Algorithmic game theory

Artur Czumaj

człowiek - najlepsza inwestycja

Publikacja współfinansowana ze środków Unii Europejskiej
w ramach Europejskiego Funduszu Społecznego

Note on lecturer

Artur Czumaj is a Professor of Computer Science and Director of the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick. He received his Ph.D. from the University of Paderborn, Germany. Before joining Warwick in 2006, he was with the University of Paderborn and then the New Jersey Institute of Technology. His main research interests are the design of randomized algorithms and their probabilistic analysis, with applications to optimization algorithms, parallel and distributed computing, string matching, and algorithmic game theory. Author of more than 60 publications (according to MR) mostly in Computer Science and related fields.

Preface

Algorithmic Game Theory is a rapidly developing area of research on the boundary of Game Theory, Economics, and Computer Science. Its explosive growth is principally due to the wide spread of computers and the Internet, which requires many classical game theoretical and economic problems to be considered from the computational perspective, to study the consequences of bounded rationality with respect to different notions of efficient computability and the use of limited computational resources. In this series of lectures, we will introduce the main concepts of Algorithmic Game Theory and we will present selected recent advances in this rapidly developing area. Our main focus will be on Computer Science and Combinatorial perspective.

Specific topics discussed during the course include:

- Game models.
- Quality of equilibria: Price of anarchy, price of stability, fairness.
- Algorithmic aspects of game theory in networks and selfish load balancing.
- Finding equilibria.
- Complexity results: Efficient algorithms, NP-completeness of decision problems relating to set of equilibria, PPAD-completeness.
- Algorithmic aspects of auctions.

The text in the following sections gives an informal (and incomplete) overview of the main subjects discussed in the lectures.

Literature. There are several books on Game Theory, though they usually describe in details topics related to the classical Game Theory. Our goal is to focus on more modern approach to the area. Some parts of the lecture are based on the material presented in a recent excellent book:

- *Algorithmic Game Theory*, edited by N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani, and published by Cambridge University Press, New York, NY, 2007.

In this extended abstract we will usually omit the references; detailed references and all appropriated credits will be provided during the lectures.

1 Introduction to Game Theory

Game Theory is a formal study of interactive situations in the framework described as *games*. It typically involves *players*, or participants of the game. Each player has some preferences, some private or public information, a set of strategic actions available, and the influence of the actions on the outcome. The success of players depends not only on their own action but also on actions taken by other players.

1.1 Examples of games

The simplest class of games involves just two players: we all know many games with two players, perfect information, and no-chance moves, specified by certain rules. Standard examples include chess, checkers, go, Othello, dots-and-boxes, etc. In these games both players have perfect information and as such, in principle, every player has one or more optimal strategies that will force the outcome of the game (either one player has a strategy that will force a win or both players can force a draw). While in theory, many such games have dominant strategies for every player, in practice, games like chess and go are well too complex to find optimal strategies, which is also the reason why they're interesting. The relation between the perfect strategy vs "realistic" strategies shows the difference between the perfect model and the *computational model*: to analyze a game, one has to take into considerations its use and availability of resource and the complexity of solving it (finding an optimal strategy). We can also have multiple-player games and games with incomplete information (e.g., poker and bridge).

But the study (and initial motivation) of games goes well beyond popular games, and so, very important classes of games are economic systems, and social systems. Without going much into details, stock markets are an example of a very interesting and in fact very complex game; so is our economy as a whole. The goal of Game Theory is to study various classes of games or systems which involve conflict and cooperation between players who have their individual preferences and relation behavior following strategic thinking.

1.2 A model of a game (one shot simultaneous move game)

In an n-player game with n rational players $[n] = \{1, 2, \ldots, n\}$ each player i has a set S_i of available strategies s_i and a *payoff* function (sometimes called a *utility* function) $\mathsf{payoff}_i : S_1 \times \cdots \times S_n \to \mathbb{R}$. Any vector $s = (s_1, \ldots, s_n)$ with $s_i \in S_i$ is called a *strategy profile*.

In a game, each player i chooses one strategy $s_i \in S_i$ and then she gets paid $\mathsf{payoff}_i(s_1, \ldots, s_n)$. The goal of each player is to maximize her payment.

The crucial feature is that the payoff of each player depends not only on her own strategy but also on the strategies chosen by all other players.

While the presented game is described in the framework of a one shot simultaneous move game, it can be used to describe more general games too.

1.3 Why Algorithmic (or Computational) Game Theory?

Very briefly and informally, Algorithmic Game Theory is the use of models, methods, and tools from game theory to study modern "computing systems" and problems arising therein. While this has been the primary motivation behind the area, now, Algorithmic Game Theory studies also game theoretical question from the "algorithmic" or "computational" perspective. In that way, the two areas, Game Theory and Computer Science, interleave and influence each other in a major way.

Example: Traffic routing in large communication networks. Let us consider a simple example of traffic routing in (large) communication networks (see, e.g., Figure 1 (a)). There are many users who want to send traffic from a source s to a sink node t. Each link in a network has a latency ℓ depending on its load (the latency function should satisfy the condition that more traffic on a link leads to a larger latency. The goal is to set up the system (send all the traffic) such that sum of all travel time (aka total latency) is minimized.

The game theoretical aspect follows from the selfishness of the users: they do not care about other users and thus they have a different goal then the system designer:

- **Users goal:** Minimize travel time for my own traffic.

To see that this framework is nontrivial, consider an example from Figure 1 (a). We have a network with 4 nodes, s, v, u, t. Suppose that one thousand

Rysunek 1: Braess's Paradox

of cars want to drive each morning from s to t. There are two routes available: $s \twoheadrightarrow v \twoheadrightarrow t$ and $s \twoheadrightarrow u \twoheadrightarrow t$. The time needed to drive from s to u or from v to t is 1 hours, independent of traffic (very wide highway), but the road from s to v and from u to t is congestion sensitive: if x fraction of the traffic is using this road than it takes x time to get through.

Let us first consider the scenario when all cars drive $s \twoheadrightarrow v \twoheadrightarrow t$. Then, we have full congestion and hence the total time is 2 hours. If each of the thousand drives will know that the alternate road $s \twoheadrightarrow u \twoheadrightarrow t$ had no congestion, each of them would prefer to use that alternative route and the time needed would be just a bit more than an hour. Therefore, all drives would like to swap to another route on the next day, and then that route would become congested, and so the roads would be congested on alternate days.

What is the best strategy for any driver: it is when we have an equilibrium: half of the cars would take the route $s \twoheadrightarrow v \twoheadrightarrow t$ and another half the route $s \twoheadrightarrow u \twoheadrightarrow t$. The time per car would be 1.5 hours.

Braess observed an interesting paradox for such system. Suppose that the state decided to help the commuters and built a fast highway, a road v to u with no congestion at all, and the time needed to drive from v to u being 0 (see Figure 1(b)). Would it help the drivers? We first observe that we have a new equilibrium: the only equilibrium now would be having all drivers taking $s \twoheadrightarrow v \twoheadrightarrow u \twoheadrightarrow t$, which leads to the driving time of 2 hours. Therefore, despite changing the system in order to "improve" its performance, the obtained network is worse for "selfish" drivers. Of course, an added edge cannot hurt the optimum, a centrally designed optimum routing can simply avoid using the edge.

