Bachelor course 64331010, Caput Operations Research, HC Caput OR 3.5 (3 ects)

Lecture Notes

Algorithmic Game Theory

Department of Econometrics and Operations Research
Faculty of Economics and Business Administration
VU University Amsterdam
March–May 2010

Prof. dr. Guido Schäfer
Email: g.schaefer@cwi.nl

Centrum Wiskunde & Informatica
Algorithms, Combinatorics and Optimization
Science Park 123, 1098 XG Amsterdam, The Netherlands

Vrije Universiteit Amsterdam
Faculty of Economics and Business Administration
Department of Econometrics and Operations Research
De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

May 2010

Contents

1 Selfish Routing

We consider network routing problems in which users choose their routes so as to minimize their own travel time. Our main focus will be to study the inefficiency of Nash equilibria and to identify effective means to decrease the inefficiency caused by selfish behavior.

We first consider two examples:

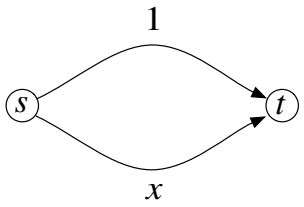

Figure 1: Pigou instance

Example 1.1 (Pigou's example)**.** Consider the parallel-arc network in Figure 1. For every arc a, we have a latency function $\ell_a : \mathbb{R}_+ \to \mathbb{R}_+$, representing the load-dependent travel time or latency for traversing this arc. In the above example, we have for the upper arc $\ell_a(x) = 1$, i.e., the latency is one independently of the amount of flow on that arc. The lower arc has latency function $\ell_a(x) = x$, i.e., the latency grows linearly with the amount of flow on that arc. Suppose we want to send one unit of flow from s to t and that this one unit of flow corresponds to infinitely many users that want to travel from s to t.

Every selfish user will reason as follows: The latency of the upper arc is one (independently of the flow) while the latency of the lower arc is at most one (and even strictly less than one if some users are not using this arc). Thus, every user chooses the lower arc. The resulting flow is a Nash flow. Since every user experiences a latency of one, the total average latency of this Nash flow is one.

We next compute an optimal flow that minimizes the total average latency of the users. Assume we send $p \in [0,1]$ units of flow along the lower arc and $1-p$ units of flow along the upper arc. The total average latency is $(1-p) \cdot 1 + p \cdot p = 1 - p + p^2$. This function is minimized for $p = \frac{1}{2}$. Thus, the optimal flow sends one-half units of flow along the upper and one-half units of flow along the lower arc. Its total average latency is $\frac{3}{4}$.

This example shows that selfish user behavior may lead to outcomes that are inefficient: The resulting Nash flow is suboptimal with a total average latency that is $\frac{4}{3}$ times larger than the total average latency of an optimal flow. This raises the following natural questions: How large can this inefficiency ratio be in general networks? Does it depend on the topology of the network?

The next example shows a similar effect, though having a slightly different flavor.

Example 1.2 (Braess' paradox)**.** Consider the network in Figure 2 (left). Assume that we want to send one unit of flow from s to t. It is not hard to verify that the Nash flow splits evenly and sends one-half units of flow along the upper and lower arc, respectively. This flow is also optimal having a total average latency of $\frac{3}{2}$.

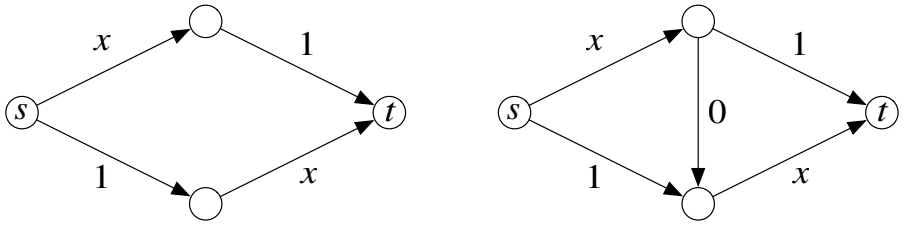

Figure 2: Braess Paradox

Now, suppose there is a global authority that wants to improve the overall traffic situation by building new roads. The network in Figure 2 (right) depicts an augmented network where an additional arc with constant latency zero has been introduced. How do selfish users react to this change? What happens is that every user chooses the zig-zag path, first traversing the upper left arc, then the newly introduced zero latency arc and then the lower right arc. The resulting Nash flow has a total average latency of 2.

The Braess Paradox shows that extending the network infrastructure does not necessarily lead to an improvement with respect to the total average latency if users choose their routes selfishly. In the above case, the total average latency degrades by a factor of $\frac{4}{3}$. In general, one may ask the following questions: How large can this degradation be? Can we develop efficient methods to detect such phenomena?

1.1 Model

We formalize the setting introduced above. An instance of a *selfish routing game* is given as follows:

- directed graph $G = (V,A)$ with vertex set V and arc set A;
- set of k commodities $[k] := \{1,\dots,k\}$, specifying for each commodity $i \in [k]$ a source vertex s_i and a target vertex t_i;
- for each commodity $i \in [k]$, a demand $r_i > 0$ that represents the amount of flow that has to be sent from s_i to t_i;
- nondecreasing and continuous latency function $\ell_a : \mathbb{R}_+ \to \mathbb{R}_+$ for every arc $a \in A$.

We use (G, r, ℓ) to refer to an instance for short.

Let $\mathscr{P}_i$ be the set of all simple paths from s_i to t_i in G and let $\mathscr{P} := \cup_i \mathscr{P}_i$. A *flow* is a function $f : \mathscr{P} \to \mathbb{R}_+$. The flow f is *feasible* (with respect to r) if for all $i \in [k]$, $\sum_{P \in \mathscr{P}_i} f_P = r_i$, i.e., the total flow send from s_i to t_i meets the demand r_i. For a given flow f, we define the aggregated flow on arc $a \in A$ as $f_a := \sum_{P \in \mathscr{P} : a \in P} f_P$.

The above representation of a flow is also known as the *path formulation*. The path formulation is not the most compact way to represent flows (e.g., the number of paths in $\mathscr{P}_i$ is exponential in general), but will sometimes be convenient for our considerations. An alternative, more compact representation of flows is to use the arc formulation: Here a flow

f is represented by k nonnegative vectors $f^i \in \mathbb{R}^A_+$, $i \in [k]$. Each flow f^i represents the flow of commodity $i \in [k]$. The flow $f = (f^1, \ldots, f^k)$ is *feasible* if for every vertex $v \in V$, we have

$$\forall i \in [k]: \quad \sum_{a \in \delta^+(v)} f^i_a - \sum_{a \in \delta^-(v)} f^i_a = \gamma^i(v), \tag{1}$$

where $\delta^+(v)$ and $\delta^-(v)$ refer to the set of arcs leaving and entering v, respectively. Here $\gamma^i(v)$ is defined as follows:

$$\gamma^i(v) := \begin{cases} r_i & \text{if } v = s_i \\ -r_i & \text{if } v = t_i \\ 0 & \text{otherwise.} \end{cases}$$

That is, (1) states that for every commodity $i \in [k]$, the inflow (with respect to f^i) is equal to the outflow for every vertex $v \in V \setminus \{s_i, t_i\}$, the outflow of $v = s_i$ is r_i and the inflow of $v = t_i$ is $-r_i$. We say that $f = (f^i)_{i \in [k]}$ is a *multi-commodity flow*. The aggregated flow of an arc $a \in A$ is $f_a := \sum_{i \in [k]} f^i_a$.

We remark that every path decomposition of a flow defines a unique decomposition into arc flows. Conversely, an arc decomposition of a flow may be represented by several path decompositions.

In selfish routing games with *nonatomic* players it is assumed that the flow f^i of commodity $i \in [k]$ is carried by a large number of players, each controlling an infinitesimal fraction of the entire demand r_i. The *total travel time* of a path $P \in \mathscr{P}$ with respect to f is defined as the sum of the latencies of the arcs on that path:

$$\ell_P(f) := \sum_{a \in P} \ell_a(f_a).$$

We assess the overall quality of a given flow f by means of a global cost function C. Though there are potentially many different cost functions that one may want to consider (depending on the application), we focus on the *total average latency* as cost function here.

Definition 1.1. The *total cost* of a flow f is defined as:

$$C(f) := \sum_{P \in \mathscr{P}} \ell_P(f) f_P. \tag{2}$$

Note that the total cost can equivalently be expressed as the sum of the average latencies on the arcs:

$$C(f) = \sum_{P \in \mathscr{P}} \ell_P(f) f_P = \sum_{P \in \mathscr{P}} \Big(\sum_{a \in P} \ell_a(f_a) \Big) f_P = \sum_{a \in A} \Big(\sum_{P \in \mathscr{P}: a \in P} f_P \Big) \ell_a(f_a) = \sum_{a \in A} \ell_a(f_a) f_a.$$

1.2 Nash Flow for Nonatomic Players

The basic viewpoint that we adopt here is that players act selfishly in that they attempt to minimize their own individual travel time. A standard solution concept to predict outcomes of selfish behavior is the one of an equilibrium outcome in which no player has an incentive

to unilaterally deviate from its current strategy. In the context of nonatomic selfish routing games, this viewpoint translates to the following definition:

Definition 1.2. A feasible flow f for the instance (G, r, ℓ) is a *Nash flow* if for every commodity $i \in [k]$ and two paths $P,\ Q \in \mathscr{P}_i$ with $f_P > 0$ and for every $\delta \in (0, f_P]$, we have $\ell_P(f) \leq \ell_Q(\tilde{f})$, where

$$\tilde{f}_P := \begin{cases} f_P - \delta & \text{if } P = P \\ f_P + \delta & \text{if } P = Q \\ f_P & \text{otherwise.} \end{cases}$$

Intuitively, the above definition states that for every commodity $i \in [k]$, shifting $\delta \in (0, f_P]$ units of flow from a flow carrying path $P \in \mathscr{P}_i$ to an arbitrary path $Q \in \mathscr{P}_i$ does not lead to a smaller latency.

A similar concept was introduced by Wardrop (1952) in his first principle: A flow for the nonatomic selfish routing game is a *Wardrop equilibrium* if for every source-target pair the latencies of the used routes are less than or equal to those of the unused routes.

Definition 1.3. A feasible flow f for the instance (G, r, ℓ) is a *Wardrop equilibrium* (or *Wardrop flow*) if

$$\forall i \in [k],\ \forall P,\ Q \in \mathscr{P}_i,\ f_P > 0 : \quad \ell_P(f) \leq \ell_Q(f). \tag{3}$$

For $\delta \to 0$ the definition of a Nash flow corresponds to the one of a Wardrop flow. Subsequently, we use the Wardrop flow definition; we slightly abuse naming here and will also refer to such flows as Nash flows.

Corollary 1.1. *Let f be a Nash flow for (G, r, ℓ) and define for every $i \in [k]$, $c_i(f) := \min_{P \in \mathscr{P}_i} \ell_P(f)$. Then $\ell_P(f) = c_i(f)$ for every $P \in \mathscr{P}_i$ with $f_P > 0$.*

Proof. By the definition of $c_i(f)$, we have that for every $P \in \mathscr{P}_i$: $\ell_P(f) \geq c_i(f)$. Using (3), we conclude that for every $P \in \mathscr{P}_i$ with $f_P > 0$: $\ell_P(f) \leq c_i(f)$. □

Note that the above corollary states that for each commodity all flow carrying paths have the same latency and all other paths cannot have a smaller latency. The flow carrying paths are thus shortest paths with respect to the total latency.

We next argue that Nash flows always exist and that their cost is unique. In order to do so, we use a powerful result from convex optimization. Consider the following program (CP):

$$\begin{aligned} \min \quad & \sum_{a \in A} h_a(f_a) & & & \\ \text{s.t.} \quad & \sum_{P \in \mathscr{P}_i} f_P &= r_i & \forall i \in [k] \\ & f_a &= \sum_{P \in \mathscr{P} : a \in P} f_P & \forall a \in A \\ & f_P &\geq 0 & \forall P \in \mathscr{P}. \end{aligned}$$

Note that the set of all feasible solutions for (CP) corresponds exactly to the set of all flows that are feasible for our selfish routing instance (G, r, ℓ). The above program is a linear program if the functions $(h_a)_{a\in A}$ are linear. (CP) is a convex program if the functions $(h_a)_{a\in A}$ are convex. A convex program can be solved efficiently by using, e.g., the ellipsoid method. The following is a fundamental theorem in convex (or, more generally, non-linear) optimization:

Theorem 1.1 (Karush–Kuhn–Tucker optimality conditions)**.** *Consider the program (CP) with continuously differentiable and convex functions $(h_a)_{a\in A}$. A feasible flow f is an optimal solution for (CP) if and only if*

$$\forall i \in [k],\ \forall P,\ Q \in \mathscr{P}_i,\ f_P > 0 : \quad h'_P(f) := \sum_{a\in P} h'_a(f_a) \leq \sum_{a\in Q} h'_a(f_a) =: h'_Q(f), \tag{4}$$

where $h'_a(x)$ refers to the first derivative of $h_a(x)$.

