Algorithms for Selfish Agents

Lecture Notes

V. Auletta, P. Penna, P. Persiano

October 25, 2003

Contents

Abstract

This document contains lecture notes for a class on algorithms for selfish agents held by the authors at Università di Salerno in Spring 2003. Some lecture have been subsequently added.

Lecture 1

Game theory and Nash equilibrium

TBC

Lecture 2

Selfish routing

In these lectures we study the problem of routing in communication networks when traffic is routed by selfish agents that aim at minimizing the latency experienced by their traffic. The selfishness of the agents is modeled by using the concept of Nash equilibrium. We are interested in the ratio between the cost of the flow in Nash equilibrium and the optimal flow.

The material is taken from [10].

2.1 The model

1. We have a graph $G = (V, E)$ and k pairs of flow requests $(s_1, t_1), \cdots, (s_k, t_k)$ each associated with a rate r_i. For each i we denote by $\mathcal{P}_i$ the set of the simple (s_i, t_i)-paths and by $\mathcal{P} = \cup_{i=1}^{k} \mathcal{P}_i$.

2. A *flow* is a function $f : \mathcal{P} \to \mathcal{R}^+ \cup \{0\}$ and the flow f_e induced by f on edge e is simply $f_e = \sum_{P:e \in P} f_p$.

3. A flow f is *feasible* if for all i we have $\sum_{P \in \mathcal{P}_i} f_P = r_i$.

4. Given a *latency function* defined over the edges e of G, we define the *latency* $l_P(f)$ of a path P with respect to flow f as $l_P(f) = \sum_{e \in P} l_e(f_e)$.

   We assume that the latency is nonnegative, nondecreasing and differentiable.

5. The cost (latency) $C(f)$ of a flow f is defined as

$$C(f) = \sum_{P \in \mathcal{P}} f_P l_P(f) = \sum_{e \in E} f_e l_e(f_e).$$

6. An instance of the flow problem is a triple (G, l, r).

2.2 Nash equilibrium

Definition 1 *A feasible flow f is at Nash equilibrium is for (G, l, r) if for i, for all δ, for all $P_1, P_2 \in \mathcal{P}_i$ with $f_{P_1} \geq \delta$, we have that*

$$l_{P_1}(f) \leq l_{P_2}(f^{\delta}),$$

where flow f^δ is defined as

$$f_P^\delta = \begin{cases} f_P - \delta, & \textit{if } P = P_1; \\ f_P + \delta, & \textit{if } P = P_2; \\ f_P, & \textit{otherwise.} \end{cases}$$

Corollary 2 *A feasible flow f is at Nash equilibrium if and only if for all i's and for all P_1, P_2 with $f_{P_1} > 0$*

$$l_{P_1}(f) \le l_{P_2}(f).$$

In particular, the above corollary implies that all paths of $\mathcal{P}_i$ that have positive flow experience the same latency that we will denote by $L_i(f)$. With this in mind, we can write

$$C(f) = \sum_{i=1}^{k} r_i L_i(f).$$

2.3 Examples

Suppose we have two parallel links joining vertices s and t. Latency on the upper link is 1 and latency of the lower link is x^p and suppose one (s,t)-request with rate $r = 1$.

For $p = 1$, the optimal flow splits the flow equally between the two edges with a total cost of 3/4. Instead the flow at Nash equilibrium send all flow on the lower link with a total cost of 1.

For $p > 1$, instead the optimal flow assigns $(p+1)^{-1/p}$ units of flow to the lower link and the remainder to the upper link. The cost is $1 - p(p+1)^{-(p+1)/p}$ which tends to 0 as p increases. The Nash flow instead is still the same as $p = 1$ and it costs 1. Therefore, the ratio between Nash and optimum grows unbounded.

2.4 Convex latency functions

The optimal flow problem can be written as an optimization problem over a convex set. This means that when the latency functions are convex then a locally optimal flow (that is one whose cost cannot be decreased by moving flow from one path to another path) is also globally optimal. We thus have the following lemma.

Lemma 3 *A flow f feasible for (G, l, r) is optimal if and only if for all i and for all paths $P_1, P_2 \in \mathcal{P}_i$ with $f_{P_1} > 0$, we have that*

$$l^*_{P_1}(f) \le l^*_{P_2}(f),$$

where $l^*_e(x) = (x l_e(x))' = l_e(x) + x l'_e(x)$ for all edges e and $l^*_P(f) = \sum_{e \in P} l^*_e(f_e)$.

Intuition: Set $c_P(x) = x \cdot l_P(x)$. A flow f is locally optimal if and only if for any $\delta > 0$ we have that

$$c_{P_1}(f) + c_{P_2}(f) \le c_{P_1}(f - \delta) + c_{P_2}(f + \delta)$$

which implies that, for all $\delta > 0$,

$$\frac{c_{P_1}(f) - c_{P_1}(f-\delta)}{\delta} \le \frac{c_{P_2}(f+\delta) - c_{P_2}(f)}{\delta}$$

which is "equivalent" to

$$c'_{P_1}(f) \le c'_{P_2}(f).$$

■

Using the characterization of flow at equilibrium of Corollary 2 and the above lemma we have

Lemma 4 *If the function $xl_e(x)$ is convex for all edges, then a feasible flow f is optimal for instance (G, l, r) iff f is at Nash equilibrium for (G, l^*, r).*

2.5 Linear latency functions

Now we study linear latency functions: $l_e(x) = a_e x + b_e$ for some constants a_e and b_e and prove that the ratio between flow at Nash equilibrium and optimal flow is at most 4/3.

Corollary 5 *A feasible flow f is at Nash equilibrium if and only if for all i's and for all P_1, P_2 with $f_{P_1} > 0$*

$$\sum_{e \in P_1} (a_e f_e + b_e) \le \sum_{e \in P_2} (a_e f_e + b_e).$$

Proof: This is a re-writing of 2. ■

Lemma 6 *A feasible flow f is optimal for instance (G, l, r) iff for all i's and for all P_1, P_2 with $f_{P_1} > 0$*

$$\sum_{e \in P_1} (2a_e f_e + b_e) \le \sum_{e \in P_2} (2a_e f_e + b_e).$$

Proof: Observe that $l_e^*(x) = 2a_e x + b_e$ and then use Lemma 4 and Corollary 5. ■

The two above results have the following simple consequence.

Corollary 7 *If f is a flow at Nash equilibrium for (G, l, r) then*

1. *$f/2$ is optimal for $(G, l, r/2)$;*
2. *the marginal cost $l_P^*(f/2)$ for incrementing flow on a path P coincides with the latency $l_P(f)$ along P with respect to f.*

Lemma 8 *If f is optimal for an instance $(G, l, r/2)$ with linear latency function and g is feasible for (G, l, r) then*

$$C(g) \ge C(f) + \sum_{i=1}^{k} L_i^*(f) r_i / 2,$$

where $L_i^*(f)$ is the minimum marginal cost of increasing flow along an (s_i, t_i) path with respect to f.

Proof: By the convexity of the function $xl_e(x)$ we have that

$$x_1 l_e(x_1) - x_2 l_e(x_2) \ge (x_1 - x_2) l_e^*(x_2).$$

Therefore

$$\begin{aligned} C(g) &= \sum_{e \in E} g_e l_e(g_e) \\ &\ge \sum_{e \in E} f_e l_e(f_e) + \sum_{e \in E} (g_e - f_e) l_e^*(f_e) \\ &= C(f) + \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i} (g_P - f_P) l_P^*(f) \end{aligned}$$

Now observe that by definition of $L_i^*(f)$, it holds that for all i and for all paths $P \in \mathcal{P}_i$, $l_P^*(f) \geq L_i^*(f)$. Whence

$$\begin{aligned} C(g) &\geq C(f) + \sum_{i=1}^{k} L_i^*(f) \sum_{P \in \mathcal{P}_i} (g_P - f_P) \\ &= C(f) + \sum_{i=1}^{k} L_i^*(f) r_i/2. \end{aligned}$$

■

We have now the main result of this section.

Theorem 9 *If (G, l, r) has linear latency functions, then $\rho(G, l, r) \leq 4/3$.*

Proof: Let f be a flow at Nash equilibrium and let $L_i(f)$ be the latency of the paths of $\mathcal{P}_i$ with positive flow. So, by Corollary 7, $f/2$ is optimal for $(G, l, r/2)$ and $L_i^*(f/2) = L_i(f)$. Therefore for flow f^* feasible for (G, l, r) we have

$$C(f^*) \geq C(f/2) + \sum_{i=1}^{k} L_i^*(f/2) \frac{r_i}{2} = C(f/2) + \frac{1}{2} \sum_{i=1}^{k} L_i(f) r_i = C(f/2) + \frac{1}{2} C(f).$$

On the other hand we have that

$$C(f/2) = \sum_e f_e/2 \cdot l_e(f_e/2) = \sum_e \left(\frac{1}{4} a_e f_e^2 + \frac{1}{2} b_e f_e \right) \geq \frac{1}{4} \sum_e \left(a_e f_e^2 + b_e f_e \right) = \frac{1}{4} C(f).$$

■

2.6 Doubling the rate

In this section we prove that the flow f at equilibrium for (G, l, r) has cost not greater than the optimal flow for $(G, l, 2r)$.

Theorem 10 *Let f be a flow at equilibrium for (G, l, r) and and let f^* be a feasible flow for $(G, l, 2r)$. Then*

$$C(f) \leq C(f^*).$$

Proof: Consider the latency function $\bar{l}_e$ defined as follows

$$\bar{l}_e(x) = \begin{cases} l_e(f_e), & \text{if } x \leq f_e; \\ l_e(x), & \text{if } x \geq f_e. \end{cases}$$

Then we have that

$$\bar{C}(f^*) - C(f^*) = \sum_e f_e^* (\bar{l}_e(f_e^*) - l_e(f_e^*)).$$

Observe that $\bar{l}_e(x) - l_e(x)$ is zero when $x \geq f_e$ and is bounded by $l_e(f_e)$ for $x \leq f_e$. Therefore $x(\bar{l}_e(x) - l_e(x))$ is bounded from above by $f_e l_e(f_e)$. Therefore we have

$$\bar{C}(f^*) - C(f^*) \leq \sum_e f_e l_e(f_e) = C(f).$$

On the other hand, we have that $\bar{l}_p(0) \geq L_i(f)$ for any path $P \in \mathcal{P}_i$ and, since $\bar{l}$ is non decreasing we have that $\bar{l}_p(f^*) \geq L_i(f)$. Therefore,

$$\begin{aligned}
\bar{C}(f^*) &= \sum_P \bar{l}_P(f^*) f_P^* \\
&\geq \sum_i \sum_{P \in \mathcal{P}_i} L_i(f) f_P^* \\
&= \sum_i L_i(f) \sum_{P \in \mathcal{P}_i} f_P^* \\
&= \sum_i 2 r_i L_i(f) = 2C(f)
\end{aligned}$$

■

Lecture 3

Selfish routing with discrete jobs

These lecture notes are based on a lecture by Pino Persiano and on notes taken by L. Blandi and D. Riccio.

The material presented is taken essentially from [6]. More general bounds are found in [8] and in the paper by Czumaj and Vöcking.

3.1 Setting

Let us consider the following setting. We have n jobs of length $w_1, \cdots, w_n$ and m machines of speed $s_1, \cdots, s_m$ and we are interested in computing a schedule $S : \{1, \cdots, n\} \to \{1, \cdots, m\}$ that assigns each job to a machine. The *load* L_j of machine j with respect to schedule S is simply the sum of the lengths of the jobs assigned by S to machine j. A machine with load l and speed s has *completion time* l/s. The *makespan* of a schedule S is the maximum over all machine of the completion time.

We are interested in the case in which jobs are ownbed by selfish agents and each agent tries to minimize the completion time of his job. We assume that if a job is assigned to a machine of load l and speed s then the job will be finished at time l/s.

We denote by $p(i, j)$ the probability that the i-th job is assigned to the j-th machine. In this setting the load L_j of machine j is a random variable with mean

$$E[L_j] = \frac{1}{s_j} \sum_i p(i, j) w_i.$$

We are interested in studying the *coordination ratio*; that is the ratio between the makespan of the best schedule and the *worst makespan* obtained by a Nash equilibrium.

