CHAPTER 9

# Introduction to Mechanism Design (for Computer Scientists)

Noam Nisan

## Abstract

We give an introduction to the micro-economic field of Mechanism Design slightly biased toward a computer-scientist's point of view.

## 9.1 Introduction

*Mechanism Design* is a subfield of economic theory that is rather unique within economics in having an engineering perspective. It is interested in designing economic mechanisms, just like computer scientists are interested in designing algorithms, protocols, or systems. It is best to view the goals of the designed mechanisms in the very abstract terms of *social choice*. A social choice is simply an aggregation of the preferences of the different participants toward a single joint decision. *Mechanism Design* attempts implementing desired social choices in a strategic setting – assuming that the different members of society each act *rationally* in a game theoretic sense. Such strategic design is necessary since usually the preferences of the participants are private.

This high-level abstraction of aggregation of preferences may be seen as a common generalization of a multitude of scenarios in economics as well as in other social settings such as political science. Here are some basic classic examples:

- **Elections:** In political elections each voter has his own preferences between the different candidates, and the outcome of the elections is a single social choice.
- **Markets:** Classical economic theory usually assumes the existence and functioning of a "perfect market." In reality, of course, we have only interactions between people, governed by some protocols. Each participant in such an interaction has his own preferences, but the outcome is a single social choice: the reallocation of goods and money.
- **Auctions:** Generally speaking, the more buyers and sellers there are in a market, the more the situation becomes close to the perfect market scenario. An extreme opposite

case is where there is only a single seller – an auction. The auction rules define the social choice: the identity of the winner.

- **Government policy:** Governments routinely have to make decisions that affect a multitude of people in different ways: Should a certain bridge be built? How much pollution should we allow? How should we regulate some sector? Clearly each citizen has a different set of preferences but a single social choice is made by the government.

As the influence of the Internet grew, it became clear that many scenarios happening there can also be viewed as instances of social choice in strategic settings. The main new ingredient found in the Internet is that it is owned and operated by different parties with different goals and preferences. These preferences, and the behavior they induce, must then be taken into account by every protocol in such an environment. The protocol should thus be viewed as taking the preferences of the different participants and aggregating them into a social choice: the outcome of the run of the protocol.

Conceptually, one can look at two different types of motivations: those that use economics to solve computer science issues and those that use computer science to solve economic issues:

- **Economics for CS:** Consider your favorite algorithmic challenge in a computer network environment: routing of messages, scheduling of tasks, allocation of memory, etc. When running in an environment with multiple owners of resources or requests, this algorithm must take into account the different preferences of the different owners. The algorithm should function well assuming strategic selfish behavior of each participant. Thus we desire a Mechanism Design approach for a multitude of algorithmic challenges – leading to a field that has been termed *Algorithmic Mechanism Design*.
- **CS for economics:** Consider your favorite economic interaction: some type of market, an auction, a supply chain, etc. As the Internet becomes ubiquitous, this interaction will often be implemented over some computerized platform. Such an implementation enables unprecedented sophistication and complexity, handled by hyperrationally designed software. Designing these is often termed *Electronic Market Design*.

Thus, both Algorithmic Mechanism Design and Electronic Market Design can be based upon the field of Mechanism Design applied in complex algorithmic settings.

This chapter provides an introduction to classical Mechanism Design, intended for computer scientists. While the presentation is not very different from the standard economic approach, it is somewhat biased toward a worst-case (non-Bayesian) point of view common in computer science.

Section 9.2 starts with the general formulation of the social choice problem, points out the basic difficulties formulated by Arrow's famous impossibility results, and deduces the impossibility of a general strategic treatment, i.e. of Mechanism Design in the general setting. Section 9.3 then considers the important special case where "money" exists, and describes a very general positive result, the incentive-compatible Vickrey–Clarke–Grove mechanism. Section 9.4 puts everything in a wider formal context of implementation in dominant strategies. Section 9.5 provides several characterizations of dominant strategy mechanisms. All the sections up to this point have considered dominant strategies, but the prevailing economic point of view is a Bayesian one that assumes a priori known distributions over private information. Section 9.6 introduces

## 9.3 Mechanisms with Money

In the previous section, we modeled a voter's preference as an order on the alternatives. $a \succ_i b$ implies that $i$ prefers $a$ to $b$, but we did not model "by how much" is $a$ preferred to $b$. "Money" is a yardstick that allows measuring this. Moreover, money can be transferred between players. The existence of money with these properties is an assumption, but a fairly reasonable one in many circumstances, and will allow us to do things that we could not do otherwise.

Formally, in this section we redefine our setting. We will still have a set of alternatives $A$ and a set of $n$ players $I$ (which we will no longer call voters). The preference of a player $i$ is now given by a *valuation function* $v_i : A \to \Re$, where $v_i(a)$ denotes the "value" that $i$ assigns to alternative $a$ being chosen. This value is in terms of some currency; i.e., we assume that if $a$ is chosen and then player $i$ is additionally given some quantity $m$ of money, then $i$'s *utility* is $u_i = v_i(a) + m$, this utility being the abstraction of what the player desires and aims to maximize. Utilities of this form are called *quasilinear preferences*, denoting the separable and linear dependence on money.

### 9.3.1 Vickrey's Second Price Auction

Before we proceed to the general setting, in this subsection we study a basic example: a simple auction. Consider a single item that is auctioned for sale among $n$ players. Each player $i$ has a scalar value $w_i$ that he is "willing to pay" for this item. More specifically, if he wins the item, but has to pay some price $p$ for it, then his utility is $w_i - p$, while if someone else wins the item then $i$'s utility is 0. Putting this scenario into the terms of our general setting, the set of alternatives here is the set of possible winners, $A = \{i\text{–wins} | i \in I\}$, and the valuation of each bidder $i$ is $v_i(i\text{–wins}) = w_i$ and $v_i(j\text{–wins}) = 0$ for all $j \neq i$. A natural social choice would be to allocate the item to the player who values it highest: choose $i$–wins, where $i = \text{argmax}_j w_j$. However, the challenge is that we do not know the values $w_i$ but rather each player knows his own value, and we want to make sure that our mechanism decides on the allocation – the social choice – in a way that *cannot be strategically manipulated*. Our degree of freedom is the definition of the payment by the winner.

Let us first consider the two most natural choices of payment and see why they do not work as intended:

- **No payment:** In this version we give the item for free to the player with highest $w_i$. Clearly, this method is easily manipulated: every player will benefit by exaggerating his $w_i$, reporting a much larger $w_i' \gg w_i$ that can cause him to win the item, even though his real $w_i$ is not the highest.
- **Pay your bid:** An attempt of correction will be to have the winner pay the declared bid. However, this system is also open to manipulation: a player with value $w_i$ who wins and pays $w_i$ gets a total utility of 0. Thus it is clear that he should attempt declaring a somewhat lower value $w_i' < w_i$ that still wins. In this case he can still win the item getting a value of $w_i$ (his real value) but paying only the smaller $w_i'$ (his declared value), obtaining a net positive utility $u_i = w_i - w_i' > 0$. What value $w_i'$ should $i$ bid then?

Well, if $i$ knows the value of the second highest bid, then he should declare just above it. But what if he does not know?

Here is the solution.

**Definition 9.12 Vickrey's second price auction:** Let the winner be the player $i$ with the highest declared value of $w_i$, and let $i$ pay the second highest declared bid $p^* = \max_{j \neq i} w_j$.

Now it turns out that manipulation never can increase any players' utility. Formally,

**Proposition 9.13 (Vickrey)** *For every $w_1, \ldots, w_n$ and every $w_i'$, Let $u_i$ be i's utility if he bids $w_i$ and $u_i'$ his utility if he bids $w_i'$. Then, $u_i \geq u_i'$.*

**PROOF** Assume that by saying $w_i$ he wins, and that the second highest (reported) value is $p^*$, then $u_i = w_i - p^* \geq 0$. Now, for an attempted manipulation $w_i' > p^*$, $i$ would still win if he bids $w_i'$ and would still pay $p^*$, thus $u_i' = u_i$. On the other hand, for $w_i' \leq p^*$, $i$ would lose so $u_i' = 0 \leq u_i$.

If $i$ loses by bidding $w_i$, then $u_i = 0$. Let $j$ be the winner in this case, and thus $w_j \geq w_i$. For $w_i' < w_j$, $i$ would still lose and so $u_i' = 0 = u_i$. For $w_i' \geq w_j$, $i$ would win, but would pay $w_j$, thus his utility would be $u_i' = w_i - w_j \leq 0 = u_i$. □

This very simple and elegant idea achieves something that is quite remarkable: it reliably computes a function (argmax) of $n$ numbers (the $w_i$'s) that are each held secretly by a different self-interested player! Taking a philosophical point of view, this may be seen as the mechanics for the implementation of Adam Smith's *invisible hand*: despite private information and pure selfish behavior, social welfare is achieved. All the field of Mechanism Design is just a generalization of this possibility.

### 9.3.2 Incentive Compatible Mechanisms

In a world with money, our mechanisms will not only choose a social alternative but will also determine monetary payments to be made by the different players. The complete social choice is then composed of the alternative chosen as well as of the transfer of money. Nevertheless, we will refer to each of these parts separately, calling the alternative chosen the social choice, not including in this term the monetary payments.

Formally, a mechanism needs to socially choose some alternative from $A$, as well as to decide on payments. The preference of each player $i$ is modeled by a valuation function $v_i : A \to \Re$, where $v_i \in V_i$. Throughout the rest of this chapter, $V_i \subseteq \Re^A$ is a commonly known set of possible valuation functions for player $i$.

Starting at this point and for the rest of this chapter, it will be convenient to use the following standard notation.

**Notation** Let $v = (v_1, \ldots, v_n)$ be an $n$-dimensional vector. We will denote the $(n-1)$-dimensional vector in which the $i$'th coordinate is removed by $v_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$. Thus we have three equivalent notations: $v = (v_1, \ldots, v_n) = (v_i, v_{-i})$. Similarly, for $V = V_1 \times \cdots \times V_n$, we will denote $V_{-i} = V_1 \times \cdots \times V_{i-1} \times V_{i+1} \times \cdots \times V_n$. Similarly we will use $t_{-i}$, $x_{-i}$, $X_{-i}$, etc.

**Definition 9.14** A (direct revelation) *mechanism* is a social choice function $f : V_1 \times \cdots \times V_n \to A$ and a vector of payment functions $p_1, \ldots, p_n$, where $p_i : V_1 \times \cdots \times V_n \to \Re$ is the amount that player $i$ pays.

The qualification "direct revelation" will become clear in Section 9.4, where we will generalize the notion of a mechanism further. We are now ready for the key definition in this area, *incentive compatibility* also called *strategy-proofness* or *truthfulness*.

**Definition 9.15** A mechanism $(f, p_1, \ldots, p_n)$ is called incentive compatible if for every player $i$, every $v_1 \in V_1, \ldots, v_n \in V_n$ and every $v_i' \in V_i$, if we denote $a = f(v_i, v_{-i})$ and $a' = f(v_i', v_{-i})$, then $v_i(a) - p_i(v_i, v_{-i}) \geq v_i(a') - p_i(v_i', v_{-i})$.

Intuitively this means that player $i$ whose valuation is $v_i$ would prefer "telling the truth" $v_i$ to the mechanism rather than any possible "lie" $v_i'$, since this gives him higher (in the weak sense) utility.

### 9.3.3 Vickrey–Clarke–Groves Mechanisms

While in the general setting without money, as we have seen, nothing nontrivial is incentive compatible, the main result in this setting is positive and provides an incentive compatible mechanism for the most natural social choice function: optimizing the social welfare. The social welfare of an alternative $a \in A$ is the sum of the valuations of all players for this alternative, $\sum_i v_i(a)$.

**Definition 9.16** A mechanism $(f, p_1, \ldots, p_n)$ is called a Vickrey–Clarke–Groves (VCG) mechanism if

- $f(v_1, \ldots, v_n) \in \operatorname{argmax}_{a \in A} \sum_i v_i(a)$; that is, $f$ maximizes the social welfare, and
- for some functions $h_1, \ldots, h_n$, where $h_i : V_{-i} \to \Re$ (i.e., $h_i$ does not depend on $v_i$), we have that for all $v_1 \in V_1, \ldots, v_n \in V_n$: $p_i(v_1, \ldots, v_n) = h_i(v_{-i}) - \sum_{j \neq i} v_j(f(v_1, \ldots, v_n))$.

The main idea lies in the term $-\sum_{j \neq i} v_j(f(v_1, \ldots, v_n))$, which means that each player is paid an amount equal to the sum of the values of all other players. When this term is added to his own value $v_i(f(v_1, \ldots, v_n))$, the sum becomes exactly the total social welfare of $f(v_1, \ldots, v_n)$. Thus this mechanism aligns all players' incentives with the social goal of maximizing social welfare, which is exactly archived by telling the truth. The other term in the payment $h_i(v_i)$ has no strategic implications for player $i$ since it does not depend, in any way, on what he says, and thus from player $i$'s point of view it is just a constant. Of course, the choice of $h_i$ does change significantly how

much money is paid and in which direction, but we will postpone this discussion. What we have just intuitively explained is as follows.

**Theorem 9.17 (Vickrey–Clarke–Groves)** *Every VCG mechanism is incentive compatible.*

Let us prove it formally.

**PROOF** Fix $i$, $v_{-i}$, $v_i$, and $v_i'$. We need to show that for player $i$ with valuation $v_i$, the utility when declaring $v_i$ is not less than the utility when declaring $v_i'$. Denote $a = f(v_i, v_{-i})$ and $a' = f(v_i', v_{-i})$. The utility of $i$, when declaring $v_i$, is $v_i(a) + \sum_{j \neq i} v_j(a) - h_i(v_{-i})$, but when declaring $v_i'$ is $v_i(a') + \sum_{j \neq i} v_j(a') - h_i(v_{-i})$. But since $a = f(v_i, v_{-i})$ maximizes social welfare over all alternatives, $v_i(a) + \sum_{j \neq i} v_j(a) \geq v_i(a') + \sum_{j \neq i} v_j(a')$ and thus the same inequality holds when subtracting the same term $h_i(v_{-i})$ from both sides. □

### 9.3.4 Clarke Pivot Rule

Let us now return to the question of choosing the "right" $h_i$'s. One possibility is certainly choosing $h_i = 0$. This has the advantage of simplicity but usually does not make sense since the mechanism pays here a great amount of money to the players. Intuitively we would prefer that players pay money to the mechanism, but not more than the gain that they get. Here are two conditions that seem to make sense, at least in a setting where all valuations are nonnegative.

**Definition 9.18**

- A mechanism is (ex-post) *individually rational* if players always get nonnegative utility. Formally if for every $v_1, \ldots, v_n$ we have that $v_i(f(v_1, \ldots, v_n)) - p_i(v_1, \ldots, v_n) \geq 0$.
- A mechanism has no positive transfers if no player is ever paid money. Formally if for every $v_1, \ldots, v_n$ and every $i$, $p_i(v_1, \ldots, v_n) \geq 0$.

The following choice of $h_i$'s provides the following two properties.

**Definition 9.19 (Clarke pivot rule)** The choice $h_i(v_{-i}) = \max_{b \in A} \sum_{j \neq i} v_i(b)$ is called the Clarke pivot payment. Under this rule the payment of player $i$ is $p_i(v_1, \ldots, v_n) = \max_b \sum_{j \neq i} v_i(b) - \sum_{j \neq i} v_i(a)$, where $a = f(v_1, \ldots, v_n)$.

Intuitively, $i$ pays an amount equal to the total damage that he causes the other players – the difference between the social welfare of the others with and without $i$'s participation. In other words, the payments make each player internalize the externalities that he causes.

**Lemma 9.20** *A VCG mechanism with Clarke pivot payments makes no positive transfers. If $v_i(a) \geq 0$ for every $v_i \in V_i$ and $a \in A$ then it is also individually rational.*

**PROOF** Let $a = f(v_1, \ldots, v_n)$ be the alternative maximizing $\sum_j v_j(a)$ and $b$ be the alternative maximizing $\sum_{j \neq i} v_j(b)$. To show individual rationality, the utility of player $i$ is $v_i(a) + \sum_{j \neq i} v_j(a) - \sum_{j \neq i} v_j(b) \geq \sum_j v_j(a) - \sum_j v_j(b) \geq 0$, where the first inequality is since $v_i(b) \geq 0$ and the second is since $a$ was chosen as to maximize $\sum_j v_j(a)$. To show no positive transfers, note that $p_i(v_1, \ldots, v_n) = \sum_{j \neq i} v_i(b) - \sum_{j \neq i} v_i(a) \geq 0$, since $b$ was chosen as to maximize $\sum_{j \neq i} v_j(b)$. □

As stated, the Clarke pivot rule does not fit many situations where valuations are negative; i.e., when alternatives have costs to the players. Indeed, with the Clarke pivot rule, players always pay money to the mechanism, while the natural interpretation in case of costs would be the opposite. The spirit of the Clarke pivot rule in such cases can be captured by a modified rule that chooses $b$ as to maximize the social welfare "when $i$ does not participate" where the exact meaning of this turns out to be quite natural in most applications.

## 9.3.5 Examples

### *9.3.5.1 Auction of a Single Item*

The Vickrey auction that we started our discussion with is a special case of a VCG mechanism with the Clarke pivot rule. Here $A = \{i\text{–wins} | i \in I\}$. Each player has value 0 if he does not get the item, and may have any positive value if he does win the item, thus $V_i = \{v_i | v_i(i\text{–wins}) \geq 0 \text{ and } \forall j \neq i,\ v_i(j\text{–wins}) = 0\}$. Notice that finding the player with highest value is exactly equivalent to maximizing $\sum_i v_i(i)$ since only a single player gets nonzero value. VCG payments using the Clarke pivot rule give exactly Vickrey's second price auction.

### *9.3.5.2 Reverse Auction*

In a reverse auction (procurement auction) the bidder wants to *procure* an item from the bidder with lowest cost. In this case the valuation spaces are given by $V_i = \{v_i | v_i(i\text{–wins}) \leq 0 \text{ and } \forall j \neq i\ v_i(j\text{–wins}) = 0\}$, and indeed procuring the item from the lowest cost bidder is equivalent to maximizing the social welfare. The natural VCG payment rule would be for the mechanism to pay to the lowest bidder an amount equal to the second lowest bid, and pay nothing to the others. This may be viewed as capturing the spirit of the pivot rule since the second lowest bid is what would happen "without $i$."

### *9.3.5.3 Bilateral Trade*

In the bilateral trade problem a seller holds an item and values it at some $0 \leq v_s \leq 1$ and a potential buyer values it at some $0 \leq v_b \leq 1$. (The constants 0 and 1 are arbitrary and may be replaced with any commonly known constants $0 \leq v_l \leq v_h$.) The possible outcomes are $A = \{\text{no–trade}, \text{trade}\}$ and social efficiency implies that trade is chosen if $v_b > v_s$ and *no-trade* if $v_s > v_b$. Using VCG payments and decreeing that no payments be made in case of *no-trade*, implies that in case of trade the buyer pays $v_s$ and the seller is paid $v_b$. Notice that since in this case $v_b > v_s$,

the mechanism subsidizes the trade. As we will see below in Section 9.5.5, this is unavoidable.

#### 9.3.5.4 Multiunit Auctions

In a multiunit auction, $k$ identical units of some good are sold in an auction (where $k < n$). In the simple case each bidder is interested in only a single unit. In this case $A = \{S\text{–wins}|S \subset I,\ |S| = k\}$, and a bidder's valuation $v_i$ gives some fixed value $v^*$ if $i$ gets an item, i.e. $v_i(S) = v^*$ if $i \in S$ and $v_i(S) = 0$ otherwise. Maximizing social welfare means allocating the items to the $k$ highest bidders, and in the VCG mechanism with the pivot rule, each of them should pay the $k + 1$'st highest offered price. (Losers pay 0.)

In a more general case, bidders may be interested in more than a single unit and have a different value for each number of units obtained. The next level of sophistication comes when the items in the auction are heterogeneous, and valuations can give a different value to each combination of items. This is called a combinatorial auction and is studied at length in Chapter 11.

#### 9.3.5.5 Public Project

The government is considering undertaking a public project (e.g., building a bridge). The project has a commonly known cost $C$, and is valued by each citizen $i$ at (a privately known) value $v_i$. (We usually think that $v_i \geq 0$, but the case of allowing $v_i < 0$, i.e., citizens who are hurt by the project is also covered.) Social efficiency means that the government will undertake this project iff $\sum_i v_i > C$. (This is not technically a subcase of our definition of maximizing the social welfare, since our definition did not assume any costs or values for the designer, but becomes so by adding an extra player "government" whose valuation space is the singleton valuation, giving cost $C$ to undertaking the project and 0 otherwise.) The VCG mechanism with the Clarke pivot rule means that a player $i$ with $v_i \geq 0$ will pay a nonzero amount only if he is pivotal: $\sum_{j \neq i} v_j \leq C$ but $\sum_j v_j > C$ in which case he will pay $p_i = C - \sum_{j \neq i} v_j$. (A player with $v_i < 0$ will make a nonzero payment only if $\sum_{j \neq i} v_j > C$ but $\sum_j v_j \leq C$ in which case he will pay $p_i = \sum_{j \neq i} v_j - C$.) One may verify that $\sum_i p_i < C$ (unless $\sum_i v_i = C$), and thus the payments collected do not cover the project's costs. As we will see in Section 9.5.5, this is unavoidable.

#### 9.3.5.6 Buying a Path in a Network

Consider a communication network, modeled as a directed graph $G = (V, E)$, where each link $e \in E$ is owned by a different player, and has a cost $c_e \geq 0$ if his link is used for carrying some message. Suppose that we wish to procure a communication path between two specified vertices $s, t \in V$; i.e., the set of alternatives is the set of all possible $s - t$ paths in $G$, and player $e$ has value 0 if the path chosen does not contain $e$ and value $-c_e$ if the path chosen does contain $e$. Maximizing social welfare means finding the shortest path $p$ (in terms of $\sum_{e \in p} c_e$). A VCG mechanism that makes no payments to edges that are not in $p$, will pay to each $e_0 \in p$ the quantity $\sum_{e \in p'} c_e - \sum_{e \in p - \{e_0\}} c_e$, where $p$ is the shortest $s - t$ path in $G$ and $p'$ is the shortest

$s - t$ path in $G$ that does not contain the edge $e$ (for simplicity, assume that $G$ is 2-edge connected so such a $p'$ always exists). This corresponds to the spirit of the pivot rule since "without $e$" the mechanism can simply not use paths that contain $e$.

## 9.4 Implementation in Dominant Strategies

In this section our aim is to put the issue of incentive compatibility in a wider context. The mechanisms considered so far extract information from the different players by motivating them to "tell the truth." More generally, one may think of other, indirect, methods of extracting sufficient information from the participants. Perhaps one may devise some complex protocol that achieves the required social choice when players act strategically. This section will formalize these more general mechanisms, and the associated notions describing what happens when "players act strategically."

Deviating from the common treatment in economics, in this section we will describe a model that does not involve any distributional assumptions. Many of the classical results of Mechanism Design are captured in this framework, including most of the existing applications in computational settings. In Section 9.6 we will add this ingredient of distributional assumptions reaching the general "Bayesian" models.

### 9.4.1 Games with Strict Incomplete Information

How do we model strategic behavior of the players when they are missing some of the information that specifies the game? Specifically in our setting a player does not know the private information of the other players, information that determines their preferences. The standard setting in Game Theory supposes on the other hand that the "rules" of the game, including the utilities of all players, are public knowledge.

We will use a model of games with *independent private values* and *strict incomplete information*. Let us explain the terms: "independent private values" means that the utility of a player depends fully on his private information and not on any information of others as it is independent from his own information. *Strict incomplete information* is a (not completely standard) term that means that we will have no probabilistic information in the model. An alternative term sometimes used is "pre-Bayesian." From a CS perspective, it means that we will use a worst case analysis over unknown information. So here is the model.

**Definition 9.21** A game with (independent private values and) strict incomplete information for a set of $n$ players is given by the following ingredients:

**(i)** For every player $i$, a set of *actions* $X_i$.

**(ii)** For every player $i$, a set of *types* $T_i$. A value $t_i \in T_i$ is the private information that $i$ has.

**(iii)** For every player $i$, a *utility function* $u_i : T_i \times X_1 \times \cdots \times X_n \to \Re$, where $u_i(t_i, x_1, \ldots, x_n)$ is the utility achieved by player $i$, if his type (private information) is $t_i$, and the profile of actions taken by all players is $x_1, \ldots, x_n$.

The main idea that we wish to capture with this definition is that each player $i$ must choose his action $x_i$ when knowing $t_i$ but not the other $t_j$'s. Note that the $t_j$'s do not

CHAPTER 20

# Selfish Load Balancing

Berthold Vöcking

## Abstract

Suppose that a set of weighted tasks shall be assigned to a set of machines with possibly different speeds such that the load is distributed evenly among the machines. In computer science, this problem is traditionally treated as an optimization problem. One of the classical objectives is to minimize the *makespan*, i.e., the maximum load over all machines. Here we study a natural game theoretic variant of this problem: We assume that the tasks are managed by selfish agents, i.e., each task has an agent that aims at placing the task on the machine with smallest load. We study the Nash equilibria of this game and compare them with optimal solutions with respect to the makespan. The ratio between the worst-case makespan in a Nash equilibrium and the optimal makespan is called the *price of anarchy*. In this chapter, we study the price of anarchy for such load balancing games in four different variants, and we investigate the complexity of computing equilibria.

## 20.1 Introduction

The problem of load balancing is fundamental to networks and distributed systems. Whenever a set of tasks should be executed on a set of resources, one needs to balance the load among the resources in order to exploit the available resources efficiently. Often also fairness aspects have to be taken into account. Load balancing has been studied extensively and in many variants. One of the most fundamental load balancing problems is *makespan scheduling on uniformly related machines*. This problem is defined by $m$ *machines* with speeds $s_1, \ldots, s_m$ and $n$ *tasks* with weights $w_1, \ldots, w_n$. Let $[n] = \{1, \ldots, n\}$ denote the set of tasks and $[m] = \{1, \ldots, m\}$ the set of machines. One seeks for an assignment $A : [n] \to [m]$ of the tasks to the machines that is as balanced as possible. The *load* of machine $j \in [m]$ under assignment $A$ is defined as

$$\ell_j = \sum_{\substack{i \in [n] \\ j = A(i)}} \frac{w_i}{s_j}.$$

The *makespan* is defined to be the maximum load over all machines. The objective is to minimize the makespan. If all machines have the same speed, then the problem is

known as *makespan scheduling on identical machines*, in which case we shall assume $s_1 = s_2 = \cdots = s_m = 1$.

In computer science, load balancing is traditionally viewed as an algorithmic problem. We design and analyze algorithms, either centralized or distributed, that compute the mapping $A$. Suppose, however, there is no global authority that can enforce an efficient mapping of the tasks to the machines. For example, in a typical Internet application, tasks might correspond to requests for downloading large files that users send to servers. To maximize the quality of service, each of the users aims at contacting a server with smallest load. This naturally leads to the following game theoretic setting in which we will be able to analyze what happens to the makespan if there is no global authority but selfish users aiming at maximizing their individual benefit decide about the assignment of tasks to machines.

This chapter differs from the other chapters in Part III of this book in two important aspects. At first, the considered objective function, the makespan, is nonutilitarian. At second, our analysis does not only consider pure but also mixed equilibria. By using the makespan as objective function, our analysis simultaneously captures the aspects of efficiency and fairness. By considering mixed equilibria, our analysis explicitly takes into account the effects of uncoordinated random behavior.

### 20.1.1 Load Balancing Games

We identify agents and tasks, i.e., task $i \in [n]$ is managed by agent $i$. Each agent can place its task on one of the machines. In other words, the set of *pure strategies* for an agent is $[m]$. A combination of pure strategies, one for each task, yields an assignment $A : [n] \to [m]$. We assume that the *cost* of agent $i$ under the assignment $A$ corresponds to the load on machine $A(i)$, i.e., its cost is $\ell_{A(i)}$. The *social cost* of an assignment is denoted $\mathrm{cost}(A)$ and is defined to be the makespan, i.e., $\mathrm{cost}(A) = \max_{j\in[m]} \left(\ell_j\right)$.

