Introduction to algorithmic mechanism design

Elias Koutsoupias
Department of Computer Science
University of Oxford

EWSCS 2014

March 5-7, 2014

Part I

The scheduling problem

Scheduling unrelated machines

Definition

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

The setting

Input

$$t = \begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,m} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,m} \\ \cdots & & & \\ t_{n,1} & t_{n,2} & \cdots & t_{n,m} \end{pmatrix} \quad \rightarrow \quad$$

Output

$$a = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \cdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}$$

- n players/machines (rows).
- m tasks (columns).
- The input consists of nonnegative values $t_{i,j}$.
- The output is an allocation:

$$a_{i,j} = \begin{cases} 1 & \text{when task } j \text{ is allocated to machine } i \\ 0 & \text{otherwise} \end{cases}$$

Results on the classical scheduling problem

Computational issues

- It is a well-studied NP-hard problem
- Lenstra, Shmoys, and Tardos showed that its approximation ratio is in $[1.5, 2]$.
- One of the important open problems in the area of approximation algorithms

Part II

VCG for scheduling

VCG for scheduling

- VCG selects an allocation a which minimizes the sum of the completion times of all machines:

$$\sum_i \sum_j a_{i,j} t_{i,j}$$

- While the optimal makespan is

$$\max_i \sum_j a_{i,j} t_{i,j}$$

- VCG approximates the makespan within a factor of n (the number or players). Why? Because $\sum_i z_i \leq n \max_i z_i$ for every z_i.

Mechanisms for scheduling

- For some inputs this is the best-possible. For example, in this instance with $n = m = 3$

$$t = \begin{pmatrix} 1-\epsilon & 1-\epsilon & 1-\epsilon \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

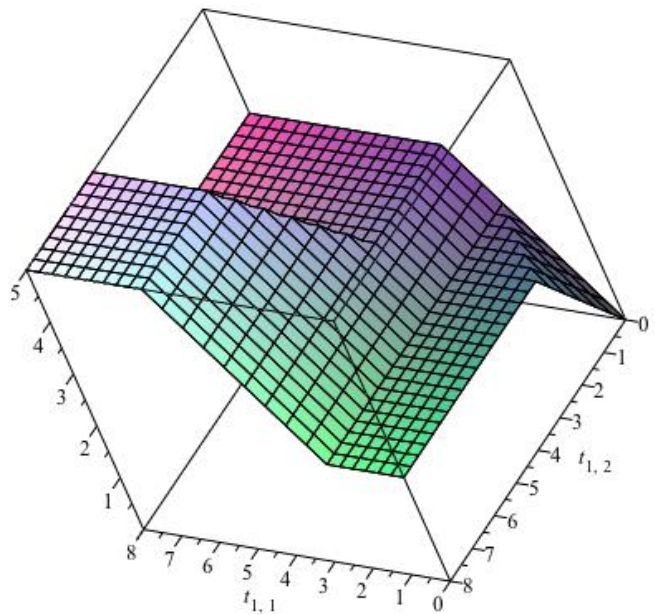
  VCG will allocate all tasks to the first machine.
- The approximation ratio of VCG for scheduling is exactly n
- Major open problem: Are there mechanisms with better approximation ratio?
- We will show that no mechanism can have approximation ratio less than 2. The best known lower bound is 2.618.

Mechanisms for scheduling

- BTW: The lower bound on the approximation ratio shows immediately that the makespan is not a concave (the negation of a convex) function
- Example:

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

- makespan $= \min(t_{1,1} + t_{1,2}, \max(t_{1,1}, 2), \max(t_{1,2}, 3), 5)$



Part III

Lower bounds for scheduling mechanisms

Truthful ≡ Monotone

We will not use convexity directly. We will use the following equivalent property:

Definition ((Weak) 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 a and a' satisfy

$$(a_i - a'_i) \cdot (t_i - t'_i) \leq 0,$$

where $\cdot$ denotes the dot product of the vectors, that is, $\sum_{j=1}^{m} (a_{i,j} - a'_{i,j})(t_{i,j} - t'_{i,j}) \leq 0$.

