Lower Bounds of Mechanisms for Scheduling Unrelated Machines

Elias Koutsoupias

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

Warwick, March 2007

Joint work with:

George Christodoulou
Annamária Kovács
Angelina Vidali

Scheduling unrelated machines

The scheduling problem for unrelated machines

- There are n players (machines) and m 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 tasks (allocation algorithm)
- The mechanism pays the players based on the declared values and the allocation (payment algorithm)
- The objective of each player is to minimize his execution time minus his payment.

Input and Output

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

n machines

m tasks

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.

The Monotonicity Property

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, $\sum_{j=1}^{m}(x_{ij} - x_{ij}')(t_{ij} - t_{ij}') \leq 0$.

$$\begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ \cdots & & & \\ t_{i1} & t_{i2} & \cdots & t_{im} \\ \cdots & & & \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{pmatrix} \Rightarrow \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ \cdots & & & \\ x_{i1} & x_{i2} & \cdots & x_{im} \\ \cdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}$$

The Monotonicity Property

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, $\sum_{j=1}^{m}(x_{ij} - x'_{ij})(t_{ij} - t'_{ij}) \leq 0$.

$$\begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ \cdots & & & \\ t'_{i1} & t'_{i2} & \cdots & t'_{im} \\ \cdots & & & \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{pmatrix} \Rightarrow \begin{pmatrix} x'_{11} & x'_{12} & \cdots & x'_{1m} \\ \cdots & & & \\ x'_{i1} & x'_{i2} & \cdots & x'_{im} \\ \cdots & & & \\ x'_{n1} & x'_{n2} & \cdots & x'_{nm} \end{pmatrix}$$

Truthful = Monotone

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).

The Monotonicity Property characterizes truthful mechanisms without any reference to payments.

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.

Major Open Problem

Open Problem

What is the best approximation ratio of monotone algorithms?

Conjecture (Nisan, Ronen 1998)

The best approximation ratio of monotone algorithms is n.

- This is conjectured to be true even for exponential time algorithms.

History of scheduling unrelated machines

- It is a well-studied NP-hard problem. Lenstra, Shmoys, and Tardos showed that its approximation ratio is between $3/2$ and 2.
- 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 showed a lower bound of 2.
  - 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.
- Kovács improved it to 3 and eventually to 2.8.

Some Recent results

- Christodoulou, Koutsoupias, and Vidali improved the lower bound from 2 to 2.41 (SODA 2007). This was further improved by Koutsoupias and Vidali to 2.61 (unpublished).
- Mu'alem and Schapira showed new randomized bounds between $2 - 1/n$ and $7/8\,n$ (SODA 2007).
- Christodoulou, Koutsoupias, and Kovacs studied the fractional version of the problem and showed that the approximation ratio is between $2 - 1/n$ and $(n + 1)/2$ (unpublished).
- Lavi and Swami considered the special case where the tasks can take only two values (low and high). They showed that the approximation ratio is between 1.14 and 2 (EC 2007).

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

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

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

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

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

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

Example

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

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

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

Example

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

The instances of the 2.61 lower bound

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

Claim

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

Therefore the approximation ratio is

$$\min\{1+a, \frac{a+a^2+\cdots+a^{n-1}}{a^{n-1}}\}.$$

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

The Proof of the Claim

- We prove the claim by induction. For this we need to strengthen the induction hypothesis. The claim holds for all instances of the form

$$\begin{pmatrix} 0 & \cdots & \infty & a^{i_1} & a^{i_2} & \cdots & a^{i_k} \\ \infty & \cdots & \infty & a^{i_1+1} & a^{i_2+1} & \cdots & a^{i_k+1} \\ \cdots & & & & & & \\ \infty & \cdots & 0 & a^{i_1+n-1} & a^{i_2+n} & \cdots & a^{i_k+n-1} \end{pmatrix}$$

- $k \in \{1, \ldots, n-1\}$ and $i_1 < i_2 < \cdots < i_k$.

The Proof of the Claim (cont.)

- Assume that the first player does not get all the non-dummy tasks.
- We first manipulate the values so that the first player gets no non-zero task and every other player gets at most one non-zero task.