Agents may use *mixed strategies*, i. e., probability distributions on the set of pure strategies. Let $p_i^j$ denote the probability that agent $i$ assigns its task to machine $j$, i.e., $p_i^j = \mathbb{P}[A(i) = j]$. A *strategy profile* $P = (p_i^j)_{i\in[n], j\in[m]}$ specifies the probabilities for all agents and all machines. Clearly, every strategy profile $P$ induces a random mapping $A$. For $i \in [n]$, $j \in [m]$, let $x_i^j$ be a random variable that takes the value 1 if $A(i) = j$ and 0, otherwise. The expected load of machine $j$ under the strategy profile $P$ is thus

$$\mathbb{E}[\ell_j] = \mathbb{E}\left[\sum_{i\in[n]} \frac{w_i\, x_i^j}{s_j}\right] = \sum_{i\in[n]} \frac{w_i\, \mathbb{E}[x_i^j]}{s_j} = \sum_{i\in[n]} \frac{w_i\, p_i^j}{s_j}.$$

The *social cost of a strategy profile* $P$ is defined as the expected makespan, i.e.,

$$\mathrm{cost}(P) = \mathbb{E}[\mathrm{cost}(A)] = \mathbb{E}\Big[\max_{j\in[m]} \left(\ell_j\right)\Big] .$$

We assume that every agent aims at minimizing its expected cost. From point of view of agent $i$, the expected cost on machine $j$, denoted by $c_i^j$, is $c_i^j = \mathbb{E}[\ell_j \mid A(i) = j]$.

For any profile $P$,

$$c_i^j = \frac{w_i + \sum_{k \neq i} w_k \, p_k^j}{s_j} = \mathbb{E}[\ell_j] + (1 - p_i^j) \cdot \frac{w_i}{s_j}. \tag{20.1}$$

In general, a strategy profile of a game is a *Nash equilibrium* if there is no incentive for any agent to unilaterally change its strategy. For the load balancing game, such a profile is characterized by the property that every agent assigns positive probabilities only to those machines that minimize its expected cost. This is formalized as follows.

**Proposition 20.1** *A strategy profile $P$ is a Nash equilibrium if and only if* $\forall i \in [n] : \forall j \in [m] : p_i^j > 0 \Rightarrow \forall k \in [m] : c_i^j \leq c_i^k$.

The existence of a Nash equilibrium in mixed strategies is guaranteed by the theorem of Nash, see Chapters 1 and 2. A strategy profile $P$ is called *pure* if, for each agent, there exists only one machine with positive probability. A Nash equilibrium in pure strategies is called a *pure Nash equilibrium*. Applying the proposition above to pure profiles and the corresponding assignments yields the following characterization of a pure Nash equilibrium.

**Proposition 20.2** *An assignment $A$ is a pure Nash equilibrium if and only if* $\forall i \in [n] : \forall k \in [m] : c_i^{A(i)} \leq c_i^k$.

In words, an assignment is a pure Nash equilibrium if and only if no agent can improve its cost by unilaterally moving its task to another machine. A special property of load balancing games is that they always admit pure Nash equilibria.

**Proposition 20.3** *Every instance of the load balancing game admits at least one pure Nash equilibrium.*

**PROOF** An assignment $A$ induces a *sorted load vector* $(\lambda_1, \ldots, \lambda_m)$, where $\lambda_j$ denotes the load on the machine that has the $j$-th highest load. If an assignment is not a Nash equilibrium, then there exists an agent $i$ that can perform an *improvement step*, i.e., it can decrease its cost by moving its task to another machine. We show that the sorted load vector obtained after performing an improvement step is lexicographically smaller than the one preceding it. Hence, a pure Nash equilibrium is reached after a finite number of improvement steps.

Suppose, given any sorted load vector $(\lambda_1, \ldots, \lambda_m)$, agent $i$ performs an improvement step and moves its task from machine $j$ to machine $k$ where the indices are with respect to the positions of the machines in the sorted load vector. Clearly, $k > j$. The improvement step decreases the load on machine $j$ and it increases the load on machine $k$. However, the increased load on machine $k$ is smaller than $\lambda_j$ as, otherwise, agent $i$ would not decrease its cost. Hence, the number of machines with load at least $\lambda_j$ is decreasing. Furthermore, the loads on all other machines with load at least $\lambda_j$ are left unchanged. Consequently, the improvement step yields a sorted load vector i.e. lexicographically smaller than $(\lambda_1, \ldots, \lambda_m)$. □

Thus improvement steps naturally lead to a pure Nash equilibrium. This issue is also discussed for a broader class of games, so-called potential games, in Chapter 19. Let us remark that this convergence result implies that there exists even a pure Nash equilibrium that minimizes the makespan. Given any optimal assignment, such an equilibrium can be found by performing improvement steps until a Nash equilibrium is reached because improvement steps do not increase the makespan. Thus, for load balancing games with social cost equal to the makespan, it does not make much sense to study the ratio between the social cost in a best Nash equilibrium and the optimal social cost. This ratio is called the "price of stability." It is studied in Chapters 17–19 in the context of other games. In this chapter, we are mainly interested in the ratio between the social cost of the worst Nash equilibrium and the optimal social cost, the so-called the "price of anarchy."

### 20.1.2 Example of a Load Balancing Game

Suppose that there are two identical machines both of which have speed 1 and four tasks, two *small tasks* of weight 1 and two *large tasks* of weight 2. An optimal assignment would map a small and a large task to each of the machines so that the load on both machines is 3. This assignment is illustrated in Figure 20.1(a).

Now consider an assignment $A$ that maps the two large tasks to the first machine and the two small tasks to the second machine as illustrated in Figure 20.1(b). This way, the first machine has a load of 4 and the second machine has a load of 2. Obviously, a small task cannot improve its cost by moving from the second to the first machine. A large task cannot improve its cost by moving from the first to the second machine either as its cost would remain 4 if it does. Thus assignment $A$ constitutes a pure Nash equilibrium with $\text{cost}(A) = 4$. Observe that all assignments that yield a larger makespan than 4 cannot be a Nash equilibrium as, in this case, one of the machines has a load of at least 5 and the other has a load of at most 1 so that moving any task from the former to the latter would decrease the cost of this task. Thus, for this instance of the load balancing game, the social cost of the worst pure Nash equilibrium is 4.

Clearly, the worst mixed equilibrium cannot be better than the worst pure equilibrium as the set of mixed equilibria is a superset of the set of pure equilibria, but can it really be worse? Suppose that each task is assigned to each of the machines with probability

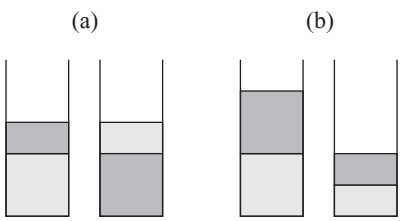



**Figure 20.1.** (a) Illustration of the optimal assignment of an instance of the load balancing game with two large tasks of size 2 and two small tasks of size 1 as described in the example given in Section 20.1.2. The social cost of this assignment is 3. (b) Illustration of a pure Nash equilibrium for the same instance. The social cost of this assignment is 4, which is the maximum among all pure Nash equilibria for this instance.

$\frac{1}{2}$. This corresponds to a strategy profile $P$ with $p_i^j = \frac{1}{2}$ for $1 \leq i \leq 4$, $1 \leq j \leq 2$. The expected load on machine $j$ is thus

$$\mathbb{E}[\ell_j] = \sum_{1 \leq i \leq 4} w_i \, p_i^j = 2 \cdot 2 \cdot \frac{1}{2} + 2 \cdot 1 \cdot \frac{1}{2} = 3.$$

It is important to notice that the expected cost of a task on a machine is larger than the expected load of the machine, unless the task is assigned with probability 1 to this machine. For example, if we assume that task 1 is a large task then Equation 20.1 yields

$$c_1^1 = \mathbb{E}[\ell_1] + (1 - p_1^1)\, w_1 = 3 + \frac{1}{2} \cdot 2 = 4,$$

and, if task 3 is a small task, then

$$c_3^1 = \mathbb{E}[\ell_1] + (1 - p_3^1)\, w_3 = 3 + \frac{1}{2} \cdot 1 = 3.5.$$

For symmetry reasons, the expected cost of each task under the considered strategy profile $P$ is the same on both machines so that $P$ is a Nash equilibrium. The social cost of this Nash equilibrium, $\text{cost}(P)$, is defined to be the expected makespan, $\mathbb{E}[\text{cost}(A)]$, of the random assignment $A$ induced by $P$. The makespan, $\text{cost}(A)$, is a random variable. This variable can possibly take one of the four values 3, 4, 5, or 6. There are $2^4 = 16$ different assignments of four tasks to two machines. The number of assignments that yield a makespan of 3 is 4, 4 is 6, 5 is 4, and 6 is 2. Consequently, the social cost of the mixed Nash equilibrium is

$$\text{cost}(P) = \mathbb{E}[\text{cost}(A)] = \frac{1}{16}\,(3 \cdot 4 + 4 \cdot 6 + 5 \cdot 4 + 6 \cdot 2) = 4.25.$$

Thus mixed equilibria can, in fact, be worse than the worst pure equilibrium.

### 20.1.3 Definition of the Price of Anarchy

Not surprisingly, the example above shows that uncoordinated, selfish behavior can lead to suboptimal assignments. We are interested in the ratio between the social cost (makespan) of a worst-case Nash equilibrium, i.e., the Nash equilibrium with highest social cost, and the social cost of an optimal assignment.

**Definition 20.4 (Price of anarchy)** For $m \in \mathbb{N}$, let $\mathcal{G}(m)$ denote the set of all instances of load balancing games with $m$ machines. For $G \in \mathcal{G}(m)$, let $\text{Nash}(G)$ denote the set of all strategy profiles being a Nash equilibrium for $G$, and let $\text{opt}(G)$ denote the minimum social cost over all assignments. Then the price of anarchy is defined by

$$\text{PoA}(m) = \max_{G \in \mathcal{G}(m)} \max_{P \in \text{Nash(G)}} \frac{\text{cost}(P)}{\text{opt}(G)}.$$

In the following, we study the price of anarchy in load balancing games in four different variants in which we distinguish, as a first criterion, between games with identical and uniformly related machines and, as a second criterion, between pure

Nash equilibria and mixed Nash equilibria. Technically, when considering the price of anarchy for load balancing games with identical machines then we restrict the set $\mathcal{G}(m)$ to instances in which the $m$ machines have all the same speed. When considering the price of anarchy with respect to pure equilibria then the set Nash(G) refers only to pure Nash equilibria rather than mixed equilibria; i.e., we take the maximum only among pure equilibrium assignments rather than among possibly mixed equilibrium strategy profiles.

The motivation behind studying the price of anarchy is to quantify the increase of the social cost due to selfish behavior. With this motivation in mind, does it make more sense to consider pure or mixed equilibria? If one wants to study a distributed system in which agents repeatedly perform improvement steps until they reach a Nash equilibrium, then pure equilibria are the right solution concept. However, there might be other means by which agents come to a Nash equilibrium. In particular, if one views load balancing games as one shot games, then mixed equilibria are a very reasonable solution concept. Moreover, upper bounds about the price of anarchy for mixed equilibria are more robust than upper bounds for pure equilibria as mixed equilibria are more general than pure ones. In this chapter, we consider both of these equilibrium concepts. Our analysis begins with the study of pure equilibria as they are usually easier to handle than mixed equilibria whose analysis requires a bit of probability theory.

## 20.2 Pure Equilibria for Identical Machines

Our analysis of equilibria in load balancing games begins with the most basic case, namely the study of pure equilibria on identical machines. Our first topic is the price of anarchy. As a second topic, we investigate how long it takes until a pure Nash equilibrium is reached when the agents repeatedly perform "best response" improvement steps.

### 20.2.1 The Price of Anarchy

In case of pure equilibria and identical machines, the analysis of the price of anarchy is quite similar to the well-known analysis of the greedy load balancing algorithm that assigns the tasks one after the other in arbitrary order giving each task to the least loaded machine. Graham (1966) has shown that the approximation factor of the greedy algorithm is $2 - \frac{1}{m}$. We show that the price of anarchy for pure equilibria is, in fact, slightly better than the approximation factor of the greedy algorithm.

**Theorem 20.5** *Consider an instance $G$ of the load balancing game with $n$ tasks of weight $w_1, \ldots, w_n$ and $m$ identical machines. Let $A : [n] \to [m]$ denote any Nash equilibrium assignment. Then, it holds that*

$$cost(A) \leq \left(2 - \frac{2}{m+1}\right) \cdot opt(G).$$

**PROOF** Let $j^*$ be a machine with the highest load under assignment $A$, and let $i^*$ be a task of smallest weight assigned to this machine. Without loss of generality, there are at least two tasks assigned to machine $j^*$ as, otherwise, $\text{cost}(A) = \text{opt}(G)$ so that the upper bound given in the theorem follows trivially. Thus $w_{i^*} \leq \frac{1}{2}\,\text{cost}(A)$.

Suppose there is a machine $j \in [n] \setminus \{j^*\}$ with load less than $\ell_{j^*} - w_{i^*}$. Then moving the task $i^*$ from $j^*$ to $j$ would decrease the cost for this task. Hence, as $A$ is a Nash equilibrium, it holds

$$\ell_j \geq \ell_{j^*} - w_{i^*} \geq \text{cost}(A) - \frac{1}{2}\,\text{cost}(A) = \frac{1}{2}\,\text{cost}(A).$$

Now observe that the cost of an optimal assignment cannot be smaller than the average load over all machines so that

$$\begin{aligned}
\text{opt}(G) &\geq \frac{\sum_{i\in[n]} w_i}{m} \\
&= \frac{\sum_{j\in[m]} \ell_j}{m} \\
&\geq \frac{\text{cost}(A) + \frac{1}{2}\,\text{cost}(A)(m-1)}{m} \\
&= \frac{(m+1)\text{cost}(A)}{2m}.
\end{aligned}$$

As a consequence,

$$\text{cost}(A) \leq \frac{2m}{m+1} \cdot \text{opt}(G) = \left(2 - \frac{2}{m+1}\right) \cdot \text{opt}(G).$$

□

Observe that the example of a game instance with two identical machines given in Section 20.1.2 has a price of anarchy of $\frac{4}{3} = 2 - \frac{2}{m+1}$, for $m = 2$. Exercise 20.2 generalizes this example. It shows that, for every $m \in \mathbb{N}$, there exists an instance $G$ of the load balancing game with $m$ identical machines and $2m$ tasks that has a Nash equilibrium assignment $A : [n] \to [m]$ with

$$\text{cost}(A) = \left(2 - \frac{2}{m+1}\right) \cdot \text{opt}(G).$$

Thus the upper bound on the price of anarchy given in Theorem 20.5 is tight.

### 20.2.2 Convergence Time of Best Responses

Our analysis about the price of anarchy leaves open the question of how agents may find or compute a Nash equilibrium efficiently. In the existence proof for pure equilibria in Proposition 20.3, we have implicitly shown that every sequence of improvement steps by the agents leads to a Nash equilibrium. However, if players do not converge to an equilibrium in reasonable time, then it might also not matter if the finally reached equilibrium is good. This naturally leads to the question of how many improvement steps are needed to reach a Nash equilibrium. The following result shows that, in case

of identical machines, there is a short sequence of improvement steps that leads from any given initial assignment to a pure Nash equilibrium. An agent is said to be *satisfied* if it cannot reduce its cost by unilaterally moving its task to another machine. The *max-weight best response policy* activates the agents one after the other always activating an agent with maximum weight among the unsatisfied agents. An activated agent plays a *best response*; i.e., the agent moves its task to the machine with minimum load.

**Theorem 20.6** *Let $A : [n] \to [m]$ denote any assignment of $n$ tasks to $m$ identical machines. Starting from $A$, the max-weight best response policy reaches a pure Nash equilibrium after each agent was activated at most once.*

**PROOF** We claim, once an agent $i \in [n]$ was activated and played its best response, it never gets unsatisfied again. This claim immediately implies the theorem. Our analysis starts with two observations both of which holding only for identical machines. At first, we observe that an agent is satisfied if and only if its task is placed on a machine on which the load due to the other tasks is minimal. At second, we observe that a best response never decreases the minimum load among the machines. As a consequence, a satisfied agent can get unsatisfied only for one reason: the load on the machine holding its task increases because another agent moves its task to the same machine. Suppose that agent $k$ is activated after agent $i$, and it moves its task to the machine holding task $i$. Let $j^*$ denote the machine on which $i$ is placed and to which $k$ is moved. For $j \in [m]$, let $\ell_j$ denote the load on machine $j$ at the time immediately after the best response of agent $k$. Since the assignment of $k$ to $j^*$ is a best response and as $w_k \leq w_i$ because of the max-weight policy, it follows

$$\ell_{j^*} \leq \ell_j + w_k \leq \ell_j + w_i,$$

for all $j \in [m]$. Hence, after the best response of $k$, agent $i$ remains satisfied on machine $j^*$ as it cannot reduce its cost by moving from $j^*$ to any other machine. □

Let us remark that the order in which the agents are activated is crucial. For example, if one would always activate an agent of minimum weight among the unsatisfied agents, then there are instances of load balancing games on identical machines where one needs an exponential number of best response steps to reach a pure Nash equilibrium (Even-Dar et al., 2003).

## 20.3 Pure Equilibria for Uniformly Related Machines

We now switch from identical to uniformly related machines. First, we study the price of anarchy. Then we discuss the complexity of computing equilibria.

### 20.3.1 The Price of Anarchy

The analysis in Section 20.2.1 shows that, in case of identical machines, the makespan of a pure Nash equilibrium is less than twice the optimal makespan. In this section, we

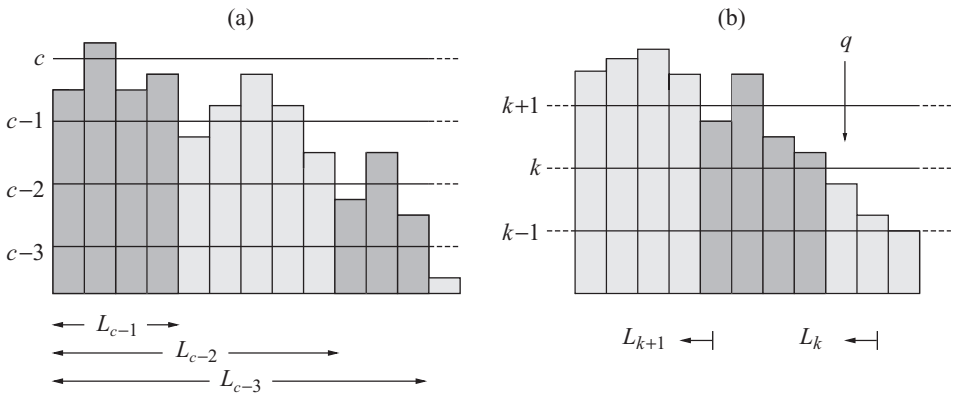



**Figure 20.2.** (a) Illustration of the definition of the lists $L_{c-1}, L_{c-2}, \ldots, L_0$ from the proof of Theorem 20.7. (b) Illustration of the lists $L_k$ and $L_{k+1}$ and the machine $q$ used in the proof of Lemma 20.8.

show that the makespan of pure equilibria on uniformly related machines can deviate by more than a constant factor. The price of anarchy is bounded, however, by a slowly growing function in the number of machines. Our analysis begins with an upper bound on the price of anarchy followed by the presentation of a family of game instances that match this upper bound up to a small constant factor.

**Theorem 20.7** *Consider an instance $G$ of the load balancing game with $n$ tasks of weight $w_1, \ldots, w_n$ and $m$ machines of speed $s_1, \ldots, s_m$. Let $A : [n] \to [m]$ denote any Nash equilibrium assignment. Then, it holds that*

$$cost(A) = \mathcal{O}\left(\frac{\log m}{\log \log m}\right) \cdot opt(G).$$

**PROOF** Let $c = \lfloor \text{cost}(A)/\text{opt}(G) \rfloor$. We show $c \leq \Gamma^{-1}(m)$, where $\Gamma^{-1}$ denotes the inverse of the *gamma function*, an extension of the factorial function with the property that $\Gamma(k) = (k-1)!$, for every positive integer $k$. This yields the theorem as

$$\Gamma^{-1}(m) = \Theta\left(\frac{\log m}{\log \log m}\right).$$

Without loss of generality, let us assume $s_1 \geq s_2 \geq \cdots \geq s_m$, and let $L = [1, 2, \ldots, m]$ denote the list of machines in nonincreasing order of speed. For $k \in \{0, \ldots, c-1\}$, let $L_k$ denote the maximum length prefix of $L$ such that the load of each server in $L_k$ is at least $k \cdot \text{opt}(G)$. Figure 20.2(a) illustrates this definition. We will show the following recurrence on the lengths of these lists.

$$|L_k| \geq (k+1) \cdot |L_{k+1}| \qquad (0 \leq k \leq c-2)$$
$$|L_{c-1}| \geq 1$$

Solving the recurrence yields $|L_0| \geq (c-1)! = \Gamma(c)$. Now observe that $L_0 = L$ and, hence, $|L_0| = m$. Consequently, $m \geq \Gamma(c)$ so that $c \leq \Gamma^{-1}(m)$, which proves the theorem.

It remains to prove the recurrence. We first prove $|L_{c-1}| \geq 1$. For the purpose of a contradiction, assume that the list $L_{c-1}$ is empty. Then the load of machine 1 is less than $(c-1) \cdot \text{opt}(G)$ in the equilibrium assignment $A$. Let $i$ be a task placed on a machine $j$ with load at least $c \cdot \text{opt}(G)$. Moving $i$ to machine 1 reduces the cost of $i$ to strictly less than

$$(c-1) \cdot \text{opt}(G) + \frac{w_i}{s_1} \leq (c-1) \cdot \text{opt}(G) + \text{opt}(G) \leq c \cdot \text{opt}(G),$$

where the inequality $\frac{w_i}{s_1} \leq \text{opt}(G)$ follows from the fact that $s_1$ is the speed of the fastest machine. Consequently, agent $i$ is able to unilaterally decrease its cost by moving its task from machine $j$ to machine 1, which contradicts the assumption that $A$ is a Nash equilibrium. Thus, we have shown that $|L_{c-1}| \geq 1$.

Next, we show $|L_k| \geq (k+1) \cdot |L_{k+1}|$, for $0 \leq k \leq c-2$. Let $A^*$ be an optimal assignment, i.e., an assignment whose makespan is equal to $\text{opt}(G)$. The following lemma relates the placement of tasks in the equilibrium assignment $A$ to the placement of tasks in the optimal assignment $A^*$.

**Lemma 20.8** *Suppose $i$ is a task with $A(i) \in L_{k+1}$. Then $A^*(i) \in L_k$.*

**PROOF** If $L \setminus L_k = \emptyset$ then this claim follows trivially. Let $q$ be the smallest index in $L \setminus L_k$, i.e., machine $q$ is one of the machines with maximum speed among the machines $L \setminus L_k$. By the definition of the group $L_k$, the load of $q$ is less than $k \cdot \text{opt}(G)$, i.e., $\ell_q < k \cdot \text{opt}(G)$. Figure 20.2(b) illustrates the situation.

By the definition of the groups, $A(i) \in L_{k+1}$ implies $\ell_{A(i)} \geq (k+1) \cdot \text{opt}(G)$. For the purpose of a contraction, assume $w_i \leq s_q \cdot \text{opt}(G)$. Then moving task $i$ to machine $q$ would reduce the cost of $i$ to

$$\ell_q + \frac{w_i}{s_q} < k \cdot \text{opt}(G) + \text{opt}(G) \leq \ell_{A(i)},$$

which contradicts the assumption that $A$ is a Nash equilibrium. Hence, every task $i$ with $A(i) \in L_{k+1}$ satisfies $w_i > s_q \cdot \text{opt}(G)$. Now, for the purpose of a contradiction, suppose $A^*(i) = j$ and $j \in L \setminus L_k$. Then the load on $j$ under $A^*$ would be at least

$$\frac{w_i}{s_j} > \frac{s_q \cdot \text{opt}(G)}{s_j} \geq \text{opt}(G)$$

because $s_j \leq s_q$. However, this contradicts that $A^*$ is an optimal assignment. Consequently, $A^*(i) \in L_k$. □

By the definition of $L_{k+1}$, the sum of the weights that $A$ assigns to a machine $j \in L_{k+1}$ is at least $(k+1) \cdot \text{opt}(G) \cdot s_j$. Hence, the total weight assigned to the machines in $L_{k+1}$ is at least $\sum_{j \in L_{k+1}} (k+1) \cdot \text{opt}(G) \cdot s_j$. By Lemma 20.8 an optimal assignment has to assign all this weight to the machines in $L_k$ such that

the load on each of these machines is at most $\mathrm{opt}(G)$. As a consequence,

$$\sum_{j \in L_{k+1}} (k+1) \cdot \mathrm{opt}(G) \cdot s_j \leq \sum_{j \in L_k} \mathrm{opt}(G) \cdot s_j.$$

Dividing by $\mathrm{opt}(G)$ and subtracting $\sum_{j \in L_{k+1}} s_j$ from both sides yields

$$\sum_{j \in L_{k+1}} k \cdot s_j \leq \sum_{j \in L_k \setminus L_{k+1}} s_j.$$

Now let $s^*$ denote the speed of the slowest machine in $L_{k+1}$, i.e., $s^* = s_{|L_{k+1}|}$. For all $j \in L_{k+1}$, $s_j \geq s^*$, and, for all $j \in L_k \setminus L_{k+1}$, $s_j \leq s^*$. Hence, we obtain

$$\sum_{j \in L_{k+1}} k \cdot s^* \leq \sum_{j \in L_k \setminus L_{k+1}} s^*,$$

which implies $|L_{k+1}| \cdot k \leq |L_k \setminus L_{k+1}| = |L_k| - |L_{k+1}|$. Thus, $|L_k| \geq (k+1) \cdot |L_{k+1}|$. This completes the proof of Theorem 20.7. □

We now prove a lower bound showing that the upper bound on the price of anarchy given in Theorem 20.7 is essentially tight.

**Theorem 20.9** *For every $m \in \mathbb{N}$, there exists an instance $G$ of the load balancing game with $m$ machines and $n \leq m$ tasks that has a Nash equilibrium assignment $A : [n] \to [m]$ with*

$$\mathit{cost}(A) = \Omega\left(\frac{\log m}{\log \log m}\right) \cdot \mathit{opt}(G).$$

**PROOF** Recall the definition of the gamma function from the proof of Theorem 20.7. We describe a game instance $G$ together with an equilibrium assignment $A$ satisfying

$$\mathrm{cost}(A) \geq \frac{1}{2} \cdot \left(\Gamma^{-1}(m) - 2 - o(1)\right) \cdot \mathrm{opt}(G),$$

which yields the theorem.

Our construction uses $q+1$ disjoint groups of machines denoted $G_0, \ldots, G_q$ with $q \approx \Gamma^{-1}(m)$. More, precisely, we set

$$q = \lfloor \Gamma^{-1}(m/3) - 1 \rfloor \geq \Gamma^{-1}(m) - 2 - o(1).$$

For $0 \leq k \leq q$, group $G_k$ consists of $q!/k!$ machines of speed $2^k$ each of which is assigned $k$ tasks of weight $2^k$. Let us remark that $0! = 1$. The total number of machines in these groups is thus

$$\sum_{k=0}^{q} |G_k| = q! \sum_{k=0}^{q} \frac{1}{k!} \leq 3\,\Gamma(q+1) \leq m$$

because $\sum_{k=0}^{q} \frac{1}{k!} \leq 3$ and $3\Gamma(q+1) \leq m$, which follows directly from the definition of $q$. As $m$ might be larger than the number of the machines in the groups, there might be some machines that do not belong to any of the groups. We assume

that these machines have the same parameters as the machines in group $G_0$; i.e., they have speed $2^0 = 1$ and $A$ does not assign a tasks to them.

We need to show that the described assignment is a Nash equilibrium. An agent with a task on a machine from group $G_k$ has cost $k$. It can neither reduce its cost by moving its task to a machine in group $G_j$ with $j \geq k$ as these machines have at least a load of $k$, nor can it reduce its cost by moving its task to a machine in group $G_j$ with $j < k$ as the load on such a machine, after the task moved to this machine, would be

$$j + \frac{2^k}{2^j} = j + 2^{k-j} \geq j + (k - j + 1) = k + 1$$

since $2^t \geq t + 1$, for every $t \geq 1$. Hence, none of the agents can unilaterally decrease its cost. In other words, $A$ is a Nash equilibrium.