Observe that (4) is very similar to the Wardrop equilibrium conditions (3). In fact, these two conditions coincide if we define for every $a \in A$:

$$h_a(f_a) := \int_0^{f_a} \ell_a(x)dx. \tag{5}$$

Corollary 1.2. *Let (G, r, ℓ) be a selfish routing instance with nondecreasing and continuous latency functions $(\ell_a)_{a\in A}$. A feasible flow f is a Nash flow if and only if it is an optimal solution to (CP) with functions $(h_a)_{a\in A}$ as defined in* (5)*.*

Proof. For every arc $a \in A$, the function h_a is convex (since ℓ_a is nondecreasing) and continuously differentiable (since ℓ_a is continuous). The proof now follows from Theorem 1.1. □

Corollary 1.3. *Let (G, r, ℓ) be a selfish routing instance with nondecreasing and continuous latency functions $(\ell_a)_{a\in A}$. Then a Nash flow f always exists. Moreover, its cost $C(f)$ is unique.*

Proof. The set of all feasible flows for (CP) is compact (closed and bounded). Moreover, the objective function of (CP) with (5) is continuous (since ℓ_a is continuous for every $a \in A$). Thus, the minimum of (CP) must exist (by the extreme value theorem of Weierstraß). Since the objective function of (CP) is convex, the optimal value of (CP) is unique. It is not hard to conclude that the cost $C(f)$ of a Nash flow is unique. □

Note that, in particular, the above observations imply that we can compute a Nash flow for a given nonatomic selfish routing instance (G, r, ℓ) efficiently by solving the convex program (CP) with (5).

1.3 Optimal Flow

We define an optimal flow as follows:

Definition 1.4. A feasible flow f^* for the instance (G, r, ℓ) is an *optimal flow* if $C(f^*) \leq C(x)$ for every feasible flow x.

The set of optimal flows corresponds to the set of all optimal solutions to (CP) if we define for every arc $a \in A$:

$$h_a(f_a) := \ell_a(f_a) f_a. \tag{6}$$

Since the cost function C is continuous (because ℓ_a is continuous for every $a \in A$), we conclude that an optimal flow always exists (again using the extreme value theorem by Weierstraß). Moreover, we will assume that h_a is convex and continuously differentiable for each arc $a \in A$; latency functions $(\ell_a)_{a \in A}$ that satisfy these conditions are called *standard*. Using Theorem 1.1, we obtain the following characterization of optimal flows:

Corollary 1.4. *Let the latency functions $(\ell_a)_{a \in A}$ be standard. A feasible flow f^* for the instance (G, r, ℓ) is an optimal flow if and only if:*

$$\forall i \in [k],\ \forall P,\ Q \in \mathscr{P}_i,\ f^*_P > 0 : \quad \sum_{a \in P} \ell_a(f^*_a) + \ell'_a(f^*_a) f^*_a \leq \sum_{a \in Q} \ell_a(f^*_a) + \ell'_a(f^*_a) f^*_a.$$

That is, an optimal flow is a Nash flow with respect to so-called *marginal latency functions* $(\ell^*_a)_{a \in A}$, which are defined as

$$\ell^*_a(x) := \ell_a(x) + \ell'_a(x) x.$$

1.4 Price of Anarchy

We study the inefficiency of Nash flows in comparison to an optimal flow. A common measure of the inefficiency of equilibrium outcomes is the *price of anarchy*.

Definition 1.5. Let (G, r, ℓ) be an instance of the selfish routing game and let f and f^* be a Nash flow and an optimal flow, respectively. The *price of anarchy* $\rho(G, r, \ell)$ of the instance (G, r, ℓ) is defined as:

$$\rho(G, r, \ell) = \frac{C(f)}{C(f^*)}. \tag{7}$$

(Note that (7) is well-defined since the cost of Nash flows is unique.) The price of anarchy of a set of instances $\mathscr{I}$ is defined as

$$\rho(\mathscr{I}) = \sup_{(G, r, \ell) \in \mathscr{I}} \rho(G, r, \ell).$$

1.5 Upper Bounds on the Price of Anarchy

Subsequently, we derive upper bounds on the price of anarchy for selfish routing games. The following variational inequality will turn out to be very useful.

Lemma 1.1 (Variational inequality)**.** *A feasible flow f for the instance (G,r,ℓ) is a Nash flow if and only if it satisfies that for every feasible flow x:*

$$\sum_{a\in A} \ell_a\big(f_a\big)(f_a - x_a) \le 0. \tag{8}$$

Proof. Given a flow f satisfying (8), we first show that condition (3) of Definition 1.3 holds. Let $P,\ Q \in \mathscr{P}_i$ be two paths for some commodity $i \in [k]$ such that $\delta := f_P > 0$. Define a flow x as follows:

$$x_a := \begin{cases} f_a & \text{if } a \in P \cap Q \text{ or } a \notin P \cup Q \\ f_a - \delta & \text{if } a \in P \\ f_a + \delta & \text{if } a \in Q. \end{cases}$$

By construction x is feasible. Hence, from (8) we obtain:

$$\sum_{a\in A} \ell_a\big(f_a\big)(f_a - x_a) = \sum_{a\in P} \ell_a(f_a)(f_a - (f_a - \delta)) + \sum_{a\in Q} \ell_a(f_a)(f_a - (f_a + \delta)) \le 0.$$

We divide the inequality by $\delta > 0$, which yields the Wardrop conditions (3).

Now assume that f is a Nash flow. By Corollary 1.1, we have for every $i \in [k]$ and $P \in \mathscr{P}_i$ with $f_P > 0$: $\ell_P(f) = c_i(f)$. Furthermore, for $Q \in \mathscr{P}_i$ with $f_Q = 0$, we have $\ell_Q(f) \ge c_i(f)$. It follows that for every feasible flow x:

$$\begin{aligned} \sum_{a\in A} \ell_a(f_a) f_a &= \sum_{i\in[k]} \sum_{P\in\mathscr{P}_i} c_i(f) f_P = \sum_{i\in[k]} c_i(f) \left(\sum_{P\in\mathscr{P}_i} f_P \right) = \sum_{i\in[k]} c_i(f) \left(\sum_{P\in\mathscr{P}_i} x_P \right) \\ &= \sum_{i\in[k]} \sum_{P\in\mathscr{P}_i} c_i(f) x_P \le \sum_{i\in[k]} \sum_{P\in\mathscr{P}_i} \ell_P(f) x_P = \sum_{a\in A} \ell_a(f_a)\, x_a. \end{aligned}$$

□

We derive an upper bound on the price of anarchy for affine linear latency functions with nonnegative coefficients:

$$\mathscr{L}_1 := \{ g : \mathbb{R}_+ \to \mathbb{R}_+ \ :\ g(x) = q_1 x + q_0 \text{ with } q_0, q_1 \in \mathbb{R}_+ \}.$$

Theorem 1.2. *Let (G,r,ℓ) be an instance of a nonatomic routing game with affine linear latency functions $(\ell_a)_{a\in A} \in \mathscr{L}_1^A$. The price of anarchy $\rho(G,r,\ell)$ is at most $\frac{4}{3}$.*

Proof. Let f be a Nash flow and let x be an arbitrary feasible flow for (G,r,ℓ). Using the variational inequality (8), we obtain

$$\begin{aligned} C(f) &= \sum_{a\in A} \ell_a(f_a) f_a \le \sum_{a\in A} \ell_a(f_a) x_a = \sum_{a\in A} \ell_a(f_a) x_a + \ell_a(x_a) x_a - \ell_a(x_a) x_a \\ &= \sum_{a\in A} \ell_a(x_a) x_a + \underbrace{[\ell_a(f_a) - \ell_a(x_a)] x_a}_{=: W_a(f_a, x_a)} = \sum_{a\in A} \ell_a(x_a) x_a + \sum_{a\in A} W_a(f_a, x_a). \end{aligned}$$

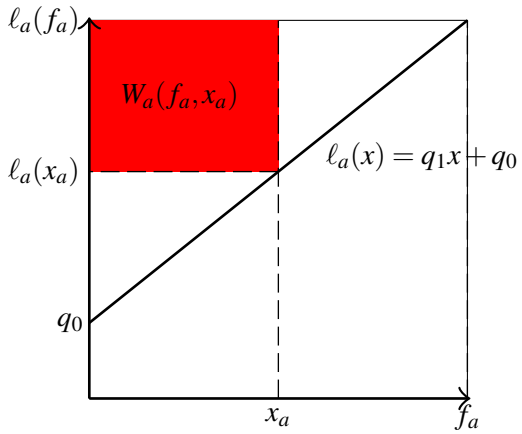

Figure 3: Illustration of the worst case ratio of $W_a(f_a,x_a)$ and $\ell_a(f_a)\,f_a$.

We next bound the function $W_a(f_a,x_a)$ in terms of $\omega \cdot \ell_a(f_a)f_a$ for some $0 \leq \omega < 1$, where

$$\omega := \max_{f_a,x_a \geq 0} \frac{(\ell_a(f_a) - \ell_a(x_a))x_a}{\ell_a(f_a)f_a} = \max_{f_a,x_a \geq 0} \frac{W_a(f_a,x_a)}{\ell_a(f_a)f_a}.$$

Note that for $x_a \geq f_a$ we have $\omega \leq 0$ (because latency functions are non-decreasing). Hence, we can assume $x_a \leq f_a$. See Figure 3 for a geometric interpretation. Since latency functions are affine linear, ω is upper bounded by $\frac{1}{4}$. We obtain

$$C(f) \leq C(x) + \sum_{a \in A} \frac{1}{4}\ell_a(f_a)f_a = C(x) + \frac{1}{4}C(f).$$

Rearranging terms and letting x be an optimal flow concludes the proof. □

We can extend the above proof to more general classes of latency functions. For the latency function ℓ_a of an arc $a \in A$, define

$$\omega(\ell_a) := \sup_{f_a,x_a \geq 0} \frac{(\ell_a(f_a) - \ell_a(x_a))x_a}{\ell_a(f_a)\,f_a}. \tag{9}$$

We assume by convention $0/0 = 0$. See Figure 4 for a graphical illustration of this value. For a given class $\mathscr{L}$ of non-decreasing latency functions, we define

$$\omega(\mathscr{L}) := \sup_{\ell_a \in \mathscr{L}} \omega(\ell_a).$$

Theorem 1.3. *Let (G,r,ℓ) be an instance of the nonatomic selfish routing game with latency functions $(\ell_a)_{a\in A} \in \mathscr{L}^A$. Let $0 \leq \omega(\mathscr{L}) < 1$ be defined as above. The price of anarchy $\rho(G,r,\ell)$ is at most $(1-\omega(\mathscr{L}))^{-1}$.*

Proof. Let f be a Nash flow and let x be an arbitrary feasible flow. We have

$$C(f) = \sum_{a \in A} \ell_a(f_a)f_a \leq \sum_{a \in A} \ell_a(f_a)x_a = \sum_{a \in A} \ell_a(f_a)x_a + \ell_a(x_a)x_a - \ell_a(x_a)x_a$$

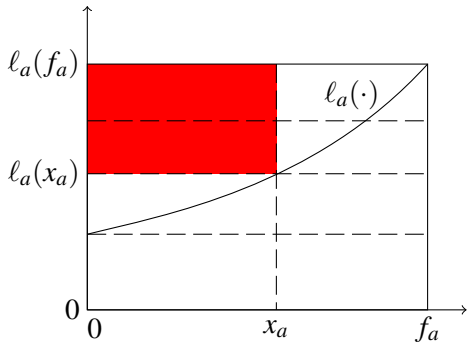

Figure 4: Illustration of $\omega(\ell_a)$.

$$= \sum_{a \in A} \ell_a(x_a)x_a + [\ell_a(f_a) - \ell_a(x_a)]x_a \leq C(x) + \omega(\mathscr{L})C(f).$$

Here, the first inequality follows from the variational inequality (8). The last inequality follows from the definition of $\omega(\mathscr{L})$. Since $\omega(\mathscr{L}) < 1$, the claim follows. □

In general, we define $\mathscr{L}_d$ as the set of latency functions $g : \mathbb{R}_+ \to \mathbb{R}_+$ that satisfy

$$g(\mu x) \geq \mu^d g(x) \quad \forall \mu \in [0,1].$$

Note that $\mathscr{L}_d$ contains polynomial latency functions with nonnegative coefficients and degree at most d.