1.3.1 Computational Aspects of Game Theory

We will study several examples where game theoretical framework is helping computer scientists, IT system designers, and optimization scientists. However, perhaps the main reason of a rapid development of the area of Algorithmic Game Theory is that it makes also a very significant contribution to modern Game Theory (and economics, and social sciences). Computer

science brings a novel perspective on game theoretic models. Questions like:

- How can we compute equilibria?
- How fast can we do it?
- Can we do it in a distributed way (letting player freely react to other players actions) or
- do we need a centralized system that computes an equilibrium and tells everybody what to do?

seemed to be of little important from the classical game theoretic point of view, but they are central if we want to explain the modern nature, and the mechanisms governing our life.

1.4 Strategic and extensive form games

The *strategic form* (also called *normal form*) is the basic type of game studied in (noncooperative) game theory. In strategic form, we list strategies of all players and the outcomes that correspond to all possible combination of choices. An outcome is represented by a separate *payoff* (also called *utility*) for each player, which is a number that measures the quality of the outcome, or in other words, how much the player likes the outcome.

The *extensive form* (called a *game tree*) is a complete description of how the game is played over time. This includes the order in which players take actions, the information that players have at the time they must take those actions, and the times at which any uncertainty in the situation is resolved. The extensive form is more detailed than the strategic form of a game. A game in extensive form may be analyzed directly, or can be converted into an equivalent strategic form.

1.4.1 Prisoner's Dilemma and dominance

The Prisoner's Dilemma is a two-players game. Consider two prisoners on a trial for a major crime. The authorities can only prove their involvement in minor offenses, for which both would have to serve two years in prison. If, however, one of them confesses, this persons sentence will be reduced to one year whereas the other prisoner will get a sentence of 25 years. I If both of them confess, non of them is needed as a witness against the other and both will be convicted to ten years. We can represent this game in the strategic form as follows:

I \ II	confess	defect
confess	-10, -10	-25, -1
defect	-1, -25	-2, -2

The Prisoner's Dilemma has some special properties: each player has a *dominant strategy*, and in particular, each player can determine her best action without reasoning about the other player's action. In this case, independently of the other prisoner's choice, it is always better to confess.

Definition 1.1 (Dominance) *Let s and t be two strategies of player i. s* dominates *t if for any fixed actions of the other players, the payoff to player i when using s is higher than when using t. We say then that t* is dominated *(by s).*

1.4.2 The battle of sexes

Alice and Bob want to meat for lunch. Alice prefers Pizza Hut, and Bob prefers Burger King. Given the strategic form of the game below, how likely is that they will meet?

Alice \ Bob	Pizza Hut	Burger King
Pizza Hut	2, 1	0, 0
Burger King	0, 0	1, 2

1.4.3 The tragedy of the commons

(An example with IT flavor) n players want to access a shared recourse, for example, send information along a shared channel of capacity 1. Player i

can choose to send x_i units of flow. The quality of the channel deteriorates with the total bandwidth: if the total bandwidth is greater than 1 then the connection breakers down and information is sent. The goal of each player is to maximize her share of the bandwidth. If the connection fails then no player gets any profit; otherwise, player's i profit is $x_i(1 - \sum_j x_j)$.

The best "selfish" response of every player is to set $x_i = \frac{1}{n+1}$, in which case each player's profit is $\frac{1}{(n+1)^2}$. However, if each player set $x_i = \frac{1}{2n}$, then her profit would be $\frac{1}{4n}$.

1.5 The notion of equilibria: stable state

As we could have seen in earlier examples, we often want to study the situation when all players play rationally (are selfish), and with that, they reach a steady state, or an equilibrium state. This notion will be defined more formally later.

2 Games

Definition 2.1 (Mixed strategies) *A mixed strategy of a player i with strategy set S_i is a probability distribution over S_i.*

We denote such a mixed strategy as an $|S_i|$-dimensional vector of which each entry is the probability that the corresponding strategy from S_i is chosen.

Let $\Delta^{|S_i|}$ denote the set of mixed strategies for player i.

We can now introduce the concept of *mixed Nash equlibria*, which is similar to that of pure equilbria but makes use of mixed strategies instead of pure ones. Roughly speaking, a profile of mixed strategies is called a (mixed) Nash equilibrium if no player can gain on average by unilateral deviation from her strategy.

$\sigma = (\sigma_1, \ldots, \sigma_n)$ is the mixed strategy profile in which player i chooses mixed strategy $\sigma_i \in \Delta^{|S_i|}$. The expected payoff of player i is denoted by $p_i(\sigma)$. Let σ_{-i} denote the strategy profile σ without player i. Thus, $\sigma = (\sigma_{-i}, \sigma_i)$.

Definition 2.2 ((Mixed) Nash equilibrium) *A strategy profile σ^* is a Nash equilibrium if for every $i \in [n]$ and every $\sigma_i \in \Delta^{|S_i|}$ holds*

$$p_i(\sigma^*_{-i}, \sigma^*_i) \geqslant p_i(\sigma^*_{-i}, \sigma_i) \ .$$

An Nash equilibrium is called a *pure Nash equilibrium* if for every $i \in [n]$, player i chooses one strategy $s_i \in S_i$ (not probabilistically).

Nash equilibrium, less formally: A Nash equilibrium is a state in which each player is assumed to know the equilibrium strategies of the other players, and no player has any incentive to change her own strategy unilaterally. The Nash equilibrium is mixed if some players make their decisions at random, and it is pure when all decisions are deterministic.

3 Two-player zero-sum games

We consider the following setting:

- There are two players, I and II, with pure strategy sets $S_1 = [m_1]$ and $S_2 = [m_2]$.
- For any two strategies $s_1 \in S_1$ and $s_2 \in S_2$, we have $p_1(s_1, s_2) = -p_2(s_1, s_2)$.
- The payoff can be written as an $m_1 \times m_2$ matrix

$$\mathbf{A} = \begin{pmatrix} p_1(1,1) & \dots & p_1(1,m_2) \\ \vdots & \ddots & \vdots \\ p_1(m_1,1) & \dots & p_1(m_1,m_2) \end{pmatrix}$$

  for player I, and $\mathbf{B} = -\mathbf{A}$ for player II.
- Let x be a mixed strategy for player I and y be a mixed strategy for player II. Then, player I has expected payoff $x^T \mathbf{A} y$ and player II has expected payoff $x^T \mathbf{B} y = -x^T \mathbf{A} y$.

3.1 Structure of Nash equilibria in two-players zero-sum games

A possible approach of player I is to try to maximize her guaranteed expected payoff. For this player I chooses a mixed strategy x that gives the highest expected payoff under the assumption that player II chooses a response that is the worst for player I. This worst-case assumption is realistic for two-player zero-sum games since the players' interests are diametrically opposed.

Definition 3.1 (Maximin and minimax strategies)
A strategy $\mathbf{x} \in \Delta^{m_1}$ *is called a maximin-strategy if*

$$\mathbf{x} \in \arg\max_{\mathbf{x}' \in \Delta^{m_1}} \min_{j \in [m_2]} (\mathbf{A}^T x')_j \ .$$

A strategy $\mathbf{y} \in \Delta^{m_2}$ *is called a minimax-strategy if*

$$\mathbf{y} \in \arg\max_{\mathbf{y}' \in \Delta^{m_2}} \min_{i \in [m_1]} (\mathbf{A}^T y')_i \ .$$

We call a pair of maximin- and minimax-strategies a minimax-equilibrium.