The social cost of the equilibrium assignment $A$ is $q$. Next we show that $\text{opt}(G) \leq 2$ so that the theorem follows. We construct an assignment with load at most 2 on every machine. For each $k \in \{1, \ldots, q\}$, the tasks mapped by $A$ to the machines in group $G_k$ are now assigned to the machines in group $G_{k-1}$. Observe that the total number of tasks that $A$ maps to the machines in $G_k$ is

$$k \cdot |G_k| = k \cdot \frac{q!}{k!} = \frac{q!}{(k-1)!} = |G_{k-1}|.$$

Hence, we can assign the tasks in such a way that each machine in group $G_{k-1}$ receives exactly one of the tasks that $A$ mapped to a machine in group $G_k$. This task has a weight of $2^k$ and the speed of the machine is $2^{k-1}$. Hence, the load of each machine in this assignment is at most 2, which completes the proof. □

### 20.3.2 Algorithms for Computing Pure Equilibria

The proof of Proposition 20.3 reveals that, starting from any initial assignment, a pure Nash equilibrium is reached after a finite number of improvement steps. Theorem 20.6 shows that there exists a sequence of improvement steps of length $O(n)$ in case of identical machines and this sequence can be computed efficiently. However, in the case of uniformly related machines, it is not known whether there always exists a short sequence of improvement steps and whether such a sequence can be efficiently computed like in the case of identical machines. However, the well-known *LPT (largest processing time)* scheduling algorithm allows us to efficiently compute a Nash equilibrium. This algorithm inserts the tasks in a nonincreasing order of weights, assigning each task to a machine that minimizes the cost of the task at its insertion time.

**Theorem 20.10** *The LPT algorithm computes a pure Nash equilibrium for load balancing games on uniformly related machines.*

**PROOF** Let the tasks be numbered from 1 to $n$ in the order of their insertion. Let *time* $t \in \{0, \ldots, n\}$ denote the point of time after the first $t$ tasks have been inserted. We show by an induction that the partial assignment $A : [t] \to [m]$ computed by LPT at time $t$ is a Nash equilibrium. By our induction assumption

the tasks $1, \ldots, t-1$ are *satisfied* at time $t-1$; i.e., none of these tasks can improve its cost by a unilateral deviation. When task $t$ is inserted, it might be mapped to a machine $j^* \in [m]$ that holds already some other tasks. We only have to show that these tasks do not get unsatisfied because of the increased load on $j^*$ because of the assignment of task $t$. Let $i < t$ be one of the tasks mapped to machine $j^*$. For $j \in [m]$, let $\ell_j$ denote the load on machine $j$ at time $t$. Since the assignment of task $t$ to machine $j^*$ minimizes the cost of agent $t$ and as $w_t \leq w_i$,

$$\frac{\ell_{j^*}}{s_{j^*}} \leq \frac{\ell_j + w_t}{s_j} \leq \frac{\ell_j + w_i}{s_j},$$

for all $j \in [m]$. Hence, also at time $t$, agent $i$ is satisfied on machine $j^*$ as it cannot reduce its cost by moving from $j^*$ to another machine. □

The assignment computed by the LPT algorithm is not only a Nash equilibrium but it also approximates the optimal makespan within a ratio of at most $\frac{5}{3}$ for uniformly related machines and $\frac{4}{3} - \frac{1}{3m}$ for identical machines, see Friesen (1987) and Graham (1966), respectively. As makespan scheduling is NP-hard even on identical machines, one cannot hope for an efficient algorithm that computes an assignment with optimal makespan, unless P $\neq$ NP. However, the polynomial time approximation scheme of Hochbaum and Shmoys (1988) computes an assignment of tasks to uniformly related machines minimizing the makespan within a ratio of $(1+\epsilon)$, for any given $\epsilon > 0$. This assignment is not necessarily a Nash equilibrium. Feldmann et al. (2003a) present an efficient algorithm that transforms any given assignment into an equilibrium assignment without increasing the makespan. This approach is called *Nashification*. Combining the polynomial time approximation scheme with the Nashification approach yields a polynomial time algorithm that computes an equilibrium assignment for scheduling on uniformly related machines minimizing the makespan within a factor of $(1+\epsilon)$, for any given $\epsilon > 0$.

## 20.4 Mixed Equilibria on Identical Machines

The example with two identical machines presented in Section 20.1.2 shows that the social cost can increase if players make use of randomization. Let us now study this effect systematically. We analyze by how much the price of anarchy is increased when the set of strategies is extended from pure to mixed strategies. First, we consider an extreme case of randomization in which every agent randomizes over all strategies.

### 20.4.1 Fully Mixed Equilibria

The *support* of an agent is the set of strategies to which the agent assigns positive probability. In a *fully mixed strategy profile* all pure strategies are in the support of every agent. There is exactly one fully mixed strategy profile for load balancing games on identical machines i.e. a Nash equilibrium. In this *fully mixed Nash equilibrium* every player assigns every task with probability $\frac{1}{m}$ to each of the machines, i.e., $P = (p_i^j)$ with $p_i^j = \frac{1}{m}$, for every $i \in [n]$ and $j \in [m]$. The fully mixed Nash equilibrium maximizes

the randomization and, hence, seems to be a good candidate to study the effects of randomization.

Our analysis begins with a particularly simple class of load balancing games: Suppose that we have not only identical machines but also identical tasks. That is, we assume that there are $m$ machines of speed 1 and $n$ tasks of weight 1. In the unique fully mixed Nash equilibrium for such a game, each task is assigned to each machine with probability $\frac{1}{m}$. This strategy profile is a Nash equilibrium as the expected cost $c_i^j$ of any task $i$ on any machine $j$ is the same. In particular, Equation 20.1 yields

$$c_i^j = \mathbb{E}[\ell_j] + \left(1 - \frac{1}{m}\right) = 2 - \frac{1}{m}.$$

This setup corresponds to a well-studied balls-and-bins experiment from probability theory in which $n$ balls are assigned independently, uniformly at random to $m$ bins, which is also discussed in Chapter 17. How bad is such a fully mixed Nash equilibrium in comparison to an optimal assignment that distributes the tasks evenly among the machines? An optimal assignment minimizes the makespan, and the optimal makespan is obviously $\lceil \frac{n}{m} \rceil$. The expected makespan of the fully mixed strategy profile corresponds to the expected *maximum occupancy* of the corresponding balls-and-bins experiment, i.e., the expected number of balls in the fullest bin. The following proposition yields a simple formula for this quantity that is exact up to constant factors for any choice of $m$ and $n$.

**Proposition 20.11** *Suppose that $n \geq 1$ balls are placed independently, uniformly at random into $m \geq 1$ bins. Then the expected maximum occupancy is*

$$\Theta\left(\frac{\ln m}{\ln\left(1 + \frac{m}{n}\ln m\right)}\right).$$

Let us illustrate the formula for the expected maximum occupancy given in the proposition with a few examples. If $n \geq m \log m$, then the expected maximum occupancy is $\Theta(\frac{n}{m})$ as, in this case, $\ln\left(1 + \frac{m}{n}\ln m\right) = \Theta\left(\frac{m}{n}\ln m\right)$. If $n \leq m^{1-\epsilon}$, for any fixed $\epsilon > 0$, then the expected maximum occupancy is $\Theta(1)$. Observe, in both of these cases, the ratio between the expected makespan for the fully mixed equilibrium and the makespan of an optimal assignment is $O(1)$. It turns out that this ratio is maximized when setting $m = n$. In this case, the expected maximum occupancy is $\Theta(\log m / \log\log m)$ while the optimal makespan is 1. This yields the following result.

**Theorem 20.12** *For every $m \in \mathbb{N}$, there exists an instance $G$ of a load balancing game with $m$ identical machines and $n = m$ tasks that has a Nash equilibrium strategy profile $P$ with*

$$cost(P) = \Omega\left(\frac{\log m}{\log\log m}\right) \cdot opt(G).$$

As the fully mixed Nash equilibrium is the equilibrium that maximizes the randomization, one could guess that this is also the equilibrium that maximizes the ratio between the expected makespan and the optimal makespan for load balancing games. This guess is known as the so-called *fully mixed Nash equilibrium conjecture*. This conjecture is appealing as it would yield a simple characterization of the worst-case Nash equilibrium for load balancing games. Unfortunately, however, the conjecture is wrong. With the help of Proposition 20.11, we can easily construct a counterexample. Let $m = 2^{2k}$, for some $k \in \mathbb{N}$. This way, $\sqrt{m}$ as well as $\log m$ are integers. Now consider the following instance of the load balancing game on $m$ identical machines. Suppose that there are $\sqrt{m}$ *large tasks* of weight 1, and $(m - \sqrt{m}) \cdot \log m$ *small tasks* of weight $\frac{1}{\log m}$. The balls-and-bins analysis above shows that the maximum number of large tasks that are assigned to the same machine by a fully mixed Nash equilibrium is $O(1)$, and the maximum number of small tasks assigned to the same machine is $O(\log m)$. Hence, the expected makespan of the fully mixed Nash equilibrium is $O(1)$. Now consider the following strategy profile: Assign the large tasks uniformly at random to the first $\sqrt{m}$ machines (called group $A$) and the small tasks uniformly at random to the other machines (called group $B$). This profile is a Nash equilibrium as Equation 20.1 yields that, for a large task, the expected cost on a machine of group $A$ is less than the expected cost on a machine of group $B$ and, for a small task, the expected cost on a machine of group $B$ is less than the expected cost on a machine of group $A$. In this equilibrium, the expected maximum occupancy among the large tasks is $\Theta(\frac{\log m}{\log \log m})$, which shows that there is a mixed Nash equilibrium whose expected makespan is larger than the expected makespan of the fully mixed Nash equilibrium by a factor of $\Omega(\frac{\log m}{\log \log m})$.

### 20.4.2 Price of Anarchy

The fully mixed Nash equilibrium is not necessarily the worst-case Nash equilibrium for every instance of the load balancing game on identical machines. Nevertheless, the following analysis shows that the lower bound on the price of anarchy that we obtained from studying this kind of equilibria is tight.

**Theorem 20.13** *Consider an instance $G$ of the load balancing game with $n$ tasks of weight $w_1, \ldots, w_n$ and $m$ identical machines. Let $P = (p_i^j)_{i \in [n], j \in [m]}$ denote any Nash equilibrium strategy profile. Then, it holds that*

$$cost(P) = \mathcal{O}\left(\frac{\log m}{\log \log m}\right) \cdot opt(G).$$

**PROOF** Without loss of generality, we assume that all machines have speed 1. Recall that $\text{cost}(P) = \mathbb{E}[\max_{j \in [m]}(\ell_j)]$, i.e., $\text{cost}(P)$ corresponds to the expected maximum load over all machines or, in other words, the expected makespan. Our analysis starts with proving an upper bound on the maximum expected load instead of the expected maximum load.

We claim that, for every $j \in [m]$, $\mathbb{E}[\ell_j] \leq (2 - \frac{2}{m+1})\,\text{opt}(G)$. The proof for this claim follows the course of the analysis for the upper bound on the price of anarchy for pure equilibria. More specifically, the proof of Theorem 20.5 can be

adapted as follows to mixed equilibria: Instead of considering a smallest weight task $i^*$ placed on a maximum load machine $j^*$, one defines $i^*$ to be the smallest weight task with positive probability on a machine $j^*$ maximizing the expected load. Also in all other occurrences one considers the expected load instead of the load.

We conclude that the maximum expected load is less than $2\,\mathrm{opt}(G)$. Next we show that the expected maximum load deviates at most by a factor of $\mathcal{O}(\frac{\log m}{\log\log m})$ from the maximum expected load. We use a weighted Chernoff bound in order to show that it is unlikely that there is a machine that deviates by a large factor from its expectation.

**Lemma 20.14 (weighted Chernoff bound)** *Let $X_1, \ldots, X_N$ be independent random variables with values in the interval $[0, z]$ for some $z > 0$, and let $X = \sum_{i=1}^{N} X_i$, then for any $t$ it holds that $\mathbb{P}[\sum_{i=1}^{N} X_i \geq t] \leq (e \cdot \mathbb{E}[X] \,/\, t)^{t/z}$.*

A description how to derive this and other variants of the Chernoff bound can be found, e.g., in Mitzenmacher and Upfal (2005).

Fix $j \in [m]$. Let $w$ denote the largest weight of any task. Applying the weighted Chernoff bound shows that, for every $t$,

$$\mathbb{P}[\ell_j \geq t] \leq \min\left\{1, \left(\frac{e \cdot \mathbb{E}[\ell_j]}{t}\right)^{t/w}\right\} \leq \left(\frac{2\,e\,\mathrm{opt}(G)}{t}\right)^{t/\mathrm{opt}(G)}.$$

because $\mathbb{E}[\ell_j] \leq 2\,\mathrm{opt}(G)$ and $w \leq \mathrm{opt}(G)$. Now let $\tau = 2\,\mathrm{opt}(G)\,\frac{\ln m}{\ln\ln m}$. Then, for any $x \geq 0$,

$$\begin{aligned}\mathbb{P}[\ell_j \geq \tau + x] &\leq \left(\frac{e \ln\ln m}{\ln m}\right)^{2\ln m/\ln\ln m + x/\mathrm{opt}(G)} \\ &\leq \left(\frac{1}{\sqrt{\ln m}}\right)^{2\ln m/\ln\ln m} \cdot e^{-x/\mathrm{opt}(G)} \\ &= m^{-1} \cdot e^{-x/\mathrm{opt}(G)},\end{aligned}$$

where the second inequality holds asymptotically as, for sufficiently large $m$, $\frac{\ln m}{e\ln\ln m} \geq \sqrt{\log m}$ and $\frac{\ln m}{e\ln\ln m} \geq e$.

Now with the help of the tail bound we can upper-bound $\mathrm{cost}(P)$ as follows. For every nonnegative random variable $X$, $\mathbb{E}[X] = \int_0^\infty \mathbb{P}[X \geq t]dt$. Consequently,

$$\mathrm{cost}(P) \;=\; \mathbb{E}\Big[\max_{j\in[m]} \ell_j\Big] \;=\; \int_0^\infty \mathbb{P}\Big[\max_{j\in[m]} \ell_j \geq t\Big]dt\,.$$

Substituting $t$ by $\tau + x$ and then applying the union bound yields

$$\mathrm{cost}(P) \;\leq\; \tau + \int_0^\infty \mathbb{P}\Big[\max_{j\in[m]} \ell_j \geq \tau + x\Big]\,dx \;\leq\; \tau + \int_0^\infty \sum_{j\in[m]} \mathbb{P}[\ell_j \geq \tau + x]\,dx\,.$$

Finally, we apply the tail bound derived above and obtain

$$\text{cost}(P) \;\le\; \tau + \int_0^\infty e^{-x/\text{opt}(G)} dx \;=\; \tau + \text{opt}(G)\,,$$

which yields the theorem as $\tau = 2\,\text{opt}(G)\,\frac{\ln m}{\ln \ln m}$. □

## 20.5 Mixed Equilibria on Uniformly Related Machines

Finally, we come to the most general case, namely mixed equilibria on uniformly related machines. The following theorem shows that the price of anarchy for this case is only slightly larger than the one for mixed equilibria on identical machines or pure equilibria on uniformly related machines. The analysis combines the methods from both of these more restricted cases: First, we show that the maximum expected makespan is bounded by

$$\mathcal{O}\left(\frac{\log m}{\log \log m}\right) \cdot \text{opt}(G)$$

using the same kind of arguments as in the analysis of the price of anarchy for pure equilibria on uniformly related machines. Then, as in the case of mixed equilibria on identical machines, we use a Chernoff bound to show that the expected maximum load is not much larger than the maximum expected load. In fact, this last step loses only a factor of order $\log \log m / \log \log \log m$, which results in an upper bound on the price of anarchy of

$$\mathcal{O}\left(\frac{\log m}{\log \log \log m}\right).$$

After proving this upper bound, we present a corresponding lower bound by adding some randomization to the lower bound construction for pure equilibria on uniformly related machines, which increases also the lower bound by a factor of order $\log \log m / \log \log \log m$ and, hence, yields a tight result about the price of anarchy.

**Theorem 20.15** *Consider an instance $G$ of the load balancing game with $n$ tasks of weight $w_1, \ldots, w_n$ and $m$ machines of speed $s_1, \ldots, s_m$. Let $P$ be any Nash equilibrium strategy profile. Then, it holds that*

$$\textit{cost}(P) = \mathcal{O}\left(\frac{\log m}{\log \log \log m}\right) \cdot \textit{opt}(G).$$

**PROOF** As in the case of identical machines, our analysis starts with proving an upper bound on the maximum expected load instead of the expected maximum load. To simplify the notation, we assume $\text{opt}(G) = 1$, which can be achieved by scaling the weights appropriately. Let $c = \left\lfloor \max_{j \in [m]} \left( \mathbb{E}[\ell_j] \right) \right\rfloor$. We first prove an upper bound on $c$ following the analysis for pure Nash equilibria in Theorem 20.7. Without loss of generality, assume $s_1 \ge s_2 \ge \cdots \ge s_m$. Let $L = [1, 2, \ldots, m]$ denote the list of machines in non increasing order of speed. For $k \in \{0, \ldots, c-1\}$, let $L_k$ denote the maximum length prefix of $L$ such that the expected load

of each server in $L_k$ is at least $k$. Analogously to the analysis in the proof of Theorem 20.7, one shows the recurrence $|L_k| \geq (k+1) \cdot |L_{k+1}|$, for $0 \leq k \leq c-2$, and $|L_{c-1}| \geq 1$. Solving the recurrence yields $|L_0| \geq (c-1)! = \Gamma(c)$. Thus, $|L_0| = m$ implies $c \leq \Gamma^{-1}(m) = \Theta(\ln m / \ln\ln m)$. Now let

$$C = \max\left\{c+1, \frac{\ln m}{\ln\ln m}\right\} = \Theta\left(\frac{\ln m}{\ln\ln m}\right).$$

In the rest of the proof, we show that the expected makespan of the equilibrium assignment can exceed $C$ at most by a factor of order $\ln\ln m / \ln\ln\ln m$ so that the expected makespan is $\mathcal{O}(\ln m / \ln\ln\ln m)$, which proves the theorem as we assume $\text{opt}(G) = 1$.

As the next step, we prove a tail bound on $\ell_j$, for any fixed $j \in [m]$ and, afterward, we use this tail bound to derive an upper bound on the expected makespan. For a machine $j \in [m]$, let $T_j^{(1)}$ denote the set of tasks $i$ with $p_i^j \geq \frac{1}{4}$ and $T_j^{(2)}$ the set of tasks $i$ with $p_i^j \in (0, \frac{1}{4})$. Let $\ell_j^{(1)}$ and $\ell_j^{(2)}$ denote random variables that describe the load on link $j$ only taking into account the tasks in $T_j^{(1)}$ and $T_j^{(2)}$, respectively. Observe that $\ell_j = \ell_j^{(1)} + \ell_j^{(2)}$. For the tasks in $T_j^{(1)}$, we immediately obtain

$$\ell_j^{(1)} \leq \sum_{i \in T_j^{(1)}} \frac{w_i}{s_j} \leq 4 \sum_{i \in T_j^{(1)}} \frac{w_i\, p_i^j}{s_j} = 4\,\mathbb{E}[\ell_j^{(1)}] \leq 4C. \tag{20.2}$$

To prove an upper bound on $\ell_j^{(2)}$, we use the weighted Chernoff bound from Lemma 20.14. This bound requires an upper bound on the maximum weight. As a first step to bound the weights, we prove a result about the relationship between the speeds of the machines in the different groups that are defined by the prefixes. For $0 \leq k \leq c-2$, let $G_k = L_k \setminus L_{k+1}$, and let $G_{c-1} = L_{c-1}$. For $0 \leq k \leq c-1$, let $s(k)$ denote the speed of the fastest machine in $G_k$. Clearly, $s(c-1) \geq s(c-2) \geq \cdots \geq s(1) \geq s(0)$. We claim that this sequence is, in fact, geometrically decreasing.

**Lemma 20.16** *For $0 \leq k \leq c-4$, $s(k+2) \geq 2\,s(k)$.*

**PROOF** To prove the claim, we first observe that there exists a task $j^*$ with $w_{j^*} \leq s(k+2)$ that has positive probability on a machine in $L_{k+3}$. This is because an optimal assignment strategy has to move some of the expected load from the machines in $L_{k+3}$ to machines in $L \setminus L_{k+3}$ and it can only assign those tasks to machines in $L \setminus L_{k+3}$ whose weights are not larger than the maximum speed among this set of machines, which is $s(k+2)$. Now suppose $s(k) > \frac{1}{2}s(k+2)$. The expected load of the fastest machine in $G_k = L_k \setminus L_{k+1}$ is at most $k+1$. Thus the expected cost of $j^*$ on the fastest machine in $G_k$ is at most

$$k+1+\frac{w_{j^*}}{s(k)} < k+1+\frac{2w_{j^*}}{s(k+2)} \leq k+3.$$

This contradicts that the expected cost of $j^*$ in the considered Nash equilibrium is at least $k+3$ as it has positive probability on a machine in $L_{k+3}$. Thus, Lemma 20.16 is shown. □

Now we apply Lemma 20.16 to prove an upper bound on the weights of the tasks in the set $T_j^{(2)}$.

**Lemma 20.17** *For every $j \in [m]$ and $i \in T_j^{(2)}$, $w_i \leq 12\, s_j$.*

**PROOF** Let $i$ be a task from $T_j^{(2)}$, i.e., $p_i^j \in (0, \frac{1}{4})$. Let $j \in G_k$, for $0 \leq k \leq c-1$. The expected cost of $i$ on $j$ is

$$c_i^j = \mathbb{E}[\ell_j] + \left(1 - p_i^j\right) \frac{w_i}{s_j} \geq k + \frac{3w_i}{4s_j}.$$

Suppose that $k \geq c-3$. In this case, $w_i > 12\, s_j$ implies $c_i^j > k + \frac{3}{4} \cdot 12 \geq c+6$, which contradicts that, under the Nash equilibrium profile, the expected cost of any task on the fastest machine is at most $c+1$. Hence, the lemma is shown for $k \geq c-3$. Now suppose $k \leq c-4$. Let $q$ denote the fastest machine from $G_{k+2}$. Lemma 20.16 yields $s_q = s(k+2) \geq 2s(k) \geq 2\, s_j$. Hence, the expected cost of $i$ on $q$ is

$$c_i^q = \mathbb{E}[\ell_q] + \left(1 - p_i^q\right) \frac{w_i}{s_q} \leq k + 3 + \frac{w_i}{2s_j}.$$

As $p_i^j > 0$, the Nash equilibrium condition yields $c_i^j \leq c_i^q$. Consequently,

$$k + \frac{3w_i}{4s_j} \leq k + 3 + \frac{w_i}{2s_j},$$

which implies $w_i \leq 12\, s_j$ and, hence, completes the proof of Lemma 20.17. □

Let $z = \max_{i \in T_j^{(2)}}(w_i/s_j)$. Lemma 20.17 implies $z \leq 12$. Now applying the weighted Chernoff bound from Lemma 20.14 yields that, for every $\alpha > 0$,

$$\mathbb{P}[\ell_j^{(2)} \geq \alpha C] \leq \left(\frac{e \cdot \mathbb{E}[\ell_j^{(2)}]}{\alpha C}\right)^{\alpha C/z} \leq \left(\frac{e}{\alpha}\right)^{\alpha C/12}$$

since $\mathbb{E}[\ell_j^{(2)}] \leq C$. We define $\tau = 24\, C \ln\ln m / \ln\ln\ln m$. As $C$ is of order $\ln m / \ln\ln m$, it follows that $\tau$ is of order $\ln m / \ln\ln\ln m$. Let $x \geq 0$. We substitute $\tau + x$ for $\alpha C$ and obtain

$$\begin{aligned}\mathbb{P}[\ell_j^{(2)} \geq \tau + x] &\leq \left(\frac{eC}{\tau + x}\right)^{(\tau+x)/12} \\ &\leq \left(\frac{e \ln\ln\ln m}{24 \ln\ln m}\right)^{2C \ln\ln m/\ln\ln\ln m + x/12}.\end{aligned}$$

Observe that $24 \ln\ln m/(e \ln\ln\ln m)$ is lower-bounded by $\sqrt{\ln\ln m}$ and also lower-bounded by $e^2$. Furthermore, $C \geq \ln m / \ln\ln m$. Applying these bounds yields

$$\mathbb{P}[\ell_j^{(2)} \geq \tau + x] \leq \left(\frac{1}{\sqrt{\ln\ln m}}\right)^{2\ln m/\ln\ln\ln m} \cdot e^{-x/6} = m^{-1} \cdot e^{-x/6}.$$

As a consequence,

$$\begin{aligned}\mathbb{E}\Big[\max_{j\in[m]} \ell_j^{(2)}\Big] &= \int_0^\infty \mathbb{P}\Big[\max_{j\in[m]} \ell_j^{(2)} \geq t\Big] dt \\ &\leq \tau + \int_0^\infty \mathbb{P}\Big[\max_{j\in[m]} \ell_j^{(2)} \geq \tau + x\Big] dx \\ &\leq \tau + \int_0^\infty \sum_{j\in[m]} \mathbb{P}[\ell_j^{(2)} \geq \tau + x] dx.\end{aligned}$$

Now applying our tail bound yields

$$\mathbb{E}\Big[\max_{j\in[m]} \ell_j^{(2)}\Big] \leq \tau + \int_0^\infty e^{-x/6} dx = \tau + 6. \tag{20.3}$$

Finally, we combine Equations 20.2 and 20.3 and obtain

$$\text{cost}(P) = \mathbb{E}\Big[\max_{j\in[m]} \ell_j\Big] \leq 4C + \tau + 6 = \mathcal{O}\Big(\frac{\log m}{\log\log\log m}\Big),$$

which completes the proof of Theorem 20.15. □

Next we show that the upper bound given in Theorem 20.15 is tight by showing that for every number of machines there exists a game instance that matches the upper bound up to a constant factor.

**Theorem 20.18** *For every $m \in \mathbb{N}$, there exists an instance $G$ of the load balancing game with $m$ machines and $n \leq m$ tasks that has a Nash equilibrium strategy profile $P$ with*

$$\mathit{cost}(P) = \Omega\left(\frac{\log m}{\log\log\log m}\right) \cdot \mathit{opt}(G).$$

**PROOF** The starting point for our construction is the game and the Nash assignment $A$ from the proof of Theorem 20.9. We use mixed strategies in only one of the groups, namely in the group $G_k$ with $k = \lceil q/2 \rceil$. Let $M$ denote the number of machines in this group, i.e., $M = q!/k! \geq (q/2)^{\lfloor q/2 \rfloor}$. Observe that $\log M = \Theta(q \log q) = \Theta(\log m)$.

Let $T$ denote the set of tasks mapped by $A$ to one of the machines in $G_k$. The tasks in $T$ have weight $2^k$. Each of these tasks is now assigned uniformly at random to a machine group $G_k$, i.e., $p_i^j = \frac{1}{M}$, for each $j \in G_k$ and each $i \in T$. For all other tasks the strategy profile $P$ corresponds without any change to the pure strategy profile of assignment $A$. Observe that the randomization increases the expected cost of the tasks. The expected cost of a task $i \in T$ on a machine

$j \in G_k$ is now

$$c_i^j = \mathbb{E}[\ell_j] + \left(1 - p_i^j\right) \frac{w_i}{s_j} = k + \left(1 - \frac{1}{M}\right) < k + 1.$$

In the proof of Theorem 20.9, we have shown that the cost of a task $i$ of weight $2^k$ on a machine of group $G_j$ with $j \neq k$ is at least $k + 1$. Thus, the strategy profile $P$ is a Nash equilibrium.

