On Mechanisms for Scheduling on Unrelated Machines

Elias Koutsoupias[1]

Department of Informatics and Telecommunications
University of Athens
http://www.di.uoa.gr/-elias

ACAC 2006/08/22

[1] Joint work with George Christodoulou and Angelina Vidali

Mechanism design

- Mechanism design is inverse engineering in Game Theory
- Given a goal, design a game so that when the players play the game the goal is the equilibrium
- Here we consider dominant equilibria (i.e., a player has an optimal strategy, no matter what the other players do)

Example: Single-item Auction

- We want to sell an object to n players (buyers).
- Each player has a value for the object, which is known only to him
- Objective: Give the item to the player with the highest value

Solution

- The players declare a value (they bid for the object once)
- Allocate the object to the player with the highest bid
- The player pays the second highest value

Truthful mechanisms

Definition (Truthful mechanisms)

A mechanism is truthful if revealing the true values is dominant strategy of each player

Theorem (The revelation principle)

For every mechanism there is an equivalent truthful one

Why? We design a new (truthful) mechanism which first simulates the lying strategies of the players and then apply the original mechanism. The players would tell the truth to this mechanism.

Combinatorial Auction

- There are n players (bidders) and m objects (items)
- Each player i has a value $u_i(S)$ for each subset (bundle) S of the objects. These are private values.
- Objective: Allocate the objects to the players to maximize the sum of the values of their bundles.

Protocol

- The players declare their values
- The mechanism allocates the objects (allocation algorithm)
- The mechanism pays the players based on the declared values and the allocation (payment algorithm)

- A mechanism consists of two parts: the allocation algorithm and the payment algorithm.

Combinatorial Auction (cont.)

- There is a truthful mechanism to achieve the objective:

The VCG mechanism

- Allocate the objects optimally
- The players do not pay full price, but they get a discount equal to the added value of their participation.

- The problem with this mechanism is that it is NP-hard to compute the optimal solution (and it has a high communication cost).

Open Problems

- Design a mechanism that achieves allocations with good approximation ratio and it has low computational and communication complexity
- Characterize the allocation algorithms of the truthful mechanisms.

Scheduling unrelated machines

- There are n players (machines) and m objects (tasks)
- Each player i has a (private) value t_{ij} for each task j
- Objective: Allocate the tasks to the players to minimize the maximum value among the players (i.e., the makespan)

Protocol

- The players declare their values
- The mechanism allocates the objects (allocation algorithm)
- The mechanism pays the players based on the declared values and the allocation (payment algorithm)

History of scheduling unrelated machines

- It is a well-studied NP-hard problem
- Nisan and Ronen in 1998 initiated the study of its mechanism-design version.
  - They gave an upper bound (a mechanism) with approximation ratio n
  - They gave a lower bound of 2
  - They conjectured that the right answer is the upper bound
  - They also gave a randomized mechanism with approximation ratio $7/4$ for 2 players

The related machines problem

- Archer and Tardos considered the related machines problem
- In this case, for each machine there is a single value (instead of a vector), its speed.
- They gave a variant of the (exponential-time) optimal algorithm which is truthful
- They also gave a polynomial-time randomized 3-approximation mechanism, which was later improved by Archer to 2-approximation.
- Andelman, Azar, and Sorani gave a 5-approximation deterministic truthful mechanism.

Main result

Theorem

There is no deterministic mechanism for the scheduling problem of unrelated machines with approximation ratio less than $1+\sqrt{2}$.

- This is the first improvement of the results of Nisan and Ronen
- It still leaves wide open the question (the approximation ratio is between 2.41 and n)

The setting

Input	Output
$t = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \cdots \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{pmatrix}$	$x = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}$
$t_{ij} \in \mathbb{R}^+$	$x_{ij} \in \{0, 1\}$ $\sum_i x_{ij} = 1$

Truthful = Monotone

Definition (Monotonicity Property)

An allocation algorithm is called monotone if it satisfies the following property: for every two sets of tasks t and t' which differ only on machine i (i.e., on the i-the row) the associated allocations x and x' satisfy

$$(x_i - x_i') \cdot (t_i - t_i') \leq 0,$$

where $\cdot$ denotes the dot product of the vectors, that is, $\sum_{j=1}^{m}(x_{ij} - x_{ij}')(t_{ij} - t_{ij}') \leq 0$.

