

A $1 + \phi$ lower bound for truthful scheduling mechanisms

MFCS '07

Angelina Vidali

Department of Informatics and Telecommunications

University of Athens

<http://users.uoa.gr/~avidali>

29, August 2007

Joint work with:

Elias Koutsoupias

Scheduling unrelated machines

The matrix of processing times

We want to process m tasks using n machines (/selfish players).

We have the following matrix of processing times:

$$\begin{array}{cccc} & \text{task 1} & \dots & \text{task } j & \dots & \text{task } m \\ \text{player 1} & t_{11} & \dots & t_{1j} & \dots & t_{1m} \\ \vdots & & & & & \\ \text{player } i & \text{needs time } t_{i1} & & \text{to process } t_{ij} & & \\ \vdots & & & & \ddots & \\ \text{player } n & t_{n1} & & & & t_{nm} \end{array}$$

Scheduling unrelated machines

The matrix of processing times

We want to process m tasks using n machines (/selfish players).

We have the following matrix of processing times:

$$\begin{array}{cccc} & \text{task 1} & \dots & \text{task } m \\ \text{player 1} & t_{11} & \dots & t_{1m} \\ \vdots & & & \\ \text{player } i & t_{i1} & \dots & t_{im} \\ \vdots & & & \\ \text{player } n & t_{n1} & \dots & t_{nm} \end{array}$$

to process task j

player i needs time t_{ij}

Only player i knows the values of his line. He can report a false value!

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	

Scheduling unrelated machines

Protocol

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

The objective of each player: **utility maximization**

maximize{payment – processing time}

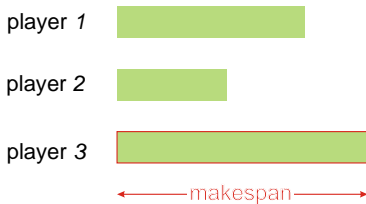
time=money...

Scheduling unrelated machines

MinMax objective of the mechanism designer

Finish with all tasks as soon as possible! i.e., **minimize the makespan**

- Tasks allocated to the **same** player are processed **in series**.
- Tasks allocated to **different** players are processed **in parallel**.



We can concentrate on truthful mechanisms

Our players are potential liars.

Definition (Truthful mechanisms)

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

that is: *"Players have nothing to gain by lying."*

Theorem (The revelation principle)

For every mechanism M there is an equivalent truthful one M' .

that is: "In M' all players achieve the same utility as in M , and without having to lie."

The Monotonicity Property

Definition (Monotonicity Property)

An allocation algorithm is monotone if for every two inputs t and t' which differ only on machine i (i.e., on the i -th 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 monotone if for every two inputs t and t' which differ only on machine i (i.e., on the i -th 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,
$$\sum_{j=1}^m (x_{ij} - x'_{ij})(t_{ij} - t'_{ij}) \leq 0.$$

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

Truthful = Monotone

Theorem (Nisan, Ronen 1998)

Every truthful mechanism satisfies the Monotonicity Property.

Theorem (Saks, Lan Yu 2005)

Every monotone allocation algorithm is truthful.

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

Monotone algorithms

- Monotonicity, which is not specific to the scheduling task problem but 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.
- 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.

Recent results

- Christodoulou, Koutsoupias, and Vidali improved the lower bound from 2 to 2.41 (SODA 2007).
- 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 $(n + 1)/2$ (ICALP 2007).
- 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

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

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

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

Example

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

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

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

Example

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

The kernel of the $1 + \phi$ lower bound

- If *one player gets all tasks*, $a > 1$

$$\frac{\text{cost}}{\text{OPT}} \geq \frac{\begin{pmatrix} 1 & a \\ a & a^2 \end{pmatrix}}{\begin{pmatrix} 1 & a \\ a & a^2 \end{pmatrix}} = \frac{1 + a}{a}$$

The kernel of the $1 + \phi$ lower bound

- If one player gets all tasks, $a > 1$

$$\frac{\text{cost}}{\text{OPT}} \geq \frac{\begin{pmatrix} 1 & a \\ a & a^2 \end{pmatrix}}{\begin{pmatrix} 1 & a \\ a & a^2 \end{pmatrix}} = \frac{1 + a}{a}$$