It remains to compare the social cost of the equilibrium profile $P$ with the optimal cost. The structure of the optimal assignment is not affected by the modifications. It has social cost $\mathrm{opt}(G) = 2$. Now we give a lower bound for the social cost of $P$. This social cost is, obviously, bounded from below by the maximum number of tasks that are mapped to the same machine in the group $G_k$. Applying Proposition 20.11 with $M$ bins and $N = kM$ balls shows that the expected makespan is

$$\Omega\left(\frac{\ln M}{\ln\left(1 + \frac{1}{k}\ln M\right)}\right) = \Omega\left(\frac{\log m}{\log\log\log m}\right),$$

where the last estimate holds as $k = \Theta(\log m / \log\log m)$ and $\log M = \Theta(\log m)$. This completes the proof of Theorem 20.18. □

## 20.6 Summary and Discussion

In this chapter, we studied the price of anarchy in load balancing games in four different variants. Table 20.1 summarizes the results about the price of anarchy that we have presented. In the case of pure equilibria on identical machines, the price of anarchy is bounded from above by a small constant term. In all other cases, the price of anarchy is bounded from above by a slowly growing, sublogarithmic function in the number of machines. One might interpret these results as a first game theoretic explanation why the resources in a large distributed system like the Internet that widely lacks global control are shared in a more or less efficient and fair way among different users with different interests, although the considered model is clearly oversimplifying in several aspects.

It is an interesting coincidence that both the price of anarchy for pure equilibria on uniformly related machines as well as the price of anarchy for mixed equilibria

**Table 20.1.** *The price of anarchy for pure and mixed equilibria in load balancing games on identical and uniformly related machines*

| | Identical | Uniformly related |
|---|---|---|
| Pure | $2 - \frac{2}{m+1}$ | $\Theta\left(\frac{\log m}{\log\log m}\right)$ |
| Mixed | $\Theta\left(\frac{\log m}{\log\log m}\right)$ | $\Theta\left(\frac{\log m}{\log\log\log m}\right)$ |

on identical machines are of order $\log m / \log \log m$. Although both of these models result in essentially the same price of anarchy, the reasons for the increase in the social cost are quite different: In the case of pure equilibria on uniformly related machines, equilibrium assignments correspond to local optima with respect to moves of single tasks. That is, tasks are placed in a suboptimal but nevertheless coordinated fashion. On the contrary, in case of mixed equilibria, the increase in cost is due to collisions between uncoordinated random decisions. If one combines these two effects, then one loses only another very small factor of order $\log \log m / \log \log \log m$, which results in a price of anarchy of order $\log m / \log \log \log m$ for mixed equilibria on uniformly related machines.

Obviously, the price of anarchy for load balancing games as we have defined them in the beginning of this chapter is well understood. As mentioned above, however, this model is very simplistic. To make these results more realistic, one needs to incorporate other aspects from practical application areas like, e.g., more realistic cost functions or other ways to define the social cost. We give pointers to studies of quite a few variants of load balancing games in the bibliographic notes. In Christodoulou et al. (2004), it is made an interesting attempt that adds an algorithmic or constructive element to the analysis of the price of anarchy. The idea behind so-called "coordination mechanisms" is not to study the price of anarchy for a fixed system, but to design the system in such a way that the increase in cost or the loss in performance due to selfish behavior is as small as possible. Similar aspects are also discussed in Chapter 17. We believe that this is a promising direction of research that might result in practical guidelines of how to build a distributed system that does not suffer from selfish behavior but might even exploit the selfishness of the agents.

Besides the price of anarchy, we have studied the question of how agents reach a Nash equilibrium. We have observed that any sequence of improvement steps reaches a pure Nash equilibrium after a finite number of steps. In case of identical machines the max-weight best-response policy reaches an equilibrium in only $O(n)$. In case of uniformly related machines, it is open whether there exists a short sequence of improvement steps that lead from any given assignment to a pure Nash equilibrium. We think that this question is of great importance as Nash equilibria are only of interest if agents can reach them quickly. It is not clear that the only reasonable approach for the agents to reach a Nash equilibrium in a distributed way is to use improvement steps. There might also be other, possibly more strategic or more coordinated behavioral rules that quickly converge to a Nash equilibrium or to an approximate Nash equilibrium. For example, Chapter 29 considers some approaches from evolutionary game theory in the context of routing in networks. It is an interesting research problem to design distributed protocols that ensure that agents reach a Nash equilibrium quickly. Pointers to first results toward this direction can be found in the bibliographic notes.

## 20.7 Bibliographic Notes

The concept of the price of anarchy was introduced by Koutsoupias and Papadimitriou (1999). In their seminal work, they study load balancing in form of a routing game

consisting of two nodes connected by parallel edges with possibly different speeds. Each agent has an amount of traffic that the agent seeks to map to one of the edges such that the load on this edge is as small as possible. In our notation, the parallel edges between the source and the sink correspond to the machines and the pieces of traffic of the agents correspond to the tasks. Let us remark that originally the ratio between the social cost in a worst-case Nash equilibrium and the optimal social cost was called *coordination ratio* but in this chapter we switched to the now commonly used term *price of anarchy*. The game theoretic model underlying the load balancing games is also known as *KP model*.

The results presented in Table 20.1 have been obtained in the following studies. The upper bound of $2 - \frac{2}{m+1}$ on the price of anarchy for pure equilibria in load balancing games with identical machines goes back to the scheduling literature (Finn and Horowitz, 1979), where the same ratio occurs in form of an approximation factor for a local search optimization heuristic. The lower bound on the price of anarchy for mixed equilibria on identical machines is presented in Koutsoupias and Papadimitriou (1999). The analysis for the corresponding upper bound is obtained in Czumaj and Vöcking (2002) and Koutsoupias et al. (2003). Let us remark that the analysis in Czumaj and Vöcking (2002) is tight up to a constant additive term. It shows that the price of anarchy for mixed equilibria in load balancing games on identical machines is $\Gamma^{-1}(m) \pm \Theta(1)$. The upper and lower bounds on the price of anarchy for pure and mixed equilibria in load balancing games with uniformly related machines are from Czumaj and Vöcking (2002) as well. This work also contains a tight characterization of the price of anarchy as a function of the ratio between the speeds of the fastest and the slowest machine.

The existence proof for pure equilibria presented in Section 20.1.1 can be found in Fotakis et al. (2002) and Even-Dar et al. (2003). The result from Section 20.3.2 that the LPT algorithm computes a pure Nash equilibrium is presented in Fotakis et al. (2002) together with several further results about the complexity of computing pure and mixed equilibria in load balancing games. The uniqueness of the fully mixed Nash equilibrium is shown in Mavronicolas and Spirakis (2001). Exercise 20.5 reworks the nice proof for this result. The counterexample to the *fully mixed Nash equilibrium conjecture* presented in Section 20.4.1 is from Fischer and Vöcking (2005). Finally, the results from Section 20.2.2 about the convergence of best response sequences are from Even-Dar et al. (2003).

Let us remark that this chapter does by far not give a complete overview of the rich literature about different variants of games for load balancing or routing on parallel links. We conclude this chapter with a few pointers to further literature. Load balancing games with more general cost functions are considered, e.g., in Caragiannis et al. (2006), Czumaj et al. (2002), Libman and Orda (1999, 2001). Other definitions of the social cost are considered, e.g., in Caragiannis et al. (2006), Gairing et al. (2004a, 2004b) and Suri et al. (2004). Another interesting variant of load balancing games assumes that agents come with subsets of the machines on which they have to place their tasks. The price of anarchy in such a restricted assignment model is investigated in Awerbuch et al. (2003), Gairing et al. (2006), and Suri et al. (2004). The price of anarchy with respect to equilibria that are robust against coalitions is studied in Andelman et al. (2007). An important aspect that we have only touched in this chapter is the complexity of

computing Nash equilibria for load balancing games. Further work dealing with the computation of Nash equilibria can be found, e.g., in Even-Dar et al. (2003), Feldmann et al. (2003a), Fotakis et al. (2002), Fischer and Vöcking (2005), and Gairing et al. (2004a). Recent work deals also with the convergence time of distributed load balancing processes in which agents make parallel attempts for improvement steps until they find a Nash equilibrium (Berenbrink et al., 2006; Even-Dar and Mansour, 2005). Another interesting topic is load balancing games with incomplete information that have been considered, e.g., in Beier et al. (2004) and Gairing et al. (2005). Finally, let us remark that the concept of coordination mechanisms has been suggested in Christodoulou et al. (2004) and some further results on this topic can be found in Immorlica et al. (2005).

Several other results for load balancing and routing on parallel links have been collected in the surveys (Czumaj, 2004; Feldmann et al., 2003b; Koutsoupias, 2003).

## Bibliography

N. Andelman, M. Feldman, and Y. Mansour. Strong price of anarchy. In *Proc. 18th Annual ACM-SIAM Symp. on Discrete Algorithms*, 2007.

B. Awerbuch, Y. Azar, Y. Richter, and D. Tsur. Tradeoffs in worst-case equilibria. In *Proc. 1st International Workshop on Approximation and Online Algorithms (WAOA)*, pp. 41–52, 2003.

R. Beier, A. Czumaj, P. Krysta, and B.Vöcking. Computing equilibria for congestion games with (im)perfect information. In *Proc. 15th Annual ACM-SIAM Symp. Discrete Algorithms*, pp. 746–755, 2004.

P. Berenbrink, T. Friedetzky, L.A. Goldberg, P.W. Goldberg, Z. Hu, and R.A. Martin. Distributed selfish load balancing. In *Proc. 17th Annual ACM-SIAM Symp. Discrete Algorithms*, pp. 354–363, 2006.

I. Caragiannis, M. Flammini, C. Kaklamanis, P. Kanellopoulos, and L. Moscardelli. Tight bounds for selfish and greedy load balancing. In *Proc. 33rd Intl. Colloq. on Automata, Languages, and Programming*, pp. 311–322, 2006.

G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination Mechanisms. In *Proc. 31st Intl. Colloq. on Automata, Languages and Programming*, pp. 345–357, 2004.

A. Czumaj. Selfish Routing on the Internet. Chapter 42 in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, edited by J. Leung, CRC Press, Boca Raton, FL, 2004.

A. Czumaj, P. Krysta, and B. Vöcking. Selfish traffic allocation for server farms. In *Proc. 34th Annual ACM Symp. Theory of Computing*, pp. 287–296, 2002.

A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *Proc. 13th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 413–420, 2002.

E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to Nash equilibria. In *Proc. 30th International Colloq. on Automata, Languages and Programming*, pp. 502–513, 2003.

E. Even-Dar and Y. Mansour. Fast convergence of selfish rerouting. In *Proc. 16th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 772–781, 2005.

R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. In *Proc. 30th International Colloq. on Automata, Languages and Programming*, pp. 414–426, 2003a.

R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Selfish routing in non-cooperative networks: a survey. In *Proc. 28th International Symp. on Mathematical Foundations of Computer Science*, pp. 21–45, 2003b.

G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT,* 19(3):312–320, 1979.

S. Fischer and B. Vöcking. On the structure and complexity of worst-case equilibria. In *Proc. 1st Workshop on Internet and Network Economics*, pp. 151–160, 2005.

D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *Proc. 29th Intl. Colloquium on Automata, Languages and Programming (ICALP)*, pp. 123–134, 2002.

D.K. Friesen. Tighter bounds for LPT scheduling on uniform processors. *SIAM J. Computing*, 16(3):554–560, 1987.

M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien. Computing Nash equilibria for scheduling on restricted parallel links. In *Proc. 36th Annual ACM Symp. on Theory of Computing*, pp. 613–622, 2004a.

M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien. The price of anarchy for polynomial social cost. In *Proc. 29th Intl. Symp. on Mathematical Foundations of Computer Science*, pp. 574–585, 2004b.

M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and M. Rode. Nash equilibria in discrete routing games with convex latency functions. In *Proc. 31st Intl. Colloq. on Automata, Languages and Programming*, pp. 645–657, 2004c.

M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien. The Price of Anarchy for Restricted Parallel Links. *Parallel Process. Lett.*, 16(1):117–132, 2006.

M. Gairing, B. Monien, and K. Tiemann. Selfish routing with incomplete information. In *Proc. 17th Annual ACM Symp. on Parallel Algorithms*, pp. 203–212, 2005.

R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Tech. J.*, 45: 1563–1581, 1966.

R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17: 263–269, 1969.

D.S. Hochbaum and D.B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors. *SIAM J. Computing*, 17(3):539–551, 1988.

N. Immorlica, L. Li, V.S. Mirrokni, and A. Schulz. Coordination mechanisms for selfish scheduling. In *Proc. 1st Workshop on Internet and Network Economics*, pp. 55–69, 2005.

E. Koutsoupias. Selfish task allocation. *Bulletin of the EATCS (81)*, pp. 79–88, 2003.

E. Koutsoupias, M. Mavronicolas, and P. Spirakis. Approximate equilibria and ball fusion. *Theory of Computing Systems*, 36(6):683–693, 2003.

E. Koutsoupias and C.H. Papadimitriou. Worst-case equilibria. In *Proc. 16th Annual Symp. on Theoretical Aspects of Computer Science*, pp. 404–413, 1999.

L. Libman and A. Orda. The designer's perspective to atomic noncooperative networks. *IEEE/ACM Trans. Networking*, 7(6):875–884, 1999.

L. Libman and A. Orda. Atomic resource sharing in noncooperative networks. *Telecommun. Systems*, 17(4):385–409, 2001.

M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proc. 33rd ACM Symp. on Theory of Computing*, pp. 510–519, 2001.

M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

S. Suri, C. Toth, and Y. Zhou. Selfish load balancing and atomic congestion games. In *Proc. 16th Annual ACM Symp. on Parallel Algorithms and Architectures*, pp. 188–195, 2005.

## Exercises

**20.1** Let $G$ be any instance of the load balancing game with three tasks that should be placed on two identical machines. Show that any pure Nash equilibrium for $G$ is optimal, i.e., $\text{cost}(A) = \text{opt}(G)$ for any equilibrium assignment $A$.
*Remark:* Interestingly, the example presented in Section 20.1.2 that yields the worst-case price of anarchy for two identical machines uses only four tasks.

**20.2** Show, for every $m \in \mathbb{N}$, there exists an instance $G$ of the load balancing game with $m$ identical machines and $2m$ tasks that has a Nash equilibrium assignment $A : [n] \to [m]$ with

$$\text{cost}(A) = \left(2 - \frac{2}{m+1}\right) \cdot \text{opt}(G).$$

*Hint:* Generalize the example with two machines given in Section 20.1.2.

**20.3** Prove that the price of anarchy for pure equilibria on instances of the load balancing game with two tasks and two machines with possibly different speeds corresponds to the golden ratio $\phi = \frac{1}{2}(1 + \sqrt{5})$. That is, show that

**a)** there is a game instance $G$ admitting an equilibrium assignment $A$ with $\text{cost}(A) = \phi \cdot \text{opt}(G)$.
**b)** for every game instance $G$ and every equilibrium assignment $A$ for this instance, it holds $\text{cost}(A) \leq \phi \cdot \text{opt}(G)$.

**20.4** Consider an instance of the load balancing game with two tasks both of which have weight 1 and two machines, one of speed 1 and the other of speed $s > 0$.

**(a)** Show that there does not exist a fully mixed Nash equilibrium if $s \leq \frac{1}{2}$ or $s \geq 2$.
**(b)** Show that there exists a unique fully mixed Nash equilibrium if $\frac{1}{2} < s < 2$. Describe the strategy profile of this equilibrium as a function of $s$.

**20.5** Show that there exists at most one fully mixed Nash equilibrium for every instance of the load balancing game.
*Hint:* Describe the conditions on the probabilities $p_i^j$ imposed by a fully mixed Nash equilibrium in form of a system of linear equations and show that this system has a unique solution. If all the values for the variables $p_i^j$ in this solution are positive then the solution describes a fully mixed Nash equilibrium. Otherwise, there does not exist a fully mixed equilibrium.

**20.6** Suppose that we are given an instance $G$ of the load balancing game with $m$ identical machines and $n$ tasks whose weights are bounded from above by $\alpha \cdot \text{opt}(G)$, for $0 < \alpha < 1$.

**(a)** Show that $\text{cost}(A) < (1 + \alpha) \cdot \text{opt}(G)$, for every equilibrium assignment $A$.
**(b)** Let $\alpha = \frac{1}{\log m}$. Show that $\text{cost}(A) = \mathcal{O}(\text{opt}(G))$, for every equilibrium strategy profile $P$.

# Routing Games

Rahul Savani

Department of Computer Science, University of Warwick
Coventry, CV4 7AL, United Kingdom
email: rahul@dcs.warwick.ac.uk

June 7, 2007

In this short section, we briefly introduce routing games, which are multi-player games played on directed graphs. We define the price of anarchy, which is a measure of the inefficiency of selfish behaviour. To illustrate the main ideas we study two important and standard examples due to Pigou (1920) and Braess (1968).

## 1 Directed graphs

A *directed graph* (or *digraph*) is a pair $(V,E)$, where $V$ is the set of *nodes* (also called *vertices*), and $E$ is a collection of (ordered) node pairs $(u,v)$, called *edges* (or sometimes *links*). Figure 1 is an example. A *path* from $u$ to $v$ of length $k$ is a sequence of edges using nodes $u_0,u_1,u_2,\ldots,u_k$, where $u_0 = u$, $u_k = v$, and $(u_i,u_{i+1}) \in E$ is an edge for $i = 0,\ldots,k-1$.

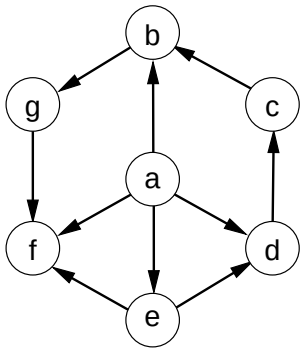



Figure 1: An example of a digraph, with vertex set $V = \{a,b,c...,g\}$ and edge set $E = \{(a,b),(a,d),(a,e),(a,f),(b,g),(c,b),(d,c),(e,d),(e,f),(g,f)\}$.

We allow multiple edges, as in Figure 2, so formally, the edges form a multiset[1] $E$ rather than a just a set.

[1]A multiset is a pair $(S,m)$, where $S$ is a set and $m : S \mapsto \mathbb{N}$ is a function from the set $S$ to the natural numbers. For $s \in S$, the multiplicity of $s$ is $m(s)$. For example, $M = \{x,x,y,z,z,z\}$ is a multiset with $S = \{x,y,z\}$ and $m(x) = 2$, $m(y) = 2$, and $m(z) = 3$.

# 2 Networks and routing games

We consider routing games played on directed graphs. The games are either *atomic* (finite number of players) or *non-atomic* with a continuum of (infinitely many) players.

Each player has a designated starting vertex and destination vertex. In our examples, often the starting vertex and/or the destination vertex of every player is the same. Each player chooses a path between his starting and destination vertex so as to *minimize* the cost of the chosen route or path.

The cost of using a path (the total travel time) is the sum of the costs of the individual edges used by the chosen path, which are determined by a *delay function* for each edge (every player using an edge experiences the same delay on that edge).

If the delay function of every edge is a constant function then the players each face a one-player decision problem, as their actions do not affect one another; these problems can be solved using a shortest path algorithm for weighted directed graphs. In all of our applications, the delay for at least one edge will depend on the number of players (for atomic games) or the fraction of players (for nonatomic games) that use the edge.

# 3 The price of anarchy

The *price of anarchy*, which is also called the *coordination ratio*, is a measure of the inefficiency of selfish behaviour. It requires a social cost function, the minimum of which is taken as the "efficient" social outcome.

**Definition 3.1** *Consider an n-player noncooperative game G. Let $W$ be a social cost function that assigns to each strategy profile of $G$ a real number that quantifies the social cost of that profile. Let $E$ be the set of Nash equilibria of $G$ ($E$ is a set of strategy profiles). Let $x^*$ be a strategy profile that attains the best (lowest) possible social cost over all possible strategy profiles of $G$. The price of anarchy is defined as*

$$\max_{e \in E} \frac{W(e)}{W(x^*)} .$$

In other words, the price of anarchy is the ratio of the social cost of the worst Nash equilibrium (in terms of social cost) to the best possible social cost (for any strategy profile). Clearly, the price of anarchy is at least one.

# 4 Pigou's example

Figure 2 shows an example of a simple nonatomic routing game in which the Nash equilibrium behaviour is not efficient, i.e., the price of anarchy is greater than one. This example is

due to Pigou (1920). In this game, one unit of splittable traffic must choose to use either the top or bottom edge to get from $s$ to $t$. So, if the flow on the bottom edge is $y \in [0,1]$, then the flow $x$ on the top edge is $(1-y)$. Delay functions are marked alongside the respecitve edge, so if a fraction $x$ of traffic uses the top edge it will experience a (constant) delay of 1 unit, and if a fraction $y$ of traffic uses the bottom edge it will experience $y$ units of delay.

**Proposition 4.1** *In the unique equilibrium of this game, all traffic uses the bottom edge.*

*Proof.* If all traffic uses the bottom edge, then all traffic experiences one unit of delay. No traffic has an incentive to deviate, i.e., switch to the top edge, since any traffic deviating would also experience one unit of delay. This shows that it is an equilibrium for all traffic to use the bottom edge. There is no other equilibrium, since if the flow $y$ on the bottom edge is less than one, then the delay on the bottom edge, which is also $y$, is less than one, so the flow of $(1-y)$ on the top edge would have an incentive to deviate and use the bottom edge. □

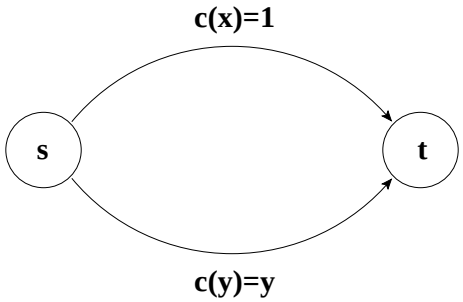



Figure 2: Pigou's example. In equilibrium, all traffic uses the bottom edge.

The social cost of an outcome is defined as the total travel time of all traffic, i.e.,

$$(1-y) \times 1 + y \times y \, .$$

**Proposition 4.2** *The price of anarchy in this game is 4/3.*

*Proof.* In order to compute the price of anarchy we must find the optimal social cost, which is equal to

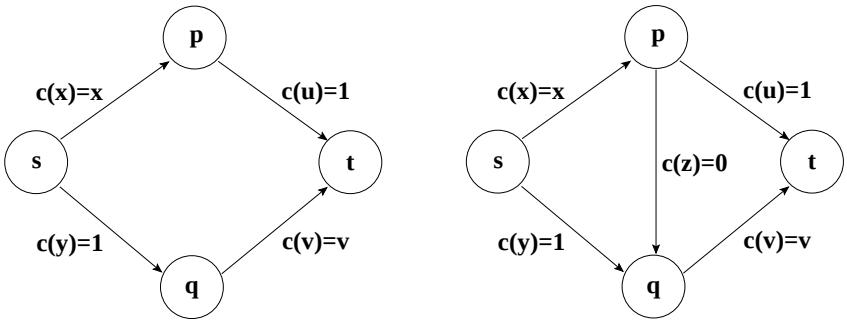
$$\min_{y \in [0,1]} (1-y) \times 1 + y \times y \, .$$

The minimum is $3/4$ and is achieved when $y = 1/2$. Thus, the price of anarchy is $\frac{1}{3/4}$, which is $4/3$, as claimed. □

It can be shown that this is as bad as it can get for any network (meaning the price of anarchy is at most 4/3), if we restrict the delay functions to be linear, i.e., they are of the form $ax+b$ for a flow $x$, for some nonnegative constants $a$ and $b$.

# 5 Braess's paradox

In the following example of Braess (1968), the addition of a new edge to the network actually makes the travel time of all users worse. In both games (a) and (b) depicted in Figure 3, one unit of splittable flow must start at $s$ and travel to $t$.




(a) Initial network (b) Augmented network

Figure 3: Braess's Paradox. The addition of an intuitively helpful edge can adversely affect all of the traffic.

**Proposition 5.1** *In the unique equilbrium of the game (a), traffic is split equally between the top route $s-p-t$ and the bottom route $s-q-t$.*

*Proof.* There are only two routes, $s-p-t$ and $s-q-t$. The delay on both routes increases as more traffic uses the route, and by symmetry, the only equilibrium is when the the flow on these two routes is equal, and all traffic experiences a travel time of 1.5 units. □

**Proposition 5.2** *In the unique equilibrium of (b), all traffic uses the route $s-p-q-t$.*

*Proof.* There are now three routes, $s-p-t$, $s-q-t$, and in addition $s-p-q-t$. The route $s-p-t$ has delay time $x+1$. The route $s-q-t$ has delay time $1+v$. The route $s-p-q-t$ has delay time $x+v$. Thus, the delay on route $s-p-q-t$, is never more than the delay of either of the other routes (since $x,v \in [0,1]$), and is striclty less that one or both of the other routes whenever $x$ or $v$ is less than one. Therefore, the unique equilibrium is for all traffic to use the route $s-p-q-t$, and hence experience a travel time of 2 units. □

The paradox is that the addition of an intuitively helpful edge (from $p$ to $q$) has actually made all of the traffic worse off. Braess's example can also occur where traffic uses all three routes in the augmented network in equilibrium (see worksheet 8). In the example here, where all traffic in the augmented network uses the additional edge in equilibrium, notice the similarity with Pigou's example.

# Algorithms, Games, and the Internet *

Christos H. Papadimitriou
University of California, Berkeley
christos@cs.berkeley.edu

## ABSTRACT

If the Internet is the next great subject for Theoretical Computer Science to model and illuminate mathematically, then Game Theory, and Mathematical Economics more generally, are likely to prove useful tools. In this talk I survey some opportunities and challenges in this important frontier.

## 1. INTRODUCTION

Over the past fifty years, researchers in Theoretical Computer Science have sought and achieved a productive foundational understanding of the von Neumann computer and its software, employing the mathematical tools of Logic and Combinatorics. The next half century appears now much more confusing (half-centuries tend to look like that in the beginning). What computational artifact will be the object of the next great modeling adventure of our field? And what mathematical tools will be handy in this endeavor?

The Internet has arguably surpassed the von Neumann computer as the most complex computational artifact (if you can call it that) of our time. Of all the formidable characteristics of the Internet (its size and growth, its almost spontaneous emergence, its open architecture, its unprecedented availability and universality as an information repository, etc.), I believe that the most novel and defining one is *its socio-economic complexity:* The Internet is unique among all computer systems in that it is built, operated, and used by a multitude of diverse economic interests, in varying relationships of collaboration and competition with each other. This suggests that the mathematical tools and insights most appropriate for understanding the Internet may come from a fusion of algorithmic ideas with concepts and techniques from Mathematical Economics and Game Theory[1] (see [18, 23] for two excellent introductions in the respective subjects, and see the web site www.cs.berkeley.edu/~christos/cs294.html for many additional references to work in this interface.)[2]

In this talk I shall review some of the many important points of contact between Game Theory and Economic Theory, Theoretical CS, and the Internet. In doing so I am necessarily (and, to an observer, arbitrarily) selective, leaving out important areas such as combinatorial auctions [5], and computational learning in games [9].

## 2. NASH EQUILIBRIUM

Game theory was founded by von Neumann and Morgenstern (in fact, about the same time von Neumann designed the EDVAC...) as a general theory of rational behavior. Game theoretic concepts are already familiar to theoretical computer scientists: Proving lower bounds is often best seen as a game between an algorithm designer and an adversary [30], while strategic two-person games are important complexity paradigms [1] and tools in finite model theory (Ehrenfeucht-Fraissé), for example. There has been fertile interaction in the recent past between Game Theory and CS Theory in the context of bounded rationality and repeated games [25] as well as learning games [29]. Game Theory's sharp but pointedly faithful modeling, twisted cleverness, and often unexpected depth make it quite akin to our field; but this may also be deceptive, since Game Theory is also characterized by a cohesive and complex research tradition and a defiantly original point of view and norms that are often hard to get accustomed to.

In a game, each of $n$ players can choose among a set of strategies $S_i, i = 1, \ldots, n$, and there are functions $u_i, i = 1, \ldots, n : S_1 \times \cdots \times S_n \mapsto \Re$ which assign to each such combined choice a payoff for each player. The fundamental question of Game Theory is, what constitutes rational behavior in such a situation? The predominant concept of rationality here (but my no means the only one) is the *Nash equlibrium:* A combination of strategies $x_1 \in S_1, \ldots, x_n \in S_n$ for which $u_i(x_1, \ldots, x_i, \ldots, x_n) \geq u_i(x_1, \ldots, x_i', \ldots, x_n)$ for all $i$ and

*An extended abstract of this paper appears in the proceedings of the 2001 ICALP Conference

[1]A more radical point of view would be this: Since computation has moved over the past twenty years decisively closer to people, interfaces with *social sciences* such as Psychology and Sociology, besides Economics, have become increasingly important, and TCS needs to build bridges with the mathematical vanguards of those fields.