Lemma 1.2. *Consider latency functions in $\mathscr{L}_d$. Then*

$$\omega(\mathscr{L}_d) \leq \frac{d}{(d+1)^{(d+1)/d}}.$$

Proof. Recall the definition of $\omega(\ell_a)$:

$$\omega(\ell_a) = \sup_{f_a, x_a \geq 0} \frac{\big(\ell_a(f_a) - \ell_a(x_a)\big)x_a}{\ell_a(f_a)\, f_a}. \tag{10}$$

We can assume that $x_a \leq f_a$ since otherwise $\omega(\ell_a) \leq 0$. Let $\mu := \frac{x_a}{f_a} \in [0,1]$. Then

$$\begin{aligned} \omega(\ell_a) &= \max_{\mu \in [0,1], f_a \geq 0} \left(\frac{\big(\ell_a(f_a) - \ell_a(\mu f_a)\big)\mu f_a}{\ell_a(f_a) f_a} \right) \leq \max_{\mu \in [0,1], f_a \geq 0} \left(\frac{\big(\ell_a(f_a) - \mu^d \ell_a(f_a)\big)\mu f_a}{\ell_a(f_a) f_a} \right) \\ &= \max_{\mu \in [0,1]} (1 - \mu^d)\mu. \end{aligned} \tag{11}$$

Here, the first inequality holds since $\ell_a \in \mathscr{L}_d$. Since this is a strictly convex program, the unique global optimum is given by

$$\mu^* = \left(\frac{1}{d+1} \right)^{\frac{1}{d}}.$$

d	1	2	3	...
$\rho(G,r,\ell)$	≈ 1.333	≈ 1.626	≈ 1.896	

Table 1: The price of anarchy for polynomial latency functions of degree d.

Replacing μ^* in (11) yields the claim. □

Theorem 1.4. *Let (G,r,ℓ) be an instance of a nonatomic routing game with latency functions $(\ell_a)_{a\in A} \in \mathscr{L}_d^A$. The price of anarchy $\rho(G,r,\ell)$ is at most*

$$\rho(G,r,\ell) \leq \left(1 - \frac{d}{(d+1)^{(d+1)/d}}\right)^{-1}.$$

Proof. The theorem follows immediately from Theorem 1.3 and Lemma 1.2. □

The price of anarchy for polynomial latency functions with nonnegative coefficients and degree d is given in Table 1 for small values of d.

1.6 Lower Bounds on the Price of Anarchy

We can show that the bound that we have derived in the previous section is actually tight.

Theorem 1.5. *Consider nonatomic selfish routing games with latency functions in $\mathscr{L}_d$. There exist instances such that the price of anarchy is at least*

$$\left(1 - \frac{d}{(d+1)^{(d+1)/d}}\right)^{-1}.$$

Proof. See Exercise 1 of Assignment 1. □

1.7 Bicriteria Results

Theorem 1.6. *Let (G,r,ℓ) be a nonatomic selfish routing instance. The cost of a Nash flow for (G,r,ℓ) is at most the cost of an optimal flow for the instance $(G,2r,\ell)$.*

Proof. Let f be a Nash flow for the instance (G,r,ℓ) and let x be an optimal flow for the instance $(G,2r,\ell)$. Note that the flow $\frac{1}{2}x$ is feasible for (G,r,ℓ). Using the variational inequality (8), we derive (similar as in the proof of Theorem 1.3)

$$C(f) = \sum_{a\in A} \ell_a(f_a) f_a \leq \sum_{a\in A} \ell_a(f_a) \cdot \frac{1}{2} x_a \leq \frac{1}{2}\Big(C(x) + \omega(\mathscr{L}) C(f)\Big).$$

Observe that $\omega(\mathscr{L}) \leq 1$ which implies that $C(f) \leq C(x)$. □

1.8 Algorithmic View on Braess' Paradox

Recall the Braess Paradox presented in Example 1.2. If we remove the shortcut edge from the instance depicted in Figure 5 (right), the total average latency of a Nash flow improves from 2 to $\frac{3}{2}$. In this section, we address the question whether such subnetworks can be found efficiently.

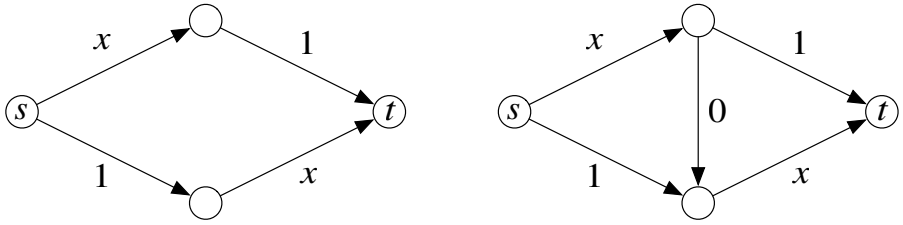

Figure 5: Braess Paradox

Suppose we are given a single-commodity instance (G,r,ℓ) of the nonatomic selfish routing game. Let f be a Nash flow for (G,r,ℓ) and define $d(G,r,\ell) := c_1(f)$ as the common latency of all flow-carrying paths (see Corollary 1.1). We study the following optimization problem: Given (G,r,ℓ), find a subgraph $H \subseteq G$ that minimizes $d(H,r,\ell)$. We call this problem the NETWORK DESIGN problem.

Corollary 1.5. *Let (G,r,ℓ) be a single-commodity instance of the nonatomic selfish routing game with linear latency functions. Then for every subgragph $H \subseteq G$:*

$$d(G,r,\ell) \leq \frac{4}{3} d(H,r,\ell).$$

Proof. Let h and f be the Nash flows for the instances (H,r,ℓ) and (G,r,ℓ), respectively. By Corollary 1.1, the latency of every flow-carrying path in a Nash flow is equal. Thus, the costs of the Nash flows f and h, respectively, are $rd(G,r,\ell)$ and $rd(H,r,\ell)$. Using that h is a feasible flow for (G,r,ℓ) and the upper bound of $4/3$ on the price of anarchy for linear latencies, we obtain

$$C(f) = rd(G,r,\ell) \leq \frac{4}{3} C(h) = \frac{4}{3} rd(H,r,\ell).$$

□

We can generalize the above proof to obtain:

Corollary 1.6. *Let (G,r,ℓ) be a single-commodity instance of the nonatomic selfish routing game with polynomial latency functions in $\mathscr{L}_d$. Then for every subgraph $H \subseteq G$:*

$$d(G,r,\ell) \leq \left(1 - \frac{d}{(d+1)^{(d+1)/d}}\right)^{-1} d(H,r,\ell).$$

We next turn to designing approximation algorithms that compute a "good" subgraph H of G with a provable approximation guarantee.

Mini-Introduction: Computational Complexity

We briefly review some basics from complexity theory. The exposition here is kept at a rather high-level; the interested reader is referred to, e.g., the book *Computers and Intractability: A Guide to the Theory of NP-Completeness* by Garey and Johnson for more details.

Definition 1.6 (Optimization problem)**.** A cost minimization problem $\mathscr{P} = (\mathscr{I}, \mathscr{S}, c)$ is given by:

- a set of instances $\mathscr{I}$ of $\mathscr{P}$;
- for every instance $I \in \mathscr{I}$ a set of feasible solutions $\mathscr{S}_I$;
- for every feasible solution $S \in \mathscr{S}_I$ a real-valued cost $c(S)$.

The goal is to compute for a given instance $I \in \mathscr{I}$ a solution $S \in \mathscr{S}_I$ that minimizes $c(S)$. We use opt_I to refer to the cost of an optimal solution for I.

Definition 1.7 (Decision problem)**.** A decision problem $\mathscr{P} = (\mathscr{I}, \mathscr{S}, c, k)$ is given by:

- a set of instances $\mathscr{I}$ of $\mathscr{P}$;
- for every instance $I \in \mathscr{I}$ a set of feasible solutions $\mathscr{S}_I$;
- for every feasible solution $S \in \mathscr{S}_I$ a real-valued cost $c(S)$.

The goal is to decide whether for a given instance $I \in \mathscr{I}$ a solution $S \in \mathscr{S}_I$ exists such that the cost $c(S)$ of S is at most k. If there exists such a solution, we say that I is a "yes" instance; otherwise, I is a "no" instance.

Example 1.3 (Traveling salesman problem)**.** We are given an undirected graph $G = (V, E)$ with edge costs $c : E \to \mathbb{R}_+$. The *traveling salesman problem (TSP)* asks for the computation of a tour that visits every vertex exactly once and has minimum total cost. The decision problem asks for the computation of a tour of cost at most k.

Several optimization problems (and their respective decision problems) are hard in the sense that there are no polynomial-time algorithms known that solve the problem exactly. Here *polynomial-time algorithm* refers to an algorithm whose running time can be bound by a polynomial function in the *size* of the input instance. For example, an algorithm has polynomial running time if for every input of size n its running time is bound by n^k for some constant k. There are different ways to encode an input instance. Subsequently, we assume that the input is encoded in binary and the *size* of the input instance refers to the number of bits that one needs to represent the instance.

Definition 1.8 (Complexity classes P and NP)**.** A decision problem $\mathscr{P} = (\mathscr{I}, \mathscr{S}, c, k)$ belongs to the complexity class P (which stands for *polynomial time*) if for every instance

$I \in \mathscr{I}$ one can find in polynomial time a feasible solution $S \in \mathscr{S}_I$ whose cost is at most k, or determine that no such solution exists.

A decision problem $\mathscr{P} = (\mathscr{I}, \mathscr{S}, c, k)$ belongs to the complexity class *NP* (which stands for *non-deterministic polynomial time*) if for every instance $I \in \mathscr{I}$ one can verify in polynomial time whether a given solution S is feasible, i.e., $S \in \mathscr{S}_I$, and has a cost of at most k, i.e., $c(S) \leq k$.

Clearly, $P \subseteq NP$. The question whether $P \neq NP$ is still unresolved and one of the biggest open questions to date.[1]

Definition 1.9 (Polynomial time reduction)**.** A decision problem $\mathscr{P}_1 = (\mathscr{I}_1, \mathscr{S}_1, c_1, k_1)$ is *polynomial time reducible* to a decision problem $\mathscr{P}_2 = (\mathscr{I}_2, \mathscr{S}_2, c_2, k_2)$ if every instance $I_1 \in \mathscr{I}_1$ of $\mathscr{P}_1$ can in polynomial time be mapped to an instance $I_2 \in \mathscr{I}_2$ of $\mathscr{P}_2$ such that: I_1 is a "yes" instance of $\mathscr{P}_1$ if and only if I_2 is a "yes" instance of $\mathscr{P}_2$.

Definition 1.10 (*NP*-completeness)**.** A problem $\mathscr{P} = (\mathscr{I}, \mathscr{S}, c, k)$ is *NP-complete* if

- $\mathscr{P}$ belongs to *NP*;
- every problem in *NP* is polynomial time reducible to $\mathscr{P}$.

Essentially, problems that are *NP*-complete are polynomial time equivalent: If we are able to solve one of these problems in polynomial time then we are able to solve all of them in polynomial time. Note that in order to show that a problem is *NP*-complete, it is sufficient to show that it is in *NP* and that an *NP*-complete problem is polynomial time reducible to this problem.

Example 1.4 (TSP)**.** The problem of deciding whether a traveling salesman tour of cost at most k exists is *NP*-complete.

Many fundamental problems are NP-complete and it is therefore unlikely (though not impossible) that efficient algorithms for solving these problems in polynomial time exist. One therefore often considers approximation algorithms:

Definition 1.11 (Approximation algorithm)**.** An algorithm ALG for a cost minimization problem $\mathscr{P} = (\mathscr{I}, \mathscr{S}, c)$ is called an *α-approximation algorithm* for some $\alpha \geq 1$ if for every given input instance $I \in \mathscr{I}$ of $\mathscr{P}$:

1. ALG computes in polynomial time a feasible solution $S \in \mathscr{S}_I$, and
2. the cost of S is at most α times larger than the optimal cost, i.e., $c(S) \leq \alpha \mathtt{opt}_I$.

