# Lecture Notes: Algorithmic Game Theory

Palash Dey
Indian Institute of Technology, Kharagpur
palash.dey@cse.iitkgp.ac.in

October 17, 2022

Statutory warning: This is a draft version and may contain errors. If you find any error, please send an email to the author.

**Notation:**

- ▷ $\mathbb{N} = \{0, 1, 2, ...\}$ : the set of natural numbers
- ▷ $\mathbb{R}$: the set of real numbers
- ▷ $\mathbb{Z}$: the set of integers
- ▷ For a set $\mathcal{X}$, we denote its power set by $2^{\mathcal{X}}$.
- ▷ For an integer $\ell$, we denote the set $\{1, 2, \ldots, \ell\}$ by $[\ell]$.

# Contents

# Part I

# Game Theory

# Chapter 1

# Introduction to Non-cooperative Game Theory

Game theory is the field in mathematics which studies "games." Intuitively speaking, a game is any "system" where there are multiple parties (called players of the game), the "outcome" depends on the actions that individual parties take, and different players derive different utilities from the outcome. Let us see some examples of game to have a better feel of the subject. Our first example is arguably the most important one for the students.

**Example 1.0.1** (Grading Game)**.** *Consider the "Algorithmic Game Theory" class in IIT Kharagpur. Suppose the instructor announces that the grading policy will be as follows — the top 10% of students get EX grade, next 20% get A, etc. Could you see the game that this grading policy induces? The players are the students in the class. The outcome of the system is the function which maps students to the grades that they get. Observe that the outcome of the system depends on the actions of all the players (also called action profile or strategy profile).*

Our next example is the famous prisoner's dilemma.

**Example 1.0.2** (Prisoner's Dilemma)**.** *Suppose two people have committed a crime and subsequently been caught by the police. However, the cops do not have enough evidence to make a strong case against them. The police interrogates them separately and simultaneously, say in two different rooms. Both the prisoners have the following options — either confess (let us call this action $\mathcal{C}$) or not confess (let us call this action $\mathcal{NC}$). The prisoners also know the following — if both the prisoners confess (that is, play $\mathcal{C}$), then the police has a strong case and both the prisoners get* 5 *years of jail terms; if both the prisoners do not confess (play $\mathcal{NC}$), then the police do not have a strong case and both the prisoners get* 2 *years of jail terms; if only one prisoner confesses (plays $\mathcal{C}$) and the other does not confess (plays $\mathcal{NC}$), then the police again have a strong case and the prisoner who confessed gets* 1 *year of jail term (less punishment*

*for cooperating with the police) whereas the other prisoners get* 10 *years of jail terms (more punishment for non-cooperation with the police).*

Our next example is called a congestion game which models the congestion on road networks, Internet traffic, etc.

**Example 1.0.3** (Congestion Games)**.** *Suppose every people in a city wants to go from some point* A *in the city to some other point* B *and each person independently decides the path that they wish to take. The speed with which people can commute on any road is inversely proportional to the number of people using that road. We can again observe that the situation induces a game where the commuters are the players, the paths taken by them are their actions, and their journey times are the outcome.*

The above way of modeling situations in terms of games is so fundamental that we can keep on giving more and more examples of games from almost every field that we can think about. We now formally define games. The most common form of games is the *normal form games* aka *strategic form games*.

## 1.1 Normal Form Game

**Normal Form Game**

**Definition 1.1.** A normal form game $\Gamma$ is defined as a tuple $\langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$ where

- $N$ is the set of players
- $S_i$ is the set of strategies for the player $i \in N$
- $u_i : \bigtimes_{i\in N} S_i \longrightarrow \mathbb{R}$ is the utility function for the player $i \in N$

Let us write the prisoners' dilemma game in the normal form.

**Example 1.1.1** (Prisoners' Dilemma in Normal Form)**.** *The prisoners' dilemma game in the normal form can be written as follows.*

- *The set of players* $(N) : \{1, 2\}$
- *The set of strategies:* $S_i = \{\mathcal{C}, \mathcal{NC}\}$ *for every* $i \in [2]$
- *Payoff matrix:*

| | | *Player* 2 | |
|---|---|---|---|
| | | $\mathcal{C}$ | $\mathcal{NC}$ |
| *Player* 1 | $\mathcal{C}$ | $(-5,-5)$ | $(-1,-10)$ |
| | $\mathcal{NC}$ | $(-10,-1)$ | $(-2,-2)$ |

Okay, suppose we have modeled some scenario as a game. Then what? What are the questions we would like to answer? Two most important questions that we would like to answer are the following — (i) as a player, how to play a game, i.e. which strategy to play? (ii) can we predict what will be the "outcome" which is equivalent to asking who will play what strategy? This is indeed the case for many applications like how to bid in an auction, predicting the volume of traffic that will use a newly built bridge, predicting the amount of revenue that a Government is likely to have by auctioning say 5G spectrum, predicting the congestion of a road network on Independence Day, etc. A little thought towards answering these kinds of prediction related questions convinces us that one cannot hope to make any "reliable" prediction without making some assumptions on how players behave.

## 1.2 Big Assumptions of Game Theory

If not mentioned otherwise, we in game theory make the following behavioral assumptions about the players.

### 1.2.1 Utility

We assume that each player has a utility function aka payoff function which maps from the set of outcomes (which is the set of tuples, called *strategy profile*, specifying which player has played what strategy) to the set $\mathbb{R}$ of real numbers denoting the amount of happiness that the player derives from some outcome. Observe that the utility of any player not only depends on the strategy that he/she plays but also depends on the strategy that other players play.

### 1.2.2 Rationality (aka Selfishness)

We assume that the players are rational – players play the game to maximize his/her utility and nothing else.

### 1.2.3 Intelligence

We assume that the players have infinite computational power. For example, we assume that the players are competent enough to make any level of reasoning, can argue like any game theorist to compute the best strategies for him/her under any number of circumstances.

### 1.2.4 Common Knowledge

In the context of games, we say that an information $\mathcal{I}$ is common knowledge if $\mathcal{I}$ is known to every player, every player knows that every player knows $\mathcal{I}$, every player knows that every player knows that every player knows $\mathcal{I}$,... $(\text{every player knows that})^n$ every player knows $\mathcal{I}$ for every natural number $n$. We assume that the set of players, the set of strategies for every player, the utility function of every player, the fact that every player is rational and intelligent is common knowledge among the players.

Let us understand the notion of common knowledge better through a puzzle. Suppose in an island, there were 50 green-eyed people and 50 black-eyed people. The green-eyed people believe that they are cursed. If anyone knows that his eye is of green color, he will publicly suicide at 12'o clock at the noon. There is no reflecting surface on the island. They were also not supposed to talk about the color of anyone's eye. One day a foreigner visited the island and, without knowing the rules, makes a remark that there exists at least

one person on the island whose eye is green. Could you see why all the 50 green-eyed people will suicide together on the $50^{th}$ day since the foreigner remarked? Do you see that although the information that "there exists at least one green-eyed people in the island" was known to everyone before foreigner's visit, it was NOT common knowledge? It became common knowledge only after the foreigner announces this publicly.

## 1.3 Examples of Normal Form Games

We now see some examples of games in their normal form. Our first example is a kind of coordination game where two players, say a husband (player 1) and a wife (player 2), need to independently decide among two options, say going for shopping (option A) and going to a cricket stadium to watch a match (option B). If the players play different strategies (that is they are not together), then both of them are unhappy and both derive 0 utility. If they play the same strategies, then both are happy and derive positive utility. However, if they both end up playing A, then the utility of player 1 is more than that of player 2; if both play B, then the utility of player 2 is more than player 1.

**Example 1.3.1** (Battle of Sexes)**.** *The game battle of sexes can be written in the normal form as follows.*

- ▷ *The set of players* $(N) : \{1, 2\}$
- ▷ *The set of strategies:* $S_i = \{A, B\}$ *for every* $i \in [2]$
- ▷ *Payoff matrix:*

| | | *Player* 2 | |
|---|---|---|---|
| | | A | B |
| *Player* 1 | A | $(1, 2)$ | $(0, 0)$ |
| | B | $(0, 0)$ | $(2, 1)$ |

Our next example is another coordination game. Suppose two companies (player 1 and player 2) want to decide on the product (between product A and product B) for which they will manufacture parts. Assume that the parts that the companies produce are complementary. So, if the players end up playing different strategies, then they both lose and derive 0 utility. On the other hand, if the players play the same strategy, then they both gain positive utility. However, the exact amount of utility that the players derive may not be the same for both the products (maybe because of some externalities like the existence of other companies in the market).

**Example 1.3.2** (Coordination Games)**.** *The coordination game in the normal form can be written as follows.*

- ▷ *The set of players* $(N) : \{1, 2\}$
- ▷ *The set of strategies:* $S_i = \{A, B\}$ *for every* $i \in [2]$

▷ *Payoff matrix:*

| | | *Player* 2 | |
|---|---|---|---|
| | | A | B |
| *Player* 1 | A | $(10, 10)$ | $(0, 0)$ |
| | B | $(0, 0)$ | $(1, 1)$ |

We next look at another kind of game which is called strictly competitive games or zero-sum games. Zero-sum games always involve two players. In this kind of game, the utility of the row player in any strategy profile is the minus of the utility of the column player in that strategy profile. Below we see two zero-sum games, namely matching pennies and rock-paper-scissor, in normal form.

**Example 1.3.3** (Matching Pennies)**.** *The game matching pennies can be written in the normal form as follows.*

▷ *The set of players* $(N) : \{1, 2\}$

▷ *The set of strategies:* $S_i = \{A, B\}$ *for every* $i \in [2]$

▷ *Payoff matrix:*

| | | *Player* 2 | |
|---|---|---|---|
| | | A | B |
| *Player* 1 | A | $(1, -1)$ | $(-1, 1)$ |
| | B | $(-1, 1)$ | $(1, -1)$ |

**Example 1.3.4** (Rock-Paper-Scissors)**.** *The game Rock-Paper-Scissors can be written in the normal form as follows.*

▷ *The set of players* $(N) : \{1, 2\}$

▷ *The set of strategies:* $S_i = \{$*Rock, Paper, Scissors*$\}$ *for every* $i \in [2]$

▷ *Payoff matrix:*

| | | *Player* 2 | | |
|---|---|---|---|---|
| | | *Rock* | *Paper* | *Scissors* |
| *Player* 1 | *Rock* | $(0, 0)$ | $(-1, 1)$ | $(1, -1)$ |
| | *Paper* | $(1, -1)$ | $(0, 0)$ | $(-1, 1)$ |
| | *Scissors* | $(-1, 1)$ | $(1, -1)$ | $(0, 0)$ |

Next, let us look at the *tragedy of common* game which involves more than two players. We have $n$ players for some natural number $n$. Each of them has two strategies – to build a factory (strategy 1) or to not build a factory (strategy 0). If someone builds a factory, then he/she derives 1 unit of utility from that factory. However, factories harm society (because they pollute the environment say). Each factory built causes 5

units of harm but that harm is shared by all players.

**Example 1.3.5** (Tragedy of the Commons)**.** *The tragedy of the commons game in the normal form can be written as follows.*

- ▷ *The set of players* $(N) : \{1, 2, \ldots, n\}$ *(we denote this set by* $[n]$*)*
- ▷ *The set of strategies:* $S_i = \{0, 1\}$ *for every* $i \in [n]$
- ▷ *Utility:*

$$u_i(s_1, \ldots, s_i, \ldots, s_n) = s_i - \left[\frac{5(s_1 + \cdots + s_n)}{n}\right]$$

Our next example is auctions. Auction (for buying) is any game where we have a set of $n$ sellers each having a valuation of the item. Each seller submits a bid (which thus forms the strategy set) for selling the item. Then an allocation function is applied to the profile of bids to decide who eventually sells the item to the buyer and a payment function is applied to decide how much the seller gets paid. Hence one can think of any number of auctions by simply choosing different allocation and payment functions. We now look at the normal form representation of the first price auction and the second price auction which by far the most used auctions around.

**Example 1.3.6** (First Price and Second Price Auction)**.** *The first and second price auctions in the normal form can be written as follows.*

- ▷ *The set of players* $(N) : \{1, 2, \ldots, n\}$ *(this set corresponds to the sellers of the item)*
- ▷ *The set of strategies:* $S_i = \mathbb{R}_{\geqslant 0}$ *for every* $i \in [n]$
- ▷ *Each player (seller)* $i \in N$ *has a valuation* $v_i$ *for the item*
- ▷ *Allocation function:* $a : \times_{i \in N} S_i \longrightarrow \mathbb{R}^N$ *defined as* $a\left((s_i)_{i \in N}\right) = \left((a_i)_{i \in N}\right)$ *where* $a_i = 1$ *(that is, player* $i$ *wins the auction) if* $s_i \leqslant s_j$ *for every* $j \in N$ *and* $s_i < s_j$ *for every* $j \in \{1, 2, \ldots, i-1\}$*; otherwise* $a_i = 0$*.*
- ▷ *Payment function:* $p : \times_{i \in N} S_i \longrightarrow \mathbb{R}^N$ *defined as* $p\left((s_i)_{i \in N}\right) = \left((p_i)_{i \in N}\right)$ *where*
  - *For the first price auction:* $p_i = s_i$ *if* $a_i = 1$ *and* $p_i = 0$ *otherwise*
  - *For the second price auction:* $p_i = \min_{j \in N \setminus \{i\}} s_j$ *if* $a_i = 1$ *and* $p_i = 0$ *otherwise*
- ▷ *Utility function:*

$$u_i(s_1, \ldots, s_i, \ldots, s_n) = a_i(p_i - v_i)$$

# Chapter 2

# Solution Concepts of Non-cooperative Game Theory

We now look at various game-theoretic solution concepts – theories for predicting the outcome of games. Our first solution concept is the dominant strategy equilibrium.

## 2.1 Dominant Strategy Equilibrium

Let's try to predict the outcome of the prisoners' dilemma game. We claim that both the players should acknowledge that they have committed the crime; that is they should play the strategy $\mathcal{C}$ (of course we assume that both the players are rational, intelligent, and the game is a common knowledge). Let's argue why we predict so. Suppose the column player plays the strategy $\mathcal{C}$. Then we observe that playing $\mathcal{C}$ gives the row player more utility than $\mathcal{NC}$. Now let us assume that the column player plays $\mathcal{NC}$. In this case also, we see that the strategy $\mathcal{C}$ gives the row player more utility than $\mathcal{NC}$. Hence, we observe that irrespective of the strategy that the column player plays, playing $\mathcal{C}$ yields strictly more utility than playing $\mathcal{NC}$. We call such a strategy a *strongly dominant strategy*; that is $\mathcal{C}$ is a strongly dominant strategy for the row player. By analogous argument, we see that $\mathcal{C}$ is also a strongly dominant strategy for the column player. We call a profile consisting of strongly dominant strategies only a *strongly dominant strategy equilibrium*. Hence $(\mathcal{C}, \mathcal{C})$ forms a strongly dominant strategy equilibrium. We now define these notions formally.

**Strongly Dominant Strategy and Strongly Dominant Strategy Equilibrium (SDSE)**

**Definition 2.1.** Given a game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, a strategy $s_i^* \in S_i$ for some player $i \in N$ is called a strongly dominant strategy for the player $i$ if the following holds.

$$u_i(s_i^*, s_{-i}) > u_i(s_i', s_{-i}) \text{ for every } s_i' \in S_i \setminus \{s_i^*\}^a$$

We call a strategy profile $(s_i^*)_{i\in N}$ a strongly dominant strategy equilibrium of the game $\Gamma$ if $s_i^*$ is a strongly dominant strategy for the player $i$ for every $i \in N$.

[a] We often use the shorthand $(s_i, s_{-i})$ to denote the strategy profile $(s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots)$.

One can easily verify that strongly dominant strategy equilibrium, if it exists, is unique in any game. The requirements for the strongly dominant strategy and strongly dominant strategy equilibrium can be relaxed a little to define what is called a weakly dominant strategy and weakly dominant strategy equilibrium.

**Weakly Dominant Strategy and Weakly Dominant Strategy Equilibrium (WDSE or DSE)**

**Definition 2.2.** Given a game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, a strategy $s_i^* \in S_i$ for some player $i \in N$ is called a weakly dominant strategy for the player $i$ if the following holds.

$$u_i(s_i^*, s_{-i}) \geqslant u_i(s_i, s_{-i}) \text{ for every } s_i \in S_i \text{ and}$$

$$\text{for every } s_i' \in S_i \setminus \{s_i^*\}, \text{there exists } s_{-i} \in S_{-i} \text{such that } u_i(s_i^*, s_{-i}) > u_i(s_i', s_{-i})$$

We call a strategy profile $(s_i^*)_{i\in N}$ a weakly dominant strategy equilibrium of the game $\Gamma$ if $s_i^*$ is a weakly dominant strategy for the player $i$ for every $i \in N$.

**Very Weakly Dominant Strategy and Very Weakly Dominant Strategy Equilibrium**

**Definition 2.3.** Given a game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, a strategy $s_i^* \in S_i$ for some player $i \in N$ is called a very weakly dominant strategy for the player $i$ if the following holds.

$$u_i(s_i^*, s_{-i}) \geqslant u_i(s_i, s_{-i}) \text{ for every } s_i \in S_i$$

We call a strategy profile $(s_i^*)_{i\in N}$ a very weakly dominant strategy equilibrium of the game $\Gamma$ if $s_i^*$ is a very weakly dominant strategy for the player $i$ for every $i \in N$.

We now prove the first theorem in our course – bidding valuations form a weakly dominant strategy equilibrium for the second price buying auction. In a buying auction, we have one buyer and $n$ sellers.

**Weakly Dominant Strategy Equilibrium of Second Price Auction**

**Theorem 2.1.** Bidding valuations form a weakly dominant strategy equilibrium for the second price buying auction.

*Proof of Theorem 2.1.* To prove that bidding valuations is a weakly dominant strategy equilibrium, we need to show that, for every player $i \in N$, the strategy $s_i^* = v_i$ for the player $i$ is a weakly dominant strategy for the player $i$. Let $s_{-i} \in S_{-i}$ be any strategy profile of the players other than the player $i$. We consider the following 2 cases. Let $\theta_i = \min_{j\in N\setminus\{i\}} s_j$ be the minimum bid of any player other than $i$.

**Case I: Player $i$ wins the auction.** We have $u_i(s_i^*, s_{-i}) = \theta_i - v_i$ which is at least 0 since we have $\theta_i \leqslant s_i^* = v_i$. Let $s_i' \in S_i$ be any other strategy for the player $i$. If the player $i$ continues to win the auction by playing $s_i'$, then we have $u_i(s_i', s_{-i}) = \theta_i - v_i = u_i(s_i^*, s_{-i})$. On the other hand, if the player $i$ loses the auction by playing $s_i'$, then we have $u_i(s_i', s_{-i}) = 0 \leqslant \theta_i - v_i = u_i(s_i^*, s_{-i})$.

**Case II: Player $i$ does not win the auction.** We have $u_i(s_i^*, s_{-i}) = 0$. If the player $i$ continues to lose the auction, then we have $u_i(s_i', s_{-i}) = 0 = u_i(s_i^*, s_{-i})$. Suppose now that the player $i$ wins the auction by playing the strategy $s_i'$. If there exists a player $j \in \{1, 2, \ldots, i-1\}$ such that $s_j = \theta_i = s_i^*$, then for player $i$ to win the auction, we have $s_i' > s_i^*$ and thus $u_i(s_i', s_{-i}) = \theta_i - \nu_i = 0 = u_i(s_i^*, s_{-i})$. If for every player $j \in \{1, 2, \ldots, i-1\}$ we have $s_j > s_i^*$, then we have $\theta_i < s_i^*$. Now for player $i$ to win the auction by playing the strategy $s_i'$, we have $s_i' \geqslant \theta_i$. Thus we have $u_i(s_i', s_{-i}) = \theta_i - \nu_i < s_i^* - \nu_i \leqslant 0 = u_i(s_i^*, s_{-i})$.

Finally, let $s_i \in S_i \setminus \{s_i^*\}$ be any other strategy. If $s_i > \nu_i$, then in the profile where every other player bids $\frac{s_i+\nu_i}{2}$, then bidding $s_i$ gives player $i$ a utility of 0 (since player $i$ loses) where as bidding $\nu_i$ gives player $i$ a utility of $\frac{s_i-\nu_i}{2}$ (player $i$ wins and gets $\frac{s_i+\nu_i}{2}$). Similarly, if $s_i < \nu_i$, then in the profile where every other player bids $\frac{s_i+\nu_i}{2}$, then bidding $s_i$ gives player $i$ a utility of $\frac{s_i-\nu_i}{2}$ (player $i$ wins and receives $\frac{s_i+\nu_i}{2}$) which is negative where as bidding $\nu_i$ gives player $i$ a utility of 0. Hence $s_i^* = \nu_i$ is a weakly dominant strategy for the player $i$ and thus $(\nu_i)_{i \in N}$ forms a weakly dominant strategy equilibrium. □

It is easy to verify that bidding valuations do not form an SDSE. Can you prove that bidding valuations forms *unique* weakly dominant strategy equilibrium in the second price auction?

## 2.2 Nash Equilibrium

We can observe that many interesting games, for example, Battle of Sexes, coordination game, matching pennies, rock-paper-scissor do not have any weakly dominant strategy equilibrium. We relax the requirement of weakly dominant strategy equilibrium substantially to define what is known as pure strategy Nash equilibrium. A strategy profile forms a Nash equilibrium in a game if *unilateral deviation* does not improve the utility of any player. More formally, pure strategy Nash equilibrium of any game is defined as follows.

**Pure Strategy Nash Equilibrium (PSNE)**

**Definition 2.4.** Given a game $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$, we call a strategy profile $(s_i^*)_{i \in N}$ a pure strategy Nash equilibrium of the game $\Gamma$ if the following holds.

$$\text{For every } i \in N, u_i(s_i^*, s_{-i}^*) \geqslant u_i(s_i', s_{-i}^*) \text{ for every } s_i' \in S_i$$

It follows immediately from the definition of PSNE is that every WDSE is also a PSNE. Now coming back to our example of games, one can easily verify that the profiles $(A, A)$ and $(B, B)$ both form a pure strategy Nash equilibrium for the Battle of sexes and the coordination games. However, both the matching pennies and the rock-paper-scissors games do not have any pure strategy Nash equilibrium. To extend the requirement of pure strategy Nash equilibrium, we now allow the players to play according to some probability distribution over their strategies.[1] We call probability distributions over strategy sets mixed strategies. We denote the set of mixed strategies (probability distributions) over a strategy set $S$ by $\Delta(S)$. Since we now allow players to play according to some mixed strategy, we need to extend the utility functions to take mixed strategies as input. Given a mixed strategy profile $(\sigma_i)_{i \in N}$, the utility of player $i \in N$ from this profile is defined as

[1] Why we did not relax the requirements of dominant strategy equilibrium by allowing mixed strategies as we do to define mixed strategy Nash equilibrium?

follows. Let $N = \{1, 2, \ldots, n\}$.

$$u_i\left((\sigma_i)_{i\in N}\right) = \sum_{s_1\in S_1}\sum_{s_2\in S_2}\cdots\sum_{s_n\in S_n}\prod_{i=1}^{n}\sigma_i(s_i)u_i\left((s_i)_{i\in N}\right)$$

**Mixed Strategy Nash Equilibrium (MSNE)**

**Definition 2.5.** Given a game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, we call a mixed strategy profile $(\sigma_i^*)_{i\in N} \in \times_{i\in N}\Delta(S_i)$ a mixed strategy Nash equilibrium of the game $\Gamma$ if the following holds.

$$\text{For every } i \in N, u_i(\sigma_i^*, \sigma_{-i}^*) \geqslant u_i(\sigma_i', \sigma_{-i}^*) \text{ for every } \sigma_i' \in \Delta(S_i)$$

It is not immediately clear how to check whether a given mixed strategy profile $(\sigma_i)_{i\in N}$ forms an MSNE of a given game. The following characterization of MSNE is computationally appealing whose proof basically follows from the averaging principle (which says that average of a set of numbers is at most the maximum).

**Characterization of MSNE (the indifference principle)**

**Theorem 2.2.** Given a game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, a mixed strategy profile $(\sigma_i^*)_{i\in N} \in \times_{i\in N}\Delta(S_i)$ forms an MSNE of the game $\Gamma$ if and only if we have the following for every player $i \in N$.

$$\text{For every strategy } s_i \in S_i, \sigma_i^*(s_i) \neq 0 \implies u_i(s_i, \sigma_{-i}^*) \geqslant u_i(s_i', \sigma_{-i}^*) \text{ for every strategy } s_i' \in S_i$$

*Proof of Theorem 2.2.* **(If part).** Suppose the mixed strategy profile $(\sigma_i^*)_{i\in N}$ satisfies the equivalent condition. We need to prove that $(\sigma_i^*)_{i\in N}$ forms a MSNE for the game. Let $i \in N$ be any player and $\sigma_i \in \Delta(S_i)$ be any mixed strategy of player $i$. Then we have the following.

$$\begin{aligned}
u_i(\sigma_i, \sigma_{-i}^*) &= \sum_{s_i\in S_i}\sum_{s_{-i}\in S_{-i}}\sigma_i(s_i)\prod_{j\in N\setminus\{i\}}\sigma_j^*(s_j)u_i\left((s_j)_{j\in N}\right)\\
&= \sum_{s_i\in S_i}\sigma_i(s_i)u_i(s_i, \sigma_{-i}^*)\\
&\leqslant \sum_{s_i\in S_i}\sigma_i^*(s_i)u_i(s_i, \sigma_{-i}^*)\\
&= u_i(\sigma_i^*, \sigma_{-i}^*)
\end{aligned}$$

The third line follows from the fact that $\sigma_i^*$ places probability mass only on those strategies $s_i \in S_i$ which maximizes utility of the player $i$ given others are playing according to $\sigma_{-i}^*$.

**(Only if part).** Suppose the mixed strategy profile $(\sigma_i^*)_{i\in N}$ forms an MSNE for the game. For the sake of arriving to a contradiction, let us assume that there exists a player $i \in N$ and a strategy $s_i \in S_i$ for which the condition does not hold. That is, we have $\sigma_i^*(s_i) \neq 0$ and there exists a strategy $s_i' \in S_i$ such that $u_i(s_i', \sigma_{-i}^*) > u_i(s_i, \sigma_{-i}^*)$. Without loss of generality, let us assume that we have $u_i(s_i', \sigma_{-i}^*) \geqslant u_i(s_i'', \sigma_{-i}^*)$ for every $s_i'' \in S_i$. Then we have the following.

$$u_i(\sigma_i^*, \sigma_{-i}^*) = \sum_{s_i\in S_i}\sum_{s_{-i}\in S_{-i}}\sigma_i^*(s_i)\prod_{j\in N\setminus\{i\}}\sigma_j^*(s_j)u_i\left((s_j)_{j\in N}\right)$$

$$= \sum_{s_i \in S_i} \sigma_i^*(s_i) u_i(s_i, \sigma_{-i}^*)$$
$$< u_i(s_i', \sigma_{-i}^*)$$

The third line follows from the assumption that $u_i(s_i', \sigma_{-i}^*) \geqslant u_i(s_i'', \sigma_{-i}^*)$ for every $s_i'' \in S_i$, $u_i(s_i', \sigma_{-i}^*) > u_i(s_i, \sigma_{-i}^*)$, and $\sigma_i^*(s_i) \neq 0$. Hence we have $u_i(\sigma_i^*, \sigma_{-i}^*) < u_i(s_i', \sigma_{-i}^*)$ which contradicts our assumption that $(\sigma_i^*)_{i \in N}$ forms an MSNE for the game. □

It immediately follows from Theorem 2.2 that, if a player mixes two or more strategies (i.e. puts non-zero probability on two or more strategies) in an MSNE, then the expected utility of that player by playing any of those pure strategies is the same (of course assuming others are playing according to the MSNE mixed strategy profile) and thus the player is "indifferent" to these pure strategies.

Using Theorem 2.2, we can check that $\{(1/2, 1/2), (1/2, 1/2)\}$ forms an MSNE for the matching pennies game. Can you prove that $\{(1/2, 1/2), (1/2, 1/2)\}$ is the only MSNE for the matching pennies game? We can also verify that $\{(1/3, 1/3, 1/3), (1/3, 1/3, 1/3)\}$ forms an MSNE for the rock-paper-scissor game. Can there exist games where there does not exist any MSNE? Yes! However, the celebrated Nash Theorem shows that every finite game[2] has at least one MSNE.

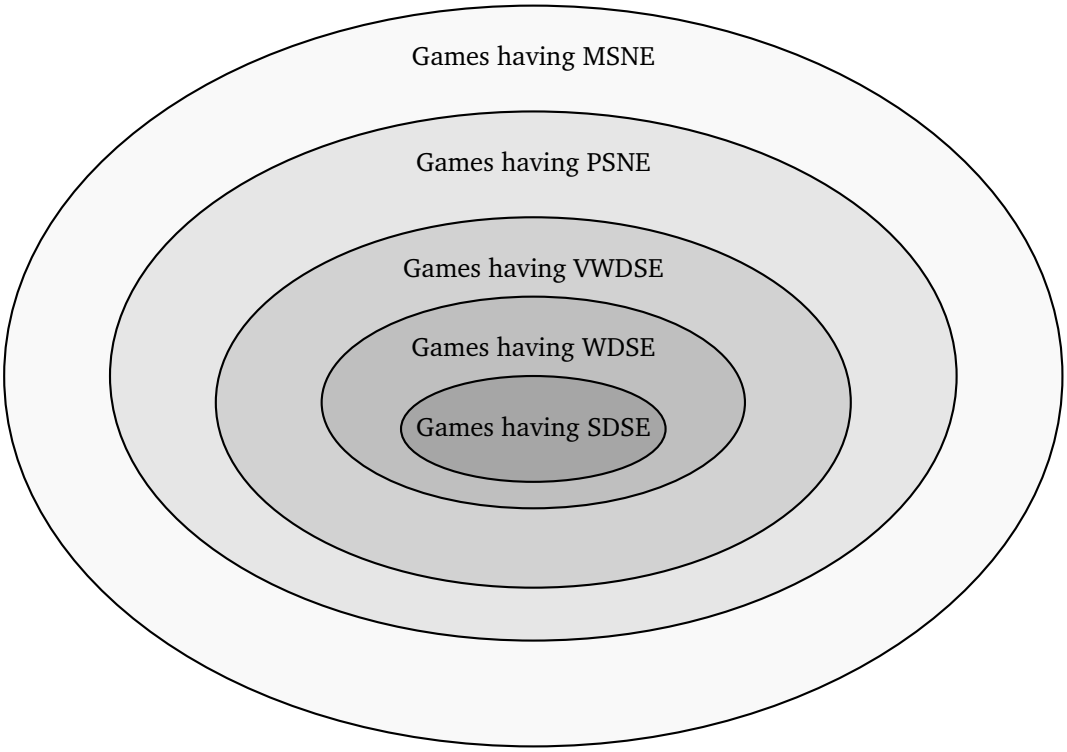



Figure 2.1: Pictorial depiction of relationship among various game theoretic solution concepts.

[2]A game is called finite if it involves a finite number of players and the strategy set of every player is a finite set.

**Nash Theorem**

**Theorem 2.3.** Every finite game has at least one MSNE.

We note that the condition about the game being finite is necessary in the Nash Theorem since there obviously exist games having no MSNE (can you give an example?). We skip the proof of the Nash theorem since the flavor of the proof does not match with the taste of the course. We refer interested readers to [WHJ+, Section 5.3]. However, the existence of MSNEs for an important class of games, called two players zero-sum games, was known long before the Nash Theorem. Two players zero-sum games is our next topic of study.

# Chapter 3

# Matrix Games

One of the many important criticisms of Nash equilibrium is that it does not take into account the notion of the "security of the players." Let us understand the notion of security with an example.

## 3.1 Security: the Maxmin Concept

Let us understand the notion of the security of players with an example. Consider the game below where we have two players each having two strategies each namely $A$ and $B$. We can see that the game has a unique Nash equilibrium, namely $(B, B)$. However, as we have mentioned before, the game-theoretic assumptions (i.e. rationality, intelligence, and common knowledge) are often quite unrealistic in many applications and in such applications, player 1 may hesitate to play according to the unique PSNE (that is playing $B$). Because if for some reason, player 2 ends up playing the strategy $A$ (maybe because of irrationality say), then player 1 incurs heavy loss. Hence, in this kind of situation, one can argue that player 1 can go for a "safer" option of playing the strategy $A$ since that will *guarantee* player 1 to have a utility of at least 2. This, in turn, reduces the motivation for the player 2 to play strategy $B$ and player 2 also ends up playing strategy $A$.

**Example 3.1.1** (Game with unique but insecure Nash equilibrium)**.** *Let us consider the following game.*

- ▷ *The set of players* $(N) : \{1, 2\}$
- ▷ *The set of strategies:* $S_i = \{A, B\}$ *for every* $i \in [2]$
- ▷ *Payoff matrix:*

| | | *Player* 2 | |
|---|---|---|---|
| | | A | B |
| *Player* 1 | A | $(2, 2)$ | $(2.5, 2.5)$ |
| | B | $(-100, 3)$ | $(3, 3)$ |

The above line of reasoning indicates another line of "rational thinking" – play the strategy which guarantees maximum utility without assuming anything about how the other player plays. This worst-case utility in the above sense is called the "security" of that player which we define now formally.

**Security of Player in Pure Strategies – Maxmin Value in Pure Strategies**

**Definition 3.1.** Given a game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$ in normal form, we define the "security level in pure strategies" or "value" $v_{-i}$ of player $i$ in pure strategies to be the maximum utility that player $i$ can get by playing any pure strategy:

$$v_{-i} = \max_{s_i \in S_i} \min_{s_{-i}\in S_{-i}} u_i(s_i, s_{-i})^a$$

A strategy $s_i^* \in S_i$ that guarantees the player $i$ his/her security level is called a *maxmin pure strategy* for the player $i$.

