Algorithmic Game Theory Autumn 2021, Week 6

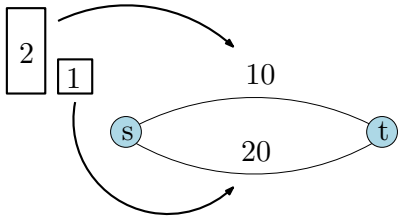
Truthful One-Parameter Mechanisms

ETH Zürich Paolo Penna

In this lecture we introduce a second technique for designing truthful mechanisms, in addition to the class of VCG mechanisms of the previous lecture. If you look carefully to the definition, you see that the algorithm is required to minimize the **social cost**, that is, the **sum** of all players' costs.

1 A simple problem (warm up)

Consider the following game. We have two jobs of a certain size we want to allocate to the two links and our optimization goal is to minimize the **maximum** cost among the players. For instance



where 10 and 20 are the cost for processing one unit of work, in the corresponding edge. So in this solution, both players have cost 20. In every other solution, one of the two players will have a strictly larger cost.

If we run VCG the solution is not what we want (it minimizes **sum** of costs)

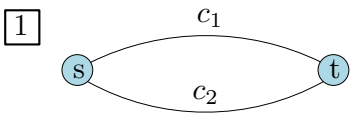
Exercise 1. *Try to use the formula of the VCG payments, but combined with the algorithm which minimizes the maximum cost in the previous example. Check that one of the two players can improve his/her utility by cheating, i.e., this naive approach does not give a truthful mechanism for our problem (assume the numbers on the edges are true costs).*

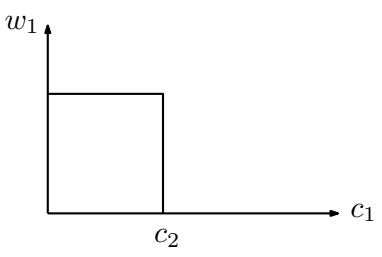
Truthful mechanism minimizing **maximum** cost?

1.1 Intuition (monotonicity)

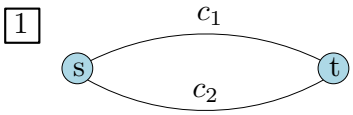
We gather some intuition by comparing the problem above with the problems studied in the previous lecture. In particular we look at what the **algorithm** does for the players:

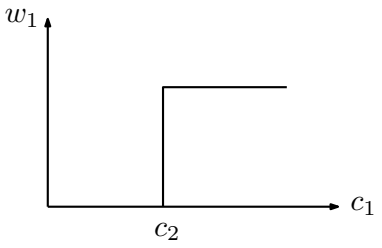
Shortest Path: (2nd price auction, VCG)



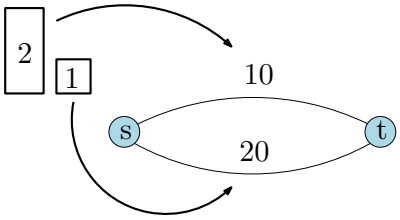


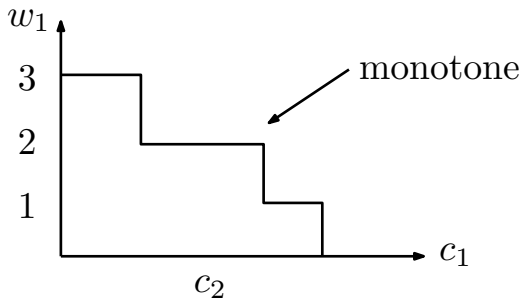
Longest Path (No mechanism!)





Two Jobs Problem: Maybe...





Truthful ⇔ Monotone Algorithm

Our new mechanism design technique will achieve the result above for a certain class of problems, and for a formal definition of 'monotone algorithm'.

2 One-Parameter Problems (formal setting)

We consider the following setting.

One-parameter problems

- Every player i has a **private** type

$$t_i \qquad \text{(single value)} \ ;$$

- Solution $a \in \mathcal{A}$ costs to player i an amount equal to

$$w_i(a) \cdot t_i$$

where $w_i(a)$ is the **work** that a allocates to i. The work is **public knowledge**. That is, the mechanism knows $w_i(a)$ for each $a \in \mathcal{A}$ and each i.

Example 1. *In the shortest-path problem, $w_i(a)$ is either 1 or 0 (the edge is selected or not-selected). In the jobs allocation problem above, $w_i(a)$ is the sum of all jobs weights that allocation a puts on machine i.*

Definition 2 (**monotone algorithm**). *We say that A is monotone, for a given one-parameter problem, if for all i, for all c_{-i}*

$$w_i(A(c_i, c_{-i}))$$

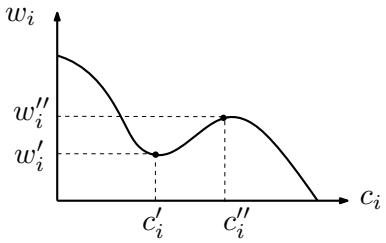
is monotone non-increasing in c_i.