α is also called the *approximation factor* or *approximation guarantee* of ALG.

A trivial approximation algorithm (called TRIVIAL subsequently) for the NETWORK DESIGN problem is to simply return the original graph as a solution. Using the above

[1] See also *The Millennium Prize Problems* at http://www.claymath.org/millennium.

corollaries, it follows that TRIVIAL has an approximation guarantee of

$$\left(1 - \frac{d}{(d+1)^{(d+1)/d}}\right)^{-1}$$

for latency functions in $\mathscr{L}_d$.

We will show that the performance guarantee of TRIVIAL is best possible, unless $P = NP$.

Theorem 1.7. *Assuming $P \neq NP$, for every $\varepsilon > 0$ there is no $(\frac{4}{3} - \varepsilon)$-approximation algorithm for the* NETWORK DESIGN *problem.*

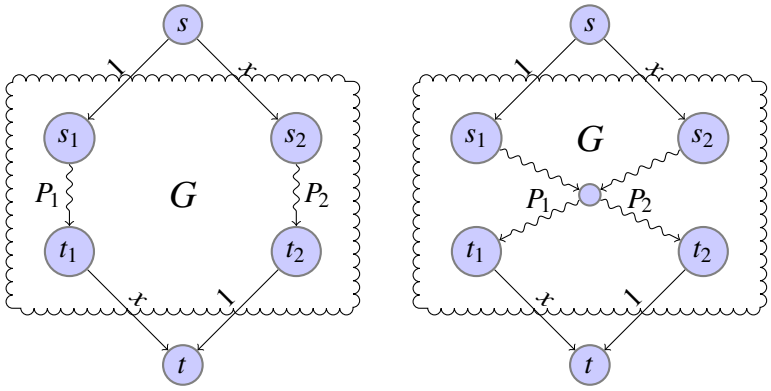

Figure 6: (a) "Yes" instance of 2DDP. (b) "No" instance of 2DDP.

Proof. We reduce from the *2-directed vertex-disjoint paths problem (*2DDP*)*, which is *NP*-complete. An instance of this problem is given by a directed graph $G = (V,A)$ and two vertex pairs (s_1,t_1), (s_2,t_2). The question is whether there exist a path P_1 from s_1 to t_1 and a path P_2 from s_2 to t_2 in G such that P_1 and P_2 are vertex disjoint. We will show that a $(\frac{4}{3} - \varepsilon)$-approximation algorithm could be used to differentiate between "yes" and "no" instances of 2DDP in polynomial time.

Suppose we are given an instance $\mathscr{I}$ of 2DDP. We construct a graph G' by adding a super source s and a super sink t to the network. We connect s to s_1 and s_2 and t_1 and t_2 to t, respectively. The latency functions of the added arcs are given as indicated in Figure 1.8, where we assume that all latency functions in the original graph G are set to zero. This can be done in polynomial time.

We will prove the following two statements:

(i) If $\mathscr{I}$ is a "yes" instance of 2DDP then $d(H,1,\ell) = 3/2$ for some subgraph $H \subseteq G'$.

(ii) If $\mathscr{I}$ is a "no" instance of 2DDP then $d(H,1,\ell) \geq 2$ for every subgraph $H \subseteq G'$.

Suppose for the sake of a contradiction that a $(\frac{4}{3} - \varepsilon)$-approximation algorithm ALG for the NETWORK DESIGN problem exists. ALG then computes in polynomial time a subnetwork $H \subseteq G'$ such that the cost of a Nash flow in H is at most $(\frac{4}{3} - \varepsilon)\mathsf{opt}$, where $\mathsf{opt} = \min_{H \subseteq G'} d(H,r,\ell)$. That is, the cost of a Nash flow for the subnetwork H computed

by ALG is less than 2 for instances in (i) and it is at least 2 for instances in (ii). Thus, using ALG we can determine in polynomial time whether $\mathscr{I}$ is a "yes" or "no" instance, which is a contradiction to the assumption that $P \neq NP$. It remains to show the above two statements.

For (i), we simply delete all arcs in G that are not contained in P_1 and P_2. Then, splitting the flow evenly along these paths yields a Nash equilibrium with cost $d(H,1,\ell) = 3/2$.

For (ii), we can assume without loss of generality that any subgraph H contains an s,t-path. If H has an (s,s_2,t_1,t) path then routing the flow along this path yields a Nash flow with cost $d(H,1,\ell) = 2$. Suppose H does not contain an (s,s_2,t_1,t) path. Because $\mathscr{I}$ is a "no" instance, we have three possibilities:

1. H contains an (s,s_1,t_1,t) path but no (s,s_2,t_2,t) paths (otherwise two such paths must share a vertex and H would contain an (s,s_2,t_1,t) path);
2. H contains an (s,s_2,t_2,t) path but no (s,s_1,t_1,t) path (otherwise two such paths must share a vertex and H would contain an (s,s_2,t_1,t) path);
3. every s,t-path in H is an (s,s_1,t_2,t) path.

It is not hard to verify that in either case, the cost of a Nash flow is $d(H,1,\ell) = 2$. □

2 Potential Games

In this section, we consider so-called *potential games* which constitutes a large class of strategic games having some nice properties. We will address issues like the existence of pure Nash equilibria, price of stability, price of anarchy and computational aspects.

2.1 Connection Games

As a motivating example, we first consider the following *connection game*.

Definition 2.1. A *connection game* $\Gamma = (G = (V,A), (c_a)_{a\in A}, N, (s_i,t_i)_{i\in N})$ is given by

- a directed graph $G = (V,A)$;
- non-negative arc costs $c : A \to \mathbb{R}_+$;
- a set of players $N := [n]$;
- for every player $i \in N$ a terminal pair $(s_i,t_i) \in V \times V$.

The goal of each player $i \in N$ is to connect his terminal vertices s_i,t_i by buying a directed path P_i from s_i to t_i at smallest possible cost. Let $S = (P_1,\dots,P_n)$ be the paths chosen by all players. The cost of an arc $a \in A$ is shared equally among the players that use this arc. That is, the total cost that player i experiences under strategy profile S is

$$c_i(S) := \sum_{a\in P_i} \frac{c_a}{n_a(S)},$$

where

$$n_a(S) = |\{i \in N \,:\, a \in P_i\}|.$$

Let $A(S)$ be the set of arcs that are used with respect to S, i.e., $A(S) := \cup_{i\in N} P_i$. The social cost of a strategy profile S is given by the sum of all arc costs used by the players:

$$C(S) := \sum_{a\in A(S)} c_a = \sum_{i\in N} c_i(S).$$

Example 2.1. Consider the connection game in Figure 7 (a). There are two Nash equilibria: One in which all players choose the left arc and one in which all players choose the right arc. Certainly, the optimal solution is to assign all players to the left arc. The example shows that the price of anarchy can be as large as n.

Example 2.2. Consider the connection game in Figure 7 (b). Here the unique Nash equilibrium is that every player uses his direct arc to the target vertex. The resulting cost is

$$H_n := 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n},$$

which is called the *n-th harmonic number*. (H_n is about $\log(n)$ for large enough n.) An optimal solution allocates all players to the $1+\varepsilon$ path. The example shows that the cost of a Nash equilibrium can be a factor H_n away from the optimal cost.

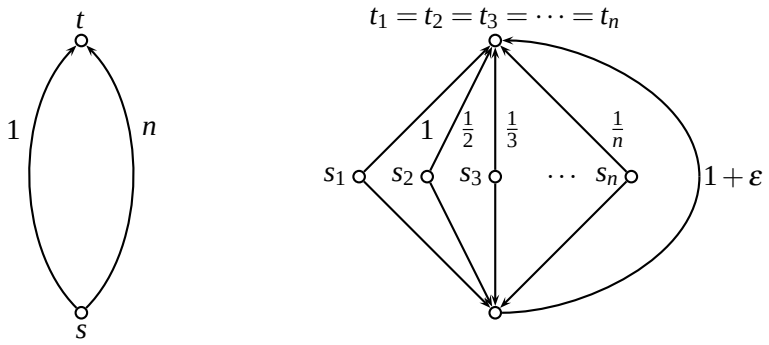

Figure 7: Examples of connection games showing that (a) Nash equilibria are not unique and (b) the price of stability is at least H_n.

Consider the following *potential function* Φ that maps every strategy profile $S = (P_1, \ldots, P_n)$ of a connection game to a real value:

$$\Phi(S) := \sum_{a \in A} c_a \left(1 + \frac{1}{2} + \cdots + \frac{1}{n_a(S)}\right) = \sum_{a \in A} c_a H_{n_a(S)}.$$

We derive some properties of $\Phi(S)$.

Lemma 2.1. *Consider an instance $\Gamma = (G, (c_a)_{a \in A}, N, (s_i, t_i)_{i \in N})$ of the connection game. We have for every strategy profile $S = (P_1, \ldots, P_n)$:*

$$C(S) \leq \Phi(S) \leq H_n C(S).$$

Proof. Recall that $A(S)$ refers to the set of arcs that are used in S. We first observe that $H_{n_a(S)} = 0$ for every arc $a \in A \setminus A(S)$ since $n_a(S) = 0$. Next observe that for every arc $a \in A(S)$ we have $c_a \leq c_a H_{n_a(S)} \leq c_a H_n$. Summing over all arcs concludes the proof. □

For a given strategy profile $S = (P_1, \ldots, P_n)$ we use (S_{-i}, P_i') to refer to the strategy profile that we obtain from S if player i deviates to path P_i', i.e.,

$$(S_{-i}, P_i') = (P_1, \ldots, P_{i-1}, P_i', P_{i+1}, \ldots, P_n).$$

The next lemma shows that the potential function reflects exactly the change in cost of a player if he deviates to an alternative strategy.

Lemma 2.2. *Consider an instance $\Gamma = (G, (c_a)_{a \in A}, N, (s_i, t_i)_{i \in N})$ of the connection game and let $S = (P_1, \ldots, P_n)$ be a strategy profile. Fix a player $i \in N$ and let $P_i' \neq P_i$ be an alternative s_i, t_i-path. Consider the strategy profile $S' = (S_{-i}, P_i')$ that we obtain if player i deviates to P_i'. Then*

$$\Phi(S') - \Phi(S) = c_i(S') - c_i(S)$$

Proof. Note that for every $a \notin P_i \cup P_i'$ we have $n_a(S') = n_a(S)$. Moreover, for every $a \in P_i \cap P_i'$ we have $n_a(S') = n_a(S)$. We thus have

$$\begin{aligned}
\Phi(S') - \Phi(S) &= \sum_{a \in A} c_a H_{n_a(S')} - \sum_{a \in A} c_a H_{n_a(S)} \\
&= \sum_{a \in P_i' \setminus P_i} c_a (H_{n_a(S')} - H_{n_a(S)}) - \sum_{a \in P_i \setminus P_i'} c_a (H_{n_a(S)} - H_{n_a(S')}) \\
&= \sum_{a \in P_i' \setminus P_i} c_a (H_{n_a(S)+1} - H_{n_a(S)}) - \sum_{a \in P_i \setminus P_i'} c_a (H_{n_a(S)} - H_{n_a(S)-1}) \\
&= \sum_{a \in P_i' \setminus P_i} \frac{c_a}{n_a(S)+1} - \sum_{a \in P_i \setminus P_i'} \frac{c_a}{n_a(S)} = c_i(S') - c_i(S).
\end{aligned}$$

□

We will see in the next section that the above two lemmas imply the following theorem.

Theorem 2.1. *Let $\Gamma = (G, (c_a)_{a \in A}, N, (s_i, t_i)_{i \in N})$ be an instance of the connection game. Then Γ has a pure Nash equilibrium and the price of stability is at most H_n, where n is the number of players.*

2.2 Potential games

The above connection game is a special case of the general class of *potential games*, which we formalize next.

Definition 2.2. A finite strategic game $\Gamma = (N, (X_i)_{i \in N}, (u_i)_{i \in N})$ is given by

- a finite set $N = [n]$ of players;
- for every player $i \in N$, a finite set of strategies X_i;
- for every player $i \in N$, a utility function $u_i : X \to \mathbb{R}$ which maps every strategy profile $x \in X := X_1 \times \cdots \times X_n$ to a real-valued utility $u_i(x)$.