- Choosing a *diagonal allocation* mars things in a single-task case (Monotonicity requires the same allocation in both instances)

$$\frac{\text{cost}}{\text{OPT}} = \frac{\begin{pmatrix} 1 & 0 \\ a & a^2 \end{pmatrix}}{\begin{pmatrix} 1 & 0 \\ a & a^2 \end{pmatrix}} = \frac{a}{1}$$

The kernel of the $1 + \phi$ lower bound

- If one player gets all tasks, $a > 1$

$$\frac{\text{cost}}{\text{OPT}} \geq \frac{\begin{pmatrix} 1 & a \\ a & a^2 \end{pmatrix}}{\begin{pmatrix} 1 & a \\ a & a^2 \end{pmatrix}} = \frac{1+a}{a}$$

- Choosing a *diagonal allocation* mars things in a single-task case (Monotonicity requires the same allocation in both instances)

$$\frac{\text{cost}}{\text{OPT}} = \frac{\begin{pmatrix} 1 & 0 \\ a & a^2 \end{pmatrix}}{\begin{pmatrix} 1 & 0 \\ a & a^2 \end{pmatrix}} = \frac{a}{1}$$

If a is such that $\frac{1+a}{a} = a$ whichever is the allocation the mechanism gives the approximation ratio is at least $\phi \approx 1.618$.

The instances of the 2.61 lower bound

$$\begin{pmatrix} \mathbf{0} & \infty & \cdots & \infty & \infty & \mathbf{1} & \mathbf{a} & \cdots & \mathbf{a^{n-2}} \\ \infty & 0 & \cdots & \infty & \infty & a & a^2 & \cdots & a^{n-1} \\ & & \ddots & & & & & & \\ \infty & \infty & \cdots & 0 & \infty & a^{n-2} & a^{n-1} & \cdots & a^{2n-4} \\ \infty & \infty & \cdots & \infty & 0 & a^{n-1} & a^n & \cdots & a^{2n-3} \end{pmatrix}$$

If the **first player gets all tasks**

$$\frac{\text{cost}}{\text{OPT}} = \frac{1 + a + a^2 + \cdots + a^{n-2} + a^{n-1}}{a^{n-1}}.$$

The instances of the 2.61 lower bound

$$\begin{pmatrix} a^{n-1} & \infty & \cdots & \infty & \infty & 1 & a & \cdots & a^{n-2} \\ \infty & 0 & \cdots & \infty & \infty & a & a^2 & \cdots & a^{n-1} \\ & & \ddots & & & & & & \\ \infty & \infty & \cdots & 0 & \infty & a^{n-2} & a^{n-1} & \cdots & a^{2n-4} \\ \infty & \infty & \cdots & \infty & 0 & a^{n-1} & a^n & \cdots & a^{2n-3} \end{pmatrix}$$

If the **first player** gets all tasks

$$\frac{\text{cost}}{\text{OPT}} = \frac{1 + a + a^2 + \cdots + a^{n-2} + a^{n-1}}{a^{n-1}}.$$

The instances of the 2.61 lower bound

$$\begin{pmatrix} a^{n-1} & \infty & \cdots & \infty & \infty & 1 & a & \cdots & a^{n-2} \\ \infty & 0 & \cdots & \infty & \infty & a & a^2 & \cdots & a^{n-1} \\ & & \ddots & & & & & & \\ \infty & \infty & \cdots & 0 & \infty & a^{n-2} & a^{n-1} & \cdots & a^{2n-4} \\ \infty & \infty & \cdots & \infty & 0 & a^{n-1} & a^n & \cdots & a^{2n-3} \end{pmatrix}$$

If the **first player** gets all tasks

$$\frac{\text{cost}}{\text{OPT}} = \frac{1 + a + a^2 + \cdots + a^{n-2} + a^{n-1}}{a^{n-1}}.$$