Theorem 3.1 *For two-player zero-sum games:*

- *A pair of maximin- and minimax-strategies forms a Nash equilibrium.*
- *All Nash equilibria result in the same expected payoff. We call the expected payoff of player I the value of the game.*
- *A Nash equilibrium can be computed in polynomial time (in the number of pure strategies).*

4 Equilibria in general games (Nash Theorem)

We were able to show that every two-player zero-sum games admits a Nash equilibrium and, what is perhaps even more astonishing, we can compute a Nash equilibrium in polynomial time. In the following we will study the existence of Nash equilibria in general games. We first observe that our approach for two-player zero-sum games does not work here, since in general, it is too pessimistic for a player to assume that the other players will act in such away that her payoff is minimized.

It turns out that, even for general games, Nash equilibria do always exist but in order to prove this we first need to learn some seemingly unrelated things.

Consider a graph consisting of a large triangle that is subdivided into an arbitrary number smaller triangles in an arbitrary way. All three corners of the large triangle are colored with three different colors red, green, and blue. A coloring of the remaining vertices is said to be valid if no vertex on one of the large three outer edges is colored with the color of the opposite corner of the large triangle.

Lemma 4.1 (Sperner's Lemma) *Each valid coloring of the graph results in at least one small triangle whose three corners are colored in three different colors. In fact, the number of such triangles is odd.*

We can now use Sperner's lemma to prove Brouwer's fixed point theorem. While at the first glace, the statement of Brouwer's fixed point theorem seems very different from Sperner's lemma, one can show that both results are essentially equivalent.

Theorem 4.1 (Brouwer's fixed point theorem) *Let f be a continuous function from a convex compact set to itself. Then, f has a fixed point, i.e., it exists an x such that $f(x) = x$.*

Nash used these results to prove one of the most fundamental results in Game Theory.

Theorem 4.2 (Nash 1951) *Every game with a finite number of players and a finite number of strategies has a Nash equilibrium.*

Let us mention that Theorem 4.2 talks about *mixed Nash equilibria*; there are many examples of games with no pure Nash equilibria.

5 The complexity of finding Nash equilibria

The classical game theory puts Nash equilibria as its central concept and in many situations the study are done under the assumption that the system under consideration is in an equilibrium. This leads to a fundamental question: how long does it take until selfish (economically driven) agents converge to an equilibrium? While the empirical evidence shows that in some scenario this happens rather quickly, there is no real indication that this always have to be the case. This leads to a very important question from the complexity point of view: what is the complexity of computing a Nash equilibrium. We have seen that it is possible to compute Nash equilibria for two-player zero-sum games in polynomial time. Algorithms to compute Nash equilibria in general games are known, but they are not known to be efficient. The complexity of computing a Nash equilibria has been a long standing open question that has been solved only very recently. In two recent papers, Daskalakis, Goldberg, and Papadimitriou, and Chen and Deng (2006) provided evidence that there are games in which convergence to an equilibrium (mixed Nash equilibrium) takes prohibitively long.

Traditionally, the study of the computational complexity of problems leads to the classification of problems into two classes: those that have polynomial time algorithms (are in the class $\mathcal{P}$) and those that are $\mathcal{NP}$-hard. However, the concept of $\mathcal{NP}$-hardness cannot be applied to the rare problems where "every instance has a solution." This is exactly the case for the study of the complexity of computing Nash equilibria in games, because Nash Theorem 4.2 ensures that every game has a mixed equilibrium. The central result of this section is a result that finding a Nash equilibrium is complete for a class of problems called PPAD, containing several other known hard problems; all problems in PPAD share the same style of proof that every instance has a solution.

5.1 PPAD

PPAD is best described by giving a representative (which means *complete*) problem of the class. We consider one such a representative problem called EndOfTheLine. We are given a directed graph $G = (V, E)$, in which each node has at most one predecessor and at most one successor. The graph is encoded by two circuits P and S with k input and k output bits. The size of the circuits is polynomial in k. The nodes of the graph are numbered from 1 to

$2^k - 1$. Intuitively, P computes the predecessor and S computes the successor of a node. More precisely, the graph contains an edge (v, w) if and only if $S(v) = w$, $P(w) = v$, and $w \neq \langle 0 \dots 0 \rangle$. The problem EndOfTheLine is to find a node other than $\langle 0 \dots 0 \rangle$ that has no predecessor or no successor.

We note that a simple parity argument can be used to show that the graph always has at least one node (other than $\langle 0 \dots 0 \rangle$) that has no predecessor or no successor. However, the difficulty of the problem is to find one such node.

A naive approach would be to follow the path originating from $\langle 0 \dots 0 \rangle$ or to just test all the nodes in the graph. However, this cannot be done efficiently, that is, in polynomial time, since the graph has up to 2^k nodes and this is exponential in the size of the input. It is not known, whether EndOfTheLine can always be solved in polynomial time. (And in fact, it is conjectured that this is impossible.)

We can replace the two circuits in this description by a Turing machine and define the class PPAD as follows.

Definition 5.1 (PPAD) *We call a problem a search problem if each instance x has a search space S_x of bit-strings of polynomial length (that is, polynomial in $|x|$) and the set of valid solutions is given by a non-empty subset Q_x of S_x.*

A search problem Π is contained in PPAD *if there is a polynomial time Turing machine M_π with the following properties. For $c \in S_x$, the Turing machine outputs a pair (c', c'') of strings from the search space S_x, i.e., $M_\Pi(x, c) = (c', c'')$ with $c', c'' \in S_x$. However, c'' is not allowed to be the zero string $\langle 0 \dots 0 \rangle$. This Turing machine encodes a directed graph whose nodes are the strings in S_x, in the way explained above. The set of valid solutions Q_x has to be equal to the set of nodes that do not have a predecessor or successor, except $\langle 0 \dots 0 \rangle$.*

We also have a concept of reduction for search problems. A reduction from one search problem Π_1 to another Π_2 is defined as a pair of polynomial time computable functions (f, g), where f maps instances of Π_1 to instances of Π_2 and g maps solutions for Π_2 to solutions for Π_1. More precisely, for $x \in \Pi_1$ we have $f(x) \in \Pi_2$ and for any solution y of Π_2 for input $f(x), g(x, y)$ is a solution of Π_1 for input x.

PPAD the minimum set of search problems such that all problems defined above are contained in the set and the set is closed under reduction.

Theorem 5.1 (Daskalakis, Goldberg & Papadimitriou, and Chen & Deng (2006)) *The problem of finding a Nash equilibrium is PPAD-complete. This holds even for two-player games.*

6 Pure Nash equilibria

The fundamental result by Nash states that every finite, noncooperative strategic game of two or more players has a (mixed) Nash equilibrium. However, in many situations, we are more concerned about pure Nash equilibria.

There are two central questions regarding pure Nash equilibria in games:

- does a given game have a pure Nash equilibrium, and
- whether it is computationally feasible to determine if a given game has a pure Nash equilibrium, and if it does, to find one.