Selfish Routing. The setting outlined above can be interpreted as selfish routing game with machine representing links, machine speeds representing link latencies and jobs representing communication requests. Some differences are to be stressed with the setting of [10]. Here jobs have finite discrete length as opposed to infinitesimal length which implies that Nash equilibrium can be mixed. Moreover, in [10] the global objective was to minimize the average completion time as opposed to the maximum completion time (the makespan).

3.2 Preliminaries

Given probabilities $p(k,j)$ for all $k \neq i$ and for all j, we denote by $\mathrm{Cost}(i,j)$ the cost that agent i incurs into if his job is assigned to machine j. It is easy to see that

$$\mathrm{Cost}(i,j) = \frac{w_i}{s_j} + \sum_{k \neq i} p(k,j)\frac{w_k}{s_j} = E[L_j] + (1 - p(i,j))\frac{w_i}{s_j}$$

and set

$$\mathrm{mCost}(i) = \min_j \mathrm{Cost}(i,j).$$

Lemma 11 *In a Nash equilibrium we have*

1. *if* $\mathrm{Cost}(i,j) > \mathrm{mCost}(i)$ *then* $p(i,j) = 0$*;*
2. *if* $p(i,j) > 0$ *then* $\mathrm{mCost}(i) = E[L_j] + (1 - p(i,j))\frac{w_i}{s_j}$.

In the rest of this lecture we are interested to the case in which we have only two machines of same speed and we give matching bounds for the coordination ratio in the case in which only pure strategies are allowed (that is, for each i there exists j such that $p(i,j) = 1$) and in the more general case in which mixed strategies are allowed.

3.3 The pure strategy case

In this section, as a warm-up, we study pure Nash equilibrium and give a matching bound on the coordination ratio for the case of two machines of equal speed. A pure Nash equilibrium is described by the two vectors $((w_{i_1}, \cdots, w_{i_l}), (w_{j_1}, \cdots, w_{j_{n-l}}))$ of the weights of the jobs assigned to the two machines. We denote by L_1 and L_2 the loads of the machines and assume without loss of generality that $L_1 > L_2$.

Theorem 12 *For any configuration in pure Nash equilibrium, the coordination ratio is at most* $4/3$*. Moreover, there exists an istance which admits a pure Nash equilibrium of cost exactly* $4/3$ *times the optimum.*

Proof: We first prove the upper bound.
Let v be $v = L_1 - L_2$ and consider two cases.

1. $v > L_2$. In this case we have that $2L_2 < L_1 = L_2 + v < 2v$.

   Suppose that the machine with load L_1 is assigned more than one job and let w be the weight of the lightest job assigned to the first machine. Obviously, we have that $w < v$ for otherwise $L_1 > 2v$. Then if the job of weight w is moved to the second machine it will experience a completion time $w + L_2 < L_1$ and thus the configuration is not in equilibrium.

   Suppose instead that the load of the first machine is due to only one job. Then the configuration is optimal and thus the coordination ratio is $1 \leq 4/3$.

2. $v \leq L_2$.

   The cost of the configuration is L_2+v and the cost of the optimum is at least $L_2+v/2$. Therefore the coordination ratio is at most

$$\frac{L_2+v}{L_2+v/2} = 1 + \frac{v/2}{L_2+v/2} \leq 1 + \frac{v/2}{v+v/2} = 4/3.$$

■

The above bound is tight. Indeed if jobs have weights $(2,2,1,1)$ then $((2,2),(1,1))$ is a configuration in pure Nash equilibrium of cost 4. The optimum $((2,1),(2,1))$ has cost 3.

3.4 The general case

Let us fix probability $p(i,j)$ that give a Nash equilibrium and let us define q_i as the probability that job i is assigned to the machine j^* with lowest index among those with highest load. Then we have that the cost of the equilibrium is $\mathrm{Cost} = \sum_{i=1}^n q_i w_i$.

More we define $t(i,k)$ as the probability that jobs i and k are assigned to the same machine.

Lemma 13

$$q_i + q_k \leq 1 + t(i,k).$$

Proof: We have that

$$1 \geq Pr(i \text{or } k \text{are assigned to machine } j^*) = q_i + q_k - Pr(i \text{and } k \text{are assigned to machine } j^*) \leq q_i + q_k - t(i,k).$$

■

The following lemma follows almost directly from the defintions.

Lemma 14

$$\mathrm{mCost}(i) \leq \frac{1}{m}\sum_k w_k + \frac{m-1}{m} w_i.$$

Proof:

$$\begin{aligned}
\mathrm{mCost}(i) &= \min_j \mathrm{Cost}(i,j) \\
&\leq \frac{1}{m}\sum_j \mathrm{Cost}(i,j) \\
&= \frac{1}{m}\sum_j (E[L_j] + (1-p(i,j))w_i) \\
&= \frac{1}{m}\sum_j E[L_j] + \frac{m-1}{m} w_i \\
&= \frac{1}{m}\sum_k w_k + \frac{m-1}{m} w_i
\end{aligned}$$

■

Next we have the following lemma.

Lemma 15

$$\sum_{k \neq i} t_{ik} w_k = \mathrm{mCost}(i) - w_i.$$

Proof: By definition we have

$$t_{ik} = \sum_j p(i,j) p(k,j).$$

Therefore we have

$$\begin{aligned} \sum_{k \neq i} t_{ik} w_k &= \sum_j p(i,j) \sum_{k \neq i} p(k,j) w_k \\ &= \sum_j p(i,j) \left(E[L_j] - p(i,j) w_j \right). \end{aligned}$$

Now observe that if $p(i,j) > 0$ then $p(i,j) w_i = E[L_j] + w_i - \mathrm{mCost}(i)$ from which the lemma follows.
■

We are now ready to prove the main theorem of this lecture.

Theorem 16 *The coordination ratio for two machines is at most* $3/2$.

Proof: We have that the cost of a configuration in Nash equilibrium is $\mathrm{Cost} = \sum_k q_k w_k$ and the cost of the optimum is at least $1/2 \sum_k w_k$.

If for all k, it holds that $q_k \leq 3/4$ then the theorem follows.

Suppose that $q_i > 3/4$.

Then

$$\begin{aligned} \sum_{k \neq i} (q_i + q_k) w_k &\leq \sum_{k \neq i} (1 + t_{ik}) w_k \\ &\leq \sum_{k \neq i} w_k + \mathrm{mCost}(i) - w_i \\ &\leq 3/2 \sum_{k \neq i} w_k \end{aligned}$$

Using the above we have that

$$\begin{aligned} \mathrm{Cost} = \sum_k q_k w_k &= \sum_{k \neq i} q_k w_k + q_i w_i \\ &\leq 3/2 \sum_{k \neq i} w_k - \sum_{k \neq i} q_i w_k + w_i q_i \\ &\leq (3/2 - q_i) \sum_k w_k + (2q_i - 3/2) w_i \end{aligned}$$

Now, since $q > 3/4$ and since $\mathrm{opt} \geq \max\{w_i, 1/2 \sum_k w_k\}$ we have that $(2q_i - 3/2)w_i \leq (2q_i - 3/2)\mathrm{opt}$ and thus

$$\mathrm{Cost} \leq (3/2 - q_i) \sum_k w_k + (2q_i - 3/2)w_i \leq (3/2 - q_i)\, 2\mathrm{opt} + (2q_i - 3/2)\mathrm{opt} = 3/2\mathrm{opt}.$$

■

The above bound is tight. Indead if we have two jobs of length 1 then if each jobs is assigned to each machine with equal probability, then we have a Nash equilibrium of cost 3/2 and the optimum has cost 1.

Lecture 4

Stackelberg games

TBC

Lecture 5

Introduction to Mechanism Design

5.1 Dominant strategies and Nash equilibria

In the previous lectures we have seen examples of games that admit several Nash equilibria. Moreover, some of these equilibria correspond to solutions that are far off the optimal ones. In particular, we have seen that

- For the "football or shopping" game,

$(\mathbf{2},\mathbf{1})$	$(0,0)$
$(0,0)$	$(\mathbf{1},\mathbf{2})$

  we could not predict the "behavior" of the agents. Indeed, it is not clear how the agents[1] can reach one of the two equilibria (shown in bold in the table of the payoffs) and which one of the two. Notice that, the best strategy for player 1 (i.e., the row) depends on the chosen strategy of player 2 (i.e., the chosen column), and vice versa. Intuitively, the two players should agree on some *joint* strategy.

- For the n links game, there exists a Nash equilibrium whose cost is $\Omega(\frac{\log n}{\log \log n})$ times the optimum.[2] Since we cannot predict which of the Nash equilibria is reached (if so), then it may be the case that the *worst* one is reached, thus a non-optimal configuration.

It is not clear whether a Nash equilibrium can be reached, even for those games that have *only one* such an equilibrium: the "matching pennies" game

$(1,-1)$	$(-1,1)$
$(-1,1)$	$(1,-1)$

Here, the unique equilibrium is reached when player 1 (resp., player 2) chooses a row (resp., column) with probability 1/2. However, how can the two players "agree" on this unique probability distribution? (consider the situation in which the game goes through several rounds and, at each round, one of the two players is allowed to change her/his probability distribution).

[1] In the sequel we use the terms 'agent' and 'player' interchangeably.

[2] The proof of this lower bound is based on a generalization of the lower bound 3/2 for two identical machines. Consider n identical machines, n identical jobs and, for each job/agent, a job chooses one machine with uniform distribution.

These two facts seem to denote the "weakness" of Nash equilibria. On one hand, there is no "preferred" Nash equilibrium (actually, it is not clear how such a configuration can be reached). On the other hand, even if we are guaranteed that a Nash equilibrium is reached, it may be a "very bad" one (i.e., the worst one, which for the m links game is non-optimal).

Let us now consider another example of a game with a *unique* Nash equilibrium: the "two prisoners' dilemma"

$(3, 3)$	$(1, 5)$
$(5, 1)$	$(\mathbf{2}, \mathbf{2})$

Besides the fact that $(2, 2)$ is the unique equilibrium, there is a very important property in this game:

- Even though the payoff of player 1 depends on the strategy of the other player (i.e., the column), no matter what this strategy is like, for player 1 it is always better to choose the second row (recall that player 1 cannot choose the column). In other words, player 1 does not need to know what the other player is doing to maximize his/her payoff!

- Similarly, no matter which strategy (row) player 1 selects, for player 2 it is always better to choose the second column. Again, player 2 does not need to know what the other player is doing to maximize his/her payoff!

Intuitively, both players have a "universally" optimal strategy which guarantees that, no other strategy would increase his/her payoff. Consider a payoff table of the form

$(v_1(1, 1), v_2(1, 1))$	$(v_1(1, 2), v_2(1, 2))$
$(\mathbf{v_1}(\mathbf{2}, \mathbf{1}), \mathbf{v_2}(\mathbf{2}, \mathbf{1}))$	$(v_1(2, 2), v_2(2, 2))$

such that

$$\forall j \in \{1, 2\},\ v_1(2, j) \geq v_1(1, j)$$

(i.e., for player 1, row 2 is never worse than row 1)
and

$$\forall i \in \{1, 2\},\ v_2(i, 1) \geq v_2(i, 2)$$

(i.e., for player 2, column 1 is never worse than column 2)
In this case, player 1 (resp., player 2) has no reason to choose a strategy different from row 2 (respectively, column 1).

In general, we say that there exists a *dominant strategy* for players i if this strategy maximizes his/her payoff, for all possible strategies adopted by the other agents. Notice that, this is a property of the table of payoff, so an agent can actually verify whether such a strategy exists!

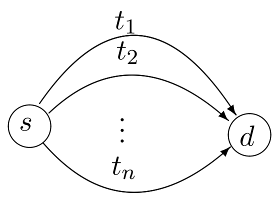
5.2 Payments inducing dominant strategies

We would like to reward each agent in such a way that

- every agent has a dominant strategy

- the dominant strategy is the "desired one"

We better explain the above two goals with an example. Consider the n links network in which each link is owned by a selfish agent i, $i = 1, 2, \ldots n$. Each link i is owned by an agent AG_i that knows the speed of that link, that is, the time t_i that a packet needs to traverse the link (i.e., $t_i = 1/\mathit{speed\_link\_i}$). This is a *private information*, in the sense that AG_i is the only one to know t_i (in particular, we do not know the value t_i). More importantly, since t_i is the time link i must be used in order to perform the transmission, t_i represents a *cost* for agent AG_i if his/her link is selected.



GOAL: We want to send one piece of traffic from s to t using the fastest of the n links, that is, the link minimizing t_i.