[2]But why, one may ask, should we embark on the foundational understanding of something that was not designed — and seems inherently undesignable? First, when faced with a novel and complex computational phenomenon, it seems to me that our community has no choice but to study it. Second, the Internet *is* being engineered —albeit in a subtle, diffuse, and indirect way. Foundational insights will be, with any luck, noted and appreciated.


*STOC'01,* July 6-8, 2001, Hersonissos, Crete, Greece.

$x_i' \in S_i$; a behavior, that is, from which no player has an incentive to deviate.

The Nash equilibrium concept is not without its problems. First, a game may not have one; here Nash himself provided an ingenious way out, by showing that Nash equilibria always exist if the $S_i$'s are convex sets (the proof makes use of Kakutani's Theorem, a deep fact from topology with combinatorial origins, a generalization of Brouwer's Fixpoint Theorem); and a general way of making any set convex is to allow *distributions* (convex combinations) over the set. Thus, if we allow the $x_i$'s to be randomized ("mixed" in the field's terminology) strategies, then a Nash equilibrium always exists. (In fact, this generalizes the well-known minmax theorem for zero-sum games, and thus linear programming duality.) Second, there is typically more than one Nash equilibrium in a game, and there is no useful way of choosing between them —it is a "declarative" concept containing no recipe for "getting there." This is an obvious invitation to algorithmic ideas, and some have been tried.

But the most interesting aspect of the Nash equilibrium concept to our community is that *it is a most fundamental computational problem whose complexity is wide open.* Suppose that $n = 2$ and $S_1, S_2$ are finite sets. *Is there a polynomial algorithm for computing a (mixed) Nash equilibrium in such a game?* Because of the guaranteed existence of a solution, the problem is unlikely to be NP-hard; in fact, it belongs to a class of problems "between" P and NP, characterized by reliance on *the parity argument* for the existence proof [24]. In a different direction, as we have already pointed out, this problem is a generalization of linear programming; in fact, there is an algorithm for it that is a combinatorial generalization of the simplex algorithm [2] (as a corollary, the solution is always a vector of rational numbers, something that is not true in general for $n \geq 3$ players).

Together with factoring, *the complexity of finding a Nash equilibrium is in my opinion the most important concrete open question on the boundary of P today.*

## 3. INTERNET EQUILIBRIA

Internet bandwidth is scarce; its allocation to individual end-to-end flows is achieved via the TCP/IP congestion control protocol: "If the previous batch of packets got through, then increase the batch size by one; if not, decrease it by half." This ingeniously simple scheme seems to work, and its users do not seem eager to abandon it for something more aggressive, but the origins of this apparent success and acquiescence are not well understood. One is justified to wonder: *Of which game is TCP/IP congestion control the Nash equilibrium?* [12][3]

If we see Internet congestion control as a game, we can be sure that its equilibrium is not achieved by rational contemplation, but by interaction and adaptation in an environment where conditions (and player populations) change rapidly (and in which changes in strategy incur costs). These considerations may lead to more sophisticated concepts of equilibria that are more appropriate for the context of the Internet; see [8] for initial work in this direction.

Games as defined above assume that players cannot negotiate with and compensate each other by side payments. *Coalitional game theory* [23] considers a game of $n$ players as a set of possible $2^n - 1$ coalitions, each of which, call it $S$, can achieve a particular *value* $v(S)$ (the best possible sum of payoffs among players in $S$, against worst-case behavior of players in $[n]-S$). The problem is now how to divide the total payoff $v([n])$ among the $n$ players. Many such notions of "fairness" have been proposed, defended and criticized over the past decades: The Shapley value, the kernel, the bargaining set, the nucleolus, the von Neumann-Morgenstern solution, and many others (see [23], and [3] for a complexity-theoretic treatment of the subject). The core is perhaps the most intuitive (and akin to the equilibrium concept); it is also the most conservative (as a result, games often have empty core): A vector $x \in \Re_+^n$ with $x[[n]] = v([n])$ (notation: $x[S] = \sum_{i \in S} x_i$) is in the core if $x[S] \geq v([S])$ for all $S$. That is, $x$, considered as a proposed splitting of the total payoff $v([n])$ among the $n$ players, is fair according to the core school if no coaltion has an incentive to sucede (because no coalition can make more by itself than it is allocated in $x$).

The Internet seems to me an intriguing theater of coalitional game theory. It is operated (and built) by thousands of large and small entities ("autonomous systems"), collaborating with admirable effectiveness to process and deliver end-to-end flows originating and terminating in any one of them, using an opaque protocol called BGP [27]. Consider the following abstraction of the situation: We are given a graph with $n$ nodes (the autonomous systems); an $n \times n$ symmetric traffic matrix $F$, where $f_{ij}$ is the total traffic requirements between customers of $i$ and customers of $j$; and a capacity $c_i$ for each node (a simplification attempting to capture the capacity of $i$'s subnetwork to carry traffic). If $S$ is a set of nodes, consider the subgraph induced by $S$ as a multicommodity network with node capacities and commodity requirements given by the entries of $F$; let $v(S)$ be the maximum total flow in this network —notice that this defines a coalitional game.

The key problem here is this: Find an optimum solution in the multicommodity flow problem for the overall network, achieving a flow matrix $F' \leq F$, such that the corresponding payoffs for the nodes $x_i = \sum_j f_{ij}'$ are in the core of the coalitional game $v$ (or abide by one of the other notions of fairness mentioned above). Here we assume that autonomous system $i$'s payoff increases with the flow to and from $i$'s customers.

## 4. THE PRICE OF ANARCHY

There is no central authority that designs, engineers and runs the Internet. [4] But what if there were such master puppeteer, a benevolent dictator who, for example, micromanaged its operation, allocating bandwidth to flows so as to maximize total satisfaction? How much better would the Internet run? *What is the price of anarchy?*

This question was posed (and partially answered in the restricted context of a network consisting of two nodes and parallel edges) in [16]. This is an instance of a more general

[3] In recent work, Kelly [10] establishes, by resorting to ODEs and Lyapunov functions, that TCP/IP is the control function that optimizes the sum of user utilities (assumed to be of the form $\arctan(cb)$ where $b$ is the bandwidth allocated to the user and $c$ a constant) minus the total number of packet drops.

[4] Recall David Clark's famous maxim: "We reject kings, presidents and voting. We believe in rough consensus and running code."

pattern, of a novel and timely genre of problems: Given a game-like situation, we seek the ratio between the worst-case Nash equilibrium and the optimum sum of payoffs. In other words, if competitive analysis reveals the price of not knowing the future, and approximability captures the price of not having exponential resources, the present analysis seeks the price of uncoordinated individual untility-maximizing decisions —that is to say, the price of anarchy. Since that paper there has been progress in this front (not least, in the present conference [19]), including a marvelous result [28] stating that, in the context of a multicommodity flow network in which message delays increase with edge congestion while flows choose paths so as to minimize delay, the price of anarchy is *two* (more precisely, the ararchistic solution is no worse than the optimum solution with double the bandwidth).

But, of course, in today's Internet flows cannot choose shortest paths. In the Internet, routers direct traffic based on local information, users respond to delay patterns by modifying their traffic, and network providers throw bandwidth at the resulting hot spots. How does this compare in efficiency with an ideal, *ab initio* optimum design? *What is the price of the Internet architecture?*

## 5. ROUGH MARKETS

As a further example of the possible interaction between Economic Theory and computational ideas, let me present an important economic model that transcends games, namely *markets,* a fundamental theorem about them, as well as a recent result by Deng Xiaotie and myself. We again have $n$ agents, each of which possesses a nonnegative vector (its *endowment*) $e_i \in \Re_+^k$ of $k$ *goods*, and a concave *utility function* $u_i$ mapping $\Re_+^k$ to $\Re_+$. The agents may be all dissatisfied with their current endowments, in that there may be a reallocation of the same goods that is higher in everybody's utility (the set of local optimal allocations for which such overall improvement is impossible comprise the *Pareto set* of the market). Bilateral and multi-way exchanges and bartering may slowly improve the situation inching towards the Pareto set, but at considerable delay and communication cost.

Prices can be seen as an ingenious and efficient (even in the CS sense of low communication complexity) way for reaching the Pareto set. Suppose that there is a per unit price $p_j$ for each good, a nonnegative real number. The only rational behavior for each agent would then be, to sell her endowment $e_i$ at these prices, and to buy with the proceeds $p \cdot e_i$ a new vector of goods $\hat{x_i} \in \Re_+^k$ that is the "best vector she can afford," that is, the solution to the following optimization problem: max $u_i(x_i)$ such that $p \cdot x_i \leq p \cdot e_i$. But what guarantees do we have that there will be enough goods to fill everybody's "optimum affordable shopping cart?" Or that no goods will be left on the shelves?

**Theorem (Arrow-Debreu, 1953):** (Under some technical but reasonable assumptions about the $u_i$'s,) there is always a price vector $p$ called the *price equilibrium* such that *the market clears*, that is, the solutions $\hat{x_i}$ to the optimization problems above satisfy $\sum_{i=1}^n \hat{x_i} = \sum_{i=1}^n e_i$.

The proof uses Brouwer's fixpoint theorem. In a situation mirroring Nash equilibrium, there is no known polynomial algorithm for computing equilibrium prices in an economy (the corresponding problem is in fact *complete* for the parity class mentioned above [24]), even though there are empirically good algorithms.

Economists had been aware for half a century that the Arrow-Debreu theorem breaks down when the goods are discrete (bridges, days of work, airplanes, etc.). The following recent result captures this diffuse awareness in a computational context, and povides a remedy:

**Theorem:** [4] If (some of) the goods are integer-valued, then a price equilibrium may not exist, and it is in fact (strongly) NP-hard to tell if a price equilibrium exists (weakly NP-hard even when $n = 3$), even when the utilities are linear. However, (under some technical but reasonable assumptions about the $u_i$'s,) there is a fully polynomial-time approximation scheme that computes a price equilibrium that is $\epsilon$-*approximate in expectation* (definition omitted) if the number of goods is fixed.

The second (approximation) part uses randomized rounding [26].

## 6. MECHANISM DESIGN

If Game Theory strives to understand rational behavior in competitive situations, the scope of Mechanism Design (an important and elegant research tradition, very extensive in both scope and accomplishment, and one that could alternatively be called "inverse game theory") is even grander: Given desired goals (such as to maximize a society's total welfare), design a game (strategy sets and payoffs) in such a clever way that individual players, motivated solely by self-interest, end up achieving the designer's goals. There have been recently interesting interactions between this fascinating area and Theoretical CS, see e.g. [22, 7], and further opportunities abound. This area is too sophisticated and developed for a brief tutorial to be meaningful (see [18] for an excellent chapter, and [21] for a TCS-friendly introduction). Instead, I shall briefly develop an argument for its importance.

The complex socio-economic context of the Internet can have a deep influence on the design process in CS, and the research agenda of Theoretical CS. Traditionally, the "goodness" or "fitness" of a computational artifact (a new compiler, say) could be captured by its time and space performance, as well as its reliability, usability, etc. Such attributes were a fair approximation to the artifact's "fitness" (its chances for success), and theoreticians strived to develop methodologies for predicting and optimizing these attributes.

In the context of the Internet, such attributes only tell a small part of the story. If an artifact (a new congestion control protocol, a new caching scheme, a new routing algorithm, etc.) is demonstrated to have superior perfomance, this does not necessarily mean that it will be successful. For the artifact to be "fit," there must exist a *path* leading from the present situation to its prevalence.[5] This path must be paved with incentives that will motivate all kinds of diverse agents to adopt it, implement it, use it, interface with it, or just tolerate it. In the absence of such a path, the most

[5]This is not unlike biological systems, where it is known that successful genes and traits are the ones for which there is a continuously fitness-increasing path leading from the current phenotype and genotype to the target ones. See [20] for a fascinating interplay between between Game Theory and Biology.

clever, fast, and reliable piece of software may stay just that. All design problems are now mechanism design problems.

## 7. THE ECONOMICS OF PRIVACY, CLUSTERING, AND THE WEB GRAPH

There is no end to the list of computational matters for which the economic viewpoint leads to interesting insights, as well as novel algorithmic problems. I briefly discuss here three more examples from my recent work.

**Privacy** is arguably the most urgent concern and mission of Computer Science, and yet there is very little foundational work about it. In a recent paper [14] we argue that it has an important economic aspect: The problem with privacy is that decisions about the use of personal information are made by entities other than the person involved (such anomalies, called *externalities*, are known in Economics to be the root of most evil); to put it otherwise, personal information is intellectual property that bears negative royalty. Coalitional game theory is an interesting modeling tool here as well, since it can help determine the fair royalty due to the individual for any use of his or her private information. In [14] we study certain stylized versions of common situations (such as marketing surveys and collaboratie filtering) in which personal information is used, and the interesting algorithmic problems involved in computing fair royalties.

**Clustering** is one of the most practically important yet foundationally underdeveloped areas of CS (despite the steady stream of clever algorithmic ideas for approximately solving optimum clustering problems). There are far too many criteria for the "goodness" of a clustering (min-sum or min-max cluster diameter, min-sum or min-max distance from the center (which is either forced to be one of the original points or is not), or novel spectral parameters, to name only the ones most popular at STOC) and far too little guidance about choosing among them. In [15] we point out that economic considerations are crucial for understanding the issues here as well. For the purpose of clustering is to improve decision-making by allowing different segments of a space or population to be treated differently. The criterion for choosing the best clustering scheme cannot be determined unless the decision-making framework that drives it is made explicit. In [15] we show that this point of view gives rise to a host of novel optimization problems, called *segmentation problems*.

Consider the following hypothetical situation: A monopolist knows the demand curve of each of its customers: If the price is $x$, the $i$th customer will buy a quantity $y = b_i - a_i \cdot x$. The monopolist wants to cluster its customers into $k$ segments, with a different price to each segment, in order to maximize revenue. How should this clustering be done? Which one of the two dozen or so criteria in the theoretical and experimental literature should be adapted here, and which approximation algorithm or heuristic should be used?

**Theorem:** The clustering that maximizes revenue subdivides the customers into segments of consecutive values of $\frac{a_i}{b_i}$. Thus, the optimum can be computed in $O(n^2)$ by dynamic programming.

The most attractive feature of the clustering algorithm suggested by this result is not that it is efficient, or that it finds the exact optimum. It is that *it optimizes the right thing*.

**The Web Graph.** It has been established recently [13, 11] that the world-wide web can be usefully considered as a directed graph with the documents as nodes and the hyperlinks as edges. Intuitively, the web is a huge "random" graph —except that it seems to violate every single prediction of the classical random graph models, such as $G_{n,p}$, so familiar to our community. For example, its indegrees and outdegrees of the nodes are distributed by a polynomial-tailed distribution (the $x$-th largest indegree or outdegree is about $c \cdot x^{-\alpha}$ for positive constants $c$ and $\alpha$) instead of the sharp Gaussian predicted by the law of large numbers, its giant strongly connected component covers about 40% of the nodes (instead of 0% or 100%), there are scores of $K_{3,3}$ subgraphs (instead of none), etc. Recently, there have been interesting efforts to model the web graph (see, for example, [17] for some of the latest). Despite much interest, we know of no model that predicts and explains to a satisfactory degree these and other features of the web graph, starting from primitive and credible assumptions about the world-wide web.

Heavy-tailed distributions were first observed in Economics: City populations are known to behave this way (the population of the $x$-th largest city of any country is eerily close to $c/x$, with $c$ depending on the country), but also market shares (the market share of the $x$th largest manufacturer in any sector is often distributed as $c \cdot x^{-\alpha}$), as well as income distributions. Is it possible that the puzzling structure of the web graph has its origins in economic phenomena of this kind? This is not as implausible as it may seem at first: A document's indegree is determined by how *interesting* the document is, and interest is intuitively analogous to market share (and as much determined by fierce competition...), while outdegree depends on the entity's *attention*, which seems to me not entirely unlike income. With several students at Berkeley we are running experiments to determine the explanatory power of such considerations.

Last, heavy-tailed distributions are also observed beyond the world-wide web, in the Internet: The degrees of the routers and the autonomous systems are also heavy-tail distributed [6]. It would be interesting to expore if this can be the result of some rough local optimization heuristic used to allocate new links and routers, in the face of exponentially growing traffic.

## 8. SOME OPEN PROBLEMS

Great new areas are full of open problems (or, more likely, no area can get off the ground until a slew of cool and challenging open problems attracts the attention of smart, ambitious researchers). To recapitulate, here are the ones proposed in this talk:

- Is there a polynomial algorithm for computing a Nash equilibrium in a 2-person game? For $n \geq 3$ players? A polynomial-time approximation scheme?
- Ditto for a market equilibrium. Is there a PTAS for integer markets whose exponent does not involve the number of goods?
- Develop a reasonably faithful game-theoretic model of Internet congestion control for which (an approximation of) TCP/IP is a Nash equilibrium. Alternatively, develop a crisp notion of equilibrium appropriate for games modeling the Internet.

- Is there a polynomial algorithm for determining, given a network, a traffic matrix, and node capacities, whether there is a flow in the game's core? More interestingly, under what circumstances (conditions relating connectivity, capacities and traffic matrix) is such a flow guaranteed to exist?

- What is the price of the Internet architecture? (See Section 4.)

- Develop a graph generation model for the world-wide web, plausibly capturing key aspects at a primitive level (my conjecture: its economic aspects are indispensible), that predicts theoretically the observed characteristics of the web graph.

- Show that a particular simple local improvement heuristic for network design in the face of increasing flow demands results in heavy tails in node degrees, if driven by exponentially increasing flow demands.

## 9. REFERENCES


[1] Chandra, Kozen, Stockmeyer, "Alternation," *JACM, 28*, 1, 1981.

[2] Cottle *The Linear Complementarity Problem*, 1992.

[3] Deng, Papadimitriou, "On the complexity of cooperative game solution concepts," *Math. Oper. Res., 19*, 1994.

[4] Deng, Papadimitriou, in preparation.

[5] de Vries, Vohra, "Combinatorial Auctions: A Survey." http://www.kellogg.nwu.edu/research/math/Down-papers.htm

[6] Faloutsos, Faloutsos, Faloutsos, "On power-law relationships of the Internet topology," *Proc. 1999 SIGCOMM.*

[7] Feigenbaum, Papadimitriou, Shenker "Sharing the cost of multicast transmissions," *Proc. 2000 STOC.*

[8] Friedman, Shenker "Learning and implementation on the Internet," 1998.

[9] Fudenberg *Learning in Games*, M.I.T. Press, 1998.

[10] Kelly "Mathematical modelling of the Internet," in *Mathematics Unlimited - 2001 and Beyond* (Editors B. Engquist and W. Schmid), Springer-Verlag, 2001.

[11] www.google.com/technology/index.html

[12] Karp, Koutsoupias, Papadimitriou, Shenker, "Optimization problems in congestion control," *Proceedings of the 41st FOCS*, 2000.

[13] Kleinberg "Authoritative sources in a hyperlinked environment," *Proc. 1999 SODA.*

[14] Kleinberg, Papadimitriou, Raghavan "On the value of private information," *Proc. 2001 Conf. on Theoretical Aspects of Reasoning about Knowledge,* Sienna, July 2001.

[15] Kleinberg, Papadimitriou, Raghavan "Segmentation problems," *Proc. 1998 STOC.*

[16] Koutsoupias, Papadimitriou "Worst-case equilibria," *Proc. 1998 STACS.*

[17] Kumar, Raghavan, Rajagopalan, Sivakumar, Tomkins, Upfal, "Random graph models for the Web graph," *Proc. 2000 FOCS.*

[18] Mas-Collel, Winston, Green *Microeconomic Theory*, Oxford University Press 1995.

[19] Mavronikolas, Spirakis, "The price of selfish routing," *this conference.*

[20] Maynard-Smith, *Evolution and the Theory of Games* Cambridge University Press, 1982.

[21] Nisan "Algorithms for selfish agents – Mechanism design for distributed computation," *Proc. 1999 STACS.*

[22] Nisan, Ronen "Algorithmic Mechanism Design," *Proc. STOC 1999.*

[23] Osborne and Rubinstein, *A Course in Game Theory*, MIT Press, 1994.

[24] Papadimitriou "On the complexity of the parity argument and other inefficient proofs of existence, *J.CSS, 48,* 3, 1994.

[25] Papadimitriou, Yannakakis "Complexity as bounded rationality," *Proc. 1994 STOC.*

[26] Raghavan, "Probabilistic construction of deterministic algorithms: approximate packing integer programs," *J.CSS, 37,* 1988.

[27] Rekhter, Li "RFC 1771, BGP-4," www.cis.ohio-state.edu/cgi-bin/rfc/rfc1771.html.

[28] Roughgarden, Tardos, "How Bad is Selfish Routing? (Extended Abstract)," *Proc. FOCS 2000.*

[29] Singh, Kearns, Mansour, "Nash Convergence of Gradient Dynamics in General-Sum Games," *Proc. UAI 2000.*

[30] Yao "Probabilistic computations: Toward a unified measure of complexity (extended abstract)," *Proc. 1977 FOCS.*

CHAPTER 19

# Network Formation Games and the Potential Function Method

Éva Tardos and Tom Wexler

## Abstract

Large computer networks such as the Internet are built, operated, and used by a large number of diverse and competitive entities. In light of these competing forces, it is surprising how efficient these networks are. An exciting challenge in the area of algorithmic game theory is to understand the success of these networks in game theoretic terms: what principles of interaction lead selfish participants to form such efficient networks?

In this chapter we present a number of network formation games. We focus on simple games that have been analyzed in terms of the efficiency loss that results from selfishness. We also highlight a fundamental technique used in analyzing inefficiency in many games: the potential function method.

## 19.1 Introduction

The design and operation of many large computer networks, such as the Internet, are carried out by a large number of independent service providers (Autonomous Systems), all of whom seek to selfishly optimize the quality and cost of their own operation. Game theory provides a natural framework for modeling such selfish interests and the networks they generate. These models in turn facilitate a quantitative study of the trade-off between efficiency and stability in network formation. In this chapter, we consider a range of simple network formation games that model distinct ways in which selfish agents might create and evaluate networks. All of the models we present aim to capture two competing issues: players want to minimize the expenses they incur in building a network, but at the same time seek to ensure that this network provides them with a high quality of service.

There are many measures by which players might evaluate the quality of a network. In this chapter, we focus primarily on measures of distance (Section 19.2) and connectivity (Section 19.3), rather than measures based on congestion effects (as is done in Chapter 18). We also assume that players have financial considerations. In Sections 19.2 and 19.3, players seek to minimize the construction costs of the networks they

create. In Section 19.4, we look at a game with a more sophisticated financial aspect: players represent service providers who set prices for users and seek to maximize their profit, namely their income from users minus the cost of providing the service.

For all of the games we consider, we use Nash equilibrium as the solution concept, and refer to networks corresponding to these equilibria as being *stable*. The models we focus on involve players who can unilaterally build edges, and thus the Nash equilibrium solution concept is appropriate.

To evaluate the overall quality of a network, we consider the *social cost*, or the sum of all players' costs. We refer to the networks that optimize social cost as *optimal* or *socially efficient*. The main goal of this chapter is to better understand the quantitative trade-off between networks that are stable and those that are socially efficient. More precisely, we are interested in bounding the price of anarchy and the price of stability (as defined in Chapter 17). The models we consider in this chapter are network formation games in which these measures are provably small.

In Section 19.2 we consider a local connection game where the nodes of the graph are players who pay for the edges that connect them directly to other nodes (incident edges). In selecting a strategy, players face two conflicting desires: to pay as little as possible, and to have short paths to all other nodes. Our goal here is to bound the efficiency loss resulting from stability. Such connection games have been extensively studied in the economics literature (see Jackson (2006) for a survey) to model social network formation, using edges to represent social relations. The local connection game can also be thought of as a simple model for the way subnetworks connect in computer networks (by establishing peering points), or as modeling the formation of subnetworks in overlay systems such as P2P (peer-to-peer) networks connecting users to each other for downloading files.

We will use a model in which players can form edges to a neighbor unilaterally, and will use Nash equilibrium as our solution concept. This differs from much of the literature in economics, where it is typically assumed that an edge between two players needs the consent or contribution from both players, and where the notion of pairwise stability is used instead of Nash equilibria. We will discuss how the results in Section 19.2 extend to models using pairwise stable equilibria in the notes in Section 19.5.1.

The model we examine was introduced by Fabrikant et al. (2003) and represents the first quantitative effort to understand the efficiency loss of stable networks. In this game, a single parameter $\alpha$ represents the cost of building any one edge. Each player (represented by a node) perceives the quality of a network as the sum of distances to all other nodes. Players aim to minimize a cost function that combines both network quality and building costs: they attempt to minimize the sum the building costs they incur and the distances to all other players. Thus, players use $\alpha$ as a trade-off parameter between their two objectives. This is perhaps the simplest way to model this type of trade-off. While the simplicity of this game makes it easy to evaluate, such a stylized model ignores a number of issues, such as varying costs and possible congestion effects. In Section 19.5.1, we discuss related models that address some of these issues.

In Section 19.3 we study a very different (and also quite simple) model of network design, introduced by Anshelevich et al. (2004), called the global connection game. Whereas players in the game of Section 19.2 only make local choices (which other nodes to link to), players in this game make global decisions, in that they may build edges

throughout the network. Unlike the local connection game, this global game attempts to model players who actually build and maintain large-scale shared networks. This model also allows for greater heterogeneity in the underlying graph.

In the global connection game, a player is not associated with an individual node of the networks, but instead has certain global connectivity goals. To achieve these goals, a player may contribute money to any set of edges in the network. As before, we view connectivity as the primary measure of quality. However, players do not desire uniform connectivity; instead, each player has a subset of nodes that it needs to connect, and aims to do so as cheaply as possible. Furthermore, unlike in the local game, players are not concerned with distance, and simply want to connect their terminals.

As in the previous model, players are sensitive to costs. Edge $e$ has a cost $c_e \geq 0$, and players who use $e$ share this cost. In particular, we focus on a *fair sharing rule*; all players using an edge must share its cost evenly. This natural cost-sharing scheme can be derived from the Shapley value, and has many nice properties. We also examine other cost-sharing games, and discuss the role of fair sharing in the price of stability results.

A key technique used in this section is the potential function method. This method has emerged as a general technique in understanding the quality of equilibria. We review this technique in detail in Section 19.3.2. While this technique provides results only regarding the price of stability, it is interesting to note that many of the currently known price of anarchy results (e.g., most of the results in Part III of this book) are for potential games.

In Section 19.4, we consider another potential game; a facility location game with a more sophisticated cost model. In the previous two sections, players simply minimized their costs. Here, edges still have costs, but players also select prices for users so as to maximize net income: price charged minus the cost paid. We again consider a very simplified model in which players place facilities to serve clients, thereby forming a network between the providers and the clients. We show that a socially efficient network is stable (i.e., the price of stability is 1), and bound the price of anarchy.

In the context of facility location games, we also bound the quality of solutions obtained after sufficiently long selfish play, without assuming that players have yet reached an equilibrium. As we have seen in part I of this book, equilibrium solutions may be hard to find (Chapter 2), and natural game play may not converge to an equilibrium (Chapter 4). Thus it is often useful to evaluate the quality of the transient solutions that arise during competitive play. The facility location game considered in this section is one of the few classes of games for which this strong type of bound is known.

## 19.2 The Local Connection Game

In this section we consider the simple network formation game of Fabrikant et al. (2003), where players can form links to other players. We consider a game with $n$ players, where each player is identified with a node. Node $u$ may choose to build edges from $u$ to any subset of nodes, thereby creating a network. Players have two competing goals; players want to build (and thus pay) for as few edges as possible, yet they also want to form a network that minimizes the distance from their own node to

all others. Our main focus in this section is to quantitatively understand the inefficiency that results from the selfish behavior of these network builders.

### 19.2.1 Model

Players in the local connection game are identified with nodes in a graph $G$ on which the network is to be built. A strategy for player $u$ is a set of undirected edges that $u$ will build, all of which have $u$ as one endpoint. Given a strategy vector $S$, the set of edges in the union of all players' strategies forms a network $G(S)$ on the player nodes. Let $dist_S(u, v)$ be the shortest path (in terms of number of edges) between $u$ and $v$ in $G(S)$. We use $dist(u, v)$ when $S$ is clear from context. The cost of building an edge is specified by a single parameter, $\alpha$. Each player seeks to make the distances to all other nodes small, and to pay as little as possible. More precisely, player $u$'s objective is to minimize the sum of costs and distances $\alpha n_u + \sum_v dist(u, v)$, where $n_u$ is the number of edges bought by player $u$.