Example

$$\begin{pmatrix} 0 & \cdots & \infty & a^{i_1} & a^{i_2} & \cdots & a^{i_k} \\ \infty & \cdots & \infty & a^{i_1+1} & a^{i_2+1} & \cdots & a^{i_k+1} \\ \cdots & & & & & & \\ \infty & \cdots & 0 & a^{i_1+n-1} & a^{i_2+n} & \cdots & a^{i_k+n-1} \end{pmatrix}$$

$$\begin{pmatrix} 0 & \cdots & \infty & a^{i_1} & a^{i_2} & \cdots & \mathbf{0} \\ \infty & \cdots & \infty & \mathbf{0} & a^{i_2+1} & \cdots & a^{i_k+1} \\ \cdots & & & & & & \\ \infty & \cdots & 0 & a^{i_1+n-1} & a^{i_2+n} & \cdots & a^{i_k+n-1} \end{pmatrix}$$

The Proof of the Claim (cont.)

- The optimum is a^{i_k}.
- We find a task with cost at least a^{i_k+1} and we raise its dummy (diagonal) value to a^{i_k}.
- The heart of the proof is that there always exists such a task which will not raise the optimum value.
- The cost of the mechanism is at least $a^{i_k} + a^{i_k+1}$ while the optimum is a^{i_k}. The approximation ratio is at least $1 + a$.

Example

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

The Proof of the Claim (cont.)

- The optimum is a^{i_k}.
- We find a task with cost at least a^{i_k+1} and we raise its dummy (diagonal) value to a^{i_k}.
- The heart of the proof is that there always exists such a task which will not raise the optimum value.
- The cost of the mechanism is at least $a^{i_k} + a^{i_k+1}$ while the optimum is a^{i_k}. The approximation ratio is at least $1 + a$.

Example

$$\begin{pmatrix} 0 & \infty & \infty & \cdots & a^{i_k-3} & a^{i_k-1} & a^{i_k} \\ \infty & 0 & \infty & \cdots & a^{i_k-2} & a^{i_k} & a^{i_k+1} \\ \infty & \infty & \color{red}{a^{i_k}} & \cdots & a^{i_k-1} & a^{i_k+1} & \color{red}{a^{i_k+2}} \\ \cdots \end{pmatrix}$$

The Fractional Version of the Problem

- In the fractional version each task can be split across the machines.
- The classical version is solvable in polynomial time.
- fractional approximation ratio $\leq$ randomized approximation ratio

Fractional Version: Lower Bound

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

- We change the value of the player with the highest allocation.
- When we change the values, the allocation remains almost the same.
- The optimal cost for the new values is 1.
- The cost of the changed player is at least $1 + \frac{n-1}{n} - \epsilon$.
- The approximation ratio is at least $2 - \frac{1}{n} - \epsilon$.

Fractional Version: Lower Bound

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

- We change the value of the player with the highest allocation.
- When we change the values, the allocation remains almost the same.
- The optimal cost for the new values is 1.
- The cost of the changed player is at least $1 + \frac{n-1}{n} - \epsilon$.
- The approximation ratio is at least $2 - \frac{1}{n} - \epsilon$.

Fractional Version: Upper Bound

The mechanism SQUARE allocates to every player i a fraction inversely proportional to t_{ij}^2 of task j.

Theorem

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

Open Problems

- The major open problem is to bridge the gap between the lower bound of 2.61 and the upper bound of n (and the same problem for the fractional mechanisms).
- How far can these techniques go?
- Most likely, not very far.
- What is needed is to find a useful characterization of monotone algorithms.

Open Problems

- There are essentially two known types of mechanisms:

  threshold They assign task j to player i iff $t_{ij} \leq f_{ij}(t_{-i})$.

  VCG It selects the allocation which minimizes the (weighted) sum of the cost of all players. More precisely, it selects the allocation x which minimizes

$$\sum_i \alpha_i t_i x_i + \gamma_x$$

  for some constants α_i and γ_x.

- Are there other types of truthful mechanisms?

Conjecture

The only truthful mechanisms are the ones which allocate some tasks with the threshold policy and the remaining tasks with the VCG policy.

Thank you!