Our main result is the following.

Theorem 3 (Myerson Lemma[1]). *For every one-parameter problem, it holds that*

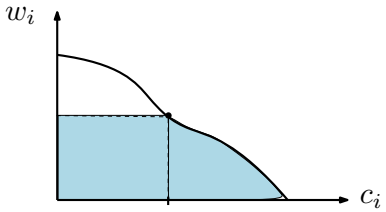
1. *A monotone $\Rightarrow$ (A, P) truthful (for suitable P) .*
2. *A not monotone $\Rightarrow$ (A, P) not truthful (no matter P) .*

Proof of Part 2. Suppose A is not monotone:



where in the y-axis we have $w_i = w_i(A(c_i, c_{-i}))$ for some c_{-i}. The rest of the proof uses the same arguments of the longest path problem in the previous lecture. (**Exercise!**) □

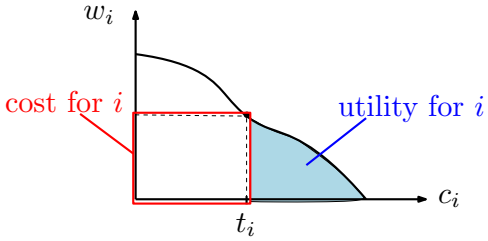
Proof of Part 1. Suppose A is monotone. The following payments will do the job:



in formulas

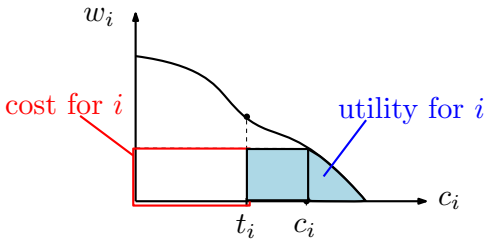
$$P_i(c_i, c_{-i}) = c_i \cdot w_i(A(c_i, c_{-i})) + \int_{c_i}^{\infty} w_i(A(u, c_{-i}))du \tag{1}$$

When player i is truth-telling, the utility is as follows:



[1]This is a slightly simpler version of the original result, and it is presented with the scheduling terminology by Archer and Tardos.

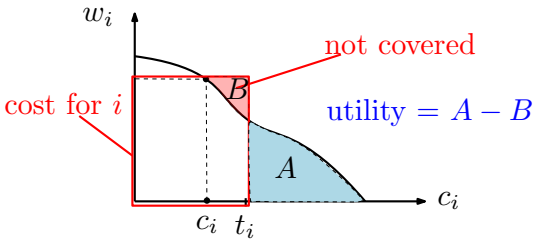
If i reports a higher cost, $c_i > t_i$, then the utility is as follows:



or in formulas

$$P_i(c_i, c_{-i}) - t_i \cdot w_i(A(c_i, c_{-i})) = (c_i - t_i) w_i(A(c_i, c_{-i})) + \int_{c_i}^{\infty} w_i(A(u, c_{-i}))du$$
$$\leq \underbrace{\int_{t_i}^{c_i} w_i(A(u, c_{-i}))du + \int_{c_i}^{\infty} w_i(A(u, c_{-i}))du}_{\text{utility when truth telling}}$$

If i reports a lower cost, $c_i < t_i$, then the utility is as follows:



which is at most the utility when truth telling (the A area). □

Remark 1 (payments)**.** *In the definition of the payments* (1)*, we need that the integral is finite. This is true if, for example, when c_i becomes sufficiently large, the algorithm does not put any work on this player. This may not be always the case (consider the shortest-path problem where removing edge i disconnects* `s` *from* `t`*). In such cases, the following generalization of* (1) *can be used to guarantee truthfulness:*

$$P_i(c_i, c_{-i}) = Q_i(c_{-i}) + c_i \cdot w_i(A(c_i, c_{-i})) - \int_0^{c_i} w_i(A(u, c_{-i}))du \tag{2}$$

where $Q_i()$ is an arbitrary function independent of c_i.

Remark 2 (voluntary participation)**.** *The above analysis shows that when a player is* ***truth-telling****, the mechanism with payments* (1) *guarantees a* ***nonnegative utility****. This condition is usually called* ***voluntary participation****.*

2.1 Selfish Related Machines

We now consider the following one-parameter problem. We have k jobs of size $J_1, J_2, \ldots, J_k$ and a set of n machines (players). Each machine i has a type t_i and an allocation a of jobs to machines specifies the amount of work $w_i(a)$ which is allocated to machine i (the sum of all jobs weights that a puts on machine i). The parameter t_i is the cost (time) required by machine i to process one unit of work, and therefore the cost of player i for allocation a is $w_i(a) \cdot t_i$.