It seems necessary that, somehow, we obtain all correct values from the agents. It is then natural to ask the following:

QUESTION: Can we guarantee that each agent reports his/her speed correctly?

Observe that this is impossible in general: a *malicious* agent may always lie attempting to making us fail in our task! However selfish agents do not behave in this way: they simply try to maximize their own payoff or utility. This means that they lie only if there is a reason for that (namely, they get some benefit out of this). We then better address the following question:

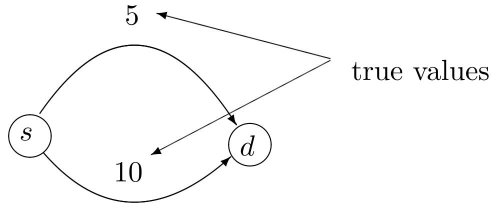
QUESTION: Can we guarantee that no agent i has an incentive in lying about his/her link speed, i.e., in reporting a value $s_i \neq t_i$?

We next consider several payments and see whether they provide a positive answer to the above question or not.

No payments. What happens if we do not provide any payment to the agents? The *utility* u_i of agent AG_i can be defined as the loss due to cost incurred by the agent, if his/her link is chosen:

$$u_i = \begin{cases} -t_i & \text{if link } i \text{ is chosen,} \\ 0 & \text{otherwise.} \end{cases}$$

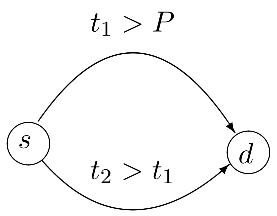
Consider the following example:



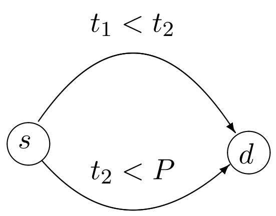
In this case, if agent AG_1 owning the link requiring 5 times unit would report a value $s_1 > 10$ then his/her utility would be 0 (the other link would be selected), while reporting $t_1 = 5$ would give $u_1 = -5$ (i.e., the cost he/she incurs when being selected).

Pay a fixed amount. We fix a value P and reward the agent reporting the largest speed an amount equal to P (i.e., this value is chosen in advance and independently of the agents declarations).

The following examples show that, doing so, we risk to overpay or underpay the agents:



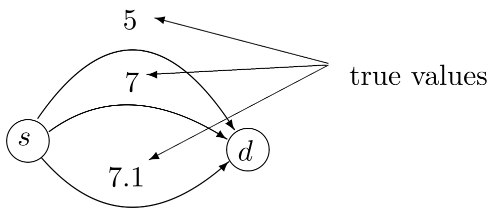
we underpay the agent(s)



we overpay both agents

On the left, we are underpaying AG_1, so he/she would be better reporting $s_1 > t_2$, since the corresponding utility would be 0 against $u_1(t_1) = P - t_1 < 0$ when telling the truth. This is essentially the same problem as in the case of no payments. If, instead, we fix a value P which is sufficiently large to guarantee that $P \geq t_1$, then we risk to pay too much: since we do not know the values t_i all we can do is to be sure that $P > \max_i t_i$ (assume we know the speed of the slowest existing link). Above on the right we have such an example: we are overpaying both agents, when selected; so AG_2 could report $s_2 < t_1 < t_2$ and obtain a positive utility equal to $u_2(s_2) = P - t_2 > 0 = u_2(t_2)$ (notice that the cost is always computed with respect to the true values).

Pay $P_i = s_i$. We choose the link with minimal reported s_i and pay to AG_i an amount $P_i = s_i$. Not surprisingly, this payments originate *speculation*. Consider the following example:



If agent AG_1 reports a value s_1 such that $t_1 = 5 < s_1 < t_2 = 7$, then her/his link is still selected and he/she gets a better payment: $P(t_1) = 5 < P(s_1)$. Therefore, also the utility would be better when reporting such an s_1: $u_1(t_1) = 5 - 5 < u_1(s_1) = s_1 - t_1$. So, AG_1 is clearly tempted to get as close to $t_2 = 7$ as possible!

This would not be harmful if *each agent would know the true/reported values of every other agent in advance*:[3] in this case, AG_1 may compute the minimum $s_i < t_2$ and report this value. If this

[3]We consider a game in which every agent/player reports a value simultaneously with all other agents. This kind of games are usually termed *revelation games*.

happens, then we are still selecting the fastest link. However, this assumption is unreasonable: recall that t_i is a private information and AG_1 does not really know t_2. Also, communication between the agents cannot be assumed to be truthful (i.e., they may lie one to the other as well) or even possible.

So, all an agent AG_i knows is the following:

- If her/his link is the fastest one, according to the reported values s_i, then he/she could get something more by reporting s_1 just below the minimum s_j, with $j \neq i$.
- If her/his link is not the fastest one, according to the reported values s_i, then the best is to report $s_i = t_i$ (if he/she tries to be selected, the he/she would receive some payment $P_i \leq \min_{j \neq i}\{s_j\} < t_i$)

It is evident that agents are tempted to overbid (underbidding never pays off!): if AG_i is able to guess a "not too high" value $s_i > t_i$, then he/she might improve his/her utility. Suppose that all agents think that $s_i = t_i + a$ would be a good choice, for some $a > 0$. In the example above, AG_2 would then declare $s_2 = 7 + a$. Now, observe that the best value for AG_1 is no longer $s_1 < t_2 = 7$: if AG_1 reports a value $s_1 < s_2 = 7 + a$, then she/he will be selected and rewarded an amount "close to" $7 + a$. So, AG_1 might think that, since AG_2 will overbid by a factor a, then it is better for him/her to overbid by a factor $2a$. But also AG_2 may reasoning in the same way, thus leading AG_2 to overbid by a factor $3a$, and so forth...When will an agent stop doing this? Will all agents stop at the same value $k \cdot a$? Will all agents start with the same $a > 0$? This seems to be very unlikely!

Essentially what is not desirable in these payments is the fact that agents know that telling the truth is *not* the best strategy for them, but the best strategy *depends* on the other agents strategies (i.e., reported values). This makes the behavior of the agents "unpredictable": they may only guess a number trying to improve the utility. It then may be the case that, in the above example, AG_2 overbids by a factor $a = 1/2$ and agent AG_1 overbids by a factor $6a = 3$. In this case, we would fail since we would select the slowest link!

5.2.1 Vickrey-Clarke-Groves (VCG) payments

Though the payments considered above do not work, they have some good features:

- The "fixed P" is good since the amount of money agent AG_i receives does not depend on s_i. So, there is *no incentive to overbid.* overbid trying to increase his/her utility when selected.
- The payment "$P_i = s_i$" guarantees that we do not underpay an agent if he/she reports the true value (so he/she will not try to be excluded from the solution). Moreover, there is *no incentive to underbid*, since this always lead to a non-positive utility (in the above example, if AG_2 reports $s_2 \leq 5$, then she/he gets $u_2 = s_2 - 7 \leq -2$)

As mentioned above, the problem with the payments "$P_i = s_i$" is the fact that the highest payment that an agent AG_i can get depends on the other values s_js. One of the ideas of the VCG[4] payments is to compute this value and provide this (maximal) amount of money to the agent AG_i anyway.

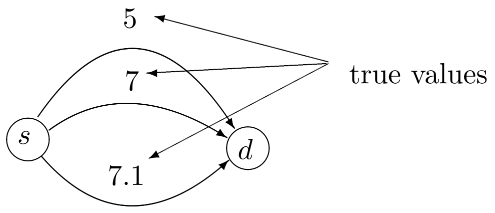
Consider the following approach. Based on $s_1, s_2, \ldots, s_n$, do the following:

1. Choose the cheapest link i corresponding the cheapest value s_i;

[4]The name VCG is doe to the three fundamental works by Vickrey [11], Clarke [3] and Groves [5] on this area. These work contain all the main ideas we are going to discuss in this lecture.

2. Reward AG_i an amount equal to the 2^{nd} best/cheapest link, i.e, $P_i = \min_{j \neq i}\{s_j\}$.

Intuitively, this approach possesses both positive features of payments "fixed P" and "$P_i = s_i$": the payment P_i does not depend on s_i, and there is no incentive to underbid since the rewarding is fixed to be the cost of the 2^{nd} cheapest link. Indeed, if an agent AG_j, owning a link of non-minimal cost, underbids pretending to be the cheapest one, then the payment he/she receives is no larger than his/her true cost t_j:



if AG_2 reports $s_2 \leq 5$, the he/she would receive $P_2 = 7$ and his/her link would be selected, thus a utility $u_2 = 7 - 7 = 0$; however, 0 is also the utility that he/she would obtain reporting $s_2 = t_2 = 7$, thus no reason to lie!

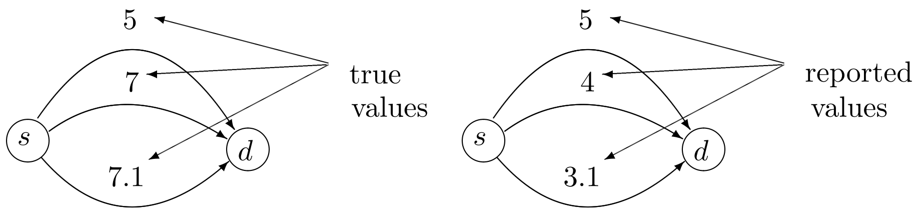
Also, the agent AG_i owning the truly cheapest link has no reason o underbid since her/his link would remain the cheapest and the payment does not depend on his/her declared value $s_i < t_i$. In the example above, if AG_1 reports $s_1 = 4$, her/his link is still selected and the payment remains $P_1 = 7$.

Finally, the most interesting feature of this payment scheme is the fact that it fixes the problem of the payment "$P_i = s_i$": now overbidding does not pay off! If, in the example above, AG_1 reports $s_1 > t_1 = 5$, then there are two possibilities:

1. link 1 is selected: AG_1 receives $P_1 = 7$, no matter what he/she declared.
2. link 1 is not selected: he/she receives no payment, thus a utility $u_1 = 0$.

It is clear that, overbidding does not increase the payment that he/she receives. However, if AG_1 overbids a too high value, he/she risks to loose the opportunity of being chosen (and having a positive utility $u_1 = 7 - 5$).

It is now natural to ask the following question: if an agent AG_j lies, is there an incentive for another agent AG_i to lie as well? Consider the following example:



According to the *reported* values, AG_1 is not owning the cheapest link. Does AG_1 have an incentive to lie? Observe that, the payment is fixed to be 4 (i.e., the 2^{nd} cheapest reported cost). So, if AG_1 tries to be selected (i.e., reports $s_1 < 3.1$) then he/she will obtain $u_1 = 4 - 5 < 0$.

More in general, the payment scheme described above guarantees the following two properties:

1. If we select link i, then $t_i \leq \min_{j \neq i}\{s_j\} = P_i$;

2. If we do not select link i, then $t_i \geq \min_{j \neq i}\{s_j\} = P_i$.

Since the value P_i does not depend on s_i, the two items above state that (i) if link i is the best, according to the reported values of the other agents, then there is an incentive for AG_1 in being selected (thus reporting $s_i = t_i$) since, in this case, the utility is positive; (ii) if link i is not the cheapest, according to the reported values of the other agents, then there is no incentive in being selected since this results in a non positive utility.

Formally, let $P_i(s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n)$ denote the payment that each agent AG_i receives when the reported values are $s = (s_1, s_2, \ldots, s_n)$, with s_i being the value reported by AG_i. For every value x, let us define $(x, s_{-i}) := (s_1, s_2, \ldots, s_{i-1}, x, s_{i+1}, \ldots, s_n)$. The utility of AG_i when he/she reports s_i and the other agents reported values are $s_{-i} = (s_1, s_2, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$ is then equal to

$$u_i(s_i, s_{-i}) = P_i(s_i, s_{-i}) + \begin{cases} -t_i & \text{if link } i \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

Recall that our goal is to guarantee that "truth-telling" is a dominant strategy for all agents, i.e., it is always the case that an agent cannot improve his/her utility by misreporting his/her true value t_i. This is actually what the above payments guarantee:

Theorem 17 *For all i, $i = 1, 2, \ldots, n$, it holds that*

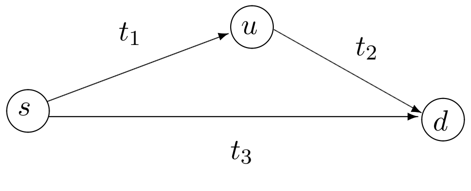
$$\forall s_{-i} \forall s_i \quad u_i(t_i, s_{-i}) \geq u_i(s_i, s_{-i}).$$

Proof: Exercise 31. ■

In game-theoretic setting the strategy $\overline{s}_i = t_i$ is termed *dominant* for this game: for all possible strategy that the other agents choose, strategy $\overline{s}_i$ yields the maximum payoff that AG_i can obtain in this situation.

