \ldots, s_n)$ to denote the *vector of strategies* selected by the players and $S = \times_i S_i$ to denote the set of all possible ways in which players can pick strategies.

The vector of strategies $s \in S$ selected by the players determine the outcome for each player; in general, the outcome will be different for different players. To specify the game, we need to give, for each player, a *preference ordering* on these outcomes by giving a complete, transitive, reflexive binary relation on the set of all strategy vectors S; given two elements of S, the relation for player i says which of these two outcomes i weakly prefers; we say that i *weakly prefers* S_1 to S_2 if i either prefers S_1 to S_2 or considers them as equally good outcomes. For example, in the matching pennies game the row player prefers strategy vectors in which the two pennies match and the column player prefers those in which the pennies do not match.

The simplest way to specify preferences is by assigning, for each player, a value to each outcome. In some games it will be natural to think of the values as the payoffs to players and in others as the costs incurred by players. We will denote these functions by $u_i : S \to \mathbf{R}$ and $c_i : S \to \mathbf{R}$, respectively. Clearly, costs and payoffs can be used interchangeably, since $u_i(s) = -c_i(s)$.

If we had defined, for each player i, u_i to be simply a function of s_i, the strategy chosen by player i, rather than s, the strategies chosen by all n players, then we would have obtained n independent optimization problems. Observe the crucial difference between this and a game – in a game, the payoff of each player depends not only on his own strategy but also on the strategies chosen by all other players.

1.2.2 Standard Form Games and Compactly Represented Games

To develop an algorithmic theory of games, we need to discuss how a game is specified. One option is to explicitly list all possible strategies, and the preferences or utilities of all players. Expressing games in this form with a cost or utility function is called the *standard form* or matrix form of a game. It is very convenient to define games in this way when there are only 2 players and the players have only a few strategies. We

have used this form in the previous section for defining the *Prisoner's Dilemma* and the *Battle of the Sexes*.

However, for most games we want to consider, this explicit representation is exponential sized in the natural description of the game (possibly bigger or even infinite). Most games we want to consider have many players, e.g., the many traffic streams or the many ISPs controlling such streams. (In fact, in Part III of this book, we will even encounter games with infinitely many players, modeling the limiting behavior as the number of players gets very large.) For an example, consider the pollution game from Subsection 1.1.2, where we have n players, each with two possible strategies. There are 2^n possible strategy vectors, so the explicit representation of the game requires assigning values to each of these 2^n strategies. The size of the input needed to describe the game is much smaller than 2^n, and so this explicit representation is exponentially larger than the description of the game.

Another reason that explicit representation of the payoffs can be exponentially large is that players can have exponentially many strategies in the natural size of the game. This happens in routing games, since the strategy space of each player consists of all possible paths from source to destination in the network. In the version of the *Tragedy of the Commons*, we discussed in Section 1.1.2 players have infinite strategy sets, since any bandwidth $x \in [0, 1]$ is a possible strategy.

Such exponential (and superexponential) descriptions can sometimes be avoided. For example, the payoff may depend on the number of players selecting a given strategy, rather than the exact subset (as was the case for the pollution game). The routing congestion game discussed in Chapter 18 provides another example, where the cost of a chosen path depends on the total traffic routed on each edge of the path. Another possibility for compact representation is when the payoff of a player may depend on the strategies chosen by only a few other players, not all participants. Games with such locality properties are discussed in detail in Chapter 7.

1.3 Basic Solution Concepts

In this section we will introduce basic solution concepts that can be used to study the kinds of games we described in the previous section. In particular, we will formalize the notion of stability that we informally used in discussing solutions to some of the games.

1.3.1 Dominant Strategy Solution

The *Prisoner's Dilemma* and the *Pollution Game* share a very special property: in each of these games, each player has a unique best strategy, independent of the strategies played by the other players. We say that a game has a *dominant strategy solution* if it has this property.

More formally, for a strategy vector $s \in S$ we use s_i to denote the strategy played by player i and s_{-i} to denote the $(n-1)$-dimensional vector of the strategies played by all other players. Recall that we used $u_i(s)$ to denote the utility incurred by player i. We will also use the notation $u_i(s_i, s_{-i})$ when it is more convenient. Using this notation,

a strategy vector $s \in S$ is a *dominant strategy solution*, if for each player i, and each alternate strategy vector $s' \in S$, we have that

$$u_i(s_i, s'_{-i}) \geq u_i(s'_i, s'_{-i}).$$

It is important to notice that a dominant strategy solution may not give an optimal payoff to any of the players. This was the case in both the *Prisoner's Dilemma* and the *Pollution Game*, where it is possible to improve the payoffs of all players simultaneously.

Having a single dominant strategy for each player is an extremely stringent requirement for a game and very few games satisfy it. On the other hand, *mechanism design*, the topic of Part II of this book, aims to design games that have dominant strategy solutions, and where this solution leads to a desirable outcome (either socially desirable, or desirable for the mechanism designer). We illustrate this, using the simple example of Vickrey auction.

1.3.2 Vickrey Auction: Designing Games with Dominant Strategy Solutions

Perhaps the most common situation in which we need to design a game is an auction. Suppose that we are faced with designing an auction to sell a valuable painting. To model this situation as a game, assume that each player (bidder) i has a value v_i for the painting. His value or payoff for not winning it is 0, and his payoff for winning it at a price of p is $v_i - p$. The strategy of each player is simply his bid. What is a good mechanism (or game) for selling this painting? Here we are considering single-shot games, so assume that each player is asked to state his bid for the painting in a sealed envelope, and we will decide who to award the painting to and for what price, based on the bids in the envelopes.

Perhaps the most straightforward auction would be to award the painting to the highest bidder and charge him his bid. This game does not have a dominant strategy solution. A player's best strategy (bid) depends on what he knows or assumes about the strategies of the other players. Deciding what value to bid seems like a hard problem, and may result in unpredictable behavior. See Section 1.6 for more discussion of a possible solution concept for this game.

Vickrey's mechanism, called *second price auction*, avoids these bidding problems. As before, the painting is awarded to the bidder with highest bid; however, the amount he is required to pay is the value of the second highest bid. This second price auction has the remarkable property that each player's dominant strategy is to report his true value as bid, independent of the strategies of the rest of the players! Observe that even if his true value happens to be very high, he is in no danger of overpaying if he reports it – if he wins, he will pay no more than the second highest bid.