[a] For games where strategy sets may be infinite, we need to replace max and min by sup and inf respectively since the concepts max and min may not make sense there.

For example, we can see that the security of both the players (in pure strategies) in the matching pennies and the rock-paper-scissor games is $-1$. We can naturally extend Definition 3.1 to mixed strategies as follows.

**Security of Player in Mixed Strategies – Maxmin Value in Mixed Strategies**

**Definition 3.2.** Given a game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$ in normal form, we define the "security level in mixed strategies" or "value" $v_{-i}$ of player $i$ in mixed strategies to be the maximum utility that player $i$ can get by playing any mixed strategy:

$$v_{-i} = \sup_{\sigma_i \in \Delta(S_i)} \inf_{\sigma_{-i}\in \times_{j\neq i}\Delta(S_j)} u_i(\sigma_i, \sigma_{-i})$$

A strategy $s_i^* \in S_i$ that guarantees the player $i$ his/her security level is called a *maxmin mixed strategy* for the player $i$.

Again by averaging principle, it follows that the value $v_{-i}$ of player $i$ in mixed strategies can be also be written as the following.

$$v_{-i} = \sup_{\sigma_i \in \Delta(S_i)} \min_{s_{-i}\in S_{-i}} u_i(\sigma_i, s_{-i})$$

Using the above characterization, we can easily compute that the value of both the players (in mixed strategies) in the matching pennies and the rock-paper-scissor games is 0 which is the same as the utility that each player receives in those games in the unique Nash equilibrium. Actually, the following more general result holds whose proof follows immediately from the definition of Nash equilibrium and the value of players. We leave its proof to the reader.

**Relationship between Nash equilibrium and value of players**

**Theorem 3.1.** Given a strategic form game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, if $(\sigma_i^*)_{i\in N} \in \times_{i\in N}\Delta(S_i)$ is an MSNE for $\Gamma$, then we have $u_i((\sigma_i^*)_{i\in N}) \geqslant v_{-i}$ for every player $i \in N$ where $v_{-i}$ is the value of the player $i$ in mixed strategies.

It turns out that the inequality in Theorem 3.1 holds with equality for two-player zero-sum games; we

will prove this in Corollary 3.3. A game is called a zero-sum game or strictly competitive game is the sum of the utilities for all the players is 0 in every strategy profile. Hence, it follows that in a zero-sum game, if we have only two players, then the game can be represented by a matrix (assuming the game is finite) – the matrix corresponds to the utility function of player 1 and the utility function of player 2 is simply the minus of that matrix. This is why two-player zero-sum games are also called "matrix games." It is assumed that the matrix in a matrix game corresponds to the utility function of the row player. Hence, for a matrix game $\Gamma$ given by a $m \times n$ matrix $A$, the security level or value of the row player in mixed strategies, denoted by $\underline{v}$, is given by the following.

$$\underline{v} = \max_{\sigma \in \Delta([m])} \min_{j \in [n]} \sum_{i=1}^{m} \sigma(i) A_{i,j}$$

Because of the above expression, the value of the row player in a zero sum game is also called the maxmin value of the game in mixed strategies. Similarly, the value of the column player in mixed strategies, denoted by $\overline{v}$, is given by the following.

$$\overline{v} = \min_{\sigma \in \Delta([n])} \max_{i \in [m]} \sum_{j=1}^{n} \sigma(j) A_{i,j}$$

Because of the above expression, the value of the column player in a zero-sum game is also called the minmax value of the game in mixed strategies. The beauty of zero-sum games is that the maxmin value in mixed strategies coincides with the minmax value in mixed strategies. Hence it follows from Theorem 3.1 that, if a Nash equilibrium exists (which we will prove that it always exists for matrix games), then the utility that each player receives is their corresponding security levels in mixed strategies. Moreover, the corresponding maxmin and minmax strategies form an MSNE for the matrix game. The utility that the row player receives in any MSNE in a matrix game is thus called the *value* (is value well-defined?) of that matrix game. We prove these facts in the next section. From now on, if not mentioned otherwise, we will assume all values to be in mixed strategies.

## 3.2 Minimax Theorem

Our first observation is that the maxmin value is at most the minmax value in every matrix game.[1]

**Maxmin value is at most minmax value**

**Lemma 3.1.** For any matrix game $A$, we have the following.

$$\max_{\sigma \in \Delta([m])} \min_{j \in [n]} \sum_{i=1}^{m} \sigma(i) A_{i,j} \leqslant \min_{\sigma \in \Delta([n])} \max_{i \in [m]} \sum_{j=1}^{n} \sigma(j) A_{i,j}$$

*Proof of Lemma 3.1.* Let $\sigma^* \in \Delta([m])$ be the mixed strategy which maximizes $\min_{j \in [n]} \sum_{i=1}^{m} \sigma(i) A_{i,j}$. Then we have the following.

$$\max_{\sigma \in \Delta([m])} \min_{j \in [n]} \sum_{i=1}^{m} \sigma(i) A_{i,j} = \min_{j \in [n]} \sum_{i=1}^{m} \sigma^*(i) A_{i,j}$$

[1] The results actually holds for any function $f : \mathcal{A} \times \mathcal{B} \longrightarrow \mathbb{R} : \sup_{a \in \mathcal{A}} \inf_{b \in \mathcal{B}} f(a, b) \leqslant \inf_{b \in \mathcal{B}} \sup_{a \in \mathcal{A}} f(a, b)$. This inequality is known as the *max-min inequality*.

$$
\begin{aligned}
&= \min_{\sigma \in \Delta([n])} \sum_{j=1}^{n} \sum_{i=1}^{m} \sigma^*(i)\sigma(j)A_{i,j} \\
&= \min_{\sigma \in \Delta([n])} \sum_{i=1}^{m} \sum_{j=1}^{n} \sigma^*(i)\sigma(j)A_{i,j} \\
&= \min_{\sigma \in \Delta([n])} \sum_{i=1}^{m} \sigma^*(i) \sum_{j=1}^{n} \sigma(j)A_{i,j} \\
&\leqslant \min_{\sigma \in \Delta([n])} \max_{i \in [m]} \sum_{j=1}^{n} \sigma(j)A_{i,j}
\end{aligned}
$$

□

We now present the most important result of matrix games which is known as the minmax Theorem.

**Minmax Theorem**

**Theorem 3.2.** For every matrix game given by a $m \times n$ matrix $\mathcal{A}$, there exists a mixed strategy $x^* = (x_1^*, \ldots, x_m^*) \in \Delta([m])$ for the row player and a mixed strategy $y^* = (y_1^*, \ldots, y_n^*) \in \Delta([n])$ for the column player such that the following holds.

$$\max_{x \in \Delta([m])} x\mathcal{A}y^* = \min_{y \in \Delta([n])} x^*\mathcal{A}y$$

We now prepare background to prove Theorem 3.2. For that, let us analyze the situation for the row player. Recall, the row player wants to play a mixed strategy, denoted by a vector $x = (x_1, \ldots, x_m)$ say, which guarantees the highest utility in the worst case. Then the row player is faced with the following optimization problem which is a linear program.

| | | |
|---|---|---|
| maximize | $\min_{j \in [n]} \sum_{i=1}^{m} \mathcal{A}_{i,j} x_i$ | |
| subject to: | $\sum_{i=1}^{m} x_i = 1$ | |
| | $x_i \geqslant 0$ | $\forall\, i \in [m]$ |

*Row player's linear program*

| | | |
|---|---|---|
| maximize | $t$ | |
| subject to: | $t \leqslant \sum_{i=1}^{m} \mathcal{A}_{i,j} x_i$ | $\forall\, j \in [n]$ |
| | $\sum_{i=1}^{m} x_i = 1$ | |
| | $x_i \geqslant 0$ | $\forall\, i \in [m]$ |

*Row player's equivalent linear program*

Similarly, the column player is faced with the following linear program.

| | | | | | |
|---|---|---|---|---|---|
| minimize | $\max_{i\in[m]} \sum_{j=1}^{n} \mathcal{A}_{i,j} y_j$ | | minimize | $w$ | |
| subject to: | $\sum_{j=1}^{n} y_j = 1$ | | subject to: | $w \geqslant \sum_{j=1}^{n} \mathcal{A}_{i,j} y_j$ | $\forall\, i \in [m]$ |
| | $y_j \geqslant 0$ | $\forall\, j \in [n]$ | | $\sum_{j=1}^{n} y_j = 1$ | |
| | | | | $y_j \geqslant 0$ | $\forall\, j \in [n]$ |
| *Column player's linear program* | | | *Column player's equivalent linear program* | | |

The proof of Theorem 3.2 follows easily from what is called the *strong duality theorem* for linear programs. We refer to Luca Trevisan's blog post on LP duality for a brief but beautiful introduction to linear programming duality [Tre].

**Strong Duality Theorem**

**Theorem 3.3.** Let $\mathrm{LP}_1$ and $\mathrm{LP}_2$ be two linear programs which are dual of each other. If either $\mathrm{LP}_1$ or $\mathrm{LP}_2$ is feasible and bounded, then so is the other, and $\mathrm{OPT}(\mathrm{LP}_1) = \mathrm{OPT}(\mathrm{LP}_2)$.

We now prove Theorem 3.2.

*Proof of Theorem 3.2.* Let us call the linear programs of the row and column players $\mathrm{LP}_1$ and $\mathrm{LP}_2$ respectively. The optimal values of the linear programs for both the row and column players is upper bounded and lower bounded by respectively the maximum and minimum value of the matrix $\mathcal{A}$. Hence, by strong duality theorem, we know that we have $\mathrm{OPT}(\mathrm{LP}_1) = \mathrm{OPT}(\mathrm{LP}_2)$. Let $x^* = (x_1^*, \ldots, x_m^*)$ and $y^* = (y_1^*, \ldots, y_n^*)$ be the solutions of $\mathrm{LP}_1$ and $\mathrm{LP}_2$ respectively (i.e. which optimizes the corresponding linear programs). Hence, we have the following.

$$\min_{j\in[n]} x^* \mathcal{A} e_j = \min_{j\in[n]} \sum_{i=1}^{m} \mathcal{A}_{i,j} x_i = \mathrm{OPT}(\mathrm{LP}_1) = \mathrm{OPT}(\mathrm{LP}_2) = \max_{i\in[m]} \sum_{j=1}^{n} \mathcal{A}_{i,j} y_j = \max_{i\in[m]} e_i \mathcal{A} y^*$$

Now the result follows from the following observation which is again due to the averaging principle.

$$\max_{i\in[m]} e_i \mathcal{A} y^* = \max_{x\in\Delta([m])} x \mathcal{A} y^*$$

$$\min_{j\in[n]} x^* \mathcal{A} e_j = \min_{y\in\Delta([n])} x^* \mathcal{A} y$$

□

We can prove the following important result from Theorem 3.2.

**Existence and Polynomial Time Computability of MSNE in Matrix Games**

**Corollary 3.1.** In every matrix game, there exists an MSNE where both the players play a mixed strategy which guarantees each of them their value in mixed strategies.

*Proof of Corollary 3.1.* Let $x^* = (x_1^*, \ldots, x_m^*)$ and $y^* = (y_1^*, \ldots, y_n^*)$ be as defined in Theorem 3.2. From the proof of Theorem 3.2, we have the following.

$$\max_{x \in \Delta([m])} x\mathcal{A}y^* = \min_{y \in \Delta([n])} x^*\mathcal{A}y$$

We now prove that $x^*$ is a utility-maximizing strategy for the row player given that the column player is playing $y^*$.

$$x^*\mathcal{A}y^* \geqslant \min_{y \in \Delta([n])} x^*\mathcal{A}y = \max_{x \in \Delta([m])} x\mathcal{A}y^*$$

Similarly for the column player, we have the following.

$$x^*\mathcal{A}y^* \leqslant \max_{x \in \Delta([m])} x\mathcal{A}y^* = \min_{y \in \Delta([n])} x^*\mathcal{A}y$$

Hence $(x^*, y^*)$ forms an MSNE for the matrix game. Since linear programs can be solved in polynomial time, it follows that $x^*$ and $y^*$ (and thus an MSNE of the matrix game) can be computed in polynomial time. □

We now show that the maxmin value in mixed strategies coincides with minmax value in mixed strategies in matrix games. Again the proof follows almost immediately from Theorem 3.2.

**Maxmin and Minmax Value Coincides in Matrix Games**

**Corollary 3.2.** For every $m \times n$ matrix $\mathcal{A}$, we have the following.

$$\max_{x \in \Delta([m])} \min_{y \in \Delta([n])} x\mathcal{A}y = \min_{y \in \Delta([n])} \max_{x \in \Delta([m])} x\mathcal{A}y$$

*Proof of Corollary 3.2.* Due to Lemma 3.1, it is enough to prove

$$\max_{x \in \Delta([m])} \min_{y \in \Delta([n])} x\mathcal{A}y \geqslant \min_{y \in \Delta([n])} \max_{x \in \Delta([m])} x\mathcal{A}y$$

Let $x^*$ and $y^*$ be the strategies as defined in the proof of Theorem 3.2 (that is, they are the mixed strategies of the row and column players which guarantee corresponding security levels in mixed strategies). We have the following.

$$\max_{x \in \Delta([m])} \min_{y \in \Delta([n])} x\mathcal{A}y \geqslant \min_{y \in \Delta([n])} x^*\mathcal{A}y = \max_{x \in \Delta([m])} x\mathcal{A}y^* \geqslant \min_{y \in \Delta([n])} \max_{x \in \Delta([m])} x\mathcal{A}y$$

□

We next deduce another important result from Theorem 3.2 which shows that in any MSNE of a matrix game, both the row and the column players' payoff are exactly their security levels.

**Payoff at MSNE in Matrix Game**

**Corollary 3.3.** Let $\mathcal{A}$ be a $m \times n$ matrix. Then, in the corresponding matrix game, the payoff of both the row and the column players at any MSNE are their corresponding security levels.

*Proof of Corollary 3.3.* Let $(x^*, y^*) \in \Delta([m]) \times \Delta([n])$ be an MSNE of the matrix game corresponding to $\mathcal{A}$. Let $v_r$ and $v_c$ be the security levels of the row and column players respectively. Then, by definition of security levels, we have the following.

$$v_r = \max_{x \in \Delta([m])} \min_{y \in \Delta([n])} x\mathcal{A}y, \qquad v_c = -\min_{y \in \Delta([n])} \max_{x \in \Delta([m])} x\mathcal{A}y, \qquad v_r = -v_c$$

Since $(x^*, y^*)$ is an MSNE for the matrix game, we have $x^*\mathcal{A}y^* \geqslant v_r$ and $-x^*\mathcal{A}y^* \geqslant -v_c$ since otherwise players can play the strategy that guarantees their security level. Also we claim that $x^*$ ($y^*$ respectively) is a mixed strategy which guarantees security level for the row player (column player respectively) irrespective of what the column player (row player respectively) plays. Indeed otherwise there exists a mixed strategy $y \in \Delta([n])$ ($x \in \Delta([m])$ respectively) for the column player (row player respectively) which gives more payoff to him/her. This contradicts our assumption that $(x^*, y^*)$ is an MSNE for the matrix game. We also observe that $x^*$ (respectively $y^*$) is a security level guaranteeing mixed strategy for the row player (respectively column player) since otherwise the column player (respectively row player) has incentive to deviate from $y^*$ (respectively $x^*$). Now we have the following for the utility of the row player.

$$x^*\mathcal{A}y^* \leqslant \max_{x \in \Delta([m])} x\mathcal{A}y^* = \min_{y \in \Delta([n])} x^*\mathcal{A}y \leqslant \max_{x \in \Delta([m])} \min_{y \in \Delta([n])} x\mathcal{A}y = v_r$$

Similarly, we have the following for the column player.

$$-x^*\mathcal{A}y^* \leqslant -\min_{y \in \Delta([n])} x^*\mathcal{A}y = -\max_{x \in \Delta([m])} x\mathcal{A}y^* \leqslant -\min_{y \in \Delta([n])} \max_{x \in \Delta([m])} x\mathcal{A}y = v_c$$

□

## 3.3 Application of Matrix Games: Yao's Lemma

We know that any comparison based deterministic sorting algorithm must make $\Omega(n \log n)$ pairwise comparisons to sort $n$ elements. The typical proof of this statement views any comparison based deterministic sorting algorithms like decision trees — the internal nodes of the decision tree represent comparisons and leaf nodes correspond to complete orders of $n$ elements. Since there must be at least $n!$ leaf nodes, it follows that the height of the decision tree, and thus the number of pairwise comparisons made by the algorithm in the worst case, is $\Omega(\log n!) = \Omega(n \log n)$. Can we beat this lower bound by designing randomized algorithms for sorting? An algorithm for a problem (sorting for example) is called a randomized algorithm if it adaptively tosses fair coin some number of times and, for every input, it outputs correctly with probability at least $\frac{2}{3}$. We will prove, using Yao's lemma that, any comparison based randomized algorithm for sorting also makes $\Omega(n \log n)$ number of queries in expectation. In general, Yao's lemma provides a generic technique to prove lower bounds for randomized algorithms.

### 3.3.1 View of Randomized Algorithm as Distribution over Deterministic Algorithms

Before presenting Yao's lemma, let us see another perspective of randomized algorithms. Let $\mathcal{A}$ be a randomized algorithm that makes $s$ number of coin tosses. Observe that if the outcomes of these $s$ random coin tosses are fixed to some $s_i$, then the randomized algorithm $\mathcal{A}$ becomes a deterministic algorithm – call it $\mathcal{A}(s_i)$. Hence, $\mathcal{A}$ is nothing but a distribution over the deterministic algorithms $\{\mathcal{A}(s_i) : s_i \in \{0,1\}^s\}$.

### 3.3.2 Yao's Lemma

**Yao's Lemma**

**Lemma 3.2.** Let $A$ be a randomized algorithm for some problem $\Pi$. For an input $x$, let $T(A, x)$ be the random variable denoting the cost (e.g. running time, number of comparisons made) of $A$ on $x$. Let $\mathcal{X}$ be the set of all inputs (of length $n$) to $\Pi$, $X$ be a random variable denoting the input chosen from $\mathcal{X}$ according to some distribution $p$ on $\mathcal{X}$, and $\mathcal{A}_\Pi$ be the set of all deterministic algorithms for the problem $\Pi$. Then we have the following.

$$\max_{x \in \mathcal{X}} \mathbb{E}[T(A, x)] \geqslant \min_{a \in \mathcal{A}_\Pi} \mathbb{E}[T(a, X)]$$

*Proof.* Consider the following matrix game $\mathcal{B}$— rows are indexed by $\mathcal{X}$, columns are indexed by $\mathcal{A}_\Pi$, the entry of the matrix indexed by $(x, a)$ is the cost of the algorithm $a$ on $x$. Let the algorithm $A$ be the probability distribution $\sigma_A \in \Delta(\mathcal{A}_\Pi)$. Then by Lemma 3.1, we have the following.

$$\begin{aligned}
&\min_{z \in \Delta(\mathcal{A}_\Pi)} \max_{x \in \mathcal{X}} e_x \mathcal{B} z \geqslant \max_{y \in \Delta(\mathcal{X})} \min_{a \in \mathcal{A}_\Pi} y \mathcal{B} e_a \\
&\quad \Rightarrow \max_{x \in \mathcal{X}} e_x \mathcal{B} \sigma_A \geqslant \min_{a \in \mathcal{A}_\Pi} X \mathcal{B} e_a && [A \text{ is one such } z; X \text{ is one such } x] \\
&\Rightarrow \max_{x \in \mathcal{X}} \mathbb{E}[T(A, x)] \geqslant \min_{a \in \mathcal{A}_\Pi} \mathbb{E}[T(a, X)]
\end{aligned}$$

□

We now prove lower bound for randomized sorting algorithms.

**Sorting Lower Bound for Randomized Algoorithms**

**Theorem 3.4.** Any comparison based randomized algorithm for sorting $n$ items makes $\Omega(n \log n)$ number of comparison queries in expectation.

*Proof.* Let $A$ be any randomized algorithm for sorting, $\mathcal{A}$ the set of deterministic algorithms for sorting, $\mathcal{X}$ the set of all possible input for it, and $T(A, x)$ the random variable denoting the number of comparisons that $A$ makes on $x$. We need to show that $\max_{x \in \mathcal{X}} \mathbb{E}[T(A, x)] = \Omega(n \log n)$. From Yao's Lemma, it is enough to show that $\min_{a \in \mathcal{A}} \mathbb{E}[T(a, X)] = \Omega(n \log n)$ for some random variable $X$ on $\mathcal{X}$. Let us define $X$ to be uniform distribution over $\mathcal{X}$. We now show that for this choice of $X$, we have $\mathbb{E}[T(a, X)] = \Omega(n \log n)$ for every deterministic sorting algorithm $a \in \mathcal{A}$; i.e. the average number of comparison queries that any deterministic sorting algorithm makes is $\Omega(n \log n)$. Again viewing deterministic sorting algorithms as decision trees, we need to prove that the average height (which is the sum of heights of all the leaf nodes divided by the number of leaf nodes) of any binary tree $\mathcal{T}$ with $t (= n!)$ nodes is $\Omega(\log t) (= \Omega(n \log n))$.

If the tree $\mathcal{T}$ is balanced (that is if the difference between the depth of any two leaf nodes is at most 1), then the result follows immediately. Otherwise, we take two leaf nodes $u$ and $v$ so that the depth of $u$ is at least 2 more than the depth of $v$. We remove $u$ and make it a child of $v$ – this operation never increases the average depth of the leaf nodes of the tree. This simple averaging argument suffices since we have chosen $\mathcal{X}$ as uniform distribution. Hence, we have proved that among all binary trees with $t$ leaves, there exists a

balanced binary tree whose average height is minimum among all binary trees on $t$ leaf nodes. Hence, the average height of the decision tree for the algorithm $a$ is $\Omega(n \log n)$. □

# Chapter 4

# Computing Equilibrium

Nash theorem guarantees the existence of a mixed strategy Nash equilibrium in every finite normal form game. However, given a finite game in its normal form, can we compute an MSNE of the game in normal form? Before we can answer this, there is another subtle but important issue – what is the guarantee that a Nash equilibrium can be written in finite space (for example, what if the probability of playing certain strategy for some player turns out to be an irrational number?). If the input game has exactly 2 players, then we will see that all the probabilities involved in any MSNE are rational numbers. However, there exists 3 player normal form game where all the MSNEs involve an irrational number even though all the input numbers are rational. To tackle this fundamental issue, we change our goal to finding an $\varepsilon$-MSNE – a strategy profile is an $\varepsilon$-MSNE if unilateral deviation can increase the utility of any player at most by an additive $\varepsilon$; here we assume that the utility of every player in every strategy profile lies between 0 and 1 (how can we assume this without loss of generality?). Given a normal form game, we denote the computational problem of finding a $\varepsilon$-MSNE by $\varepsilon$-NASH. Can you see why there is always an $\varepsilon$-MSNE involving only rational numbers in every normal form game?

## 4.1 Computing MSNEs of Bimatrix Games by Enumerating Support

We next see an algorithm for NASH for 2 player games. Let player 1 has $m$ strategies and the player 2 has $n$ strategies. The payoffs of player 1 and 2 are given by the matrices $\mathcal{A} = (b_{i,j})_{i\in[m],j\in[n]}$ and $\mathcal{B} = (b_{i,j})_{i\in[m],j\in[n]}$. For this reason, 2 player normal form games are also called *bimatrix games*. We guess the support $\mathcal{I} \subseteq [m]$ of player 1 and the support $\mathcal{J} \subseteq [n]$ of player 2 in an MSNE $(\sigma_1^*, \sigma_2^*) \in \Delta([m]) \times \Delta([n])$. That is, for $i \in [m], j \in [n]$, we have $\sigma_1^*(i) \neq 0$ if and only if $i \in \mathcal{I}$ and $\sigma_2^*(j) \neq 0$ if and only if $j \in \mathcal{J}$. Then we compute if there exist vectors $x \in \mathbb{R}^m$ (corresponds to $\sigma_1^*$) and $y \in \mathbb{R}^n$ (corresponds to $\sigma_2^*$) such that the following holds.

$$\sum_{j=1}^{n} a_{i,j}y_j = u \quad \forall\, i \in \mathcal{I}, \quad \sum_{j=1}^{n} a_{i,j}y_j \leqslant u \quad \forall\, i \in [m] \setminus \mathcal{I}$$

$$\sum_{i=1}^{m} b_{i,j}x_i = v \quad \forall\, j \in \mathcal{J}, \quad \sum_{i=1}^{m} b_{i,j}x_i \leqslant v \quad \forall\, j \in [n] \setminus \mathcal{J}$$

$$\sum_{i \in \mathcal{I}} x_i = 1, \ \sum_{j \in \mathcal{J}} y_j = 1$$

$$x_i \geqslant 0 \ \ \forall\, i \in \mathcal{I}, \ y_j \geqslant 0 \ \ \forall\, j \in \mathcal{J}, x_i = 0 \ \ \forall\, i \in [m] \setminus \mathcal{I}, y_j = 0 \ \ \forall\, j \in [n] \setminus \mathcal{J}$$

The correctness of the above algorithm follows from the indifference principle (Theorem 2.2). We leave its formal proof of correctness to the reader. Since linear programs can be solved in polynomial time, the running time of the algorithm above is $\mathcal{O}(2^{m+n}\text{poly}(m,n))$. However, the length of the input is $\mathcal{O}(mn)$ and hence the above algorithm runs in exponential time. Indeed, we do not know any polynomial-time algorithm for computing an MSNE even for Bimatrix games. We will see why we do not hope to have a polynomial-time algorithm for computing an MSNE even for Bimatrix games. However, before we proceed let's shed light on the input size again. To describe a normal form game with $n$ players each having $s$ strategies, we need $ns^n$ numbers. We observe that, unlike MSNE, a PSNE can be computed in polynomial time by simply iterating over all possible strategy profiles. This observation trivializes the computation of PSNEs. However, the above algorithm runs in polynomial time because "the input itself is quite large." Indeed, in many applications of game theory, network congestion games, for example, the input is given succinctly and never presented explicitly by listing down the utilities of every player in every strategy profile. Below we see a list of interesting types of games where the input can be given succinctly. Consequently, the computational problem of finding a PSNE becomes interesting for some of these classes of games. The reader is strongly encouraged to think of interesting examples of games for all the types of games discussed below.

## 4.2 Important Classes of Succinct Games

1. *Graphical games:* Let $\mathcal{G}$ be a directed graph on the set $N$ of players which encodes the following information: the utility of any player $i \in N$ depends only on the strategy played by those players (including $i$ of course) which has an outgoing edge into $i$. If the in degree of any node in $\mathcal{G}$ is at most $d$, then the game can be described by $ns^{d+1}$ numbers.

2. *Sparse games:* In this type of game, all the $ns^n$ utilities are zero except a few which is given explicitly. Graphical games is a form of sparse game.

3. *Symmetric games:* In these games, the players are all identical. Hence, the utility of a player in a strategy profile depends on how many players play each of the $s$ strategies. In any strategy profile, all the players who play the same strategy receive the same utility. Such games can be described using $s\binom{n+s-1}{s-1}$ numbers. We observe that a two-player bimatrix game given by matrices $\mathcal{A}$ and $\mathcal{B}$ is symmetric if and only if $\mathcal{A} = \mathcal{B}^T$ (where $\mathcal{B}^T$ denotes the transpose of the matrix $\mathcal{B}$).

4. *Anonymous games:* This is a generalization of symmetric games. Here all the players are different but every player cannot distinguish others. So the utility of a player depends on how many other players play each of the strategies. However, players who play the same strategy in a strategy profile may receive different utilities. These games can be described using $ns\binom{n+s-2}{s-1}$ numbers.

5. *Network congestion games:* Here we have a graph $\mathcal{G}$. Every player has a source and a destination vertex in $\mathcal{G}$; the strategy set of each player is the set of all paths from his source to his destination. The load $\ell(e)$ of an edge $e \in \mathcal{G}$ is the number of players using that edge. Each edge $e$ also has a congestion function $c_e$ which is assumed to be a non-decreasing cost function. The utility of player $i$

in the strategy profile $P = (P_j)_{j \in N}$ is $\sum_{e \in P_i} c_e(\ell(P))$. For network congestion games, we usually work with cost functions and players try to minimize their cost (so utility function is the minus of the cost function).

6. *Congestion games:* Congestion games are a generalization of network congestion games. We have a ground set $\mathcal{E}$ (think of this as the set of edges in network congestion games). The strategy set of each player is some set of subsets of $\mathcal{E}$ (think of these as paths). Every element $e \in \mathcal{E}$ has a function $c_e$ associated with it which is assumed to be a non-decreasing cost function. Let $\ell_e$ be the number of players whose action includes $e$. The utility of player $i$ in the strategy profile $P = (P_j)_{j \in N}$ is $\sum_{e \in P_i} c_e(\ell(P))$. For congestion games also, we usually work with cost functions and players try to minimize their cost (so utility function is the minus of the cost function).

7. *Multi-matrix games:* Suppose in an $n$ player game, each player has $m$ strategies. For each ordered pair $(i, j) \in [n] \times [n]$ of players, we have an $m \times m$ utility matrix $\mathcal{A}^{ij}$. The utility of player $i$ in a strategy profile $(s_i)_{i \in [n]}$ is $\sum_{j \neq i} \mathcal{A}^{ij}_{s_i s_j}$. These type of games arise often in networks where player $i$ receives benefits from all its neighbor and his/her utility is the sum of benefit received.

We will see next that a PSNE can be computed efficiently for some of the above succinct games, but we do not have much hope to have an efficient algorithm for computing a PSNE for some of the succinct games, for example, congestion games.

## 4.3 Potential Games

Recall, Nash theorem guarantees the existence of an MSNE for finite games. We have seen games, matching pennies for example, that do not have any PSNE. Potential games form an important class of games which have PSNEs. Intuitively speaking, potential games are those games that admit a "potential function" – a function which all the players are collectively trying to optimize. Let us postpone the definition of potential games for the moment and prove that network congestion games (which is a potential game) aka atomic selfish routing games always have a PSNE.

**Rosenthal's Theorem [Ros73]**

**Theorem 4.1.** Every network congestion game, with arbitrary real-valued cost functions, has at least one PSNE.

*Proof.* Let $f$ be a flow in the given graph $\mathcal{G}$.[1] We define a *potential function* on the set of flows in $\mathcal{G}$ as

$$\Phi(f) = \sum_{e \in \mathcal{G}} \sum_{i=1}^{f_e} c_e(i)$$

We will now show that, if there exists a player $i$ who can reduce his cost by shifting to another path $P'_i$ from his current path $P_i$, then the reduction of the cost of player $i$ is exactly the same as the corresponding reduction of the potential function (this is exactly the requirement for a function to be called a potential

[1] Recall an atomic flow $f$ is a function $f : \mathcal{E}[\mathcal{G}] \longrightarrow \mathbb{N}$ which satisfies capacity and conservation conditions.

function). Let $\hat{f}$ be the new flow after player $i$ changes his strategy to $P_i'$ from $P_i$, then we have the following.

$$\Phi(\hat{f}) - \Phi(f) = \sum_{e \in P_i'} c_e(\hat{f}_e) - \sum_{e \in P_i} c_e(f_e)$$

Once we have a potential function, the proof of the existence of PSNE is straight forward. Since there are only finitely many atomic flows in $\mathcal{G}$, there exists a flow $f^*$ which minimizes $\Phi(f^*)$. Since $\Phi$ is a potential function, it follows that any strategy profile corresponding to the flow $f^*$ is a PSNE. □

We now define a potential game formally.

**Potential Games**

**Definition 4.1.** A game $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ is called a potential game if there exists a function $\Phi$ from the set of all strategy profiles to real numbers which satisfies the following.

$$\forall\, i \in N, s_i, s_i' \in S_i, s_{-i} \in S_{-i}, \Phi(s_i, s_{-i}) - \Phi(s_i', s_{-i}) = u_i(s_i, s_{-i}) - u_i(s_i', s_{-i})$$

Such a function $\Phi$ is called a potential function of the normal form game.

The proof of Theorem 4.1 immediately gives an algorithm for finding a PSNE in any potential game which is called "best response dynamics."

Best response dynamics

- ▷ Pick any arbitrary strategy profile $s$ of the game.
- ▷ While $s$ is not a PSNE
  - pick a player $i$ who has a profitable move from $s_i$ to $s_i'$.
  - replace $s_i$ with $s_i'$ in $s$

The following result follows immediately from the proof of Theorem 4.1.

**Convergence of best response dynamics [MS96]**

**Corollary 4.1.** Best response dynamics converge to a PSNE in every finite potential game.

*Proof.* Follows from the observation that the best response dynamics never consider any strategy profile more than one as the potential strictly decreases in every iteration. □

Corollary 4.1 is appealing from the point of view of the predictive power of Nash equilibrium also. It says that the players can eventually reach a pure strategy Nash equilibrium in potential games by repeatedly playing the game and following best response dynamics. Although Corollary 4.1 guarantees convergence, it does not talk about the speed of convergence. Obviously, if the potential function happens to take only polynomially many distinct values, then the best response dynamics converge in polynomial time. However, potential functions in typical applications take exponentially (in the number of players and the sum of the

number of strategies of the players) many values and it is indeed possible for the best response dynamic to take exponentially many steps to reach any PSNE. Can you find a potential game where the best response dynamics can take exponentially many steps to reach a PSNE? To achieve fast convergence, we weaken our need – we settle for $\varepsilon$-PSNE.

