CIRM, May 06

Algorithmic Game Theory and Scheduling

Eric Angel, Evripidis Bampis, Fanny Pascual

IBISC, University of Evry, France

Outline

- Scheduling vs. Game Theory
- Stability, Nash Equilibrium
- Price of Anarchy
- Coordination Mechanisms
- Truthfulness

Scheduling

(A set of tasks) + (a set of machines)

(an objective function)

Aim: Find a feasible schedule optimizing the objective function.

Game Theory

(A set of agents) + (a set of strategies)

(an individual obj. function for every agent)

Aim: Stability, i.e. a situation where no agent has incentive to unilaterally change strategy.

Central notion: Nash Equilibrium (pure or mixed)

Game Theory (2)

Nash: For any finite game, there is always a (mixed) Nash Equilibrium.

Open problem: Is it possible to compute a Nash Equilibrium in polynomial time, even for the case of games with only two agents ?

Scheduling & Game Theory

The KP model:

(Agents: tasks) + (Ind. Obj. F. of agent i: the completion time of the machine on which task i is executed)

The CKN model:

(Agents: tasks) + (Ind. Obj. F. of agent i: the completion time of task i)

Scheduling & Game Theory (2)

The AT model:

(Agents: uniform machines) + (Ind. Obj. F. of agent i: the profit defined as $P_i - w_i/s_i$)

P_i: payment given to i

w_i: load of machine i

s_i: the speed of machine i

The Price of Anarchy (PA)

Aim: Evaluate the loss due to the absence of coordination.

[Koutsoupias, Papadimitriou: STACS'99]

Need of a Global Objective Function (GOF)

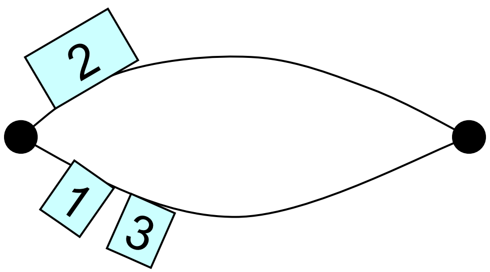
PA=(The value of the GOF in the worst NE)/(OPT)

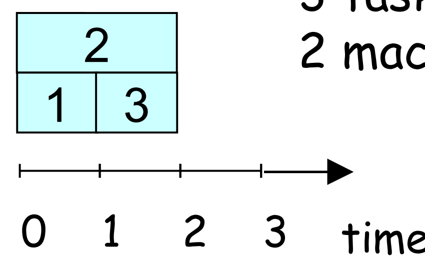
It measures the impact of the absence of coordination

[In what follows, GOF: makespan]

An example: KP model

[Koutsoupias, Papadimitriou: STACS'99]





3 tasks
2 machines

A (pure) Nash Equilibrium

Question:

How bad can be a Nash Equilibrium ?

An example: KP model

p_i^j : the probability of task i to go on machine j

The expected cost of agent i, if it decides to go on machine j with $p_i^j = 1$:

$$C_i^j = l_i + \sum_{k \neq i} p_k^j \, l_k$$

In a NE, agent i assigns non zero probabilities only to the machines that minimize C_i^j

An example

Instance: 2 tasks of length 1, 2 machines.

A NE: $p_i^j = 1/2$ for i=1,2 and j=1,2

$$C_1^1 = 1 + 1/2*1 = 3/2$$

$$C_1^2 = C_2^1 = C_2^2 = 3/2$$

Expected makespan

$$1/4*2+1/4*2+1/4*1+1/4*1 = 3/2$$

OPT = 1

The PA for the KP model

Thm [KP99]: The PA is (at least and at most) 3/2 for the KP model with two machines.

Thm [CV02]: The PA is $\Theta(\log m/(\log\log\log m))$ for the KP model with m uniform machines.

Pure NE for the KP model

Thm [FKKMS02]: There is always a pure NE for the KP model.

Thm [CV02]: The PA is $O(\log m/(\log\log m))$ for the KP model with m identical machines.

[O(log s_{max}/s_{min} for uniform machines]

Thm [FKKMS02]: It is NP-hard to find the best and worst equilibria.

Nashification for the KP model

Thm [E-DKM03++]: There is a polynomial time algorithm which starting from an arbitrary schedule computes a NE for which the value of the GOF is not greater than the one of the original schedule.

Thus: There is a PTAS for computing a NE of minimum social cost for the KP model.