Let us observe two more properties of the Vickrey auction. First, it leads to the desirable outcome of the painting being awarded to the bidder who values it most. Indeed, the larger goal of mechanism design is often to design mechanisms in which the selfish behavior of players leads to such a socially optimal outcome. For example, when the government auctions off goods, such as the wireless spectrum auctions, their

goal is typically not to make as large a profit as possible, but rather to get the spectrum in the hands of companies that have the best technology to offer to customers.

Another nice feature of a dominant strategy game, such as Vickrey auction, is that it is extremely simple for the players to play such a game, since each player's optimal strategy is independent of other players' choices. In fact, one can implement all dominant strategy games by simply asking all players for their valuation functions and letting the game designer "play" the game for them. This is called the *revelation principle* (see Chapter 9). (In this book, we will not consider the complex issue of how players arrive at their own valuation function.) Unfortunately, in many contexts the valuation function of a player can be very complex and direct revelation may lead to extensive, maybe even exponential, communication (see Chapter 11). Another problem with direct revelation mechanisms is that they assume the presence of a central trusted party. Chapter 8 shows how cryptographic techniques can help a group of players implement such a mechanism or game without a trusted party.

1.3.3 Pure Strategy Nash Equilibrium

Since games rarely possess dominant strategy solutions, we need to seek a less stringent and more widely applicable solution concept. A desirable game-theoretic solution is one in which individual players act in accordance with their incentives, maximizing their own payoff. This idea is best captured by the notion of a Nash equilibrium, which, despite its shortcomings (mentioned below), has emerged as the central solution concept in game theory, with extremely diverse applications. The Nash equilibrium captures the notion of a stable solution, discussed in Section 1.1 and used in the *Tragedy of the Commons* and the *Battle of the Sexes* – a solution from which no single player can individually improve his or her welfare by deviating.

A strategy vector $s \in S$ is said to be a *Nash equilibrium* if for all players i and each alternate strategy $s_i' \in S_i$, we have that

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

In other words, no player i can change his chosen strategy from s_i to s_i' and thereby improve his payoff, assuming that all other players stick to the strategies they have chosen in s. Observe that such a solution is self-enforcing in the sense that once the players are playing such a solution, it is in every player's best interest to stick to his or her strategy.

Clearly, a dominant strategy solution is a Nash equilibrium. Moreover, if the solution is strictly dominating (i.e., switching to it always strictly improves the outcome), it is also the unique Nash equilibrium. However, Nash equilibria may not be unique. For example, coordination games have multiple equilibria.

We already know that Nash equilibria may not be optimal for the players, since dominant strategy solutions are Nash equilibria. For games with multiple Nash equilibria, different equilibria can have (widely) different payoffs for the players. For example, by a small change to the payoff matrix, we can modify the *Battle of the Sexes* game so that it still has two stable solutions (the ones in which both players go to the same activity); however, both players derive a much higher utility from one of these solutions. In

Part III of this book we will look more carefully at the quality of the best and worst equilibria in different games.

The existence of multiple Nash equilibria makes this solution concept less convincing as a prediction of what players will do: which equilibrium should we expect them to play? And with independent play, how will they know which equilibrium they are supposed to coordinate on? But at least a Nash equilibrium is stable – once proposed, the players do not want to individually deviate.

1.3.4 Mixed Strategy Nash Equilibria

The Nash equilibria we have considered so far are called *pure strategy* equilibria, since each player deterministically plays his chosen strategy. As illustrated by the *Matching Pennies* game, a game need not possess any pure strategy Nash equilibria. However, if in the matching pennies game, the players are allowed to randomize and each player picks each of his two strategies with probability $1/2$, then we obtain a stable solution in a sense. The reason is that the expected payoff of each player now is 0 and neither player can improve on this by choosing a different randomization.

When players select strategies at random, we need to understand how they evaluate the random outcome. Would a player prefer a choice that leads to a small positive utility with high probability, but with a small probability leads to a large negative utility? Or, is it better to have a small loss with high probability, and a large gain with small probability? For the notion of mixed Nash equilibrium, we will assume that players are risk-neutral; that is, they act to maximize the *expected payoff*.

To define such randomized strategies formally, let us enhance the choices of players so each one can pick a probability distribution over his set of possible strategies; such a choice is called a *mixed strategy*. We assume that players independently select strategies using the probability distribution. The independent random choices of players leads to a probability distribution of strategy vectors s. Nash (1951) proved that under this extension, every game with a finite number of players, each having a finite set of strategies, has a Nash equilibrium.

Theorem 1.8 *Any game with a finite set of players and finite set of strategies has a Nash equilibrium of mixed strategies.*

This theorem will be further discussed and proved for the two player case in Chapter 2. An important special case of 2 player games is zero-sum games, games in which the gain of one player is exactly the loss of the other player. Nash equilibria for these games will be further discussed in Section 1.4.

1.3.5 Games with No Nash Equilibria

Both assumptions in the theorem about the finite set of players and finite strategy sets are important: games with an infinite number of players, or games with a finite number of players who have access to an infinite strategy set may not have Nash equilibria. A simple example of this arises in the following pricing game.

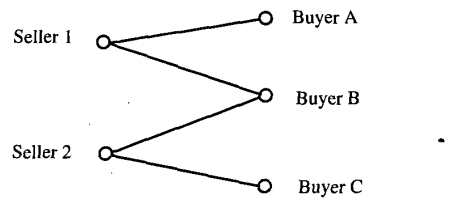


Figure 1.3. Sellers 1 and 2 are selling identical products to buyers A, B, and C.

Example 1.9 (Pricing game) Suppose two players sell a product to three possible buyers, as shown in Figure 1.3. Each buyer wants to buy one unit of the product.

Buyers A and C have access to one seller only, namely 1 and 2, respectively. However, buyer B can buy the product from any of the two sellers. All three buyers have a budget of 1, or have maximum value 1 for the item, i.e., will not buy the product if the price is above 1. The sellers play a pricing game – they each name a price p_i in the interval $[0, 1]$. Buyers A and C buy from sellers 1 and 2, respectively. On the other hand, B buys from the cheaper seller. To fully specify the game, we have to set a rule for breaking ties. Let us say that if both sellers have the same price, B buys from seller 1. For simplicity, we assume no production costs, so the income of a seller is the sum of the prices at which they sold goods.

Now, one strategy for each seller is to set a price of $p_i = 1$, and guarantee an income of 1 from the buyer who does not have a choice. Alternatively, they can also try to compete for buyer B. However, by the rules of this game they are not allowed to price-discriminate; i.e., they cannot sell the product to the two buyers at different prices. In this game, each player has uncountably many available strategies, i.e., all numbers in the interval $[0, 1]$. It turns out that this game does not have a Nash equilibrium, even if players are allowed to use mixed strategies.