## 4.4 Approximate PSNE

**$\varepsilon$-PSNE**

**Definition 4.2.** Given a normal form game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, a strategy profile $s = (s_i)_{i\in N}$ is called an $\varepsilon$-PSNE for $\Gamma$ if no player can improve his utility more than $\varepsilon$ fraction of his utility in $s$ by deviating. That is, we have the following.

$$\forall\, i \in N,\ \forall\, s_i' \in S_i, u_i(s_i', s_{-i}) \leqslant (1+\varepsilon)u_i(s)$$

For congestion games (where the goal is to minimize cost), the definition of $\varepsilon$-PSNE requires that the cost of any player should not drop by more than $\varepsilon$ fraction of his current cost. That is, we should have the following for a strategy profile $s$ to be a $\varepsilon$-PSNE.

$$\forall\, i \in N,\ \forall\, s_i' \in S_i, C_i(s_i', s_{-i}) \geqslant (1-\varepsilon)C_i(s)$$

To ensure fast convergence, we modify the best response dynamics to $\varepsilon$ best response dynamics.

**$\varepsilon$-Best response dynamics**

- ▷ Pick any arbitrary strategy profile $s$ of the game.
- ▷ While $s$ is not a $\varepsilon$-PSNE
  - pick a player $i$ who has a move from $s_i$ to $s_i'$ that guarantees a reduction of cost by at least $\varepsilon$ fraction of his current cost.
  - replace $s_i$ with $s_i'$ in $s$

We now prove that $\varepsilon$-best response dynamics guarantee fast convergence in network potential games under some mild conditions.

**Fast Convergence of $\varepsilon$-Best Response Dynamics**

**Theorem 4.2.** In an atomic network congestion game, suppose the following holds.

- ▷ All the players have the same source and the same destination.
- ▷ Cost function satisfies *"$\alpha$-bounded jump condition"*: $C_e(x+1) \in [C_e(x), \alpha C_e(x)]$ for every edge $e$ and every positive integer $x$ where $\alpha \geqslant 1$.
- ▷ The max-gain variant of the $\varepsilon$-best response dynamics is used – among all players who have an $\varepsilon$ best response move, pick a player and a move which achieves largest absolute cost decrease

(for that player).

Then an $\varepsilon$-PSNE will be reached in $\mathcal{O}\left(\frac{n\alpha}{\varepsilon}\log\frac{\Phi(s^0)}{\Phi(s^{\min})}\right)$ where $\Phi(s^0)$ and $\Phi(s^{\min})$ are respectively the initial and minimum value of the potential function $\Phi$.

*Proof.* Let $s$ be a strategy profile which is not an $\varepsilon$-PSNE. The proof has two parts. In the first part, we show that there always exists a player $i^* \in N$ whose current cost is high. In the second part, we show that if player $j \in N$ is chosen by the max-gain version of the $\varepsilon$-best response dynamics, then the drop in the cost of player $j$ is at least some significant fraction of the cost of player $i^*$ (in the current profile).

**Claim 4.4.1.** *In every strategy profile $s$, there exists a player $i^* \in N$ such that $C_{i^*}(s) \geqslant \frac{\Phi(s)}{n}$.*

*Proof.* Let us define the cost $C(s)$ of a strategy profile $s$ as the sum of costs of all the players in $s$. That is

$$C(s) = \sum_{i \in N} C_i(s) = \sum_{e \in E[G]} f_e c_e(f_e)$$

Recall the potential function used in the proof of Theorem 4.1,

$$\Phi(s) = \sum_{e \in E[G]} \sum_{i=1}^{f_e} c_e(i)$$

Since cost functions are assumed to be non-decreasing, we have $\Phi(s) \leqslant C(s) = \sum_{i \in N} C_i(s)$. Thus we have $\max_{i \in N} C_i(s) \geqslant \frac{\Phi(s)}{n}$. Let $i^* \in \operatorname{argmax}_{i \in N} C_i(s)$ and we have $C_{i^*}(s) \geqslant \frac{\Phi(s)}{n}$. □

**Claim 4.4.2.** *Let $j \in N$ be the player chosen by the max-gain version of the $\varepsilon$-best response dynamics and the player $j$ moves to strategy $s'_j$ from his current strategy $s_j$. Then we have the following.*

$$C_j(s) - C_j(s'_j, s_{-j}) \geqslant \frac{\varepsilon}{\alpha} C_i(s) \text{ for every player } i \in N$$

*Proof.* Let us fix any player $i \in N$. We consider two cases.

*Case I: Suppose player $i$ has an $\varepsilon$ move.* Then since player $j$ has been chosen by the max-gain version of the $\varepsilon$-best response dynamics, we have

$$C_j(s) - C_j(s'_j, s_{-j}) \geqslant \max_{s'_i \in S_i} (C_i(s) - C_i(s'_i, s_{-i})) \geqslant \varepsilon C_i(s) \geqslant \frac{\varepsilon}{\alpha} C_i(s)$$

*Case II: Suppose player $i$ does not have an $\varepsilon$ move.* Since player $j$ has been chosen by the max-gain version of the $\varepsilon$-best response dynamics, we have

$$C_j(s'_j, s_{-j}) \leqslant (1 - \varepsilon) C_j(s) \tag{4.1}$$

Since all the players have the same source and destination, $s'_j$ is a valid strategy for the player $i$ too. Since player $i$ does not have any $\varepsilon$ move, we have

$$C_i(s'_j, s_{-i}) > (1 - \varepsilon) C_i(s) \tag{4.2}$$

We now have:

$$C_i(s'_j, s_{-i}) = C_i(\ldots, s_{i-1}, s'_j, s_{i+1}, \ldots)$$

$$
\begin{aligned}
&= C_j(\ldots, s_{i-1}, s_j, s_{i+1}, \ldots, s'_j, \ldots) && \text{[Players have the same source and destination]} \\
&\leqslant \alpha C_j(\ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s'_j, \ldots) && [\alpha - \text{bounded jump}] \\
&= \alpha C_j(s'_j, s_{-j}) && (4.3)
\end{aligned}
$$

From Equations (4.1) to (4.3), we have $C_i(s) < \alpha C_j(s)$. Now we have the following chain of inequalities.

$$C_j(s) - C_j(s'_j, s_{-j}) \geqslant \varepsilon C_j(s) > \frac{\varepsilon}{\alpha} C_i(s)$$

□

Hence the drop in potential in one iterations is given by

$$\Phi(s) - \Phi(s'_j, s_{-j}) = C_j(s) - C_j(s'_j, s_{-j}) \geqslant \frac{\varepsilon}{\alpha} \max_{i \in [n]} C_i(s) \geqslant \frac{\varepsilon}{\alpha n} \Phi(s)$$

Hence we have $\Phi(s'_j, s_{-j}) \leqslant \left(1 - \frac{\varepsilon}{\alpha n}\right) \Phi(s)$ and thus the potential drops by a constant fraction in $\frac{\alpha n}{\varepsilon}$ iterations. Since the potential function begins at value $\Phi(s^0)$, the max-gain version of the $\varepsilon$-best response dynamics converges to an $\varepsilon$-PSNE in $\mathcal{O}\left(\frac{n\alpha}{\varepsilon} \log \frac{\Phi(s^0)}{\Phi(s^{\min})}\right)$ iterations. □

We can immediately derive the following important result from Theorem 4.2.

**Polynomial Time Computability of $\varepsilon$-PSNE for Network Congestion Games**

**Corollary 4.2.** In the setting as in Theorem 4.2, we can compute an $\varepsilon$-PSNE in time polynomial in $n, N, \alpha, \frac{1}{\varepsilon}$, and $\log \frac{\Phi(s^0)}{\Phi(s^{\min})}$.

*Proof.* Follows from the fact that every iteration of the max-gain variant of the $\varepsilon$-best response dynamics can be computed in polynomial time. Indeed, for every player, his/her best alternative move can be computed in polynomial time using any single source shortest path algorithm (Dijkstra's algorithm for example) in an appropriate graph. □

The $\varepsilon$-best response dynamics based algorithm for computing PSNE in network congestion games (under certain assumptions) can be seen as computing an approximate minimizer of the potential function $\Phi$. Can this result be generalized to any potential game or at least to any congestion game?[2] We will see now that we do not hope to have such a general result.

## 4.5 Local Search Problem

A local search problem is to find a local optimum solution for an optimization problem. Johnson, Papadimitriou, and Yannakakis [JPY88] initiated the complexity-theoretic study of local search problems. A canonical local search problem is the maximum cut problem. In the maximum cut problem, we are given a weighted graph $G = (V, E)$ and we wish to compute a partition $(S, \overline{S})$ of the set $V$ of vertices such that $w(S, \overline{S}) = \sum_{e \in \delta(S)} w_e$ is maximized. Computing the maximum cut of a graph is known to be NP-hard.

Local search heuristics often turn out to be useful for many NP-hard problems. For the maximum cut problem, a natural heuristic would be the following.

[2] It can be shown using almost the same potential function that congestion games are also potential games.

**Local Search for Maximum Cut**

- ▷ Let $(S, \overline{S})$ be any arbitrary cut
- ▷ While there exists a vertex $v$ whose movement can increase the cut value
  - Shift $v$ to the other side

Since the maximum cut problem is NP-hard, we do not expect the above local search for the maximum cut problem to reach a "global optimum" in polynomial time. However, can we expect that the above local search finds a "local optimum" in polynomial time? A local optimum is a solution that can not be improved by local moves only. Obviously, if the input graph is unweighted, then the above local search terminates at a local optimum in polynomial time. We will now learn the complexity-theoretic framework which would indicate that we do not expect to have a polynomial-time algorithm to find a local optimum for the maximum cut problem for weighted graphs.

## 4.6 Complexity Class: Polynomial Local Search (PLS)

We intend to show that if we can solve the local search problem for the maximum cut problem (that is, find a local optimum), then we can solve "many other" interesting local search problems for which we do not have a polynomial-time algorithm even after decades of effort. Towards that, we need to define a local search problem abstractly (like in the classical complexity-theoretic framework, we define decision problems as language membership questions). An *abstract local search* problem is defined by three polynomial-time algorithms.

(i) An algorithm to pick an initial solution.

(ii) An algorithm to compute the value of a solution.

(iii) An algorithm that says whether a solution is a local optimum and if not then executes a "local move" to improve the value of the solution.

The complexity class PLS is the set of all abstract local search problems. Thus, by definition, every problem in PLS admits a local search algorithm although that algorithm may not run in polynomial time. What we are interested to know if there exists a polynomial-time algorithm which finds a local optimum of a given local search problem. Loosely speaking, most of the problems that we encounter can be cast as a local search problem and thus the corresponding local search versions belong to the complexity class PLS. Observe that, given any optimization problem, depending on how we define local move, we can have different local search versions of the problem.

## 4.7 PLS-Completeness

The notion of PLS-completeness characterizes the hardest problems in the class PLS. A problem $\mathcal{P}$ is PLS complete if the problem $\mathcal{P}$ belongs to the class PLS and every problem in PLS "reduces" to $\mathcal{P}$.

**Reduction in PLS Class**

**Definition 4.3.** We say that a PLS problem $\mathcal{P}_1$ reduces to another PLS problem $\mathcal{P}_2$ if we have the following two polynomial-time algorithms.

(i) An algorithm $\mathcal{A}$ that maps every instance $x$ of $\mathcal{P}_1$ to an instance $\mathcal{A}(x)$ of $\mathcal{P}_2$.

(ii) An algorithm $\mathcal{B}$ that maps every local optimum of $\mathcal{A}(x)$ to a local optimum of $x$.

Observe that the definition of reduction ensures that if we can solve the problem $\mathcal{P}_2$ in polynomial time, then we can solve the problem $\mathcal{P}_1$ also in polynomial time. It is known that the (discussed above) local search version of the maximum cut problem is PLS-complete [Sch91].

## 4.8 PSNE Computation in Congestion Games

We will now show that finding a PSNE in a congestion game is PLS-complete.

**Computing PSNE in Congestion Games**

**Theorem 4.3.** Finding a PSNE in a congestion game is PLS-complete.

*Proof.* The problem of finding a PSNE of a congestion game clearly belongs to the class PLS since congestion games are potential games. To prove PLS-hardness, we reduce from the local search version of the maximum cut problem. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ be an arbitrary instance of maximum cut. Consider the following congestion game.

- ▷ The players are the vertices of $\mathcal{G}$.
- ▷ The set of resources: $\{r_e, \bar{r}_e : e \in \mathcal{E}\}$
- ▷ Strategy set of player $v$ : $\{\{r_e : e \in \delta(v)\}, \{\bar{r}_e : e \in \delta(v)\}\}$
- ▷ Cost of the resource $r_e$ or $\bar{r}_e$ is 0 if less than two players use it and $w_e$ otherwise.

The above construction is the algorithm $\mathcal{A}$ in Definition 4.3. We now describe how to map a PSNE of the constructed congestion game to a local optimum of the maximum cut. Before presenting the algorithm, let us explain the high level idea first. Recall the potential function

$$\Phi(s) = \sum_{e \in \mathcal{E}} \sum_{i=1}^{f_e} c_e(i), \text{ where } f_e \text{ is the number of players using resource } e$$

Each player has only 2 strategies to choose from – these strategies correspond to the side of the cut the corresponding vertex belongs. For the player corresponding to the vertex $v$, playing $\{r_e : e \in \delta(v)\}$ corresponds to $v$ belonging to $\mathcal{S}$ and playing $\{\bar{r}_e : e \in \delta(v)\}$ corresponds to $v$ belonging to $\bar{\mathcal{S}}$. We next observe that, for every resource $e = \{u, v\}$, the resources $\{r_e, \bar{r}_e\}$ combined will always be used exactly twice (by players corresponding to $u$ and $v$). The cost function is designed in such a way that an edge $e$ contributes 0 to the potential $\Phi(s)$ if $e$ belongs to the cut and $w_e$ otherwise. We can see that a cut of weight $w(\mathcal{S}, \bar{\mathcal{S}})$

corresponds to the value of the potential function being $\sum_{e \in \mathcal{E}} w_e - w(\mathcal{S}, \overline{\mathcal{S}})$ – hence a local maximum cut corresponds to a strategy profile where no local move (that is, move by any one player) can decrease the potential further. Hence the algorithm for mapping a solution of local maximum cut to a PSNE of the congestion game goes as follows. Let $s$ be a PSNE of the congestion game constructed above. We define $\mathcal{S} = \{v \in \mathcal{V} :$ the strategy of $v$ in $s$ is $\{r_e : e \in \delta(v)\}\}$. From the discussion above, it follows that $(\mathcal{S}, \overline{\mathcal{S}})$ is a solution of local search version of the maximum cut problem. □

The proof of Theorem 4.3 actually shows more – it establishes a bijection between local moves in the maximum cut problem and the moves in best response dynamics. Since we know that local search can take exponentially many steps to reach a local optimum solution, we have the following immediate corollary.

**Best Response Dynamics for Congestion Game**

**Corollary 4.3.** Best response dynamics can take exponentially (in the sum of the number of strategies of the players) many steps to reach a PSNE for congestion games.

The result in Theorem 4.3 can be extended to symmetric congestion games too by reducing any arbitrary congestion game to a symmetric congestion game. A congestion game is called *symmetric* if the set of strategies available to each player is the same.

**Computing PSNE in Symmetric Congestion Game**

**Theorem 4.4.** Finding a PSNE in a symmetric congestion game is PLS-complete.

*Proof.* Again the problem of finding a PSNE in a symmetric congestion game belongs to PLS due to best response dynamics. To show PLS-hardness, we reduce from the problem of finding a PSNE in arbitrary congestion game $\Gamma$. Suppose we have $n$ players where the strategy set of the player $i$ is $\mathcal{S}_i$ for $i \in [n]$ in $\Gamma$. We construct a symmetric congestion game $\Gamma'$ where the set of players remains the same as $\Gamma$. In $\Gamma'$, we have a set $\{r_i : i \in [n]\}$ of $n$ new resources; that is the set of resources in $\Gamma'$ is $\mathcal{E} \cup \{r_i : i \in [n]\}$. The idea is to augment resource $r_i$ with every strategy in $\mathcal{S}_i$. That is the strategy set of every player is $\{s_i \cup \{r_i\} : s_i \in \mathcal{S}_i, i \in [n]\}$. The cost of every resource in $\mathcal{E}$ remains the same as in $\Gamma$. The cost of $r_i$ is 0 if at most one player uses it and $\infty$ if more than one player uses it for every $i \in [n]$. We observe that in any PSNE of $\Gamma'$, every new resource $r_i, i \in [n]$ will be used by exactly one player –intuitively speaking, the player who uses the resource $r_i$ simulates the $i$-th player of the original game $\Gamma$. Formally, given a PSNE $s'$ of $\Gamma'$, we are supposed to map it to a PSNE $s$ of $\Gamma$ and this is how we do it – if $s_i \cup \{r_i\}$ is a strategy played in $s'$, then player $i$ plays $s_i$ in $s$. We can see that the value of the potential function is the same for both the games and thus the PSNEs of $\Gamma$ and $\Gamma'$ are in one-to-one correspondence. □

Again the proof of Theorem 4.4 establishes a bijection between the moves in best response dynamics of corresponding games. Thus we conclude the following.

**Best Response Dynamics for Symmetric Congestion Game**

**Corollary 4.4.** Best response dynamics can take exponentially (in the sum of the number of strategies of the players) many steps to reach a PSNE even for symmetric congestion games.

We now return to the question of computing an MSNE of a bimatrix game and see why we do not expect to have a polynomial-time algorithm for this problem.

## 4.9 Complexity Class: Functional NP

We observe that the problems in the class NP are decision problems – the output should be either YES or NO– whereas the problem of computing an MSNE is not a decision problem. An obvious decision version of the problem is to decide whether a given bimatrix game has an MSNE whose answer is always YES due to Nash Theorem. One way around this issue is to ask little more than what Nash Theorem guarantees. Researchers have indeed study this direction and the computational task of answering the following questions is known to be NP-complete even for symmetric games (the list is no way exhaustive) [GZ89]:

(i) do there exist at least two MSNE?

(ii) does there exist an MSNE in which player 1 receives certain given payoff?

(iii) does there exist an MSNE in which the sum of payoffs of both the players is some given value?

(iv) does there exist an MSNE where the support size of player 1 is at least some given number?

(v) does there exist an MSNE where the support of player 1 contains some given strategy?

(vi) does there exist an MSNE where the support of player 1 does not contain some given strategy?

However, the above results do not shed any light on the complexity of the original problem of computing an MSNE. Towards that, we define the complexity class "Functional NP " or FNP for short. The class FNP consists of all problems in NP with an extra demand that one needs to exhibit a certificate also for YES instances. That is, an algorithm for an FNP problem can either say that the instance is a NO instance or, if it says that the instance is a YES instance, then the algorithm must output a certificate also. Abstractly, the problems in NP are modeled by languages whereas the problems in FNP are modeled by relations. A binary relation $\mathcal{P}(x, y)$ where the length of $y$ is polynomially bounded in the length of $x$ belongs to the class FNP if there is a deterministic polynomial-time verifier algorithm (Turing machine more concretely) that can determine whether a given $x$ is related to a given $y$. The proof of Cook-Levin theorem shows that the functional version of SAT, call it *functional SAT*, (where we demand a satisfying assignment too in the output if the instance is a YES instance) is complete for the FNP class too. Problems in FNP are also called search problems. The class PLS is a sub-class of FNP since every problem in PLS has a polynomial-time algorithm that determines whether a given solution is locally optimum for a given problem instance. The certificates of a problem in PLS are its local optimum.

It follows from the definition of FNP that the problem of computing an MSNE of a bimatrix game belongs to the class FNP since it is enough to check for deviations in pure strategies. Can we prove that MSNE

computation is FNP-complete[3] to settle its complexity? Unfortunately NO unless we believe NP $=$ co $-$ NP [MP91].

**MSNE Computation is unlikely to be FNP-complete**

**Theorem 4.5.** If the problem of computing an MSNE of a bimatrix game is FNP-complete, then NP $=$ co $-$ NP.

*Proof.* We observe that it is enough to prove that NP $\subseteq$ co $-$ NP (why?) which we show below. Suppose the problem of computing an MSNE of a bimatrix game is FNP-complete. Then there is a reduction from functional SAT to MSNE computation for bimatrix games. This implies the existence of the algorithms $\mathcal{A}$ and $\mathcal{B}$ which does the following for any instance $x$ of SAT:

- ▷ Algorithm $\mathcal{A}$ maps $x$ to a bimatrix game, call it $\mathcal{A}(x)$.
- ▷ Algorithm $\mathcal{B}$ maps every MSNE $s^*$ to either of the following:
  - a satisfying assignment $\mathcal{B}(s^*)$ of $x$.
  - NO.

We observe that, existence of $\mathcal{A}$ and $\mathcal{B}$ implies that every SAT instance has a polynomial sized certificate for NO instances which can be verified by a polynomial-time verifier: for any NO instance $x$ of SAT, the certificate is the bimatrix game $\mathcal{A}(x)$ along with an MSNE $s^*$ of the game; the verifier is the algorithm $\mathcal{B}$. Hence SAT belongs to co-NP. Hence, we have NP $\subseteq$ co $-$ NP. □

Intuitively speaking, the hurdle in proving that the problem of computing an MSNE is FNP-complete, is that problems in FNP may have NO instances which is never the case for MSNE computation. This motivates us to define a subclass of FNP which consists of only those search problems where the existence of at least one solution is guaranteed.

## 4.10 Complexity Class TFNP: NP Search Problems with Guaranteed Witness

The complexity class TFNP consists of all problems in FNP for which there is always a solution. By definition, MSNE computation for bimatrix games belongs to TFNP. Examples of other important problems in TFNP are factoring integers, etc. Observe that every problem in PLS also belongs to TFNP. The proof of Theorem 4.5 actually shows more: if any problem in TFNP is FNP-complete, then we have NP $=$ co $-$ NP. Can we show that MSNE computation is TFNP-complete? The answer is again NO and the reason is completely different: while common complexity classes like NP, co-NP, FNP, PSPACE, etc. are "syntactic" complexity class, TFNP is a "semantic" complexity class! What do we mean by "syntactic" vs "semantic" complexity classes? For any syntactic complexity class, we define that a problem belongs to the class if there is a certain kind of Turing machine (polynomial time for example) which accepts some subset of inputs. On a high level, membership can be defined abstractly. For any semantic complexity class, we do not know if this is the case. More

[3]The notion of reduction for the class FNP is exactly the same as the class PLS.

concretely, the problem of computing an MSNE in a bimatrix game belongs to TFNP because of some result from real analysis, factoring numbers belongs to TFNP because of some number-theoretic reason, etc.

## 4.11 Complexity Class PPAD: A Syntactic Subclass of TFNP

We want to define a syntactic subclass of TFNP which would contain the MSNE computation problem. Recall the success story of the class PLS in characterizing complexity of the problem of computing PSNE for important class of games like network congestion games, congestion games, etc.[4] This motivates us to define the complexity class PPAD (stands for Polynomial Parity Arguments on Directed graphs) along the lines of PLS. The problems in PLS can be thought of as traversing a directed acyclic graph (call it search graph) whose vertices correspond to solutions of the problem and edges lead to the next solution. The goal there is to find a sink vertex of that graph. The difference between problems in PLS and PPAD are (i) the search graph need not be an acyclic one, and (ii) there need not be any value associated with solutions. The class PPAD consists of every search problem which admits the following two algorithms.

(i) An algorithm to pick an initial solution.

(ii) Given an "intermediate solution", an algorithm to return a next "intermediate solution" or output that the current solution is a local optimum.

Although it is not immediately clear, there exists an algorithm for computing an MSNE for bimatrix games which traverses a search graph (indeed all state of the art algorithms e.g. Lemke-Howson algorithm traverses certain search spaces which are the edges of the best response polytope). Hence, we conclude that MSNE computation is in PPAD. Can we prove MSNE computation to be PPAD-complete? YES!

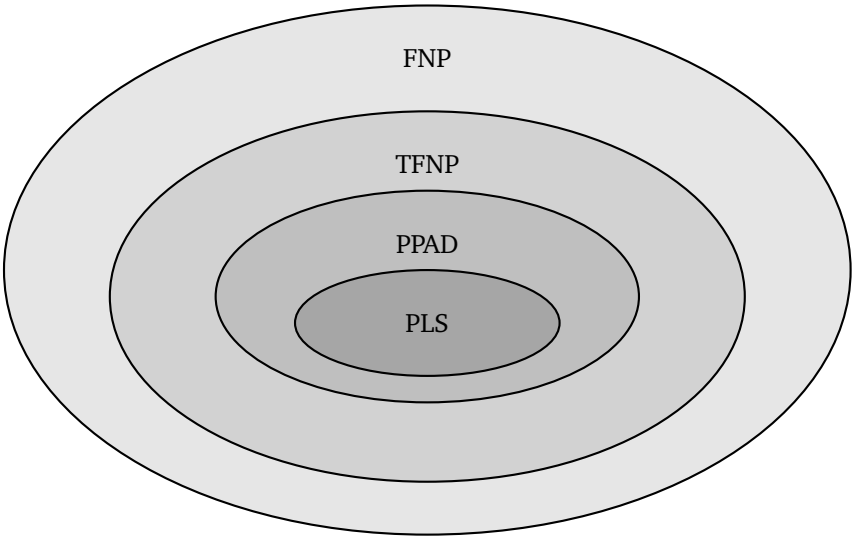



Figure 4.1: Pictorial depiction of relationship among various complexity classes of search problems.

[4]For any generic non-succinct game, because of large input size, the computational task of PSNE can always be done in polynomial time and thus was always settled.

## 4.12 A Canonical PPAD-complete Problem: Sperner's Lemma

We now discuss a computational problem that belongs to the class PPAD more clearly. The problem is on what is known as Sperner's Lemma which turns out to be the heart of the proof of Nash Theorem. Consider a subdivided simplex in a 2 dimensional plane. We color the vertices of this subdivided simplex as follows: the top vertex is colored red, the left vertex is colored green, and the right vertex is colored blue; the vertices in the edges can be colored with any color of its endpoints; the internal vertices can be colored arbitrarily using any of red, green, or blue. Refer to Figure 4.2 for an example of legal coloring. A triangle is called *trichromatic* if all its 3 vertices have different colors.

Sperner's Lemma says that, in every legal coloring of subdivided simplex, there is an odd number of (and thus at least one) trichromatic triangles. We now present a constructive proof for Sperner's lemma. We construct an undirected graph $\mathcal{G}$ whose vertices correspond to the triangles which do not have further subdivision. The graph $\mathcal{G}$ also has a "special vertex" $s$ which corresponds to the outside (unbounded) region. We have an edge between two non-special vertices in $\mathcal{G}$ if and only if their corresponding triangles share an edge whose end points are colored red and green. The special vertex $s$ has an edge with a vertex $x$ if the traingle corresponds to $x$ has one boundary edge (boundary edges are edges which are part of the edges of the outermost traingle) whose end points are colored red and green. We observe that the degree of $s$ is an odd integer. Also the degree of every vertex whose corresponding triangle is trichromatic, is 1 because a trichromatic traingle has exactly red-green edge. Also, we observe that the degree of every non-special vertices is either 0 or 1 or 2. Hence, the degree of a non-special vertex is an odd integer if and only if its corresponding triangle is trichromatic. Now the result follows from the *handshaking lemma* in graph theory which states that the number of vertices having an odd degree is an even integer. The above proof can be generalized inductively to $n$ dimensions using $(n+1)$ colors.

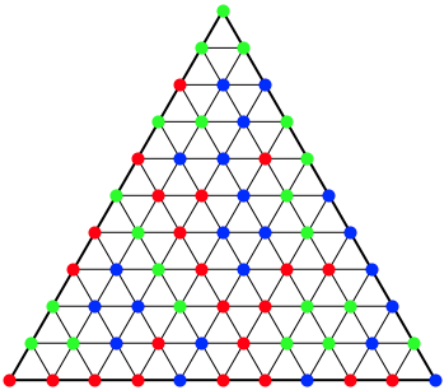


Figure 4.2: A legal coloring of a triangulation. The picture is taken from Google images.

The *Sperner's problem* is the following: given an oracle access to a coloring of vertices of an $n$-dimensional simplex (observe that, coloring is not given explicitly as input), find a $(n+1)$−chromatic simplex.

The proof of Sperner's lemma can be turned into a naïve path following algorithm to compute a trichromatic triangle of any subdivision by directing the edges in $\mathcal{G}$ appropriately. This proves that the Sperner's problem is in PPAD. It turns out that the above problem is PPAD-complete. The following result is one of the cornerstone results in Algorithmic Game Theory.

**MSNE is PPAD-complete**

**Theorem 4.6.** Computing an MSNE of a bimatrix game is PPAD-complete.

We omit the proof of Theorem 4.6 since it is quite technical. We refer interested readers to [Rou10, Section 4], The proof of Theorem 4.6 goes by reducing Sperner's problem to what is known as Brower's problem and then reducing Brower's problem to the problem of computing an MSNE for bimatrix games. Brower's Theorem states that any continuous function $f : C \longrightarrow C$ mapping a compact set $C \subset \mathbb{R}^n$ to itself has at least one fixed point: a $x \in C$ such that $f(x) = x$. However, since C can be an uncountably infinite set (which is indeed the case most often), the above line of reasoning proves that the problem of computing an $\varepsilon$-MSNE (where unilateral deviation cannot increase the payoff of any player by more than $\varepsilon$) is PPAD complete. For $\varepsilon$-MSNE to make any sense, we assume that the payoff of both the players in every strategy profile is a real number in $[0, 1]$ which we can assume without loss of generality by adding and then multiplying each payoff values by suitable constants and this does not change/introduce any MSNE (why?).

For 3 or more player, the proof for PPAD-hardness goes as it is since other players can act as dummies (for example, their payoffs may be a constant function). However, the Lemke-Howson line of argument for proving membership in PPAD does not go through for 3 or more players and the problem of computing an exact MSNE for games with 3 or more players seems to be strictly harder [EY10].

## 4.13 Algorithms for $\varepsilon$-MSNE

We end this chapter by describing some results on computing an $\varepsilon$-MSNE of a bimatrix game.

### 4.13.1 A Simple Polynomial Time Algorithm for $\frac{1}{2}$ Approximate MSNE

We first present a simple $\frac{1}{2}$ approximate MSNE for bimatrix games [DMP06]. The algorithm goes as follows.

Let $i \in [m]$ be any arbitrary pure strategy of player 1. Let $j \in [n]$ be a best response pure strategy for player 2 against $i$. Let $k \in [m]$ be a best response pure strategy of player 1 against $j$. We define a mixed strategy $x^*$ of player 1 as $i$ with probability $\frac{1}{2}$ and $j$ with probability $\frac{1}{2}$. We claim that $(x^*, j)$ is a $\frac{1}{2}$ approximate MSNE for the given bimatrix game. Let $\mathcal{A}$ and $\mathcal{B}$ be $m \times n$ payoff matrices of player 1 and 2 respectively. Then we have the following for any pure strategy $x \in [m]$ and $y \in [n]$ of row and column players respectively.

$$u_1(x, j) = x\mathcal{A}e_j \leqslant e_k\mathcal{A}e_j \leqslant \frac{1}{2}e_i\mathcal{A}e_j + \frac{1}{2}e_k\mathcal{A}e_j + \frac{1}{2} = u_1(x^*, j) + \frac{1}{2}$$

$$u_2(x^*, y) = \frac{1}{2}e_i\mathcal{A}y + \frac{1}{2}e_k\mathcal{A}y \leqslant \frac{1}{2}e_i\mathcal{A}e_j + \frac{1}{2}e_k\mathcal{A}y \leqslant \frac{1}{2}e_i\mathcal{A}e_j + \frac{1}{2} \leqslant \frac{1}{2}e_i\mathcal{A}e_j + \frac{1}{2}e_k\mathcal{A}e_j + \frac{1}{2} = u_2(x^*, j) + \frac{1}{2}$$

# Chapter 5

# Correlated and Coarse Correlated Equilibrium

We have seen in the last chapter that computing an MSNE even for a bimatrix game may be a computationally challenging task. Even computing a PSNE for various succinct games e.g. congestion games seems to be a computationally intractable task. These results cast doubt on the usefulness of the concept of Nash equilibrium as a tool for predicting outcomes – if computing equilibriums are so hard, how do we expect the players to discover it! This motivates us to relax the requirements of Nash equilibrium even further and develop some computationally tractable notions of equilibrium.

## 5.1 Correlated Equilibrium (CE)

Intuitively, a probability distribution $\mu$ over strategy profiles is called a correlated equilibrium if no player has any incentive to deviate even after knowing his/her pure strategy in the sampled strategy profile from $\mu$. It is defined formally as follows.

**Correlated Equilibrium (CE)**

**Definition 5.1.** Given a game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, a probability distribution $\sigma$ over $S = \prod_{i\in N} S_i$ is called a correlated equilibrium if for every $i \in N$, we have the following for every $s_i, s_i' \in S_i$.

$$\mathbb{E}_{s\sim\sigma}[u_i(s)|s_i] \geqslant \mathbb{E}_{s\sim\sigma}[u_i(s_i', s_{-i})|s_i]$$

That is

$$\sum_{s\in S} \sigma(s|s_i)u_i(s) \geqslant \sum_{s\in S} \sigma(s|s_i)u_i(s_i', s_{-i})$$

Alternatively, a probability distribution $\sigma$ over $S = \prod_{i\in N} S_i$ is called a correlated equilibrium if for every $i \in N$, we have the following for every (switching) function $\delta_i : S_i \longrightarrow S_i$.

$$\mathbb{E}_{s\sim\sigma}[u_i(s)] \geqslant \mathbb{E}_{s\sim\sigma}[u_i(\delta_i(s_i), s_{-i})]$$

Note that the distribution $\sigma$ is over the strategy profiles. Any MSNE is also a correlated equilibrium where the distribution $\sigma$ is a product distribution. Hence, it follows from Nash Theorem that every finite game has at least one correlated equilibrium.

One usual interpretation of correlated equilibrium is the following: suppose players agree that they will collectively play according to some distribution $\sigma$ over the set of strategy profiles. Some trusted third party samples a strategy profile $(s_i)_{i\in N}$ from the distribution $\sigma$. The trusted third party conveys each player $i$ his/her strategy $s_i$ in the sampled strategy profile. Each player $i$ has two options at the time of playing – either to follow the suggestion of the trusted third party and play $s_i$ or to not follow the suggestion of the trusted third party and play some other strategy $s_i'$. At the time of playing, each player $i$ thus knows two things – (i) the strategy $s_i$ suggested to him/her (ii) a posterior distribution on $s_{-i}$ derived from $\sigma$ and $s_i$. The requirement for $\sigma$ to be a correlated equilibrium is that every player cannot improve his/her expected utility by deviating to any other strategy $s_i'$.