The instances of the 2.61 lower bound

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \infty & 1 & a & \cdots & a^{n-2} \\ \infty & 0 & \cdots & \infty & \infty & a & a^2 & \cdots & a^{n-1} \\ & & \ddots & \ddots & & & & & \\ \infty & \infty & \cdots & 0 & \infty & a^{n-2} & a^{n-1} & \cdots & a^{2n-4} \\ \infty & \infty & \cdots & \infty & 0 & a^{n-1} & a^n & \cdots & a^{2n-3} \end{pmatrix}$$

Claim

If at least one task goes to another player, we will prove that

$$\frac{\text{cost}}{\text{OPT}} \geq 1 + a.$$

The instances of the 2.61 lower bound

If the **first player gets all tasks**

$$\frac{\text{cost}}{\text{OPT}} = \frac{1 + a + a^2 + \dots + a^{n-2} + a^{n-1}}{a^{n-1}}.$$

Claim

If at least one task goes to another player, we will prove that

$$\frac{\text{cost}}{\text{OPT}} \geq 1 + a.$$

Therefore

$$\frac{\text{cost}}{\text{OPT}} \geq \min\left\{1 + a, \frac{1 + a + a^2 + \dots + a^{n-1}}{a^{n-1}}\right\}.$$

For $n \rightarrow \infty$ and $a = \phi$, the ratio is $1 + \phi \approx 2.618\dots$

The Proof of the Claim is by induction

We need to **strengthen the induction hypothesis**

The claim also holds if we eliminate columns from the original instance

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \infty & 1 & a & a^2 & \cdots & a^{n-2} \\ \infty & 0 & \cdots & \infty & \infty & a & a^2 & a^3 & \cdots & a^{n-1} \\ & & \ddots & & & & & & & \\ \infty & \infty & \cdots & 0 & \infty & a^{n-2} & a^{n-1} & a^n & \cdots & a^{2n-4} \\ \infty & \infty & \cdots & \infty & 0 & a^{n-1} & a^n & a^{n+1} & \cdots & a^{2n-3} \end{pmatrix}$$

If a column has a **0**-valued task eliminate it!

The Proof of the Claim is by induction

We need to **strengthen the induction hypothesis**

The claim also holds if we eliminate columns from the original instance

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \infty & 1 & a & a^2 & \cdots & a^{n-2} \\ \infty & 0 & \cdots & \infty & \infty & a & a^2 & a^3 & \cdots & a^{n-1} \\ & & \ddots & & & & & & & \\ \infty & \infty & \cdots & 0 & \infty & a^{n-2} & a^{n-1} & a^n & \cdots & a^{2n-4} \\ \infty & \infty & \cdots & \infty & 0 & a^{n-1} & a^n & a^{n+1} & \cdots & a^{2n-3} \end{pmatrix}$$

If a column has a **0**-valued task eliminate it!

The Proof of the Claim is by induction

We need to **strengthen the induction hypothesis**

The claim also holds if we eliminate columns from the original instance

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \infty & 1 & \mathbf{0} & a^2 & \cdots & a^{n-2} \\ \infty & 0 & \cdots & \infty & \infty & a & a^2 & a^3 & \cdots & a^{n-1} \\ & & \ddots & & & & & & & \\ \infty & \infty & \cdots & 0 & \infty & a^{n-2} & a^{n-1} & a^n & \cdots & a^{2n-4} \\ \infty & \infty & \cdots & \infty & 0 & a^{n-1} & a^n & a^{n+1} & \cdots & a^{2n-3} \end{pmatrix}$$

If a column has a **0**-valued task eliminate it!

Remark

Gaps are generated by setting a processing time to **0**: we suppose that the algorithm is clever enough not to lose the chance to process the task in zero time as $\frac{a}{0} = \infty \dots$

Strengthening the induction hypothesis, formally now

We prove the claim for all instances of the form

$$\begin{pmatrix} 0 & \infty & \dots & \infty & a^{i_1} & a^{i_2} & \dots & a^{i_k} \\ \infty & 0 & \dots & \infty & a^{i_1+1} & a^{i_2+1} & \dots & a^{i_k+1} \\ \vdots & & \ddots & & \vdots & & \ddots & \vdots \\ \infty & \infty & \dots & 0 & a^{i_1+n-1} & a^{i_2+n-1} & \dots & a^{i_k+n-1} \end{pmatrix},$$

where $k \in \mathbb{N}$, $1 \leq k \leq n - 2$ and $i_1 < i_2 < \dots < i_k$.