How can we improve the PA ?

Coordination mechanisms

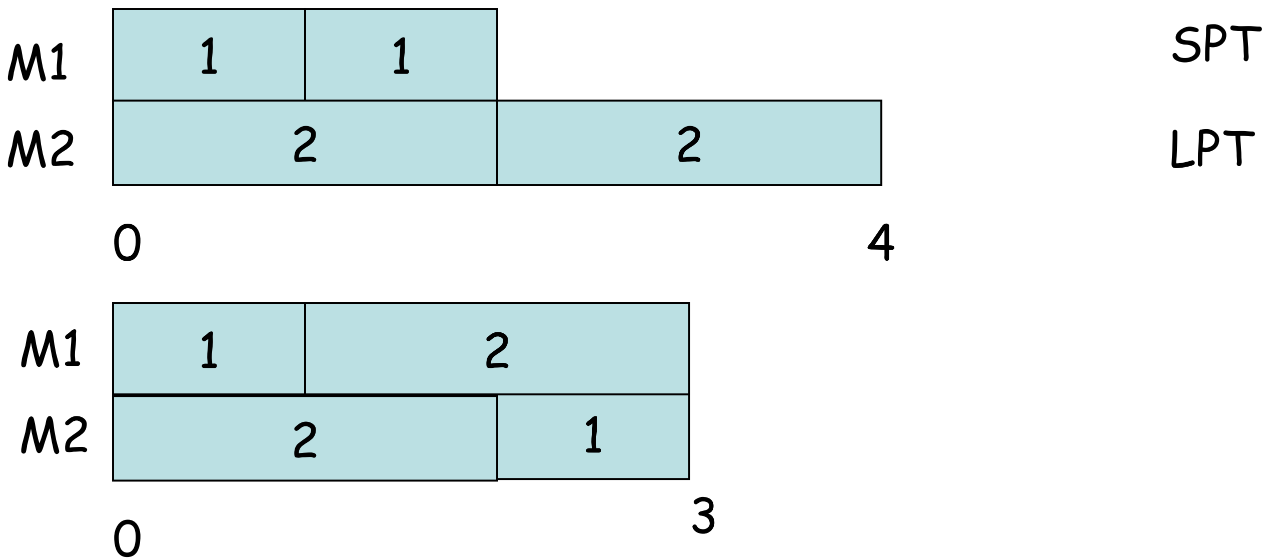
Aim: force the agents to cooperate willingly in order to minimize the PA

What kind of mechanisms ?

-Local scheduling policies in which the schedule on each machine depends only on the loads of the machine.

-each machine can give priorities to the tasks and introduce delays.

The LPT-SPT c.m. for the CKN model



Thm [CKN03]: The LPT-SPT c.m. has a price of anarchy of 4/3 for m=2.

[The LPT c.m. has a PA of 4/3-1/3m]

The Price of Stability (PS)

The framework: A protocol wishes to propose a collective solution to the users that are free to accept it or not.

Aim: Find the best (or a near optimal) NE

PS = (value of the GOF in the best NE)/OPT

Example:

- PS=1 for the KP model

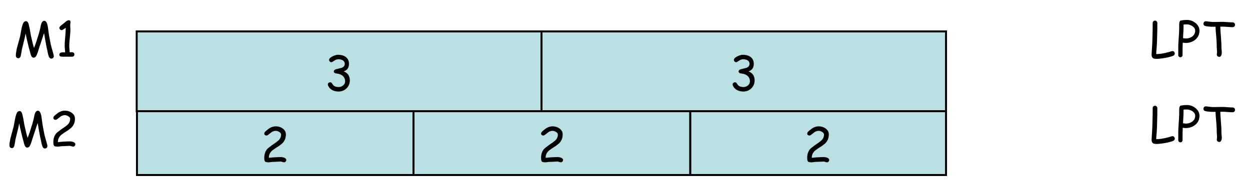
- PS=4/3-1/3m for the CKN model (with LPT l.p.)

Approximate Stability

Aim: Relax the notion of stable schedule in order to improve the price of anarchy.

α-approx. NE: a situation in which no agent has sufficient incentive to unilaterally change strategy, i.e. its profit does not increase more than α times its current profit.