Goal: Minimize **maximum** cost or **makespan**

We want an allocation minimizing the **makespan** or **maximum** cost:

$$makespan(a, t) := \max_i t_i \cdot w_i(a)$$

Since players can misreport their types, we want a truthful mechanism. All we need to do is to design a monotone algorithm which minimizes the makespan.

Example 4. *Suppose we have jobs of size* $100, 2, 1$ *to be allocated on three machines. Consider an algorithm which allocates these jobs as follows in these types:*

types	*1*	*2*	*10*
allocation	*100*	*1*	*2*

types	*1*	*10*	*10*
allocation	*100*	*2*	*1*

In both cases the algorithm has an optimal makespan, but it is ***not monotone****.*

The previous example suggests that we have to consider some kind of optimal algorithms.

Theorem 5 (Archer-Tardos)**.** *There exists a monotone algorithm minimizing the makespan on related machines, and therefore an exact truthful mechanism for the problem of scheduling selfish related machines.*

Proof. Consider the set of all allocations, sorted in some order

$$\mathcal{A} = \{a^1, a^2, \ldots, a^N\}$$

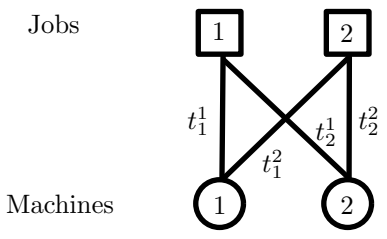
The algorithm returns the first (lexicographically minimal) allocation among those minimizing the makespan with respect to the input costs $c_1, \ldots, c_n$. That is, the minimum s such that

$$makespan(a^s, c) \leq makespan(a^h, c)$$

for all h. We argue that this algorithm A must be monotone (**Exercise!**). □

3 Multi-Parameter Problems (unrelated machines)

In this section we consider a natural extension of the above problem called *unrelated* machine scheduling.



Here the time to process job j on machine i is some number t_i^j and there is no particular relation between t_i^j and say $t_i^{j'}$. The type of a machine is a (private) vector

$$t_i = (t_i^1, t_i^2, \ldots, t_i^k).$$

Each player i can report any cost vector $c_i = (c_i^1, \ldots, c_i^k)$, that is, a different cost for each job. For any job allocation a, the cost for player i is the total time required by machine i to process the jobs it gets:

$$t_i(a) := \sum_{j=1}^{k} t_i^j \cdot a_i^j$$

where

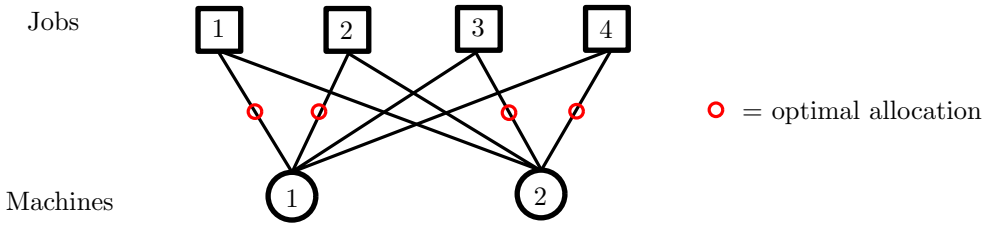
$$a_i^j = \begin{cases} 1 & \text{if } a \text{ allocates } j \text{ to machine } i \\ 0 & \text{otherwise} \end{cases}$$

The **makespan** or **max** players' cost with respect to $t = (t_1, \ldots, t_n)$ is

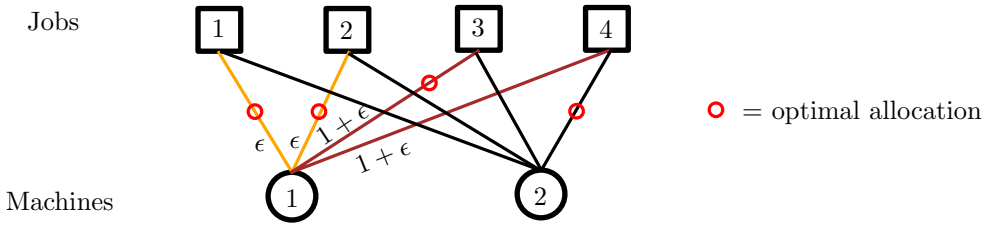
$$makespan(a, t) = \max_i t_i(a) = \max_i \sum_j t_i^j \cdot a_i^j \quad .$$

Theorem 6 (Nisan-Ronen)**.** *There is no exact truthful mechanism for scheduling selfish unrelated machines.*