5.2.2 The shortest path problem

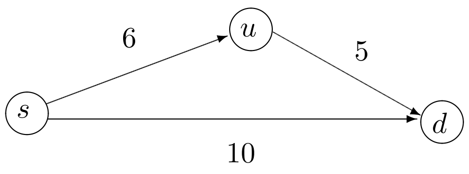
Consider now the following natural extension of the problem considered above: the nodes s and d are any two given nodes of a directed weighted graph; each edge i is owned by an agent and the corresponding weight is defined as in the previous problem. Assume also that every agent owns exactly one edge. [5] Here is an example:



[5] We make this assumption only for the sake of presentation/semplicity. However, this is not really necessary (see Exercise 35).

A first attempt: using the same payments for the n links problem

Consider a straightforward application of the payments for the n links problem to the example with three edges above: we select the cheapest path from s to d and we pay, to each agent owning a selected edge, the cost of the 2^{nd} cheapest path. Let us see what happens in the following instance (numbers represent the true costs/weights):



Let AG_1, AG_2 and AG_3 own the edge of cost 6, 5 and 10, respectively. If all agents are truth telling, then AG_1 and AG_2 have utility equal to 0 (we would select edge (s, d) and pay AG_3 only). However, if AG_1 reports a value $s_1 \leq 4$, then he/she would obtain something better: the payment would be 10 (now the 2^{nd} best path is the edge (s, d)) and AG_1 would be selected and receive that payment, thus a utility of $10 - 5 > 0$.

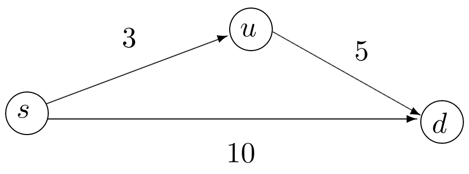
What is the problem here? It seems that we are paying too much, since a non-optimal edge is incentivated to get into the chosen path by lowering its cost.

n links revised: pay the "maximum speculation"

We briefly review the payments for the n parallel links, i.e., a special case of the shortest path problem. Consider the following instance:

Let us now see what would happen with payments "$P_i = s_i$". In particular, how much could AG_1, owning the link of cost 5, speculate? In order to receive some payment, he/she needs to be selected, thus implying that he/she must report some value $s_1 \leq 7$. This will also be the payment that, in this case, he/she will receive. So, 7 is the maximum he/she could: this is exactly the cost of the 2^{nd} cheapest link!

This seems to see another idea contained in the payments using the 2^{nd} cheapest link: pay the maximum amount that an agent could speculate if we were using the payments "$P_i = s_i$". Let us apply this idea to the following instance:



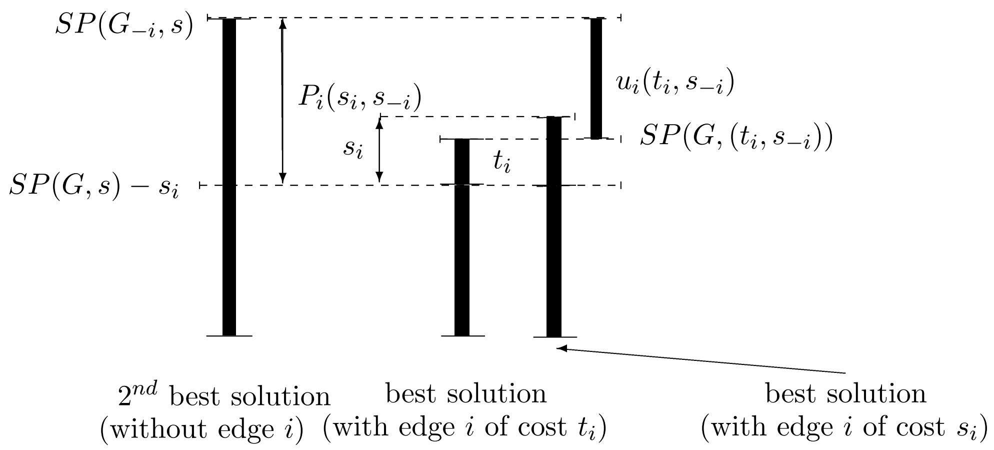
Assume all agents report the truth values, thus selecting the upper path. How much could get the agent AG_1 by lying about her cost, if $P_1 = s_1$? It must be $s_1 + 5 \leq 10$ (otherwise, we would select the other path). So, the payment P_1 should be equal to $10 - 5$. What does this value represent? The value 10 is the cost of the 2^{nd} cheapest path, while 5 is the cost of the chosen path (i.e., the cheapest one) *without counting edge of AG_1*.

VCG payments for the shortest path problem

Let us define $SP(G, w)$ the cost of the shortest path from s to d when the weight of edge i is equal to w_i, $w = (w_1, w_2, \ldots, w_n)$. Also let $SP(G_{-i}, w)$ denote the shortest path from s to d without using edge i. We then define the payment function of agent i as

$$P_i(G, (s_i, s_{-i})) := SP(G_{-i}, (s_i, s_{-i})) - (SP(G, (s_i, s_{-i})) - s_i). \tag{5.1}$$

Observe that the quantity $SP(G, (s_i, s_{-i})) - s_i$ corresponds to the cost of the shortest path, according to weights (s_i, s_{-i}), without counting s_i. Here is a pictorial explanation of the idea behind these payments for the shortest path problem:


$SP(G_{-i}, s)$
$P_i(s_i, s_{-i})$
$u_i(t_i, s_{-i})$
s_i
t_i
$SP(G, (t_i, s_{-i}))$
$SP(G, s) - s_i$
2^{nd} best solution (without edge i)
best solution (with edge i of cost t_i)
best solution (with edge i of cost s_i)

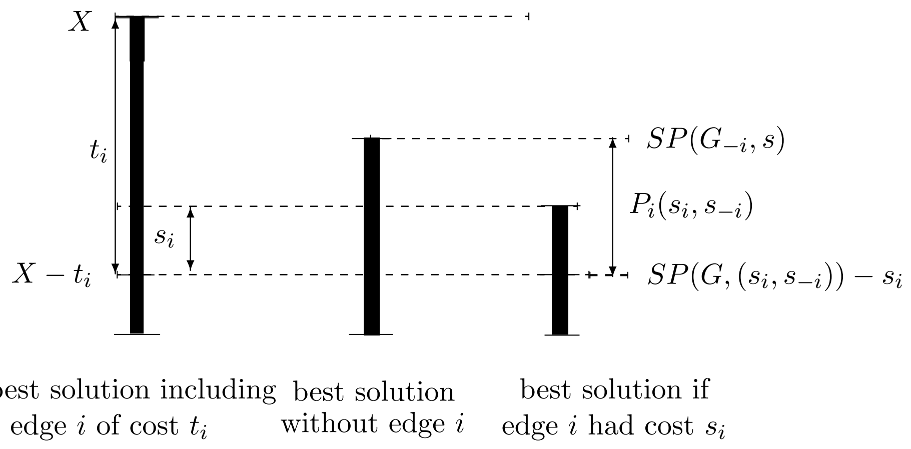
Observe that, the utility of an agent i whose edge is selected is equal to

$$\begin{aligned} P_i(t_i, s_{-i}) - t_i &= SP(G_{-i}, (s_i, s_{-i})) - (SP(G, (s_i, s_{-i})) - s_i) - t_i & (5.2)\\ &= SP(G_{-i}, (s_i, s_{-i})) - (SP(G, (t_i, s_{-i})). & (5.3)\end{aligned}$$

(The second equality es graphically shown above and easy to prove by considering the definition of $SP(\cdot)$.)

So, if agent i reports t_i, then he/she gets a non-negative utility. Moreover, if he/she reports a higher value $s_i > t_i$, such that edge i is still in the shortest path, then his/her payment does not change, thus also the utility does not change. Moreover, if s_i is “too high”, so that the other solution becomes better, then the utility of agent i becomes 0 (the edge would not be selected anymore). Finally, lowering the reported cost to some $s_i < t_i$ does not change the payment nor the utility (the edge would be selected as well and payed the same amount).

The above considerations show that, if the shortest path with edge costs (t_i, s_{-i}), then AG_i has no incentive in reporting $s_i \neq t_i$. The next picture shows why the same holds when i is not in a shortest path for the weights (t_i, s_{-i}):



X
t_i
s_i
$X - t_i$
$SP(G_{-i}, s)$
$P_i(s_i, s_{-i})$
$SP(G, (s_i, s_{-i})) - s_i$
best solution including edge i of cost t_i
best solution without edge i
best solution if edge i had cost s_i

In this case, if AG_i reports $s_i < t_i$ so that his/her edge would be selected, then the payment he/she would get would not cover his/her cost t_i. Thus a non-positive utility, which cannot be better than reporting t_i and being not selected.

Using these arguments it is easy to show that the above payment functions guarantee that no agent owning an edge can improve his/her utility by misreporting his/her edge cost to the *mechanism* that selects the shortest path and pays the selected edges according to the payment functions in Eq. 5.1.

5.3 The VCG Theorem

The shortest path problem contains all ingredients to show the full power of VCG payments. Indeed, we did not use any particular property of the shortest path in order to prove that the payment functions in Eq. 5.1 work for this problem. In this section we formally define a class of optimization problem and a generalization of these payment functions such that, every problem in this class admits a so called *truthful mechanism*, that is, an algorithm A and a payment function $P = (P_1, P_2, \ldots, P_n)$ such that their combination $M = (A, P)$ incentivates the agents to report their true values t_is.

5.3.1 A general setting

We consider optimization problems defined as fourtuples $(\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$, where:

1. $\mathcal{I}$ is the set of *instances* of the problem;

2. $\mathsf{sol}(\cdot)$ is a function mapping every instance $I \in \mathcal{I}$ into a set $\mathsf{sol}(I)$ of *feasible solutions*;

3. $\mathsf{m}(\cdot)$ is the *measure* or *optimization function* mapping every $X \in \mathsf{sol}(I)$ into a non-negative real number $\mathsf{m}(X, I)$;

4. $\mathsf{goal} \in \{\min, \max\}$; the goal is to find an $X^* \in \mathsf{sol}(I)$ such that $\mathsf{m}(X^*, I) = \mathrm{opt}(I) := \mathsf{goal}_{X \in \mathsf{sol}(I)} \mathsf{m}(X, I)$. If $\mathsf{goal} = \min$ (respectively, $\mathsf{goal} = \max$) then Π is a *minimization* (respectively, *maximization*) problem.

We the consider a set of *selfish agents* that privately hold part of the input. In particular, for every $I \in \mathcal{I}$, we have that $I = (t, \sigma)$, where

1. $t = (t_1, t_2, \ldots, t_n)$ is the *private input*, and t_i is the *type* of agent AG_i (e.g., the cost of his/her edge);

2. σ is the *public part* of the input which is public knowledge (e.g., the nodes and the edges of a graph, in the case of the shortest path problem).

We consider agents that can report a value s_i in a set S_i, with $t_i \in S_i$. We also assume that the set of feasible solutions $\mathsf{sol}(I)$ does *not* depend on the agents type t. For every solution $X \in \mathsf{sol}(I)$, every agent AG_i has a *valuation function* $v_i(X, t_i)$: the function $v_i(\cdot,\cdot)$ is also public knowledge, but the type t_i is not. The value $v_i(X, t_i)$ represents a benefit (when positive) or a cost (when negative) for AG_i when a solution X is implemented (e.g., when a certain path containing his/her edge is selected). In our shortest path problem, the valuation functions are naturally defined as

$$v_i(X, t_i) = \begin{cases} -t_i & \text{if } X \text{ contains link } i, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 18 (mechanism) *A mechanism for a problem* $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$ *is a pair* $M = (A, P)$ *where, given an instance* $I = (t, \sigma)$, *(i)* A *is an algorithm computing a solution* $A(s, \sigma) \in \mathsf{sol}(I)$ *and (ii) each agent* AG_i *receives an amount of money equal to* $P_i(s_i, s_{-i})$, *with* $P = (P_1, P_2, \ldots, P_n)$. *In this case, the* utility *of* AG_i *is equal to* $u_i(s_i, s_{-i}) := v_i(A(s_i, s_{-i}), t_i) + P_i(s_i, s_{-i})$.