Example: a 2-approx. NE

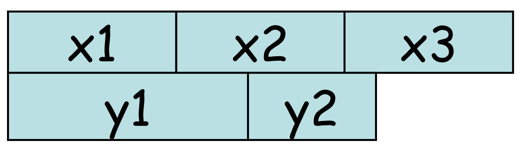


The algorithm LPT_{swap}

Thm[ABP05]: LPT_{swap} returns a 3-approx. NE and has a PA of 8/7.

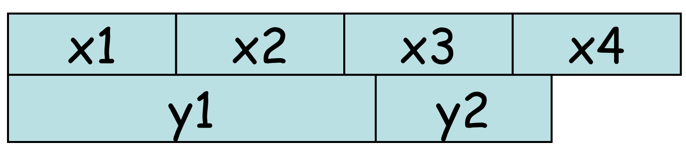
-construct an LPT schedule

-1st case:



Exchange: (x1,y1), or (x1,y2), or (x2,y2)
Return the best or LPT

-2nd case:



Exchange: (x3+x4,y2)
Compare with LPT and return the best

-3rd case: Return LPT

Thm[ABP05]: There is a polynomial time algorithm which returns a cste-approx. NE and has a PA of $1+\varepsilon$.

Truthful algorithms

The framework:

Even the most efficient algorithm may lead to unreasonable solutions if it is not designed to cope with the selfish behavior of the agents.

CKN model: Truthful algorithms

- The approach:
  - Task i has a secret real length l_i.
  - Each task bids a value $b_i \geq l_i$.
  - Each task knows the values bidded by the other tasks, and the algorithm.

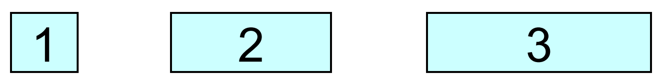
- Each task wishes to reduce its completion time.
- Social cost = maximum completion time (makespan)

- **Aim :** An algorithm truthful and which minimizes the makespan.

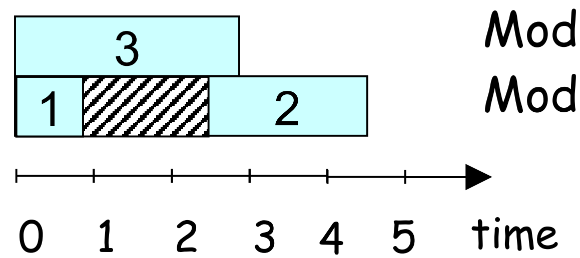
[Christodoulou, Koutsoupias, Nanavati: ICALP'04]

Two models

- Each task wish to reduce its completion time (and may lie if necessary).
- 2 models:
  - Model 1: If i bids b_i, its length is l_i
  - Model 2: If i bids b_i, its length is b_i
- Example: We have 3 tasks: , ,



Task 1 bids 2.5 instead of 1:

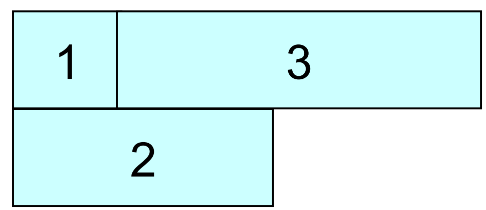


Model 1: $C_1 = 1$

Model 2: $C_1 = 2.5$

SPT: a truthful algorithm

- SPT: Schedules greedily the tasks from the smallest one to the largest one.
  - Example:



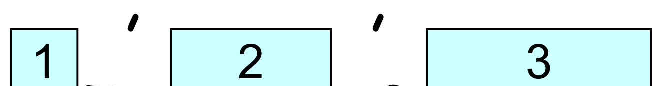
  - Approx. Ratio = 2 - 1/m [Graham]

- Are there better truthful algorithms ?

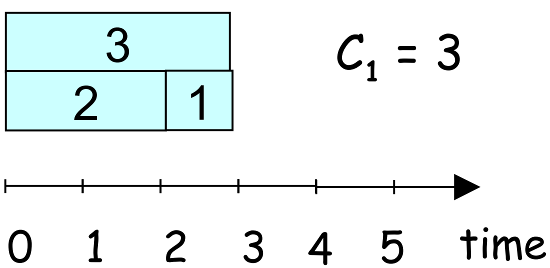
LPT

- LPT: Schedules greedily the tasks from the largest one to the smallest one.
  - Approx. Ratio = 4/3 - 1/(3m) [Graham]

- We have 3 tasks:

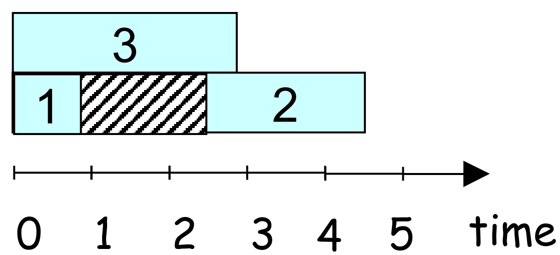


Task 1 bids 1:



$C_1 = 3$

Task 1 bids 2.5:



$C_1 = 1$

⇨ Task 1 has incentive to bid 2.5, and **LPT is not truthful**.