Observe that since edges are undirected, when a node $u$ buys an edge $(u, v)$, that edge is also available for use from $v$ to $u$, and in particular, is available for node $v$. Thus, at Nash equilibrium at most one of the nodes $u$ and $v$ pay for the connecting edge $(u, v)$. Also, since the distance $dist(u, v)$ is infinite whenever $u$ and $v$ are not connected, at equilibrium we must have a connected graph. We say that a network $G = (V, E)$ is *stable* for a value $\alpha$ if there is a stable strategy vector $S$ that forms $G$.

The social cost of a network $G$ is $SC(G) = \sum_{u \neq v} dist(u, v) + \alpha|E|$, the sum of players' costs. Note that the distance $dist(u, v)$ contributes to the overall quality twice (once for $u$ and once for $v$). We will be comparing solutions that are stable to those that are optimal under this measure.

### 19.2.2 Characterization of Solutions and the Price of Stability

We now characterize the structure of an optimal solution as a function of $\alpha$. A network is *optimal* or *efficient* if it minimizes the social cost $SC(G)$.

**Lemma 19.1** *If $\alpha \geq 2$ then any star is an optimal solution, and if $\alpha \leq 2$ then the complete graph is an optimal solution.*

**PROOF** Consider an optimal solution $G$ with $m$ edges. We know $m \geq n - 1$; otherwise, the graph would be disconnected, and thus have an infinite cost. All ordered pairs of nodes not directly connected by an edge must have a distance of at least 2 from each other, and there are $n(n-1) - 2m$ such pairs. Adding the remaining $2m$ pairs with distance 1 yields $\alpha m + 2n(n-1) - 4m + 2m = (\alpha - 2)m + 2n(n-1)$ as a lower bound on the social cost of $G$. Both a star and the complete graph match this bound. Social cost is minimized by making $m$ as small as possible when $\alpha > 2$ (a star) and as large as possible when $\alpha < 2$ (a complete graph). □

Both the star and the complete graph can also be obtained as a Nash equilibrium for certain values of $\alpha$, as shown in the following lemma.

**Lemma 19.2** *If $\alpha \geq 1$ then any star is a Nash equilibrium, and if $\alpha \leq 1$ then the complete graph is a Nash equilibrium.*

**PROOF** First suppose $\alpha \geq 1$, and consider a star. It turns out that any assignment of edges to incident players corresponds to a Nash equilibrium, but for this result, we need only demonstrate a single solution. In particular, consider the strategy in which player 1 (the center of the star) buys all edges to the other players, while the remaining $n - 1$ leaf players buy nothing. Player 1 has no incentive to deviate, as doing so disconnects the graph and thus incurs an infinite penalty. Any leaf player can deviate only by adding edges. For any leaf player, adding $k$ edges saves $k$ in distance but costs $\alpha k$, and thus is not a profitable deviation. Thus the star is a Nash equilibrium.

Now suppose $\alpha \leq 1$. Consider a complete graph, with each edge assigned to an incident player. A player who stops paying for a set of $k$ edges saves $\alpha k$ in cost, but increases total distances by $k$, so this outcome is stable. □

There are other equilibria as well, some of which are less efficient (see Exercise 19.6). However, these particular Nash equilibria, in conjunction with the above optimal solutions, suffice to upper bound the price of stability.

**Theorem 19.3** *If $\alpha \geq 2$ or $\alpha \leq 1$, the price of stability is 1. For $1 < \alpha < 2$, the price of stability is at most 4/3.*

**PROOF** The statements about $\alpha \leq 1$ and $\alpha \geq 2$ are immediate from Lemmas 19.1 and 19.2. When $1 < \alpha < 2$, the star is a Nash equilibrium, while the optimum structure is a complete graph. To establish the price of stability, we need to compute the ratio of costs of these two solutions. The worst case for this ratio occurs when $\alpha$ approaches 1, where it attains a value of

$$\frac{2n(n-1) - 2(n-1)}{2n(n-1) - n(n-1)/2} = \frac{4n^2 - 6n + 2}{3n^2 - 3n} < 4/3.$$

□

Exercise 19.3 shows that the complete graph is the unique equilibrium for $\alpha < 1$, so we also have that the price of anarchy is 1 in this range. We now address the price of anarchy for larger values of $\alpha$.

### 19.2.3 The Price of Anarchy

The first bound on the price of anarchy for this game was given by Fabrikant et al. (2003), and involves two steps: bounding the diameter of the resulting graph, and using the diameter to bound the cost. We begin with the second step.

**Lemma 19.4** *If a graph $G$ at Nash equilibrium has diameter $d$, then its social cost is at most $O(d)$ times the minimum possible cost.*

**PROOF** The cost of the optimal solution is at least $\Omega(\alpha n + n^2)$, as we need to buy a connected graph, which costs at least $(n-1)\alpha$, and there are $\Omega(n^2)$ distances, each of which is at least 1. To bound the quality of the solution, consider the distance costs and edge costs separately. The distance cost is at most $n^2 d$, and thus is at most $d$ times the minimum possible.

We now examine edge costs. First we consider *cut edges*, those edges whose removal disconnects $G$. There are at most $n-1$ cut edges, so the total cost of all cut edges is at most $\alpha(n-1)$, which in turn is at most the optimal solution cost. Now consider the set of all noncut edges paid for by a vertex $v$. We will argue that there are $O(nd/\alpha)$ such edges, with cost $O(dn)$ for node $v$, and thus the total cost of all noncut edges is $O(dn^2)$. This will establish that the cost of $G$ is $O(\alpha n + dn^2)$, completing the proof.

Pick a node $u$, and for each edge $e = (u, v)$ paid for by node $u$, let $V_e$ be the set of nodes $w$, where the shortest path from $u$ to $w$ goes through edge $e$. We will argue that the distance between nodes $u$ and $v$ with edge $e$ deleted is at most $2d$. Thus deleting $e$ increases the total distance from $u$ to all other nodes by at most $2d|V_e|$. Since deleting the edge would save $\alpha$ in edge costs and $G$ is stable, we must have that $\alpha \leq 2d|V_e|$, and hence $|V_e| \geq \alpha/2d$. If there are at least $\alpha/2d$ nodes in each $V_e$, then the number of such edges adjacent to a node $v$ must be at most $2dn/\alpha$, as claimed.

We now bound the distance between nodes $u$ and $v$ with edge $e$ deleted. Consider Figure 19.1, depicting a shortest path avoiding edge $e$. Let $e' = (u', v')$ be the edge on this path entering the set $V_e$. The segment $P_u$ of this path from $u$ to node $u'$ is the shortest path from $u$ to $u'$ as $u' \notin V_e$, and hence deleting $e$ does not affect the shortest path. So $P_u$ is at most $d$ long. The segment $P_v$ from $v'$ to $v$ is at most $d-1$ long, as $P_v \cup e$ forms the shortest path between $u$ and $v'$. Thus the total length is at most $2d$. □

Using this lemma, we can bound the price of anarchy by $O(\sqrt{\alpha})$.

**Theorem 19.5** *The diameter of a Nash equilibrium is at most $2\sqrt{\alpha}$, and hence the price of anarchy is at most $O(\sqrt{\alpha})$.*

**PROOF** From Lemma 19.4, we need only prove that for any nodes $u$ and $v$, $dist(u, v) < 2\sqrt{\alpha}$. Suppose for nodes $u$ and $v$, $dist(u, v) \geq 2k$, for some $k$. The

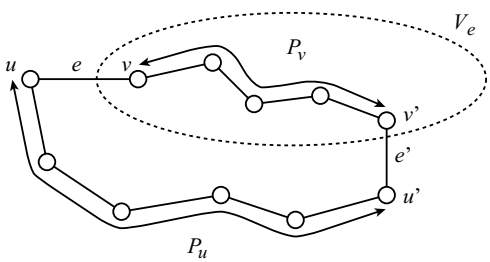



**Figure 19.1.** Path $P_u, (u', v')P_v$ is the $u$–$v$ shortest path after edge $e = (u, v)$ is deleted.

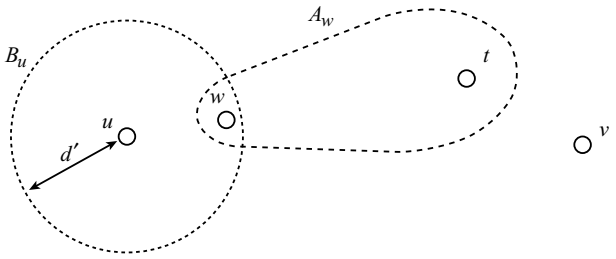



**Figure 19.2.** Nodes $u$ and $v$ that are at maximum distance $d$ apart. $B$ is the set of nodes at most $d' = (d-1)/4$ away from node $u$, and $A_w$ is the set of nodes whose shortest path leaves $B$ at $w$.

main observation is that by adding the edge $(u, v)$, the node $u$ would pay $\alpha$ and improve her distance to the nodes on the second half of the $u - v$ shortest path by $(2k-1) + (2k-3) + \cdots + 1 = k^2$. So if $dist(u, v) > 2\sqrt{\alpha}$, node $u$ would benefit from adding the edge $(u, v)$, a contradiction. □

We now show an $O(1)$ bound on the price of anarchy that was given by Lin (2003) (and independently also by Albers et al., 2006) for $\alpha = O(\sqrt{n})$.

**Theorem 19.6** *The price of anarchy is $O(1)$ whenever $\alpha$ is $O(\sqrt{n})$. More generally, price of anarchy is $O(1 + \alpha/\sqrt{n})$.*

**PROOF** We again use Lemma 19.4, so all we have to do is improve our bound on the diameter $d$. Consider nodes $u$ and $v$ with $dist(u, v) = d$. Let $d' = \lfloor (d-1)/4 \rfloor$ and let $B$ be the set of nodes at most $d'$ away from $u$, as shown on Figure 19.2. Consider how the distance $d(v, w)$ changes for nodes $w \in B$ by adding edge $(v, u)$. Before adding the edge $dist(v, w) \geq d - d'$. After adding $(v, u)$, the distance decreases to at most $d' + 1$. Thus $v$ saves at least $(d - 2d' - 1)$ in distance to all nodes in $B$, and hence would save at least $(d - 2d' - 1)|B| \geq (d-1)|B|/2$ in total distance costs by buying edge $(v, u)$. If $G$ is stable, we must have $(d-1)|B|/2 \leq \alpha$.

For a node $w \in B$ let $A_w$ contain all nodes $t$ for which the $u$–$t$ shortest path leaves the set $B$ after the node $w$. Note that if $A_w$ is nonempty, then $w$ must be exactly at distance $d'$ from $u$. Therefore, node $u$ would save $|A_w|(d'-1)$ in distance cost by buying edge $(u, w)$. If the network is at equilibrium, then we must have that $|A_w|(d'-1) \leq \alpha$. There must be a node $w \in B$ that has $|A_w| \geq (n - |B|)/|B|$. Combining these, we get that

$$(d'-1)(n - |B|)/|B| \leq \alpha.$$

This implies that $|B|(1 + \alpha/(d'-1)) \geq n$, and since $\alpha > d > d'$,

$$|B| \geq n(d'-1)/2\alpha.$$

Combining this with the previous bound of $\alpha \geq (d-1)|B|/2$ yields

$$\alpha \geq (d-1)|B|/2 \geq (d-1)n(d'-1)/4\alpha \geq n(d'-1)^2/\alpha.$$

Thus $\alpha^2 \geq n(d'-1)^2$ and hence $d \leq 4(d'+1)+1 \leq 4\alpha/\sqrt{n}+9$, which implies the claimed bound by Lemma 19.4. □

## 19.3 Potential Games and a Global Connection Game

In this section we introduce a broad class of games known as *potential games*. This class encompasses a number of natural and well-studied network-based games. As we will see, potential games possess many nice properties; pure equilibria always exist, best response dynamics are guaranteed to converge, and the price of stability can be bounded using a technique called the *potential function method*. Our motivating example for this class of games is a network formation game called the *global connection game*, which was discussed in Chapter 17. We begin by defining this game, and present some theorems about pure equilibria and the price of stability. We then introduce potential games, and provide generalized results for this broader framework.

The network formation game discussed in Section 19.2 is local in the sense that a player can build links to other nodes, but has no direct means for affecting distant network structure. Such might be the case with social networks or peering relationships in a digital network. The global connection game, in contrast, models players who make global structural decisions; players may build edges throughout the network, and thus consider relatively complex strategies. This game might be more appropriate for modeling the actual construction and maintenance of large-scale physical networks.

Beyond the varying scope of players' strategies, there are two additional features that differentiate these network formation games. First, in exchange for the global connection game's broader strategy space, we consider a relatively simplified player objective function. In particular, we assume that players are unconcerned with their distance to other nodes in the network, and instead want only to build a network that connects their terminals as cheaply as possible. The second notable distinction is that the global connection game supports cooperation, in that multiple players may share the cost of building mutually beneficial links. In the local connection game, an edge might benefit multiple players, and yet the edge's cost is always covered fully by one of the two incident players. We now give a formal description of the global connection game.

### 19.3.1 A Global Connection Game

We are given a directed graph $G = (V, E)$ with nonnegative edge costs $c_e$ for all edges $e \in E$. There are $k$ players, and each player $i$ has a specified source node $s_i$ and sink node $t_i$ (the same node may be a source or a sink for multiple players). Player $i$'s goal is to build a network in which $t_i$ is reachable from $s_i$, while paying as little as possible to do so. A strategy for player $i$ is a path $P_i$ from $s_i$ to $t_i$ in $G$. By choosing $P_i$, player $i$ is committing to help build all edges along $P_i$ in the final network. Given a strategy for each player, we define the constructed network to be $\cup_i P_i$.

It remains to allocate the cost of each edge in this network to the players using it, as this will allow players to evaluate the utility of each strategy. In principle, there are

a vast number of possible cost-sharing mechanisms, each of which induces a distinct network formation game. We will briefly touch on this large space of games at the end of the section, but for now, our primary focus will be on a single cost-sharing mechanism with a number of nice properties, that is both simple and easy to motivate.

In particular, we consider the mechanism that splits the cost of an edge evenly among all players whose path contains it. More concretely, if $k_e$ denotes the number of players whose path contains edge $e$, then $e$ assigns a cost share of $c_e/k_e$ to each player using $e$. Thus the total cost incurred by player $i$ under a strategy vector $S$ is given by

$$\text{cost}_i(S) = \sum_{e \in P_i} c_e/k_e.$$

Note that the total cost assigned to all players is exactly the cost of the constructed network. This equal-division mechanism was suggested by Herzog et al. (1997), and has a number of basic economic motivations. Moulin and Shenker prove that this mechanism can be derived from the Shapley (2001) value, and it can be shown to be the unique cost-sharing scheme satisfying a number of natural sets of axioms (see Feigenbaum et al., 2001; Moulin and Shenker, 2001). We refer to it as the *fair* or *Shapley cost-sharing mechanism*. The social objective for this game is simply the cost of the constructed network.

One may view this game as a competitive version of the generalized Steiner tree problem; given a graph and pairs of terminals, find the cheapest possible network connecting all terminal pairs. Indeed, an optimal generalized Steiner tree is precisely the outcome against which we will compare stable solutions in evaluating the efficiency of equilibria. This connection highlights an important difference between this game and routing games; in routing games such as those discussed in Chapter 18, players are sensitive to congestion effects, and thus seek sparsely used paths. But in the global connection game, as with the Steiner forest problem, the objective is simply to minimize costs, and thus sharing edges is in fact encouraged.

The two examples in Chapter 17 provide a few useful observations about this game. Example 17.2 (see Figure 19.3(a)) shows that even on very simple networks, this game has multiple equilibria, and that these equilibria may differ dramatically in quality. There are two equilibria with costs $k$ and 1 respectively. Since the latter is also optimal

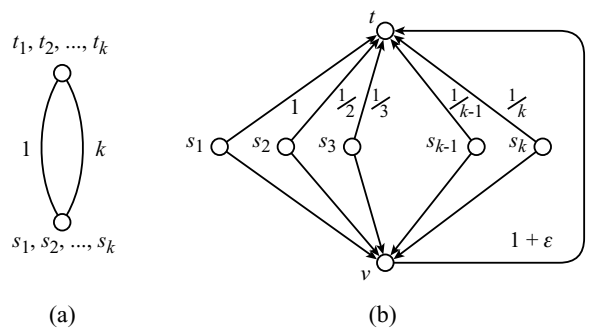



**Figure 19.3.** An instance of the global connection game with price of anarchy $k$ (a) and an instance with price of stability $\mathcal{H}_k$ (b).

solution, the price of anarchy is $k$, while the price of stability is 1. It is not hard to show that the price of anarchy can never exceed $k$ on any network (see Exercise 19.9), and thus this simple example captures the worst-case price of anarchy. Our primary goal will be to bound the price of stability in general.

Example 17.3 (see Figure 19.3(b)) shows that the price of stability can indeed exceed 1; this network has a unique Nash equilibrium with cost $\mathcal{H}_k$, the $k$th harmonic number, while the optimal solution has a cost of $1 + \epsilon$. Thus, the price of stability on this network is roughly $\mathcal{H}_k$. Our aim is to prove that pure equilibria always exist and provide an upper bound the price of stability. Both of these results make use of a *potential function*, which we will formally introduce in Section 19.3.2.

Consider an instance of the global connection game, and a strategy vector $S = (P_1, P_2, \ldots, P_k)$ containing an $s_i - t_i$ path for each player $i$. For each edge $e$, define a function $\Psi_e(S)$ mapping strategy vectors to real values as

$$\Psi_e(S) = c_e \cdot \mathcal{H}_{k_e},$$

where $k_e$ is the number of players using edge $e$ in $S$, and $\mathcal{H}_k = \sum_{j=1}^{k} 1/j$ is the $k$th harmonic number. Let $\Psi(S) = \sum_e \Psi_e(S)$. While this function does not obviously capture any important feature of our game, it has the following nice property.

**Lemma 19.7** *Let $S = (P_1, P_2, \ldots, P_k)$, let $P_i' \neq P_i$ be an alternate path for some player $i$, and define a new strategy vector $S' = (S_{-i}, P_i')$. Then*

$$\Psi(S) - \Psi(S') = u_i(S') - u_i(S).$$

**PROOF** This lemma states that when a player $i$ changes strategies, the corresponding change in $\Psi(\cdot)$ exactly mirrors the change in $i$'s utility. Let $k_e$ be the number of players using $e$ under $S$. For any edge $e$ that appears in both or neither of $P_i$ and $P_i'$, the cost paid by $i$ toward $e$ is the same under $S$ and $S'$. Likewise, $\Psi_e(\cdot)$ has the same value under $S$ and $S'$. For an edge $e$ in $P_i$ but not in $P_i'$, by moving from $S$ to $S'$, $i$ saves (and thus increases her utility by) $c_e/k_e$, which is precisely the decrease in $\Psi_e(\cdot)$. Similarly, for an edge $e$ in $P_i'$ but not in $P_i$, player $i$ incurs a cost of $c_e/(k_e + 1)$ in switching from $S$ to $S'$, which matches the increase in $\Psi_e(\cdot)$. Since $\Psi(\cdot)$ is simply the sum of $\Psi_e(\cdot)$ over all edges, the collective change in player $i$'s utility is exactly the negation of the change in $\Psi(\cdot)$. □

We also note that $\Psi(S)$ is closely related to $cost(S)$, the cost of the network generated by $S$. More precisely, consider any edge $e$ used by $S$. The function $\Psi_e(S)$ is at least $c_e$ (any used edge is selected by at least 1 player), and no more than $\mathcal{H}_k c_e$ (there are only $k$ players). Thus we have

**Lemma 19.8** $\text{cost}(S) \leq \Psi(S) \leq \mathcal{H}_k \text{cost}(S)$.

These two lemmas are used to prove the following two theorems, which will follow from Theorems 19.11, 19.12, and 19.13.

**Theorem 19.9** *Any instance of the global connection game has a pure Nash equilibrium, and best response dynamics always converges.*

**Theorem 19.10** *The price of stability in the global connection game with $k$ players is at most $\mathcal{H}_k$, the $k$th harmonic number.*

Since the proofs of these two results actually apply to a much broader class of games (i.e., potential games), we now introduce these games and prove the corresponding results in this more general context.

### 19.3.2 Potential Games and Congestion Games

For any finite game, an *exact potential function* $\Phi$ is a function that maps every strategy vector $S$ to some real value and satisfies the following condition: If $S = (S_1, S_2, \ldots, S_k)$, $S_i' \neq S_i$ is an alternate strategy for some player $i$, and $S' = (S_{-i}, S_i')$, then $\Phi(S) - \Phi(S') = u_i(S') - u_i(S)$. In other words, if the current game state is $S$, and player $i$ switches from strategy $S_i$ to strategy $S_i'$, then the resulting savings $i$ incurs exactly matches the decrease in the value of the potential function. Thus Lemma 19.7 simply states that $\Psi$ is an exact potential function for the global connection game.

It is not hard to see that a game has at most one potential function, modulo addition by a constant. A game that does possess an exact potential function is called an *exact potential game*. For the remainder of this chapter, we will drop the word "exact" from these terms (see Exercise 19.13 for an inexact notion of a potential function). A surprising number of interesting games turn out to be potential games, and this structure has a number of strong implications for the existence of and convergence to equilibria.

**Theorem 19.11** *Every potential game has at least one pure Nash equilibrium, namely the strategy $S$ that minimizes $\Phi(S)$.*

**PROOF** Let $\Phi$ be a potential function for this game, and let $S$ be a pure strategy vector minimizing $\Phi(S)$. Consider any move by a player $i$ that results in a new strategy vector $S'$. By assumption, $\Phi(S') \geq \Phi(S)$, and by the definition of a potential function, $u_i(S') - u_i(S) = \Phi(S) - \Phi(S')$. Thus $i$'s utility can not increase from this move, and hence $S$ is stable. □

Going one step further, note that any state $S$ with the property that $\Phi$ cannot be decreased by altering any one strategy in $S$ is a Nash equilibrium by the same argument. Furthermore, best response dynamics simulate local search on $\Phi$; improving moves for players decrease the value of the potential function. Together, these observations imply the following result.

**Theorem 19.12** *In any finite potential game, best response dynamics always converge to a Nash equilibrium.*

Note that these two results imply Theorem 19.9.

A less abstract characterization of potential games can be found in a class of games called *congestion games* (Rosenthal, 1973). A congestion game has $k$ players and $n$ resources. Player $i$ has a set $\mathcal{S}_i$ of allowable strategies, each of which specifies a subset of resources. Each resource $j$ has a load-dependent cost function $c_j(x)$, indicating the cost incurred by any player $i$ whose chosen strategy includes resource $j$ if there are $x$ such players in total. The total cost charged to player $i$ who chooses a strategy $S_i$ is simply the sum of the costs incurred from each resource in $S_i$. Thus if the total load on link $j$ is $x_j$, then $i$ pays $\sum_{j \in S_i} c_j(x_j)$. The Global Connection game is clearly a congestion game; edges are resources, $s_i - t_i$ paths are allowable strategies for player $i$, and the cost functions are $c_e(x) = c_e/x$.

Rosenthal (1973) proved that any congestion game is a potential game (see Exercise 19.15). Monderer and Shapley (1996) proved the converse; for any potential game, there is a congestion game with the same potential function.

We now present a generic upper bound on the price of stability for an arbitrary potential game.

### 19.3.3 The Potential Function Method and the Price of Stability

Suppose that we have a potential game $G$ with a potential function $\Phi(S)$ and social cost function $c(S)$. If $\Phi(S)$ and $c(S)$ are similar, then the price of stability must be small. We make this precise in the following theorem.

**Theorem 19.13** *Suppose that we have a potential game with potential function $\Phi$, and assume further that for any outcome $S$, we have*

$$\frac{\text{cost}(S)}{A} \leq \Phi(S) \leq B \cdot \text{cost}(S)$$

*for some constants $A, B > 0$. Then the price of stability is at most $AB$.*

**PROOF** Let $S^N$ be a strategy vector that minimizes $\Phi(S)$. From Theorem 19.11, $S^N$ is a Nash equilibrium. It suffices to show that the actual cost of this solution is not much larger than that of a solution $S^*$ of minimal cost. By assumption, we have that $\frac{\text{cost}(S^N)}{A} \leq \Phi(S^N)$. By the definition of $S^N$, we have that $\Phi(S^N) \leq \Phi(S^*)$. Finally, the second inequality of our assumption implies that $\Phi(S^*) \leq B \cdot \text{cost}(S^*)$. Stringing these inequalities together yields $\text{cost}(S^N) \leq AB \cdot \text{cost}(S^*)$, as desired. □

Note that this result, taken together with Lemma 19.8, directly implies Theorem 19.10. This technique for bounding the price of stability using a potential function is known as the *potential function method*.

In general, outcomes that minimize the potential function may not be the best Nash equilibrium, and thus this bound is not always tight (see Exercise 19.14). However, in the case of the global connection game, we have seen that the price of stability is at least $\mathcal{H}_k$. Thus, for this class of games, the bound given by the potential function method is the best possible.

# Truthful Mechanisms for One-Parameter Agents

Aaron Archer* Éva Tardos†

## Abstract

*In this paper, we show how to design* truthful *(dominant strategy) mechanisms for several combinatorial problems where each agent's secret data is naturally expressed by a single positive real number. The goal of the mechanisms we consider is to allocate* loads *placed on the agents, and an agent's secret data is the cost she incurs per unit load. We give an exact characterization for the algorithms that can be used to design truthful mechanisms for such load balancing problems using appropriate side payments.*

*We use our characterization to design polynomial time truthful mechanisms for several problems in combinatorial optimization to which the celebrated VCG mechanism does not apply. For scheduling related parallel machines ($Q||C_{\max}$), we give a 3-approximation mechanism based on randomized rounding of the optimal fractional solution. This problem is NP-complete, and the standard approximation algorithms (greedy load-balancing or the PTAS) cannot be used in truthful mechanisms. We show our mechanism to be* frugal*, in that the total payment needed is only a logarithmic factor more than the actual costs incurred by the machines, unless one machine dominates the total processing power. We also give truthful mechanisms for maximum flow, $Q||\sum C_j$ (scheduling related machines to minimize the sum of completion times), optimizing an affine function over a fixed set, and special cases of uncapacitated facility location. In addition, for $Q||\sum w_j C_j$ (minimizing the weighted sum of completion times), we prove a lower bound of $\frac{2}{\sqrt{3}}$ for the best approximation ratio achievable by a truthful mechanism.*

## 1 Introduction

In economics, social choice theory addresses the problem of aggregating individuals' preferences to make a group decision. As indicated by Arrow's impossibility theorem for satisfactory voting systems [2], this is a thorny problem. It is further complicated by the possibility that the participants (usually called *players* or *agents*) might try to manipulate the system by misrepresenting their preferences. The field of *mechanism design* recognizes this game theoretic aspect and aims to arrange things so that a rational player will never find it in her self-interest to lie. Mechanisms that do this are called *strategyproof* or *truthful*.

Because of some stifling negative results that apply when the agents' preferences can be arbitrary, it is common to restrict the domain of preferences by assuming *additive separability*. Each agent is assumed to incur some intrinsic benefit or loss (called its *valuation*) depending on the outcome of the mechanism, and this valuation is expressible in some common unit of currency. The mechanism also makes *payments* to the agents in this currency, and each agent aims to maximize the sum of her valuation and payment. The most famous positive result in this area is the Vickrey-Clarke-Groves (VCG) mechanism [31, 4, 12].

Nisan and Ronen [24] considered discrete optimization problems in this game theoretic context, where the correct data is not directly available to the algorithm. Instead, there are several economic agents who each know some of the data and report it to the algorithm, but they might lie. Nisan and Ronen apply this framework to some standard problems in computer science, including shortest paths, minimum spanning trees, and scheduling on unrelated machines. They make a significant conceptual departure from the bulk of the economics literature in that the mechanism's objective function may have nothing to do with social welfare. Here, the agents' preferences are relevant to the goal of the mechanism only because both are tied to the agents' secret data, and because they determine the agents' strategies.