Definition 19 (truthful mechanism) *A mechanism* $M = (A, P)$ *for a problem* $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$ *is* truthful *if, for every agent* AG_i, $i = 1, 2, \ldots, n$, *the strategy* $\overline{s}_i = t_i$ *is dominant, that is,*

$$\forall s_{-i} \forall s_i \quad u_i(t_i, s_{-i}) \geq u_i(t_i, s_{-i}),$$

where $u_i(s_i, s_{-i}) = v_i(A(s_i, s_{-i}), t_i) + P_i(s_i, s_{-i})$.

Remark 1 *Observe that, if* M *is truthful, then we can assume that all agents will report their true type* t_i. *Therefore, algorithm* A *is provided with the correct input. So, if* A *is an exact (resp., c-approximate) algorithm for* Π *(without selfish agents), then the mechanism* A *guarantees that an exact (resp., c-approximate) solution.*

5.3.2 One more intuition on the shortest path problem: the agents "want to help" the algorithm

Let us step back to the shortest path problem for a second. Consider the utility function of agent AG_i when reporting s_i. With some abuse of notation, we let $SP((G, w')|w'')$ denote the cost of the shortest path for the instance (G, w') evaluated w.r.t. the instance (G, w''). By simply replacing t_i with s_i in Eq. 5.3 we obtain:

$$\begin{aligned} u_i(s_i, s_{-i}) &:= P_i(t_i, s_{-i}) - t_i = SP(G_{-i}, (s_i, s_{-i})) + && (5.4)\\ &- (SP(G, (s_i, s_{-i})) - s_i) - t_i && (5.5)\\ &= \underbrace{SP(G_{-i}, (s_i, s_{-i}))}_{\text{independent of } s_i} - SP((G, \underbrace{(s_i, s_{-i})}_{\text{provided input}})| \underbrace{(t_i, s_{-i})}_{\text{"correct" input}}). && (5.6) \end{aligned}$$

A few more words are needed. Clearly, the first quantity in Eq. 5.6 does not depend on s_i (by definition, edge i is not in the solution). Let us ask ourselves the following: how can AG_i maximize her/his utility?

The only way is to minimize the quantity $SP((G(t_i, s_{-i}))|\,(t_i, s_{-i}))$, which we subtract. This quantity is defined as follows: a solution $X = A(G, (s_i, s_{-i}))$ is obtained by running a shortest path algorithm A on input (s_i, s_{-i}), and then the cost of X is computed w.r.t. the instance $(G, (t_i, s_{-i}))$. Since A finds a path of minimal cost when provided with the "correct " input (t_i, s_{-i}), the best that AG_i can do is to report $s_i = t_i$ so to "help" algorithm A in its job. So, the maximum value of $u_i(\cdot, s_{-i})$ is achieved for $s_i = t_i$.

This is yet another proof that the mechanism for the shortest path is truthful. There are two fundamental ingredients that we have used in the "proof":

1. In Eq. 5.5 we can "combine" the cost of a solution with the terms '$-s_i$' and 't_i': adding '$t_i - s_i$' to the cost of a solution yields the cost of the same solution on input (t_i, s_{-i});

2. We are using an algorithm A for the shortest path problem which returns a minimum-cost path when provided with the correct input. This is also fundamental for the mechanism in order to be truthful (see Exercise 32).

The first property yields a class of optimization functions for which the same idea works: the commonly termed *utilitarian* problems (see next section). The second property determines, for the class of utilitarian problems, which algorithms A can be turned into a truthful mechanism $M = (A, P)$: the optimal ones!

5.3.3 Truthful VCG mechanisms for Utilitarian Problems

As for the shortest path problem, utilitarian problems have the objective function which is the sum of all agents costs/valuations:

Definition 20 (utilitarian problem) *A minimization (resp., maximization) problem* $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$ *is utilitarian if, for every instance* $I = (t, \sigma)$ *and for any* $X \in \mathsf{sol}(I)$*, it holds that*

$$\mathsf{m}(X, (t, \sigma)) = -\sum_{i=1}^{n} v_i(X, t_i)$$

(resp., $\mathsf{m}(X, (t, \sigma)) = \sum_{i=1}^{n} v_i(X, t_i)$*).*

Clearly, maximization utilitarian problems can be transformed into minimization ones and vice versa.

Example 21 (minimum spanning tree) *Consider the minimum spanning tree problem involving selfish agents owning one edge of a weighted graph each. The valuation function is defined analogously to the shortest path problem. It is easy to see that also this problem is a utilitarian optimization one.*

Definition 22 (VCG mechanism) *Let* A^* *be an optimal algorithm for a minimization utilitarian problem* $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$*. Let*

$$\mathsf{m}(X, (\sigma, s_{-i})) := -\sum_{j \neq i}^{n} v_j(X, s_j)$$

and let

$$P_i^{VCG}(s_i, s_{-i}) := h_i(s_{-i}) - \mathsf{m}(A^*(\sigma, s), (\sigma, s_{-i})), \tag{5.7}$$

where $h_i(\cdot)$ *is any function independent of* s_i*. Then the resulting mechanism* $M = (A^*, P^{VCG})$ *is called* VCG mechanism *for* Π*.*

Theorem 23 (VCG theorem) *Any VCG mechanism for $M = (A^*, P^{VCG})$ a minimization utilitarian problem is truthful.*

The proof goes through a number of relatively easy steps/observations (see Exercise 33).

Observation 24 *If $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$ is a minimization utilitarian problem, then*

$$\mathsf{m}(X, (\sigma, s_{-i})) - v_i(X, t_i) = \mathsf{m}(X, (\sigma, (t_i, s_{-i}))).$$

Lemma 25 *Let $M = (A^*, P^{VCG})$ be a VCG mechanism for a minimization utilitarian problem $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$. Then, the utility function of agent AG_i is equal to*

$$u_i(s_i, s_{-i}) = h_i(s_{-i}) - \mathsf{m}(A^*(\sigma, (s_i, s_{-i})), (\sigma, (t_i, s_{-i}))).$$

Observation 26 *If A^* is an optimal algorithm for a minimization utilitarian problem $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$, then the quantity*

$$\mathsf{m}(A^*(\sigma, (s_i, s_{-i})), (\sigma, (t_i, s_{-i})))$$

is minimized for $s_i = t_i$.

Now proving Theorem 23 is an easy quite simple (see Exercise 34).

5.3.4 Voluntary participation

The mechanism presented in Sect. 5.2.2 is a VCG mechanism in which $h_i(s_{-i})$ is the cost of the 2^{nd} best shortest path. If we look at the picture at pag. 31, we realize that this choice has a good feature: we always cover the costs of a truthful agent, that is, $u_i(t_i, s_{-i}) \geq 0$. This property is called *voluntary participation.*

Intuitively, we can achieve voluntary participation using VCG mechanisms whenever there exists an "alternative solution" X_{-i} that does not "involve" AG_i.

Definition 27 *Let $\mathsf{sol}(\sigma_{-i})$ denote the set of solutions $X_{-i} \in \mathsf{sol}(\sigma)$ such that, for every t_i, $v_i(X_{-i}, t_i) = 0$.*

Theorem 28 *Let $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$ be a minimization utilitarian problem $M = (A^*, P^{VCG})$ such that, for every $I = (\sigma, t)$ and for every $i = 1, 2, \ldots, n$, $\mathsf{sol}(\sigma_{-i}) \neq \emptyset$. Let M be the VCG mechanism for Π obtained by setting, in Def. 22, $h_i(s_{-i}) = \min_{X \in \mathsf{sol}(\sigma_{-i})}\{\mathsf{m}(X, (\sigma, s_{-i}))\}$. Then, M satisfies the voluntary participation constraint.*

Proof: Exercise 36. ■

Exercises

Exercise 29 *Show that, for the n links problem, if we pay the cost of the 3^{rd} cheapest link (instead of the 2^{nd} one), then there is an instance for which some agent can improve her/his utility by reporting a false cost.*

Exercise 30 *Consider the following payment for the n links problem. We pay to the selected link i a quantity which is the average between the best (i.e., the cost t_i of the selected link) and the 2^{nd} best link:*

$$P_i = \frac{t_i + \min_{j \neq i}\{t_j\}}{2}$$

Show that, in this case, there is an instance for which some agent can improve her/his utility by reporting a false cost.

Exercise 31 *Give a proof from scratch of Theorem 17 , i.e., without making use of Theorem 23.*

Exercise 32 *Consider the mechanism for the shortest path problem and replace the algorithm A computing a shortest path with an algorithm A' which computes a 2^{nd} shortest path. Let $SP'(\cdot)$ be defined as $SP(\cdot)$ with the only difference that we now consider the cost of the 2^{nd} shortest path. Let us define payments $P'_i(\cdot)$ by replacing, in Eq. 5.1, $SP(\cdot)$ by $SP'(\cdot)$. Is the mechanism $M' = (A', P')$ truthful?*

Exercise 33 *Prove Observation 24, Lemma 25 and Observation 26.*

Exercise 34 *Prove Theorem 23. Also, define the payment functions for a utilitarian maximization problem.*

Exercise 35 *Using Theorem 23, show that the shortest path problem admits a truthful mechanism also when an agent owns more than one edge. Also, define the payment functions for this case.*

Exercise 36 *Prove Theorem 28.*

Lecture 6

Cost sharing mechanisms

TBC

Lecture 7

Combinatorial Auctions

These lecture notes are based on a lecture by Enzo Auletta and on notes takes by A. Ferrante and S. Senatore.

In the previous lecture, necessary conditions to guarantee the truthfulness of auction mechanisms for single-minded agents, were given and was described a $\sqrt{m}$-approximation algorithm for single-minded agents. We recall that a single-minded agent is an agent that is interested to a single bundle of goods. In this lesson we deal with known-single-minded agents, that is agents that are interested to a single bundle of goods, but can't lie on the bundle but only on bids. In the following, first we will show that the greedy algorithm does not work for k-minded agents with any fixed k, moreover we will show that the necessary conditions are sufficient too for known-single-minded agents and finally we will describe an $\epsilon\sqrt{m}$-approximation algorithm for any fixed $\epsilon > 0$ (described by Mu'alen and Nisan in [9]) which works only for known-single-minded agents.

7.1 Greedy does not work for k-minded agents

In the following, we present an example that proves that the algorithm shown in the last lesson does not work for k-minded agents with any fixed $k \geq 2$. In particular we prove it with $k = 2$, i.e. *double-minded* agents (agents that are interested at most to two bundles of goods).

Example 1

Let us suppose to have two items ($M = \{a, b\}$) and two agents ($N = \{1, 2\}$) and let:

$$\begin{aligned} t_1 &= (\{a\}, 12) \\ t_2 &= \{(\{b\}, 10), (\{a, b\}, 30)\} \end{aligned}$$

be the types of the two agents. Suppose first agents bid truthfully, then according to the allocation algorithm, we have the following allocations and payments:

$$\begin{array}{ll} f_1 = \emptyset & p_1 = 0 \\ f_2 = \{a, b\} & p_2 = 2 \cdot \frac{12}{1} = 24 \end{array}$$

then the utilities are:

$$\begin{aligned} u_1 &= 0 \\ u_2 &= 30 - 24 = 6 \end{aligned}$$

If, instead, agent 2 bids $b2\{(\{b\},10),(\{a,b\},23)\}$ in accord to the allocation algorithm, we have:

$$\begin{array}{ll} f_1 = \{a\} & p_1 = 1 \cdot \frac{23}{2} = 11.5 \\ f_2 = \{b\} & p_2 = 0 \end{array}$$

Then, it follows that the utilities are:

$$\begin{aligned} u_1 &= 11.5 - 12 = -0.5 \\ u_2 &= 10 - 0 = 10 \end{aligned}$$

We notice the agent 2 sacrifices the item $\{a\}$ to get the item $\{b\}$ for a lesser price. In conclusion, the agent 2 gets a greater utility if it lies, so the mechanism is not truthful. ■

7.2 Necessary and sufficient conditions for known-single-minded agents

Previously, we said the type of agent j is represented by (S_j, t_j), compound by the bundle of goods on which the agent makes a bid and the real bid on that set. The agent lied both on the bundle and on the bid. Now a restricted condition is imposed: the agents can lie only on the price offered on the set of goods. It follows the conditions shown in last lesson become necessary too (in this case, the agent are called *one-parameter* that means the hidden information of each agent is expressed trough a single number). In the auction language this kind of agents is called *KNOWN-SINGLE-MINDED*: they can lie on the price offered during the auction, but not on the bundle of goods. In particular, in this case, the type of agent j is $t_j = v_j$.

