PART ONE

Computing in Games

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,

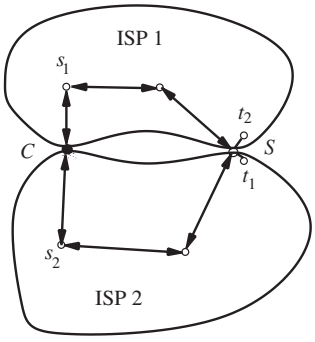


Figure 1.1. The ISP routing problem.

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.

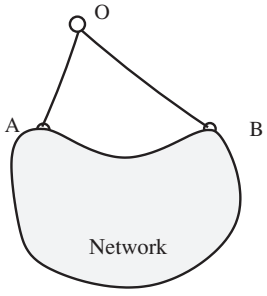
Example 1.5 (Battle of the sexes) Consider that two players, a boy and a girl, are deciding on how to spend their evening. They both consider two possibilities: going to a baseball game or going to a softball game. The boy prefers baseball and the girl prefers softball, but they both would like to spend 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.



Traffic 2 \ Traffic 1	A	B
A	5, 5	2, 1
B	1, 2	6, 6

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.

1 \ 2	H	T
H	-1, 1	1, -1
T	1, -1	-1, 1

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