In this paper, we show how to design truthful (dominant strategy) mechanisms for problems where each agent's secret data is naturally expressed by a single positive real number. Our mechanisms allow general objective functions but restrict the form of the valuations. This is in contrast to VCG mechanisms, which allow arbitrary valuations but apply only to the *utilitarian* objective, which is the sum of the agents' valuations. The output of the mechanisms we consider will always define some set of *loads* placed on the

*Operations Research Department, Cornell University, Ithaca, NY 14853. Email: aarcher@orie.cornell.edu. Supported by the Fannie and John Hertz Foundation.

†Computer Science Department, Cornell University, Ithaca, NY 14853. Email: eva@cs.cornell.edu Research supported in part by NSF grant CCR-9700163 and ONR grant N00014-98-1-0589.

agents (e.g. the total size of jobs assigned to a machine, or the total flow through a network link). An agent's secret data will always be the cost she incurs per unit load, and this data will generally also have some physical significance (e.g. processing speed of the machine, or capacity of the link). Her goal is to maximize her profit, which is her payment minus her cost.

In Section 4 we characterize which output functions can be used to design truthful mechanisms. Our mechanisms use side payments to induce the agents to tell the truth. The idea is that if revealing the true parameter would result in an increased load for the agent, we can compensate for this increased load by a payment. However, for some output functions, no side payments can make the resulting mechanism truthful. We prove that the output functions that can be used in truthful mechanisms are exactly those in which the load assigned to an agent decreases monotonically as her announced cost increases, and the payment is given by an explicit formula involving an integral of the load curve. We will use this characterization to design truthful mechanisms for some non-utilitarian objective functions. Our characterization can also be used to design polynomial time truthful approximation mechanisms for utilitarian objective functions, when the VCG mechanism is impractical because the optimal output is hard to compute.

Our main example is the problem of scheduling jobs on related parallel machines to minimize makespan. This problem is commonly denoted $Q||C_{\max}$ in the scheduling literature, and is NP-hard. Each job $j$ has a processing requirement $p_j$ (the amount of work it represents), and each machine $i$ runs at some speed $s_i$. If job $j$ is scheduled on machine $i$, it takes $p_j/s_i$ units of time to complete. The goal is to allocate the jobs to machines so that the last job finishes as soon as possible. Each machine is a distinct economic agent, which incurs a cost proportional to the total time it spends processing, and only the machine $i$ knows the true value of $s_i$.

Our mechanism will ask each agent $i$ to report its speed $s_i$, then allocate the jobs to machines using some algorithm and hand a payment $P_i$ to each machine. The machines know the allocation algorithm and payment scheme in advance, and we assume each machine wants to choose its strategy (i.e. what speed it reports) in order to maximize its profit (the payment it receives minus the cost it incurs from running the jobs assigned to it). Our challenge as the mechanism designer is to find an allocation algorithm and payment scheme that yields a good makespan according to the reported rates and motivates rational agents to report their true rates. Notice that truthful mechanisms are not easy to design even with unlimited computational power. However, we also want to be able to compute the allocation and payments in polynomial time. Since $Q||C_{\max}$ is NP-hard, we will use an approximation algorithm to find the allocation.

In Section 5 we show several applications of our characterization of output functions that can be used to design truthful mechanisms. Designing a truthful mechanism now reduces to designing allocation algorithms with decreasing load curves. Our main application is for the problem $Q||C_{\max}$ discussed above, where we give a randomized 3-approximation mechanism. The problem is NP-complete, but a greedy load-balancing scheme provides a 2-approximation, and there is also a PTAS[1] based on rounding and dynamic programming [16]. However, these types of combinatorial approximation algorithms do not provide monotone work-curves, as the effect of changing the parameter of a single agent is hard to control throughout the algorithm. Our 3-approximation mechanism is based on randomized rounding using an optimal solution for the corresponding fractional problem. We also give an optimal truthful mechanism using unlimited computational power.

We use our characterization to design polynomial time truthful mechanisms for several other combinatorial problems. We design optimal mechanisms for maximum flow, $Q||\sum C_j$ (scheduling related machines to minimize the sum of completion times), optimizing an affine function over a fixed set, and special cases of uncapacitated facility location. We also get a constant approximation mechanism for the general uncapacitated facility location problem, provided the facility costs come from a bounded interval.

In contrast to our optimal truthful mechanism for $Q||\sum C_j$, we prove in Section 6 that no truthful mechanism can achieve an approximation ratio better than $\frac{2}{\sqrt{3}}$ for $Q||\sum w_j C_j$ (scheduling related machines to minimize the weighted sum of completion times).

In the problems discussed above and throughout the paper of Nisan and Ronen [24], the mechanism cares only about the outcome, and the payments exist only to induce truth-telling by the agents. In Section 7, we consider the issue of *frugality* – whether truthful mechanisms can keep the total payment low, by some measure. The shortest path mechanism of Nisan and Ronen behaves poorly in this regard, as some cases force the mechanism to pay $\Omega(n)$ times the cost of the shortest path, even when there is an alternate path of similar cost. Surprisingly, we show in [1] that *every* reasonable mechanism for this problem exhibits this bad behavior. In contrast, we show here that our mechanism for $Q||C_{\max}$ pays out only a logarithmic factor more than the actual costs incurred by the machines, so long as no single machine dominates the total processing power.

## 2 Terminology and notation

We now introduce our notation. There are $m$ agents, represented by the index set $I$. Each agent $i$ has some private

[1] A PTAS is a family of algorithms that, for fixed $\epsilon$ yields a $1+\epsilon$ approximation in polynomial time.

data consisting of a single parameter $t_i \in \mathbb{R}$ that describes the agent. We call this the agent's *true data* or *true value*. In the literature, it is sometimes called the agent's *type*. Only agent $i$ knows $t_i$. Everything else is public knowledge. Each agent will report some value $b_i$ to the mechanism. We call this the agent's *bid*. Let $t$ denote the vector of true values, and $b$ the vector of bids.

There is some allowable set of outcomes $O$ that the mechanism is allowed to choose. The mechanism's *output algorithm* computes a function $o(b) \in O$ according to the agents' bids. The mechanism tries to maximize or minimize some function $g(o, t)$, but of course it does not know $t$ directly. An algorithm that computes an output whose value is guaranteed to be within an $\alpha$ factor of the optimum is called an *$\alpha$-approximation algorithm*. An *$\alpha$-approximation mechanism* is one whose output algorithm is an $\alpha$-approximation.

Each agent $i$ incurs some monetary *cost*, $\text{cost}_i(t_i, o)$, depending on the output and its private data. In order to offset these costs, the mechanism makes a *payment* $P_i(b)$ to agent $i$ (a negative payment means the agent pays money to the mechanism). We assume that each agent $i$ always attempts to maximize her *profit*, $\text{profit}_i(t_i, b) = P_i(b) - \text{cost}_i(t_i, o(b))$. Notice that agent $i$ cares about the other agents' bids only insofar as they influence the outcome and the payment. While $t_i$ is known only to agent $i$, the function $\text{cost}_i$ is public.

In this paper we will assume that the costs have a particularly nice form. Namely, our outcomes $o$ will assign some amount of *load* or *work* $w_i(o)$ to each agent $i$, and we will assume $\text{cost}_i(t_i, o) = t_i w_i(o)$. Thus, agent $i$'s private data $t_i$ measures her cost per unit work.

Let $b_{-i}$ denote the vector of bids, not including agent $i$. We sometimes write $b$ as $(b_{-i}, b_i)$. We say that *truth-telling* is a *dominant strategy* for agent $i$ if bidding $t_i$ always maximizes her profit, regardless of what the other agents bid. That is, $\text{profit}_i(t_i, (b_{-i}, t_i)) \geq \text{profit}_i(t_i, (b_{-i}, b_i))$ for all $b_{-i}$ and $b_i$. We are interested in designing mechanisms such that truth-telling is a dominant strategy for each agent. We call such a mechanism *truthful*.

Formally, the mechanism $\mathcal{M}$ consists of the pair $\mathcal{M} = (o, P)$, where $o$ is the output function and $P$ is the payment *scheme*, i.e. the vector of payment functions $P_i$. We say that an output function *admits* a truthful payment scheme if there exist payments $P$ such that the mechanism $\mathcal{M} = (o, P)$ is truthful. Some output functions admit a truthful payment scheme, and some do not. Our goal is to choose an output function that both admits a truthful payment scheme and achieves (or approximates) the optimal value of $g(o, b)$. In addition, we will usually require that we can compute the output and payments in polynomial time.

One could consider games in which the agents act in a more complicated way than just submitting a bid, instead selecting their courses of action from some broader class of *strategies*. We would then try to design mechanisms where each agent has a dominant strategy. However, it is easy to see that we lose no generality by restricting to mechansims in which agents directly reveal their parameters [22].

## 3 Related work

The economics and game theory literature contains an enormous body of work on mechanism design, also called *implementation theory* or the *theory of incentives*. See [22, ch. 23] or [26, ch. 10] for an introduction to the field, or the surveys [20, 13]. The Gibbard-Satterthwaite theorem [8, 28] is the main negative result, which states that truthful non-dictatorial mechanisms do not exist, when the players' domain of possible preferences is sufficiently rich.

In light of this, it is common to specialize by allowing side payments to the players, and assuming each player tries to maximize the sum of her payment plus her intrinsic valuation of the outcome. The celebrated Vickrey-Clarke-Groves (VCG) mechanism [31, 4, 12] is the main general positive result here. It handles arbitrary valuation functions, but only the utilitarian objective function, which maximizes the sum of the agents' valuations. Nevertheless, this objective function captures some interesting combinatorial problems [24], in addition to the more usual social welfare functions. For example, the shortest path in a graph with respect to edge costs maximizes social welfare because it minimizes the total cost incurred. For the utilitarian objective function, [11] proves that VCG is the only optimal truthful mechanism. In the case of one-parameter agents with some differentiability assumptions, [19] gives a simplified proof.

Much work has addressed computational issues surrounding VCG mechanisms. The main difficulty is that in many settings, the VCG mechanism is NP-hard to compute, since it requires finding an optimal output. One approach is to compute the output using a fast heuristic, and still try to use the VCG payment scheme. Such mechanisms are studied in [18], which gives three properties of the allocation algorithm that will allow the VCG payments to induce truth-telling. However, [25] exhibits a broad class of problems for which no mechanism that uses VCG payments is truthful, if its output algorithm is suboptimal. On the bright side, it also shows that if the mechanism lets the agents suggest ways to improve the output, these mechanisms can be made to satisfy a modified notion of truthfulness. A different approach, taken in [21], is to use a heuristic for the output and use a non-VCG payment scheme to induce truth-telling. They consider a simple type of auction in which computing the socially optimal assignment of goods is hard, and propose a greedy allocation algorithm with a non-VCG payment scheme. Even though their bids are two-dimensional, their problem essentially boils down to a one-parameter problem that is a special case of ours, as

we show in the full version of this paper. While all of these papers depart from the standard VCG mechanism, they still aim to maximize the utilitarian objective, whereas we look at general objective functions.

Both [14] and [3] consider cases where the VCG mechanism can be computed in polynomial time, and address how to speed up this run time. While we are interested in polytime computable mechanisms, we make no attempt to optimize the run time.

Nisan and Ronen [24] applied the mechanism design framework to some standard optimization problems in computer science, suggesting general objective functions. Some subsequent algorithmically-oriented work involves cost-sharing mechanisms for multicast trees [7, 17], auctions for digital goods [10], and the use of auctions to elicit information [30]. The digital goods paper [10] is notable because it explicitly chooses *not* to maximize the social welfare. In their model, the marginal cost of creating an extra copy of the good is negligible, so the socially optimal allocation is to sell this good to everyone, but they do not do this because it generates no revenue. Highlighting the fact that revenue is a major concern, [27] suggests looking at auctions of a single good that do not necessarily maximize the social welfare, and characterizes all truthful mechanisms for this problem. His characterization is a special case of ours, for 0-1 load functions, and it also appears implicitly in [21] and [10]. The paper [30] also ignores the social welfare, instead attempting to compute various functions of the agent's valuations (such as the order statistics) using auctions of minimal communication complexity.

The paper of Nisan and Ronen [24] is closest in spirit to our work. While our main example is the problem $Q||C_{\max}$, scheduling on machines with speeds, their main focus is a similar NP-hard problem $R||C_{\max}$, scheduling on unrelated machines. In that problem, each machine has $n$ items of private data, the amounts of time it would take for it to process each job (so our one-parameter results do not apply). The output is an allocation of jobs to machines, and the cost to a machine equals the total time it spends processing its jobs. Nisan and Ronen provide a simple truthful mechanism (consisting of a separate Vickrey auction for each job) that yields an $m$-approximation. They conjecture that *no* truthful mechanism has a better approximation guarantee, although the best lower bound they prove is 2. With strong additional restrictions on the types of payment schemes allowed, they prove a lower bound of $m$. Note the large gap between the best approximation factors known for a polytime algorithm (2) and for a truthful mechanism ($m$). We have a similar gap for $Q||C_{\max}$ between the PTAS of [16] and the truthful 3-approximation of Theorem 5.4.

The lower bound of 2 for $R||C_{\max}$ stands in contrast to our truthful (non-polytime) mechanism that exactly solves $Q||C_{\max}$. Ronen and Nisan do give a mechanism that solves $R||C_{\max}$ exactly, but only in a much stronger model in which the mechanism is allowed to observe the machines process their jobs and compute the payments *afterwards*, which makes it easy to penalize lying agents.

While revenue is heavily studied in auctions, the important corresponding issue of frugality for task allocation problems is not addressed in [24]. We contribute the first positive frugality result in this area.

## 4 Characterization of truthful mechanisms

Here we completely characterize which output functions do and do not admit truthful payment schemes for mechanism design problems where the cost to agent $i$ is of the form $t_i w_i(o)$, its privately-known cost per unit work times the amount of work assigned. We also characterize the accompanying truthful payment schemes.

In order to motivate our theorem, we first assume all our functions are smooth, and use calculus to derive a formula for the payments and a condition on the output algorithm. Theorem 4.2 below shows that these conditions are actually necessary and sufficient to obtain a truthful mechanism, whether or not the functions are smooth. Let us assume that mechanism $\mathcal{M} = (o, P)$ is truthful and each payment $P_i(b_{-i}, b_i)$ and load $w_i(b_{-i}, b_i)$ is twice differentiable with respect to $b_i$, for all values of $b_{-i}$. We fix some agent $i$ and derive a formula for $P_i$. Fixing the other agents' bids $b_{-i}$, we can consider the payment $P_i$, work $w_i$, and profit to be functions of just agent $i$'s bid $b_i$. Since agent $i$'s profit is always maximized by bidding truthfully, the derivative is zero and the second derivative is non-positive at $t_i$. Since this holds no matter what the value of $t_i$ is, we can integrate to obtain an expression for $P_i$. Specifically, $\text{profit}_i = P_i - t_i w_i$, so the first order condition gives

$$\left[\frac{dP_i(b_i)}{db_i} - t_i \frac{dw_i(b_i)}{db_i}\right]\Bigg|_{b_i = t_i} = 0 \qquad (1)$$

for all values of $t_i$. Integrating by parts gives

$$P_i(b_i) = P_i(0) + b_i w_i(b_i) - \int_0^{b_i} w_i(u)du. \qquad (2)$$

The second order condition says $P_i''(t_i) - t_i w''(t_i) \leq 0$. Using (1), this reduces to $w'(t_i) \leq 0$. Thus, in order to be truthful, the mechanism should have decreasing "work curves" $w_i$, and the payments should be given by (2).

**Definition 4.1** *With the other agents' bids $b_{-i}$ fixed, consider $w_i(b_{-i}, b_i)$ as a single-variable function of $b_i$. We call this the* work curve *or* work profile *for agent $i$. We say the output function $o$ is* decreasing *if each of the associated work curves is decreasing (i.e. $w_i(b_{-i}, b_i)$ is a decreasing function of $b_i$, for all $i$ and $b_{-i}$).*

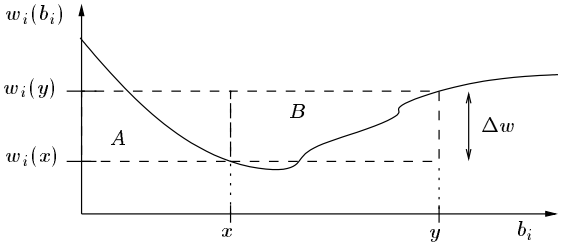



**Figure 1. This graph shows why we cannot allow the work curve to increase.**

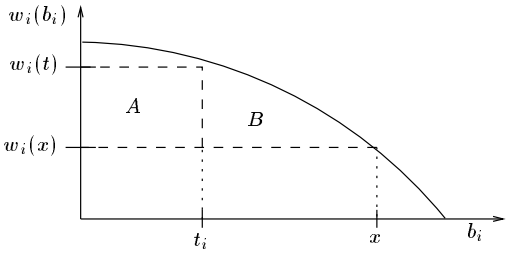



**Figure 2. This picture shows why agent $i$ never gains by overbidding.**

**Theorem 4.2** *The output function $o(b)$ admits a truthful payment scheme if and only if it is decreasing. In this case, the mechanism is truthful if and only if the payments $P_i(b_{-i}, b_i)$ are of the form*

$$h_i(b_{-i}) + b_i w_i(b_{-i}, b_i) - \int_0^{b_i} w_i(b_{-i}, u)du \qquad (3)$$

*where the $h_i$ are arbitrary functions.*

**Proof sketch:** We explain the pictorial proofs of Figures 1 and 2. In Figure 1, $A$ and $B$ denote the areas of the rectangles they label. If $i$'s true value is $y$, she would save cost $A + B$ by bidding $x$. If her true value is $x$, she would incur an extra cost of $A$ by bidding $y$. To motivate truth-telling, the extra payment for bidding $y$ instead of $x$ should be at least $A+B$ and at most $A$, which is impossible since $B > 0$. Therefore, the work curve must decrease monotonically.

In Figure 2, the work curve is decreasing and the payments are given by (3). Geometrically, the payment to $i$ if she bids $x$ is a constant minus the area between the work curve and the horizontal line at height $w_i(x)$. If agent $i$'s true value is $t_i$ and she bids $x > t_i$, then her cost decreases by $A$ from the decreased load, but her payment decreases by $A + B$. Since $B \geq 0$, she never benefits from overbidding. Similarly, she never benefits from underbidding.

To prove that all truthful payment schemes take the form (2) even when $w_i$ is not smooth, we follow essentially the same reasoning as in the calculus derivation. ■

By Theorem 4.2, the only flexibility we have when designing a truthful payment scheme is in the additive constant terms $h_i(b_{-i})$. Consider the profit for a truth-telling agent if we set all of these terms to zero. Her cost is $t_i w_i(t_i)$, which exactly cancels out the second term in the payment formula (3). On net, she incurs a loss equal to the area under her work curve from zero to $t_i$. Since the agents cannot even hope for a profit under this scheme, they presumably would not participate in such a mechanism unless they were coerced. This motivates the following definition.

**Definition 4.3** *A mechanism satisfies the* voluntary participation *condition if agents who bid truthfully never incur a net loss, i.e. $\text{profit}_i(t_i, (b_{-i}, t_i)) \geq 0$ for all agents $i$, true values $t_i$, and other agents' bids $b_{-i}$.*

We want to design mechanisms satisfying voluntary participation. To do this, we need to set $h_i(b_{-i})$ to a constant that is larger than the area under the work curve to the left of $t_i$, no matter what the value of $t_i$ is. If the total area under the work curve is infinite then no such constant exists. If the area is finite then we can set $h_i(b_{-i})$ to be this area, in which case a truth-telling agent $i$ is guaranteed a profit equal to the area under the work curve to the right of $t_i$.

**Theorem 4.4** *A decreasing output function admits a truthful payment scheme satisfying voluntary participation if and only if $\int_0^\infty w_i(b_{-i}, u)du < \infty$ for all $i, b_{-i}$. In this case, we can take the payments to be*

$$P_i(b_{-i}, b_i) = b_i w_i(b_{-i}, b_i) + \int_{b_i}^{\infty} w_i(b_{-i}, u)du. \qquad (4)$$

**Remarks** Our characterization of truthful mechanisms in terms of monotone decreasing outputs should not be confused with other uses of the word "monotone." In particular, a theorem in [6] characterizes truthful mechanisms in terms of "independent person-by-person monotonicity" (IPM). In our context, IPM would be a property of the output and payments together, whereas the beauty of Theorem 4.2 is that it allows us to focus only on the output function.

Our result yields the low-bid Vickrey auction as a special case. Here, the agents are bidding their costs to perform some job, so the load is either 0 or 1. The auction assigns the job to the lowest bidder, and pays her the amount of the second lowest bid. This is the same payment given by (4).

We also note that the obvious analog of Theorem 4.2 holds for the case where $t_i$ denotes agent $i$'s *benefit* per unit load, i.e. where $\text{profit}_i(t_i, b) = P_i(b) + t_i w_i(o)$.

## 5 Designing truthful mechanisms

In this section we utilize Theorem 4.2 to design truthful mechanisms for several problems with one-parameter

agents. Theorem 4.2 neatly separates the problem of designing the output function and the payment scheme – we just have to design an output that assigns decreasing work to agent $i$ as her announced cost per unit work increases. Thus, the challenge is to find an output function $o$ that optimizes (exactly or approximately) our function of interest, $g(o(b), b)$, such that the work curves are all decreasing.

### 5.1 Scheduling to Minimize Makespan

We consider the problem $Q||C_{\max}$, which we mentioned in the introduction. This problem is NP-hard, although there is a PTAS [16]. We are given $n$ jobs and $m$ machines. The jobs represent amounts of work $p_1 \geq \ldots \geq p_n$. The output is an assignment of jobs to machines. Machine $i$ runs at some speed $s_i$, so it must spend $\frac{p_j}{s_i}$ units of time processing each job $j$ assigned to it. The load on machine $i$ is $w_i(b) = \sum p_j$, where the sum runs over jobs $j$ assigned to $i$. Each machine incurs a cost proportional to the time it spends processing its jobs. For simplicity of presentation, we choose our unit of currency so that the constant of proportionality is one.[2] We take the true data to be $t_i = \frac{1}{s_i}$ so that the machines' costs are of the correct form, $\text{cost}_i(t_i, o(b)) = t_i w_i(b)$. The mechanism's goal is to minimize the completion time of the last job on the last machine, i.e. $g(o(b), t) = C_{\max} = \max_i t_i w_i(b)$.

The mechanism design problem for $Q||C_{\max}$ contrasts sharply with the mostly negative results of Nisan and Ronen [24] (see Section 3). We show that truthfulness alone does not prohibit achieving the optimal allocation. Then we give a randomized polytime truthful mechanism that yields a 3-approximation for $C_{\max}$.

**Definition 5.1** *A vector $(w_1, \ldots, w_m)$ is smaller than $(\bar{w}_1, \ldots, \bar{w}_m)$* lexicographically *if, for some $i$, $w_i < \bar{w}_i$ and $w_k = \bar{w}_k$ for all $k < i$.*

**Proposition 5.2** *There is a truthful mechanism (not polytime) that outputs an optimal solution for $Q||C_{\max}$ and satisfies voluntary participation.*

**Proof sketch:** Among the optimal allocations of jobs, our algorithm selects the one in which the load vector $(w_1, \ldots, w_m)$ is lexicographically minimum. Clearly, a machine raising its bid $b_i$ (i.e. announcing it is slower) will not cause the allocation to change unless that machine is the bottleneck. In this case raising $b_i$ will only cause machine $i$ to get less work. Thus, the output function $o$ is decreasing, so by Theorem 4.2 it admits a truthful payment scheme given by (3). As we just argued, $w_i(b_{-i}, \cdot)$ is constant except for jumps at the breakpoints where machine $i$ becomes the bottleneck, so $P_i$ is easily computed. Moreover, a sufficiently slow machine receives no work, so by Theorem 4.4 we can choose $P$ to satisfy voluntary participation. ■

We now move to polytime mechanisms. We cannot simply use any existing approximation algorithm because the work assigned to agent $i$ typically changes in complicated ways as her bid $b_i$ changes. In particular, the PTAS in [16] relies on dynamic programming and rounding the job sizes. If a machine were to announce a slightly slower speed, causing it to receive a different set of jobs, the load could actually increase because of the rounding. The greedy load balancing algorithm also fails to be monotone. Consider scheduling three jobs on two machines of almost equal speeds, where $p_1 = 2$ and $p_2 = p_3 = 1 + \epsilon$. First job 1 is assigned to the faster machine, then jobs 2 and 3 both go on the slower machine, so it gets more work. We need to construct an approximation algorithm with decreasing work curves.

We first note that our problem is equivalent to bin packing with uneven bins, which leads to a lower bound on $C^*_{\max}$, the optimal makespan. This bound is implicit in the $\frac{3}{2}$-approximation algorithm of [29]. Given a guess $T$ at the value of $C^*_{\max}$, we create a bin of size $T/b_i$ for each machine $i$. The size of a machine's bin is the maximum load we can assign to it if the machine is to finish all its jobs by time $T$. Then $T \geq C^*_{\max}$ if and only if there exists an assignment of jobs to bins such that each bin is at least as large as the total size of all the jobs assigned to it. We can relax this requirement by allowing fractional assignments. A *fractional assignment* of jobs to bins consists of a partition of each job $j$ into pieces whose sizes sum to $p_j$ and an assignment of these pieces to the bins. A fractional assignment is *valid* if each bin is at least as large as the total size of all fractional jobs assigned to it, and every bin receiving a piece of a job is large enough to contain that entire job. The smallest $T$ for which there exists a valid fractional assignment is a lower bound on $C^*_{\max}$. We now derive a formula for this lower bound.

If a valid fractional assignment exists, the following greedy algorithm clearly finds it. Number both the bins and jobs from largest to smallest, i.e. $b_1 \leq \ldots \leq b_m$ and $p_1 \geq \ldots \geq p_n$. Assign jobs $1, 2, \ldots, (k-1)$ to bin 1, where $k$ is the first job that would cause the bin to overflow. Then assign to bin 1 a piece of job $k$ exactly as large as the remaining capacity in bin 1. Continue by assigning jobs to bin 2, starting with the rest of job $k$, and so on.

Under what conditions is the greedy assignment valid? For each job $j$, let $i(j)$ denote the last bin that is at least as large as job $j$. The greedy assignment is valid if and only if, for each $j$, the total capacity of the first $i(j)$ bins is at least the total size of the first $j$ jobs. So if the greedy assignment is valid and $i$ is the last bin to which job $j$ is partially as-

[2] Everything that follows still works if we let the constant vary from machine to machine, so long as the constants are known to the mechanism (not part of the private data).

signed, then $T \geq \max\{b_i p_j, \sum_{k=1}^{j} p_k / \sum_{l=1}^{i} 1/b_l\}$. Thus,

$$T_{LB} = \max_j \min_i \max \left\{ b_i p_j, \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i} \frac{1}{b_l}} \right\} \qquad (5)$$

is our lower bound on $C^*_{\max}$. For each job $j$, $i(j)$ is at least as large as the $i$ that attains the min for job $j$ in equation (5), so the first $i(j)$ bins are large enough to accommodate the first $j$ jobs, and the greedy assignment is valid.

**Lemma 5.3** *Sizing the bins according to $T_{LB}$, the greedy algorithm yields a valid fractional assignment such that each bin contains some number of full jobs plus at most two partial jobs.*

Now a natural algorithm suggests itself. Starting with the greedy assignment, round each split job to the faster of its two machines. The load on each machine is now the total size of the jobs fully assigned to that bin in the fractional assignment, plus at most one more job. Since the fractional assignment is valid, the rounded one overflows each bin by at most a factor of 2, so this algorithm is a 2-approximation.

Unfortunately, the algorithm does not yield decreasing work curves. Suppose $b_i p_j$ is the bottleneck term in (5) with $i > 1$, $j < n$, and job $j$ exactly finishing off bin $i$. If $i$ perturbs its bid upwards then $T_{LB}$ increases, so job $j+1$ gets split across bins $i$ and $i+1$ then rounded to $i$, increasing $i$'s load. It seems difficult to overcome this problem with a deterministic algorithm, so we turn to randomized ones.