The exponents **might not be successive** numbers anymore.

Example

Before we had only $(i_1, i_2, i_3, \dots, i_k) = (1, 2, 3, \dots, n - 3, n - 2)$.

Now we can also have $(i_1, i_2, i_3, \dots, i_k) = (1, 3, \dots, n - 3)$.

The Proof of the Claim – Preprocessing

Repeat until: player 1 gets nothing and the others at most one task

- Set all tasks assigned to the first player to zero
- Leave only one task to players $2, \dots, n$

Invariant after each step (by monotonicity):

One of the players $2, \dots, n$ gets a (non-zero) task.

Example

$$\begin{pmatrix} 0 & \infty & \dots & 1 & \mathbf{a} & a^2 & \mathbf{a^3} & a^4 \\ \infty & 0 & \dots & \mathbf{a} & a^2 & a^3 & a^4 & a^5 \\ \infty & \infty & \dots & a^2 & a^3 & \mathbf{a^4} & a^5 & a^6 \\ \infty & \infty & \dots & a^3 & a^4 & a^5 & a^6 & \mathbf{a^7} \\ \infty & \infty & \dots & a^4 & a^5 & a^6 & a^7 & a^8 \end{pmatrix}$$

The Proof of the Claim – Preprocessing

Repeat until: player 1 gets nothing and the others at most one task

- Set all tasks assigned to the first player to zero
- Leave only one task to players $2, \dots, n$

Invariant after each step (by monotonicity):

One of the players $2, \dots, n$ gets a (non-zero) task.

Example

$$\begin{pmatrix} 0 & \infty & \dots & 1 & 0 & a^2 & 0 & a^4 \\ \infty & 0 & \dots & \mathbf{a} & a^2 & \mathbf{a^3} & a^4 & a^5 \\ \infty & \infty & \dots & a^2 & a^3 & a^4 & a^5 & a^6 \\ \infty & \infty & \dots & a^3 & a^4 & a^5 & a^6 & \mathbf{a^7} \\ \infty & \infty & \dots & a^4 & a^5 & a^6 & a^7 & a^8 \end{pmatrix}$$

The Proof of the Claim – Preprocessing

Repeat until: player 1 gets nothing and the others at most one task

- Set all tasks assigned to the first player to zero
- Leave only one task to players $2, \dots, n$

Invariant after each step (by monotonicity):

One of the players $2, \dots, n$ gets a (non-zero) task.

Example

$$\begin{pmatrix} 0 & \infty & \dots & 1 & \mathbf{0} & a^2 & \mathbf{0} & a^4 \\ \infty & 0 & \dots & \mathbf{0} & a^2 & \mathbf{a^3} & a^4 & a^5 \\ \infty & \infty & \dots & a^2 & a^3 & a^4 & a^5 & a^6 \\ \infty & \infty & \dots & a^3 & a^4 & a^5 & a^6 & a^7 \\ \infty & \infty & \dots & a^4 & a^5 & a^6 & a^7 & \mathbf{a^8} \end{pmatrix}$$

By eliminating a column we get new optima

For i_1, \dots, i_k

- **successive** natural numbers: the allocation achieving **OPT** is **unique**.

$$\left(\begin{array}{cccccc|cccccc} 0 & \infty & \infty & \infty & \infty & \infty & 1 & a & a^2 & a^3 & a^4 \\ \infty & 0 & \infty & \infty & \infty & \infty & a & a^2 & a^3 & a^4 & a^5 \\ \infty & \infty & 0 & \infty & \infty & \infty & a^2 & a^3 & a^4 & a^5 & a^6 \\ \infty & \infty & \infty & 0 & \infty & \infty & a^3 & a^4 & a^5 & a^6 & a^7 \\ \infty & \infty & \infty & \infty & 0 & \infty & a^4 & a^5 & a^6 & a^7 & a^8 \\ \infty & \infty & \infty & \infty & \infty & 0 & a^5 & a^6 & a^7 & a^8 & a^9 \end{array} \right)$$