In the most basic setting, we consider an n-player game defined by n utility functions, each of them given as a table of size m^n of numbers (where we assume for simplicity that each player has a strategy space of size m.) We first observe that one can easily find a pure Nash equilibrium (if there is one) in such a setting by going over all possible m^n possible strategy profiles and then checking for each of them whether some player can gain by deviating we get an efficient algorithm.

Finding a pure Nash equilibrium:

- **for every** strategy profile s **do**:
  - pure = TRUE
  - **for every** player $i = 1 \dots n$ **do**
    - ◇ **for every** strategy $s^{\langle i \rangle}$ of player i
      - ▷ **if** $\mathsf{payoff}_i(s^{\langle i \rangle}, s_{-i}) < \mathsf{payoff}_i(s)$ **then** pure = FALSE
  - **if** pure **then**
    - ◇ report s as a pure Nash equilibrium
- **if** no pure Nash equilibrium has been found so far **then**
  - report that no pure Nash equilibrium exists

It is not difficult to see that the obtained running time is $\mathcal{O}(nm^{n+1})$, where we measure the running time by the number of queries to the utility functions of the players.

This result does not look very impressive though, because the running time is "exponential"; still, assuming that the input is given as n tables of size m^n of numbers each, the running time of this algorithm is polynomial. But can we do better? In many cases, we really have a "very large" game, and we would like to find an equilibrium in time which is significantly sublinear in the size of the game's utility tables. Of course, we have to ask ourselves what do we mean by a sublinear-time.

To cope with this question, we will consider two basic models: the *black box model* and the *Turing-Machine complexity model*.

6.1 Finding pure Nash equilibria in the Black box model

In the black box model, we assume that the utility tables are given by a "black box" which can answer queries of a certain type (e.g., given a strategy profile s, can return the value of $u_i(s)$ as a single atomic operation). The set of queries that the black box supports will of course determine what can be done efficiently in this model.

We will focus on the model in which we get n "black boxes," one for each player, that can answer utility queries: given a strategy profile s return $u_i(s)$. In fact, we also could consider best-reply queries: given a strategy profile of the others s_{-i}, return the best-reply s_i of player i. (Here we would need to specify how ties are handled.) Since we will concentrate on cases where m is not too large and we allow the running time to be polynomial in m, the fact that a best-reply query can easily be simulated by m utility queries to all possible values of s_i, we observe that adding a best-reply query does not significantly change the power of the model.

The algorithm described in the previous section shows that one can determine if a given game has a pure Nash equilibrium in the black box model with $\mathcal{O}(nm^{n+1})$ queries; if the game has a pure Nash equilibrium then the algorithm will also find one within the same complexity. This of course rises the question if it's possible to obtain a better algorithm, and in particular, if it's possible to obtain the query complexity that is polynomial in both n and m. For the case of general games it is easy to show that the answer is negative.

Theorem 6.1 *There are games with n players, each having m strategies, such that every algorithm that finds a pure Nash equilibrium requires $\Omega(m^n)$ queries in the black box model.*

6.2 Congestion Games and Potential Games

Now, we shall discuss two interesting classes of games which always posses at least one pure Nash equilibrium, and in fact, the players can reach a pure Nash equilibrium using a very simple "best-response" dynamics, regardless of their opening strategies. Moreover, despite being of different flavour, these two classes are in some sense equivalent, and we will sketch a proof of one side of this equivalence.

6.2.1 Potential Games

Definition 6.1 *A game G is called a* potential game *(exact potential game) if there is a potential function $\Phi : S \to \mathbb{R}$ such that for every player i, for every strategy profile $s = (s_i, s_{-i})$, and for every strategy s'_i, we have the following identity:*

$$\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) \;=\; u_i(s_i, s_{-i}) - u_i(s'_i, s_{-i}) \;.$$

In words, potential games are the games for which there is a function that maps strategy of profiles to real numbers, such that when player i deviates from strategy s_i to s'_i, the change in the players utility function is exactly the same as the change in the potential function.

Potential games have been introduced largely because of the following result.

Theorem 6.2 *Every potential game has at least one pure Nash equilibrium, namely the strategy that maximizes Φ.*

6.2.2 Best-response dynamics

The proof of Theorem 6.2 shows immediately that the best-response dynamics will find a pure Nash equilibrium:

Theorem 6.3 *In any finite potential game, best-response dynamics always converge to a pure Nash equilibrium.*

While this is a very powerful and in fact, a very useful result, it does not say much about the number of steps needed to find a pure Nash equilibrium. Indeed, if we consider the model in which we count the number of queries to the utility functions of the players (the black box model), then all what the result above can ensure is that the total running time of this algorithm is $\mathcal{O}(nm^n)$ operations. Later we will discuss if this bound is tight or if one can find a pure Nash equilibrium in potential games in polynomial time.

6.2.3 Congestion Games

Let us now introduce another class of games which, as we will argue, is very closely related to potential games.

Definition 6.2 *A* congestion game *is defined by a set N of n players $N = \{1, \ldots, n\}$ and a finite set S of resources. Each player i has a set S_i of allowable strategies, with $S_i \subseteq S$. For each resource $j \in S$, we have 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 cost of player i is then defined as $\sum_{j \in S_i} c_j(x_j)$, where x_j is the number of players using resource j.*

It is not difficult to see that nonatomic network routing games, which we discussed in details earlier, are a special case of congestion game.

6.2.4 Congestion Games and Potential Games

The following theorem shows that congestion games posses all properties of potential games.

Theorem 6.4 (Rosenthal'73) *Every congestion game is a potential game.*

Theorem 6.4 immediately shows that congestion games have pure Nash equilibria that can be found using best-response dynamics.

6.2.5 Finding pure Nash equilibria in potential games in the black box model

So far we have seen several desirable properties of potential games and it might be natural to hope that since each potential game has a pure Nash equilibrium, finding one is computationally hard.

Theorem 6.5 *Let G be a potential game with n players, each having 2 strategies. Every algorithm that finds a pure Nash equilibrium for G requires at least $\Omega(2^n/\sqrt{n})$ value queries in the black box model.*

7 Routing in a flow model

We consider a directed network $\mathcal{N} = (V, E)$ with node set V, edge set E, and k source-destination node pairs $\{s_1, t_1\}, \{s_2, t_s\}, \ldots, \{s_k, t_k\}$. We denote the set of (simple) s_i–t_i paths by $\mathcal{P}_i$ and define $\mathcal{P} = \bigcup_{i=1}^{k} \mathcal{P}_i$. A *flow* is a function $f : \mathcal{P} \to \mathbb{R}_{\geqslant 0}$, and for a fixed flow f and for any $e \in E$, we define $f_e = \sum_{\pi \in \mathcal{P}: e \in \mathcal{P}} f_\pi$. With each pair $\{s_i, t_i\}$, we associate a positive *traffic rate* r_i that denotes the amount of flow to be sent from node s_i to node t_i. We say a flow f is *feasible* if for all $i \in [k]$, $\sum_{\pi \in \mathcal{P}_i} f_\pi = r_i$.