To see that no pure strategy equilibrium exists, note that if $p_1 > 1/2$, player 2 will slightly undercut the price, set it at $1/2 < p_2 < p_1$, and have income of more than 1, and then in turn player 1 will undercut player 2, etc. So we cannot have $p_1 > 1/2$ in an equilibrium. If $p_1 \leq 1/2$, the unique best response for player 2 is to set $p_2 = 1$. But then player 1 will increase his price, so $p_1 \leq 1/2$ also does not lead to an equilibrium. It is a bit harder to argue that there is also no mixed strategy equilibrium in this game.

1.3.6 Correlated Equilibrium

A further relaxation of the Nash equilibrium notion was introduced by Aumann (1959), called correlated equilibrium. The following simple example nicely illustrates this notion.

Example 1.10 (Traffic light) The game we consider is when two players drive up to the same intersection at the same time. If both attempt to cross, the result is a fatal traffic accident. The game can be modeled by a payoff matrix where crossing successfully has a payoff of 1, not crossing pays 0, while an accident costs -100.

1 \ 2	Cross	Stop
Cross	-100 -100	0 1
Stop	1 0	0 0

This game has three Nash equilibria: two correspond to letting only one car cross, the third is a mixed equilibrium where both players cross with an extremely small probability $\epsilon = 1/101$, and with ϵ^2 probability they crash. The first two equilibria have a payoff of 1. The last one is more fair, but has low expected payoff (≈ 0.0001), and also has a positive chance of a car crash.

In a Nash equilibrium, players choose their strategies independently. In contrast, in a correlated equilibrium a coordinator can choose strategies for both players; however, the chosen strategies have to be stable: we require that the each player find it in his or her interest to follow the recommended strategy. For example, in a correlated equilibrium the coordinator can randomly let one of the two players cross with any probability. The player who is told to stop has 0 payoff, but he knows that attempting to cross will cause a traffic accident.

Correlated equilibria will be discussed in detail in Section 2.7. Formally, this notion assumes an external correlation device, such as a trusted game coordinator, or some other physical source. A correlated equilibrium is a probability distribution over strategy vectors $s \in \times_i S_i$. Let $p(s)$ denote the probability of strategy vector s, where we will also use the notation $p(s) = p(s_i, s_{-i})$ when talking about a player i. The distribution is a correlated equilibrium if for all players i and all strategies $s_i, s_i' \in S_i$, we have the inequality

$$\sum_{s_{-i}} p(s_i, s_{-i}) u_i(s_i, s_{-i}) \geq \sum_{s_{-i}} p(s_i, s_{-i}) u_i(s_i', s_{-i}).$$

In words, if player i receives a suggested strategy s_i, the expected profit of the player cannot be increased by switching to a different strategy $s_i' \in S_i$. Nash equilibria are special cases of correlated equilibria, where the distribution over S is the product of independent distributions for each player. However, correlation allows a richer set of equilibria as we will see in Section 2.7.

1.4 Finding Equilibria and Learning in Games

In this section we consider two closely related issues: how easy is it to find an equilibrium, and does "natural game play" lead the players to an equilibrium? Ideally, a perfect solution concept is one which is computationally easy to find, and also easy to find by players playing independently.

1.4.1 Complexity of Finding Equilibria

The complexity of finding Nash and correlated equilibria will be discussed in detail in Chapters 2 and 3. Here we give a short overview. We then discuss two-player zero-sum games in more detail and show that for such games a Nash equilibrium can be found efficiently using linear programming. It turns out that even general two-player games have a character different from that of games with three or more players. For example, two-player games where payoffs are rational numbers always admit a solution with rational probabilities, and this is not true for games with three or more players. Games with two players will be discussed in greater detail in Chapter 3.

We will discuss the complexity of finding Nash equilibrium in Chapter 2. NP-completeness, the "standard" way of establishing intractability of individual problems, does not seem to be the right tool for studying the complexity of Nash equilibria. Instead, we will use PPAD-completeness (see Chapter 2 for the definition). The problem of finding a Nash equilibrium is PPAD-complete even for two-player games in standard form.

In contrast, we will see in Section 2.7 that correlated equilibria are computationally easier. Correlated equilibria form a convex set and hence can be found in polynomial time for games defined explicitly via their payoff matrices, and finding a correlated equilibrium is polynomially solvable even in many compactly represented games. However, finding an "optimal" correlated equilibrium is computationally hard in many natural classes of compactly represented games.

1.4.2 Two-Person Zero-Sum Games

Here we consider two-player zero-sum games in more detail. A two-player game is a *zero-sum game* if the sum of the payoffs of the two players is zero for any choice of strategies. For such games it is enough to give the payoffs of the row player. Let A be the matrix of these payoffs, representing the winnings of the row player and the loss of the column player.

Recall from Theorem 1.8 that a Nash equilibrium of mixed strategies always exists. We will use this fact to show that an equilibrium can be found using linear programming. Consider a pair of probability distributions p^* and q^* for the row and column players that form a Nash equilibrium. The expected value paid by the column player to the row player can be expressed as $v^* = p^*Aq^*$ (if we think of p^* as a row vector and q^* as a column vector).

A Nash equilibrium has the property that even if the players know the strategies played by the other players (the probability distribution they are using), they cannot be better off by deviating. With this in mind, consider a strategy p for the row player. The expected payoffs for different strategies of the column player will be pA. Once

p is known, the column player will want to minimize his loss, and play strategies that correspond to the minimum entries in pA. So the best publicly announced strategy for the row player is to maximize this minimum value. This best public strategy can be found by solving the following linear program:

$$\begin{aligned} v_r &= \max v \\ p &\geq 0 \\ \sum_i p_i &= 1 \\ (pA)_j &\geq v \text{ for all } j, \end{aligned}$$

where we use $(pA)_j$ to denote the jth entry of the vector pA. The optimum value v_r is the row player's maximum safe value, the maximum value he or she can guarantee to win by playing a mixed strategy p that will be known to the column player.

How does v_r and the Nash value v^* compare? Clearly $v_r \leq v^*$, since the row player, can guarantee to win v_r, so must win at least this much in any equilibrium. On the other hand, an equilibrium is a strategy that is stable even if known to the opponent, so it must be the case that the column player is in fact selecting the columns with minimum value p^*A, so we must have $v^* \leq v_r$, and hence $v_r = v^*$.