There is flexibility in defining what it means for a randomized algorithm to be truthful. Here we assume that each agent aims to maximize her *expected* profit. Thus, truth-telling is a dominant strategy for agent $i$ if bidding $t_i$ maximizes her expected profit regardless of what the other agents bid, and a mechanism is truthful if truth-telling is always a dominant strategy for each agent.[3] We now interpret $w_i$ as the *expected* load on agent $i$. By Theorem 4.2, our randomized output algorithm admits a truthful payment scheme if and only if the expected load on $i$ is a decreasing function of $i$'s bid $b_i$. We choose our payment scheme to be given by formula (3) deterministically, but notice that it would be enough for our payments to be random variables whose expectation is given by this formula.

We use randomization to obtain a monotone work curve. Starting with our greedy fractional assignment of jobs to bins, we randomly assign jobs as follows. Job $j$ is assigned to machine $i$ with probability equal to the proportion of $j$ that is fractionally assigned to bin $i$.

**Theorem 5.4** *The randomized allocation described above admits a truthful payment scheme satisfying voluntary participation, and deterministically yields a polytime 3-approximation mechanism for $Q||C_{\max}$.*

**Proof:** Since the fractional assignment is valid and the rounding gives each machine at most 2 extra jobs, each bin is at most triply full. Thus, our allocation is a 3-approximation, no matter how the random choices turn out.

We now show that the expected load on each machine $i$ decreases as $i$ bids higher (i.e. claims to be slower). The expected load on $i$ is precisely the load in the greedy fractional assignment. For full bins this is $T_{LB}/b_i$, for the (at most) one partially full bin it is the work left over from the full ones, and for the empty bins it is 0. Suppose some machine claims she is slower, replacing her bid $b_i$ with $\alpha b_i$, where $\alpha > 1$. This yields a new lower bound $T'_{LB}$ from (5). Clearly $T'_{LB} \geq T_{LB}$, but also $T'_{LB} \leq \alpha T_{LB}$, since shrinking bin $i$ by a factor of $\alpha$ then blowing up *all* bins by $\alpha$ would allow for a valid fractional assignment. Thus, the overall effect of increasing $i$'s bid is to enlarge the other bins while shrinking bin $i$, so the greedy fractional assignment gives $i$ less work. The expected load $w_i(b_{-i}, b_i)$ is a decreasing function of $b_i$, so by Theorem 4.2, we can design a truthful payment scheme. Since machines bidding sufficiently high receive no jobs, we can choose the payments to satisfy voluntary participation, by Theorem 4.4.

To compute the payments, we must compute the function $w_i(b_{-i}, \cdot)$ and the integral $\int_{b_i}^{\infty} w_i(b_{-i}, x)dx$. Let $T_{LB}(x)$ denote our lower bound when agent $i$ bids $x$ and the others bid $b_{-i}$. For small bids (fast speeds) bin $i$ is full, so $w_i(b_{-i}, x) = T_{LB}(x)/x$. For large bids the load is zero. For the interval inbetween, the load is just the leftover work from the larger bins. Thus, we just need to find $T_{LB}(x)$. On different intervals it is either constant, of the form $cx$, or of the form $\frac{c}{d+1/x}$ (where $c$ and $d$ are constants), depending on which term is the bottleneck in formula (5). Breakpoints occur only when $x$ coincides with another agent's bid or when two of the terms inside the braces in (5) (considered as a function of $x$) cross. Thus the number of intervals is polynomial and the integral over each interval is a closed-form expression, so the mechanism is polytime computable.[4] ■

This mechanism has the peculiar feature that we introduced the randomness not to improve the objective function, but to cause the expected load to decrease monotonically as the bid increases.

## 5.2 Affine Functions of the Loads: LP and Uncapacitated Facility Location

Here we consider a general class of problems admitting truthful mechanisms. The main result is the existence of a

[3] A more restrictive definition used in [24] requires truth-telling to be the best strategy, regardless of the outcome of the algorithm's random coin flips.

[4] The closed form expressions giving the payments in the mechanism described above may contain natural logarithms, so our model of computation must allow us to compute these if we wish to obtain a numerical answer for the payment.

truthful mechanism; at this level of generality we cannot say whether we can compute it. Suppose the mechanism wishes to minimize some affine function of the loads

$$g(o, b) = d(o) + \sum_{i \in I} c_i(b_i) w_i(b), \tag{6}$$

where $d$ is some function of the output not depending on the bids, $c_i(b_i)$ is an increasing function for each agent $i$, and the set of allowable outputs $o$ does not depend on the bids. One special case is linear programming, where some of the decision variables are the loads $w_i$, $d(o)$ is the part of the objective depending on the other variables, the cost coefficient on $w_i$ depends on $i$'s bid $b_i$, and the feasible set is given by linear inequalities *not* depending on the bids. Another special case is uncapacitated facility location, where each facility is an agent whose private data is the facility cost, and $d(o)$ is the (publicly known) transportation cost.

Assume that, for every set of bids, there exists an optimal solution. Fix an ordering of the agents.

**Theorem 5.5** *For the problem stated above, if each coefficient $c_i(b_i)$ is strictly increasing in $b_i$, then any optimal output function $o(b)$ admits a truthful payment scheme. Otherwise, any output function that gives an optimal solution whose vector of loads $(w_1, \ldots, w_m)$ is lexicographically minimal admits a truthful payment scheme.*

**Proof sketch:** Raising $i$'s bid only raises the cost of $w_i$, which can only lower the optimal $w_i$, since the set of feasible outputs is fixed. ■

We can apply this theorem to the uncapacitated facility location problem. In the standard problem, we are given a set of facilities and a set of customers. We need to select some subset of the facilities to open, and then assign each customer to be served by some open facility. Each facility has a *facility cost* associated with opening it, and for each customer $j$ and facility $i$, there is a *transportation cost* incurred if customer $j$ is assigned to facility $i$. There are no capacities, so an open facility may serve an arbitrary number of customers. The goal is to minimize the sum of the facility costs and the transportation costs. As explained above, we can apply Theorem 5.5.

**Theorem 5.6** *Any algorithm that solves the uncapacitated facility location problem optimally admits a truthful payment scheme.*

Uncapacitated facility location is NP-hard, so we cannot expect to find a polynomial time algorithm to solve it. However, there are some special cases that can be solved in polynomial time, such as if the facilities and customers lie on a line, circle, or tree. Slight generalizations of these cases are solved in [9]. In these cases, we can also compute each payment easily, as it just involves finding the threshold bid at which a facility would no longer be open (i.e. where $w_i(b_{-i}, \cdot)$ jumps from 1 to 0).

We can use the algorithm of [23] as the basis for a truthful mechanism for facility location in an arbitrary metric space. Since it considers each facility one at a time and opens it with probability inversely proportional to its cost, the load curve decreases with the bid.

**Theorem 5.7** *There is a constant-approximation truthful mechanism for uncapacitated facility location where every customer point is also a potential facility, and all the facility costs are known to lie in an interval $[c_1, c_2]$, where $c_2/c_1$ is bounded by a constant.*

### 5.3 Sum of Completion Times and Max Flow

Here we briefly mention two other problems to which we can apply Theorems 4.2 and 4.4.

The problem of scheduling on related machines to minimize the sum of completion times, commonly denoted $Q||\sum C_j$, can be solved optimally by a simple algorithm [5]. We can prove that this algorithm results in work curves that decrease monotonically to zero. Thus, by Theorems 4.2 and 4.4, we obtain a truthful mechanism that solves $Q||\sum C_j$ exactly and satisfies voluntary participation.

Now we consider the maximum flow problem. We are given a directed graph with source and sink nodes. Each edge $e$ is an agent and has a finite non-zero capacity $c_e$, known only to itself. The mechanism wishes to find a maximum flow from source to sink respecting the capacity constraints on all edges. We assume that each edge incurs a cost equal to the congestion on that edge. That is, if we send $f_e$ units of flow on an edge, that agent incurs a cost of $f_e/c_e$. In order for this problem to fit the form we have been considering (i.e. cost equals $t_e w_e$, the private data times the load), we take the private data to be $t_e = 1/c_e$, and the load on edge $e$ to be $w_e = f_e$. Thus, in truthful mechanisms the flow on edge $e$ must decrease as its announced capacity decreases.

We can guarantee this property by using max flows that are lexicographically minimal (in the sense of Theorem 5.5). We can compute such a flow using $m$ max flows (where $m$ is the number of edges). There is a closed-form expression for the payments in terms of the flow, so we can solve max flow exactly in polynomial time. Unfortunately, the work curves have infinite integrals, so we cannot satisfy voluntary participation. However, we could choose to ignore edges of capacity below $\epsilon/m$, in which case the work curves would drop to zero at that point, so we could satisfy voluntary participation and obtain a flow within an additive $\epsilon$ of optimal.

## 6 Lower bounds

We can use our characterization theorem to prove lower bounds on approximability by truthful mechanisms. In particular we consider scheduling on machines with speeds to minimize the weighted sum of completion times, usually denoted $Q||\sum w_jC_j$. The idea behind the lower bound is that in an optimal allocation, the fast machines should get the important jobs, whereas in a truthful allocation the fast machines should get the bulk of the work. If we arrange the job weights $w_j$ so that these two principles conflict, then the truthful allocation will be suboptimal.

**Theorem 6.1** *No truthful mechanism for $Q||\sum w_jC_j$ can achieve an approximation ratio better than $\frac{2}{\sqrt{3}}$, even on instances with just two jobs and two machines.*

**Proof:** Consider an instance with two jobs and two machines. Job 1 has weight and processing requirement 1, while job 2 has weight $w$ and processing requirement $p$, where $p > 1$. Suppose machine 1 runs at speed 1, and machine 2 runs at speed $s$. In order for our mechanism to be truthful, the load on machine 2 must increase monotonically as its bid decreases (i.e. as its speed increases). To show that any truthful mechanism is suboptimal, we must select $p$ and $w$ so that in the optimal schedule, the load on machine 2 is non-monotone in $s$. To this end, we set $p$ and $w$ such that $pw < 1$.

In the optimal schedule, for small $s$, both jobs will be assigned to machine 1. As we raise $s$, the jobs will eventually be split between machines, the machines will swap jobs, then eventually machine 2 will get both jobs, for large enough $s$. When the jobs are split, the job with larger weight-processing product goes on the faster machine. Since $pw < 1$, job 2 goes to the slower machine. Thus, the load on machine 2 is non-monotone in the optimal assignment. It is easy to check that the optimal assignment is to give both jobs to machine 1 when $s \in (0, \frac{p}{p+1})$, put job 1 on machine 1 and job 2 on machine 2 when $s \in (\frac{p}{p+1}, 1)$, swap the jobs when $s \in (1, 1+\frac{1}{p})$, and put both jobs on machine 2 for $s \in (1+\frac{1}{p}, \infty)$. Whenever both jobs are on the same machine, job 1 goes first (by Smith's rule).

Now we reason about the behavior of any truthful mechanism, as $s$ increases from 0 to $\infty$. If the mechanism is to achieve a finite approximation ratio, then it must assign both jobs to machine 1 when $s \ll 1$ and both jobs to machine 2 when $s \gg 1$. For intermediate values of $s$ it may split the jobs. The key is that if machine 2 gets job 2 for some value of $s$, then it must keep job 2 for all larger values of $s$, since the load may only increase and $p > 1$. The optimal schedule violates this. Because $pw < 1$, we have $\frac{1}{1+w} \in (\frac{p}{p+1}, 1)$. Thus, when $s = \frac{1}{1+w}$, the optimal schedule gives job 2 to machine 2, but when $s = 1 + w$, it gives only job 1 to machine 2. Therefore, the truthful mechanism is suboptimal either when $s = \frac{1}{1+w}$ or when $s = 1 + w$. We discuss the first case. The second is symmetric.

Since we are assuming the truthful mechanism does not split the jobs the optimal way when $s = \frac{1}{1+w}$, the best possible schedule it can use is to split the jobs the other way. Thus, the mechanism gives a schedule with objective function value at least $pw + (1 + w)$, while the optimal schedule has value $1 + pw(1 + w)$. For any fixed $p$, the ratio of these two values is maximized when $w = \frac{-p+\sqrt{2p^2+p}}{p^2+p}$. The ratio increases as $p \downarrow 1$, approaching a limit of $\frac{2}{\sqrt{3}}$. ■

## 7 Frugal mechanisms

To this point, we have viewed payments only as an inducement to the agents to bid truthfully, while the mechanism cared about minimizing some unrelated objective function. We are also interested in mechanisms whose payments are small by some measure. We describe these qualitatively as *frugal*. Since subtracting a constant from the payment functions preserves truthfulness, it is only interesting to consider mechanisms satifying voluntary participation. The total cost incurred by the agents is then a lower bound on the total payment.

We show in [1] that the shortest path mechanism of [24] can be forced to pay $\Omega(n)$ times the cost of the shortest path, even when there is an alternate path of similar cost. Surprisingly, this pitfall is intrinsic to the problem, since we also prove that *every* reasonable mechanism exhibits this bad behavior. In contrast, our 3-approximation algorithm for $Q||C_{\max}$ never pays more than a logarithmic factor more than the expected costs incurred by the machines, provided no single machine dominates the processing power.

**Theorem 7.1** *The payment to each machine $i \geq 2$ is at most $T_{LB}(1+2\ln\frac{x}{b_i})$, where $x = b_1(p_1+\ldots+p_n)/p_n$ and $T_{LB}$ is given by* (5) *. The payment to machine 1 is at most $T_{LB}(\frac{b_2}{b_1} + 2\frac{b_2}{b_1}\ln\frac{x_1}{b_2})$, where $x_1 = b_2(p_1 + \ldots + p_n)/p_n$.*

**Proof sketch:** The payment to machine $i$ consists of two terms (formula (4)). The first term $b_iw_i(b_i)$ exactly compensates machine $i$ for its expected cost, which is $T_{LB}$ for all machines that are full in the greedy fractional assignment of Lemma 5.3. The second term, $\int_{b_i}^{\infty} w_i(u)du$, is the (expected) profit. We always have $w_i(u) \leq T_{LB}(u)/u$, where $T_{LB}(u)$ is the lower bound on the optimal makespan $C^*_{\max}$ computed in formula (5). Equality holds as long as bin $i$ is full in the fractional assignment. But (for $i > 1$) $T_{LB}(u)$ stays approximately constant, since machine $i$ constitutes at most half of the processing power, so decreasing its speed can at most double $T_{LB}(u)$. That is, $T_{LB}(\infty) \leq 2T_{LB}$. Moreover, $w_i(u)$ drops to zero at some point $y$. Thus, the integral is at most $2T_{LB}\ln\frac{y}{b_i}$.

This argument breaks for machine 1 if it is much faster than all the rest combined, since the load on this machine stays nearly constant as its announced speed decreases to that of the second fastest machine. Therefore, our bound on its profit depends on the ratio between the speeds of the fastest two machines.

It then remains to bound $y$. If machine $i > 1$ has speed $1/x = p_n/(b_1(p_1 + \ldots + p_n))$, then placing even the smallest job on $i$ would take longer than processing all jobs on machine 1. Thus, when $i$ bids $x$, it gets no work, so $y \leq x$. Similarly, when machine 1 bids $x_1$, it gets no work. ■

**Corollary 7.2** *If the sizes of all $n$ jobs differ by a factor of at most $r_1$, and the speeds of the two fastest machines differ by a factor of $r_2$, then the payment given by the mechanism exceeds the total expected cost incurred by all the agents by a factor of at most $O(r_2 \ln(r_1 n))$.*

**Proof:** We bound the ratio of payment to expected cost, machine by machine. For each machine $i$ whose bin is full in the greedy fractional assignment, the cost is $T_{LB}$. There is at most one machine with a partially full bin. We cannot bound its ratio, but clearly its payment is no greater than that of the next faster machine. For machine 1, the ratio is at most $r_2(1 + 2\ln(r_1 n))$, and for each other full machine the ratio is at most $1 + \ln(\frac{r_1 n}{r_2})$. ■

**Acknowledgments** We thank David Shmoys for several helpful discussions on scheduling, and the anonymous reviewers for many excellent suggestions.

## References

[1] A. Archer and É. Tardos. Frugal path mechanisms. Unpublished manuscript.

[2] K. Arrow. *Social Choice and Individual Values.* Wiley, 1951.

[3] S. Bikhchandani, S. de Vries, J. Schummer, and R. Vohra. Linear programming and vickrey auctions. Unpublished manuscript, 2001.

[4] E. Clarke. Multipart pricing of public goods. *Public Choice*, 8:17–33, 1971.

[5] R. Conway, W. Maxwell, and L. Miller. *Theory of Scheduling*. Addison-Wesley, 1967.

[6] P. Dasgupta, P. Hammond, and E. Maskin. The implementation of social choice rules: Some general results on incentive compatibility. *Rev. Econ. Stud.*, 46:185–216, 1979.

[7] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of muliticast transmissions. In *STOC*, 218–227, 2000.

[8] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, 1973.

[9] M. Goemans and M. Skutella. Cooperative facility location games. In *SODA*, 76–85, 2000.

[10] A. Goldberg, J. Hartline, and A. Wright. Competitive auctions and digital goods. In *SODA*, 735–744, 2001.

[11] J. Green and J.-J. Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45(2):427–438, 1977.

[12] T. Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

[13] T. Groves. On theories of incentive compatible choice with compensation. In Hildenbrand [15], 1–29.

[14] J. Hershberger and S. Suri. Vickrey pricing in network routing: Fast payment computation. In *FOCS*, 2001.

[15] W. Hildenbrand, ed. *Advances in Economic Theory*. Cambridge UP, 1982.

[16] D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comp.*, 17(3):539–551, 1988.

[17] K. Jain and V. Vazirani. Applications of approximation algorithms to cooperative games. In *STOC*, 364–372, 2001.

[18] N. Kfir-Dahav, D. Monderer, and M. Tennenholtz. Mechanism design for resource bounded agents. In *Intl. Conf. on MultiAgent Systems*, 2000.

[19] J.-J. Laffont and E. Maskin. A differential approach to dominant strategy mechanisms. *Econometrica*, 48(6):1507–1520, 1980.

[20] J.-J. Laffont and E. Maskin. The theory of incentives: An overview. In Hildenbrand [15], 31–94.

[21] D. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. In *ACM Conf. on Electronic Commerce*, 96–102, 1999.

[22] A. Mas-Collel, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford UP, 1995.

[23] A. Meyerson. Online facility location. In *FOCS*, 2001.

[24] N. Nisan and A. Ronen. Algorithmic mechanism design. In *STOC*, 129–140, 1999.

[25] N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. In *ACM Conf. on Electronic Commerce*, 242–252, 2000.

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

[27] A. Ronen. Algorithms for rational agents. In *Conf. on Current Trends in Theory and Practice of Informatics*, 56–70, 2000.

[28] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *J. Econ. Theory*, 10:187–217, 1975.

[29] D. Shmoys, J. Wein, and D. Williamson. Scheduling parallel machines on-line. *SIAM J. Comp.*, 24(6):1313–1331, 1995.

[30] Y. Shoham and M. Tennenholtz. On rational computability and communication complexity. *Games and Econ. Behavior*, 35:197–211, 2001.

[31] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *J. Finance*, 16:8–37, 1961.

# CO7212 Game Theory in Computer Science

Vickrey Auctions, Complexity of Computing Nash Equilibria, and Applications of Game Theory in CS

## Vickrey Auctions

- Bidders submit bids for an item.
- Every bidder i has a private valuation v(i) for the item.
- Payoff to bidder i is v(i)-p if he gets the item and pays p, and 0 if he does not get the item.
- Vickrey Auction: Award item to highest bidder, set the price equal to the second-highest bid.

## Truthfulness of Vickrey Auctions

- Vickrey auctions are truthful: Bidding the true value of v(i) is a (weakly) dominant strategy.
  - If bidder i wins the item and the second-highest bid is x (so bidder i pays x and has payoff v(i)-x):
    - Bidding less than x, he would lose the item and get payoff 0.
    - Bidding any other value greater than x, he will still win the item and pay x, so his payoff is unchanged
  - If bidder i does not win the item, and the highest bid is y > v(i):
    - Bidding any other value less than y, bidder i's payoff is still 0.
    - Bidding a value greater than y, bidder i wins the item for price y and has negative payoff

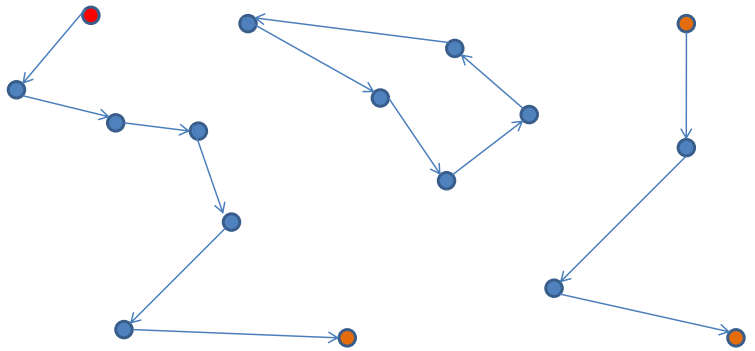
## The Complexity of Finding Equilibria

- How difficult is it to find a Nash equilibrium of a given game?
- Is there an efficient algorithm that can be used to calculate Nash equilibria?
- **Problem NASH**: Given a bimatrix game G, compute any Nash equilibrium of G.

# Lemke-Howson Algorithm

- Best known algorithm for finding Nash equilibria of bimatrix games.
- Works by moving from vertex to vertex of a certain polytope.
- Every vertex has at most one incoming edge and at most one outgoing edge.
- Any vertex, except the start vertex, with only one incident edge, gives a Nash equilibrium.
- Can require exponentially many steps to find a Nash equilibrium.

# Lemke-Howson Illustration



# NP-Completeness

- **NP** = Class of decision problems that can be solved in polynomial time by a non-deterministic Turing machine
- **NP-hard:** A problem P is NP-hard if a polynomial-time algorithm that solves P implies that all problems in NP can be solved in polynomial time.
- **NP-complete:** A problem that is NP-hard and in NP.

# Can NASH be NP-hard?

- Every game has a Nash equilibrium, so the decision problem “Does a given game G have a Nash equilibrium?” cannot be NP-hard. (The answer is always YES.)
- Therefore, NP-hardness is not the right concept for showing that it is hard to compute Nash equilibria.
- But related problems can be proved NP-hard.

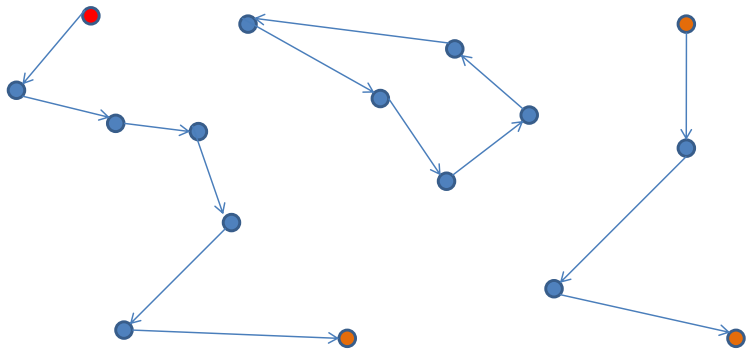
# NP-Complete Problems

- Given a two-player game in strategic form, does it have:
  - At least two Nash equilibria?
  - A NE in which player I has payoff at least a certain amount?
  - A NE in which the two players have total payoff at least a certain amount?
  - A NE in which strategy s has positive probability?
  - A NE in which strategy s has probability 0?
  - Etc.

# The Class PPAD

- Problems that can be represented as follows:
  - Directed graph on an exponentially large set of vertices.
  - Each vertex has indegree and outdegree at most one. Easy to determine neighbours of a vertex.
  - One known source (indegree 0), called the "standard source"
  - Any sink of the graph (outdegree 0), or any source other than the standard source is a solution.

# PPAD Illustration



# PPAD-Hardness of NASH

- The problem NASH is PPAD-complete. (Daskalakis et al, 2006; Goldberg and Papadimitriou, 2006; Chen and Deng, 2006)
- If NASH could be solved in polynomial time, all problems in PPAD could be solved in polynomial time.
- It is considered very unlikely that PPAD-complete problems (including NASH) admit polynomial-time algorithms.

## Zero-Sum Games

- In zero-sum games, Nash equilibria are simply the combinations of any max-min strategy of player I and any min-max strategy of player II.
- Max-min and min-max strategies can be computed efficiently by **linear programming**.
- Therefore, problem NASH is efficiently solvable for zero-sum games.

## Applications of Game Theory in Computer Science

- John von Neumann was instrumental in initiating game theory (1944) and the use of digital computers and their software (algorithms) (1946).
- Especially after the advent of the Internet, topics at the intersection of game theory and computer science are becoming increasingly important.

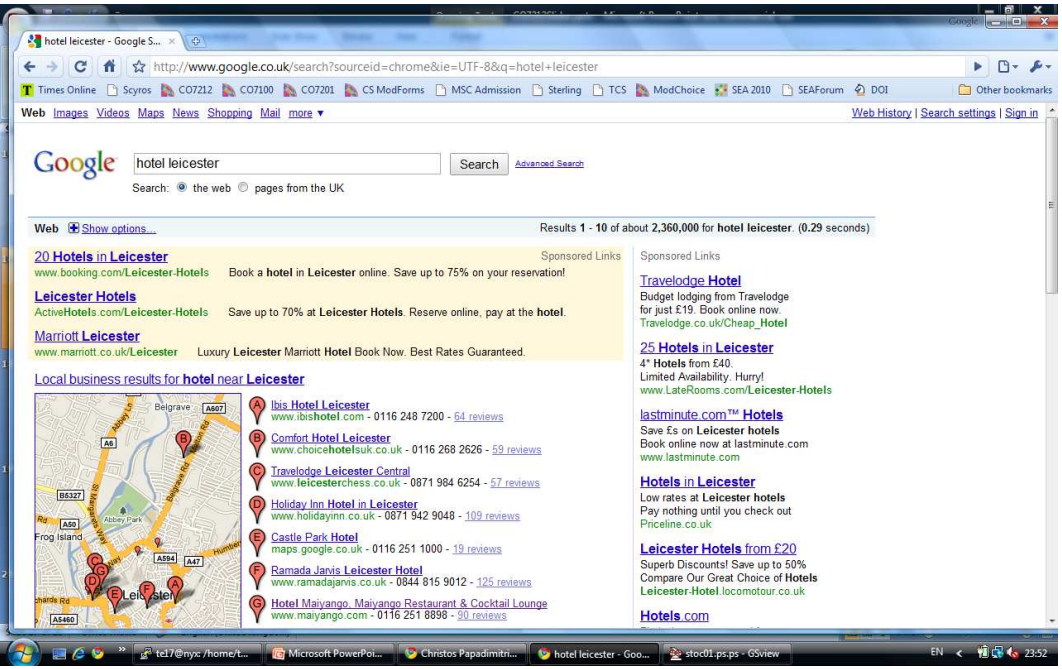
## The Internet

- Large, distributed system without central control.
- Large socio-economic complexity: Built, operated and used by a multitude of diverse economic interests, in varying relationships of collaboration and competition with each other.
- Interaction of many agents (Internet Service Providers (ISPs), Routers, Users), and new types of markets (e.g. e-bay).

## Games in the Internet

- TCP congestion control: The resulting bandwidth sharing is the equilibrium of which game?
- Interactions between ISPs: How are links between ISPs formed in the Internet?

# Google AdWords




# Peer-to-peer Networks

- Users want to download files, but may be reluctant to contribute their bandwidth and files to the network.
- Mechanism design is needed to motivate selfish users to collaborate.

# Wireless Ad-Hoc Networks

- Networks without pre-installed infrastructure
- Every node acts as a router that forwards messages to other nodes
- Selfish nodes have no interest in forwarding traffic for other nodes
- Game theoretic concepts can be used to analyse mechanisms that provide incentives for nodes to cooperate.

# Traffic Planning and Routing

- Satellite navigation systems will be increasingly networked and can then be coordinated centrally.
- This has the potential of improving traffic flow as different drivers can be given individual route recommendations.
- Route recommendations representing a Nash equilibrium are stable (drivers have no incentive to use other routes).

# Games in Theoretical Computer Science

- Some types of games (Ehrenfeucht games, mean payoff games) are used in logic and model checking, for example in checking whether two models are equivalent
- A variant of von Neumann's minimax theorem is used to prove lower bounds on the competitive ratio (worst-case deviation fromt he optimum) of randomised online algorithms.