The goal of every player is to choose a strategy $x_i \in X_i$ so as to maximize his own utility $u_i(x)$.

A strategy profile $x = (x_1, \ldots, x_n) \in X$ is a *pure Nash equilibrium* if for every player $i \in N$ and every strategy $y_i \in X_i$, we have

$$u_i(x) \geq u_i(x_{-i}, y_i).$$

Here x_{-i} denotes the strategy profile $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ excluding player i. Moreover, $(x_{-i}, y_i) = (x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$ refers to the strategy profile that we obtain from x if player i deviates to strategy y_i.

In general, Nash equilibria are not guaranteed to exist in strategic games. Suppose x is not a Nash equilibrium. Then there is at least one player $i \in N$ and a strategy $y_i \in X_i$ such that

$$u_i(x) < u_i(x_{-i}, y_i).$$

Algorithmus 1 IMPROVING MOVES

Input: arbitrary strategy profile $x \in X$
Output: Nash equilibrium x^*

1: $x^0 := x$
2: $k := 0$
3: **while** x^k is not a Nash equilibrium **do**
4: determine a player $i \in N$ and $y_i \in X_i$, such that $u_i(x^k_{-i}, y_i) > u_i(x^k)$
5: $x^{k+1} := (x^k_{-i}, y_i)$
6: $k := k+1$
7: **end while**
8: **return** $x^* := x^k$

We call the change from strategy x_i to y_i of player i an *improving move*.

A natural approach to determine a Nash equilibrium is as follows: Start with an arbitrary strategy profile $x^0 = x$. As long as there is an improving move, execute this move. The algorithm terminates if no improving move can be found. Let the resulting strategy profile be denoted by x^*. A formal description of the algorithm is given in Algorithm 1. Clearly, the algorithm computes a pure Nash equilibrium if it terminates.

Definition 2.3. We associate a directed *transition graph* $G(\Gamma) = (V,A)$ with a finite strategic game $\Gamma = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$ as follows:

- every strategy profile $x \in X$ corresponds to a unique node of the transition graph $G(\Gamma)$;
- there is a directed edge from strategy x to $y = (x_{-i}, y_i)$ in $G(\Gamma)$ iff the change from x_i to y_i corresponds to an improving move of player $i \in N$.

Note that the transition graph is finite since the set of players N and the strategy set X_i of every player are finite. Every directed path $P = (x^0, x^1, \dots)$ in the transition graph corresponds to a sequence of improving moves. We therefore call P an *improvement path*. We call x^0 the *starting configuration* of P. If P is finite its last node is called the *terminal configuration*.

Definition 2.4. A strategic game $\Gamma = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$ has the *finite improvement property (FIP)* if every improvement path in the transition graph $G(\Gamma)$ is finite.

Consider the execution of IMPROVING MOVES. The algorithm computes an improving path $P = (x^0, x^1, \dots)$ with starting configuration x^0 and is guaranteed to terminate if Γ has the FIP. That is, Γ admits a pure Nash equilibrium if it has the FIP. In order to characterize games that have the FIP, we introduce *potential games*.

Definition 2.5. A finite strategic game $\Gamma = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$ is called *exact potential game* if there exists a function (also called *potential function*) $\Phi : X \to \mathbb{R}$ such that for every player $i \in N$ and for every $x_{-i} \in X_{-i}$ and $x_i, y_i \in X_i$:

$$u_i(x_{-i}, y_i) - u_i(x_{-i}, x_i) = \Phi(x_{-i}, x_i) - \Phi(x_{-i}, y_i).$$

Γ is an *ordinal potential game* if for every player $i \in N$ and for every $x_{-i} \in X_{-i}$ and $x_i, y_i \in X_i$:

$$u_i(x_{-i}, y_i) - u_i(x_{-i}, x_i) > 0 \quad \Leftrightarrow \quad \Phi(x_{-i}, x_i) - \Phi(x_{-i}, y_i) > 0.$$

Γ is a *generalized ordinal potential game* if for every player $i \in N$ and for every $x_{-i} \in X_{-i}$ and $x_i, y_i \in X_i$:

$$u_i(x_{-i}, y_i) - u_i(x_{-i}, x_i) > 0 \quad \Rightarrow \quad \Phi(x_{-i}, x_i) - \Phi(x_{-i}, y_i) > 0.$$

2.2.1 Existence of Nash Equilibria

Theorem 2.2. *Let $\Gamma = (N, (X_i)_{i \in N}, (u_i)_{i \in N})$ be an ordinal potential game. The set of pure Nash equilibria of Γ coincides with the set of local minima of Φ, i.e., x is a Nash equilibrium of Γ iff*

$$\forall i \in N,\ \forall y_i \in X_i : \qquad \Phi(x) \leq \Phi(x_{-i}, y_i).$$

Proof. The proof follows directly from the definition of ordinal potential games. □

Theorem 2.3. *Every generalized ordinal potential game Γ has the FIP. In particular, Γ admits a pure Nash equilibrium.*

Proof. Consider an improvement path $P = (x^0, x^1, \dots)$ in the transition graph $G(\Gamma)$. Since Γ is a generalized ordinal potential game, we have

$$\Phi(x^0) > \Phi(x^1) > \dots$$

Because the transition graph has a finite number of nodes, the path P must be finite. Thus, Γ has the FIP. The existence follows now directly from the FIP and the IMPROVING MOVES algorithm. □

One can show the following equivalence (we omit the proof here).

Theorem 2.4. *Let Γ be a finite strategic game. Γ has the FIP if and only if Γ admits a generalized ordinal potential function.*

2.2.2 Price of Stability

Consider an instance $\Gamma = (N, (X_i)_{i \in N}, (u_i)_{i \in N})$ of a potential game and suppose we are given a social cost function $c : X \to \mathbb{R}$ that maps every strategy profile $x \in X$ to some cost $c(x)$. We assume that the global objective is to minimize $c(x)$ over all $x \in X$. (The definitions are similar if we want to maximize $c(x)$.) Let $\mathsf{opt}(\Gamma)$ refer to the minimum cost of a strategy profile $x \in X$ and let $\mathsf{NE}(\Gamma)$ refer to the set of strategy profiles that are Nash equilibria of Γ.

The *price of stability* is defined as the worst case ratio over all instances of the game of the cost of a best Nash equilibrium over the optimal cost; more formally,

$$\mathsf{POS} := \max_{\Gamma} \min_{x \in \mathsf{NE}(\Gamma)} \frac{c(x)}{\mathsf{opt}(\Gamma)}.$$

In contrast, the *price of anarchy* is defined as the worst case ratio over all instances of the game of the cost of a worst Nash equilibrium over the optimal cost; more formally,

$$\mathsf{POA} := \max_{\Gamma} \max_{x \in \mathsf{NE}(\Gamma)} \frac{c(x)}{\mathsf{opt}(\Gamma)}.$$

Theorem 2.5. *Consider a potential game $\Gamma = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$ with potential function Φ. Let $c : X \to \mathbb{R}_+$ be a social cost function. If Φ satisfies for every $x \in X$:*

$$\frac{1}{\alpha} c(x) \leq \Phi(x) \leq \beta c(x)$$

for some $\alpha, \beta > 0$, then the price of stability is at most $\alpha\beta$.

Proof. Let x be a strategy profile that minimizes Φ. Then x is a Nash equilibrium by Theorem 2.2. Let x^* be an optimal solution of cost $\mathsf{opt}(\Gamma)$. Note that

$$\Phi(x) \leq \Phi(x^*) \leq \beta c(x^*) = \beta \mathsf{opt}(\Gamma).$$

Moreover, we have $c(x) \leq \alpha\Phi(x)$, which concludes the proof. □

2.2.3 Characterization of Exact Potential Games

We show that every exact potential game can be decomposed into a *coordination game* and a *coordination game*.

Definition 2.6. A strategic game $\Gamma = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$ is a

- *coordination game* if there exists a function $u : X \to \mathbb{R}$ such that $u_i = u$ for every $i \in N$ (all players have the same utility function);
- *dummy game* if for every $i \in N$, every $x_{-i} \in X_{-i}$ and every $x_i, y_i \in X_i$: $u_i(x_{-i}, x_i) = u_i(x_{-i}, y_i)$ (each player's utility is independent of his own strategy choice).

Theorem 2.6. *Let $\Gamma = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$ be a finite strategic game. Γ is an exact potential game if and only if there exist functions $(c_i)_{i\in N}$ and $(d_i)_{i\in N}$ such that*

- *$u_i = c_i + d_i$ for all $i \in N$;*
- *$(N, (X_i)_{i\in N}, (c_i)_{i\in N})$ is a coordination game;*
- *$(N, (X_i)_{i\in N}, (d_i)_{i\in N})$ is a dummy game.*

Proof. Let $(c_i)_{i\in N}$ and $(d_i)_{i\in N}$ satisfy the statement of the theorem. We can then define a potential function

$$\Phi(x) := -\sum_{i\in N} c_i(x).$$

Fix an arbitrary strategy profile $x \in X$ and a player $i \in N$. Then for every $y_i \in X_i$, we have

$$u_i(x_{-i}, y_i) - u_i(x) = c_i(x_{-i}, y_i) - c_i(x) + d_i(x_{-i}, y_i) - d_i(x) = \Phi(x) - \Phi(x_{-i}, y_i),$$

where the last equality holds because $(N, (X_i)_{i\in N}, (d_i)_{i\in N})$ is a dummy game. That is, Γ is an exact potential game.

Let Φ be an exact potential function for Γ. For every player $i \in N$ we have $u_i(x) = (u_i(x) + \Phi(x)) - \Phi(x)$. Clearly, $(N, (X_i)_{i\in N}, (-\Phi)_{i\in N})$ is a coordination game. Fix some player $i \in N$ and $x_{-i} \in X_{-i}$. Since Γ is an exact potential game, we have for every $x_i, y_i \in X_i$

$$\begin{aligned} & u_i(x_{-i}, y_i) - u_i(x_{-i}, x_i) = \Phi(x_{-i}, x_i) - \Phi(x_{-i}, y_i) \\ \Leftrightarrow \quad & u_i(x_{-i}, y_i) + \Phi(x_{-i}, y_i) = u_i(x_{-i}, x_i) + \Phi(x_{-i}, x_i). \end{aligned}$$

Thus, $(N, (X_i)_{i\in N}, (u_i + \Phi)_{i\in N})$ is a dummy game. □

2.2.4 Computing Nash Equilibria

We next consider the problem of computing a pure Nash equilibrium of an exact potential game $\Gamma = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$. The question of whether or not a Nash equilibrium can be computed in polynomial time is still open. We will relate the complexity of finding Nash equilibria for potential games to the complexity of computing local optima for local search problems. In particular, we will show that the problem of computing Nash equilibria is *PLS-complete*, where PLS stands for *polynomial local search*. PLS-complete problems constitute a large class of search problems for which (so far) no polynomial time algorithms are known.

We first define local search problems:

Definition 2.7. A *local search problem* Π is given by

- a set of instances $\mathscr{I}$;
- for every instance $I \in \mathscr{I}$:
  - a set $F(I)$ of feasible solutions;
  - a cost function $c : F(I) \to \mathbb{Z}$ that maps every feasible solution $S \in F(I)$ to some value $c(S)$;
  - for every feasible solution $S \in F(I)$, a neighborhood $N(S, I) \subseteq F(I)$ of S.

The goal is to find a feasible solution $S \in F(I)$ that is a local minimum, i.e., $c(S) \leq c(S')$ for every $S' \in N(S, I)$.

We associate a *transition graph* with an instance $I \in \mathscr{I}$ of a local search problem Π: Every solution $S \in F(I)$ corresponds to a unique node $v(S)$ and there is a directed arc from $v(S_1)$ to $v(S_2)$ if and only if $S_2 \in N(S_1, I)$ and $c(S_2) < c(S_1)$. The sinks of this graph are the local optima of Π.