We assume that each edge $e \in E$ is given a load-dependent *latency function* denoted by $\ell_e(\cdot)$. We will later make some more precise assumptions about the latency functions, but for now, one can assume that every ℓ_e is nonnegative, differentiable, and nondecreasing. The latency of a path π with respect to a flow f is defined as the sum of the latencies of the edges in the path, denoted by $\ell_\pi(f)$, $\ell_\pi(f) = \sum_{e \in \pi} \ell_e(f_e)$. Finally, we define the cost $C(f)$ of a flow f in $\mathcal{N}$ as the total latency incurred by f, that is, $C(f) = \sum_{\pi \in P} \ell_\pi(f) f_\pi$. Notice also that by summing over the edges in a path π and reversing the order of summation, we may also write $C(f) = \sum_{e \in E} \ell_e(f_e) f_e$. We call the triple $(\mathcal{N}, r, \ell)$ an *instance*.

7.1 Flows at Nash equilibria

Since we assume that all agents carry a negligible fraction of the overall traffic, we can informally assume that each agent chooses a single path of the network. Therefore, we focus our discussion on pure strategies and define Nash equilibria for *pure strategies* only.

A flow f feasible for instance $(\mathcal{N}, r, \ell)$ is at *Nash equilibrium* if for all $i \in [k]$, $\pi_1, \pi_2 \in \mathcal{P}_i$ with $f_{\pi_1} > 0$, and $\delta \in (0, f_{\pi_1})$, we have $\ell_{\pi_1}(f) \leqslant \ell_{\pi_2}(\tilde{f})$, where

$$\tilde{f}_\pi = \begin{cases} f_\pi - \delta & \text{if } \pi = \pi_1 \\ f_\pi + \delta & \text{if } \pi = \pi_2 \\ f_\pi & \text{if } \pi \notin \{\pi_1, \pi_2\} \end{cases} .$$

In other words, a flow is at Nash equilibrium if no network user has an incentive to switch paths.

If we let δ tend to 0, continuity and monotonicity of the edge latency function yields the following useful characterization of a flow at Nash equilibrium, called in this context also a *Wardrop equilibrium*.

Theorem 7.1 *A flow f feasible for instance $(\mathcal{N}, r, \ell)$ is at Nash equilibrium if and only if for every $i \in [k]$, and $\pi_1, \pi_2 \in \mathcal{P}_i$ with $f_{\pi_1} > 0$, it holds that $\ell_{\pi_1}(f) \leqslant \ell_{\pi_2}(f)$.*

An important consequence of this characterization is that if f is Nash equilibrium then all s_i–t_i paths to which f assigns a positive amount of flow have *equal latency*, that we will denote by $L_i(f)$. Therefore, we can express the cost of a flow f at Nash equilibrium in the following useful form.

Theorem 7.2 *If f is a flow at Nash equilibrium for instance $(\mathcal{N}, r, \ell)$, then $C(f) = \sum_{i=1}^{k} L_i(f)\, r_i$.*

7.2 Optimal flows vs flows at Nash equilibrium

Beckmann et al. (1956) noticed a similarity between the characterizations of optimal solutions and of flows at Nash equilibrium that leads to a very elegant characterization of an optimal flow as a flow at Nash equilibrium with a different set of edge latency functions. For any edge $e \in E$, let us denote by ℓ_e^* the marginal cost of increasing flow on edge e,

$$\ell_e^*(x) = \tfrac{d}{dy}(y \cdot \ell_e(y))(x) = \ell_e(x) + x \cdot \ell_e'(x) \ .$$

Then, one can show the following theorem:

Theorem 7.3 *Let $(\mathcal{N}, r, \ell)$ be an instance in which $x \cdot \ell_e(x)$ is a convex function for each edge and let ℓ^* be the marginal cost function as defined above. Then a flow f feasible for $(\mathcal{N}, r, \ell)$ is optimal if and only if f is at Nash equilibrium for the instance $(\mathcal{N}, r, \ell^*)$.*

This precise and simple characterization of optimal flows plays a crucial role in the analysis of the flow model and in the understanding of that model. From now on, we will denote the optimal flow by f^* and the marginal cost function by ℓ^*.

We notice that the convexity of $x \cdot \ell_e(x)$ is necessary in Theorem 7.3. Therefore, in this survey we shall mostly focus on standard latency functions, where a latency function ℓ is *standard* if it is differentiable and if $x \cdot \ell_e(x)$ is convex on $[0, \infty)$. Many but not all of interesting latency functions are standard. Typical examples of standard latency functions include differentiable convex functions, and examples of non-standard latency functions are differentiable approximations of step functions.

An important consequence of using standard latency functions is that in network with standard latency functions the optimal flow (up to an arbitrarily small additive error term) can be found in polynomial time using convex programming.

(a) Latency functions (b) Marginal cost functions

Rysunek 2: Pigou's example with the source node s, the sink node t, and the latency functions $\ell_1(x) = 1$ and $\ell_2(x) = x^p$.

Theorem 7.4 *Any instance $(\mathcal{N}, r, \ell)$ with continuous, nondecreasing latency functions admits a feasible flow at Nash equilibrium. Moreover, if f and $\tilde{f}$ are flows at Nash equilibrium, then $C(f) = C(\tilde{f})$.*

We notice that actually, any such an instance admits a feasible acyclic flow at Nash equilibrium.

Finally, following our definition from the previous sections, we define a *price of anarchy* as the ratio between the cost of a flow at Nash equilibrium in an instance $(\mathcal{N}, r, \ell)$ over the minimum cost of the optimal flow for that instance, $\frac{C(f)}{C(f^*)}$, where f is a flow at Nash equilibrium and f^* denote the optimal flow. Notice that by Theorem 7.4, all Nash equilibria have the same cost and thus for any instance $(\mathcal{N}, r, \ell)$ the price of anarchy is the same for all flows at Nash equilibrium.

7.3 Price of anarchy

In this section, we show that without making any assumption about the latency functions (besides being standard) the price of anarchy can be arbitrary large.

Let us call a class $\mathcal{L}$ of latency functions *standard* if it contains a nonzero function and each function $\ell \in \mathcal{L}$ is standard. By analyzing the scenario of the Pigou's network (see Figure 2) with various standard latency functions, we observe that in many cases the price of anarchy for that example leads to the worst possible price of anarchy. This motivated us to define the anarchy value to be the worst-case price of anarchy in Pigou's example from Figure 2 that use the latency function ℓ on the 2nd link. More formally, if ℓ is a nonzero standard latency function, then we define the *anarchy value* $\alpha(\ell)$ of ℓ as

$$\alpha(\ell) \;=\; \sup_{r>0:\ell(r)>0} \frac{1}{\lambda\mu + (1-\lambda)} \;,$$

where $\lambda \in [0,1]$ is a solution of $\ell^*(\lambda\, r) = \ell(r)$ and $\mu = \ell(\lambda\, r)/\ell(r)$.

We further extend this notion to classes of functions, and define the *anarchy value of a standard class $\mathcal{L}$ of latency functions*, by

$$\alpha(\mathcal{L}) = \sup_{0 \neq \ell \in \mathcal{L}} \alpha(\ell) \ .$$

With these definitions, we can give now a precise upper bound for the price of anarchy.