Similarly, we can set up the analogous linear program to get the value v_c, the column player's minimum safe value, the minimum loss the column player can guarantee by playing a mixed strategy q that will be known to the row player:

$$\begin{aligned} v_c &= \min v \\ q &\geq 0 \\ \sum_j q_j &= 1 \\ (Aq)_i &\leq v \text{ for all } i. \end{aligned}$$

where we use $(Aq)_i$ to denote the ith entry of the vector Aq. We can argue that $v^* = v_c$ also holds. Hence we get that $v_c = v_r$, the row players' maximum guaranteed win is the same as the column players' minimum guaranteed loss. This will imply that the optimal solutions to this pair of linear programs form a Nash equilibrium.

Theorem 1.11 *Optimum solutions for the above pair of linear programs give probability distributions that form a Nash equilibrium of the two-person zero-sum game.*

PROOF Let p and q denote optimum solutions to the two linear programs. We argued above that $v_c = v_r$. If the players play this pair of strategies, then the row player cannot increase his win, as the column player is guaranteed by his strategy not to lose more than v_c. Similarly, the column player cannot decrease his loss, as the row player is guaranteed to win v_r by his strategy. So the pair of strategies is at equilibrium. □

Readers more familiar with linear programming will notice that the two linear programs above are duals of each other. We established that $v_r = v_c$ using the existence

of a Nash equilibrium from Theorem 1.8. Linear programming duality also implies that the two values v_r and v_c are equal. Once we know the values are equal, the proof of Theorem 1.4.2 shows that the optimal solutions form a Nash equilibrium, so linear programming duality yields a proof that a Nash equilibrium exists in the special case of zero-sum two-person games.

1.4.3 Best Response and Learning in Games

It would be desirable for a solution concept to satisfy a stronger condition than simply being polynomial computable: it should be the case that natural game playing strategies quickly lead players to either find the equilibrium or at least converge to an equilibrium in the limit.

Maybe the most natural "game playing" strategy is the following "best response." Consider a strategy vector s, and a player i. Using the strategy vector s player i gets the value or utility $u_i(s)$. Changing the strategy s_i to some other strategy $s_i' \in S_i$ the player can change his utility to $u_i(s_i', s_{-i})$, assuming that all other players stick to their strategies in s_{-i}. We say that a change from strategy s_i to s_i' is an *improving response for player i* if $u_i(s_i', s_{-i}) > u_i(s)$ and *best response* if s_i' maximizes the players' utility $\max_{s_i' \in S_i} u_i(s_i', s_{-i})$. Playing a game by repeatedly allowing some player to make an improving or a best response move is perhaps the most natural game play.

In some games, such as the *Prisoner's Dilemma* or the *Coordination Game*, this dynamic leads the players to a Nash equilibrium in a few steps. In the *Tragedy of the Commons* the players will not reach the equilibrium in a finite number of steps, but the strategy vector will converge to the equilibrium. In other games, the play may cycle, and not converge. A simple example is matching pennies, where the payers will cycle through the 4 possible strategy vectors if they alternate making best response moves. While this game play does not find a pure equilibrium (as none exists) in some sense we can still say that best response converges to the equilibrium: the average payoff for the two players converges to 0, which is the payoff at equilibrium; and even the frequencies at which the 4 possible strategy vectors are played converge to the probabilities in equilibrium (1/4 each).

Results about the outcome of such game playing strategies will be discussed in Chapter 4. We will see that best response behavior is not strong enough to guarantee convergence in most games. Instead, we will consider improving response type "learning" strategies that react to the frequencies played so far, rather than just to the current game play. We will show that in the special case of 2-player zero-sum games such natural game playing does converge to a Nash equilibrium. In general, even learning strategies do not converge to Nash equilibria, instead they converge to the larger region of correlated equilibria.

1.5 Refinement of Nash: Games with Turns and Subgame Perfect Equilibrium

Nash equilibria has become the central solution concept in game theory, despite its shortcomings, such as the existence of multiple equilibria. Since the emergence of this

concept in the 1950s, there have been many refinements considered that address the selection of the "right" equilibrium concept. Here we will consider one such refinement for games with turns.

Many games have multiple turns of moves. Card games or board games all have turns, but games modeling many economic situations also have this form: a service provider sets up a basic service (turn 1) and then users decide to use the service or decide not to (turn 2).

How does Nash equilibrium extend to games with turns? We can reduce such games to simultaneous move games by having each player select a "full strategy" up front, rather than having them select moves one at a time. By a "full strategy" we mean a strategy for each turn, as a function of the state of the game. One issue with such strategies is that they tend to become rather large: a full strategy for chess would state the next move for any possible sequence of previous moves. This is a huge set in the natural description of the game in terms of the rules of chess. Games with turns is another example of a compactly represented game. We will see more on how to work with this type of compactly represented games in Chapter 3.

Here our focus is to point out that in this context the notion of Nash equilibrium seems a bit weak. To see why, consider the following simple game.

Example 1.12 (Ultimatum game) Assume that a seller S is trying to sell a good to buyer B. Assume that the interaction has two steps: first seller S offers a price p, and then buyer B reacts to the price. We assume the seller has no value for the good, his payoff is p if the sale occurs, and 0 otherwise. The buyer has a value v for the good, so his payoff is $v - p$ if he buys, and 0 if he does not. Here we are considering a full information game in which seller S is aware of the buyer's value v, and hence we expect that the seller offers price p just under v, and the buyer buys. (Ignore for now the issue of what happens if the price is exactly v.)

This game allows the first player to lead, and collect (almost) all the profit. This game is known as the *ultimatum game* when two players S and B need to divide up v amount of money. The game allows the first player S to make an "ultimatum" (in the form of a price in our context) on how to divide up the money.

To think about this game as a one-shot simultaneous move game, we need to think of the buyer's strategy as a function or the offered price. A natural strategy is to "buy if the price is under v." This is indeed an equilibrium of the game, but the game has many other equilibria. The buyer can also have the strategy that he will buy only if the price p is at most some smaller value $m \leq v$. This seems bad at first (why leave the $v - p$ profit on the table if the price is in the range $m < p < v$), but assuming that the buyer uses this alternate strategy, the seller's best move is to offer price $p = m$, as otherwise he makes no profit. This pair of strategies is also a Nash equilibrium for any value m.

The notion of subgame perfect equilibrium formalizes the idea that the alternate buyer strategy of buying only at $p \leq m$ is unnatural. By thinking of the game as a simultaneous move game, the difference between the two players in terms of the order of moves, is diminished. The notion of *subgame perfect* Nash equilibrium has been introduced to strengthen the concept of Nash, and make the order of turns part of the definition. The idea is to require that the strategy played is Nash, even after any prefix

of the game is already played. We will see more about subgame perfect equilibrium as well as games with turns in Chapters 3 and 19.