The problem of finding a Nash equilibrium of an exact potential game $\Gamma = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$ with potential function Φ can be formulated naturally as a local search problem: The set of feasible solutions corresponds to the set of possible strategy profiles X and the objective function is the potential function Φ. The neighborhood of a strategy profile $x \in X$ refers to the set of all possible strategy profiles that are obtainable from x

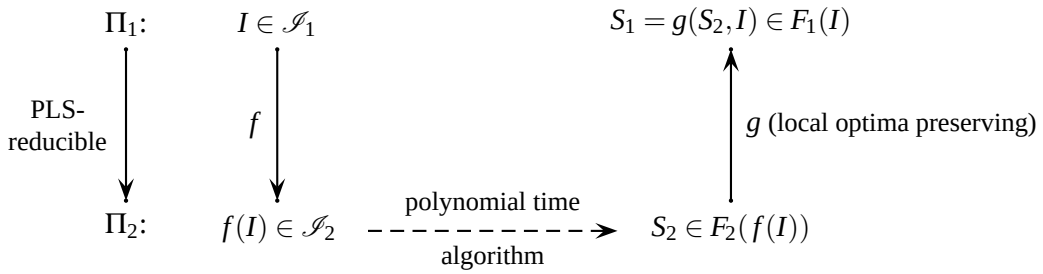

Figure 8: Illustration of PLS-reduction. A polynomial time algorithm for solving Π_2 gives rise to a polynomial time algorithm for solving Π_1.

by single-player deviations. The local minima of the resulting local search problem corresponds exactly to the set of Nash equilibria of Γ (see Theorem 2.2). Note that the transition graph $G(\Gamma)$ as defined in Definition 2.3 coincides with the transition graph of the resulting local search problem.

Definition 2.8. A local search problem Π belongs to the complexity class *PLS (polynomial local search)* if the following can be done in polynomial time for every given instance $I \in \mathscr{I}$:

- compute an initial feasible solution $S \in F(I)$;
- compute the objective value $c(S)$ for every solution $S \in F(I)$;
- determine for every feasible solution $S \in F(I)$ whether S is locally optimal or not and, if not, find a better solution S' in the neighborhood of S, i.e., some $S' \in N(S,I)$ with $c(S') < c(S)$.

It is not hard to see that the problem of computing a Nash equilibrium for potential games is in PLS.

We next define the concept of *PLS-reducibility* (see also Figure 8):

Definition 2.9. Let $\Pi_1 = (\mathscr{I}_1, F_1, c_1, N_1)$ and $\Pi_2 = (\mathscr{I}_2, F_2, c_2, N_2)$ be two local search problems in PLS. Π_1 is *PLS-reducible* to Π_2 if there are two polynomial time computable functions f and g such that

- f maps every instance $I \in \mathscr{I}_1$ of Π_1 to an instance $f(I) \in \mathscr{I}_2$ of Π_2;
- g maps every tuple (S_2, I) with $S_2 \in F_2(f(I))$ to a solution $S_1 \in F_1(I)$;
- for all $I \in \mathscr{I}_1$: if S_2 is a local optimum of $f(I)$, then $g(S_2, I)$ is a local optimum of I.

Definition 2.10. A local search problem Π is *PLS-complete* if

- Π belongs to the complexity class PLS;
- every problem in PLS is PLS-reducible to Π.

The above definitions imply the following: If there is a polynomial time algorithm that computes a local optimum for a PLS-complete problem Π, then there exists a polynomial time algorithm for finding a local optimum for every problem in PLS. This holds since every problem in PLS can be reduced to Π in polynomial time (see Figure 8) and the reduction preserves local optima.

We will show that computing a Nash equilibrium for exact potential games is PLS-complete. We do so by a reduction from the *weighted satisfiability problem*.

Example 2.3. The *weighted satisfiability problem* is given as follows. We are given a formula in conjunctive normal form:

$$f = C_1 \wedge C_2 \wedge \cdots \wedge C_k,$$

where each clause C_j, $j \in [k]$, is a disjunction of literals. (Example: $(y_1 \vee y_2) \wedge (\bar{y}_1 \vee y_3)$.) We assume that f consists of n variables $y_1, \ldots, y_n$. Every clause C_j, $j \in [k]$, has a non-negative weight w_j. A feasible solution is an assignment $y = (y_1, \ldots, y_n) \in \{0,1\}^n$ of $0/1$-values to the variables. The total cost $c(y)$ of an assignment y is given by the sum of the weights of the clauses that are *false* with respect to y. Define the neighborhood of an assignment y as the set of assignments that are obtainable from y by changing a single variable y_i, $i \in [n]$, from 0 to 1 or vice versa. The problem is to determine an assignment y that is a local minimum with respect to c.

Clearly, the weighted satisfiability problem belongs to PLS. Moreover, the problem is PLS-complete.

Theorem 2.7. *The problem of finding a pure Nash equilibrium for exact potential games is PLS-complete.*

Proof. We observed above that the problem belongs to PLS. We reduce the weighted satisfiability problem to the problem of finding a Nash equilibrium in an exact potential game.

Consider an instance of the weighted satisfiability problem

$$f = C_1 \wedge C_2 \wedge \cdots \wedge C_k$$

with n variables $y_1, \ldots, y_n$ and weight w_j for clause C_j, $j \in [k]$.

We derive a strategic game $\Gamma = (N, (X_i)_{i \in N}, (u_i)_{i \in N})$ from this instance as follows: We associate a player $i \in N := [n]$ with every variable y_i of f. Imagine that every clause C_j, $j \in [k]$, corresponds to a unique resource j and that each player can allocate certain subsets of these resources: For player $i \in N$, define

$$J(i) := \{j \in [k] \ : \ y_i \text{ occurs in } C_j\} \quad \text{and} \quad \bar{J}(i) := \{j \in [k] \ : \ \bar{y}_i \text{ occurs in } C_j\}$$

as the resource sets that correspond to clauses that contain the literals y_i and $\bar{y}_i$, respectively. The strategy set X_i of player $i \in N$ consists of two strategies: $X_i = \{J(i), \bar{J}(i)\}$. Our interpretation will be that $y_i = 0$ if player $i \in N$ chooses strategy $x_i = J(i)$, while $y_i = 1$ if player $i \in N$ chooses $x_i = \bar{J}(i)$.

The crucial observation is that a clause C_j, $j \in [k]$, with l_j literals is false iff there are exactly l_j players that have chosen resource j. For a given strategy profile $x \in X$, let $n_j(x)$ refer to the number of players that have chosen resource $j \in [k]$, i.e.,

$$n_j(x) := |\{i \in N \,:\, j \in x_i\}|.$$

Assign every resource $j \in [k]$ a cost $c_j(x)$ as follows:

$$c_j(x) := \begin{cases} 0 & \text{if } n_j(x) < l_j \\ w_j & \text{otherwise.} \end{cases}$$

Each player's goal is to minimize the total cost of the resources he allocates. That is, the utility $u_i(x)$ of player $i \in N$ (which he wants to maximize) is defined as $u_i(x) := -\sum_{j \in x_i} c_j(x)$. This reduction can be done in polynomial time. Moreover, every assignment of the weighted satisfiability problem can be mapped in polynomial time to a corresponding strategy profile of the resulting strategic game Γ and vice versa.

We show that

$$\Phi(x) := \sum_{j \in [k]} c_j(x)$$

is an exact potential function for Γ. To see this note that $\Phi(x)$ is equal the total cost $c(y)$ of the corresponding variable assignment y. Moreover, $-u_i(x)$ accounts for the total cost of all false clauses that contain variable y_i (either negated or not). If player i changes his strategy from x_i to x_i' we therefore have:

$$u_i(x_{-i}, x_i') - u_i(x_{-i}, x_i) = \Phi(x_{-i}, x_i) - \Phi(x_{-i}, x_i').$$

That is, Φ is an exact potential function. By Theorem 2.2, the local minima of Φ correspond exactly to the set of pure Nash equilibria of Γ. The described reduction preserves local optima and thus all conditions of Definition 2.9 are met. This concludes the proof. □

3 Congestion Games

In this section, we consider a general class of resource allocation games, called *congestion games*.

Definition 3.1 (Congestion model)**.** A *congestion model* $\mathscr{M} = (N, F, (X_i)_{i\in N}, (c_f)_{f\in F})$ is given by

- a set of players $N = [n]$;
- a set of facilities F;
- for every player $i \in N$, a set $X_i \subseteq 2^F$ of subsets of facilities in F;[2]
- for every facility $f \in F$, a cost function $c_f : \mathbb{N} \to \mathbb{R}$.

For every player $i \in N$, X_i is the strategy set from which i can choose. A strategy $x_i \in X_i$ is a subset of facilities; we think of x_i as the facilities that player i uses. Fix some strategy profile $x = (x_1, \ldots, x_n) \in X := X_1 \times \cdots \times X_n$. The cost incurred for the usage of facility $f \in F$ with respect to x is defined as $c_f(n_f(x))$, where

$$n_f(x) := |\{i \in N \,:\, f \in x_i\}|$$

refers to the total number of players that use f.

Definition 3.2 (Congestion game)**.** The *congestion game* corresponding to the congestion model $\mathscr{M} = (N, F, (X_i)_{i\in N}, (c_f)_{f\in F})$ is the strategic game $\Gamma = (N, (X_i)_{i\in N}, (c_i)_{i\in N})$, where every player $i \in N$ wants to minimize his cost

$$c_i(x) = \sum_{f\in x_i} c_f(n_f(x)).$$

(Equivalently, every player wants to maximize his utility $u_i = -c_i$.) The game is called *symmetric* if all players have the same strategy set, i.e., $X_i = Q$ for all $i \in N$ and some $Q \subseteq 2^F$.

Example 3.1 (Atomic network congestion game)**.** The *atomic network congestion game* can be modeled as a congestion game: We are given a directed graph $G = (V, A)$, a single commodity $(s,t) \in V \times V$, and a cost function $c_a : \mathbb{N} \to \mathbb{R}_+$ for every arc $a \in A$. Every player $i \in N$ wants to send one unit of flow from s to t along a single path. The set of facilities is $F := A$ and the strategy set X_i of every player $i \in N$ is simply the set of all directed s,t-paths in G. (Note that the game is symmetric.) The goal of every player $i \in N$ is to choose a path $x_i \in X_i$ so as to minimize his cost

$$c_i(x) := \sum_{a\in x_i} c_a(n_a(x)),$$

[2]For a given set S, we use 2^S to refer to the *power set* of S, i.e., the set of all subsets of S.

where $n_a(x)$ refers to the total number of players using arc a. This example corresponds to a selfish routing game, where every player controls one unit of flow (i.e., we have atomic players) and has to route his flow unsplittably from s to t.

3.1 Equivalence to Exact Potential Games

Theorem 3.1. *Every congestion game $\Gamma = (N, (X_i)_{i\in N}, (c_i)_{i\in N})$ is an exact potential game.*

Proof. Rosenthal's potential function $\Phi : X \to \mathbb{R}$ is defined as

$$\Phi(x) := \sum_{f\in F} \sum_{k=1}^{n_f(x)} c_f(k). \tag{12}$$

We prove that Φ is an exact potential function for Γ. To see this, fix some $x \in X$, a player $i \in N$ and some $y_i \in X_i$. We have

$$\begin{aligned}\Phi(x_{-i}, y_i) &= \sum_{f\in F} \sum_{k=1}^{n_f(x)} c_f(k) + \sum_{f\in y_i\setminus x_i} c_f(n_f(x)+1) - \sum_{f\in x_i\setminus y_i} c_f(n_f(x)) \\ &= \Phi(x) + c_i(x_{-i}, y_i) - c_i(x).\end{aligned}$$

Thus, Φ is an exact potential function. □

By Theorem 2.3 it follows that every congestion game has the FIP and admits a pure Nash equilibrium. Moreover, by Theorem 2.7, the problem of computing a Nash equilibrium in congestion games is PLS-complete.

One can even prove the converse of Theorem 3.1: Every potential game can be transformed into an appropriate congestion game.