**Example 5.1.1** (Traffic Light)**.** *Let us consider the following situation of two players wanting to cross a traffic signal. If both of them go together, then there will be accident. But any one (not both) of them can go.*

▷ *The set of players* $(N) : \{1, 2\}$

▷ *The set of strategies:* $S_i = \{$*Stop, Go*$\}$ *for every* $i \in [2]$

▷ *Utility matrix:*

| | | *Player* 2 | |
|---|---|---|---|
| | | *Stop* | *Go* |
| *Player* 1 | *Stop* | $(0, 0)$ | $(0, 1)$ |
| | *Go* | $(1, 0)$ | $(-100, -100)$ |

Let us understand the concept of CE with the example game of traffic lights. The game has two PSNEs – (Stop, Go) and (Go, Stop). The game also has a CE $\sigma$ which assigns the strategy profiles (Stop, Go) and (Go, Stop) each a probability of $\frac{1}{2}$. Clearly $\sigma$ is not a product distribution (the reader is encouraged to prove it formally) and thus does not correspond to any MSNE. One can observe that the requirement in Definition 5.1 can be written as a linear program and thus a CE can be computed in polynomial time. Given a game $\Gamma = \langle N = [n], (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, to write an LP for CE, we introduce a variable $x_s$ for every strategy profile $s \in S = \prod_{i\in N} S_i$. Then any solution to the following LP is a CE.

$$x_s \geqslant 0, \quad \forall\, s \in S$$
$$\sum_{s\in S} x_s = 1$$
$$\sum_{(s_i,s_{-i})\in S} x_{(s_i,s_{-i})} u_i(s_i, s_{-i}) \geqslant \sum_{(s_i,s_{-i})\in S} x_{(s_i,s_{-i})} u_i(s_i', s_{-i}), \quad \forall\, i \in N, s_i, s_i' \in S_i$$

Figure 5.1: Linear program to compute CE.

## 5.2 Coarse Correlated Equilibrium (CCE)

The requirement for CE can be relaxed further to define what is called a Coarse Correlated Equilibrium (CCE).

**Coarse Correlated Equilibrium (CCE)**

**Definition 5.2.** Given a game $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$, a probability distribution $\sigma$ over $S = \prod_{i\in N} S_i$ is called a coarse correlated equilibrium if, for every $i \in N$, we have the following for every $s'_i \in S_i$.

$$\mathbb{E}_{s\sim\sigma}[u_i(s)] \geqslant \mathbb{E}_{s\sim\sigma}[u_i(s'_i, s_{-i})]$$

That is

$$\sum_{s\in S} \sigma(s)u_i(s) \geqslant \sum_{s\in S} \sigma(s)u_i(s'_i, s_{-i})$$

The following LP captures the requirement of CCE.

$$\begin{gathered} x_s \geqslant 0, \quad \forall\, s \in S \\ \textstyle\sum_{s\in S} x_s = 1 \\ \textstyle\sum_{s\in S} x_s u_i(s) \geqslant \sum_{s\in S} x_s u_i(s'_i, s_{-i}), \quad \forall\, i \in N, s'_i \in S_i \end{gathered}$$

Figure 5.2: Linear program to compute CCE.

Intuitively speaking, while CE safe-guards against conditional (since the player still has the option of deviating after knowing its strategy) unilateral deviation by any player, CCE safe-guards against unconditional (since the player does not have the option of deviating after knowing its strategy) unilateral deviation by any player. Another interpretation of CE and CCE is using non-binding and binding contracts. Think of players signing a contract to play according to some distribution $\sigma$ over the set of strategy profiles. The notion of CE captures the dynamics of non-binding contract – any player is allowed to deviate even after knowing what realization of strategy it is suggested to play. Whereas CCE models binding contracts – no player is allowed to deviate from his/her suggested strategy after they agree on $\sigma$. We leave it to readers to verify that every CE is also a CCE. The intuitive reason for this is if players do not have any incentive to deviate even after knowing the realization of his/her strategy, then he/she does not have any incentive to deviate in the first place (before knowing his/her realized strategy). Hence, it follows that every finite strategy form game has at least one CCE and it can be found in polynomial time. However, we will see that the algorithm for finding a CCE is much simpler and natural than the algorithm for finding a CE (linear programming is not considered "natural" since "normal" human being does not take decision by solving linear programs ☺).

## 5.3 External Regret Framework

Let us take a diversion from equilibrium concepts and study the following *regret-minimization problem*. We will later see the connection with this problem and the equilibrium concepts that we have studied so far. Let $\mathcal{A}$ be any set of $n \geqslant 2$ actions. The external regret framework is the following.

▷ In every time step $t = 1, 2, \ldots, T$,

- We pick a mixed strategy $p_t$ which is a probability distribution over $\mathcal{A}$.
- An adversary picks a utility function $\pi_t : \mathcal{A} \longrightarrow [0, 1]$.
- An action $a_t$ is sampled from the distribution $p_t$. Our utility is $\pi_t(a_t)$ and we come to know the utility function $\pi_t$.

The above framework of external regret models various real world scenarios like network routing (where $\mathcal{A}$ is the set of paths from a source to a destination), investment strategies (where $\mathcal{A}$ is the set of investment strategies we have), etc. The adversary above is called an *adaptive adversary* since before choosing the utility function $\pi_t$ on t-th day, it knows everything that happened so far, that is, $p_1, \ldots, p_t$ and $a_1, \ldots, a_{t-1}$. An adversary is called *oblivious* if $\pi_t$ depends only on $t$ and (of course) the algorithm under consideration; specifically, it does not depend on $p_1, \ldots, p_t$ and $a_1, \ldots, a_{t-1}$.

Given the external regret framework, how "good" we expect to perform. To study this, we need to fix a benchmark $\mathcal{B}$. Our goal is to achieve time averaged regret $\frac{1}{T}(\sum_{t=1}^{T} \sum_{a_t \in \mathcal{A}} p_t(a_t)\pi_t(a_t) - \mathcal{B})$ to go to 0 as $T$ goes to $\infty$ – we call such an algorithm a *no-regret algorithm*. One obvious benchmark is the "best action sequence" – compare the performance of our algorithm with $\sum_{t=1}^{T} \max_{a_t \in \mathcal{A}} \pi_t(a_t)$. However, the following adversary shows that we do not hope to achieve our goal even when $|\mathcal{A}| = 2$. If $p_t$ puts at least a probability of $1/2$ to strategy 1 (respectively strategy 2), then the adversary sets the utility of strategy 1 (respectively strategy 2) to be 0 and the utility of strategy 2 (respectively strategy 1) to be 1. Obviously we have $\sum_{t=1}^{T} \max_{a_t \in \mathcal{A}} \pi_t(a_t) = T$ where as $\sum_{t=1}^{T} \sum_{a_t \in \mathcal{A}} p_t(a_t)\pi_t(a_t) \leqslant T/2$. Hence the time averaged regret is at least $1/2$. The intuitive reason for this strong lower bound is two fold: (i) the adversary has too much power, (ii) the benchmark is too strong. To achieve a time averaged regret of $o_T(1)$ (that is, tends to 0 ad $T$ tends to $\infty$), we weaken a benchmark – we compare with a "best fixed" action. That is, we fix our benchmark to be $\max_{a \in \mathcal{A}} \sum_{t=1}^{T} \pi_t(a)$. We now see that, with respect to this benchmark, there exists a simple and natural no-regret algorithm.

## 5.4 No-Regret Algorithm

The following result is the most important result of this chapter.

**No-Regret Algorithm Exists**

**Theorem 5.1.** Let $|\mathcal{A}| = n$. Then there exists a no-regret algorithm whose expected time averaged regret is $\mathcal{O}\left(\sqrt{\frac{\log n}{T}}\right)$ with respect to every fixed action.

An immediate corollary of Theorem 5.1 is the following.

**Rate of Convergence of No-Regret Algorithm**

**Corollary 5.1.** There exists a no-regret algorithm whose expected time averaged regret is at most $\varepsilon$ with respect to every fixed action after $\mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ iterations for every $\varepsilon > 0$.

We now present the algorithm in Theorem 5.1 which is popularly known as *multiplicative weights (MW)*

algorithm or *randomized weighted majority* or *Hedge* algorithm.

**Algorithm 1** Multiplicative Weight Algorithm

1: $w_0(a) = 1$ for every $a \in \mathcal{A}$
2: **for** $t = 1, \ldots, T$ **do**
3: Play an action $a$ with probability proportional to $w_{t-1}(a)$.
4: Given payoff function $\pi_t$, define $w_t(a) = w_{t-1}(a)(1+\varepsilon)^{\pi_t(a)}$ for every action $a \in \mathcal{A}$.
5: **end for**

## 5.5 Analysis of Multiplicative Weight Algorithm

*Proof of Theorem 5.1.* We now prove that the time-averaged regret of the multiplicative weight algorithm, for appropriate choice of $\varepsilon$, is $\mathcal{O}\left(\sqrt{\frac{\log n}{T}}\right)$ thereby proving Theorem 5.1. We may assume without loss of generality that the adversary we are working against is an oblivious one (how?). Let $u_1, \ldots, u_T$ be the payoff functions in $T$ iterations.

Let $\Gamma_t = \sum_{a \in \mathcal{A}} w_t(a)$ be the sum of weights all the actions in $t$-th iteration for $t \in \{0, 1, \ldots, T\}$. The idea of the proof is to relate both the benchmark $\text{OPT} = \max_{a \in \mathcal{A}} \sum_{t=1}^{T} \pi_t(a)$ and the expected payoff $\sum_{t=1}^{T} \sum_{a \in \mathcal{A}} p_t(a)\pi_t(a)$ of the MW algorithm to $\Gamma_T$. We begin with relating OPT with $\Gamma_T$. Let $a^* \in \mathcal{A}$ be the action such that $\text{OPT} = \sum_{t=1}^{T} \pi_t(a^*)$.

$$\begin{aligned}
\Gamma_T &\geqslant w_T(a^*) \\
&= w_0(a^*) \prod_{t=1}^{T} (1+\varepsilon)^{\pi_t(a^*)} \\
&= (1+\varepsilon)^{\sum_{t=1}^{T} \pi_t(a^*)} && \text{[Since } w_0(a^*) = 1\text{]} \\
&= (1+\varepsilon)^{\text{OPT}} && (5.1)
\end{aligned}$$

We now relate the expected payoff $\sum_{t=1}^{T} \sum_{a \in \mathcal{A}} p_{t-1}(a)\pi_t(a)$ of the MW algorithm to $\Gamma_T$. The payoff in the $t$-th step is given by,

$$\sum_{a \in \mathcal{A}} p_{t-1}(a)\pi_t(a) = \sum_{a \in \mathcal{A}} \frac{w_{t-1}(a)}{\Gamma_{t-1}} \pi_t(a)$$

The idea is to relate $\Gamma_T$ with $\Gamma_0$.

$$\begin{aligned}
\Gamma_T &= \sum_{a \in \mathcal{A}} w_T(a) \\
&= \sum_{a \in \mathcal{A}} w_{T-1}(a)(1+\varepsilon)^{\pi_T(a)} \\
&\leqslant \sum_{a \in \mathcal{A}} w_{T-1}(a)(1+\varepsilon\pi_T(a)) && \text{[Since } \pi_t(a) \in [0,1]\text{]} \\
&= \sum_{a \in \mathcal{A}} w_{T-1}(a) + \Gamma_{T-1}\varepsilon \sum_{a \in \mathcal{A}} \left( \frac{w_{T-1}(a)}{\Gamma_{T-1}} \pi_t(a) \right) \\
&= \Gamma_{T-1}(1+\varepsilon v_T) && \text{[}v_T \text{ is expected payoff in } t\text{-th iteration]}
\end{aligned}$$

$$= \Gamma_0 \prod_{t=1}^{T}(1+\varepsilon v_t)$$
$$= n \prod_{i=t}^{T}(1+\varepsilon v_t) \qquad [\Gamma_0 = |\mathcal{A}| = n]$$

In particular, we have $\Gamma_T \leqslant \Gamma_0 \prod_{t=1}^{T}(1+\varepsilon v_t)$. Now from Equation (5.1), we have the following.

$$\begin{aligned}
(1+\varepsilon)^{OPT} &\leqslant \Gamma_T \\
&\leqslant n \prod_{i=1}^{t}(1+\varepsilon v_i) \\
\Rightarrow OPT \ln(1+\varepsilon) &\leqslant \ln n + \sum_{t=1}^{T} \ln(1+\varepsilon v_t) \\
\Rightarrow OPT\left(\varepsilon - \varepsilon^2\right) &\leqslant \ln n + \sum_{t=1}^{T} \varepsilon v_t \qquad \left[x - x^2 \leqslant \ln(1+x) \leqslant x, \ \ \forall\, x \in [0,1]\right] \\
\Rightarrow OPT\,(1-\varepsilon) &\leqslant \frac{\ln n}{\varepsilon} + \sum_{t=1}^{T} v_t \\
\Rightarrow OPT - \sum_{t=1}^{T} v_t &\leqslant \varepsilon OPT + \frac{\ln n}{\varepsilon} \\
&\leqslant \varepsilon T + \frac{\ln n}{\varepsilon} \qquad [OPT \leqslant T] \\
&\leqslant 2\sqrt{T \ln n} \qquad \left[\text{Put } \varepsilon = \sqrt{\frac{\ln n}{T}}\right] \\
\Rightarrow \frac{1}{T}(OPT - \sum_{t=1}^{T} v_t) &\leqslant 2\sqrt{\frac{\ln n}{T}}
\end{aligned}$$

□

Before we show the connection between no-regret algorithms and various game-theoretic solution concepts, let us make some remarks. In the proof of Theorem 5.1, we have set $\varepsilon = \sqrt{\frac{\ln n}{T}}$ – so we assumed that we know the time horizon $T$ is known a priori. We show next that the same guarantee can be achieved without knowing the time horizon $T$.

**No-Regret Algorithm for Unknown Time Horizon**

**Remark 5.1.** Let $|\mathcal{A}| = n$. Then there exists a no-regret algorithm whose expected time averaged regret is $\mathcal{O}\left(\sqrt{\frac{\log n}{T}}\right)$ with respect to every fixed action. Moreover the algorithm does not know a priori the time horizon $T$.

*Proof of Remark 5.1.* On $t$-th day, set the value of $\varepsilon$ to be $\sqrt{\frac{\ln n}{T_t}}$ where $T_t$ is the smallest power of 2 larger than $t$. If $t$ is a power of 2, then we reset $w_t(a) = 1$ for every action $a \in \mathcal{A}$. Let us define $\ell = \lfloor \lg T \rfloor$. We partition the time interval $[1, T]$ to $\ell$ epochs $\xi_i, i \in \{0, 1, \ldots, \ell\}$ — $\xi_i$ is the time interval $[2^i, \min\{2^{i+1} - 1, T\}]$.

Without loss of generality we may assume that the length of every epoch is some power of 2 (by adding more rounds of play where adversary chooses constant payoff function and this at most doubles $T$ which does not affect the result that we wish to prove). We observe that the value of $\varepsilon$ remains same in an epoch – in epoch $\xi_i$, the value of $\varepsilon$ is $\varepsilon_i = \sqrt{\frac{\ln n}{2^i}}$. Let $\mathrm{OPT}_i$ be the payoff of optimal action with respect to the epoch $\xi_i$ alone for $i \in \{0, 1, \ldots, \ell\}$; that is $\mathrm{OPT}_i = \max_{a \in A} \sum_{t \in \xi_i} \pi_t(a)$. Let $v_t$ be the expected payoff in the $t$-th step for $t \in [T]$. We now have the following.

$$\begin{aligned}
\mathrm{OPT} - \sum_{t=1}^{T} v_t &\leqslant \sum_{i=0}^{\ell} \left( \mathrm{OPT}_i - \sum_{t \in \xi_i} v_t \right) \\
&\leqslant \sum_{i=0}^{\ell} 2\sqrt{2^i \ln n} \\
&= \sqrt{\ln n} \sum_{i=0}^{\ell} 2^{\frac{i}{2}+1} \\
&\leqslant 2^{\frac{\ell}{2}+2} \sqrt{\ln n} \\
&= 4\sqrt{T \ln n}
\end{aligned}$$

□

**Combining Expert Advice**

**Remark 5.2.** The problem of designing a no-regret algorithm is sometimes called *combining expert advice* — at every iteration, the mixed strategy can be thought of as an expert's advice and the goal is to perform as good as the best expert asymptotically.

One can ask if there exists a no-regret algorithm whose time-averaged regret is smaller than $\mathcal{O}\left(\sqrt{\frac{\log n}{T}}\right)$. It turns out that the simple oblivious adversary which picks a cost vector uniformly at random from $\{e_i : i \in [n]\}$ enforces any algorithm to incur a time-averaged regret to be $\Omega\left(\sqrt{\frac{\log n}{T}}\right)$ (can you prove it?).

## 5.6 No-Regret Dynamic

We now see the connection between no-regret algorithm and game theoretic solution concept. Suppose we have a strategic form game $\Gamma = \langle [n], (S_i)_{i \in [n]}, (u_i)_{i \in [n]} \rangle$ with $n$ players. Each player runs any no-regret algorithm. The set of actions available to player $i$ is his/her strategy set $S_i$. Suppose each player $i \in [n]$ chooses a mixed strategy $p_i \in \Delta(S_i)$. The payoff function $\pi_i$ (in the external regret framework) given to player $i$ is his/her expected utility $u_i(s_i, p_{-i})$ for playing a strategy $s_i$ given every other play $j \in [n] \setminus \{i\}$ plays the mixed strategy $p_i$. That is,

$$u_i(s_i, p_{-i}) = \sum_{s_{-i} \in S_{-i}} \left( u_i(s_i, s_{-i}) \prod_{j \in [n], j \neq i} p_j(s_j) \right)$$

This is called *no-regret dynamic*. The following result shows a fundamental connection between no-regret dynamic and coarse correlated equilibrium.

**No-Regret Dynamic and CCE**

**Theorem 5.2.** For $\varepsilon > 0$, suppose after $T$ iterations, the time averaged regret of every player is at most $\varepsilon$ (if every player runs MW algorithm, then $T = \mathcal{O}\left(\frac{\ln n}{\varepsilon^2}\right)$). Let $\sigma_t = \prod_{i=1}^{n} p_i^t$ be the product distribution in $t$-th iteration for $t \in [T]$ and $\sigma = \frac{1}{T}\sum_{t=1}^{T} \sigma_t$ be the time averaged distribution. Then $\sigma$ is an $\varepsilon$-coarse correlated equilibrium of the game $\Gamma$; that is

$$\mathbb{E}_{s\sim\sigma}[u_i(s)] \geqslant \mathbb{E}_{s\sim\sigma}[u_i(s'_i, s_{-i})] - \varepsilon$$

*Proof.* By definition, we have

$$\mathbb{E}_{s\sim\sigma}[u_i(s)] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{s\sim\sigma_i}[u_i(s)]$$

Since the time averaged regret of player $i$ after $T$ iterations is at most $\varepsilon$ with respect to any fixed action, we have the following for every $s'_i \in S_i$.

$$\frac{1}{T}\left(\sum_{t=1}^{T} \mathbb{E}_{(s_i,s_{-i})\sim\sigma_i}[u_i(s'_i, s_{-i})] - \sum_{t=1}^{T} \mathbb{E}_{s\sim\sigma_i}[u_i(s)]\right) \leqslant \varepsilon$$

$$\mathbb{E}_{(s_i,s_{-i})\sim\sigma}[u_i(s'_i, s_{-i})] - \mathbb{E}_{s\sim\sigma}[u_i(s)] \leqslant \varepsilon$$

□

## 5.7 Swap Regret

We have seen that no-regret dynamics converge to coarse correlated equilibrium. Does there exist any such "natural" dynamic which converge to correlated equilibrium? The answer is swap regret. To motivate swap regret, let us view external regret from the perspective of a *modification rule*. A modification rule $F$ takes as input the history of $T$ sequence of plays, i.e. the sequence $p_1, \ldots, p_T$ of probability distributions committed, the sequence $a_1, \ldots, a_T$ of actions played, and the payoff functions $\pi_1, \ldots, \pi_T$, and outputs another sequence $a'_1, \ldots, a'_T$ of actions. A modification rule is called *memoryless* if its output does not depend on the history (i.e. $p_1, \ldots, p_T, a_1, \ldots, a_T, \pi_1, \ldots, \pi_T$). The notion of external regret tries to perform as good as any function in the class $\mathcal{F}^{ex}$ where $\mathcal{F}^{ex}$ is the set of modification rules $F_a, a \in \mathcal{A}$ which outputs the constant sequence $a, \ldots, a$ (there is $n$ such functions). The notion of internal regret sets benchmark as the class $\mathcal{F}^{in}$ consisting of functions $F_{a,b}, a, b \in \mathcal{A}$, which plays $b$ wherever $a$ was played and vice versa and does not disturb other actions (there is $n(n-1)$ such functions). Swap regret sets benchmark as the class $\mathcal{F}^{swap}$ of all switching functions $\delta : \mathcal{A} \longrightarrow \mathcal{A}$ (there are $n^n$ such functions).

**No Swap-Regret**

**Definition 5.3.** An online algorithm is said to have no swap regret if its time averaged expected swap regret

$$\frac{1}{T}\left(\left(\max_{\delta\in\mathcal{F}^{swap}} \sum_{t=1}^{T}\sum_{a\in\mathcal{A}} p_t(a)\pi_t(\delta(a))\right) - \sum_{t=1}^{T}\sum_{a\in\mathcal{A}} p_t(a)\pi_t(a)\right)$$

goes to 0 as $T$ goes to $\infty$.

We show next that if every player runs a no-swap-regret algorithm, then they converge to a correlated equilibrium. The setup is exactly same as before. That is the set of actions available to player $i$ is his/her strategy set $S_i$. Suppose each player $i \in [n]$ chooses a mixed strategy $p_i \in \Delta(S_i)$. The payoff function $\pi_i$ (in the external regret framework) given to player $i$ is his/her expected utility $u_i(s_i, p_{-i})$ for playing a strategy $s_i$, given every other player $j \in [n] \setminus \{i\}$ plays the mixed strategy $p_j$.

**No-Swap-Regret Dynamic and CE**

**Theorem 5.3.** For $\varepsilon > 0$, suppose after $T$ iterations, the time averaged swap regret of every player is at most $\varepsilon$. Let $\sigma_t = \prod_{i=1}^{n} p_i^t$ be the product distribution in $t$-th iteration for $t \in [T]$ and $\sigma = \frac{1}{T}\sum_{t=1}^{T} \sigma_t$ be the time averaged distribution. Then $\sigma$ is an $\varepsilon$-correlated equilibrium of the game $\Gamma$; that is, for every player $i \in N$ and every switching function $\delta_i : S_i \longrightarrow S_i$, we have the following

$$\mathbb{E}_{s\sim\sigma}[u_i(s)] \geqslant \mathbb{E}_{s\sim\sigma}[u_i(\delta_i(s_i), s_{-i})] - \varepsilon$$

*Proof.* By definition, we have

$$\mathbb{E}_{s\sim\sigma}[u_i(s)] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{s\sim\sigma_i}[u_i(s)]$$

Since the time-averaged swap regret of player $i$ after $T$ iterations is at most $\varepsilon$, we have the following for every player $i \in N$ and every switching function $\delta : S_i \longrightarrow S_i$.

$$\begin{aligned}
&\frac{1}{T}\left(\sum_{t=1}^{T}\sum_{a\in S_i} p_i^t(a)\pi_t(\delta(a)) - \sum_{t=1}^{T}\sum_{a\in\mathcal{A}} p_i^t(a)\pi_t(a)\right) \leqslant \varepsilon \\
\Rightarrow &\frac{1}{T}\left(\sum_{t=1}^{T} \mathbb{E}_{s\sim\sigma}[u_i(\delta_i(s_i), s_{-i})] - \sum_{t=1}^{T} \mathbb{E}_{s\sim\sigma_i}[u_i(s)]\right) \leqslant \varepsilon \\
&\Rightarrow \mathbb{E}_{s\sim\sigma}[u_i(\delta_i(s_i), s_{-i})] - \mathbb{E}_{s\sim\sigma}[u_i(s)] \leqslant \varepsilon
\end{aligned}$$

□

So, we have seen that, if players run no-swap-regret dynamics, then the corresponding distribution converges to a correlated equilibrium. However, does there exist a no-swap-regret algorithm? Obviously, every no-swap-regret algorithm is also a no-external-regret algorithm and the opposite is clearly not true. This makes the following result interesting.

## 5.8 Black Box Reduction from Swap Regret to External Regret

We show next that the existence of a no-external-regret algorithm guarantees the existence of a no-swap-regret algorithm.

**Swap Regret to External Regret**

**Theorem 5.4.** Let $n$ be the number of actions in a swap regret setting. Suppose there exists an algorithm with time averaged external-regret $R(T, n)$. Then there exists an algorithm with time averaged swap-regret $nR(T, n)$. In particular, if there exists a no-external-regret algorithm (i.e.

$\lim_{T\to\infty} R(T,n) = 0$), then there exists a no-swap-regret algorithm.

*Proof.* Let the action set be $\mathcal{A} = \{1, 2, \ldots, n\}$. Let $\mathcal{B}$ be an algorithm with time averaged external-regret $R(T,n)$. We take $n$ instances $\mathcal{B}_1, \ldots, \mathcal{B}_n$ of the algorithm $\mathcal{B}$. We build a "master algorithm" $\mathcal{H}$ using $\mathcal{B}_1, \ldots, \mathcal{B}_n$. At every time $t = 1, \ldots, T$, the following happens

1. The master algorithm $\mathcal{H}$ receives distributions $p_1^t, \ldots, p_n^t$ from $\mathcal{B}_1, \ldots, \mathcal{B}_n$ respectively.
2. Using $p_1^t, \ldots, p_n^t$, $\mathcal{H}$ computes a distribution $p^t$ (let the need dictate how it should be done).
3. $\mathcal{H}$ receives a payoff function $\pi_t$ from the adversary.
4. $\mathcal{H}$ provides $p^t(i)\pi_t$ to $\mathcal{B}_i$ as the payoff function for $i \in [n]$.

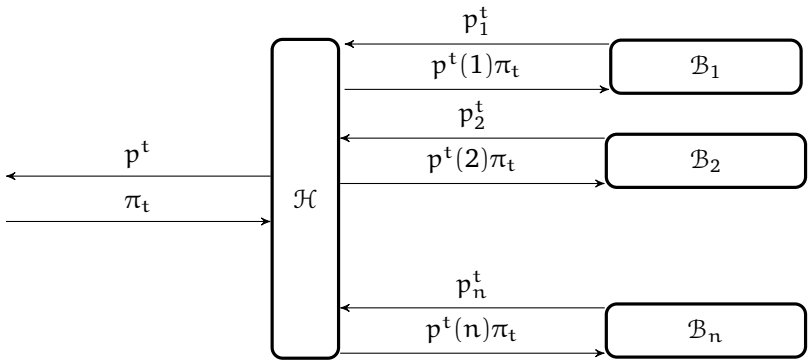



Figure 5.3: Schematic diagram of the master algorithm in the proof of Theorem 5.4.

The time averaged expected payoff of the master algorithm is

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i\in\mathcal{A}} p^t(i)\pi_t(i)$$

Let $\delta : \mathcal{A} \longrightarrow \mathcal{A}$ be any switching function. Then the time averaged expected payoff under this switching function is

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i\in\mathcal{A}} p^t(i)\pi_t(\delta(i))$$

We need to show that the following goes to 0 as $T$ goes to $\infty$.

$$\frac{1}{T}\left(\sum_{t=1}^{T}\sum_{i\in\mathcal{A}} p^t(i)\pi_t(\delta(i)) - \sum_{t=1}^{T}\sum_{i\in\mathcal{A}} p^t(i)\pi_t(i)\right) \tag{5.2}$$

Since each $\mathcal{B}_i$ has external regret $R(T,n)$, we have the following for every action $\lambda \in \mathcal{A}$.

$$\frac{1}{T}\left(\sum_{t=1}^{T} p^t(i)\pi_t(\lambda) - \sum_{t=1}^{T}\sum_{j\in\mathcal{A}} p_i^t(j)p^t(i)\pi_t(j)\right) \leqslant R(T,n)$$

We put $\lambda = \delta(i)$ in the above inequality.

$$\frac{1}{T}\left(\sum_{t=1}^{T} p^t(i)\pi_t(\delta(i)) - \sum_{t=1}^{T}\sum_{j\in\mathcal{A}} p_i^t(j)p^t(i)\pi_t(j)\right) \leqslant R(T,n)$$

We add the above inequality for all the instantiations $\mathcal{B}_i, i \in \mathcal{A}$.

$$\frac{1}{T}\left(\sum_{t=1}^{T}\sum_{i\in\mathcal{A}} p^t(i)\pi_t(\delta(i)) - \sum_{t=1}^{T}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}} p_i^t(j)p^t(i)\pi_t(j)\right) \leqslant nR(T,n)$$

Exchanging the role of $i$ and $j$ in the second term, we obtain

$$\frac{1}{T}\left(\sum_{t=1}^{T}\sum_{i\in\mathcal{A}} p^t(i)\pi_t(\delta(i)) - \sum_{t=1}^{T}\sum_{i\in\mathcal{A}} \pi_t(i)\sum_{j\in\mathcal{A}} p_j^t(i)p^t(j)\right) \leqslant nR(T,n) \tag{5.3}$$

From Equation (5.2) and Section 5.8, it is obvious how we should set the consensus distribution $p^t$ from $p_1^t, \ldots, p_n^t$.

$$p^t(i) = \sum_{j\in\mathcal{A}} p_j^t(i)p^t(j), \quad \forall\, i \in \mathcal{A} \tag{5.4}$$

And we of course have

$$\sum_{i\in\mathcal{A}} p^t(i) = 1 \tag{5.5}$$

Since Equations (5.4) and (5.5) form a steady-state equation of a finite Markov chain (for finding a stationary distribution $(p^t(i))_{i\in\mathcal{A}}$), there is a solution to the system of Equations (5.4) and (5.5) — observe that $p_j^t(i)$ is a constant for every $i, j \in \mathcal{A}$ as we need to compute the distribution $p^t$ given $p_j^t$ for every $j \in \mathcal{A}$. This concludes the proof of the result.

□

Theorem 5.4 and Theorem 5.1 immediately give us the following result.

**No Swap-Regret Algorithm**

**Corollary 5.2.** Let $|\mathcal{A}| = n$. Then there exists an algorithm whose expected time averaged swap-regret is $\mathcal{O}\left(n\sqrt{\frac{\log n}{T}}\right)$.

# Chapter 6

# Price of Anarchy

Let $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$ be a game in normal form. Since players are assumed to be self-interested, the strategy profile that they collectively play in some equilibrium may not be good collectively. The notion of *social welfare* captures the goodness of strategy profiles. For example, in a network congestion game, a natural social welfare of a strategy profile would be the average delay of the players. In general a social welfare function $w : \prod_{i\in N} S_i \longrightarrow \mathbb{R}_{>0}$ maps strategy profiles to positive real numbers. The notion of *price of anarchy (PoA)* quantifies how bad a strategy profile in equilibrium be with respect to a best strategy profile (which maximizes social welfare).

**Price of Anarchy (PoA)**

**Definition 6.1.** The price of anarchy of a normal form game $\Gamma$ is defined as

$$\max_{s^{eq}\in \mathrm{PSNE}(\Gamma)} \frac{\max_{s^*\in\prod_{i\in N} S_i} w(s^*)}{w(s^{eq})}$$

## 6.1 Braess's Paradox

Consider the networks shown in Figure 6.1. Suppose $n$ players (assume $n$ is a sufficiently large number) wish to travel from $s$ to $t$ at the same time. The variable $x$ denotes the fraction of players using that edge and $c(x)$ is the time that the players using that edge would spend to traverse the edge. For the network on the left of Figure 6.1, in every PSNE, half of the traffic chooses each of the paths from $s$ to $t$ resulting in an average delay of $\frac{3}{2}$. However, for the network on the right of Figure 6.1, the path $s \to A \to B \to t$ is a weakly dominant strategy for every player. Hence every player choosing the path $s \to A \to B \to t$ is a weakly dominant strategy equilibrium for the corresponding game resulting players incurring an average delay of 2. Hence the price of anarchy of the game is $\frac{4}{3}$.[1]

[1] Strictly speaking, the above argument shows that the price of anarchy is at least $\frac{4}{3}$. Can you show that this bound is tight?

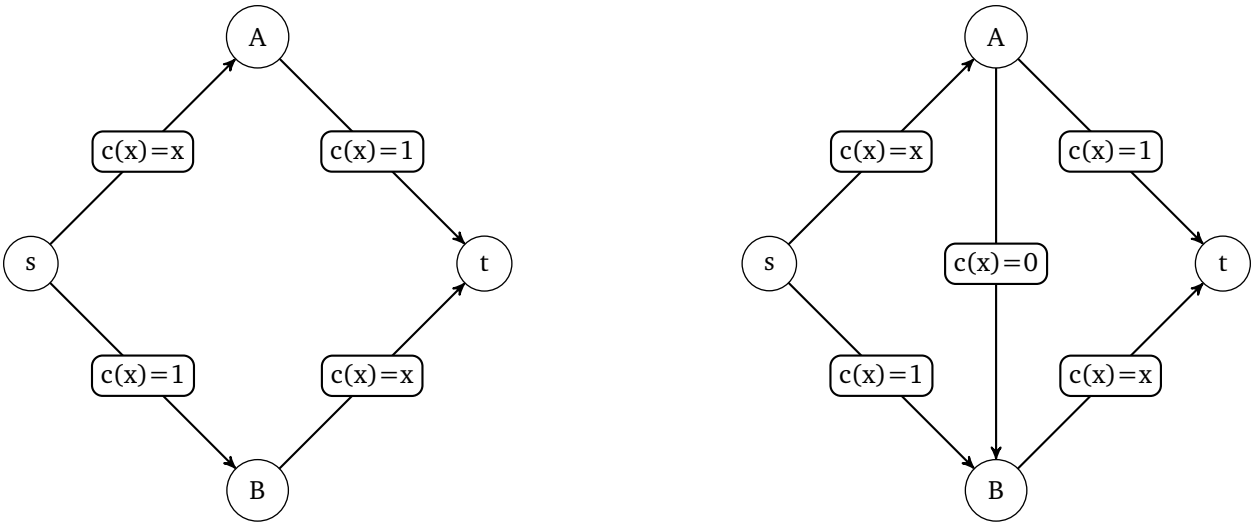



Figure 6.1: Braess's paradox.