Theorem 7.5 *Let $\mathcal{L}$ be a standard class of latency functions with anarchy value $\alpha(\mathcal{L})$. Let $(\mathcal{N}, r, \ell)$ denote an instance with latency functions drawn from $\mathcal{L}$. Then, the price of anarchy is upper bounded by $\alpha(\mathcal{L})$.*

The bound in Theorem 7.5 is tight, but actually, the class of networks for which it matches lower bounds can be even simplified further if some additional conditions are satisfied. Let us define two new classes of latency functions: a class $\mathcal{L}$ is *diverse* if for each positive constant $c > 0$ there is a latency function $\ell \in \mathcal{L}$ with $\ell(0) = c$, and a class $\mathcal{L}$ is *nondegenerate* if $\ell(0) \neq 0$ for certain $\ell \in \mathcal{L}$. A network is called a *union of paths* if it can be obtained from a network of parallel links by repeated subdivisions.

In the following theorem, the term worst-case price of anarchy is with respect to the worst-case choice of the underlying instances.

Theorem 7.6 *Let $\mathcal{N}_m$ denote the network with one source node, one sink node, and m parallel links directed from source to sink.*

- *If $\mathcal{L}$ is a* standard *class of latency functions* containing the constant functions*, then the worst-case price of anarchy for all instances $(\mathcal{N}, r, \ell)$ with latency functions in $\mathcal{L}$ is identical to the worst-case price of anarchy for all instances $(\mathcal{N}_2, r, \ell)$ with latency functions in $\mathcal{L}$.*

- *If $\mathcal{L}$ is a* standard and diverse *class of latency functions, then the worst-case price of anarchy for all instances $(\mathcal{N}, r, \ell)$ with latency functions in $\mathcal{L}$ is identical to the worst-case price of anarchy for all instances $(\mathcal{N}_m, r, \ell)$ with latency functions in $\mathcal{L}$.*

- *If $\mathcal{L}$ is a* standard and nondegenerate *class of latency functions, then the worst-case price of anarchy for all instances $(\mathcal{N}, r, \ell)$ with latency functions in $\mathcal{L}$ is identical to the worst-case price of anarchy for all instances $(\mathcal{N}^*, r, \ell)$ with latency functions in $\mathcal{L}$ and with the underlying networks $\mathcal{N}^*$ that are union of paths.*

7.4 Atomic network routing games

The intuitive difference between nonatomic and atomic routing:

- each commodity represents a large population (each of whom control a negligible amount of traffic) vs. a single player

The input consists of a graph $G = (V, E)$ and k source-sink pairs (s_i, t_i). Each edge $e \in E$ has a positive continuous, nondecreasing *latency function*. The goal is to sent a positive amount of traffic r_i from s_i to t_i.

The strategy of a player i is a set P_i of $s_i - t_i$-paths. If player i chooses path $P \in P_i$ then it routes all r_i units of traffic on P.

Definition 7.1 (Atomic equilibrium flow) *Let f be a feasible flow for the atomic instance (G, r, ℓ). The flow f is an equilibrium flow if, for every player i and every pair of paths P, $P*$ in P_i of $s_i - t_i$ paths with $f_P^{(i)}$ holds*

$$\ell_P(f) \leqslant \ell_{P*}(g) \ ,$$

where g is the flow identical to f except that $g_P(i) = 0$ and $g_{P*}(i) = r_i$.

We consider only pure strategies.

Claim 7.1 *Not every instance of atomic network routing game has a pure equilibrium.*

Theorem 7.7 (Equilibrium flows in unweighted atomic instances) *Let (G, r, ℓ) be an atomic instance in which every traffic amount r_i is equal to a common positive value R. Then (G, r, ℓ) admits at least one equilibrium flow.*

Theorem 7.8 (Equilibrium flows with linear latency functions) *Let (G, r, ℓ) be an atomic instance with linear latency functions (linear = of the form $ax + b$ with $a, b \geqslant 0$). Then (G, r, ℓ) admits at least one equilibrium flow.*

7.4.1 Atomic flows with linear latency functions

Theorem 7.9 *If (G, r, ℓ) is an atomic instance with linear latency functions, then the price of anarchy (G, r, ℓ) is at most $\frac{3+\sqrt{5}}{2} \approx 2.618$.*

Theorem 7.10 *If (G, r, ℓ) is an atomic instance with linear latency functions in which all of the players control the same amount of flow, then the price of anarchy (G, r, ℓ) is at most 2.5.*

7.5 Further extensions

- **Theorem 7.11** *If f is a flow at Nash equilibrium for $(\mathcal{N}, r, \ell)$ and f^* is feasible for $(\mathcal{N}, 2r, \ell)$, then $C(f) \leqslant C(f^*)$.*

  Moreover, for any $\gamma > 0$, if f is a flow at Nash equilibrium for $(\mathcal{N}, r, \ell)$ and f^* is feasible for $(\mathcal{N}, (1+\gamma)r, \ell)$, then $C(f) \leqslant \frac{1}{\gamma} C(f^*)$.

- **Theorem 7.12** *A flow f feasible for instance $(\mathcal{N}, r, \ell)$ is at* ε-approximate *Nash equilibrium if and only if for every $i \in [k]$, and $\pi_1, \pi_2 \in \mathcal{P}_i$ with $f_{\pi_1} > 0$, it holds that $\ell_{\pi_1}(f) \leqslant (1+\varepsilon)\ell_{\pi_2}(f)$.*

- Unsplittable flows

- Fairness of optimal routing

- Complexity of constructing "Nash-"efficient networks

8 Load balancing games

Koutsoupias and Papadimitriou (1999) introduced another selfish routing model, in which the goal is to route the traffic on parallel links with linear latency functions. One can describe this model as a scheduling problem with m independent machines with speeds $s_1, \ldots, s_m$ and n independent tasks with weights $w_1, \ldots, w_n$. The goal is to allocate the tasks to the machines to minimize the maximum load of the links in the system.

It is assumed that all tasks are assigned by non-cooperative agents. The set of *pure strategies* for task i is the set $[m]$ and a *mixed strategy* is a distribution on this set.

Given a combination $(j_1, \ldots, j_n) \in [m]^n$ of pure strategies, one for each task, the *cost* for task i is $\sum_{j_k=j_i} \frac{w_k}{s_{j_i}}$, which is the time needed for machine j_i chosen by task i to complete all tasks allocated to that machine. Similarly, for a combination of pure strategies $(j_1, \ldots, j_n) \in [m]^n$, the *load* of machine j is defined as $\sum_{j_k=j} \frac{w_k}{s_j}$.

Given n tasks of length $w_1, \ldots, w_n$ and m machines with the speeds $s_1, \ldots, s_m$, let $\mathtt{opt}$ denote the *social optimum*, that is, the minimum cost over all combinations of pure strategies:

$$\mathtt{opt} \;=\; \min_{(j_1,\ldots,j_n)\in[m]^n} \; \max_{j=1}^{m} \; \sum_{i:j_i=j} \frac{w_i}{s_j} \;\; .$$

For example, if all machines have the same unit speed ($s_j = 1$ for every j, $1 \leqslant j \leqslant m$) and all tasks have the same unit weight ($w_i = 1$ for every i, $1 \leqslant i \leqslant n$), then the social optimum is $\lceil \frac{n}{m} \rceil$. It is also easy to see that in any system $\mathtt{opt} \geqslant \frac{\max_i w_i}{\max_j s_j}$. One can show that computing the social optimum is $\mathcal{NP}$-hard even for identical speeds.