Theorem (Nisan, Ronen 1998, Saks, Lan Yu 2005)

$$\textit{Truthful} \equiv \textit{Weak Monotone}$$

The Monotonicity Property

First Input

$$\begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,m} \\ \cdots & & & \\ t_{i,1} & t_{i,2} & \cdots & t_{i,m} \\ \cdots & & & \\ t_{n,1} & t_{n,2} & \cdots & t_{n,m} \end{pmatrix} \Rightarrow \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ \cdots & & & \\ a_{i,1} & a_{i,2} & \cdots & a_{i,m} \\ \cdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}$$

Second Input

$$\begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,m} \\ \cdots & & & \\ t'_{i,1} & t'_{i,2} & \cdots & t'_{i,m} \\ \cdots & & & \\ t_{n,1} & t_{n,2} & \cdots & t_{n,m} \end{pmatrix} \Rightarrow \begin{pmatrix} a'_{1,1} & a'_{1,2} & \cdots & a'_{1,m} \\ \cdots & & & \\ a'_{i,1} & a'_{i,2} & \cdots & a'_{i,m} \\ \cdots & & & \\ a'_{n,1} & a'_{n,2} & \cdots & a'_{n,m} \end{pmatrix}$$

Monotonicity

$$\sum_{j=1}^{m} (a_{i,j} - a'_{i,j})(t_{i,j} - t'_{i,j}) \leq 0$$

How to use the Monotonicity Property

We manipulate the values of one player in a particular way which guarantees that his allocation remains the same.

Example (Change the values, keep the allocation)

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

How to use the Monotonicity Property

We manipulate the values of one player in a particular way which guarantees that his allocation remains the same.

Example (Change the values, keep the allocation)

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

How to use the Monotonicity Property

We manipulate the values of one player in a particular way which guarantees that his allocation remains the same.

Example (Change the values, keep the allocation)

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

Example (Increase a value, keep the allocation)

$$t = \begin{pmatrix} \mathbf{0} & \cdots \\ \infty & \cdots \\ \infty & \cdots \end{pmatrix}$$

How to use the Monotonicity Property

We manipulate the values of one player in a particular way which guarantees that his allocation remains the same.

Example (Change the values, keep the allocation)

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

Example (Increase a value, keep the allocation)

$$t = \begin{pmatrix} \mathbf{0} & \cdots \\ \infty & \cdots \\ \infty & \cdots \end{pmatrix} \quad \rightarrow \quad t' = \begin{pmatrix} \mathbf{1} & \cdots \\ \infty & \cdots \\ \infty & \cdots \end{pmatrix}$$

Easy proof of lower bound 2

Proposition

No mechanism can approximate the makespan better than a factor of 2.

2 players, 3 tasks

Either the mechanism partitions the tasks to the two machines

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

or gives all tasks to the same machine

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

Easy proof of lower bound 2

Proposition

No mechanism can approximate the makespan better than a factor of 2.

2 players, 3 tasks

Either the mechanism partitions the tasks to the two machines

$$\begin{pmatrix} \mathbf{1} & 1 & 1 \\ 1 & \mathbf{1} & \mathbf{1} \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} \mathbf{0} & 1 & 1 \\ 1 & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

or gives all tasks to the same machine

$$\begin{pmatrix} 1 & 1 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

The instances of the 2.618 lower bound

Theorem

No mechanism can approximate the makespan better than a factor of 2.618.

$$\begin{pmatrix} 0 & \cdots & \infty & z & z^2 & \cdots & z^{n-1} \\ \infty & \cdots & \infty & z^2 & z^3 & \cdots & z^n \\ \cdots & & & & & & \\ \infty & \cdots & 0 & z^n & z^{n+1} & \cdots & z^{2n-1} \end{pmatrix}$$

Claim

For every z, if the first player does not get all the non-dummy tasks (the z^j tasks), then the approximation ratio is at least $1 + z$.