## 6.2 Pigou's Network

We next see an even simpler network, known as Pigou's network shown in Figure 6.2, with the same PoA guarantee as Braess's network. Let us first consider the network on the left of Figure 6.2. The bottom edge is a weakly dominant strategy for every player. Hence, in the weakly dominant strategy equilibrium, every player takes the bottom edge resulting in an average travel time of 1 for every player. Whereas a strategy profile where half of the players take the top edge and other half take the bottom edge results in average travel time of $\frac{3}{4}$. Hence the PoA of the network is $\frac{4}{3}$.

For the network on the right of Figure 6.2, the cost function is non-linear. However, the bottom edge continues to be a weakly dominant strategy for every player. Whereas the socially optimal strategy would be to split total traffic as $\varepsilon$ (for some appropriate $\varepsilon$) fraction using the top edge and the rest $(1-\varepsilon)$ fraction using the bottom edge resulting in average delay $\varepsilon+(1-\varepsilon)^{p+1}$. Hence the average travel time in the optimal strategy goes to 0 as $p \to \infty$. Hence, as $p \to \infty$, the PoA also goes to $\infty$.

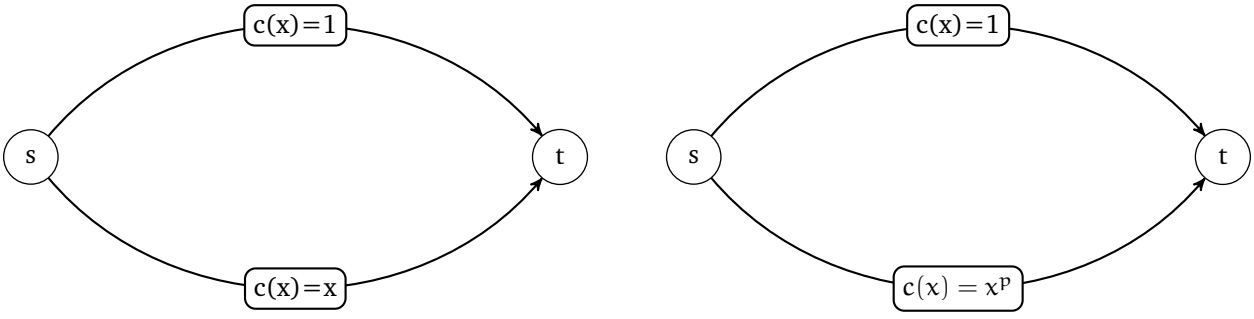



Figure 6.2: Pigou's network.

## 6.3 PoA for Selfish Networks

In this section, we now discuss PoA for arbitrary networks. We have already observed in Pigou's network that PoA can become worse if we allow "more non-linearity." Loosely speaking, the main result that we prove now is that among all networks, Pigou's network gives the worst PoA bound. Hence we may have bad PoA because of non-linear cost function and *not* because of some complicated network structure. In the rest of this section, we will make the above statement precise. We begin with formalizing Pigou's network.

**Pigou's Network**

**Definition 6.2.** A Pigou's network is a network of the following kind.

- ▷ It has exactly two vertices: one source $s$ and one sink $t$.
- ▷ It has exactly two edges: both the edges are from $s$ to $t$.
- ▷ Traffic rate from $s$ to $t$ is $r > 0$
- ▷ The cost of one edge is $c(x)$ where $x$ is the fraction of traffic using that edge. The function $c$ is non-decreasing, non-negative, and continuous.
- ▷ The cost of the other edge is $c(r)$.

Given a Pigou's network, its PoA is given by

$$\alpha(c) = \sup_{\varepsilon \in [0,r]} \left[ \frac{rc(r)}{\varepsilon c(\varepsilon) + (r-\varepsilon)c(r)} \right]$$

Since cost function $c$ is non-decreasing and non-negative, $\varepsilon c(\varepsilon) + (r-\varepsilon)c(r)$ is non-decreasing in $[r, \infty)$ and thus $\alpha(c)$ can also be written as

$$\alpha(c) = \sup_{\varepsilon \geqslant 0} \left[ \frac{rc(r)}{\varepsilon c(\varepsilon) + (r-\varepsilon)c(r)} \right]$$

Expressing $\alpha(c)$ where $\varepsilon$ ranges over all non-negative real numbers will be convenient later.

We have seen in non-linear Pigou's network that $\alpha(c)$ can get worse if we "increase" the non-linearity of the function $c$. Hence the idea is to restrict the function $c$ to vary within a class of functions and study how bad the PoA can become. Important examples of such classes include set of all affine functions $\{ax + b : a, b \in \mathbb{R}_{\geqslant 0}\}$, quadratic functions $\{ax^2 + bx + c : a, b, c \in \mathbb{R}_{\geqslant 0}\}$, etc. For a class $\mathcal{C}$ of cost function, we define $\alpha(\mathcal{C})$ as

$$\alpha(\mathcal{C}) = \sup_{c \in \mathcal{C}} \sup_{r>0} \alpha(c) = \sup_{c \in \mathcal{C}} \sup_{r>0} \sup_{\varepsilon \geqslant 0} \left[ \frac{rc(r)}{\varepsilon c(\varepsilon) + (r-\varepsilon)c(r)} \right]$$

**PoA Bound for Selfish Routing**

**Theorem 6.1.** For any class $\mathcal{C}$ of cost functions, the PoA of any network with one source and one destination with cost functions from $\mathcal{C}$ is at most $\alpha(\mathcal{C})$ with respect to PSNEs.

*Proof.* Let $\mathcal{G}$ be the graph under consideration and $f : \mathcal{E}[\mathcal{G}] \longrightarrow \mathbb{R}_{\geqslant 0}$ be an $s-t$ flow of $\mathcal{G}$. We observe that every strategy profile corresponds to an $s-t$ flow in $\mathcal{G}$. Thus, we will be working directly with flows instead of strategy profiles. Our first observation is that $f$ forms a PSNE if and only if every player is following a shortest path with respect to cost function $c_e(f)$ for the edge $e$. Formally, let $\mathcal{P}$ denote the set of all paths from $s$ to $t$ in $\mathcal{G}$ and $\overline{P}$ be any path from $s$ to $t$. Then we have the following if $f$ is a PSNE.

$$f_{\overline{P}} > 0 \Rightarrow \overline{P} \in \underset{P \in \mathcal{P}}{\operatorname{argmin}} \left\{ \sum_{e \in P} c_e(f_e) \right\}$$

Indeed otherwise, because of continuity of $c$, a suitably small fraction of players (assume that the number of players is sufficiently high) from the path $\overline{P}$ can be shifted to a shorter path which would be beneficial for these shifted players. Let $f^{eq}$ be a flow which constitutes a PSNE.

Let $f^*$ be a socially optimal flow. That is

$$f^* \in \underset{f \text{ is a } s-t \text{ flow of value } r}{\operatorname{argmin}} \left\{ \sum_{e \in \mathcal{E}} f_e c_e(f_e) \right\}$$

Since equilibrium flow $f^{eq}$ only uses shortest paths according to the cost function $c_e(f_e^{eq}), e \in \mathcal{E}[\mathcal{G}]$, all paths used by the equilibrium flow $f^{eq}$ have the same cost; let us call it $L$. Also, for any path $P \in \mathcal{P}$ from $s$ to $t$, its cost $c_P(f^{eq}) \geqslant L$. Hence, we have the following.

$$\sum_{P \in \mathcal{P}} \underbrace{f_P^{eq}}_{\text{sums to } r} \underbrace{c(f_P^{eq})}_{=L \text{ if } f_P^{eq} > 0} = rL \tag{6.1}$$

$$\sum_{P \in \mathcal{P}} \underbrace{f_P^{*}}_{\text{sums to } r} \underbrace{c(f_P^{eq})}_{\geqslant L} \geqslant rL \tag{6.2}$$

From Equations (6.1) and (6.2), we have the following.

$$\sum_{P \in \mathcal{P}} \left(f_P^{eq} - f_P^{*}\right) c(f_P^{eq}) \leqslant 0$$

Since $\alpha(\mathcal{C})$ is supremum over all $c \in \mathcal{C}, r > 0, \varepsilon \geqslant 0$, we have the following for any edge $e \in \mathcal{E}[\mathcal{G}]$ (by putting $r = f_e^{eq}$ and $\varepsilon = f_e^*$).

$$\begin{aligned}
\alpha(\mathcal{C}) &\geqslant \frac{f_e^{eq} c_e(f_e^{eq})}{f_e^* c_e(f_e^*) + (f_e^{eq} - f_e^*) c_e(f_e^{eq})} \\
\Rightarrow f_e^* c_e(f_e^*) &\geqslant \frac{1}{\alpha(\mathcal{C})} f_e^{eq} c_e(f_e^{eq}) - (f_e^{eq} - f_e^*) c_e(f_e^{eq}) \\
\Rightarrow \sum_{e \in \mathcal{E}} f_e^* c_e(f_e^*) &\geqslant \frac{1}{\alpha(\mathcal{C})} \sum_{e \in \mathcal{E}} f_e^{eq} c_e(f_e^{eq}) - \sum_{e \in \mathcal{E}} (f_e^{eq} - f_e^*) c_e(f_e^{eq}) \\
\Rightarrow c(f^*) &\geqslant \frac{c(f^{eq})}{\alpha(\mathcal{C})} \\
\Rightarrow \text{PoA} &\leqslant \alpha(\mathcal{C})
\end{aligned}$$

□

Hence it follows from Theorem 6.1 that the PoA of any selfish routing instance with affine cost functions is at most $\frac{4}{3}$. We end our discussion of PoA with another important example, namely selfish load balancing.

## 6.4 Selfish Load Balancing

Suppose we have $n$ jobs $J_1, \ldots, J_n$ with respective weights $w_1, \ldots, w_n$. We have $m$ machines $M_1, \ldots, M_m$ with respective speed $s_1, \ldots, s_m$. We wish to schedule these $n$ jobs into $m$ machines in a non-preemptive fashion — each job has to be executed in exactly one machine. Executing the $i$-th job in the $j$-th machine takes $\frac{w_i}{s_j}$ unit of time. This is the classical problem of job scheduling. For an assignment $A : [n] \longrightarrow [m]$ of jobs to machines, the load $\ell_j$ of machine $j$ is the total time it operates to execute the jobs assigned to it: $\ell_j = \sum_{i \in [n]: A(i)=j} \frac{w_i}{s_j}$. The *makespan* of a schedule is the maximum load of any machine, i.e. $\max_{j \in [m]} \ell_j$. One of the well studied optimization problem is to compute a schedule which has minimum possible makespan. We now study this problem with a game theoretic twist to it — each job is a player who itself chooses the machine for it. The social welfare of a schedule is its makespan.

## 6.5 Selfish Load Balancing Game

The load balancing game can be written in strategic form as follows:

(i) Players: $[n]$. That is the set of jobs.

(ii) Strategy set of any player: $[m]$. That is the set of machines.

(iii) Strategy profile: An assignment $A : [n] \longrightarrow [m]$

(iv) Cost function: suppose $i$-th job is run on machine $j_i \in [m]$ in an assignment. Then the cost of $i$-th job $c_i(j_1, \ldots, j_n)$ is the makespan $\ell_{j_i}$ of the machine $j_i$; that is the makespan of the machine where the $i$-th job runs.

We show that every load balancing game has a PSNE.

**PSNE of Load Balancing Game**

**Theorem 6.2.** Every load balancing game has a PSNE.

*Proof.* Let us associate each assignment $A : [n] \longrightarrow [m]$ with its *sorted load vector* $(\lambda_1, \ldots, \lambda_m)$ where $\lambda_j$ is the $j$-th highest load (breaking ties arbitrarily). Given two load vectors $\lambda = (\lambda_1, \ldots, \lambda_m)$ and $\lambda' = (\lambda'_1, \ldots, \lambda'_m)$, we say that $\lambda$ is lexicographically smaller than $\lambda'$ if there exists a $j \in [m]$ such that $\lambda_k = \lambda'_k$ for every $1 \leqslant k \leqslant j-1$ and $\lambda_j < \lambda'_j$.

Let $A$ be an assignment whose corresponding sorted load vector $(\lambda_1, \ldots, \lambda_m)$ is a lexicographically smallest vector.[2] We claim that $A$ is a PSNE. Suppose not, then there exists a job $i \in [n]$ who benefits from moving from its current machine $s$ to a new machine $t$. Suppose the load of machine $s$ and $t$ in the assignment $A$ are respectively $\lambda_a$ and $\lambda_b$ for some $a, b \in [m]$. Then the movement benefits the job $i$, we must have $\lambda_b < \lambda_a$. Let the new assignment be $A'$ having corresponding sorted load vector $(\lambda'_1, \ldots, \lambda'_m)$. Then we have $\lambda_k = \lambda'_k$ for every $1 \leqslant k \leqslant a-1$ and $\lambda_a > \lambda'_a$. This contradicts our assumption that $(\lambda_1, \ldots, \lambda_m)$ is a lexicographically smallest vector. □

[2] A smallest vector always exists since $<$ is a total order on a finite set.

## 6.6 Price of Anarchy of Selfish Load Balancing

Having shown the existence of a PSNE, we now prove that the PoA of a selfish load balancing game is small for identical machines.

**PoA of Selfish Load Balancing**

**Theorem 6.3.** The PoA of any selfish load balancing game with $n$ jobs and $m$ identical (having same speed) machines is at most $\left(2 - \frac{2}{m+1}\right)$ with respect to PSNEs.

*Proof.* Since machines are identical, by scaling the weights of the jobs appropriately, we may assume without loss of generality that the speed of every machine is 1. Let $A$ be a PSNE assignment, its makespan be $c(A)$, and OPT be the minimum makespan possible. Let $j^* \in [m]$ be a machine having the highest load under the assignment $A$ and $i^* \in [n]$ be a job assigned to $j^*$. We observe that there must be at least 2 jobs which are assigned to the machine $j^*$ otherwise we have $c(A) = \mathsf{OPT}$ (why?) and we are done. So we can assume without loss of generality that we have $w_{i^*} \leqslant \frac{1}{2}c(A)$. Also for any other machine $j \in [m] \setminus \{j^*\}$, its load $\ell_j$ under the assignment $A$ is at least $\ell_{j^*} - w_{i^*}$, otherwise the job $i^*$ will benefit by moving to machine $j$. Hence, we have the following.

$$\ell_j \geqslant \ell_{j^*} - w_{i^*} \geqslant c(A) - \frac{1}{2}c(A) \geqslant \frac{1}{2}c(A)$$

We now bound OPT which will prove the result.

$$\begin{aligned}
\mathsf{OPT} &\geqslant \frac{\sum_{i\in[n]} w_i}{m} \\
&= \frac{\sum_{j\in[m]} \ell_j}{m} \\
&= \frac{\ell_{j^*} + \sum_{j\in[m]\setminus\{j^*\}} \ell_j}{m} \\
&\geqslant \frac{c(A) + \frac{1}{2}(m-1)c(A)}{m} \\
&= \frac{m+1}{2m}c(A)
\end{aligned}$$

Hence the PoA is bounded as

$$\mathrm{PoA} = \frac{c(A)}{\mathsf{OPT}} \leqslant \frac{2m}{m+1} = \left(2 - \frac{2}{m+1}\right)$$

□

# Chapter 7

# Other Forms of Games

Till now we have studied *simultaneous move* games in the *complete information setting*. In this chapter, we would relax both the conditions. Relaxing the notion of simultaneous move gives rise to, what is called, *extensive form* games. Relaxing the assumption of complete information gives birth to Bayesian games.

## 7.1 Bayesian Games

**Bayesian Game**

**Definition 7.1.** A Bayesian game $\Gamma$ is given by $\langle N, (\Theta_i)_{i\in N}, (S_i)_{i\in N}, p, (u_i)_{i\in N}\rangle$ where

- $\triangleright$ $N = [n]$ is the set of players
- $\triangleright$ $\Theta_i$ is the set of types of player $i \in N$
- $\triangleright$ $S_i$ is the set of strategies of player $i \in N$
- $\triangleright$ $p$ is a probability distribution over $\Theta = \times_{i\in N}\Theta_i$
- $\triangleright$ $u_i : \times_{i\in N}\Theta_i \times_{i\in N} S_i \longrightarrow \mathbb{R}$

In a Bayesian game, the game $\Gamma$ is a common knowledge to the set of players. Player $i$ knows her realization $\theta_i \in \Theta_i$ of her type but does not know the realization of types of other players. However, since player $i$ knows the joint probability distribution $p$ over $\Theta$, she forms a posterior distribution $p(\cdot|\theta_i)$ on $\Theta_{-i}$. Let us see an example.

**Example 7.1.1** (Sealed Bid Selling Auction as Bayesian Game)**.** *Suppose there is a seller who wishes to buy an item from one of $n$ potential buyers by following an auction.*

- $\triangleright$ *The set of players:* $N = \{1, 2, \ldots, n\}$ *(this set corresponds to the buyers of the item).*
- $\triangleright$ *The set of types:* $\Theta_i = [0, 1]$ *for every* $i \in [n]$*. This corresponds to the set of valuations of buyer* $i$*.*

▷ *The set of strategies:* $S_i = [0,1]$ *for every* $i \in [n]$.

▷ $p$ *is the product distribution:* $p = \times_{i\in[n]} p_i$ *where* $p_i$ *is the uniform distribution over* $[0,1]$.

▷ *Allocation function:* $a : \times_{i\in N} S_i \longrightarrow \mathbb{R}^N$ *defined as* $a\left((s_i)_{i\in N}\right) = (a_i)_{i\in N}$ *where* $a_i = 1$ *if* $i$*-th buyer wins the auction and buys the item.*

▷ *Payment function:* $q : \times_{i\in N} S_i \longrightarrow \mathbb{R}^N$ *defined as* $q\left((s_i)_{i\in N}\right) = (q_i)_{i\in N}$ *where* $q_i$ *is the payment made by buyer* $i \in [n]$.

▷ *Utility function:*

$$u_i(\theta_1, \ldots, \theta_n, s_1, \ldots, s_i, \ldots, s_n) = a_i(\theta_i - q_i)$$

## 7.2 Selten Game

Any Bayesian game can be converted to an equivalent normal form game (with complete information) which is called the Selten game corresponding to the given Bayesian game. The high-level idea of the Selten game is the following. In the Selten game, we have a player corresponding to each type $\theta_i \in \Theta_i$ in the Bayesian game for every $i \in N$ — think of these players as possible instantiations of player $i$ which gets realized depending on her type. The utility of the player corresponding to $\theta_i$ is the expected utility (with respect to her posterior distribution $p(\cdot|\theta_i)$) of player $i$ in the original Bayesian game under type $\theta_i$. We now formally define the Selten game.

**Selten Game**

**Definition 7.2.** Let $\Gamma = \langle N, (\Theta_i)_{i\in N}, (S_i)_{i\in N}, p, (u_i)_{i\in N}\rangle$ be any Bayesian game. The corresponding normal form game $\Gamma^s$ is defined as $\Gamma^s = \langle N^s, (S_{\theta_i})_{\theta_i\in\Theta_i, i\in N}, (U_{\theta_i})_{\theta_i\in\Theta_i, i\in N}\rangle$ as

▷ $N^s = \cup_{i\in N}\Theta_i$

▷ $S_{\theta_i} = S_i$ for every $\theta_i \in \Theta_i, i \in N$

▷ $U_{\theta_i} : \times_{i\in N} \times_{\theta_i\in\Theta_i} S_i \longrightarrow \mathbb{R}$ defined as

$$\begin{aligned} U_{\theta_i}((s_{\theta_i})_{\theta_i\in\Theta_i, i\in N}) &= \mathbb{E}_{\theta_{-i}}\left[u_i\left((\theta_i, \theta_{-i}), (s_{\theta_i})_{i\in N}\right)|\theta_i\right] \\ &= \sum_{\theta_{-i}\in\Theta_{-i}} p(\theta_{-i}|\theta_i) u_i\left((\theta_i, \theta_{-i}), (s_{\theta_i})_{i\in N}\right) \end{aligned}$$

Observe that, viewing a Bayesian game as its corresponding Selten game, a pure strategy for player $i$ in the Bayesian game is nothing but the pure strategy profile $(s_{\theta_i})_{\theta_i\in\Theta_i} \in \times_{\theta_i\in\Theta_i} S_i$ of the corresponding players in the Selten game. Rephrasing, a pure strategy $s_i$ of a player $i$ in a Bayesian game is a function $s_i : \Theta_i \longrightarrow S_i$. This allows us to extend any game-theoretic solution concepts that we have learned for normal form games to Bayesian games. For example, Pure strategy Bayesian Nash equilibrium is defined as follows.

**Pure Strategy Bayesian Nash Equilibrium**

**Definition 7.3.** A pure strategy Bayesian Nash equilibrium of a Bayesian game $\Gamma = \langle N, (\Theta_i)_{i\in N}, (S_i)_{i\in N}, p, (u_i)_{i\in N}\rangle$ is a pure strategy profile $(s_1^*, \ldots, s_n^*)$ such that, for every player $i \in N$, pure strategy $s_i : \Theta_i \longrightarrow S_i$, and type $\theta_i \in \Theta_i$, we have the following

$$U_i((s_i^*, s_{-i}^*)|\theta_i) \geqslant U_i((s_i, s_{-i}^*)|\theta_i)$$

Equivalently, for every $i \in N$, $a_i \in S_i$, $\theta_i \in \Theta_i$, we have the following

$$\mathbb{E}_{\theta_{-i}}\left[u_i\left(\theta_i, \theta_{-i}, s(\theta_i), s_{-i}(\theta_{-i})\right)\right] \geqslant \mathbb{E}_{\theta_{-i}}\left[u_i\left(\theta_i, \theta_{-i}, a_i, s_{-i}(\theta_{-i})\right)\right]$$

We conclude our exposition on Bayesian games by exhibiting a pure strategy Bayesian Nash equilibrium for the first price sealed bid auction.

**Equilibrium of First Price Auction**

**Theorem 7.1.** In the Bayesian game corresponding to the first price selling auction, each buyer bidding half their individual valuation forms a pure strategy Bayesian Nash equilibrium under the following assumptions.

(i) We have 2 buyers.

(ii) Each buyer's valuation $\theta_i$ is uniformly distributed over $[0, 1]$.

(iii) Each buyer is *risk neutral:* bid $b_i(\theta_i)$ of player $i \in [2]$ is of the form $\alpha_i\theta_i$ for some $\alpha_i \in (0, 1]$.

*Proof.* The utility of player 1 is given by

$$\begin{aligned}
u_1(\theta_1, \theta_2, b_1, b_2) &= (\theta_1 - b_1(\theta_1))\Pr[b_2(\theta_2) \leqslant b_1(\theta_1)] \\
&= (\theta_1 - b_1(\theta_1))\Pr[\alpha_2\theta_2 \leqslant b_1(\theta_1)] \\
&= (\theta_1 - b_1(\theta_1))\Pr\left[\theta_2 \leqslant \frac{b_1(\theta_1)}{\alpha_2}\right] \\
&= \begin{cases} (\theta_1 - b_1(\theta_1))\frac{b_1(\theta_1)}{\alpha_2} & \text{if } \alpha_2 \geqslant b_1(\theta_1) \\ (\theta_1 - b_1(\theta_1)) & \text{if } \alpha_2 < b_1(\theta_1) \end{cases}
\end{aligned}$$

The following $b_1^*(\theta_1)$ maximizes $u_1(\theta_1, \theta_2, b_1, b_2)$.

$$b_1^*(\theta_1) = \begin{cases} \frac{\theta_1}{2} & \text{if } \frac{\theta_1}{2} \leqslant \alpha_2 \\ \alpha_2 & \text{if } \frac{\theta_1}{2} > \alpha_2 \end{cases}$$

Using above line of arguments for player 2, the following $b_2^*(\theta_2)$ maximizes $u_2(\theta_1, \theta_2, b_1, b_2)$.

$$b_2^*(\theta_2) = \begin{cases} \frac{\theta_2}{2} & \text{if } \frac{\theta_2}{2} \leqslant \alpha_1 \\ \alpha_1 & \text{if } \frac{\theta_2}{2} > \alpha_1 \end{cases}$$

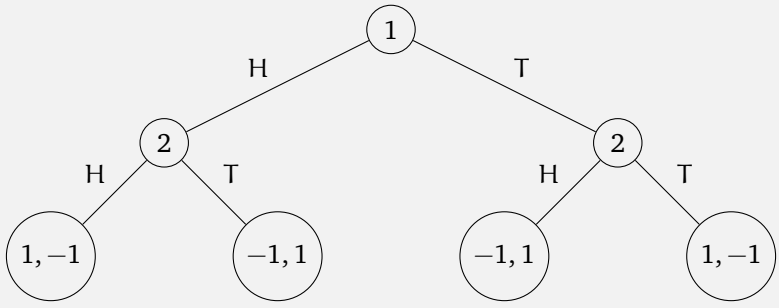
Solving the above equations for risk-neutral players, we get $\alpha_1 = \alpha_2 = \frac{1}{2}$, and thus $b_1^*(\theta_1) = \frac{\theta_1}{2}, b_2^*(\theta_2) = \frac{\theta_2}{2}$. Hence, with $\alpha_1 = \alpha_2 = \frac{1}{2}, (b_1^*(\theta_1) = \frac{\theta_1}{2}, b_2^*(\theta_2) = \frac{\theta_2}{2})$ forms a pure strategy Bayesian Nash equilibrium. □

The above proof can be generalized to $n$ risk-neutral players where every player's valuation is uniformly distributed over $[0, 1]$. It can be shown that $(\frac{n-1}{n}\theta_i)_{i\in[n]}$ forms a pure strategy Bayesian Nash equilibrium.

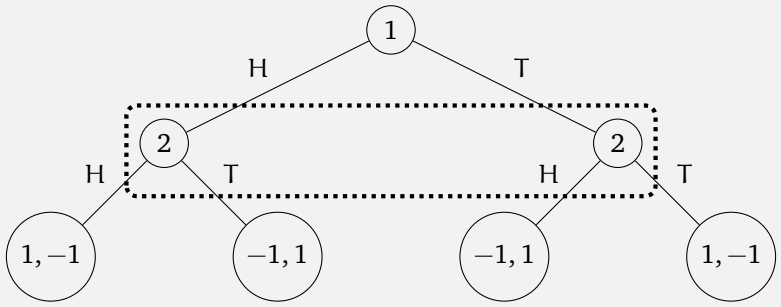
## 7.3 Extensive Form Games

We now relax the condition that players move simultaneously in a normal form game. We standard way to model a sequential move game is using *game tree*. Let us see few examples of sequential move games.

**Example 7.3.1** (Matching Pennies with Observation)**.** *We have seen the matching pennies game in normal form where players move simultaneously. Let us now consider the following new game: player* 1 *first makes a move, player* 2 *observes the action played by player* 1 *(and thus the name with observation), and player* 2 *makes a move, and then both players receive utilities. Each game tree has three types of nodes: (i) a root node corresponding to initial decision node, (ii) internal nodes corresponding to decisions of players, and (iii) leaf nodes corresponding to outcomes.*




**Example 7.3.2** (Matching Pennies without Observation)**.** *We have seen the matching pennies game in normal form where players move simultaneously. Let us now consider the following new game: player* 1 *first makes a move, player* 2 *makes a move before observing player* 1*'s action (and thus the name without observation), and then both players receive utilities. Observe that, when player* 2 *plays her action, then she does not know which of the two nodes within the dotted box she is in. The dotted box is called an* information set.




We now formally define extensive form games. For that, we need to define the concept of an information set.

**Information Set**

**Definition 7.4.** An information set of a player is a subset of that player's decision nodes which are indistinguishable to her.

For example, in the matching pennies game with observation, player 1 has one information set $\{\varepsilon\}$ consisting of empty history (corresponding to node 1) whereas player 2 has two information sets $\{H\}$ and $\{T\}$ consisting of two sub-histories which are distinguishable for player 2. On the other hand, in the matching pennies game without observation, player 1 still has one information set $\{\varepsilon\}$ consisting of empty history and player 2 also has one information set, namely $\{H, T\}$ consisting of two sub-histories which are indistinguishable for player 2. We now define extensive form games.

**Extensive Form Game**

**Definition 7.5.** An extensive form game $\Gamma$ is a tuple $\langle N, (S_i)_{i\in N}, \mathbb{H}, P, (\mathbb{I}_i)_{i\in N}, C, (u_i)_{i\in N}\rangle$ where

- $\triangleright$ $N = [n]$ : set of players.
- $\triangleright$ $S_i$ : set of strategies for player $i \in N$.
- $\triangleright$ $\mathbb{H}$ : set of all paths to leaf nodes called histories. $S_{\mathbb{H}}$ is the set of all proper sub-histories which are paths to internal nodes.
- $\triangleright$ $P : S_{\mathbb{H}} \longrightarrow N$ maps each sub-history (an internal node) to a player.
- $\triangleright$ $\mathbb{I}_i$ : information set for player $i \in N$.
- $\triangleright$ $C : \cup_{i\in N}\mathbb{I}_i \longrightarrow \cup_{i\in N} S_i$ with $C(J) \subseteq S_i$ for every $J \in \mathbb{I}_i, i \in N$ maps each information set to the strategies available at that information set.
- $\triangleright$ $u_i : \mathbb{H} \longrightarrow \mathbb{R}$ maps each history (leaf node) to the utility of player $i \in N$.

**Perfect Information and Imperfect Information Games**

**Definition 7.6.** An extensive form game is called a perfect information game if every information set is singleton. Otherwise, the extensive form game is called an imperfect information game.

For example, the matching pennies game with observation is a perfect information game whereas the matching pennies game without observation is an imperfect information game. The strategy set of player $i \in N : \{s_i : \mathbb{I}_i \longrightarrow S_i | s_i(J) \in C(J) \; \forall \; J \in \mathbb{I}_i\}$. Like Bayesian games, any extensive form game can be expressed equivalently as a strategic form game.

**Extensive Form Game to Strategic Form Game**

**Definition 7.7.** Given an extensive form game $\Gamma$ is a tuple $\langle N, (S_i)_{i\in N}, \mathbb{H}, P, (\mathbb{I}_i)_{i\in N}, C, (u_i)_{i\in N}\rangle$, the corresponding strategic form game $\Gamma^s\langle N^s, (S'_i)_{i\in N^s}, (u'_i)_{i\in N^s}\rangle$ is given by

- $N^s = N$
- $S'_i = \{s_i : \mathbb{I}_i \longrightarrow S_i | s_i(J) \in C(J) \; \forall \; J \in \mathbb{I}_i\}$
- $u'_i(s_1, \ldots, s_n)$ is the utility that player $i$ receives in $\Gamma$ at the leaf node if player play according to $(s_1, \ldots, s_n)$.

# Part II

# Mechanism Design

# Chapter 8

# Mechanism, Social Choice Function and Its Implementation

In the first part of the course, we have analyzed various games and explored concepts to predict the behavior of the system which consists of rational and intelligent players. Now we explore the theory of mechanism design which concerns designing rules of the game to achieve a certain objective.

On a high level, mechanism design is the same as algorithm design but some part of the input is held with strategic agents. So, in a generic mechanism design setting, we have a set of $n$ players, each player $i \in [n]$ has a private data $\theta_i \in \Theta_i$, and our goal is to compute a function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$ where $\mathcal{X}$ is the set of outcomes. To complete the picture, we need to define the utility function of every player $i \in [n]$ (otherwise saying that players are rational and intelligent does not make sense). Each player $i \in [n]$ has a utility function $u_i : \mathcal{X} \times \Theta \longrightarrow \mathbb{R}$ where $\Theta = \times_{i=1}^{n}\Theta_i$. On top of that, we assume a prior distribution $p \in \Delta(\Theta)$. The function $f$ is called a *social choice function*. Whenever we talk about any social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$, it should be understood that the $n$ parts of the input is held by $n$ different players with utility functions $u_i : \mathcal{X} \longrightarrow \mathbb{R}, i \in [n]$.

Loosely speaking, the job of the mechanism designer is to come up with a mechanism (an algorithm) which ensures that the social choice function $f$, that is under consideration, is indeed computed by the mechanism on every input $\theta \in \Theta$. To achieve this goal, the mechanism designer introduces a game among the players by defining two things: (i) strategy set $S_i$ for player $i$ for every $i \in N$ and (ii) a function $g : \times_{i\in[n]}S_i \longrightarrow \mathcal{X}$ which decides the outcome based on strategies played by each player. We now formally define what is called an indirect mechanism.

**Indirect Mechanism**

**Definition 8.1.** An indirect mechanism is a tuple $\mathcal{M} = \langle N = [n], \mathcal{X}, (\Theta_i)_{i\in N}, (S_i)_{i\in N}, \mathbb{P} \in \Delta(\Theta), g : \times_{i\in N}S_i \longrightarrow \mathcal{X}, u_i : \mathcal{X} \times \Theta \longrightarrow \mathbb{R}\rangle$ where

- $\triangleright$ $N$ is the set of players.
- $\triangleright$ $\mathcal{X}$ is the set of outcomes.
- $\triangleright$ $\Theta_i$ is the type set of player $i \in N$.