Definition 3.3. Let $\Gamma_1 = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$ and $\Gamma_2 = (N, (Y_i)_{i\in N}, (v_i)_{i\in N})$ be two strategic games with player set $N := [n]$. Γ_1 and Γ_2 are *isomorphic* if for every player $i \in N$ there exists a bijection $\phi_i : X_i \to Y_i$ such that for all $x \in X$

$$u_i(x_1, \ldots, x_n) = v_i(\phi_1(x_1), \ldots, \phi_n(x_n)).$$

Lemma 3.1. *Every coordination game is isomorphic to a congestion game.*

Proof. Let $\Gamma_1 = (N, (X_i)_{i\in N}, (u)_{i\in N})$ be a coordination game with n players that want to maximize a common utility function u. Introduce for every strategy profile $x \in X$ a unique facility $f(x)$ and let $F := \{f(x) \,:\, x \in X\}$ be the set of all facilities. We derive a congestion model $\mathscr{M} := (N, F, (Y_i)_{i\in N}, (c_f)_{f\in F})$ from Γ_1 as follows: The strategy set Y_i of player $i \in N$ is defined as

$$Y_i := \{\phi_i(x_i) \,:\, x_i \in X_i\},$$

where $\phi_i(x_i)$ refers to the set of all facilities $f(x)$ that are associated with strategy profiles x in which player i chooses strategy x_i, i.e., for every $x_i \in X_i$

$$\phi_i(x_i) := \{f(x_{-i}, x_i) \,:\, x_{-i} \in X_{-i}\}.$$

Moreover, define the cost $c_{f(x)}$ of a facility $f(x) \in F$ as

$$c_{f(x)}(k) := \begin{cases} u(x) & \text{if } k = n \\ 0 & \text{otherwise.} \end{cases}$$

That is, facility $f(x)$ has cost $u(x)$ if every player in N uses this facility; otherwise, its cost is zero. Let Γ_2 be the congestion game that we obtain from $\mathscr{M}$.

Fix some strategy profile $x = (x_1, \ldots, x_n) \in X$ of Γ_1. We define the bijection ϕ_i of every player $i \in N$ as $x_i \mapsto \phi_i(x_i)$. Note that for every strategy profile $x \in X$

$$\bigcap_{i \in N} \phi_i(x_i) = f(x).$$

Therefore there is exactly one facility, namely $f(x)$, that is used by n players in Γ_2 with respect to the strategy profile $(\phi_1(x_1), \ldots, \phi_n(x_n))$. Thus, we have for every player $i \in N$

$$u(x) = c_{f(x)}(n) = c_i(\phi_1(x_1), \ldots, \phi_n(x_n)),$$

which shows that Γ_1 and Γ_2 are isomorphic. □

Lemma 3.2. *Every dummy game is isomorphic to a congestion game.*

The proof of this lemma is slightly more complicated but similar in flavor to the one above and omitted here.

Theorem 3.2. *Every potential game is isomorphic to a congestion game.*

Proof. By Theorem 2.6, we can decompose the potential game into a coordination game and a dummy game. Apply the above theorems to obtain congestion games Γ_1 and Γ_2 (with disjoint facility sets) that are isomorphic to the coordination and the dummy game, respectively. We can then construct an isomorphic congestion game by taking the union of the facility sets and players' strategies (strategy-wise) of Γ_1 and Γ_2. □

3.2 Price of Anarchy

Define the *social cost* of a strategy profile $x \in X$ as the total cost of all players, i.e.,

$$c(x) := \sum_{i \in N} c_i(x) = \sum_{f \in F} n_f(x) c_f(n_f(x)).$$

As before, we define $\mathsf{opt}(\Gamma) := \min_{x \in X} c(x)$ as the optimal cost for Γ.

We derive an upper bound on the price of anarchy for congestion games with respect to the social cost function c defined above. Here we only consider the case that the cost of every facility $f \in F$ is given as $c_f(k) = k$. The proof extends to arbitrary linear latency functions.

Theorem 3.3. *Let $\mathscr{M} = (N, (X_i)_{i\in N}, (c_i)_{i\in N})$ be a congestion model with linear latency functions $c_f(k) = k$ for every $f \in F$ and let $\Gamma = (N, (X_i)_{i\in N}, (u_i)_{i\in N})$ be the corresponding congestion game. The price of anarchy is at most $5/2$.*

We will use the following fact to prove this theorem (whose proof we leave as an exercise):

Fakt 3.1. *Let α and β be two non-negative integers. Then*

$$\alpha(\beta+1) \leq \frac{5}{3}\alpha^2 + \frac{1}{3}\beta^2.$$

Proof of Theorem 3.3. Let x be a Nash equilibrium and x^* be an optimal strategy profile minimizing c. Since x is a Nash equilibrium, the cost of every player $i \in N$ does not decrease if he deviates to his optimal strategy x_i^*, i.e.,

$$c_i(x) \leq c_i(x_{-i}, x_i^*) = \sum_{f\in x_i^*} c_f(n_f(x_{-i}, x_i^*)) = \sum_{f\in x_i^*} n_f(x_{-i}, x_i^*) \leq \sum_{f\in x_i^*} n_f(x) + 1,$$

where the last inequality follows since player i increases the number of players on each $f \in x_i^*$ by at most 1 with respect to $n_f(x)$. Summing over all players, we obtain

$$c(x) = \sum_{i\in N} c_i(x) \leq \sum_{i\in N} \sum_{f\in x_i^*} n_f(x) + 1 = \sum_{f\in F} n_f(x^*)(n_f(x)+1).$$

Using Fact 3.1, we therefore obtain

$$c(x) \leq \sum_{f\in F} n_f(x^*)(n_f(x)+1) \leq \frac{5}{3}\sum_{f\in F}(n_f(x^*))^2 + \frac{1}{3}\sum_{f\in F}(n_f(x))^2 = \frac{5}{3}c(x^*) + \frac{1}{3}c(x),$$

where the last equality follows from $c_f(k) = k$ for every $f \in F$ and the definition of c. We conclude that $c(x) \leq \frac{5}{2}c(x^*)$. □

4 Combinatorial Auctions

In this section, we present a few examples from the area of *mechanism design*. The fundamental questions that one attempts to address in mechanism design is the following: Assuming that players act strategically, how should we design the rules of the game such that the players' strategic behavior leads to a certain desirable outcome of the game? As a motivating example, we first consider one of the simplest auctions, known as *Vickrey Auction*. We then turn to more general combinatorial auctions.

4.1 Vickrey Auction

Suppose there is an auctioneer who wishes to auction off a single item. An instance of the *single-item auction* consists of

- a set of players $N = [n]$ that are interested in obtaining the item;
- every player $i \in N$ has a *private valuation* v_i which specifies how much the item is worth to player i; v_i is only known to player i.
- every player i has a *bid* b_i which represents the maximum amount player i declares to be willing to pay for the item.

The auctioneer receives the bids and needs to determine who receives the item and at what price. A *mechanism* can be thought of as a protocol (or algorithm) that the auctioneer runs in order to make this decision. That is, based on the submitted bids $(b_i)_{i \in N}$, the mechanism determines

1. a player i^* in N, called the *winner*, who receives the item, and
2. a price p that this player has to pay for the item.

We define $x_i = 1$ if player $i \in N$ wins the auction and $x_i = 0$ otherwise. We model a player's preferences over different outcomes of the game by means of a utility function. Lets assume that the utility function of player i represents the *net gain*, defined as $u_i = x_i(v_i - p)$. Note that the utility is zero if the player does not receive the item. Otherwise, it is his private valuation minus the price he has to pay. Such utility functions are also called *quasi-linear*.

There are several natural properties that we want to achieve:

(P1) *Strategyproofness*: Every player maximizes his utility by bidding *truthfully*, i.e., $b_i = v_i$.

(P2) *Efficiency*: Assuming that every player bids truthfully, the mechanism computes an outcome that maximizes the *social welfare*, i.e., among all possible outcomes x it chooses one that maximizes the total valuation $\sum_{i \in N} x_i v_i$; here, this is equivalent to require that the mechanism chooses the player with maximum valuation as the winner.

(P3) *Polynomial-time computability*: The outcome should be computable in polynomial time.

As it turns out, there is a remarkable mechanism due to Vickrey that satisfies all these properties; this mechanism is also known as *Vickrey auction* or *second-price auction* (see Algorithm 2).

Algorithmus 2 Vickrey Auction

1: Collect the bids $(b_i)_{i \in N}$ of all players.
2: Choose a player $i^* \in N$ with highest bid (break ties arbitrarily).
3: Charge i^* the second highest bid $p := \max_{i \neq i^*} b_i$.

Lemma 4.1. *In a Vickrey Auction, bidding truthfully $b_i = v_i$ is a dominant strategy for every player $i \in N$. More formally, for every player $i \in N$ and every bidding profile b_{-i} of the other players, we have*

$$u_i(b_{-i}, v_i) \geq u_i(b_{-i}, b_i) \quad \forall b_i.$$

Proof. Consider player i and fix a bidding profile b_{-i} of the other players. Let $B = \max_{j \neq i} b_j$ be the highest bid if player i does not participate in the game.

Assume $v_i \leq B$. Then player i has zero utility if he bids truthfully: Note that player i loses if $v_i < B$ and may win if $v_i = B$ (depending on the tie breaking rule); however, in both cases his utility is zero. His utility remains zero for every bid $b_i < B$ or if $b_i = B$ and i loses (due to the tie breaking rule). Otherwise, $b_i = B$ and i wins or $b_i > B$. In both cases i wins and pays B. However, his utility is then $u_i = v_i - B \leq 0$, which is less than or equal to the utility he obtains if he bids truthfully.

Next assume that $v_i > B$. If player i bids truthfully, he wins and receives a positive utility $u_i = v_i - B > 0$. He is worse off by obtaining a utility of zero if he bids $b_i < B$ or if he bids $b_i = B$ and loses (due to the tie breaking rule). Otherwise $b_i = B$ and i wins or $b_i > B$. In both cases, i wins and receives a utility of $u_i = v_i - B > 0$, which is the same as if he had bid $b_i = v_i$. □

It is easy to see that the Vickrey Auction satisfies (P2) and (P3) as well. More specifically, it satisfies (P2) since it selects the winner i^* to be a player whose valuation is maximum, assuming that every bidder bids truthfully. Moreover, its computation time is linear in the number of players n. We can thus summarize:

Theorem 4.1. *The Vickrey Auction is strategyproof, efficient and runs in polynomial time.*

4.2 Combinatorial Auctions and the VCG Mechanism

We now turn to a more general model of auctions. Suppose there is a set M of $m \geq 1$ items to be auctioned off to n players. A player may now be interested in a *bundle* $S \subseteq M$ of items. Every player $i \in N$ has a private valuation function $v_i : 2^M \to \mathbb{R}^+$, where $v_i(S)$ specifies player i's value for receiving the items in $S \subseteq M$. We say $v_i(S)$ is the valuation of player i for bundle S. We assume that $v_i(\emptyset) = 0$. (Although this and the assumption that $v_i(\cdot)$ is non-negative is not essential here).

If every player has a separate value for each item and the value of a subset $S \subseteq M$ is equal to the sum of all values of the items in S, then we can simply run a separate Vickrey Auction for every item. However, this assumption ignores the possibility that differnt bundles may have different values. More precisely, for a player i, items in $S \subseteq M$ might be

Algorithmus 3 VCG mechanism

1: Collect the bids $(b_i(S))$ for every player $i \in N$ and every set $S \subseteq M$.

2: Choose an allocation $a^* \in O$ such that

$$a^* = \arg\max_{a \in O} \sum_{i \in N} b_i(a).$$

3: Compute the price p_i of player i as

$$p_i := b_i(a^*) - \Bigg(\underbrace{\max_{a \in O} \sum_{j \in N} b_j(a) - \max_{a \in O} \sum_{j \in N, j \neq i} b_j(a)}_{i\text{'s contribution to the total social welfare}} \Bigg).$$

4: **return** a^*

- *substitutes*: the player's valuation to obtain the entire bundle S might be less than or equal to the individual valuations of the items in S, i.e., $v_i(S) \leq \sum_{k \in S} v_i(\{k\})$; for example, if the items in S are (partially) redundant.
- *complements*: the player's valuation to obtain the entire bundle S might be greater or equal to the individual valuations of the items in S, i.e., $v_i(S) \geq \sum_{k \in S} v_i(\{k\})$; for example, if the items in S are (partially) dependent.

Here, we consider the most general setting, where we do not make any assumption on the valuation functions v_i of the players.

Let O denote the set of all possible allocations of the items in M to the players. An allocation $a \in O$ is a function $a : M \to N \cup \{\perp\}$ that maps every item to one of the players in N or to $\perp$, which means that the item remains unassigned. Let $a^{-1}(i)$ be the subset of items that player $i \in N$ receives. Every player declares a bid $b_i(S)$ for every bundle $S \subseteq M$. (Lets not care about polynomial-time computability for a moment.) For the sake of conciseness, we slightly abuse notation: Given an allocation $a \in O$, we write $v_i(a)$ and $b_i(a)$ to refer to $v_i(a^{-1}(i))$ and $b_i(a^{-1}(i))$, respectively. The auctioneer needs to decide how to distribute the items among the players in N and at what price. That is, he determines an allocation $a \in O$ and a pricing vector $p = (p_i)_{i \in N}$, where player i obtains the bundle $a^{-1}(i)$ at a price of p_i. As before, we consider quasi-linear utility functions: The utility of player i, given the outcome (a, p), is $u_i = v_i(a) - p_i$.