We will show that for this class of agents, a truthful mechanism exists if and only if conditions given in the previous lesson hold. Hence, the four conditions become the followings:

- **EXACTNESS:** each agent obtains the whole required bundle or nothing (it obtains no goods);
- **MONOTONICITY:** for any $j = 1, \cdots, n$, if $f_j(b_{-j}, b_j) \neq \emptyset$ then for any $b'_j \geq b_j$ follows $f_j(b_{-j}, b'_j) \neq \emptyset$;
- **CRITICAL:** for any $j = 1, \cdots, n$ a value $\theta_j(b_{-j}, f)$ exists such that:
  - $f_j(b_{-j}, b_j) \neq \emptyset$ if and only if $b_j \geq \theta_j(b_{-j}, f)$;
  - the price is $p_j = \theta_j(b_{-j}, f)$.
- **PARTICIPATION:** if an agent wins the auction, its payment is not greater than its bid; if the agent loses, it pays nothing.

In the following we will say that a mechanism holds these four conditions (or that an algorithm holds the first two and a payment schema holds the last two), so we will say "the mechanism is exact (first condition)"; "the mechanism is monotone (second condition)"; "the mechanism (payment schema) is critical (third condition)" and finally "the mechanism (payment schema) is voluntary participation (fourth condition)" For instance, Vickrey's auction holds these four conditions (recall in this kind of auction there is a only good: the agent with the greatest bid wins and pays the second largest bid). In this way:

- EXACTNESS: the winner agent gets the good; the others do not receive anything;

- MONOTONICITY: goods are given to agent with the largest bid;
- CRITICAL: the critical value is the second largest bid;
- PARTICIPATION: the agents that do not win, do not pay: who wins, has to pay the second largest bid.

The following theorem holds:

Theorem 37 *An auction mechanism for known-single-minded agents is truthful if and only if it satisfies the four properties EXACTNESS, MONOTONICITY, CRITICAL and PARTICIPATION.*

Proof: In the proof of this theorem we deal only with known-single-minded agents.

Let us proof the two directions of theorem. Starting with the sufficient condition: let us consider a truthful mechanism for an auction and prove the four properties hold. The properties of EXACTNESS and PARTICIPATION are trivial to prove, so we prove the remaining part in the following two steps:

1. if the mechanism is truthful and the properties EXACTNESS and PARTICIPATION are true, then the property MONOTONICITY is true too;
2. if the mechanism is truthful and the properties EXACTNESS, PARTICIPATION and MONOTONICITY are valid, then the property CRITICAL holds too;

Let us start with point 1. By contradiction, let us suppose a mechanism truthful exists that holds the properties EXACTNESS and PARTICIPATION, but not MONOTONICITY. Then two offers exist for the agent j, $b_j = t_j$ e $b'_j < b_j$ so that $f_j(b_{-j}, b_j) = \emptyset$ and $f_j(b_{-j}, b'_j) \neq \emptyset$. Consider the utilities for the agent j with the two bids:

- For the bid b_j, because the mechanism is truthful and the agent j loses with bid b_j, follows:
$$u_j = v_j(f_j(b_{-j}, b_j)) - p_j(b_{-j}, b_j) = 0$$
- For the bid b'_j:
$$u'_j = v_j(f_j(b_{-j}, b'j)) - p_j(b_{-j}, b'_j)$$
because $t_j \leq v_j$, and because the mechanism is truthful, the agent does not pay more than it offers, then it results:
$$u'_j \geq t_j - b'_j > t_j - b_j = 0$$

Because the mechanism is truthful, it must be $u_j \geq u'_j$, so the mechanism is not truthful, which is a contradiction.

Let us consider the point 2. Let us show that if the mechanism is truthful and holds the properties EXACTNESS, PARTICIPATION and MONOTONICITY, then it holds the CRITICAL property too; it follows $\theta_j(b_{-j}, f) = p_j$ for any agent j.

Let us prove that for any j one critical value exists $\theta_j(b_{-j}, f)$ such as the agent j wins if and only if it declare a value greater than $\theta_j(b_{-j}, f)$. That means $f_j(b_{-j}, b_j) \neq \emptyset$ if and only if $b_j > \theta_j(b_{-j}, f)$. In order to prove the uniqueness of this value, we will show first that at least one critical value exists and then that cannot exist two different critical values.

To show that at least one critical value exists, let us suppose by contradiction that such a value does not exist. So for any $c > 0$, a winning declaration b_j and a losing declaration b'_j exist such that

$b_j \le c < b'_j$ or $b_j < c \le b'_j$, that is a contradiction because the property MONOTONICITY is true. Let us prove now that no two critical values exist $\theta_j^1 < \theta_j^2$ exist. Consider the declaration $\frac{\theta_j^1+\theta_j^2}{2}$ and observe that at the same time it is winning (because $\frac{\theta_j^1+\theta_j^2}{2} > \theta_j^1$ which is a critical value) and losing (because $\frac{\theta_j^1+\theta_j^2}{2} < \theta_j^2$ which is a critical value), that is a contradiction.

Now, to complete the proof we must show that the payment is equal to critical value, that is $p_j = \theta_j(b_{-j}, f)$. By contradiction $p_j(b) = \theta_j + \epsilon$ for any $\epsilon > 0$ (recall b represents the vector of bids when the agent j says the truth). Because the mechanism is voluntary participation, it follows $f_j(b) \neq \emptyset$. By critical value definition, we have that the bid $b'_j = \theta_j + \frac{\epsilon}{2}$ is winning that means $f_j(b_{-j}, b'_j) \neq \emptyset$. So the utilities in both cases will be:

- if the agent bids truthfully, trivially we have:

$$u_j = t_j - p_j = t_j - \theta_j - \epsilon$$

- if the agent bids $b'_j = \theta_j + \frac{\epsilon}{2}$, because the declaration b'_j is winning and the voluntary participation holds (therefore no agent pays more than it offers), it follows:

$$\begin{aligned} u'_j &= t_j - p_j \ge t_j - \theta_j - \frac{\epsilon}{2} > \\ &> t_j - \theta_j - \epsilon = u_j \end{aligned}$$

so the mechanism will not be truthful, that is a contradiction.

Let us prove now that conditions are necessary, that is any mechanism for known-single-minded auction that holds the properties EXACTNESS, MONOTONICITY, CRITICAL and PARTICIPATION, is truthful. Let $b_j = t_j$ and $b'_j \neq b_j$ be two bids of the agent j and $f_j(b_{-j}, b_j) = S_j$ and $f_j(b_{-j}, b'_j) = S'_j$. Then we have three possible situations:

1. if $S_j = S'_j$, because the mechanism is critical, it results: $p_j = p'_j$, and $u_j = u'_j$
2. if $S_j \neq \emptyset$ e $S'_j = \emptyset$, because the mechanism is critical it follows $b'_j < \theta_j \le b_j$, and because the voluntary participation holds,

$$u'_j = 0 \le b_j - \theta_j = t_j - \theta_j = u_j$$

3. if $S_j = \emptyset$ and $S'_j \neq \emptyset$, the mechanism is critical, so $b_j < \theta_j \le b'_j$, and because the mechanism is voluntary participation,

$$u'_j = t_j - \theta_j = b_j - \theta_j < 0 = u_j$$

it follows the mechanism is truthful, completing the proof. ■

Thus, in order to design a truthful mechanism for the auctions with known-single minded agents, we only need an allocation algorithm that holds the properties EXACTNESS e MONOTONICITY and a payment schema that holds CRITICAL and PARTICIPATION.

We give a preliminary definition to describe the algorithm. Let the welfare of a set of bids the sum of the valuations of all agents. i.e. $welfare(b_{-j}, b_j) = \sum_{i\in N} v_j(f_j(b_{-j}, b_j))$. Let us start with the following definitions:

Definition 38 *A monotone allocation algorithm A is bitonic if $\forall\ j \in N$, $\forall\ b_{-j}$ one of following conditions holds:*

1. *$welfare(b_{-j}, b_j)$ is a non-increasing function for $b_j < \theta_j(b_{-j}, f)$ and a non-decreasing for $b_j \geq \theta_j(b_{-j}, f)$*
2. *$welfare(b_{-j}, b_j)$ is a non-increasing function for $b_j \leq \theta_j(b_{-j}, f)$ and is a non-decreasing function for $b_j > \theta_j(b_{-j}, f)$.*

Notice that a monotone allocation algorithm is not necessarily bitonic. The following example gives a monotonic allocation algorithm that is not bitonic.

Example 2

Let us consider the algorithm:

```
**XOR-algorithm**$(c, i, j, k)$
**begin**
    **if** $(v_i(b) < c)$ **then**
        $j$ wins
    **else**
        **if** $(v_i(b) < 2c)$ **then**
            $k$ wins
        **else**
            $i$ wins
        **endif**
    **endif**
**end**
```

This algorithm is monotone (the critical value for the agents j and k can be both 0 and ∞, while the critical value of agent i is $2c$). Anyway, the welfare of agent i may be increasing in the interval $[0, 2c)$, so the XOR-algorithm is not bitonic. ■

There are several allocation algorithms which are bitonic. For example, the following are two bitonic allocation algorithm:

- **EXHAUSTIVE-k:** selects, through exhaustive search, the subset of k not-conflict bids in order to maximize the welfare.
- $\mathbf{G_k}$**:** executes greedy algorithm using as sorting rule: (rendering function)

$$l(b_j) = \begin{cases} b_j & \text{if } |S_j| \leq \sqrt{\frac{m}{k}} \\ 0 & \text{otherwise} \end{cases}$$

We now show how a bitonic algorithm can be obtained by combining bitonic algorithms. Let A_1 and A_2 be two allocation algorithms and consider the following algorithm:

```
MAX($A_1, A_2$)-algorithm
begin
    run $A_1$
    run $A_2$
    if ($welfare_{A_1}(b) > welfare_{A_2}(b)$) then
        return $A_1(b)$
    else
        return $A_2(b)$
    endif
end
```

By this algorithm, the following theorem may be prove.

Theorem 39 *If A_1 e A_2 are monotone and bitonic algorithms, then $MAX(A_1, A_2)$ is monotone e bitonic.*

Proof: Fix v_j and set $\theta' = \theta_j(A_1, v_{-j})$ and $\theta'' = \theta_j(A_2, v_{-j})$. Let us define three functions $f_1(y) = w_{A_1}(v_-j, y)$, $f_2(y) = w_{A_2}(v_-j, y)$ and $f_m(y) = w_M(v_-j, y) = \max\{f_i(y), f_2(y)\}$ that represent the welfares as function of the declaration y of bidder j.

Notice that if two functions are non-decreasing (non-increasing) the max of them is non-decreasing (non-increasing) too. Let us suppose that $\theta' \le \theta''$. By Definition 1, we have $\forall\ y < \theta'$ the function $f_m(y)$ is decreasing while $\forall\ y > \theta''$ the function $f_m(y)$ is increasing. Now consider the interval $I = [\theta', \theta'']$, we have the following cases:

- $\theta' = \theta''$. The function f_1 and f_2 have the same global minimum. So M is a bitonic allocation algorithm and follows $\theta_j(M, v_{-j})$ $=\theta' = \theta''$
- $\theta' < \theta''$ and $\forall\ y \in I$: $f_1(y) \le f_2(y)$. So $f_m(y) \equiv f_1()$. Then M is a bitonic as A_1 and $\theta_j(M, v_{-j})$ $=\theta'$.
- $\theta' < \theta''$ and $\forall\ y \in I$: $f_1(y) < f_2(y)$. So $f_m(y) \equiv f_2()$. Then M is a bitonic as A_2 and $\theta_j(M, v_{-j})$ $=\theta''$.
- Otherwise we must have $f_1(\theta') \le f_2(\theta')$ and $f_2(\theta'') \le f_1(\theta'')$. Let us define $J = \{y | y \in I, f_1(y) \le f_2(y)\}$ and let $\theta_0 = sup$ J. J is not empty, and $\theta' \le \theta_0 \le \theta''$. We may say $\forall\ y \in [\theta', \theta_0]$: $f_2(y) \le f_1(y)$ and $f_m() = f_2()$ is decreasing. At the same way $\forall\ y \in [\theta_0, \theta'']$: $f_1(y) \ge f_2(y)$ and $f_m() = f_1()$ is increasing. M is bitonic and $\theta_j(M, v_-j) = \theta_0$.