By eliminating a column we get new optima

For i_1, \dots, i_k

- **successive** natural numbers: the allocation achieving **OPT** is **unique**.

$$\left(\begin{array}{cccccc} 0 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 \end{array} \middle| \begin{array}{ccccc} 1 & a & 0 & a^3 & a^4 \\ a & a^2 & a^3 & a^4 & a^5 \\ a^2 & a^3 & a^4 & a^5 & a^6 \\ a^3 & a^4 & a^5 & a^6 & a^7 \\ a^4 & a^5 & a^6 & a^7 & a^8 \\ a^5 & a^6 & a^7 & a^8 & a^9 \end{array} \right)$$

- having a **gap**: more than one optimal assignments. **Exploit it!**

By eliminating a column we get new optima

For i_1, \dots, i_k

- **successive** natural numbers: the allocation achieving **OPT** is **unique**.

$$\left(\begin{array}{cccccc} 0 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 \end{array} \mid \begin{array}{ccccc} 1 & a & 0 & a^3 & a^4 \\ a & a^2 & a^3 & a^4 & a^5 \\ a^2 & a^3 & a^4 & a^5 & a^6 \\ a^3 & a^4 & a^5 & a^6 & a^7 \\ a^4 & a^5 & a^6 & a^7 & a^8 \\ a^5 & a^6 & a^7 & a^8 & a^9 \end{array} \right)$$

- having a **gap**: more than one optimal assignments. **Exploit it!**

By eliminating a column we get new optima

For i_1, \dots, i_k

- **successive** natural numbers: the allocation achieving **OPT** is **unique**.

$$\left(\begin{array}{cccccc} 0 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 \end{array} \right) \left| \begin{array}{ccccc} 1 & a & 0 & \begin{matrix} \xrightarrow{q} \\ a^3 & a^4 \\ \downarrow q \\ a^4 & a^5 \end{matrix} \\ a & a^2 & a^3 & a^5 & a^6 \\ a^2 & a^3 & a^4 & a^6 & a^7 \\ a^3 & a^4 & a^5 & a^7 & a^8 \\ a^4 & a^5 & a^6 & a^8 & a^9 \end{array} \right)$$

- having a **gap**: more than one optimal assignments. **Exploit it!**

Suppose there are q tasks after the last **gap**. The assignment for the players in the $q \times q$ **block** is the same for all optima.

The proof of the claim – the crux

Can a player raise his dummy to a^{i_k} without affecting OPT?

- If the player is in the **block** the processing time of the optimal allocation changes.

Example

$$\begin{array}{l} \text{player 1} \rightarrow \\ \text{player 2} \rightarrow \\ \text{OPT} = a^{i_k} \end{array} \left(\begin{array}{ccccccc} 0 & \infty & \infty & \dots & a^{i_k-3} & 0 & a^{i_k-1} & a^{i_k} \\ \infty & 0 & \infty & \dots & a^{i_k-2} & & a^{i_k} & a^{i_k+1} \\ \infty & \infty & 0 & \dots & a^{i_k-1} & & a^{i_k+1} & a^{i_k+2} \\ \infty & \infty & \infty & \dots & a^{i_k} & & a^{i_k+1} & a^{i_k+3} \\ \dots & & & & & & & \end{array} \right)$$

The proof of the claim – the crux

Can a player raise his dummy to a^{i_k} without affecting OPT?

- If the player is in the **block** the processing time of the optimal allocation changes.