For mixed strategies, let p_i^j denote the probability that an agent i sends the entire traffic w_i to a machine j. Let ℓ_j denote the *expected load* on a machine j, that is,

$$\ell_j \;=\; \frac{1}{s_j} \cdot \sum_{i=1}^{n} w_i \, p_i^j \;\; .$$

For a task i, the *expected cost of task i on machine j* is equal to

$$c_i^j \;=\; \frac{w_i}{s_j} + \sum_{t\neq i} \frac{w_t \, p_t^j}{s_j} \;=\; \ell_j + (1 - p_i^j) \, \frac{w_i}{s_j} \;\; .$$

The expected cost c_i^j corresponds to the expected finish time of task i on machine j under the processor sharing scheduling policy. This is an appropriate cost model with respect to the underlying traffic routing application.

Definition 8.1 (Nash equilibrium) *The probabilities* $(p_i^j)_{1\leqslant i\leqslant n, 1\leqslant j\leqslant m}$ *define a* Nash equilibrium *if and only if any task i will assign non-zero probabilities only to machines that minimize c_i^j, that is, $p_i^j > 0$ implies $c_i^j \leqslant c_i^q$, for every q, $1 \leqslant q \leqslant m$.*

As an example, in the system considered above in which all machines have the same unit speed and all weights are the same, the uniform probabilities $p_i^j = \frac{1}{m}$ for all $1 \leqslant j \leqslant m$ and $1 \leqslant i \leqslant n$ define a system in a Nash equilibrium.

The existence of a Nash equilibrium over mixed strategies for non-cooperative games was shown by Nash (1951). In fact, the routing game considered here admits an equilibrium even if all players are restricted to pure strategies.

Claim 8.1 *Every instance of the load balancing game admits at least one pure Nash equilibrium.*

Fix an arbitrary Nash equilibrium, that is, fix the probabilities (p_i^j) for $1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m$ that define a Nash equilibrium. Consider the randomized allocation strategies in which each task i is allocated to a single machine chosen independently at random according to the probabilities p_i^j, that is, task i is allocated to machine j with probability p_i^j. Let C_j, $1 \leqslant j \leqslant m$, be the random variable indicating the *load of machine j* in our random experiment. Observe that C_j is the weighted sum of independent 0–1 random variables J_i^j, $\mathbf{Pr}[J_i^j = 1] = p_i^j$, such that

$$C_j \;=\; \frac{1}{s_j} \sum_{i=1}^{n} w_i \cdot J_i^j \;\; .$$

Let $\mathfrak{c}$ denote the *maximum expected load* over all machines, that is,

$$\mathfrak{c} \;=\; \max_{1\leqslant j\leqslant m} \ell_j \;\; .$$

Notice that $\mathbf{E}[C_j] = \ell_j$, and therefore $\mathfrak{c} = \max_{1\leqslant j\leqslant m} \mathbf{E}[C_j]$.

Finally, let the *social cost* C be defined as the expected maximum load (instead of maximum expected load), that is,

$$\mathsf{C} \;=\; \mathbf{E}[\max_{1\leqslant j\leqslant m} C_j] \;\; .$$

Observe that $c \leqslant C$ and possibly $c \ll C$.

Our goal is to estimate the *price of anarchy* which is the worst-case ratio

$$R = \max \frac{\mathsf{C}}{\mathtt{opt}} ,$$

where the maximum is over all Nash equilibria.

8.1 Analysis of selfish behavior in load balancing games

The motivation behind the study of the price of anarchy is to quantify decline of the performance of the system (the social cost) caused by selfish behavior of the participants.

8.1.1 Pure Nash equilibria for identical machines

In the case of *pure Nash equilibria* for *identical machines*, the analysis of the price of anarchy is very similar to the classical analysis of the greedy load balancing algorithm that allocates the tasks one by one in arbitrary order, and assigns each task to the least loaded machine (Graham 1966).

Theorem 8.1 *For pure strategies the price of anarchy for m identical machines is upper bounded by*

$$2 - \frac{2}{m+1} .$$

This bound is tight.

Unlike in many other cases, for identical machines, we can show not only that they admit pure Nash equilibria, but also we can even find one efficiently. We consider a decentralized algorithm in which the agents are trying to dynamically improve its cost by moving its task to other machines. An agent is called *satisfied* if he cannot lower its cost by unilaterally moving its task to another machine. The *max-weight best response* policy considers the agents one by one and activates an agent with maximum weight among the unsatisfied agents. Then, the activate agent plays a *best response*: the agent moves its task to the machine with minimum load.

Theorem 8.2 *Starting from any assignment of the tasks to identical machines, the max-weight best response policy reaches a pure Nash equilibrium after each agent was activated at most once.*

In particular, one can find a pure Nash equilibrium for identical machines in time polynomial in the number of tasks and machines.

8.1.2 Pure and mixed Nash equilibria for uniformly related machines

The analysis for uniformly related machines (not necessarily identical) is more complicated, but also in that case one can give an exact description of the price of anarchy as a function of the number of machines and the ratio of the speed of the fastest machine over the speed of the slowest machine. [1]

Theorem 8.3 (Mixed equilibria; upper bound) *The price of anarchy for m machines is bounded from above by*

$$\mathcal{O}\left(\min\left\{\frac{\log m}{\log\log\log m},\ \frac{\log m}{\log\left(\frac{\log m}{\log(s_1/s_m)}\right)}\right\}\right)\ ,$$

where it is assumed that the speeds satisfy $s_1 \geqslant \cdots \geqslant s_m$.

In particular, the price of anarchy for m machines is $\mathcal{O}\left(\frac{\log m}{\log\log\log m}\right)$.

The theorem follows directly from the following two results: the maximum expected load $\mathfrak{c}$ satisfies $\mathfrak{c} = \mathfrak{opt}\cdot\Gamma^{(-1)}(m) = \mathfrak{opt}\cdot\mathcal{O}\left(\min\left\{\frac{\log m}{\log\log m}, \log\left(\frac{s_1}{s_m}\right)\right\}\right)$ and the social cost C satisfies $\mathsf{C} = \mathfrak{opt}\cdot\mathcal{O}\left(\frac{\log m}{\log\left(\frac{\mathfrak{opt}\cdot\log m}{\mathfrak{c}}\right)}+1\right)$.

If one applied these results to systems in which all agents follow only *pure strategies*, then since then $\ell_j = C_j$ for every j, it holds that $\mathsf{C} = \mathfrak{c}$. This leads to the following result.

Corollary 8.2 (Pure equilibria; upper bound) *For pure strategies the price of anarchy for m machines is upper bounded by*

$$\mathcal{O}\left(\min\left\{\frac{\log m}{\log\log m},\ \log\left(\frac{s_1}{s_m}\right)\right\}\right)\ ,$$

where it is assumed that the speeds satisfy $s_1 \geqslant \cdots \geqslant s_m$.

Theorem 8.5 below proves that this corollary gives an asymptotically tight bound for the price of anarchy for pure strategies.

By Theorem 8.3, in the special case when all *machines are identical*, the price of anarchy is $\mathcal{O}\left(\frac{\log m}{\log\log m}\right)$. However, in this special case one can get a stronger bound that is tight up to an additive constant.