- ▷ $S_i$ is the strategy set of player $i \in N$.
- ▷ $\mathbb{P}$ is common prior over type profiles.
- ▷ $g$ maps strategy profiles to outcomes.
- ▷ $u_i$ is the utility function of player $i$.

If $N, \mathcal{X}, (\Theta_i)_{i\in N}, \mathbb{P}, u_i$ is clear from the context, we will denote an indirect mechanism by $((S_i)_{i\in N}, g(\cdot))$.

A mechanism is called a *direct mechanism* if we have $S_i = \Theta_i$ for every player $i \in [n]$ and $g = f$. Hence, by definition, the set of direct mechanisms forms a strict subset of the set of all indirect mechanisms. Typically $N, \mathcal{X}, (\Theta_i)_{i\in N}$, and $u_i$ will be clear from the context and we will assume that we are always given a prior distribution $\mathbb{P} \in \Delta(\Theta)$. Therefore, to avoid cumbersome notation, we will denote a direct mechanism by $((\Theta_i)_{i\in N}, f(\cdot))$. Observe the simplicity of direct mechanisms: unlike indirect mechanism, the mechanism designer does not need to worry about the choice of $S_i, i \in [n]$ and $g$. Actually, given a social choice function $f$, there is nothing left to design in direct mechanism. The mechanism designer simply asks every player to reveal his/her true type and applies $f$ on revealed type profile to choose an outcome from $\mathcal{X}$.

**Example 8.0.1** (Buying Auction)**.** *An auction, say for a buyer to buy an item from $n$ potential sellers, is a mechanism.*

- ▷ *The set of players* $(N) : \{0, 1, \ldots, n\}$*. Say player* $0$ *is the buyer and the remaining players are the sellers.*
- ▷ *The set of outcomes*

$$\mathcal{X} = \left\{ (a_0, \ldots, a_n, p_0, \ldots, p_n) \in \mathbb{R}^{2n+2} : a_i \in \{0, \pm 1\}\ \forall\ i \in \{0, \ldots, n\}, \sum_{i=0}^{n} a_i = 0, \sum_{i=0}^{n} p_i = 0 \right\}$$

*Think of* $a_i = 1$ *if player* $i$ *receives the item,* $0$ *if player* $i$ *neither receives nor gives the item, and* $-1$ *if player* $i$ *gives the item. Similarly* $p_i$ *is the payment made by player* $i$*; if* $p_i$ *is negative, that means player* $i$ *receives payment. So the condition* $\sum_{i=0}^{n} p_i = 0$ *guarantees that the mechanism neither generates any surplus money nor requires any money from outside. This is called the* budget balance *property.*

- ▷ *The set* $\Theta_i$ *is the set of all possible valuations of the item to player* $i$*.*
- ▷ *The set of strategies:* $S_i, i \in [n]$ *is the set of possible bids for the item. Observe that the buyer also submits a bid. Think of it like the maximum amount of money the buyer is willing to pay for the item.*
- ▷ *The function* $g$ *decides the outcome. For different types of auctions,* $g$ *will be different.*
- ▷ $u_i((a_0, \ldots, a_n, p_0, \ldots, p_n), (\theta_0, \ldots, \theta_n)) = a_i\theta_i - p_i$

## 8.1 Bayesian Game Induced by a Mechanism

One can immediately see the similarity between mechanisms and Bayesian games. Indeed, every mechanism $\mathcal{M} = \langle N = [n], \mathcal{X}, (\Theta_i)_{i\in N}, (S_i)_{i\in N}, \mathbb{P} \in \Delta(\Theta), g : \times_{i\in N} S_i \longrightarrow \mathcal{X}, u_i : \mathcal{X} \times \Theta \longrightarrow \mathbb{R}\rangle$ induces a Bayesian game $\Gamma_{\mathcal{M}} = \langle N = [n], (\Theta_i)_{i\in N}, (S_i)_{i\in N}, \mathbb{P} \in \Delta(\Theta), U_i : S_1 \times \cdots \times S_n \times \Theta_1 \times \cdots \times \Theta_n \longrightarrow \mathbb{R}\rangle$ where

$$U_i(s_1, \ldots, s_n, \theta_1, \ldots, \theta_n) = u_i\left(g\left(s_1, \ldots, s_n\right), \theta_1, \ldots, \theta_n\right)$$

## 8.2 Implementation of Social Choice Function

We now study the following question:

Given a social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$, is it possible to compute $f$ in a strategic environment?

Towards answering the above question, we first need to formalize what do we mean by computing $f$ in a strategic environment. This notion is called the implementability of a social choice function.

**Implementation of Social Choice Function**

**Definition 8.2.** We say that a (indirect) mechanism $\mathcal{M} = ((S_i)_{i\in[n]}, g(\cdot))$ implements a social choice function $f : \Theta \longrightarrow \mathcal{X}$ if there exists an equilibrium $s^*(\cdot) = ((s_i^*(\cdot))_{i\in[n]})$ in the induced Bayesian game $\Gamma_{\mathcal{M}}$ such that $g(s_1^*(\theta_1), \ldots, s_n^*(\theta_n)) = f(\theta_1, \ldots, \theta_n)$.

If the equilibrium in Definition 8.2 happens to be a very weakly dominant strategy equilibrium, then we say that $\mathcal{M}$ implements $f$ in dominant strategy equilibrium. In this case, we say that the mechanism $\mathcal{M}$ is Dominant Strategy Incentive Compatible (DSIC in short). On the other hand, if the equilibrium is a pure strategy Bayesian Nash equilibrium, then we say that $\mathcal{M}$ implements $f$ in Bayesian Nash equilibrium. In this case, we say that $\mathcal{M}$ is Bayesian Incentive Compatible (BIC in short).

In the context of a buying auction (i.e. one buyer and multiple sellers), consider the following two social choice functions using notation the same as Example 8.0.1.

(i) $f_{fp} : \Theta_0 \times \cdots \times \Theta_n \longrightarrow \mathcal{X}$ where $f_{fp}(\theta_0, \ldots, \theta_n) = (a_0, \ldots, a_n, p_0, \ldots, p_n)$ is defined as follows.

$$\text{If } \theta_0 < \min_{i\in[n]} \theta_i : a_i = 0 \ \forall\ i \in \{0, 1, \ldots, n\}, p_i = 0 \ \forall\ i \in \{0, \ldots, n\}$$
$$\text{Otherwise} : a_0 = 1, a_j = -1, a_i = 0 \ \forall\ i \in [n] \setminus \{j\}$$
$$\text{where } a_k > a_j \ \forall\ k \in [j-1], a_k \geqslant a_j \ \forall\ k \in \{j+1, \ldots, n\}$$
$$p_0 = \theta_j, p_j = -\theta_j, p_i = 0 \ \forall\ i \in [n] \setminus \{j\}$$

(ii) $f_{sp} : \Theta_0 \times \cdots \times \Theta_n \longrightarrow \mathcal{X}$ where $f_{sp}(\theta_0, \ldots, \theta_n) = (a_0, \ldots, a_n, p_0, \ldots, p_n)$ is defined as follows.

$$\text{If } \theta_0 < \min_{i\in[n]} \theta_i : a_i = 0 \ \forall\ i \in \{0, 1, \ldots, n\}, p_i = 0 \ \forall\ i \in \{0, \ldots, n\}$$
$$\text{Otherwise} : a_0 = 1, a_j = -1, a_i = 0 \ \forall\ i \in [n] \setminus \{j\}$$
$$\text{where } a_k > a_j \ \forall\ k \in [j-1], a_k \geqslant a_j \ \forall\ k \in \{j+1, \ldots, n\}$$
$$p_0 = \min_{i\in[n]\setminus\{j\}} \theta_i, p_j = -p_0, p_i = 0 \ \forall\ i \in [n] \setminus \{j\}$$

We have seen that the second price auction mechanism implements $f_{sp}$ in dominant strategies. One can also observe that there is no direct mechanism that implements $f_{fp}$ even in a pure strategy Bayesian Nash equilibrium simply because revealing types truthfully is not even a Nash equilibrium of the induced Bayesian game. We will show next that there is actually no mechanism that implements $f_{fp}$.

## 8.3 Revelation Principle

Revelation principle, informally speaking, says that direct mechanisms are as powerful as indirect mechanisms as far as implementability of any social choice function is concerned.

**Revelation Principle for DSIC**

**Theorem 8.1.** Suppose there exists an indirect mechanism $\mathcal{M} = ((S_i)_{i\in[n]}, g(\cdot))$ that implements a social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$ in dominant strategy equilibrium. Then the direct mechanism $((\Theta_i)_{i\in[n]}, f(\cdot))$ also implements $f$ in dominant strategy equilibrium.

*Proof.* Since $\mathcal{M}$ implements $f$ in dominant strategy equilibrium, there exists a very weakly dominant strategy equilibrium $(s_i^*(\cdot))_{i\in[n]}$ of the Bayesian game $\Gamma_{\mathcal{M}}$ such that we have the following.

$$g((s_i^*(\theta_i))_{i\in[n]}) = f((\theta_i)_{i\in[n]}) \ \ \forall \ i \in [n], \theta_i \in \Theta_i \tag{8.1}$$

$$u_i(g((s_i^*(\theta_i))_{i\in[n]}), (\theta_i)_{i\in[n]}) \geqslant u_i(g((a_i, s_{-i}^*(\theta_{-i}))_{i\in[n]}), (\theta_i)_{i\in[n]}) \quad \forall \ i \in [n], \theta_i \in \Theta_i, a_i \in S_i \tag{8.2}$$

Using Equations (8.1) and (8.2), we have the following for every $i \in [n], \theta_i, \theta_i' \in \Theta_i, \theta_{-i} \in \Theta_{-i}$.

$$\begin{aligned}
u_i(f(\theta_i, \theta_{-i}), (\theta_i)_{i\in[n]}) &= u_i(g((s_i^*(\theta_i))_{i\in[n]}), (\theta_i)_{i\in[n]}) && \text{[Equation (8.1)]}\\
&\geqslant u_i(g((s_i^*(\theta_i'), s_{-i}^*(\theta_{-i}))_{i\in[n]}), (\theta_i)_{i\in[n]}) && [a_i = s_i^*(\theta_i') \text{ in Equation (8.2)}]\\
&= u_i(f(\theta_i', \theta_{-i}), (\theta_i)_{i\in[n]}) && \text{[Equation (8.1)]}
\end{aligned}$$

□

The above proof can be closely imitated to prove the revelation principle for BIC mechanisms too. The statement is as follows. We leave its proof to the reader.

**Revelation Principle for BIC**

**Theorem 8.2.** Suppose there exists an indirect mechanism $\mathcal{M} = ((S_i)_{i\in[n]}, g(\cdot))$ that implements a social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$ in a pure strategy Bayesian Nash equilibrium. Then the direct mechanism $((\Theta_i)_{i\in[n]}, f(\cdot))$ also implements $f$ in a pure strategy Bayesian Nash equilibrium.

Due to revelation principle, we can now easily check implementability of any social choice function $f$. The social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$ is implementable if and only if the direct mechanism $((\Theta_i)_{i\in[n]}, f(\cdot))$ is incentive compatible. The next question that we would like to investigate is the following.

Which social choice functions are implementable?

Obviously not all social choice functions are implementable. For example, the social choice function $f_{fp}$ is not implementable because the corresponding direct mechanism $((\Theta_i)_{i\in[n]}, f_{fp})$ is not incentive compatible. We will see that only a very limited class of social choice function is implementable if we allow the players to have any arbitrary utility functions. This result is due to Gibbard and Satterwaite which is one of the cornerstone results in game theory. We grab this occasion to introduce some useful properties of social choice functions.

## 8.4 Properties of Social Choice Function

### 8.4.1 Ex-Post Efficiency or Pareto Optimality

A social choice function $f$ is called *ex-post efficient or paretian* if, for every type profile, the outcome chosen by $f$ cannot be improved to any player without "hurting" somebody. Formally, it is defined as follows.

**Ex-Post Efficiency**

**Definition 8.3.** A social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$ is called ex-post efficient or paretian or simply efficient if, for every type profile $\theta \in \times_{i\in[n]}\Theta_i$, the outcome $f(\theta)$ chosen by $f$ is pareto optimal. That is, for every type profile $\theta \in \times_{i\in[n]}\Theta_i$, there does not exist any outcome $x \in \mathcal{X}$ such that

$$u_i(x,\theta) \geqslant u_i(f(\theta),\theta)\ \forall\, i \in [n] \text{ and } u_j(x,\theta) > u_j(f(\theta),\theta) \text{ for some } j \in [n]$$

### 8.4.2 Non-Dictatorship

A social choice function $f$ is called *dictatorship* if there exists a player, called *dictator*, such that the outcome that $f$ chooses for any type profile is best possible for the dictator. Formally, the notion of dictatorship is defined as follows.

**Non-Dictatorship**

**Definition 8.4.** A social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$ is called a dictatorship if there exists a player $d \in [n]$, called dictator, such that we have the following for every type profile $\theta \in \Theta$.

$$u_d(f(\theta),\theta) \geqslant u_d(x,\theta),\ \forall\, x \in \mathcal{X}$$

A social choice function is called non-dictatorship if it is not a dictatorship.

### 8.4.3 Individual Rationality

The property individual rationality deals with voluntary participation of players in a mechanism. That is, when a rational and self-interested player will participate in the mechanism $((\Theta_i)_{i\in[n]}, f(\cdot))$. To formalize this concept, suppose $\overline{u_i}(\theta_i)$ is the utility of player $i \in [n]$ if her type is $\theta_i \in \Theta_i$ even if she does not participate in the mechanism. The most well studied notion in this context is *ex-post individual rationality* or simply *individual rationality*.

**Ex-Post Individual Rationality**

**Definition 8.5.** A social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$ is said to be ex-post individually rational or simply individually rational if we have the following for every player $i \in [n]$

$$u_i(f(\theta_i, \theta_{-i}), \theta) \geqslant \underline{u_i}(\theta_i),\ \forall\ (\theta_i, \theta_{-i}) \in \Theta$$

We can see that ex-post individual rationality is relevant in applications where players are allowed to withdraw from the mechanism even after every player has revealed their type. Our next form of individual rationality, called *interim individual rationality*, is required in applications where players are allowed to withdraw from the mechanism even after she has learned her type but other players are yet to declare their types.

**Interim Individual Rationality**

**Definition 8.6.** A social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$ is said to be interim individually rational if we have the following for every player $i \in [n]$ and every $\theta_i \in \Theta_i$

$$\mathbb{E}\left[u_i\left(f\left(\theta_i, \theta_{-i}\right), \theta\right) \mid \theta_i\right] \geqslant \underline{u_i}(\theta_i)$$

The last type of individual rationality, called *ex-ante individual rationality* ensures that players do not withdraw before knowing their actual type.

**Ex-Ante Individual Rationality**

**Definition 8.7.** A social choice function $f : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{X}$ is said to be ex-ante individually rational if we have the following for every player $i \in [n]$

$$\mathbb{E}\left[u_i\left(f\left(\theta_i, \theta_{-i}\right), \theta\right)\right] \geqslant \mathbb{E}\left[\underline{u_i}(\theta_i)\right]$$

**Remark:** Observe that, whether a given social choice function $f$ is interim individually rational or ex-ante individually rational depends on the prior distribution $\mathbb{P}$. Actually, it does not make sense to talk about interim individual rationality or ex-ante individual rationality without fixing a prior $\mathbb{P}$ (whose existence we have always assumed without mentioning explicitly every time). On the other hand, whether a social choice function is ex-post individually rational does not depend on the prior distribution at all.

# Chapter 9

# Gibbard-Satterwaite Theorem

We now present and prove the celebrated Gibbard-Satterwaite (GS in short) Theorem. GS Theorem essentially says that, if players are allowed to have any arbitrary utility function, then, loosely speaking, only dictatorship social choice functions are dominant strategy incentive compatible. For the GS theorem, we assume that the utility of a player $i$ depends only on the outcome and her own type (that is, it does not depend on the types of other players). This only makes the GS Theorem stronger. Under this assumption, we observe that any given type $\theta_i \in \Theta_i$ of player $i \in [n]$ induces a preference relation $\mathcal{R}_i$ on $\mathcal{X}$ defined as follows.

$$x \mathcal{R}_i y \Leftrightarrow u_i(x, \theta_i) \geqslant u_i(y, \theta_i)$$

The relation $\mathcal{R}_i$ is called the *rational preference relation* of player $i$. A player is said to have a *strict* rational preference relation if $\mathcal{R}_i$ is a total order. In the context of GS Theorem, it is convenient to work directly with strict relational preference relations than type sets. We denote the set of all possible strict rational preference relations (which are complete orders over $\mathcal{X}$) by $\mathcal{L}(\mathcal{X})$. Hence a social choice function, in the context of GS Theorem, maps a preference profile in $\mathcal{L}(\mathcal{X})^n$ to $\mathcal{X}$. A social choice function is called *unanimous* if $f(P_1, \ldots, P_n) = x$ whenever every player has a strict rational preference relation and $x$ is most preferred in every $P_i, i \in [n]$. We now state the GS Theorem.

## 9.1 Statement and Proof of Gibbard-Satterwaite Theorem

**Gibbard-Satterwaite Theorem**

**Theorem 9.1.** Let $f : \mathcal{L}(\mathcal{X})^n \longrightarrow \mathcal{X}$ be a social choice function such that

(i) The outcome set $\mathcal{X}$ is finite and $|\mathcal{X}| \geqslant 3$

(ii) Every player has strict rational preference relation

(iii) The function $f$ is unanimous

Then $f$ is dominant strategy incentive compatible if and only if $f$ is a dictatorship.

There have been many proofs known for the GS Theorem and most of them are completely accessible

and uses elementary techniques only. Here we present one such simple proof due to Benoît [Ben00].

*Proof.* If $n = 1$, then, by unanimity, $f$ is dictatorship and thus we have nothing to prove. Hence, let us assume that $n \geqslant 2$. We first show that $f$ is *monotone* in the following sense.

**Claim 9.1.1.** *Let $(P_i, P_{-i})$ be a preference profile, $f(P_i, P_{-i}) = A$. Let $P'_i$ be another preference where the position of an outcome $B$ improves from $P_i$ and the order of every other outcomes remain unchanged from $P_i$. Then $f(P_i, P_{-i})$ is either $A$ or $B$.*

*Proof.* Suppose $f(P_i, P_{-i}) = C$ for some $C \in \mathcal{X} \setminus \{A, B\}$. If $C$ is preferred over $A$ in $P_i$, then the $i$-th player in the profile $(P_i, P_{-i})$ can manipulate by misreporting her preference to be $P'_i$. On the other hand, if $A$ is preferred over $C$ in $P_i$ and thus in $P'_i$, then the $i$-th player in the profile $(P'_i, P_{-i})$ can manipulate by misreporting her preference to be $P_i$. □

Fix any outcome say $B \in \mathcal{X}$. Consider an arbitrary profile $\mathcal{P}^0$ where $B$ appears at the last position in every preference. We place $B$ at the first position one preference at a time from the preference of player 1 keeping the order of every other outcomes same until $f$ outputs $B$. Suppose $r$ is the first player such that, when we place $B$ at the first position of the first $r$ players, $f$ chooses $B$. Such an $r$ always exists since, due to unanimity, $f$ chooses $B$ if $B$ appears at the first position of every player's preference. Consider the following two profiles $\mathcal{P}_B$ and $\mathcal{Q}_B$.

$\mathcal{P}_B$ :

| $1$ | $\cdots$ | $r-1$ | $r$ | $r+1$ | $\cdots$ | $n$ |
|---|---|---|---|---|---|---|
| $B$ | $\cdots$ | $B$ | | $\cdots$ | $\cdots$ | |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| | $\cdots$ | | $B$ | $B$ | $B$ | $B$ |

$\mathcal{Q}_B$ :

| $1$ | $\cdots$ | $r-1$ | $r$ | $r+1$ | $\cdots$ | $n$ |
|---|---|---|---|---|---|---|
| $B$ | $\cdots$ | $B$ | $B$ | $\cdots$ | $\cdots$ | |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| | $\cdots$ | | $\cdot$ | $B$ | $B$ | $B$ |

Hence, $f$ does not choose $B$ in $\mathcal{P}_B$ and chooses $B$ in $\mathcal{Q}_B$. We call $r$ the pivotal player for $B$ and will denote it by $r^B$ when there is scope for confusion. In what follows, we will show that the $r$-th player is a dictator. We now make the following three important observations.

**Observation 9.1.1.** *Let $\mathcal{P}'_B$ be any profile of the following form. Then $f$ does not choose $B$ in $\mathcal{P}'_B$.*

$\mathcal{P}'_B$ :

| $1$ | $\cdots$ | $r-1$ | $r$ | $r+1$ | $\cdots$ | $n$ |
|---|---|---|---|---|---|---|
| $\cdot$ | $\cdots$ | $\cdot$ | | $\cdots$ | $\cdots$ | |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| | $\cdots$ | | $B$ | $B$ | $B$ | $B$ |

*Proof.* For the sake of arriving to a contradiction, let us assume that $f(\mathcal{P}'_B) = B$. From the profile $\mathcal{P}_B$, replace the preference of player 1 to $n$ one at a time with their corresponding preference from $\mathcal{P}'_B$. Suppose there exists an $i \in [n]$ such that $f$ chooses the outcome $B$ at the $i$-th step. Let the profiles at $(i-1)$-th and $i$-th steps be $\mathcal{P}_{i-1}$ and $\mathcal{P}_i$ respectively. If $i < r$, then the $i$-th player in $\mathcal{P}_{i-1}$ can manipulate by misreporting her preference to be the one in $\mathcal{P}_i$. On the other hand, if $i \geqslant r$, then the $i$-th player in $\mathcal{P}_i$ can manipulate by misreporting her preference to be the one in $\mathcal{P}_{i-1}$. □

**Observation 9.1.2.** *Let $\mathcal{Q}'_B$ be any profile of the following form. Then $f$ chooses $B$ in $\mathcal{Q}'_B$.*

$$\mathcal{Q}'_B:\quad \begin{array}{|ccccccc|}\hline 1 & \cdots & r-1 & r & r+1 & \cdots & n\\ B & \cdots & B & B & & \cdots\ \cdots & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ & \cdots & & \cdot & \cdot & \cdot & \cdot\\ \hline\end{array}$$

*Proof.* For the sake of arriving to a contradiction, let us assume that $f(\mathcal{Q}'_B) \neq B$. From the profile $\mathcal{Q}'_B$, replace the preference of player 1 to $n$ one at a time with their corresponding preference from $\mathcal{Q}_B$. Suppose there exists an $i \in [n]$ such that $f$ chooses the outcome $B$ at the $i$-th step. Let the profiles at $(i-1)$-th and $i$-th steps be $\mathcal{Q}_{i-1}$ and $\mathcal{Q}_i$ respectively. If $i < r$, then the $i$-th player in $\mathcal{Q}_i$ can manipulate by misreporting her preference to be the one in $\mathcal{Q}_{i-1}$. On the other hand, if $i \geqslant r$, then the $i$-th player in $\mathcal{Q}_{i-1}$ can manipulate by misreporting her preference to be the one in $\mathcal{Q}_i$. □

We observe that Observations 9.1.1 and 9.1.2 imply that the pivotal player $r^B$ does not depend on the profile $\mathcal{P}^0$ we start with.

**Observation 9.1.3.** *Let $A \in \mathcal{X} \setminus \{B\}$ be any outcome other than $B$ and $\mathcal{R}_B^A$ a profile of the following form. Then $f$ chooses $A$ in $\mathcal{R}_B^A$.*

$$\mathcal{R}_B^A:\quad \begin{array}{|ccccccc|}\hline 1 & \cdots & r-1 & r & r+1 & \cdots & n\\ \cdot & \cdots & \cdot & A & & \cdots\ \cdots & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ B & \cdots & B & B & B & \cdots & B\\ \hline\end{array}$$

*Proof.* From $\mathcal{R}_B^A$ construct the following profile $\mathcal{R}_1$ by putting $A$ at the first position and keeling the relative order of every other outcomes same in every preference. By unanimity, $f$ chooses $A$ in $\mathcal{R}_1$.

$$\mathcal{R}_1:\quad \begin{array}{|ccccccc|}\hline 1 & \cdots & r-1 & r & r+1 & \cdots & n\\ A & \cdots & A & A & A & \cdots & A\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ B & \cdots & B & B & B & \cdots & B\\ \hline\end{array}$$

From $\mathcal{R}_1$, put $B$ at the first position in the first $r-1$ preferences keeping every other order same to obtain another profile $\mathcal{R}_2$. By Claim 9.1.1 and Observation 9.1.1, $f$ chooses $A$ in $\mathcal{R}_2$.

$$\mathcal{R}_2:\quad \begin{array}{|ccccccc|}\hline 1 & \cdots & r-1 & r & r+1 & \cdots & n\\ B & \cdots & B & A & A & \cdots & A\\ A & \cdots & A & \cdot & \cdot & \cdots & \cdot\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ \cdot & \cdots & \cdot & B & B & \cdots & B\\ \hline\end{array}$$

From $\mathcal{R}_2$, put $B$ at the second position in the $r$-th preference keeping every other order same to obtain another profile $\mathcal{R}_3$. By Claim 9.1.1, $f$ chooses either $B$ or $A$ in $\mathcal{R}_3$. However if $f$ chooses $B$ in $\mathcal{R}_3$, then the $r$-th player can manipulate in $\mathcal{R}_3$ by misreporting her preference according to $\mathcal{R}_2$. Hence $f$ chooses $A$ in $\mathcal{R}_3$.

$\mathcal{R}_3$ :

| $1$ | $\cdots$ | $r-1$ | $r$ | $r+1$ | $\cdots$ | $n$ |
|---|---|---|---|---|---|---|
| $B$ | $\cdots$ | $B$ | $A$ | $A$ | $\cdots$ | $A$ |
| $A$ | $\cdots$ | $A$ | $B$ | $\cdot$ | $\cdots$ | $\cdot$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\cdot$ | $\cdots$ | $\cdot$ | $\cdot$ | $B$ | $\cdots$ | $B$ |

Now suppose $f$ chooses some outcome $D \neq A$ in $\mathcal{R}^A_B$. From $\mathcal{R}^A_B$, put $B$ at the first position in the first $r-1$ preferences keeping every other order same to obtain another profile $\mathcal{R}^A_{B,1}$. By Claim 9.1.1 and Observation 9.1.1, $f$ chooses $D$ in $\mathcal{R}^A_{B,1}$.

$\mathcal{R}^A_{B,1}$ :

| $1$ | $\cdots$ | $r-1$ | $r$ | $r+1$ | $\cdots$ | $n$ |
|---|---|---|---|---|---|---|
| $B$ | $\cdots$ | $B$ | $A$ | $\cdot$ | $\cdots$ | $\cdot$ |
| $\cdot$ | $\cdots$ | $\cdot$ | $\cdot$ | $\cdot$ | $\cdots$ | $\cdot$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\cdot$ | $\cdots$ | $\cdot$ | $B$ | $B$ | $\cdots$ | $B$ |

From $\mathcal{R}^A_{B,1}$, put $B$ at the second position in the $r$-th preference keeping every other order same to obtain another profile $\mathcal{R}^A_{B,2}$. Due to Claim 9.1.1, $f$ chooses either $D$ or $B$ in $\mathcal{R}^A_{B,2}$. If $f$ chooses $D$ in $\mathcal{R}^A_{B,2}$, then the

$\mathcal{R}^A_{B,2}$ :

| $1$ | $\cdots$ | $r-1$ | $r$ | $r+1$ | $\cdots$ | $n$ |
|---|---|---|---|---|---|---|
| $B$ | $\cdots$ | $B$ | $A$ | $\cdot$ | $\cdots$ | $\cdot$ |
| $\cdot$ | $\cdots$ | $\cdot$ | $B$ | $\cdot$ | $\cdots$ | $\cdot$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\cdot$ | $\cdots$ | $\cdot$ | $\cdot$ | $B$ | $\cdots$ | $B$ |

$r$-th player can manipulate by putting $B$ at the first position since, by Observation 9.1.2, $f$ chooses $B$ in the resulting profile. On the other hand, if $f$ chooses $B$ in $\mathcal{R}^A_{B,2}$, then we construct another profile $\mathcal{R}^A_{B,3}$ by putting $A$ at the second position in every preference from $1$ to $r-1$ and at the first position in every preference from $r+1$ to $n$ keeping the order of every other outcome same (of course one preference at a time). The function $f$ continues to choose $B$ otherwise, there exists an $i \in [n] \setminus \{r\}$ such that, when we modify the preference of the $i$-th player, then $B$ does not win in the resulting profile. Let the profiles at $(i-1)$-th step and $i$-th step be $\mathcal{R}'_{i-1}$ and $\mathcal{R}'_i$. Then, if $i < r$, then player $i$ can manipulate in $\mathcal{R}'_i$ by misreporting her preference to be as in profile $\mathcal{R}'_{i-1}$. On the other hand, if $i > r$, then player $i$ can manipulate in $\mathcal{R}'_{i-1}$ by misreporting her preference to be as in profile $\mathcal{R}'_i$.

$\mathcal{R}^A_{B,3}$ :

| $1$ | $\cdots$ | $r-1$ | $r$ | $r+1$ | $\cdots$ | $n$ |
|---|---|---|---|---|---|---|
| $B$ | $\cdots$ | $B$ | $A$ | $A$ | $\cdots$ | $A$ |
| $A$ | $\cdots$ | $A$ | $B$ | $\cdot$ | $\cdots$ | $\cdot$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\cdot$ | $\cdots$ | $\cdot$ | $\cdot$ | $B$ | $\cdots$ | $B$ |

However the profile $\mathcal{R}^A_{B,3}$ is the same as $\mathcal{R}_3$ where we have already argued that $f$ chooses $A$. This is a contradiction. Hence $f$ chooses $A$ in $\mathcal{R}^A_B$. □

We now show that $r$ is the pivotal player for every outcome $B$. That is, for any $C \in \mathcal{X} \setminus \{B\}$,we have $r^C = r^B$.

**Observation 9.1.4.** *For any outcome $C \in \mathcal{X} \setminus \{B\}$, we have $r^C = r^B$.*

*Proof.* Let $A \in \mathcal{X} \setminus \{B, C\}$ be an outcome other than $B$ and $C$. We consider the following profile $\mathcal{R}_3$. By Observation 9.1.2, $f$ chooses $B$ in $\mathcal{R}_3$. Using Observation 9.1.3 with respect to $C$ instead of $B$, $f$ chooses

$$\mathcal{R}_3 : \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & \cdots & r^B - 1 & r^B & r^B + 1 & \cdots & n \\ B & \cdots & B & B & A & \cdots & A \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ C & \cdots & C & C & C & \cdots & C \\ \hline \end{array}$$

the first outcome of the player $r^C$ in $\mathcal{R}_3$. Hence we have $r^C \leqslant r^B$. Exchanging the role of $B$ and $C$ in the argument so far, we have $r^B \leqslant r^C$. Hence we have $r^B = r^C$. □

We now prove our final observation to wrap up the proof.

**Observation 9.1.5.** *The player $r$ is a dictator for the social choice function $f$.*

*Proof.* Consider an arbitrary profile $\mathcal{R}_1$ where player $r$ places an outcome $K \in \mathcal{X}$ at the first position. We construct $\mathcal{R}_2$ by placing $B \in \mathcal{X} \setminus \{K\}$ at the last position of every preference keeping the order of every other alternatives same in every preference as in $\mathcal{R}_1$. Then, by Observation 9.1.3, $f$ chooses $K$ in $\mathcal{R}_2$. Next we restore $B$ in every preference in $\mathcal{R}_2$ to its original position in $\mathcal{R}_1$ thereby constructing $\mathcal{R}_1$ back. Now, by Claim 9.1.1, $f$ chooses either $B$ or $K$ in $\mathcal{R}_1$. Repeating the above argument using another outcome $C \in \mathcal{X} \setminus \{K, B\}$, $f$ chooses either $C$ or $K$ in $\mathcal{R}_1$ and thus $f$ chooses $K$ in $\mathcal{R}_1$. □

□

## 9.2 Way-outs from GS Impossibility

People often use one or more of the following ways to circumvent the theoretical bottleneck that the GS Theorem implies.

(i) Among most popular and successful approaches is to restrict the kind of utility functions that players are allowed to have. The popular *quasi-linear environment* which we will see soon is one of such success stories. Another success story along this line is single-peaked domain where the median social choice function is dominant strategy incentive compatible. Actually, there is nothing special about median social choice function, any social choice function which peaks say some (fixed) $i$-th maximum is also dominant strategy incentive compatible.

(ii) We can weaken our requirement and be satisfied with weaker equilibrium concepts like Bayesian Nash equilibrium.

(iii) Can we have a social choice function where manipulation is computationally intractable, NP-hard say.

# Chapter 10

# Mechanism Design in Quasi-linear Environment

Among the most successful way-outs from the impossibility result by Gibbard and Satterhwaite, has been to work with quasi-linear environment. Suppose we have $n$ players. In this environment, every outcome $x$ is a tuple $(k, t_1, \ldots, t_n)$ where $k$ belongs to some set $\mathcal{K}$ of *allocations* and $t_i$ is the money received by player $i$ (from the system). We make the convention that, if $t_i < 0$, then player $i$ pays $-t_i$ (to the system). Hence, in the quasi-linear environment, the set $\mathcal{X}$ of alternatives is

$$\mathcal{X} = \left\{ (k, t_1, \ldots, t_n) : k \in \mathcal{K}, t_i \in \mathbb{R} \ \ \forall\, i \in [n], \sum_{i \in [n]} t_i \leqslant 0 \right\}$$

In the quasi-linear environment, the utility function of player $i \in [n]$ has the following form.

$$u_i(x, \theta_1, \ldots, \theta_n) = u_i(x, \theta_i) = u_i((k, t_1, \ldots, t_n), \theta_i) = v_i(k, \theta_i) + t_i$$

A social choice function $f(\theta_1, \ldots, \theta_n) = (k(\theta_1, \ldots, \theta_n), t_1(\theta_1, \ldots, \theta_n), \ldots, t_n(\theta_1, \ldots, \theta_n))$ can be seen as two functions.