Proof. By contradiction, suppose (A, P) is a truthful mechanism and A minimizes the makespan. Consider this instance with 4 jobs and 2 machines in which all t_i^j's are equal to 1. The optimum makespan algorithm must allocate two jobs per machine, say like in this picture:



Now change the cost of player 1 as follows. The previously allocated jobs cost ϵ and the non-allocated cost $1 + \epsilon$, for tiny $\epsilon > 0$:



We show that truthfulness cannot be achieved by considering $c_1 = (1, 1, 1, 1)$ and $c_1' = (\epsilon, \epsilon, 1 + \epsilon, 1 + \epsilon)$. Truthfulness requires to consider the two cases for player 1:

1. c_1 is the true cost and player 1 reports c_1':

$$P_1(c_1, c_2) - 2 \geq P_1(c_1', c_2) - 3$$

2. c_1' is the true cost and player 1 reports c_1:

$$P_1(c_1', c_2) - (1 + 3\epsilon) \geq P_1(c_1, c_2) - 2\epsilon$$

By summing up these inequalities we get a contradiction $-3 - 3\epsilon \geq -3 - 2\epsilon$. □

Remark 3. *Note that the above negative result does not require any computational assumption (that is, it also holds if we consider exponential-time algorithms).*

It is natural to ask if one can have **approximation** truthful mechanisms for this problem. That is, a truthful mechanism (A, P) such that the algorithm guarantees

$$\frac{makespan(A(c), c)}{opt_{makespan}(c)} \leq \alpha$$

where $opt_{makespan}(c) := \min_{a \in \mathcal{A}} makespan(a, c)$, for some constant α.

Remark 4. *For n unrelated machines, the VCG mechanism is an n-approximation truthful mechanism. (**Exercise!**)*

By looking at the proof of Theorem 6, it is possible to show that for 2 machines, a 2-approximation is the best possible.

Corollary 7. *Even for two unrelated machines, there is no α-approximation truthful mechanism with $\alpha < 2$.*

Proof. Consider a larger number of jobs, and use the same arguments of the proof to conclude that if we modify the optimal allocation in one of the two cases, the approximation ratio will necessarily be worse that $2 - \delta$. (**Exercise!**) □

4 Main Points of This Lecture

We have now two techniques to construct truthful mechanisms:

1. VCG mechanisms which minimize the sum of players' costs (previous lecture);
2. In one-parameter setting, construct a monotone algorithm (this lecture);

We have also seen techniques to prove impossibility results:

1. An algorithm cannot be transformed into a truthful mechanism (monotonicity);
2. No mechanism at all (makespan minimization in unrelated machine scheduling).

These negative results are proved in essentially the same way (see also longest path problem in previous lecture).

Finally, the results on scheduling problems suggest that multi-parameter problems are more difficult than one-parameter ones (i.e., related vs unrelated machines).

Related Literature

For the monotonicity in one-parameter problems:

- R. Myerson, Optimal Mechnism Design, Mathematics of Operations Research, 6:58–73, 1981. (Original characterization of truthful mechanisms for one-parameter settings)
- A. Archer and É. Tardos, Truthful Mechanisms for One-Parameter Agents. FOCS 2001. (Characterization of truthful mechanisms, which is deemed more accessible to computer scientists, including the results on scheduling related machines)

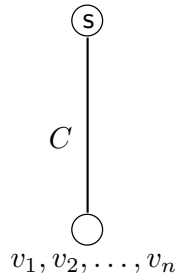
The unrelated machines problem is studied in:

- N. Nisan and A. Ronen, Algorithmic mechanism design. STOC 1999. (First inapproximability results, presented in this lecture. This is the first paper on algorithmic mechanism design.)
- A. Mu'alem and M. Schapira, Setting lower bounds on truthfulness. SODA 2007. (Improved lower bounds, also for several other problems in both one-parameter and multi-parameter versions)
- I. Gamzu, Improved lower bounds for non-utilitarian truthfulness. Theoretical Computer Science 412.7 (2011): 626-632. (Inapproximability of three unrelated machines, and other problems)
- I. Ashlagi, S. Dobzinski, R. Lavi, Optimal Lower Bounds for Anonymous Scheduling Mechanisms. Mathematics of Operations Research 37.2 (2012): 244-258. (Strong inapproximability results for a very 'natural' class of mechanisms)
- G Christodoulou, E Koutsoupias, A Kovács. On the Nisan-Ronen conjecture for submodular valuations. STOC 2020. (Strong inapproximability when one machine has a 'slightly more general' cost function.)

Exercises
(during the exercise class - 2.11.2021)

We shall discuss and solve together this exercise.

Exercise 2. *Consider again the scenario with a server* s *and* n *potentially interested users (see previous exercise):*



Discuss which truthful mechanism whose payments are as close as possible to the cost C *of providing the service.*

Note: *We shall discuss together which of these mechanisms make sense in practice, and refine the problem and the mechanism.*