[1] To simplify the notation, for any real $x \geqslant 0$, let $\log x$ denote $\log x = \max\{\log_2 x, 1\}$. Also, following standard convention, $\Gamma(N)$ is used to denote the *Gamma (factorial) function*, which for any natural N is defined by $\Gamma(N+1) = N!$ and for an arbitrary real $x > 0$ is $\Gamma(x) = \int_0^\infty t^{x-1}\,e^{-t}\,dt$. For the inverse of the Gamma function, $\Gamma^{(-1)}(N)$, it is known that $\Gamma^{(-1)}(N) = x$ such that $\lfloor x\rfloor! \leqslant N-1 \leqslant \lceil x\rceil!$. It is well known that $\Gamma^{(-1)}(N) = \frac{\log N}{\log\log N}\,(1+o(1))$.

Theorem 8.4 (Identical machines) *For m identical machines the price of anarchy is at most*

$$\Gamma^{(-1)}(m) + \Theta(1) \;=\; \frac{\log m}{\log\log m} \cdot (1 + o(1)) \;\; .$$

One can obtain a lower bound for the price of anarchy for m *identical machines* by considering the system in which $p_i^j = \frac{1}{m}$ for every i, j. Gonnet proved that then the price of anarchy is $\Gamma^{(-1)}(m) - \frac{3}{2} + o(1)$, which implies that Theorem 8.4 is tight up to an additive constant.

The next theorem shows that the upper bound in Theorem 8.3 is asymptotically tight.

Theorem 8.5 (Mixed equilibria; lower bound) *The price of anarchy for m machines is lower bounded by*

$$\Omega\left(\min\left\{\frac{\log m}{\log\log\log m},\; \frac{\log m}{\log\left(\frac{\log m}{\log(s_1/s_m)}\right)}\right\}\right) \;\; .$$

In particular, the price of anarchy for m machines is $\Omega\left(\frac{\log m}{\log\log\log m}\right)$.

In fact, it can be shown (analogously to the upper bound) that for every positive integer m, positive real r, and $S \geqslant 1$, there exists a set of m machines with $\frac{s_1}{s_m} = S$ being in a Nash equilibrium and satisfying $\mathfrak{opt} = r$, $\mathfrak{c} = \mathfrak{opt} \cdot \Omega\left(\min\left\{\frac{\log m}{\log\log m},\; \log\left(\frac{s_1}{s_m}\right)\right\}\right)$, and $\mathsf{C} = \mathfrak{opt} \cdot \Omega\left(\frac{\log m}{\log\left(\frac{\mathfrak{opt}\cdot\log m}{c}\right)}\right)$.

8.1.3 Computing pure equilibria

In Theorem 8.2, we presented a decentralized algorithm that dynamically improves the allocation of the tasks to efficiently reach a pure Nash equilibrium in load balancing games for identical machines. No such efficient dynamic algorithm is known for uniformly related machines. However, we still can find efficiently a pure Nash equilibrium.

The *LPT algorithm* (*largest processing time*) scheduling algorithm inserts the tasks in a nonincreasing order of weight, assigning each task to a machine that minimizes the cost of the task at the insertion time.

Theorem 8.6 *The LPT algorithm finds a pure Nash equilibrium for load balancing games on uniformly related machines.*

The LPT algorithm has also one very useful feature: it does not only returns a pure Nash equilibrium, but in fact it returns a good one: it approximates the optimal makespan within a ratio of at most $\frac{5}{3}$. Furthermore, while the problem of finding a pure Nash equilibrium that computes an assignment with optimal makespan is $\mathcal{NP}$-hard, it is possible to find one that minimizes the makespan within a factor of $(1+\varepsilon)$ for any given $\varepsilon > 0$.

9 Auctions and mechanism design

An auction is one of the most fundamental process of buying and selling goods or services. It is a process of exchanging/transferring (buying or selling) goods or services by offering them up for a bid, taking bids, and then transferring/exchanging them.

An example of modeling:

- Seller wants to sell an item
- Bidder i has valuation v_i
  - maximum willingness to pay
  - known to bidder, unknown to seller
- Utility of player i is equal to $v_i -$ price paid; or 0 if loses auction
- Submits bid b_i to maximize its utility

1st price auction: seller selects the highest bidder and ask him the bidding value v_i.

Vickrey auction and truthfulness: A Vickrey auction is a sealed-bid auction, in which the bidders bid for a single item, the highest bidder wins, and the price paid is the second-highest bid.

Theorem 9.1 *Vickrey auction is thruthful.*

- truthful bidding ($b_i = v_i$) is "foolproof" i.e., a false bid never outperforms a true bid.

9.1 VCG Mechanism

A Vickrey-Clarke-Groves (VCG, 1961/71/73) auction is a sealed-bid auction where *multiple* items are up for bid, and each bidder submits a possibly different value for each item. The VCG mechanism assigns the items in a truthful ("socially optimal") manner, while ensuring each bidder receives at most one item. It ensures that the optimal strategy for a bidder is to bid the true valuations of the objects.

Utility model: bidder i's utility is equal to $v_i(\omega)$ − payment.

VCG mechanism:

- collect bid $b_i(\omega)$ for all i and all outcomes $\omega \in \Omega$
- select ω^* in $\arg\max \sum_i b_i(\omega)$
- charge $p_i = -\sum_{j \neq i} b_j(\omega)$ + suitable constant (a function that only depends on the valuation of the other players)

Theorem 9.2 *The mechanism is truthful and it maximizes welfare* $\sum_i v_i(\omega)$ *over* Ω *(assuming truthful bids).*

9.2 Overarching goals

Our goal is to design "optimal" truthful mechanisms and auctions for a wide range of problem (combinatorial auctions, scheduling, etc) and for different objectives (welfare, revenue). We also often have complexity requirements (e.g., polynomial-time running time). We would like to design general techniques, frameworks, and provide their analyzes, and proofs of their limitations.

Recently we have seen that the following "algorithmic" philosophy has been playing important role: *"Philosophy: designing truthful mechanisms boils down to designing algorithms in a certain restricted computational model"*

Let us consider single-parameter problems. The outcome space is a set of vectors of the form $\mathbf{x} = (x_1, \ldots, x_n)$. The utility model assumes that each bidder i has private valuation v_i, has utility function equal to $v_i x_i -$ payment, and it submits bid b_i to maximize its utility.

Any truthful mechanism follows the following scheme:

- collects bid b_i from each player i
- invoke a possibly randomized allocation rule which maps the bid b_i into x_i
- invoke a possibly randomized payment rule which maps the bid b_i to the payment p_i

The essence of being truthful is that for every i, v_i, and other bids, setting $v_i = b_i$ maximizes the expected utility $v_i x_i(\mathbf{b}) - p_i(\mathbf{b})$.

There are two types of allocation rules:

- **Implementable allocation rule:** is a function χ from bids to expected allocations that admits a payment rule p such that (χ, p) is truthful.
  - i.e., truthful bidding always maximizes bidder's expected utility
- **Monotone allocation rule:** for every fixed bidder i, given fixed other bids b_i, expected allocation only increases in the bid b_i.
  - example: highest bidder wins

Theorem 9.3 (Myerson 1981) *An allocation rule χ is implementable if and only if it is monotone.*

Moreover, for every monotone allocation rule χ, there is a unique payment rule p such that (χ, p) is truthful and losers always pay 0.