Therefore the approximation ratio is

$$\min\left\{1 + z, \frac{z + z^2 + \cdots + z^{n-1}}{z^{n-1}}\right\}.$$

For $n \to \infty$ and $z = \phi$, the ratio is $2.618\ldots$.

Part IV

Related machines (single-parameter)

Related machines scheduling

- In this scheduling problem, machines differ only in their speeds (which are their private value)
- This is a typical **single-parameter domain**
- The problem is NP-hard, but there is a PTAS (it can be approximated to within any ϵ in polynomial time)
- Archer and Tardos gave a variant of the (exponential-time) optimal algorithm which is truthful
- Christodoulou and Kovács gave a deterministic truthful PTAS

Part V

Fractional scheduling

The fractional scheduling

Fractional allocations

- With fractional allocations each task can be split across the machines.
- The classical version of the problem is solvable in polynomial time (by linear programming).
- It is closely related to randomized mechanisms
  fractional approx ratio $\leq$ randomized approx ratio
- It is known that the approximation ratio of mechanisms is between 2 and $(n+1)/2$

Fractional Version: Upper Bound

The SQUARE Algorithm

The mechanism SQUARE is a task independent algorithm which allocates to every player i a fraction inversely proportional to $t_{i,j}^2$ of task j.

Theorem

The mechanism SQUARE is truthful with approximation ratio $\frac{n+1}{2}$.

Fractional Version: Lower Bound

A bad input

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \cdots & \infty & n-1 \\ \infty & 0 & \cdots & \infty & \cdots & \infty & n-1 \\ \cdots & & & & & & \\ \infty & \infty & \cdots & 0 & \cdots & \infty & n-1 \\ \cdots & & & & & & \\ \infty & \infty & \cdots & \infty & \cdots & 0 & n-1 \end{pmatrix}$$

Proving a lower bound of $2 - 1/n$

- Find the player who gets the largest fraction of the last task and raise its diagonal 0 value to 1.

Fractional Version: Lower Bound

A bad input

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \cdots & \infty & n-1 \\ \infty & 0 & \cdots & \infty & \cdots & \infty & n-1 \\ \cdots & & & & & & \\ \infty & \infty & \cdots & \mathbf{1} & \cdots & \infty & \mathbf{n-1} \\ \cdots & & & & & & \\ \infty & \infty & \cdots & \infty & \cdots & 0 & n-1 \end{pmatrix}$$

Proving a lower bound of $2 - 1/n$

- Find the player who gets the largest fraction of the last task and raise its diagonal 0 value to 1.
- When we change the values, the allocation remains almost the same.
- The optimal cost for the new input is 1.
- The cost of the changed player is at least $1 + \frac{n-1}{n} - \epsilon$.
- The approximation ratio is at least $2 - \frac{1}{n} - \epsilon$.

Part VI

Characterization of truthful mechanisms

Beyond VCG and affine maximizers?

Theorem (Roberts, 1979)

For unrestricted domains with 2 or more players and at least 3 outcomes, the only truthful deterministic mechanisms are the affine maximizers.

- Major open problem: extend Roberts theorem to combinatorial auctions and other domains.

Why payments?

Theorem (Gibbard-Satterthwaite)

The only truthful mechanisms without money are dictatorships

(In retrospect, a corollary of Roberts theorem.)

- In other words, no non-trivial problems have truthful mechanisms without payments.
- On the other hand, there are some very interesting truthful mechanisms without payments: For example the stable matching algorithm.

Part VII

Open problems

Some open problems

Scheduling unrelated machines

- Characterize the truthful mechanisms for scheduling unrelated machines.
- Close the gap between 2.618 and n of the deterministic approximation ratio
- Similarly for the gap between 2 and $\Theta(n)$ for randomized an fractional mechanisms.

Combinatorial auctions

- Characterize the truthful mechanisms for more general settings such as the combinatorial auction problem