(i) An allocation function $k : \times_{i \in [n]} \Theta_i \longrightarrow \mathcal{K}$ decides the allocation for every type profile.

(ii) Payment functions $t_i : \times_{i \in [n]} \Theta_i \longrightarrow \mathbb{R}$ decides the payment for player $i \in [n]$ for every type profile.

## 10.1 Allocative Efficiency (AE)

Allocative efficiency is a property of the allocation function. We say a social choice function $f(\cdot) = (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ is allocatively efficient if, for each $\theta \in \Theta (= \times_{i \in [n]} \Theta_i)$, we have the following

$$k(\theta) \in \operatorname*{argmax}_{k \in \mathcal{K}} \sum_{i=1}^{n} v_i(k, \theta_i)$$

alternatively

$$\sum_{i=1}^{n} v_i(k(\theta), \theta_i) = \max_{k \in \mathcal{K}} \sum_{i=1}^{n} v_i(k, \theta_i)$$

## 10.2 Budget Balance (BB)

Budget balance is a property of the payment functions $t_1(\cdot), \ldots, t_n(\cdot)$ in quasi-linear environment. We say a social choice function $f(\cdot) = (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ is *strongly budget balance* if, for each $\theta \in \Theta (= \times_{i\in[n]}\Theta_i)$, we have the following

$$\sum_{i\in[n]} t_i(\theta) = 0$$

and *weakly budget balance* if , for each $\theta \in \Theta (= \times_{i\in[n]}\Theta_i)$, we have the following

$$\sum_{i\in[n]} t_i(\theta) \leqslant 0$$

We will assume that the payment functions satisfy weakly budget balance.
Let us see what we achieved by restricting ourselves to quasi-linear environment.

**No Dictatorship Function in Quasi-Linear Environment**

**Observation 10.1.** If we have at least two players, then no social choice function in quasi-linear environment is dictatorship.

*Proof.* Let $n$ $(\geqslant 2)$ be the number of players. For the sake of finding a contradiction, let $f(\cdot) = (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ be a dictatorship social choice function with dictator $d$. Let $\varepsilon > 0$ be any positive real number. For any $\theta \in \Theta$, consider the outcome $x \in \mathcal{X}$ defined as

$$x = (k(\theta), (t_i(\theta) - \varepsilon)_{i\neq d}, t_d(\theta) + (n-1)\varepsilon)$$

We observe $u_d(x, \theta_d) > u_d(f(\theta), \theta_d)$ which contradicts our assumption that $f$ is a dictatorship social choice function with dictator $d$. □

Next we see that ex-post efficient social choice functions can be characterized by allocative efficiency and budget balance conditions together in quasi-linear environments.

**Characterization of Ex-Post Efficiency**

**Observation 10.2.** A social choice function is ex-post efficient in a quasi-linear environment if and only if it is allocatively efficient and strictly budget balanced.

*Proof.* For the if direction: let us assume that a social choice function $f(\cdot) = (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ is allocatively efficient and strictly budget balanced. Then, for every $\theta \in \Theta$, we have the following which proves that $f$ is ex-post efficient.

$$\begin{aligned}
\sum_{i\in[n]} u_i(f(\theta), \theta_i) &= \sum_{i\in[n]} v_i(k(\theta), \theta_i) + \sum_{i\in[n]} t_i(\theta) \\
&= \sum_{i\in[n]} v_i(k(\theta), \theta_i) && \text{[strict budget balance]} \\
&\geqslant \sum_{i\in[n]} v_i(k, \theta_i) + \sum_{i\in[n]} t_i,\ \forall\, (k, t_1, \ldots, t_n) \in \mathcal{X} && \text{[allocative efficiency and}
\end{aligned}$$

$$\sum_{i\in[n]} t_i = 0]$$

$$= \sum_{i\in[n]} u_i(x,\theta_i),\ \forall\, x \in \mathcal{X}$$

For the only if direction: let us assume that a social choice function $f(\cdot) = (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ is ex-post efficient. We first claim that $f$ is allocatively efficient. Suppose not, then there exists a type profile $\theta \in \Theta$ and an allocation $k \in \mathcal{K}$ such that

$$\sum_{i\in[n]} v_i(k(\theta),\theta_i) < \sum_{i\in[n]} v_i(k,\theta_i)$$

Then there exists an agent $j \in [n]$ such that $v_j(k(\theta),\theta_j) < v_j(k,\theta_j)$. Let $\varepsilon = \sum_{i\in[n]} v_i(k,\theta_i) - \sum_{i\in[n]} v_i(k(\theta),\theta_i)$. We observe that $\varepsilon > 0$. We consider the outcome $x$ defined as follows.

$$x = (k, (t_i = t_i(\theta) + v_i(k(\theta),\theta_i) - v_i(k,\theta_i))_{i\neq j}, t_j = t_j(\theta) + v_j(k(\theta),\theta_j) - v_j(k,\theta_j) + \varepsilon)$$

The following shows that $x \in \mathcal{X}$.

$$\begin{aligned}
\sum_{i\in[n]} t_i &= \sum_{i\in[n]} t_i(\theta) + \sum_{i\in[n]} v_i(k(\theta),\theta_i) - \sum_{i\in[n]} v_i(k,\theta_i) + \varepsilon \\
&= \sum_{i\in[n]} t_i(\theta) \\
&\leqslant 0 && [\text{Since } (k(\theta), t_1(\theta), \ldots, t_n(\theta)) \in \mathcal{X}]
\end{aligned}$$

We now have the following for every agent $i \in [n] \setminus \{j\}$ which contradicts our assumption that $f$ is ex-post efficient.

$$u_i(x,\theta_i) = v_i(k,\theta_i) + t_i = t_i(\theta) + v_i(k(\theta),\theta_i) = u_i(f(\theta),\theta_i)$$

On the other hand, for the agent $j$, we have

$$u_j(x,\theta_j) = v_j(k,\theta_i) + t_j = t_j(\theta) + v_j(k(\theta),\theta_j) + \varepsilon > u_j(f(\theta),\theta_j)$$

We now show that $f$ must be strictly budget balanced. Suppose not, then there exists a type profile $\theta \in \Theta$ such that $\sum_{i\in[n]} t_i(\theta) < 0$. We consider the outcome $x$ defined as follows.

$$x = \left(k(\theta), t_1 = t_1(\theta) - \sum_{i\in[n]} t_i(\theta), (t_i = t_i(\theta))_{i\neq 1}\right)$$

Then we have $\sum_{i\in[n]} t_i = 0$ and thus $x \in \mathcal{X}$. Also $u_1(x,\theta_1) > u_1(f(\theta),\theta_1)$ and $u_i(x,\theta_i) > u_i(f(\theta),\theta_i)$ for every agent $i \in [n] \setminus \{1\}$. However, this contradicts our assumption that $f$ is ex-post efficient. □

## 10.3 Groves Mechanism

We now see a large class of mechanisms, called VCG (Vickrey-Clarke-Groves) mechanisms, which are dominant strategy incentive compatible. The most general mechanisms in the VCG class of mechanisms are the Groves mechanisms.

**Groves Theorem**

**Theorem 10.1.** Let $f(\cdot) = (k^*(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ be a allocatively efficient social choice function. Then $f(\cdot)$ is dominant strategy incentive compatible if payment functions satisfy the following structure (known as Groves payment (or incentive) scheme):

$$t_i(\theta_i, \theta_{-i}) = \left[\sum_{j \neq i} v_j(k^*(\theta), \theta_j)\right] + h_i(\theta_{-i}), \quad \forall\, i \in [n]$$

where $h_i : \Theta_{-i} \longrightarrow \mathbb{R}$ is any function.

*Proof.* Suppose not, then there exist a player $i \in [n]$ and $\theta_i, \theta_i' \in \Theta, \theta_{-i} \in \Theta_{-i}$ such that

$$u_i(f(\theta_i', \theta_{-i}), \theta_i) > u_i(f(\theta_i, \theta_{-i}), \theta_i)$$

Then we have the following.

$$\begin{aligned} v_i(k^*(\theta_i', \theta_{-i}), \theta_i) + t_i(\theta_i', \theta_{-i}) &> v_i(k^*(\theta_i, \theta_{-i}), \theta_i) + t_i(\theta_i, \theta_{-i}) \\ \Rightarrow \sum_{j \in [n]} v_j(k^*(\theta_i', \theta_{-i}), \theta_j) &> \sum_{j \in [n]} v_j(k^*(\theta_i, \theta_{-i}), \theta_j) \end{aligned}$$

However this contradicts our assumption that $f$ is allocatively efficient. □

*Remark:* Observe that, for any type profile $\theta_{-i}$ of players other than $i$, the money received by player $i$ depends on her type $\theta_i$ only through the allocation function $k^*(\theta_i, \theta_{-i})$. The change in money received by player $i$ if she changes her type from $\theta_i$ to $\theta_i'$, given other players play $\theta_{-i}$, is

$$t_i(\theta_i, \theta_{-i}) - t_i(\theta_i', \theta_{-i}) = \sum_{j \in [n] \setminus \{i\}} [v_j(k^*(\theta_i, \theta_{-i}), \theta_j) - v_j(k^*(\theta_i', \theta_{-i}), \theta_j)]$$

In other words, the change of money that player $i$ receives is the *externality* she imposes on the system.

The direct revelation mechanism of a social choice function $f$ which is allocatively efficient and satisfies Groves payment scheme is called a *Groves mechanism*.

## 10.4 Clarke (Pivotal) Mechanism

Groves mechanism does not specify the functions $h_{-i}, i \in [n]$. A special case of Groves mechanisms are Clarke mechanisms. For every $i \in [n]$, let $k^*_{-i} : \Theta_{-i} \longrightarrow \mathcal{K}$ be an allocatively efficient rule which allocates in the absence of agent $i$. In the Clarke mechanism, the function $h_i$ is defined as

$$h_i(\theta_{-i}) = -\sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j)$$

Hence, the Clarke payment scheme is the following.

$$t_i(\theta) = \left[\sum_{j \neq i} v_j(k^*(\theta), \theta_j)\right] - \left[\sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j)\right]$$

Hence the money that player $i$ receives is given by the total value of all agents other than $i$ in an efficient allocation in the presence of $i$ minus their total value in an efficient allocation in the absence of $i$.

Clarke mechanism is also sometimes called pivotal mechanism since each allocated player plays a pivotal role in deciding the value that other players receive in an efficient allocation due to her presence in the system.

Another interpretation of Clarke payment scheme comes from the following observation.

$$t_i(\theta) = \left[\sum_{j\in[n]} v_j(k^*(\theta),\theta_j)\right] - \left[\sum_{j\neq i} v_j(k^*_{-i}(\theta_{-i}),\theta_j)\right] - v_i(k^*(\theta),\theta_i)$$

Since $k^*$ is allocatively efficient, the following expression

$$\gamma_i = \left[\sum_{j\in[n]} v_j(k^*(\theta),\theta_j)\right] - \left[\sum_{j\neq i} v_j(k^*_{-i}(\theta_{-i}),\theta_j)\right]$$

is always non-negative and thus player $i$ is given a discount of $\gamma_i$ from her valuation $v_i(k^*(\theta),\theta_i)$ while calculating the amount she need to pay. This quantity $\gamma_i$ is called the Vickrey discount for player $i$.

## 10.5 Examples of VCG Mechanisms

VCG mechanisms are often one of the first mechanisms to study whenever we have any mechanism problem at hand. This is so because, on the one hand, VCG mechanisms are simple and, on the other hand, they guarantee strong properties like DSIC. Of course VCG mechanisms also has certain drawbacks which we will explore in due course. But, before that, let us see some concrete example of VCG mechanism.

**Example 10.5.1** (Vickrey Auction for Selling Multiple Identical Items)**.** *VCG Mechanism (aka Vickrey Auction) for Selling Multiple Identical Items by One Seller to Multiple Buyers.*

- ▷ *Suppose we have one seller who has* 3 *identical items to sell and* 5 *buyers. The set of players* $(N) : \{1, 2, \ldots, 5\}$*. Players* $1, \ldots, 5$ *are the* 5 *potential buyers each wanting to buy one unit of the item. Suppose the valuation of one unit of the item for the buyers are respectively* $20, 15, 12, 10, 8$*.*
- ▷ *Suppose we use Clarke payment rule. Since VCG mechanisms are DSIC, we may assume without loss of generality that every buyer bids her valuation.*
- ▷ *Any allocatively efficient rule (*$k^*(\cdot)$ *and* $k^*_{-i}(\cdot), i \in [5]$ *in particular) will allocate the* 3 *items to the* 3 *players (one item each) having highest valuations. Hence, players* $1, 2,$ *and* 3 *receives one item each.*
- ▷ *Payment received by agent* 1 *is*

$$\left[\sum_{j\neq 1} v_j(k^*(\theta),\theta_j)\right] - \left[\sum_{j\neq 1} v_j(k^*_{-1}(\theta_{-1}),\theta_j)\right] = (15+12) - (15+12+10) = -10$$

*Similar calculation shows that payment received by agent* 2 *and* 3 *is also* $-10$ *(that is, they pay* 10*). And thus the Vickrey discounts to buyers* $1, 2,$ *and* 3 *are respectively* $10, 5,$ *and* $2$*.*

**Example 10.5.2** (Combinatorial Auction)**.** *VCG Mechanism for Selling Multiple Non-identical Items by One Seller to Multiple Buyers.*

▷ *Suppose we have one seller with two items, namely* A *and* B*, and* 3 *buyers. The set of players* $(N) : \{1, 2, , 3\}$*. Players* 1, 2*, and* 3 *are the* 3 *potential buyers each wanting to buy one unit of the item. The valuation of each player for different bundles of items are shown as below. The notation* $\star$ *means that the player is not interested in that bundle.*

| | $\{A\}$ | $\{B\}$ | $\{A, B\}$ |
|---|---|---|---|
| *Buyer* 1 | $\star$ | $\star$ | 12 |
| *Buyer* 2 | 5 | $\star$ | $\star$ |
| *Buyer* 3 | $\star$ | 4 | $\star$ |

▷ *Suppose we use Clarke payment rule. Since VCG mechanisms are DSIC, we may assume without loss of generality that every buyer reports her valuation function truthfully.*

▷ *Any allocatively efficient rule (*$k^*(\cdot)$ *and* $k^*_{-i}(\cdot), i \in [3]$ *in particular) will allocate the bundle* $\{A, B\}$ *to buyer* 1 *and nothing to buyers* 2 *and* 3*.*

▷ *Payment received by agent* 1 *is*

$$\left[\sum_{j\neq 1} v_j(k^*(\theta), \theta_j)\right] - \left[\sum_{j\neq 1} v_j(k^*_{-1}(\theta_{-1}), \theta_j)\right] = (0) - (5+4) = -9$$

*Hence the Vickrey discount for the buyer* 1 *is* 3*.*

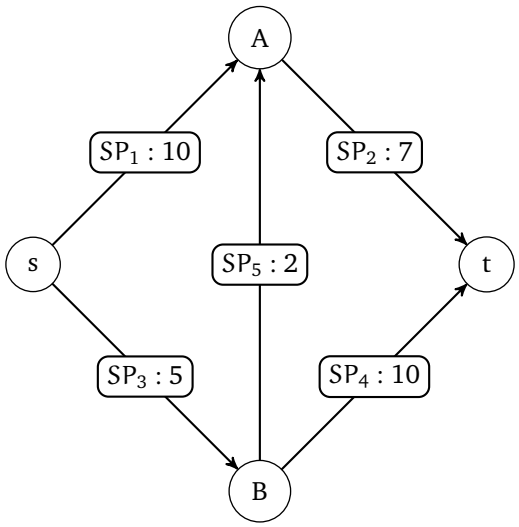



Figure 10.1: Strategic Network Formation

**Example 10.5.3.** *VCG Mechanism for Strategic Network Formation.*

▷ *Consider the network in Figure 10.1 whose each edge belongs to some service provider. Each service provider has a private type denoting the cost she incurs if one uses her edge. We wish to buy a path from* $s$ *to* $t$ *in this strategic environment. We call the edge held by* $SP_i$ *by* $i$ *for* $i \in [5]$.

▷ *We use VCG mechanism to learn the types of the players. The set* $\mathcal{K}$ *of allocations is*

$$\mathcal{K} = \{(1,1,0,0,0), (0,0,1,1,0), (0,1,1,0,1)\}$$

▷ *Any allocatively efficient allocation rule (*$k^*(\cdot)$ *and* $k^*_{-i}(\cdot), i \in [5]$ *in particular) will choose a shortest path from* $s$ *to* $t$. *Hence the allocation rule chooses the path* $s \to B \to A \to t$, *i.e.* $(0,1,1,0,1)$.

▷ *Payment received by agent* 2 *is*

$$\left[\sum_{j\neq 2} v_j(k^*(\theta), \theta_j)\right] - \left[\sum_{j\neq 2} v_j(k^*_{-2}(\theta_{-2}), \theta_j)\right] = (-5-2) - (-5-10) = 8$$

*Payment received by agent* 3 *is*

$$\left[\sum_{j\neq 3} v_j(k^*(\theta), \theta_j)\right] - \left[\sum_{j\neq 3} v_j(k^*_{-3}(\theta_{-3}), \theta_j)\right] = (-2-7) - (-10-7) = 8$$

*Payment received by agent* 5 *is*

$$\left[\sum_{j\neq 5} v_j(k^*(\theta), \theta_j)\right] - \left[\sum_{j\neq 5} v_j(k^*_{-5}(\theta_{-5}), \theta_j)\right] = (-5-7) - (-5-10) = 3$$

## 10.6 Weighted VCG

VCG mechanism shows that an allocatively efficient allocation rule is implementable in dominant strategy equilibrium using certain payment rules. One can ask which other allocation rules are implementable in dominant strategy equilibrium using some payment rule. It turns out a generalized class of "allocatively efficient" allocations functions, called affine maximizer, characterizes the class of allocation functions which are implementable in dominant strategy equilibrium. These generalizations allow giving weights to players, weights to allocations, and allows restricting the set of feasible allocations.

**Affine Maximizer**

**Definition 10.1.** An allocation rule $k : \times_{i\in[n]}\Theta_i \longrightarrow \mathcal{K}$ is called an affine maximizer if there exist $\mathcal{K}' \subseteq \mathcal{K}, w_1, \ldots, w_n \in \mathbb{R}$, and $c_{k'} \in \mathbb{R} \ \ \forall\, k' \in \mathcal{K}'$ such that, for every $\theta \in \Theta$, we have the following.

$$k(\theta) \in \operatorname*{argmax}_{k'\in\mathcal{K}'} \left[ c_{k'} + \sum_{i\in[n]} w_i v_i(k', \theta) \right]$$

The Groves payment scheme for affine maximizers can be generalized as follows. The proof of Theorem 10.2 follows closely the proof of Groves Theorem.

**Groves Payment for Affine Maximizer**

**Theorem 10.2.** Let $f(\cdot) = (k^*(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ be a social choice function such that $k^*(\cdot)$ is an affine maximizer. Then $f(\cdot)$ is dominant strategy incentive compatible if payment functions satisfy the following structure:

$$t_i(\theta_i, \theta_{-i}) = \sum_{j\neq i} \left[ \frac{w_j}{w_i} v_j(k^*(\theta), \theta_j) + \frac{c_{k^*(\theta)}}{w_i} \right] + h_i(\theta_{-i}), \quad \forall\, i \in [n]$$

where $h_i : \Theta_{-i} \longrightarrow \mathbb{R}$ is any function.

*Proof.* Suppose not, then there exist a player $i \in [n]$ and $\theta_i, \theta_i' \in \Theta, \theta_{-i} \in \Theta_{-i}$ such that

$$u_i(f(\theta_i', \theta_{-i}), \theta_{-i}) > u_i(f(\theta_i, \theta_{-i}), \theta_{-i})$$

Then we have the following.

$$\begin{aligned} & v_i(k^*(\theta_i', \theta_{-i}), \theta_{-i}) + t_i(\theta_i', \theta_{-i}) > v_i(k^*(\theta_i, \theta_{-i}), \theta_{-i}) + t_i(\theta_i, \theta_{-i}) \\ \Rightarrow\ & c_{k^*(\theta_i', \theta_{-i})} + \sum_{j\in[n]} w_j v_j(k^*(\theta_i', \theta_{-i}), \theta_j) > c_{k^*(\theta)} + \sum_{j\in[n]} w_j v_j(k^*(\theta_i, \theta_{-i}), \theta_j) \end{aligned}$$

However this contradicts our assumption that $k^*(\cdot)$ is allocatively efficient. □

Robert's Theorem states that, for unrestricted domain, affine maximizers are essentially the only allocation rules which are dominant strategy incentive compatible.

**Robert's Theorem**

**Theorem 10.3.** If $|\mathcal{K}| \geqslant 3$, the allocation rule $k^*(\cdot)$ is onto, the valuation functions are arbitrary, and the allocation rule $k^*(\cdot)$ is implementable in dominant strategy, then $k^*(\cdot)$ is an affine maximizer.

The proof Theorem 10.3 is nontrivial. We refer to [NRTV07, Chapter 12] for a proof of Theorem 10.3. Let us see another useful characterization DSIC mechanisms in quasi-linear environment which is (i) easy to prove and (ii) does not depend on the domain of $\Theta_i$.

**Characterization of DSIC Mechanisms**

**Proposition 10.1.** A social choice function $f(\cdot) = (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ is DSIC if and only if the following conditions hold for every $i \in [n]$ and $\theta_{-i} \in \Theta_{-i}$

(i) The payment $t_i(\theta)$ for player $i$ depends on $\theta_i$ only via $k(\theta_i, \theta_{-i})$. That is, for any two $\theta_i, \theta_i' \in \Theta_i$ and $\theta_{-i} \in \Theta_{-i}$ such that $k(\theta_i, \theta_{-i}) = k(\theta_i', \theta_{-i})$, we have $t_i(\theta_i, \theta_{-i}) = t_i(\theta_i', \theta_{-i})$. That is, $t_i$ is a function of $\mathcal{K}$ only.

(ii) The allocation function simultaneously optimizes for each player. That is, we have $k(\theta_i, \theta_{-i}) \in \operatorname{argmax}_{K \in k(\cdot, \theta_{-i})} (v_i(K, \theta_i) + t_i(\theta_i, \theta_{-i})) = \operatorname{argmax}_{K \in k(\cdot, \theta_{-i})} (v_i(K, \theta_i) + t_i(K))$ for every $i \in [n]$.

*Proof.* If part: Let $\theta_i, \theta_i' \in \Theta_i$ and $\theta_{-i} \in \Theta_{-i}$. Then we have the following.

$$\begin{aligned} u_i(\theta_i, \theta_{-i}) &= v_i(k(\theta_i, \theta_{-i})) + t_i(\theta_i, \theta_{-i}) \\ &= v_i(k(\theta_i, \theta_{-i})) + t_i(k(\theta_i, \theta_{-i})) && \text{[Property (i)]} \\ &\geqslant v_i(k(\theta_i', \theta_{-i})) + t_i(k(\theta_i', \theta_{-i})) && \text{[Property (ii)]} \end{aligned}$$

Only if part; property (i): Suppose for some $\theta_i, \theta_i' \in \Theta_i$ and $\theta_{-i} \in \Theta_{-i}$ such that $k(\theta_i, \theta_{-i}) = k(\theta_i', \theta_{-i})$, we have $t_i(\theta_i, \theta_{-i}) > t_i(\theta_i', \theta_{-i})$. Then, in the profile $(\theta_i', \theta_{-i})$, player $i$ is better off misreporting her type to be $\theta_i$ instead of $\theta_i'$ which contradicts DSIC assumption of $f$.

Only if part; property (ii): Suppose for some $\theta_i, \theta_i' \in \Theta_i$ and $\theta_{-i} \in \Theta_{-i}$, we have

$$v_i(k(\theta_i, \theta_{-i})) + t_i(k(\theta_i, \theta_{-i})) < v_i(k(\theta_i', \theta_{-i})) + t_i(k(\theta_i', \theta_{-i}))$$

Then, in the profile $(\theta_i, \theta_{-i})$, player $i$ is better off misreporting her type to be $\theta_i'$ instead of $\theta_i$ which contradicts DSIC assumption of $f$. □

We will next show that the assumption $|\mathcal{K}| \geqslant 2$ in Theorem 10.3 is an important one since $|\mathcal{K}| = 2$ falls into the category called "single parameter" domain. In the single parameter domain, we have allocation rules which are not an affine maximizer but dominant strategy incentive compatible.

## 10.7 Single Parameter Domain

The assumption in Theorem 10.3 that $\Theta$ is unrestricted basically means that the valuation function $v_i : \mathcal{K} \times \Theta_i \longrightarrow \mathbb{R}$ could be arbitrary; in other terms, the valuation space of every player could have full dimensionality. The other extreme of full dimensionality is one dimensionality – there is a single real parameter that determines the function $v_i$. By single parameter domain, we will assume the following which is not completely general but sufficient to capture most applications.

**Single-Parameter Domain**

**Definition 10.2.** A single parameter domain $\Theta_i$ is defined by subset $\mathcal{K}_i \subseteq \mathcal{K}$ of allocations, $\Theta_i$ is a real interval and $v_i(k, \theta_i) = \theta_i$ for every $k \in \mathcal{K}_i$ and $v_i(k, \theta_i) = 0$ for every $k \in \mathcal{K} \setminus \mathcal{K}_i$. The set $\mathcal{K}_i$ and $\Theta_i$ are common knowledge. Intuitively speaking, we can think of $\mathcal{K}_i$ as the set of allocations

where player $i$ "wins."

We will characterize the set of implementable social choice function in single parameter setting. Towards that, we next define the notion of monotonicity.

**Monotone Allocation Function in Single Parameter Domain**

**Definition 10.3.** An allocation function $k^* : \Theta \longrightarrow \mathcal{K}$ in single parameter domain is called monotone in $\theta_i$ if, for every $\theta_{-i} \in \Theta_{-i}$ and $\theta_i \leqslant \theta_i'$, we have $k^*(\theta_i, \theta_{-i}) \in \mathcal{K}_i$ implies that $k^*(\theta_i', \theta_{-i}) \in \mathcal{K}_i$. That is, intuitively speaking, if $\theta_i$ makes player $i$ win, then any type greater than $\theta_i$ also makes player $i$ win.

We will show that every monotone allocation rule in single parameter domain is implementable using suitable payment rule. To define the payment rule, we next define the notion of critical value function.

**Critical Value Function**

**Definition 10.4.** Given an allocation rule, the critical value function $c_i(\theta_{-i})$ of player $i$ for a type profile $\theta_{-i} \in \Theta_{-i}$ is defined as

$$c_i(\theta_{-i}) = \sup_{\theta_i \in \Theta_i : k^*(\theta_i, \theta_{-i}) \notin \mathcal{K}_i} \theta_i$$

For some $\theta_{-i} \in \Theta_{-i}$, if there does not exist any $\theta_i \in \Theta_i$ such that $k^*(\theta_i, \theta_{-i}) \notin \mathcal{K}_i$, then $c_i(\theta_{-i})$ is undefined. That is, critical value is the real number above which the player wins and below which the player loses.

We now state and prove the main result of mechanism design in single parameter domain.

**Characterization of DSIC Social Choice Functions in Single Parameter Domain**

**Theorem 10.4.** A social choice function $f(\cdot) = (k^*(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ in a single parameter domain is DSIC and losers do not pay anything if and only if the following conditions hold:

(i) The allocation function $k^*(\cdot)$ is monotone in every $\theta_i$.

(ii) Every winning player essentially pays her critical value. That is,

$$t_i(\theta_i, \theta_{-i}) = \begin{cases} -c_i(\theta_{-i}) & \text{if } k^*(\theta_i, \theta_{-i}) \in \mathcal{K}_i \\ 0 & \text{otherwise} \end{cases}$$

For some $\theta_{-i} \in \Theta_{-i}$, if $c_i(\theta_{-i})$ is undefined, then we define $c_i(\theta_{-i}) = \lambda_i$ where $\lambda_i$ is any real number.

*Proof.* If part: For any $i \in [n], \theta_i \in \Theta_i, \theta_{-i} \in \Theta_{-i}$, if player $i$ wins, then her utility is $\theta_i - c_i(\theta_{-i})$ and if she

loses, then her utility is 0. Hence she prefers winning if and only if $\theta_i \leqslant c_i(\theta_{-i})$ which is exactly what she achieves by reporting $\theta_i$ (truthfully).

Only if part, monotonicity condition: Since $k^*(\cdot)$ is not monotone for some player $i \in [n]$, there exist $\theta_i, \theta_i' \in \Theta_i, \theta_i < \theta_i', \theta_{-i} \in \Theta_{-i}$ such that $k^*(\theta_i, \theta_{-i}) \in \mathcal{K}_i$ but $k^*(\theta_i', \theta_{-i}) \notin \mathcal{K}_i$. Since $f$ is DSIC, player $i$ prefers winning in the profile $(\theta_i, \theta_{-i})$. Thus we have $\theta_i - c_i(\theta_{-i}) \geqslant 0$. Again since $f$ is DSIC, player $i$ prefers losing in the profile $(\theta_i', \theta_{-i})$. Thus we have $\theta_i' - c_i(\theta_{-i}) \geqslant 0$. However this contradicts our assumption that $\theta_i < \theta_i'$. Hence $k^*(\cdot)$ is monotone.

Only if part, payment rule: From Proposition 10.1, we know that, for every $i \in [n]$, $\theta_, \theta_i' \in \Theta_i, \theta_{-i} \in \Theta_{-i}$ such that $k^*(\theta_i, \theta_{-i}), k^*(\theta_i', \theta_{-i}) \in \mathcal{K}_i$, we have $t_i(\theta_i, \theta_{-i}) = t_i(\theta_i', \theta_{-i})$. Suppose the payment rule is not what is specified in the statement. Then there exist $\theta_i \in \Theta_i, \theta_{-i} \in \Theta_{-i}$ such that player $i$ wins in $(\theta_i, \theta_{-i})$ but $t_i(\theta_i, \theta_{-i}) \neq -c_i(\theta_{-i})$. Let $-t_i(\theta_i, \theta_{-i}) > c_i(\theta_{-i})$ and thus, due to monotonicity, there exists a type $\theta_i' \in \Theta_i$ such that $-t_i(\theta_i, \theta_{-i}) > \theta_i' > c_i(\theta_{-i})$ and player $i$ wins in $(\theta_i', \theta_{-i})$. Then, in the profile $(\theta_i', \theta_{-i})$, player $i$ wins and thus have utility $\theta_i' + t_i(\theta_i', \theta_{-i}) = \theta_i' + t_i(\theta_i, \theta_{-i}) < 0$. Hence, in the profile $(\theta_i', \theta_{-i})$, player $i$ is better off reporting some type $\theta_i^{\text{lose}} < c_i(\theta_{-i})$ and have utility 0 by losing. This contradicts the DSIC assumption of $f$. On the other hand, if we have $-t_i(\theta_i, \theta_{-i}) < c_i(\theta_{-i})$, then we have a type $\theta_i' \in \Theta_i$ such that $-t_i(\theta_i, \theta_{-i}) < \theta_i' < c_i(\theta_{-i})$ and player $i$ loses in $(\theta_i', \theta_{-i})$. Then, in the profile $(\theta_i', \theta_{-i})$, player $i$ is better off reporting $\theta_i$ and receive a utility $\theta_i' + t_i(\theta_i, \theta_{-i}) > 0$ by winning than reporting true type $\theta_i'$ and receive an utility of 0 by losing. □

We see that Theorem 10.4 leaves enough room for implementing allocation rules which are not affine maximizer. All we need is that the allocation rule is monotone in each component. For example, in single parameter domain, the allocation rule $k^*(\theta) = \text{argmax}_{k \in \mathcal{K}} \sum_{i=1}^n v_i(k, \theta_i)^\lambda$ is implementable for any constant $\lambda > 1$. Recall that, we have assumed a stylized special case of the single-parameter domain (refer to Definition 10.2) and shown in Theorem 10.4 that monotone allocation rules are exactly the rules which are implementable in dominant strategy in single-parameter domain. The celebrated Mayerson's Lemma shows that, in every single-parameter domain, monotone allocation rules are exactly the allocation rules which are implementable in dominant strategy. We now formally state the Mayerson's Lemma and refer to the lecture notes of Prof. Tim Roughgarden for its proof.

**Mayerson's Lemma**

**Lemma 10.1.** We have the following in any single-parameter domain.

(i) An allocation rule $k : \Theta \longrightarrow \mathcal{K}$ is implementable in dominant strategy equilibrium if and only if $k$ is monotone.

(ii) If $k$ is monotone, then there is a unique payment rule $(t_1, \ldots, t_n)$ where players bidding 0 pays 0 (these are called normalized payment rules) such that the mechanism $(k, t_1, \ldots, t_n)$ is DSIC.

(iii) The payment rule is given by the explicit formula for differentiable $k$ as

$$t_i(\theta_i, \theta_{-i}) = -\int_0^{\theta_i} z \frac{d}{dz} k_i(z, \theta_i) dz$$

where $k_i(\cdot)$ is the allocation for player $i$. For other type of allocation rule $k$, similar payment rules can be used to implement $k$.

We finally conclude this chapter by stating that the payment functions are essentially unique in VCG mechanisms. The proof is not very difficult and we refer interested readers to [NRTV07, Theorem 9.37] for its proof.

**Uniqueness of Payment Rule of VCG Mechanisms**

**Theorem 10.5.** Assume $\Theta_i$ is a connected in some Euclidean space for every $i \in [n]$. Let $(k, t_1, \ldots, t_n)$ be a a DSIC mechanism. Then a mechanism $(k, t'_1, \ldots, t'_n)$ is DSIC if and only if we have the following for every $i \in [n]$.

$$t'_i(\theta) = t_i(\theta) + h_i(\theta_i)$$

for some $h_i : \Theta_{-i} \longrightarrow \mathbb{R}$.