1.6 Nash Equilibrium without Full Information: Bayesian Games

So far we talked about equilibrium concepts in full information games, where all players know the utilities and strategies of all other players. When players have limited information, we need to consider strategies that are only based on the available information, and find the best strategy for the player, given all his or her available information. Such games will be discussed in more detail in Section 9.6.

One source of limited information can come from not knowing properties and preferences of other players, and hence not knowing what strategies they will select. It is easiest to understand this issue by considering a game of cards, such as bridge. In such a game the players have information about the probability distribution of the other players' cards, but do not know exactly what cards they have. A similar information model can also be used to model many other situations. We illustrate this by the Bayesian first price auction game.

Example 1.13 (Bayesian First Price Auction) Recall the first price auction: all players state a bid, and the winner is the player with maximum bid, and has to pay his bid value as the price. What are optimal strategies for players in this auction? If the valuations of all players are common knowledge, then the player with maximum valuation would state the second valuation as his bid, and win the auction at the same (or slightly bigger) price as in the second price auction. But how should players bid if they do not know all other players' valuations? Naturally, their bids will now depend on their beliefs about the values and knowledge of all other players.

Here we consider the simple setup where players get their valuations from independent probability distributions, and these distributions are public knowledge. How should player i bid knowing his own valuation v_i, and the distribution of the valuation of the other players? Such games are referred to as Bayesian games, and are discussed in Section 9.6. For example, it is shown there that the unique Nash equilibrium in the case when player valuations come from independent and identical distributions is a nice analog of the second price auction: player i, whose own valuation is v_i, should bid the expected second valuation conditioned on v_i being the maximum valuation.

1.7 Cooperative Games

The games we talked about so far are all non-cooperative games – we assumed that individual players act selfishly, deviate alone from a proposed solution, if it is in their interest, and do not themselves coordinate their moves in groups. Cooperative game theory is concerned with situations when groups of players coordinate their actions.

First, in Section 1.7.1 we define the concept of strong Nash equilibrium, a notion extending the Nash equilibrium concept to cooperative situations.

Then we consider games with *transferable utility*, i.e., games where a player with increased utility has the ability to compensate some other player with decreased utility. When considering games with transferable utility the main concern is to develop solution concepts for formalizing fair ways of sharing a value or dividing up a cost in a cooperative environment. There have been many different notions of fairness proposed. In Section 1.7.2 we will briefly review two of them. We refer the reader to Chapter 15 for a more in-depth discussion of these two and other concepts.

1.7.1 Strong Nash Equilibrium

The closest notion from cooperative game theory to our discussion thus far is the concept of strong Nash equilibrium introduced by Aumann (1974). Consider a game and a proposed solution, a strategy for each player. In a cooperative game we assume that some group A of players can change their strategies jointly, assuming that they all benefit. Here we are assuming that the game has nontransferable utility, which means that in order for a coalition to be happy, we need to make sure that the utility of each member is increasing (or at least is not decreasing).

We say that a vector of strategies forms a *strong Nash equilibrium* if no subset A of players has a way to simultaneously change their strategies, improving each of the participant's welfare. More formally, for a strategy vector s and a set of players A let s_A denote the vector of strategies of the players in A and let s_{-A} denote the vector of strategies of the players not in A. We will also use $u_i(s_A, s_{-A})$ for the utility for player i in the strategy s. We say that in a strategy vector s a subset A of players has a *joint deviation* if there are alternate strategies $s'_i \in S_i$ for $i \in A$ forming a vector s'_A, such that $u_i(s) \leq u_i(s'_A, s_{-A})$ for all $i \in A$, and for at least one player in A the inequality is strict. A strategy vector s is *strong Nash* if no subset A has a joint deviation.

The concept of strong Nash is very appealing, for strong Nash equilibria have a very strong reinforcing property. One problem with this concept is that very few games have such equilibria. A nice example of a game with strong Nash equilibria is the game version of the stable marriage problem where boys and girls form pairs based on preference lists for the other sex. For a proposed matching, the natural notion of deviation for this game is a pair deviating (a couple who prefer each other to their current partners). This game will be reviewed in detail in Chapter 10. Chapter 19 considers network formation games, and will discuss another class of games where coalitions of size 2 (pairs) are the natural units causing instability of a solution.

1.7.2 Fair Division and Costsharing: Transferable Utility Games

When utility is transferable, we can think of the game as dividing some value or sharing a cost between a set of players. The goal of this branch of game theory is to understand what is a fair way to divide value or cost between a set of participants. We assume that there is a set N of n participants, or players, and each subset A of players is associated with a cost $c(A)$ (or value $v(A)$). We think of $c(A)$ as a cost associated with serving the group A of players, so $c(N)$ is the cost of serving all N players. The problem is to

divide this cost $c(N)$ among the n players in a "fair" way. (In case of dividing a value $v(A)$, we think of $v(A)$ as the value that the set A can generate by itself.)

A *cost-sharing* for the total cost $c(N)$ is a set of cost-shares x_i for each player $i \in N$. We assume that cost-sharing needs to be *budget balanced*; i.e., we require that $\sum_{i \in N} x_i = c(N)$. One of the key solution concepts in this area is that of a *core*. We say that the cost-sharing is in the core if no subset of players would decrease their shares by breaking away from the whole set. More formally, we say that the cost-share vector c is *in the core* if $\sum_{i \in A} x_i \leq c(A)$ for all sets A. A violation of this inequality precisely corresponds to a set A of players who can benefit by breaking away.

Given a notion of fair sharing, such as the core, there are a number of important questions one can ask. Given a cost function c, we want to know whether there is a cost-sharing x that is in the core. In Chapter 15 we will see that there are nice ways of characterizing problems that have a nonempty core. We will also be concerned with the complexity of finding a cost-sharing in the core, and deciding whether the core is nonempty. The computational complexity of determining whether the core is empty has been extensively studied for many fundamental games. If the core is empty or finding a solution in the core is an intractable problem, one can consider a relaxed version of this notion in which subsets of players secede only if they make substantial gains over being in the whole set N. We will discuss these ideas in Chapter 15.

Here we briefly review a very different proposal for what is a "fair" way to share cost, the Shapley value. One advantage of the Shapley value is that it always exists. However, it may not be in the core, even for games that have nonempty core.

Example 1.14 (Shapley Value) Shapley value is based on evaluating the marginal cost of each player. If we order the player set N as $1, \ldots, n$ and use the notation that $N_i = \{1, \ldots, i\}$ then the marginal cost of player i is $c(N_i) - c(N_{i-1})$. Of course, this marginal cost depends on the order the players are considered. The *Shapley value* assigns cost-share x_i to player i that is the expected value of this marginal cost over a random order of the players.