■

¿From this theorem we can easily derive the following theorem.

Corollary 40 *Let $A_1, A_2, \cdots, A_n$ be allocation algorithms. If $A_1, A_2, \cdots, A_n$ are monotone and bitonic, then $MAX(A_1, A_2, \cdots, A_n)$ is monotone and bitonic.*

Proof: If the max of two monotone and bitonic algorithms is a monotone and bitonic algorithm too, it inductively follows that the max of a number of monotone and bitonic algorithms is a monotone and bitonic algorithm again. ■

Now, for any fixed $k > 0$, consider the following algorithm:

```
k-APPROX-CA(b)-algorithm
    return MAX(EXHAUSTIVE-$k(b)$,$G_k(b)$)
end
```

For this algorithm, the following theorem holds.

Theorem 41 *The mechanism with k-APPROX-CA , $k = \lfloor \frac{4}{\epsilon^2} \rfloor$ as the allocation algorithm and the associated critical value payment scheme is an $\epsilon\sqrt{m}$-APX truthful mechanism.*

Proof: The algorithms EXHAUSTIVE-k and G_k are monotone and bitonic and then k-APPROX-CA is monotone and bitonic from theorem 39. Using the proof **MAX WEIGHTED IND. SET** (HALLDORSON) follows that it is $\epsilon\sqrt{m}$-APX. Then, due to the fact that the mechanism is monotone and exact, and value payment schema is critical and voluntary participation, it follows the mechanism is truthful from theorem 37. ■

Using similar techniques, Surely and Nisan (see [12]) have also proved the following results:

- For the MULTI UNIT SIMPLE AUCTION (actions where there are more occurrences of one good) the mechanism truthful is 2-APX It uses the **MAX(**G_d**,**G_k**)**-algorithm, where G_d is greedy algorithm with the Lehemann's rendering function and G_k is greedy algorithm with the Nisan's rendering function.
- For the MULTI UNIT COMBINATORIAL AUCTION (actions where there are more goods and more occurrences of them) a LP and LARGEST-based $(m + 1)$-APX truthful mechanism is known.

Lecture 7

Competitive auctions

TBC

Lecture 9

Scheduling on related machines

In this lecture we present some results on approximated mechanism for scheduling related machines. We start by reviewing the basic definitions of *mechanism* and *truthful mechanism.*

9.1 Basic definitions

We consider a system with n agents indexed by $i = 1, \cdots, n$ and a set of outcomes $\mathcal{O}$. Agent i has a *type* $\theta_i \in \Theta_i$ that determines its *utility* over possible different outcomes by means of the *utility function* $u_i : \Theta_i \times \mathcal{O} \to \mathcal{R}$. More precisely, for each possible type $\theta_i \in \Theta_i$ of agent i and for each possible outcome $o \in \mathcal{O}$, $u_i(\theta_i, o)$ is a measure of how much agent i likes outcome o. Thus, we say that *agent i prefers outcome o_1 to outcome o_2* if and only if $u_i(\theta_i, o_1) > u_i(\theta_i, o_2)$.

The *global goal* is to compute a function $f : (\Theta_1 \times \cdots \times \Theta_n) \to \mathcal{O}$ that selects an *optimal* outcome based on the types of the agents. For *utilitarian* problems *optimal* means that the sum of the utility of the agents (the total wealth of the system) is maximized. That is,

$$f(\theta_1, \cdots, \theta_n) = \arg\max_{o \in \mathcal{O}} \sum_i u_i(o, \theta_i).$$

As we shall see, the problem of scheduling on related machine is not optimal as we try to minimize the *makespan* (which corresponds to the maximum finish time of all the machines).

Achieving the global goal in a context in which the agents are not selfish is the object of investigation of the *classical* theory of algorithm design: each agent reveals its type and then the function f is computed. On the other hand, if agents are selfish they might lie about their type so to force an outcome in which their personal utility is maximized even though the global goal is not reached.

The mechanism design problem consists in implementing the *rules of a game* (this is done by defining the possible strategies available to each user and how the outcome is selected based on the agents strategies) so to reach the global goal despite the fact that the agents pick their strategies so to maximize their own utility.

We start by defining the notion of a mechanism.

Definition 42 (Mechanism.) *A mechanism $\mathcal{M} = (A_1, \cdots, A_n, o, p_1, \cdots, p_n)$ consists of n sets of* strategies *$A_1, \cdots, A_n$, an* outcome rule *$o : A_1 \times \cdots \times A_n \to \mathcal{O}$ and* payment functions *$p_i : A_1 \times \cdots \times A_n \to \mathcal{R}$.*

If agents pick strategies $(a_1, \cdots, a_n) \in A_1 \times \cdots \times A_n$ then the outcome defined by $\mathcal{M}$ is $o = o(a_1, \cdots, a_n)$ and user i receives a payment $p_i(a_1, \cdots, a_n)$.

For a vector $a = (a_1, \cdots, a_n)$ of strategies, will denote by a^{-i} the vector $(a_1, \cdots, a_{i-1}, a^{i+1}, \cdots a_n)$. The writing (a^{-i}, b) denotes the vector $(a_1, \cdots, a_{i-1}, b, a^{i+1}, \cdots a_n)$.

Various notions have been put forth in Game Theory to model the behavior of selfish agents: Nash equilibrium, dominant strategies, Bayesian-Nash equilibrium. The notion of dominant strategy is the most used within the theory of mechanism design.

Definition 43 (Implementation by dominant strategy.) *We say that a mechanism* $\mathcal{M} = (A_1, \cdots, A_n, o, p_1, \cdots, p_n)$ *is an* implementation with dominant strategy *for the social function* f *if*

1. *for every agent i and each type $\theta_i \in \Theta_i$ there exists a* dominant *strategy $a_i \in A_i$. That is, a strategy a_i such that for every strategy a^{-i} of the other agents and for every $b_i \in A_i$ we have that*

$$v_i(o, \theta_i) + p_i \geq v_i(o', \theta_i) + p'_i,$$

   where $o = o(a^{-i}, a_i)$, $o' = o(a^{-i}, b_i)$, $p_i = p_i(a^{-i}, a_i)$, $p'_i = p_i(a^{-i}, b_i)$.

2. $o(a_1, \cdots, a_n) = f(\theta_1, \cdots, \theta_n)$.

In other words, a dominant strategy for an agent i is guarantee to be the best strategy (that is, the one that guarantees the highest utility) independently from the strategy of the other agents.

9.1.1 The revelation principle

The simplest type of mechanisms are those in which, for each agent, the set of possible strategies coincide with the set Θ_i of their types. These mechanisms are called *direct revelation mechanism.* Among these of great importance are the truthful mechanisms.

Definition 44 (Truthful mechanism.) *We say that a mechanism is* truthful *if*

1. *for all agents $A_i = \Theta_i$;*
2. *strategy $a_i = \theta_i$ is dominant.*

The following is a classical result from Microeconomic Theory.

Theorem 45 (Relevation mechanism; see [7]) *If a given problem admits an implementation with dominant strategy then it admits also a truthful implementation.*

9.2 Scheduling related machines

In this lecture we are mainly interested in the problem of scheduling related machines. The classical scheduling problem can be described as follows. We are given a set of m jobs of length $t_1 \leq t_2 \leq \cdots \leq t_m$ and a n machines of speeds $s_1, \cdots, s_n$. A *schedule* of the m jobs on the n machines is a function $S : \{1, \cdots, m\} \to \{1, \cdots, n\}$ that specifies one machine for each job. The *makespan* of a job is defined as

$$MS(S) = \max_{i=1,\cdots,n} s_i \sum_{j:S(j)=i} t_j.$$

The problem of computing a schedule of minimum makespan is NP-hard.

Remark 2 *The definition above is different from usual definitions of the scheduling problem. In fact, typically the speed of a machine denotes the amount of work it can perform in one unit of time and thus a machine of speed s takes time w/s to complete a load w. Instead we assume, without loss of generality, that the speed is the amount of time it takes to perform one unit of work. Thus a machine of speed s takes time $s \cdot w$ to complete load w.*

We consider the problem of designing a truthful mechanism for computing a schedule with small makespan in the case in which machines are owned by selfish agents and the speed of machine is the type (private information) of the agent. In this case a mechanism elicits the declared speed b_i from each machine i and assigns a load to each of them along with a payment P_i. Fixing the declared speeds b_{-i} of all the machines but the i-th, we can define the load function $w_i(b)$ as the function that specifies the load of machine i as function of the declared speed b and denote by $P_i(b)$ the payment received by machine i to perform the work assigned. When we wish to make explicit the dependence of the load and of the payment on the declared speeds of the other machines we will use the writings $w_i(b_{-i}, b_i)$ and $P_i(b_{-i}, b_i)$. Being selfish, machine i of speed s_i will declare a speed b_i that maximizes her profit $G_i(b_i|s_i) \overset{\text{def}}{=} P_i(b_i) - s_i \cdot w_i(b_i)$.

We have the following theorem.

Theorem 46 ([1]) *A mechanism for scheduling on related machines is truthful if and only if for all i the load function w_i is non-increasing for all i. In this case the payments $P_i(b_{-i}, b_i)$ are of the form*

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

where the h_i's are arbitrary functions.

Proof: Suppose for sake of contradiction that the load function is not non-increasing; that is, for some $x < y$, it holds that $w_i(x) < w_i(y)$. We show that in this case there are no payments that make the mechanism truthful. For the mechanism to be truthful, the payment function must be such that

1. When the real speed of the machine i is x it must be convenient to report x instead of y. That is,
$$G_i(x|x) \geq G_i(x|y)$$
that is
$$P_i(x) - w_i(x) \cdot x \geq P_i(y) - w_i(y) \cdot x,$$
which is equivalent to
$$P_i(x) - P_i(y) \geq (w_i(x) - w_i(y)) \cdot x.$$

2. When the real speed of the machine i is y it must be convenient to report y instead of x. That is,
$$P_i(y) - w_i(y) \cdot y \geq P_i(x) - w_i(x) \cdot y,$$
which is equivalent to
$$P_i(x) - P_i(y) \leq (w_i(x) - w_i(y)) \cdot y.$$

By combining the two conditions we obtain

$$(w_i(x) - w_i(y)) \cdot x \leq (w_i(x) - w_i(y)) \cdot y$$

which implies $x \geq y$. Contradiction.

Let us now prove that if the load function in nondecreasing then the payment specified in the statement make the mechanism truthful.

Suppose that the real speed of the i-th machine is x and suppose that the i-th machine declares speed y. In this case machine i earns

$$G_i(y|x) = c + (y - x)w_i(y) - \int_0^y w_i(u)du.$$

If instead machine i declares her real speed, she gains

$$G_i(x|x) = c - \int_0^x w_i(u)du.$$

Then, if $y > x$, we have that

$$G_i(x|x) - G_i(y|x) = w_i(y)(x - y) + \int_x^y w_i(u)du \geq 0,$$

since w_i is non-increasing. Therefore machine i has no incentive in declaring a larger speed.

Let us now consider the case in which $y < x$.

$$G_i(x|x) - G_i(y|x) = w_i(y)(x - y) - \int_y^x w_i(u)du \geq 0$$

■

At the light of the previous theorem we have reduced the problem of designing a truthful mechanism to that of designing a *monotone* scheduling algorithm.

The following theorem tells us that the algorithm that computes the minimum makespan scheduling is indeed *monotone*.

Theorem 47 *The algorithm that returns the lexicographically minimum scheduling with the shortest makespan is monotone.*

Known algorithms from the literature are easily seen to be non-monotone.

Greedy algorithm [4]. Let us first consider the non ordered sequence of jobs $\sigma = 1, \epsilon, 1, 2 - 3\epsilon$, for $0 < \epsilon < 1/3$, to be allocated on two machines. The following table shows how `Greedy` allocates the jobs with respect to different speeds.

speeds (s_1, s_2)	machine 1	machine 2
$(1, 1)$	ϵ, 1	1,$2 - 3\epsilon$
$(1, 1/2)$	ϵ, $2 - 3\epsilon$	1,1

Clearly, machine 2 gets more work when declaring speed $s_2 = 1$ than declaring speed $s_2 = 2$, thus implying the non-monotonicity of `Greedy` when processing jobs without ordering them.