A mechanism is strategyproof in this setting if a dominant strategy for every player is to bid $b_i(S) = v_i(S)$ for every $S \subseteq M$. Moreover, a mechanism is efficient, if it outputs an allocation a^* that maximizes the total social welfare, i.e., $a^* = \arg\max_{a \in O} \sum_{i \in N} v_i(a)$, assuming that every player truthfully reports his valuation.

A powerful mechanism for this quite general class of combinatorial auctions is known as *VCG mechanism* due to Vickrey, Clarke and Groves (see Algorithm 3). In particular, as we will see, the VCG mechanism is strategyproof and efficient.

Theorem 4.2. *The VCG mechanism is strategyproof and efficient.*

Proof. Clearly, if every player bids truthfully the allocation a^* output by the VCG mechanism maximizes total social welfare. Thus, the VCG mechanism is efficient.

We next prove that the VCG mechanism is strategyproof. Consider an arbitrary player $i \in N$. Let $b = (b_{-i}, b_i)$ be the bid vector of some arbitrary bids and let $\bar{b} = (b_{-i}, v_i)$ be the same bid vector, except that player i reports his private valuations truthfully. Moreover, let (a^*, p) and $(\bar{a}^*, \bar{p}_i)$ be the outcome computed by the VCG mechanism for input b and $\bar{b}$, respectively. Observe that we have

$$\bar{b}_i(a) = v_i(a) \quad \forall a \in O \quad \text{and} \quad \bar{b}_j(a) = b_j(a) \quad \forall j \neq i,\ \forall a \in O. \tag{13}$$

Moreover, $\bar{a}^*$ has been chosen such that

$$\sum_{j \in N} \bar{b}_j(\bar{a}^*) \geq \sum_{j \in N} \bar{b}_j(a) \quad \forall a \in O. \tag{14}$$

Using these two observations, we can infer:

$$\begin{aligned}
v_i(\bar{a}^*) - \bar{p}_i &= v_i(\bar{a}^*) - \left[\bar{b}_i(\bar{a}^*) - \left(\max_{a \in O} \sum_{j \in N} \bar{b}_j(a) - \max_{a \in O} \sum_{j \in N, j \neq i} \bar{b}_j(a)\right)\right] \\
&\overset{(13)}{=} \max_{a \in O} \sum_{j \in N} \bar{b}_j(a) - \max_{a \in O} \sum_{j \in N, j \neq i} b_j(a) \\
&= \sum_{j \in N} \bar{b}_j(\bar{a}^*) - \max_{a \in O} \sum_{j \in N, j \neq i} b_j(a) \\
&\overset{(14)}{\geq} \sum_{j \in N} \bar{b}_j(a^*) - \max_{a \in O} \sum_{j \in N, j \neq i} b_j(a) \\
&\overset{(13)}{=} \sum_{j \in N, j \neq i} b_j(a^*) + v_i(a^*) - \max_{a \in O} \sum_{j \in N, j \neq i} b_j(a) \\
&= v_i(a^*) - \left[b_i(a^*) - \left(\max_{a \in O} \sum_{j \in N} b_j(a) - \max_{a \in O} \sum_{j \in N, j \neq i} b_j(a)\right)\right] \\
&= v_i(a^*) - p_i.
\end{aligned}$$

Thus, $b_i = v_i$ is a dominant strategy for player i. □

Although the VCG mechanism satisfies strategyproofness and efficiency, it is highly computationally intractable. In particular, the mechanism relies crucially on the fact that one can compute an optimal allocation $a^* \in O$. This problem is typically also called the *allocation problem*.

4.3 Single-Minded Bidders

In this section, we consider the special case of a combinatorial auction, where all bidders are said to be *single-minded*. More precisely, we say that player i is single-minded if there

is some (private) set $\Sigma_i \subseteq M$ and a (private) value $\theta_i \geq 0$ such that for every $S \subseteq M$,

$$v_i(S) = \begin{cases} \theta_i & \text{if } S \supseteq \Sigma_i \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, player i is only interested in getting the items in Σ_i (or some more) and its valuation for these items is θ_i. Note that in the single-minded case every player simply reports a pair (S_i, b_i) (not necessarily equal to (Σ_i, θ_i)) to the auctioneer. Thus, the input can be represented compactly and is polynomial in n and m.

The allocation problem for the single-minded case is as follows: Given the bids $\{(S_i, b_i)_{i \in N}\}$, determine a subset $W \subseteq N$ of *winners* such that $S_i \cap S_j = \emptyset$ for every $i, j \in W$, $i \neq j$, with maximum social welfare $\sum_{i \in W} b_i$.

Theorem 4.3. *The allocation problem for single-minded bidders is NP-hard.*

Proof. We give a polynomial-time reduction from the NP-complete problem *independent set*. The independent set problem is as follows: Given an undirected graph $G = (V, E)$ and a non-negative integer k, determine whether there exists an independent set of size at least k.[3]

Given an instance (G, k) of the independent set problem, we can construct a single-minded combinatorial auction as follows: The set of items M corresponds to the edge set E of G. We associate a player $i \in N$ with every vertex $u_i \in V$ of G. The bundle that player i desires corresponds to the set of all adjacent edges, i.e., $S_i := \{e = \{u_i, u_j\} \in E\}$, and the value that i assigns to its bundle S_i is $b_i = 1$.

Now observe that a set $W \subseteq N$ of winners satisfies $S_i \cap S_j = \emptyset$ for every $i \neq j \in W$ iff the set of vertices corresponding to W constitute an independent set in G. Moreover, the social welfare obtained for W is exactly the size of the independet set. □

Given the above hardness result and insisting on polynomial-time computability, we are thus forced to consider approximation algorithms. The idea is to relax the efficiency condition and to ask for an outcome that is (only) approximately efficient. We call a mechanism α-*efficient* for some $\alpha \geq 1$ if it computes an allocation $a \in O$ (assuming truthful bidding $(S_i, b_i) = (\Sigma_i, \theta_i)$ for all $i \in N$) such that

$$\sum_{i \in N} v_i(a) \geq \frac{1}{\alpha} \max_{a \in O} \sum_{i \in N} v_i(a).$$

The proof of Theorem 4.3 even shows that the reduction is *approximation preserving*. That is, it specifies a bijection that preserves the objective function values of the corresponding solutions (of the allocation problem and the independent set problem). It is known that the independent set problem is hard even from an approximation point of view:

Fact 4.1. *For every fixed $\varepsilon > 0$, there is no $O(n^{1-\varepsilon})$-approximation algorithm for the*

[3]Recall that an independent set $I \subseteq V$ of G is a subset of the vertices such that no two vertices in I are connected by an edge.

independent set problem, where n denotes the number of vertices in the graph (unless NP $\subseteq$ ZPP).

Since the number of edges in a (simple) directed graph is at most $O(n^2)$, we obtain the following corollary:

Corollary 4.1. *For every fixed $\varepsilon > 0$, there is no $O(m^{1/2-\varepsilon})$-efficient mechanism for single-minded bidders, where m denotes the number of items (unless NP $\subseteq$ ZPP).*

The following lemma characterizes properties of strategyproof mechanisms for single-minded bidders.

Lemma 4.2. *A mechanism for single-minded bidders is strategyproof if and only if it satisfies the following two conditions:*

1. *Monotonicity: A bidder who wins with bid (S_i, b_i) keeps winning for any $b_i' > b_i$ and for any $S_i' \subset S_i$ (for any fixed bids of the other players).*
2. *Critical value: A bidder who wins pays the minimum value, also called the* critical value*, needed for winning, i.e., the payment is the minimum over all values b_i' such that (S_i, b_i') still wins.*

Proof. We only prove the "if" part of the lemma. We first observe that a truthful bidder will never receive a negative utility: His utility is zero when he loses. In order to win, his valuation $v_i = b_i$ must be at least the critical value, which is exactly his payment.

We next show that a bidder i can never improve his utility by reporting some bid $(S_i, b_i) \neq (\Sigma_i, \theta_i)$. If (S_i, b_i) is a losing bid (zero utility), or if S_i does not contain Σ_i (non-positive utility), then clearly reporting (Σ_i, θ_i) can only help.

Assume that (S_i, b_i) is a winning bid and $S_i \supseteq \Sigma_i$. We first show that i is never worse off by reporting (Σ_i, b_i) instead of (S_i, b_i). Let p_i be the payment for (S_i, b_i) and let p_i' be the payment for (Σ_i, b_i). Note that (Σ_i, b_i) is a winning bid by monotonicity. For every $x < p_i'$, bidding (Σ_i, x) will lose since p_i' is a critical value. By monotonicity, (S_i, x) will also be a losing bid for every $x < p_i'$ and therefore the critical value p_i is at least p_i'. It follows that by bidding (Σ_i, b_i) instead of (S_i, b_i), player i still wins and his payment does not increase.

Next, we show that player i is not worse off by bidding (Σ_i, θ_i) instead of bidding the winning bid (Σ_i, b_i). First suppose that (Σ_i, θ_i) is a winning bid with payment (critical value) $\bar{p}_i$. As long as $b_i \geq \bar{p}_i$, player i still wins by bidding (Σ_i, b_i) (by monotonicity) and receives the same payment (by critical value). If $b_i < \bar{p}_i$, player i loses and receives zero utility. In both cases, misreporting does not increase the utility of player i. Finally, suppose that player i loses by bidding (Σ_i, θ_i). Then θ_i must be smaller than the critical value and thus the payment for the winning bid (Σ_i, b_i) will be greater than θ_i. Therefore, the utility that player i receives by bidding (Σ_i, b_i) is negative. □

We next devise a mechanism that is strategyproof and $\sqrt{m}$-approximate efficient. Thus, from a computational point of view, this is the best we can hope for. The mechanism is the greedy algorithm described in Algorithm 4.

Algorithmus 4 Greedy mechanism for single-minded bidders.

1: Collect the bids $\{(S_i, b_i)_{i \in N}\}$ of all players.
2: Reindex the bids such that

$$\frac{b_1}{\sqrt{|S_1|}} \geq \frac{b_2}{\sqrt{|S_2|}} \geq \cdots \geq \frac{b_n}{\sqrt{|S_n|}}.$$

3: $W \leftarrow \emptyset$
4: **for** $i = 1, \ldots, n$ **do**
5: **if** no items in S_i have been assigned to players in W, i.e., $S_i \cap \left(\bigcup_{j \in W} S_j\right) = \emptyset$ **then**
6: add i to W: $W \leftarrow W \cup \{i\}$.
7: **end if**
8: **end for**
9: **for** each $i \in W$ **do**
10: define i's payment as:

$$p_i := \frac{b_j}{\sqrt{|S_j|}} \cdot \sqrt{|S_i|},$$

where $j > i$ is the smallest index such that $S_i \cap S_j \neq \emptyset$ and for all $k < j$, $k \neq i$, $S_k \cap S_j = \emptyset$; if no such j exists, set $p_i := 0$.
11: **end for**
12: **return** (W, p)

Theorem 4.4. *The greedy mechanism is strategyproof, $\sqrt{m}$-efficient and runs in polynomial-time.*

Lets verify that the greedy mechanism satisfies the two properties of Lemma 4.2 and is thus strategyproof. It is easy to see that the greedy mechanism satisfies monotonicity: Suppose (S_i, b_i) is a winning bid. If player i increases his bid or submits a subset $S_i' \subset S_i$, he can only move further to the front of the greedy ordering. Since S_i is disjoint from all sets S_j of previously picked players $j \in W$, i remains a winner. Next consider the critical value property. The critical value of a winning bid (S_i, b_i) corresponds to the bid b_i' for which (S_i, b_i') still wins. Consider Step 10 of Algorithm 4. (S_i, b_i') remains a winning bid as long as $b_i' \geq p_i$, since if $b_i' < p_i$ player j precedes i in the greedy order and thus j wins and prevents i to enter the winning set W. Thus the payment p_i corresponds to the critical value.