Example

$$\begin{array}{l} \text{player 2} \rightarrow \\ \text{OPT}' > a^{i_k} \end{array} \left(\begin{array}{cccccc} 0 & \infty & \infty & \dots & a^{i_k-3} & 0 & a^{i_k-1} & a^{i_k} \\ \infty & \mathbf{a^{i_k}} & \infty & \dots & a^{i_k-2} & & a^{i_k} & a^{i_k+1} \\ \infty & \infty & 0 & \dots & a^{i_k-1} & & a^{i_k+1} & a^{i_k+2} \\ \infty & \infty & \infty & \dots & a^{i_k} & & a^{i_k+1} & a^{i_k+3} \\ \dots & & & & & & & \end{array} \right)$$

The proof of the claim – the crux

Can a player raise his dummy to a^{i_k} without affecting OPT?

- If the player is outside the **block** the processing time of the optimal allocation does not increase!

Example

$$\begin{array}{l} \text{OPT}' = \text{OPT} \\ \text{player 3} \rightarrow \end{array} \begin{pmatrix} 0 & \infty & \infty & \dots & a^{i_k-3} & 0 & a^{i_k-1} & a^{i_k} \\ \infty & 0 & \infty & \dots & a^{i_k-2} & a^{i_k} & a^{i_k+1} & \\ \infty & \infty & a^{i_k} & \dots & a^{i_k-1} & a^{i_k+1} & a^{i_k+2} & \\ \infty & \infty & \infty & \dots & a^{i_k} & a^{i_k+1} & a^{i_k+3} & \\ \dots & & & & & & & \end{pmatrix}$$

The Proof of the Claim – the crux

- Find a player outside this block who gets a task $\geq a \cdot OPT$.

Example

$$\begin{array}{l} \text{OPT} = a^{i_k} \\ \text{player 3} \rightarrow \end{array} \begin{pmatrix} 0 & \infty & \infty & \dots & a^{i_k-3} & 0 & a^{i_k-1} & a^{i_k} \\ \infty & 0 & \infty & \dots & a^{i_k-2} & a^{i_k} & a^{i_k+1} & \\ \infty & \infty & 0 & \dots & a^{i_k-1} & a^{i_k+1} & a^{i_k+2} & \\ \infty & \infty & \infty & \dots & a^{i_k} & a^{i_k+1} & a^{i_k+3} & \\ \dots & & & & & & & \end{pmatrix}$$

The Proof of the Claim – the crux

- Find a player outside this block who gets a task $\geq a \cdot OPT$.
- Raise his dummy value to a^{i_k} . OPT doesn't change!

$$\frac{\text{cost}}{OPT} \geq \frac{a^{i_k} + a^{i_k+1}}{a^{i_k}} = 1 + a.$$

Example

$$\text{player 3} \rightarrow \begin{pmatrix} 0 & \infty & \infty & \dots & a^{i_k-3} & 0 & a^{i_k-1} & a^{i_k} \\ \infty & 0 & \infty & \dots & a^{i_k-2} & & a^{i_k} & a^{i_k+1} \\ \infty & \infty & a^{i_k} & \dots & a^{i_k-1} & & a^{i_k+1} & a^{i_k+2} \\ \infty & \infty & \infty & \dots & a^{i_k} & & a^{i_k+1} & a^{i_k+3} \\ \dots & & & & & & & \end{pmatrix}$$

A player outside the block has assignment $\geq a^{i_{k+1}}$

Proof.

- One of the last q tasks is assigned to a player outside the block:
 - ▶ The block has size $q \times q$.
 - ▶ Player 1 gets nothing.
 - ▶ Each player gets at most one task.
 - ▶ We can't assign q tasks, to $q - 1$ players (pigeonhole principle).



□

A player outside the block has assignment $\geq a^{i_k+1}$

Proof.

- One of the last q tasks is assigned to a player outside the block:
 - ▶ The block has size $q \times q$.
 - ▶ Player 1 gets nothing.
 - ▶ Each player gets at most one task.
 - ▶ We can't assign q tasks, to $q - 1$ players (pigeonhole principle).
- Any player outside the block has value $\geq a^{i_k+1}$ for the last q tasks!
 - ▶ Because the last q exponents i_{k-q+1}, \dots, i_k are successive natural numbers.



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.