We now consider ordered jobs but arbitrary speeds. First, consider jobs $3, 2, 2$ and two machines with speeds $(1, s_2)$. For any $1 < s_2 < 5/4$, `Greedy` will assign both jobs of weight 2 to the slower machine, and thus it gets more load than the faster one. As for the case of jobs processed in non decreasing order, consider weights $1, 2, 5/2$ and speeds $(1, 4/3)$. It is easy to see that machine 1 will get jobs of weight 1 and $5/2$. Observe that, in both cases we are using speeds whose ratio is smaller than 2.

9.2.1 A monotone algorithms

In this section we present a simple monotone algorithm EZ for the case of two machines. The algorithm receives two speeds s_1, s_2 and distinguishes two cases. We denote by $\phi = 1,61\ldots$ the golden ratio.

Case I. $\max\{s_1, s_2\}/\min\{s_1, s_2\} \leq \phi$.

In this case the algorithm runs the PTAS for two machine and obtains a $(1+\epsilon)$-approximation for two machines of equal speed. Let L_1 and L_2 be the loads of the two machines. Then the algorithm assigns the larger of the two loads to the machine of speed $\min\{s_1, s_2\}$ and the smaller of the two to the machine of speed $\max\{s_1, s_2\}$.

Case II. $\max\{s_1, s_2\}/\min\{s_1, s_2\} > \phi$.

In this case all the jobs are assigned to the machine of speed $\max\{s_1, s_2\}$.

We next show that the algorithm is monotone and then prove a bound on its approximation ratio.

Lemma 48 *Algorithm* EZ *is monotone.*

Proof: If the slower of the two machines increases her speed (*i.e.*, it becomes even slower) then the following cases are possible.

1. EZ stays in Case I. In this case the machine get the same load.
2. EZ stays in Case II. In this case the machine gets the same load; that is, 0.
3. EZ switches from Case I to Case II. In this case the load of the machine becomes 0 and thus does not increase.

Thus, in no case the load of the machine increases.

If instead the slower of the two machines decreases her speed (*i.e.*, becomes faster) while remaining the slower of the two, nothing changes. If instead the machine becomes the faster of the two then it will be certainly assigned more load.

The case in which the fastest of the two machines changes her speed can be argued similarly. ■

Now we give a bound on the approximation ratio of the algorithm.

Lemma 49 *Algorithm* EZ *is ϕ-approximated.*

Proof: Let us assume without loss of generality that $1 = s_1 \leq s_2$.

In Case I, we have that the algorithm produces an s_2-approximation with $s_2 \leq \phi$. In Case II, instead the algorithm outputs a $(1 + 1/s_2)$-approximation with $s_2 \geq \phi$. ■

Other algorithms. A more general algorithm is found in [1] where the authors present a 3-approximation algorithm for any number of machines. The algorithm is randomized and is *truthful on average* which is a weaker notion than the one discussed in this lecture. Later Auletta *et. al*, presented a deterministic 4-approximation algorithm for any fixed number of machines.

Lecture 10

Online mechanisms

In this lecture we present a result of [2] that reduces the design of *online* truthful mechanisms to the design of *online* algorithms. This reduction comes very hand since several optimization problems have good online algorithms.

10.1 The model

We consider the general problem of *single-option auctions* in which requests come from agents and each request is represented by a pair (r_i, b_i) where r_i is a description of the resource requested and b_i is the agent's bid; that is, the amount that the agent is willing to pay for the resource. Agent i has a private valuation v_i that describes how much the agent is really willing to pay. The mechanism decides either to grant the request in which case the price $p_i \geq 0$ that the agent has to pay is decided; or to reject the request in which case the agent does not have to pay, *i.e.* $p_i = 0$. The utility u_i of agent i is $v_i - p_i$ if the request is accepted and 0 if the request is rejected.

10.2 Nice online algorithms

Definition 50 *An online algorithm for the single-option auctions is* nice *if*

1. *if a request (r, b) is accepted then also request (r, b') with $b' > b$ is accepted;*
2. *the decision of whether request i is accepted may not depend on the bids of the requests accepted in the past. It may depend though on the set of requests already accepted and by the bids of the rejected requests.*

Theorem 51 *If there exists a deterministic online ρ-competitive algorithm then there exists a* nice *online ρ-competitive algorithm.*

Proof: Let $\mathcal{B}$ be a deterministic online ρ-competitive algorithm and consider the following algorithm $\mathcal{B}'$. When the i-th request (r_i, b_i) arrives the algoritm computes the smallest c_i such that if $\mathcal{B}$ was presented the sequence $(r_1, \min(b_1, c_1)), \cdots, (r_i, c_i)$ then $\mathcal{B}$ would have accepted request (r_i, c_i). Then $\mathcal{B}'$ accepts (r_i, b_i) if and only if $b_i \geq c_i$.

$\mathcal{B}'$ is clearly nice. Increasing the bid of an accepted request will not cause the request to be rejected. Whether a request is accepted or not depends on the previous requests and on the minimum between b_i and c_i. Thus the bid of accepted requests (those with $b_i \geq c_i$) do not influence the decision.

Let us now prove that for a stream $R = ((r_i, b_i))_{i>1}$, algorithm $\mathcal{B}'$ is ρ-competitive. Consider stream R' in which request (r_i, b_i) is replaced by request (r_i, d_i) with $d_i = \min(b_i, c_i))$.

First of all, observe that $A(\mathcal{B}', R) = A(\mathcal{B}, R')$. Indeed suppose that request the i-th request of R is rejected by $\mathcal{B}'$. Then it must be the case that $b_i < c_i$ and thus the i-th request of R' (that is (r_i, b_i)) is also rejected by $\mathcal{B}$. Suppose instead that the i-th request of R' is accepted by the $\mathcal{B}$. Then it must be the case that $d_i \geq c_i$ or, in other words, that $b_i \geq c_i$ and thus the i-th request (r_i, b_i) of R is accepted by $\mathcal{B}'$. Now let $A^*(R')$ be one of the optimal solution for the stream R'. Since $\mathcal{B}$ is ρ-competitive it must be the case that

$$\rho \sum_{i \in A(\mathcal{B}, R')} \min(b_i, c_i) \geq \sum_{i \in A^*(R')} \min(b_i, c_i). \tag{10.1}$$

Now we have that

$$\begin{aligned}
\sum_{i \in A(\mathcal{B}, R')} \min(b_i, c_i) &= \sum_{i \in A(\mathcal{B}, R')} b_i - \sum_{i \in A(\mathcal{B}, R')} (b_i - c_i) \\
&\quad (\text{since } \min(b_i, c_i) = c_i, \text{ for } i \in A(\mathcal{B}, R')) \\
&\quad \sum_{i \in A(\mathcal{B}', R)} b_i - \sum_{i \in A(\mathcal{B}', R)} (b_i - c_i) \\
&\quad (\text{since } A(\mathcal{B}, R') = A(\mathcal{B}', R))
\end{aligned}$$

For the righthand side of Equation 10.1, we observe that

$$\sum_{i \in A^*(R')} \min(b_i, c_i) = \sum_{i \in A^*(R')} b_i - \sum_{i \in A^*(R') \cap A(\mathcal{B}', R)} (b_i - c_i)$$

Therefore we have that

$$\begin{aligned}
\rho \sum_{i \in A(\mathcal{B}', R)} b_i &= \rho \sum_{i \in A(\mathcal{B}, R')} \min(b_i, c_i) + \rho \sum_{i \in A(\mathcal{B}, R')} (b_i - c_i) \\
&\geq \sum_{i \in A^*(R')} \min(b_i, c_i) + \sum_{i \in A(\mathcal{B}', R)} (b_i - c_i) \\
&= \sum_{i \in A^*(R')} b_i - \sum_{i \in A^*(R') \cap A(\mathcal{B}', R)} (b_i - c_i) + \sum_{i \in A(\mathcal{B}', R)} (b_i - c_i) \\
&\geq \sum_{i \in A^*(R')} b_i
\end{aligned}$$

and thus $\mathcal{B}'$ is ρ-competitive. ■

10.3 Truthful Online Mechanism for Single Option Auction

In this section we show how to compile an online ρ-competitive algorithm $\mathcal{B}$ into an online $(\rho + \log \mu)$-competitive truthful mechanism $\mathcal{B}'$, where μ is the largest possible bid (and 1 is the smallest possible).

```
1.  Set $m = 1$ with probability $1/2$ and $m = 2^i$ with probability $\frac{1}{2 \log \mu}$ for $i = 1, \cdots, \log \mu$.
2.  **foreach** request $(r_i, b_i)$ **do**
3.      let $c_i$ be the smallest $v$ such that $\mathcal{B}$ accepts $(r_i, v)$;
4.      if $b_i \geq mc_i$ then accept and charge $p_i = mc_i$;
5.      feed $(r_i, b_i)$ to $\mathcal{B}$ and update $\mathcal{B}$'s state;
```

10.3.1 Competitive analysis

We denote by A the set of requests accepted by algorithm $\mathcal{B}'$, by Q the set of requests with $b_i \geq c_i$ and by P the set of requests with $b_i < c_i$ which are part of the optimal solution. Clearly the profit of the optimal solution is upper bounded by

$$\sum_{i \in Q} v_i + \sum_{i \in P} v_i,$$

while the profit obtained by $\mathcal{B}'$ is given by

$$\sum_{i \in A} p_i.$$

Lemma 52

$$\sum_{i \in P} v_i \leq \rho \sum_{i \in Q} c_i.$$

Proof: Observe that Q is exactly the set of requests accepted by $\mathcal{B}$ on input the sequence of requests $((r_i, b_i))_i$. Since $\mathcal{B}$ is nice then $\mathcal{B}$ has the same behaviour on the sequence of requests $((r_i, \min(b_i, c_i)))_i$. Therefore by the ρ competitiveness of $\mathcal{B}$ we have that

$$\sum_{i \in P} b_i = \sum_{i \in P} \min(b_i, c_i) \leq \rho \sum_{i \in Q} \min(b_i, c_i) = \rho \sum_{i \in Q} c_i.$$

Then the lemma follows from the fact that the algorithm is truthfull and thus $b_i = v_i$. ■

Lemma 53

$$E\left[\sum_{i \in A} p_i\right] \geq \frac{1}{2} \sum_{i \in Q} c_i.$$

Proof: Consider a request $i \in A$. Then with probability $1/2$, $m = 1$ and thus i is accepted and it contributes c_i to the lefthand sum. ■

Lemma 54

$$\sum_{i \in Q} v_i \leq 4 \log \mu E\left[\sum_{i \in A} p_i\right].$$

Proof: With probability $\frac{1}{2 \log \mu}$ we have that $mc_i \leq v_i \geq 2mc_i$ in which case request i is accepted with a payment of $p_i \geq 1/2 v_i$. ■

We thus have the following theorem.

Theorem 55 *Algorithm $\mathcal{B}'$ is $(2\rho + 4 \log \mu)$ competitive.*

Bibliography

[1] Aaron Archer and Eva Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[2] Adam Meyerson Baruch Awerbuch, Yossi Azar. Reducing truth-telling online mechanisms to online optimization. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing, 2003, ACM*, 2003.

[3] E.H. Clarke. Multipart Pricing of Public Goods. *Public Choice*, pages 17–33, 1971.

[4] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[5] T. Groves. Incentive in Teams. *Econometrica*, 41:617–631, 1973.

[6] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In G. Meinel and S. Tison, editors, *Proceedings of the 16th Annual ACM Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, pages 404–413. Springer Verlag, March 1999.

[7] A. Mas-Collel, W. Whinston, and J. Green. *Microeconomic Theory.* Oxford University Press, 1995.

[8] M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing, 6–8 july , 2001, Crete, Greece, ACM*, pages 510–519, 2001.

[9] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proceedings of AAAI02*, 2002.

[10] T. Roughgarden and E. Tardos. How bad is selfish routing? In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, pages 93–102, 2000.

[11] W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.

[12] E. Zurel and N. Nisan. An efficient approximate allocation algorithm for combinatorial auction. In *Proceedings of the ACM Conference on Electornic Commerce*, 2001.