In Chapter 15 we will show that the Shapley value can be characterized as the unique cost-sharing scheme satisfying a number of different sets of axioms.

1.8 Markets and Their Algorithmic Issues

Some of the most crucial regulatory functions within a capitalistic economy, such as ensuring stability, efficiency, and fairness, are relegated to pricing mechanisms, with very little intervention. It is for this reason that general equilibrium theory, which studied equilibrium pricing, occupied a central place within mathematical economics.

From our viewpoint, a shortcoming of this theory is that it is mostly a nonalgorithmic theory. With the emergence of numerous new markets on the Internet and the availability of massive computational power for running these markets in a centralized or distributed manner, there is a need for a new, inherently algorithmic theory of market equilibria. Such algorithms can also help understand the repercussions to existing

prices, production, and consumption caused by technological advances, introduction of new goods, or changes to the tax structure. Chapters 5 and 6 summarize recent work along these lines.

Central to ensuring stability of prices is that there be parity between the demand and supply of goods. When there is only one good in the market, such an *equilibrium price* is easy to determine – it is simply the price at which the demand and supply curves intersect. If the price deviates from the equilibrium price, either demand exceeds supply or vice versa, and the resulting market forces tend to push the price back to the equilibrium point. Perhaps the most celebrated result in general equilibrium theory, due to Arrow and Debreu (1954), shows the existence of equilibrium prices in a very general model of the economy with multiple goods and agents.

It turns out that equilibria for several fundamental market models can be captured as optimal solutions to certain nonlinear convex programs. As a result, two algorithmic approaches present themselves – combinatorial algorithms for solving these convex programs and convex programming based approaches. These are covered in Chapters 5 and 6, respectively.

1.8.1 An Algorithm for a Simple Market

In this section, we will give a gist of the models and algorithms studied using a very simple market model. Consider a market consisting of a set A of divisible *goods* and a set B of *buyers*. We are specified for each buyer i, the amount $m_i \in \mathbf{Z}^+$ of money she possesses, and for each good j, the amount $a_j \in \mathbf{Z}^+$ of this good. Each buyer i *has access to* only a subset, say $S_i \subseteq A$ of the goods. She is indifferent between goods in S_i, but is interested in maximizing the total amount of goods obtained. An example of such a situation is when identical goods are sold in different markets and each buyer has access to only a subset of the markets; such a model is studied in Chapter 7. Without loss of generality we may assume that $m_i \neq 0$, $a_j \neq 0$, for each buyer i, $S_i \neq \emptyset$, and for each good j, there is a buyer i such that $j \in S_i$.

Once the prices $p_1, \ldots, p_n$ of the goods are fixed, a buyer i is only *interested in* the cheapest goods in S_i, say $S_i' \subseteq S_i$. Any allocation of goods from S_i' that exhausts her money will constitute her *optimal basket of goods* at these prices.

Prices are said to be *market clearing or equilibrium prices* if there is a way to assign to each buyer an optimal basket of goods so that there is no surplus or deficiency of any of the goods i.e., demand equals supply. It turns out that equilibrium prices are unique for this market; see Chapter 5 for a proof in a more general setting.

We will need the following notations and definitions. Define a bipartite graph $G = (A, B, E)$ on vertex sets A and B as shown on Figure 1.4. The edge (j, i) connects a good j to a buyer i such that $j \in S_i$. Because of the assumptions made, each vertex in G has non zero degree. For $S \subseteq A$ of goods, let $a(S)$ denote the total amount of goods in S, i.e., $a(S) = \sum_{j \in S} a_j$. For a subset $T \subseteq B$ of buyers, let $m(T) = \sum_{i \in T} m_i$ denote the total money possessed by buyers in T.

The algorithm given below is iterative and always assigns uniform prices to all goods currently under consideration. For a set S of goods, let $\Gamma(S)$ denote the set of buyers who are interested in goods in S; $\Gamma(S) = \{i \in B \mid S_i \cap S \neq \emptyset\}$. This is the

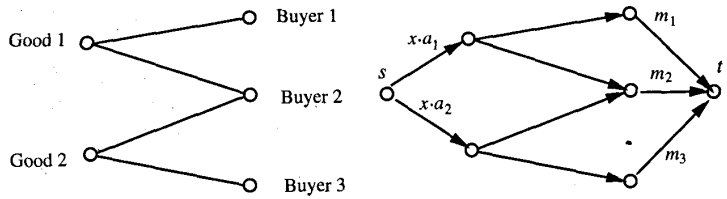


Figure 1.4. The graph G on the left and the corresponding max-flow network N.

neighborhood of S in G. We say that a uniform price x is *feasible* if

$$\forall S \subseteq A, \quad x \cdot a(S) \leq m(\Gamma(S)),$$

i.e., the total cost of S is at most the total money possessed by buyers interested in goods in S. With respect to a feasible x, we will say that set $S \subseteq A$ is *tight* if $x \cdot a(S) = m(\Gamma(S))$. The importance of feasibility is established by the following lemma.

Lemma 1.15 *A uniform price of x on all goods is feasible if and only if all goods can be sold in such a way that each buyer gets goods that she is interested in.*

PROOF One direction is straightforward. If there is a subset $S \subseteq A$ such that $x \cdot a(S) > m(\Gamma(S))$ then goods in S cannot all be sold at price x since buyers interested in these goods simply do not have enough money.

To prove the other direction, we will use network N (see Figure 1.4) obtained from the bipartite graph G for computing allocations of goods to buyers. Direct the edges of G from A to B and assign a capacity of infinity to all these edges. Introduce source vertex s and a directed edge from s to each vertex $j \in A$ with a capacity of $x \cdot a_j$. Introduce sink vertex t and a directed edge from each vertex $i \in B$ to t with a capacity of m_i.

Clearly, a way of selling all goods corresponds to a feasible flow in N that saturates all edges going out of s. We will show that if x is feasible, then such a flow exists in N. By the max-flow min-cut theorem, if no such flow exists, then the minimum cut must have capacity smaller than $x \cdot a(A)$. Let S be the set of goods on the s-side of a minimum cut. Since edges (j, i) for goods $j \in S$ have infinite capacity, $\Gamma(S)$ must also be on the s-side of this cut. Therefore, the capacity of this cut is at least $x \cdot a(A - S) + m(\Gamma(S))$. If this is less than $x \cdot a(A)$ then $x \cdot a(S) > m(\Gamma(S))$, thereby contradicting the feasibility of x. □

If with respect to a feasible x, a set S is tight, then on selling all goods in S, the money of buyers in $\Gamma(S)$ will be fully spent. Therefore, x constitutes market clearing prices for goods in S. The idea is to look for such a set S, allocate goods in S to $\Gamma(S)$, and recurse on the remaining goods and buyers.