Theorem (Nisan, Ronen 1998)

Every truthful mechanism satisfies the Monotonicity Property.

Theorem (Saks, Lan Yu 2005)

Every monotone allocation algorithm is truthful (i.e. it is part of a truthful mechanism).

Monotone algorithms

- Monotonicity, which is not specific to the scheduling task problem but it has much wider applicability, poses a new challenging framework for designing algorithms.
- In the traditional theory of algorithms, the algorithm designer could concentrate on how to solve every instance of the problem by itself.
- With monotone algorithms, this is no longer the case. The solutions for one instance must be consistent with the solutions of the remaining instances—they must satisfy the Monotonicity Property.
- Monotone algorithms are holistic algorithms: they must consider the whole space of inputs together.

The main tool

Lemma

Let t be a set of tasks and let $x = x(t)$ be the allocation produced by a truthful mechanism. Suppose that we change only the tasks of machine i and in such a way that $t'_{ij} > t_{ij}$ when $x_{ij} = 0$, and $t'_{ij} < t_{ij}$ when $x_{ij} = 1$. The mechanism does not change the allocation to machine i, i.e., $x_i(t') = x_i(t)$. (However, it may change the allocation of other machines).

Example

$$t = \begin{pmatrix} \mathbf{1} & 2 & \mathbf{2} \\ 2 & \mathbf{3} & 1 \\ 1 & 2 & 2 \end{pmatrix}$$

The main tool

Lemma

Let t be a set of tasks and let $x = x(t)$ be the allocation produced by a truthful mechanism. Suppose that we change only the tasks of machine i and in such a way that $t'_{ij} > t_{ij}$ when $x_{ij} = 0$, and $t'_{ij} < t_{ij}$ when $x_{ij} = 1$. The mechanism does not change the allocation to machine i, i.e., $x_i(t') = x_i(t)$. (However, it may change the allocation of other machines).

Example

$$t = \begin{pmatrix} \mathbf{1} & 2 & \mathbf{2} \\ 2 & \mathbf{3} & 1 \\ 1 & 2 & 2 \end{pmatrix} \quad \rightarrow \quad t' = \begin{pmatrix} 1-\epsilon_1 & 2+\epsilon_2 & 2-\epsilon_3 \\ 2 & 3 & 1 \\ 1 & \mathbf{2} & 2 \end{pmatrix}$$

Proof of lower bound 2

$$\begin{pmatrix} \mathbf{1} & 1 & 1 \\ 1 & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

or

$$\begin{pmatrix} 1 & 1 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

Proof of lower bound 2

$$\begin{pmatrix} \color{red}{\mathbf{1}} & 1 & 1 \\ 1 & \color{red}{\mathbf{1}} & \color{red}{\mathbf{1}} \end{pmatrix} \quad \to \quad \begin{pmatrix} \color{red}{\mathbf{0}} & 1 & 1 \\ 1 & \color{red}{\mathbf{1}} & \color{red}{\mathbf{1}} \end{pmatrix}$$

or

$$\begin{pmatrix} 1 & 1 & 1 \\ \color{red}{\mathbf{1}} & \color{red}{\mathbf{1}} & \color{red}{\mathbf{1}} \end{pmatrix} \quad \to \quad \begin{pmatrix} 1 & 1 & 1 \\ \color{red}{\mathbf{0}} & \color{red}{\mathbf{1}} & \color{red}{\mathbf{1}} \end{pmatrix}$$

A geometric approach

Fix all values of t except t_{11} and t_{12}. Consider how the space of t_{11} and t_{12} is partitioned by a truthful mechanism.

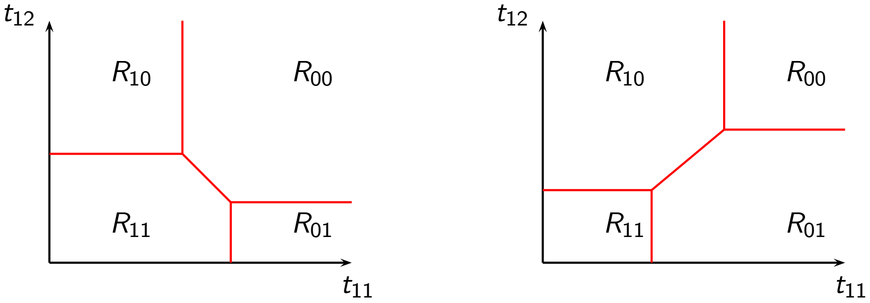
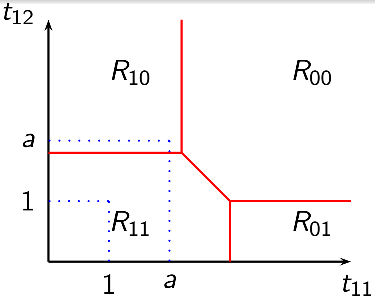