## 10.8 Implementability in Intermediate Domains

We have good a understanding of DSIC allocation rules for two extreme cases: (i) when each $\Theta_i$ is unrestricted (ii) when each $\Theta_i$ is single dimensional. For these two extremes we exactly know the set of DSIC allocation rules in Theorems 10.3 and 10.4. How about other domains? We now present a partial characterization for convex domains – each $\Theta_i$ is a convex set in some Euclidean space. For this, we introduce the notion of weak monotonicity.

**Weak Monotonicity**

**Definition 10.5.** An allocation rule $k^* : \Theta \longrightarrow \mathcal{K}$ is called weakly monotone if we have the following for every $i \in [n], \theta_i, \theta'_i \in \Theta_i, \theta_{-i} \in \Theta_{-i}, k^*(\theta_i, \theta_{-i}) = x \neq y = k^*(\theta'_i, \theta_{-i})$

$$v_i(x, \theta_i) - v_i(y, \theta_i) \geqslant v_i(x, \theta'_i) - v_i(y, \theta'_i)$$

**Partial Characterization of Implementable Allocation rules in Intermediate Domains**

**Theorem 10.6.** If a mechanism $(k^*, t_1, \ldots, t_n)$ is DSIC, then $k^*$ is weakly monotone. On the other hand, if $\Theta_i$ is a convex set for every $i \in [n]$, then, for every weakly monotone allocation rule $k^*$, there exists payment rules $t_1, \ldots, t_n$ such that the mechanism $(k^*, t_1, \ldots, t_n)$ is DSIC.

The first part of the statement above is easy to prove. Proving second part of the theorem is quite non-trivial and we refer interested reader to [BCL+06, ABHM10].

*Proof of first part.* Suppose the mechanism $(k^*, t_1, \ldots, t_n)$ is DSIC. Like in the proof of Theorem 10.4, we have the following: for every $i \in [n]$, $\theta_i, \theta'_i \in \Theta_i, \theta_{-i} \in \Theta_{-i}$ such that $k^*(\theta_i, \theta_{-i}), k^*(\theta'_i, \theta_{-i}) \in \mathcal{K}_i$, we have $t_i(\theta_i, \theta_{-i}) = t_i(\theta'_i, \theta_{-i})$. Since the mechanism is DSIC, in the type profile $(\theta_i, \theta_{-i})$, player $i$ does not benefit by reporting $\theta'_i$. Hence, we have the following.

$$\begin{aligned} & u_i(x, \theta_i) - u_i(y, \theta_i) \geqslant 0 \\ \Rightarrow\ & v_i(x, \theta_i) - v_i(y, \theta_i) \geqslant 0 \qquad [\text{since } t_i(\theta_i, \theta_{-i}) = t_i(\theta'_i, \theta_{-i})] \end{aligned}$$

Similarly, in the type profile $(\theta_i', \theta_{-i})$, player $i$ does not benefit by reporting $\theta_i$. Hence, we have the following.

$$\begin{aligned} & u_i(y, \theta_i') - u_i(x, \theta_i') \geqslant 0 \\ \Rightarrow\ & v_i(y, \theta_i') - v_i(x, \theta_i') \geqslant 0 \qquad\qquad [\text{since } t_i(\theta_i, \theta_{-i}) = t_i(\theta_i', \theta_{-i})] \end{aligned}$$

Hence, we have the following.

$$v_i(x, \theta_i) - v_i(y, \theta_i) \geqslant v_i(x, \theta_i') - v_i(y, \theta_i')$$

□

## 10.9 A Glimpse of Algorithmic Mechanism Design: Knapsack Allocation

In the classical Knapsack problem, we have a set $\mathcal{I}$ of $n$ items and a bag of size $W$. Each item $i \in [n]$, has a weight $w_i$ and a valuation $v_i$. The goal is to find a subset $\mathcal{S} \subseteq \mathcal{I}$ of items of maximum total value $\sum_{j \in \mathcal{S}} v_j$ subject to the constraint that $\sum_{j \in \mathcal{S}} w_j \leqslant W$. The quantity $\sum_{j \in \mathcal{S}} v_j$ is called societal surplus or simply surplus for $\mathcal{S}$. Let us consider the Knapsack problem in a strategic environment – We have $n$ players each having an item. The weight of the item of player $i$ is $w_i$ which is common knowledge. The size $W$ of the knapsack is also common knowledge. However the valuation $v_i$ of item $i$ is known to player $i$ only. For a motivating example, think of $n$ advertisers want to show their ads during a popular TV show. The total duration $W$ of commercial break of the TV show and the duration $w_i$ of each ad $i$ are publicly known. However, the valuation $v_i$ of the ad $i$ is known to the advertiser $i$ only. The value of the1 advertiser $i$ is $v_i$ if her ad is chosen and 0 otherwise. A natural goal in this application is to select a subset of ads with maximum total valuations subject to the constraint that the sum of the duration of the chosen ads is at most $W$. We wish to design a mechanism in this strategic setting.

We first observe that the mechanism design problem we have at hand belongs to single-parameter domain — the type of player $i$ can be completely revealed by knowing a single real parameter $v_i$. From Theorem 10.4, we know that only monotone allocation rules are implementable. Is the Knapsack allocation rule monotone? Yes! If an item $i \in [n]$ is chosen and we increase its valuation keeping everything else fixed, item $i$ will continue to be chosen. However, choosing a Knapsack allocation is NP-hard. Can we have an implementable allocation rule which approximates the Knapsack objective well? Designing such rule is a two step process.

(i) Assume DSIC for free!

(ii) Design an allocation rule which (i) can be computed in polynomial time, (ii) approximates our objective function (in this case, the total valuation of items selected) well, and (iii) the allocation rule is monotone. Observe that, due to Theorem 10.4, monotonicity of allocation rule justifies our assumption of DSIC.

So, at least for single-parameter domain, algorithmic mechanism design is almost like designing approximation algorithms with an extra requirement that our algorithm should be monotone. In case of Knapsack problem in the above strategic environment, we now design a $\frac{1}{2}$ factor approximation algorithm which is also monotone.

We now define a simple allocation rule $k^{\text{greedy}}$ based on a natural greedy algorithm for Knapsack. Without loss of generality, we assume that $w_i \leqslant W$ for every $i \in [n]$. By renaming we assume that

$$\frac{v_1}{w_1} \geqslant \frac{v_2}{w_2} \geqslant \cdots \geqslant \frac{v_n}{w_n}$$

Pick items till we have space. That is, we define $S$ as

$$i^* = \max\left\{ i \in [n] : \sum_{j=1}^{i} w_j \leqslant W \right\}, S = [i^*]$$

The allocation rule $k$ picks either $S$ or a most valuable item whichever gives more valuation.

**$\frac{1}{2}$ Factor Approximation for Knapsack**

**Proposition 10.2.** The allocation rule $k^{\text{greedy}}$ approximates maximum surplus within a factor of $\frac{1}{2}$.

*Proof.* Let OPT be the maximum surplus and $v_{\max}$ be the maximum valuation of any item. We obviously have

$$\begin{aligned} k^{\text{greedy}} &\geqslant \max\left\{ \sum_{j=1}^{i^*} v_j, v_{i^*+1} \right\} \\ &\geqslant \frac{1}{2} \sum_{j=1}^{i^*+1} v_j \\ &\geqslant \frac{1}{2}\text{OPT} \end{aligned}$$

□

Also $k^{\text{greedy}}$ is obviously monotone — increasing valuation of an already selected item makes that item remain selected. Can we have an even better approximate Knapsack allocation rule? For the classical Knapsack problem, we have a fully polynomial time approximation scheme (FPTAS) — for every $\varepsilon > 0$, there is an algorithm running in time $\mathcal{O}\left(\frac{n^3}{\varepsilon}\right)$ which approximates optimal solution within a factor of $(1+\varepsilon)$. We encourage the readers to verify that the natural allocation rule for the FPTAS is not monotone. However, the FPTAS can easily be modified so that the allocation rule is monotone and thus implementable with suitable payment rules. This is a typical characteristic of works in algorithmic mechanism design. Consider a NP-hard problem, verify whether the best known approximation algorithm induces a DSIC mechanism, and if not, then either tweak the existing approximation algorithm or design a new approximation algorithm whose corresponding mechanism is DSIC.

# Chapter 11

# Mechanism Design Without Money

Chapter 10 talks about mechanism design with money — we have an allocation rule and we wish to implement it in a strategic environment by incentivizing players suitably. We now discuss mechanism design without money. We begin with one of the most successful application of mechanism design without money which is stable matching.

## 11.1 Stable Matching

In the stable matching problem, we have two sets $\mathcal{A}$ and $\mathcal{B}$ of $n$ vertices each. Each vertex in $\mathcal{A}$ has a strict preference order (which is a complete order) over $\mathcal{B}$ and each vertex in $\mathcal{B}$ has a strict preference order over $\mathcal{A}$. We are interested in a matching $\mathcal{M}$ of $\mathcal{A}$ with $\mathcal{B}$ which is "stable" in the following sense. Intuitively speaking, a matching $\mathcal{M}$ of $\mathcal{A}$ and $\mathcal{B}$ is "not stable" is there exist two vertices $a \in \mathcal{A}$ and $b \in \mathcal{B}$ which are not matched together and both of them prefer to be matched together compared to their current states. Such a pair $(a, b)$ of vertices is called a "blocking pair" of the matching $\mathcal{M}$. In an unstable matching, a blocking pair can match themselves with one another thereby breaking the matching $\mathcal{M}$ justifying the instability of $\mathcal{M}$. The framework of stable matching has a lot of applications, for example, matching men with women in a marriage portal, doctors with hospitals, applicants with jobs. We now formally define the stable matching problem.

**Blocking Pair**

**Definition 11.1.** Suppose we have two sets $\mathcal{A}$ and $\mathcal{B}$ of $n$ vertices each. Every vertex $a \in \mathcal{A}$ has a preference order $\succ_a$ over $\mathcal{B}$ and every vertex $b \in \mathcal{B}$ has a preference order $\succ_b$ over $\mathcal{A}$. Given a matching $\mathcal{M}$ between $\mathcal{A}$ and $\mathcal{B}$, a pair $a \in \mathcal{A}$ and $b \in \mathcal{B}$ is called a blocking pair for the matching $\mathcal{M}$ if one of the following holds.

(i) Both $a$ and $b$ are not matched to anyone in $\mathcal{M}$.

(ii) The vertex $a$ is unmatched, $b$ is matched with some $\mathcal{M}(b) \in \mathcal{A}$ in $\mathcal{M}$, and $a \succ_b \mathcal{M}(b)$.

(iii) The vertex $b$ is unmatched, $a$ is matched with some $\mathcal{M}(a) \in \mathcal{B}$ in $\mathcal{M}$, and $b \succ_a \mathcal{M}(a)$.

(iv) Both $a$ and $b$ are matched to $\mathcal{M}(a)$ and $\mathcal{M}(b)$ respectively in $\mathcal{M}$, and we have $a \succ_b \mathcal{M}(b)$ and

$b \succ_a \mathcal{M}(a)$.

A matching is called stable if there is no blocking pair.

**Stable Matching**

**Definition 11.2.** Suppose we have two sets $\mathcal{A}$ and $\mathcal{B}$ of $n$ vertices each. Every vertex $a \in \mathcal{A}$ has a preference order $\succ_a$ over $\mathcal{B}$ and every vertex $b \in \mathcal{B}$ has a preference order $\succ_b$ over $\mathcal{A}$. A matching $\mathcal{M}$ is called stable if there is no blocking pair.

It follows from Definition 11.2 that every stable matching is a perfect matching. We now show the main result of stable matching theory – every instance of stable matching problem has a stable matching. Moreover, such a stable matching can be computed in polynomial time. This is the celebrated Theorem by Gale and Shapley.

**Gale-Shapley Theorem**

**Theorem 11.1.** Every stable matching instance has a stable matching. Moreover it can be computed in polynomial time.

*Proof.* Let us call the sets $\mathcal{A}$ and $\mathcal{B}$ as the sets of men and women respectively. We now describe an algorithm called "proposal algorithm by men" or "men-proposing deferred acceptance algorithm." Initially everyone is unmatched. We pick any arbitrary unmatched man and let him propose his most preferred woman whom he has not yet proposed (irrespective of whether she is currently matched or not). Whenever an unmatched woman receives a proposal she accepts the proposal and gets matched. When a matched woman receives a proposal, she accepts her proposal if she prefers the proposer more than her current matched partner; in this case, her previous matched partner becomes unmatched again. The algorithm terminates when all men (and thus all women) are matched.

Observe that, once a woman gets matched, she never becomes unmatched although she can change her partner. Also, every woman who has received at least on proposal is matched. Since there are $n$ men and each has a preference list over $n$ women, after at most $n^2$ iterations, every woman receives at least one proposal and thus all the women (and thus all the men) are matched. Hence, the algorithm terminates after at most $n^2$ iterations. What we show next is that the matching $\mathcal{M}$ computed by the algorithm is a stable matching. For that, we take an arbitrary pair $a \in \mathcal{A}$ and $b \in \mathcal{B}$ of man and woman who are not matched together by $\mathcal{M}$. We will show that $(a, b)$ does not form a blocking pair. If $a$ has never proposed to $b$ during the run of the algorithm, then $a$ prefers his matching partner in $\mathcal{M}$ than $b$ and thus $(a, b)$ cannot form a blocking pair. On the other hand, if $a$ has proposed to $b$ and still $b$ is not matched with $a$ in $\mathcal{M}$, then $b$ prefers her partner in $\mathcal{M}$ than $a$ and thus $(a, b)$ cannot form a blocking pair. □

We observe that the proposal algorithm described in Theorem 11.1 does not specify which unmatched man is picked to propose. Can different order of unmatched men result into different stable matching to be output by the algorithm? The answer is no! To prove this, we will actually prove something stronger. For a

man $a \in \mathcal{A}$, let us define $h(a)$ to be the most preferred woman (according to $\succ_a$) that $a$ can be matched in any stable matching (as there could be multiple stable matchings). The following result shows that, in the matching output by the Gale-Shapley algorithm, every man $a$ is matched with $h(a)$. This matching is called men-optimal stable matching.

**Men-Optimal Stable Matching**

**Theorem 11.2.** In any matching $\mathcal{M}$ computed by the proposal algorithm by men, every man $a \in \mathcal{A}$ is matched with $h(a)$ in $\mathcal{M}$.

*Proof.* Consider any fixed execution of the Gale-Shapley algorithm by men. Let us define $\mathcal{R}_i = \{(a, b) \in \mathcal{A} \times \mathcal{B} : b \text{ has rejected } a \text{ in the first } i \text{ iterations}\}$ and $\mathcal{R} = \cup_{i \in [n^2]} \mathcal{R}_i$. To prove the statement, it is enough to show that, for any pair $(a, b) \in \mathcal{R}$, there is no stable matching which matches $a$ with $b$. We show this claim by induction on the number of iterations $i$.

For $i = 0$, $\mathcal{R}_0$ is an empty set and thus the statement is vacuously true. Hence the induction starts. Let us assume the statement for $i$ and prove for $i + 1$. If no woman has rejected any matched man in the $(i + 1)$-th iteration, then we have $\mathcal{R}_{i+1} = \mathcal{R}_i$ and the claim follows from induction hypothesis. So let us assume that some woman $b \in \mathcal{B}$ received a proposal from a man $a \in \mathcal{A}$ in the $(i + 1)$-th iterations and $b$ has rejected her previous partner, say $a' \in \mathcal{A}$. That is, we have $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \{(a', b)\}$. Then we obviously have $a \succ_b a'$. By induction hypothesis, the man $a$ is not matched with any woman whom he prefers more than $b$ in any stable matching. For the sake of finding a contradiction, let us assume that there is a stable matching $\mathcal{M}'$ which matches the woman $b$ with $a'$. Since the man $a$ is not matched with any woman whom he prefers more than $b$ in $\mathcal{M}'$, the pair $(a, b)$ forms a blocking pair for $\mathcal{M}'$ which contradicts our assumption that $\mathcal{M}'$ is a stable matching. □

Along the line of men-optimal stable matching, one can define notion of men-pessimal, women-optimal, and women-pessimal stable matchings. It can be shown using similar argument as the proof of Theorem 11.2 that the stable matching computed by the Gale-Shapley proposal algorithm by men is actually a women-pessimal stable matching.

The notion of stable matching safe-guards against every pair of man and woman deviating from a matching – in a stable matching there is no pair of man and woman so that they deviate from the matching under consideration and both become happier by getting matched with one another. One can think that we could be more demanding that we wish to safe-guard against any coalition of men and women. In a stable matching, can it happen that there is a coalition of men and women who can deviate from the matching under consideration and get matched among themselves in a way that no one in the coalition is unhappier than before and at least one of them is happier than previous matching? The next result shows that it cannot happen – it is enough to safe-guard against a pair of man and woman. In the context of the stable matching problem, a matching is called a core solution if no coalition of men and women get matched among themselves in a way so that no one is the coalition is worse off and at least one of them is better off. More abstractly, a core allocation is defined as follows.

**Core Allocation**

**Definition 11.3.** For every subset $\mathcal{S} \subset [n]$ of players, let $\mathcal{I}_\mathcal{S}$ be an index set and $\mathcal{K}_\mathcal{S} = \{(k_i^j)_{i\in\mathcal{S}} : j \in \mathcal{I}_\mathcal{S}\}$ the set of allocations when only players in $\mathcal{S}$ are present; define $\mathcal{K}_i = \{k_i : \exists \mathcal{S} \subseteq [n], (k_i, (k_j)_{j\in\mathcal{S}\setminus\{i\}}) \in \mathcal{K}_\mathcal{S}\}$ for $i \in [n]$. Every player $i \in [n]$ has a preference $\succcurlyeq_i$ which is a partial order over $\mathcal{K}_i$. We say an allocation $(k_1, \ldots, k_n) \in \mathcal{K}_{[n]}$ is not a core allocation if there exists a subset $\mathcal{S} \subseteq [n]$ of players and an allocation $(k_i, (k_j)_{j\in\mathcal{S}\setminus\{i\}}) \in \mathcal{K}_\mathcal{S}$ such that

(i) $k_i' \succcurlyeq_i k_i \;\; \forall\, i \in \mathcal{S}$

(ii) $k_i' \succ_i k_i$ for some $i \in \mathcal{S}$

Otherwise the allocation $(k_1, \ldots, k_n)$ is called a core allocation.

In case of the stable matching problem, the set $\mathcal{K}_\mathcal{S}$ of allocations for any $\mathcal{S} \subseteq [n]$ is

$$\mathcal{K}_\mathcal{S} = \left\{\left((\mathcal{M}(a))_{a\in\mathcal{A}\cap\mathcal{S}}, (\mathcal{M}(b))_{b\in\mathcal{B}\cap\mathcal{S}}\right) : \mathcal{M} \text{ is a perfect matching in } (\mathcal{A}\cup\mathcal{B})\cap\mathcal{S}\right\}$$

Typically, the set $\mathcal{K}$ of allocations would be clear from the context and we often skip defining it formally. Observe that, in general, we may not have any core allocation in some situations. However, in case of stable matching problem, core allocations are precisely the stable matchings.

**Stable Matchings are Core Allocations**

**Theorem 11.3.** In every instance of the stable matching problem, a matching is a core if and only if it is a stable matching.

*Proof.* By definition, every core matching is a stable matching. For the other direction, let $\mathcal{M}$ be a stable matching. If $\mathcal{M}$ is not a core matching, let $\mathcal{S}$ be a subset of men and women of smallest cardinality who can opt out of the matching $\mathcal{M}$ and get matched among themselves which makes, w.l.o.g, a man $a \in \mathcal{S}$ happier by getting matched with a woman $b \in \mathcal{S}$; that is $b \succ_a \mathcal{M}(a)$ where $\mathcal{M}(a)$ is the partner of $a$ in $\mathcal{M}$. Observe that, $a \neq \mathcal{M}(b)$ and thus we have $a \succ_b \mathcal{M}(b)$. Then $(a, b)$ forms a blocking pair for $\mathcal{M}$ contradicting our assumption that $\mathcal{M}$ is a stable matching. □

The last question on stable matching which we discuss here is regarding strategy-proofness. Is the men-proposing deferred acceptance algorithm strategy-proof for the players? The following example answers this question negatively. Consider a setting with 3 men $m_1, m_2, m_3$ and 3 women $w_1, w_2, w_3$ and the preferences be such that the men-proposing deferred acceptance algorithm outputs the matching $\{\{m_1, w_2\}, \{m_2, w_3\}, \{m_3, w_1\}\}$ and only woman $w_1$ receives 2 proposals whereas the other 2 women receive 1 proposal each. Suppose the woman $w_1$ received proposals from $m_2$ and $m_3$ (in this order), and rejects $m_3$. One such concrete example is the following and run of the men-proposal deferred acceptance algorithm with following proposal order of men: $m_2, m_1, m_2, m_3$.

| | | | |
|---|---|---|---|
| Preference of $m_1$ : | $w_2 \succ w_1 \succ w_3$ | Preference of $w_1$ : | $m_1 \succ m_3 \succ m_2$ |
| Preference of $m_2$ : | $w_1 \succ w_3 \succ w_2$ | Preference of $w_2$ : | $m_3 \succ m_1 \succ m_2$ |
| Preference of $m_3$ : | $w_1 \succ w_2 \succ w_3$ | Preference of $w_3$ : | $m_1 \succ m_3 \succ m_2$ |

Now suppose, the woman $w_1$ rejects $m_3$ instead of $m_2$ when she receives a proposal from $m_2$ – this happens if the woman $w_1$ misreports her preference to be $m_1 \succ m_2 \succ m_3$. Then the run of the men-proposal deferred acceptance algorithm with proposal order $m_2, m_1, m_1, m_3, \ldots$ of men results in matching $\{\{m_1, w_1\}, \{m_2, w_3\}, \{m_3, w_2\}\}$ which is better for the woman $w_1$ compared to the matching $\{\{m_1, w_2\}, \{m_2, w_3\}, \{m_3, w_1\}\}$.

So the men-proposal deferred acceptance algorithm can be manipulated by women. Does there exist any algorithm for stable matching which is strategy-proof? The answer turns out to be negative [Rot08]. However, the following result shows that any man does not have any incentive to misreport in the men-proposal deferred acceptance algorithm. That is, the men-proposal deferred acceptance algorithm is strategy-proof for men.

**Strategy-Proofness of Men-Proposal Deferred Acceptance Algorithm for Men**

**Theorem 11.4.** The men-proposal deferred acceptance algorithm is strategy proof for men.

*Proof.* Suppose not, then there exists a man, say $m \in \mathcal{A}$ who can get better partner by misreporting his preference to be $\succ'_m$ instead of $\succ_m$. Let $\mathcal{M}$ and $\mathcal{M}'$ be the stable matchings output by the men-proposal deferred acceptance algorithm on true preference profile $(\succ_m, \succ_{-m})$ and the profile $(\succ'_m, \succ_{-m})$. Let $w \in \mathcal{B}$ be the partner of $m$ in $\mathcal{M}'$. By assumption, we have $w \succ_m \mathcal{M}(m)$. Let $m_2 \in \mathcal{A}$ be the partner of $w$ in $\mathcal{M}$. Since $(m, w)$ does not form a blocking pair for $\mathcal{M}$, we have $\mathcal{M}(w) = m_2 \succ_w m$. Let $\mathcal{M}'(m_2) = w_2$ and again we have $\mathcal{M}'(m_2) = w_2 \succ_{m_2} \mathcal{M}(m_2) = w$ otherwise $(m_2, w)$ forms a blocking pair for $\mathcal{M}'$. Let us define $\mathcal{A}_1 = \{a \in \mathcal{A} : \mathcal{M}'(a) \succ_a \mathcal{M}(a)\}$. Then the above argument shows that the partner of any partner of any man in $\mathcal{A}_1$ also belongs to $\mathcal{A}_1$. Let us define $\mathcal{B}_1 = \{\mathcal{M}'(a) : a \in \mathcal{A}_1\} = \{\mathcal{M}(a) : a \in \mathcal{A}_1\}$.

Obviously we have $m \in \mathcal{A}_1$ and thus $\mathcal{A}_1, \mathcal{B}_1 \neq \emptyset$. Consider the run of the men-proposal deferred acceptance algorithm on input $(\succ_m, \succ_{-m})$. Suppose $t$ be the last iteration where some man $m_1 \in \mathcal{A}_1$ makes a proposal to some woman $w_1$; by the choice of $t$, the woman $w_1$ must have accepted the proposal by the man $m_1$ in the $t$-th iteration. Since every man in $\mathcal{A}_1$ gets worse partner in $\mathcal{M}$ than $\mathcal{M}'$, every man $m' \in \mathcal{A}_1$ has made a proposal to the woman $\mathcal{M}'(m')$ within the first $(t-1)$ iterations and eventually gets rejected and thus every woman has received at least one proposal within the first $(t-1)$ iterations. By the choice of $m_1$, the woman $w_1$ rejects some man $m'' \in \mathcal{A} \setminus \mathcal{A}_1$. So we have $m_1 \succ_{w_1} m''$ and $m'' \succ_{w_1} \mathcal{M}'(w_1)$ since $\mathcal{M}'(w_1)$ also must have proposed $w_1$. On the other hand, since we have $m'' \notin \mathcal{A}_1$, it follows that $w_1 \succ_{m''} \mathcal{M}(m'') \succcurlyeq_{m''} \mathcal{M}'(m'')$. Then $(m'', w_1)$ forms a blocking pair for $\mathcal{M}'$ which is a contradiction. □

## 11.2 House Allocation

In the stable matching problem, both the entities, namely men and women, have preferences over other. However, in many applications of matching, for example, campus house allocation, school admission, etc. only one type of entity has a preference. These situations are modelled by house allocation problem. In the house allocation problem, we have $n$ agents each occupying a house (so we have $n$ houses in total). Each agent also has a preference (a complete order) over the set of $n$ houses. The question that we are interested in is whether we can reallocate the houses which makes the agents better off. We consider the *Top Trading Cycle Algorithm (TTCA)* which was credited to Gale in [SS74].

**Top Trading Cycle Algorithm (TTCA):** We construct a directed graph $\mathcal{G}$ on the vertex set $[n]$. We have an edge from $i$ to $j$ if agent $i$ prefers the house of agent $j$ most; we are allowed to have $i = j$ which results into self-loop. Since the out-degree of every vertex is exactly 1, the graph $\mathcal{G}$ contains a cycle (self-loops are considered to be cycle too). Also all cycles are vertex disjoint. We allocate houses along every cycles. That is, for a cycle $i_1 \to i_2 \to i_3 \to \cdots \to i_\ell \to i_{\ell+1}(= i_1)$, we allocate the house of agent $i_{j+1}$ to the agent $i_j$ for every $j \in [\ell]$. We now delete the agents $i_1, \ldots, i_\ell$ and repeat on the remaining agents.

We now show that TTCA is DSIC.

**TTCA is DSIC**

**Theorem 11.5.** The mechanism induced by TTCA is DSIC.

*Proof.* On an instance of the house allocation problem, consider an arbitrary run of TTCA. Suppose TTCA runs for $\ell$ iterations – let $N_i$ be the set of agents who are allocated houses in the $i$-th iteration of the algorithm for $i \in [\ell]$. We will show that truth-telling is best strategy for every agent in $\cup_{j=1}^{i} N_j$ for every $i \in [\ell]$. For that we use induction on $i$. For the base case, i.e. for $i = 1$, we need to show that no agent in $N_1$ can be better off by lying. However, this is obvious since every agent in $N_1$ is allocated their top choice. Inductively, let us assume the statement for $i$ and prove for $i+1$. We observe that, any agent in $N_{i+1}$ is never pointed to by any agent in $\cup_{j=1}^{i} N_j$ in the first $i$ iterations. Hence, even if any agent outside $\cup_{j=1}^{i} N_j$ misreport their preference, the allocations in the first $i$ iterations remain the same. Hence no agent in $N_{i+1}$ is allocated a house of any agent in $\cup_{j=1}^{i} N_j$. TTCA allocates every agent in $N_{i+1}$ her top choice outside the houses initially occupied by someone in $\cup_{j=1}^{i} N_j$. Hence there is no incentive for any agent in $N_{i+1}$ to misreport. Hence the mechanism induced by TTCA is DSIC. □

Showing DSIC is not enough in this case since the trivial mechanism which allocates every agent their initial house is also DSIC. So we need to show that the performance of TTCA is good in some sense which we show next.

**TTCA Allocation is The Unique Core Allocation**

**Theorem 11.6.** For every instance of the house allocation problem, the allocation computed by TTCA is the unique core allocation of the instance.

*Proof.* We first prove that TTCA allocation $k_T$ is a core allocation. Let $S \subseteq [n]$ be a set of agents and $k$ an allocation of houses of $S$ among themselves such that no agent in $S$ is worse off and at least one agent in $S$ is better off in $k$ compared to $k_T$. Define $\ell, N_i, i \in [\ell]$ as in the proof of Theorem 11.5. Since the agents in $N_1$ get their best house in $k_T$, each the agent in $S \cap N_1$ receives the same house in both $k_T$ and $k$. Given this, it follows that each agent in $S \cap (N_1 \cup N_2)$ receives the same house in both $k_T$ and $k$. Inductively it follows that each agent in $S \cap (N_1 \cup \cdots \cup N_\ell)$ receives the same house in both $k_T$ and $k$. This contradicts our assumption that at least one agent in $S$ receives a better house in $k$ than in $k_T$.

We now prove uniqueness of core allocation. Any core allocation $k$ must allocate the top choice for

every agent in $N_1$. Otherwise, the agents in $N_1$ can form a coalition and redistribute their houses among themselves like TTCA allocation – no agent is worse off since everyone receives her best choice and at least one agent (who did not receive her best choice) is better off. So every core allocation must agree with TTCA allocation on allocation of houses to agents in $N_1$. TTCA allocates best choice among houses in $N_2 \cup \cdots \cup N_\ell$ to the agents in $N_2$. Every core allocation $k$ must agree with TTCA allocation for agents in $N_2$ also. Otherwise, since $k$ already agrees with TTCA allocation on $N_1$, the agents in $N_2$ can form a coalition and redistribute their houses among themselves like TTCA allocation – no agent is worse off since everyone receives her best choice in $N_2 \cup \cdots \cup N_\ell$ and at least one agent (who did not receive her best choice in $N_2 \cup \cdots \cup N_\ell$) is better off. Hence every core allocations must agree with TTCA allocation on allocation of houses to agents in $N_1 \cup N_2$. Continuing inductively, we obtain the result. □

## Acknowledgement

I have closely followed the following material for preparing this lecture note.

- ▷ Nisan/Roughgarden/Tardos/Vazirani (eds), Algorithmic Game Theory, Cambridge University, 2007 (available for free from here).
- ▷ Game Theory by Michael Maschler, Eilon Solan, and Shmuel Zamir.
- ▷ Game Theory and Mechanism Design by Y. Narahari.
- ▷ Lecture notes of Prof. Tim Roughgarden

I also thank all the students of the CS60025 Algorithmic Game Theory course in the years 2019 and 2020 for their invaluable inputs with special mention to Aditya Anand, Arnab Maiti, Madhu Sudan Reddy, Nishant Baranwal Somy, Debajyoti Kar, and Manideep Mamindlapally.

# Bibliography


[ABHM10] Itai Ashlagi, Mark Braverman, Avinatan Hassidim, and Dov Monderer. Monotonicity and implementability. *Econometrica*, 78(5):1749–1772, 2010.

[BCL+06] Sushil Bikhchandani, Shurojit Chatterji, Ron Lavi, Ahuva Mu'alem, Noam Nisan, and Arunava Sen. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica*, 74(4):1109–1132, 2006.

[Ben00] Jean-Pierre Benoıt. The gibbard–satterthwaite theorem: a simple proof. *Econ. Lett.*, 69(3):319–322, 2000.

[DMP06] Constantinos Daskalakis, Aranyak Mehta, and Christos Papadimitriou. A note on approximate nash equilibria. In *International Workshop on Internet and Network Economics*, pages 297–306. Springer, 2006.

[EY10] Kousha Etessami and Mihalis Yannakakis. On the complexity of nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010.

[GZ89] Itzhak Gilboa and Eitan Zemel. Nash and correlated equilibria: Some complexity considerations. *GEB*, 1(1):80–93, 1989.

[JPY88] David S Johnson, Christos H Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci*, 37(1):79–100, 1988.

[MP91] Nimrod Megiddo and Christos H Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

[MS96] Dov Monderer and Lloyd S Shapley. Potential games. *GEB*, 14(1):124–143, 1996.

[NRTV07] Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[Ros73] Robert W Rosenthal. A class of games possessing pure-strategy nash equilibria. *Int. J. Game Theory*, 2(1):65–67, 1973.

[Rot08] Alvin E. Roth. Deferred acceptance algorithms: history, theory, practice, and open questions. *Int. J. Game Theory*, 36(3-4):537–569, 2008.

[Rou10] Tim Roughgarden. Computing equilibria: a computational complexity perspective. *Economic Theory*, 42(1):193–236, 2010.

[Sch91] Alejandro A Schäffer. Simple local search problems that are hard to solve. *SIAM J. Comput*, 20(1):56–87, 1991.

[SS74] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37, 1974.

[Tre] Luca Trevisan. Duality in linear programming. https://lucatrevisan.wordpress.com/2011/01/24/cs261-lecture-6-duality-in-linear-programming/. Accessed: 2019-07-21.

[WHJ+] David Wells, Roger A Horn, Charles R Johnson, Michael Maschler, Eilon Solan, Shmuel Zamir, Robin Pemantle, Mark C Wilson, Joachim von zur Gathen, Jürgen Gerhard, et al. Great new titles in combinatorics from cambridge university press.