The algorithm starts with $x = 0$, which is clearly feasible, and raises x continuously, always maintaining its feasibility. It stops when a nonempty set goes tight. Let x^* be

the smallest value of x at which this happens and let S^* be the maximal tight set (it is easy to see that S^* must be unique).

We need to give procedures for finding x^* and S^*. Observe that x^* is the largest value of x at which $(s, A \cup B \cup t)$ remains a min-cut in N. Therefore, x^* can be computed via a binary search. After computing x^*, compute the set of nodes that can reach t in the residual graph of this flow. This set, say W, is the t-side of the (unique) maximal min-cut in N at $x = x^*$. Then, $S^* = A - W$, the set of goods on the s side of this cut.

At prices x^*, buyers in $\Gamma(S^*)$ will have no surplus money left and increasing x any more will lead to infeasibility. At this point, the algorithm fixes the prices of goods in S^* at x^*. It computes a max-flow in N for $x = x^*$, as suggested by Lemma 1.15. This flow gives an allocation of goods in S^* to buyers in $\Gamma(S^*)$, which fully spends all the money $m(\Gamma(S^*))$. The same flow also shows that x^* is feasible for the problem for goods $A - S^*$ and buyers $B - \Gamma(S^*)$.

In the next iteration, the algorithm removes S^* and $\Gamma(S^*)$, initializes the prices of the goods in $A - S^*$ to x^*, and raises prices until a new set goes tight. The algorithm continues in this manner, iteratively finding prices of sets of goods as they go tight. It terminates when all goods have been assigned prices.

Lemma 1.16 *The value x^* is feasible for the problem restricted to goods in $A - S^*$ and buyers in $B - \Gamma(S^*)$. Furthermore, in the subgraph of G induced on $A - S^*$ and $B - \Gamma(S^*)$, all vertices have nonzero degree.*

PROOF In the max-flow computed in N for $x = x^*$, the flow going through nodes in S^* completely uses up the capacity of edges from $\Gamma(S^*)$ to t. Therefore, all the flow going through nodes in $A - S^*$ must exit via nodes in $B - \Gamma(S^*)$. Now, the first claim follows from Lemma 1.15. Furthermore, a good $j \in A - S^*$ must have nonzero degree to $B - \Gamma(S^*)$. Finally, since each buyer $i \in (B - \Gamma(S^*))$ has nonzero degree in G and has no edges to S^*, it must have nonzero degree to $A - S^*$. □

Theorem 1.17 *The above-stated algorithm computes equilibrium prices and allocations in polynomial time.*

PROOF At termination, all goods are assigned prices and are therefore fully sold. By the second claim in Lemma 1.16, when the algorithm terminates, each buyer must be in the neighborhood of one of the tight sets found and therefore must be allocated goods in return for her money. We need to show that each buyer gets her optimal bundle of goods. Let S^* be the first tight set found by the algorithm. Since S^* was a maximal tight set at x^*, prices must strictly rise before a new set goes tight in the second iteration. Therefore, prices are monotone increasing across iterations and all goods in $A - S^*$ are assigned higher prices than x^*. Since each buyer $i \in \Gamma(S^*)$ is allocated goods from S^* only, she was given an optimal bundle. Now, the claim follows by induction.

Clearly, the algorithm will execute at most $|A|$ iterations. The time taken for one iteration is dominated by the time required for computing x^* and S^*. Observe that $x^* = m(\Gamma(S^*))/a(S^*)$, i.e., its numerator and denominator are polynomial

sized integers. Therefore binary search for finding x^* will take polynomial time. □

Acknowledgments

We would like to thank Christos Papadimitriou, Bernhard von Stengel, Tim Roughgarden, and Rakesh Vohra for their extremely valuable critiques and suggestions on an early draft of this chapter. We also thank Ramesh Johari and Tim Roughgarden for suggesting the ISP routing version of the Prisoners' Dilemma in Section 1.1.

Bibliography

K.K. Arrow and G. Debreu. Existence of an equilibrium for competitive economy. *Econometrica*, 22:265–290, 1954.

R.J. Aumann. Acceptable points in general cooperative n-person games. In: *Contributions to the Theory of Games IV*, Princeton University Press, 1959.

R.J. Aumann. Subjectivity an correlation in randomized strategies. *J. Math. Econ.*, 1:67–96, 1974.

D. Fudenberg and J. Tirole. *Game Theory*, MIT Press, 1991.

D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*, Oxford Press, 1995.

D. Monderer and L. Shapley. Potential games. *Games and Economic Behavior* 14:124–143, 1996.

J. Nash. Noncooperative games. *Annals of Mathematics*, 54:289–295, 1951.

M. Osborne and A. Rubinstein. *A Course in Game Theory*, MIT Press, 1994.

Exercises

1.1 Give a finite algorithm for finding a Nash equilibrium for a game with two players defined by a game matrix. Your algorithm may run in exponential time.

1.2 Consider a two-player game given in matrix form where each player has n strategies. Assume that the payoffs for each player are in the range [0, 1] and are selected independently and uniformly at random. Show that the probability that this random game has a pure (deterministic) Nash equilibrium approaches $1 - 1/e$ as n goes to infinity. You may use the fact that $\lim(1 - 1/n)^n = 1/e$ as n goes to infinity.

1.3 We have seen that finding a Nash in a two-person zero-sum game is significantly easier than general two-person games. Now consider a three-person zero-sum game, that is, a game in which the rewards of the three players always sums to zero. Show that finding a Nash equilibrium in such games is at least as hard as that in general two-person games.

1.4 Consider an n person game in which each player has only two strategies. This game has 2^n possible outcomes (for the 2^n ways the n players can play), therefore the game in matrix form is exponentially large. To circumvent this, in Chapter 7 we will consider a special class of games called graphical games. The idea is that the value (or payoff) of a player can depend only on a subset of players. We will define a dependence graph G, whose nodes are the players, and an edge between two players i and j represents the fact that the payoff of player i depends on the strategy of player j or vice versa. Thus, if node i has k neighbors, then its payoff depends only on its own strategy and the strategies of its k neighbors.