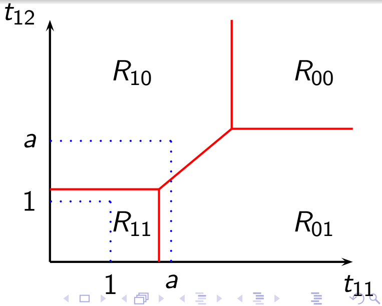
Figure: The two possible ways to partition the positive orthant.

Another useful lemma

Lemma

Fix all values of m tasks except of the values t_{1j} *and* t_{1k}*. Assume that a truthful mechanism assigns both tasks to machine 1 when* $(t_{1j}, t_{1k}) = (1, 0)$ *and when* $(t_{1j}, t_{1k}) = (0, 1)$*. Assume also that the mechanism assigns the first but not the second task to machine 1 when* $(t_{1j}, t_{1k}) = (a, a)$ *for some* $a > 1$*. Then the mechanism assigns both tasks to machine 1 when* $(t_{1j}, t_{1k}) = (1, 1)$*.*





The general idea of main proof

The proof of the main result is technical and complicated. The general idea is this:

- We start with the set of tasks

$$t = \begin{pmatrix} 1 & \infty & \infty & a & a \\ \infty & 1 & \infty & a & a \\ \infty & \infty & 1 & a & a \end{pmatrix}$$

  where $a > 1$ is a parameter (the optimal value is $a = \sqrt{2}$).
- This admits two distinct allocations (up to symmetry)
- We then show that in every case, the following tasks have the following allocation (otherwise the approximation ratio is high)

$$t = \begin{pmatrix} \mathbf{1} & \infty & \infty & \mathbf{1} & \mathbf{1} \\ \infty & \mathbf{1} & \infty & a & \infty \\ \infty & \infty & \mathbf{1} & \infty & a \end{pmatrix}$$

Proof (cont.)

$$\begin{pmatrix} \mathbf{1} & \infty & \infty & \mathbf{1} & \mathbf{1} \\ \infty & \mathbf{1} & \infty & a & \infty \\ \infty & \infty & \mathbf{1} & \infty & a \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{1} & \infty & \infty & \mathbf{1} & \mathbf{1} \\ \infty & \mathbf{1} & \infty & a & \infty \\ \infty & \infty & \mathbf{0} & \infty & a \end{pmatrix} \qquad \begin{pmatrix} \mathbf{1} & \infty & \infty & \mathbf{1} & \mathbf{1} \\ \infty & \mathbf{0} & \infty & a & \infty \\ \infty & \infty & \mathbf{1} & \infty & a \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{1} & \infty & \infty & \mathbf{1} & \mathbf{1} \\ \infty & \mathbf{0} & \infty & a & \infty \\ \infty & \infty & \mathbf{0} & \infty & a \end{pmatrix}$$

$$\downarrow$$

$$\begin{pmatrix} \mathbf{a} & \infty & \infty & \mathbf{1} & \mathbf{1} \\ \infty & \mathbf{0} & \infty & a & \infty \\ \infty & \infty & \mathbf{0} & \infty & a \end{pmatrix}$$

Open problems

- Improve the lower bound to n (or to the modest $\sqrt{n}$)
- Study randomized and fractional mechanisms
- Characterize the truthful mechanisms
- Based on a useful characterization of the case of 2 players and 2 tasks, we can show that even for this case the lower bound is 2.
- Consider the common generalization of the combinatorial auction and the scheduling problem. This is equivalent to the combinatorial auction problem with the max-min objective. The upper bound is still n for this generalization of the scheduling problem. Show a lower bound of n for this easier (?) case.

Thank you.
Time for dinner!