Consider a game where the players have 2 pure strategies each and assume that the graph G is a tree with maximum degree 3. Give a polynomial time algorithm to decide if such a game has a pure Nash equilibrium. (Recall that there are 2^n possible pure strategy vectors, yet your algorithm must run in time polynomial in n.)

1.5 Consider an n player game in which each player has 2 strategies. For this problem, think of the strategies as "on" and "off." For example, the strategy can be either to participate or not to participate in some event. Further more, assume that the game is symmetric, in that all players have the same payoff functions, and that the payoff for a player depends only on the strategy of the player and the number of people playing strategy "on." So the game is defined by $2n$ values: $u_{on}(k)$ and $u_{off}(k)$, which denote the payoff for playing the "on" and "off" strategies, assuming that k of the other players chose to play "on" for $k = 0, \ldots, n-1$.

Give a polynomial time algorithm to find a correlated equilibrium for such a game. Note that the input to this problem consists of the $2n$ numbers above. As usual, polynomial means polynomial in this input length. You may use the fact that linear programming is solvable in polynomial time.

1.6 Consider a 2-person game in matrix form. Assume that both players have n pure strategies. In a Nash equilibrium a player may be required to play a mixed strategy that gives nonzero probability to all (or almost all) of his pure strategies. Strategies that mix between so many pure options are hard to play, and also hard to understand. The goal of this problem is to show that one can reach an almost perfect Nash equilibrium by playing strategies that only choose between a few of the options.

We will use p^j to be the probability distribution for player j, so p_i^j is the probability that player j will use his ith pure strategy. The support of a mixed strategy p^j for player j is the set $S^j = \{i : p_i^j > 0\}$, i.e., the set of different pure strategies that are used with nonzero probability. We will be interested in solutions where each player has a strategy with small support.

For a given $\epsilon > 0$, we will say that a set of mixed strategies p^1, p^2 is ϵ-approximate Nash if for both players $j = 1$ or 2, and all other strategies $\bar{p}^j$ for this player, the expected payoff for player j using strategy $\bar{p}^j$ is at most ϵM more than his expected payoff using strategy p^j, where M is the maximum payoff.

Show that for any fixed $\epsilon > 0$ and any 2-player game with all nonnegative payoffs, there is an ϵ-approximate Nash equilibrium such that both players play the following simple kind of mixed strategy. For each player j, the strategy selects a subset $\hat{S}_j$ of at most $O(\log n)$ of player j's pure strategies, and makes player j select one of the strategies in $\hat{S}_j$ uniformly at random. The set $\hat{S}_j$ may be a multiset, i.e., may contain the same pure strategy more than once such a strategy is more likely to be selected by the random choice). The constant in the $O(.)$ notation may depend on the parameter ϵ.

Hint: Consider any mixed Nash strategy with possibly large support, and try to simplify the support by selecting the subsets $\hat{S}_j$ for the two players based on this Nash equilibrium.

1.7 The classical Bertrand game is the following. Assume that n companies, which produce the same product, are competing for customers. If each company i has a production level of q_i, there will be $q = \sum_i q_i$ units of the product on the market.

Now, demand for this product depends on the price and if q units are on the market, price will settle so that all q units are sold. Assume that we are given a "demand-price curve" $p(d)$, which gives the price at which all d units can be sold. Assume that $p(d)$ is a monotone decreasing, differentiable function of d. With this definition, the income of the firm i will be $q_i\, p(q)$. Assume that production is very cheap and each firm will produce to maximize its income.

(a) Show that the total income for a monopolistic firm, can be arbitrarily higher than the total income of many different firms sharing the same market. Hint: this is true for almost all price curves; you may want to use, e.g., $p(d) = 1 - d$.

(b) Assume that $p(d)$ is twice differentiable, monotone decreasing, and $p''(d) \leq 0$. Show that the monopolistic income is at most n times the total income of the n competing companies.

1.8 Let V denote a set of n agents, labeled $1, 2, \ldots, n$. Let 0 denote the root node and for any subset $S \subseteq V$, S^+ denote the set $S \cup \{0\}$. Let $G = (V^+, E)$ be a complete, undirected graph with edge costs $c : E \rightarrow \mathbf{Z}^+$ which satisfy the triangle inequality. For a subset $S \subseteq V$, let $c(S)$ denote the cost of a minimum spanning tree in the subgraph of G induced on S^+. The spanning tree game asks for a budget balanced cost-sharing method for minimum spanning tree that lies on the core.

Consider the following cost-sharing method for sharing the cost of building a minimum spanning tree in G among the n agents. Find any minimum spanning tree, say T, and root it at vertex 0. Define the cost of agent i to be the cost of the first edge on the unique path from i to 0 in T. Clearly, this cost-sharing method is budget balanced; i.e., the total cost retrieved from the n agents is precisely the cost of a minimum spanning tree in G. Show that this cost-sharing method is in the core, i.e., for any subset $S \subseteq V$, the total cost charged to agents in S is at most the cost they would incur if they were to directly connect to the root, i.e., $c(S)$.

CHAPTER 2

The Complexity of Finding Nash Equilibria

Christos H. Papadimitriou

Abstract

Computing a Nash equilibrium, given a game in normal form, is a fundamental problem for Algorithmic Game Theory. The problem is essentially combinatorial, and in the case of two players it can be solved by a pivoting technique called the Lemke–Howson algorithm, which however is exponential in the worst case. We outline the recent proof that finding a Nash equilibrium is complete for the complexity class PPAD, even in the case of two players; this is evidence that the problem is intractable. We also introduce several variants of succinctly representable games, a genre important in terms of both applications and computational considerations, and discuss algorithms for correlated equilibria, a more relaxed equilibrium concept.

2.1 Introduction

Nash's theorem – stating that every finite game has a mixed Nash equilibrium (Nash, 1951) – is a very reassuring fact: Any game can, in principle, reach a quiescent state, one in which no player has an incentive to change his or her behavior. One question arises immediately: Can this state be reached in practice? Is there an *efficient algorithm* for finding the equilibrium that is guaranteed to exist? This is the question explored in this chapter.

But why should we be interested in the issue of computational complexity in connection to Nash equilibria? After all, a Nash equilibrium is above all a conceptual tool, a prediction about rational strategic behavior by agents in situations of conflict – a context that is completely devoid of computation.

We believe that this matter of computational complexity is one of central importance here, and indeed that the algorithmic point of view has much to contribute to the debate of economists about solution concepts. The reason is simple: If an equilibrium concept is not efficiently computable, much of its credibility as a prediction of the behavior of rational agents is lost – after all, there is no clear reason why a group of agents cannot be simulated by a machine. Efficient computability is an important modeling