Randomized Algorithm

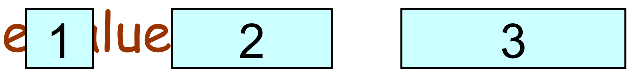
- Idea: to combine:
  - A truthful algorithm
  - An algorithm not truthful but with a good approx. ratio.
- Task: wants to minimize its expected completion time.
- Our Goal: A truthful randomized algorithm with a good approx. ratio.

Outline

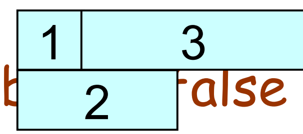
- Truthful algorithm
  - SPT-LPT is not truthful
  - Algorithm: SPTδ
  - A truthful algorithm: SPTδ-LPT

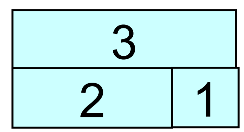
SPT-LPT is not truthful

- Algorithm SPT-LPT:
  - The tasks bid their values
  - With a proba. p, returns an SPT schedule.
    With a proba. (1-p), returns an LPT schedule.

- We have 3 tasks : 

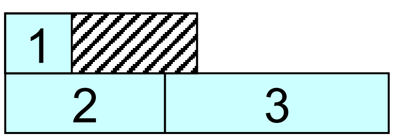
  - Task 1 bids its true value

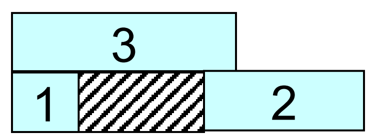
SPT : 

LPT : 

$C_1 = p + 3(1-p)$
$= 3 - 2p$

  - Task 1 bids a false value : 2.5

SPT : 

LPT : 

$C_1 = 1$

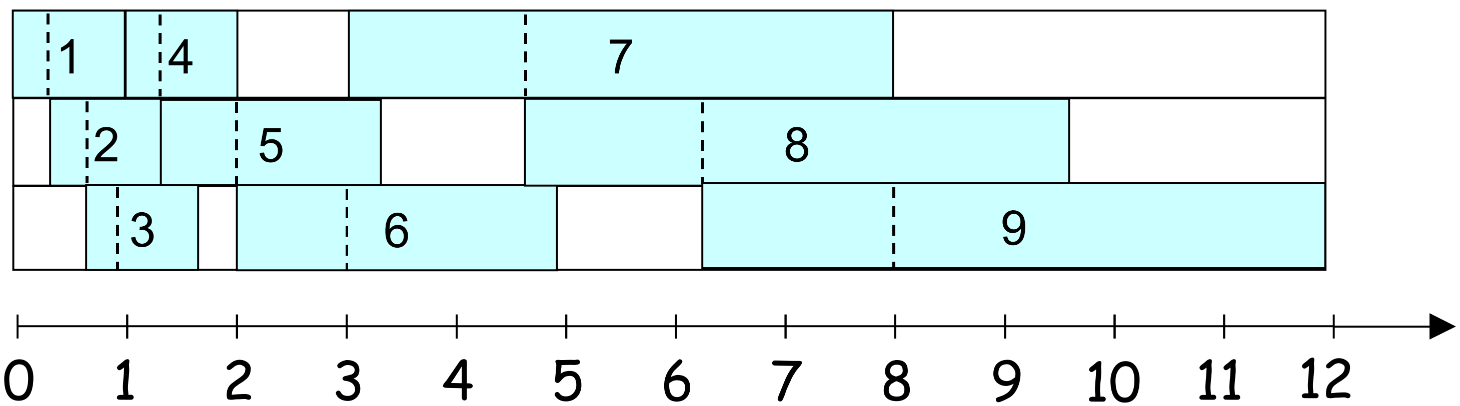
Algorithm SPTδ

- SPTδ:

  Schedules tasks 1,2,…,n s.t. $l_1 < l_2 < \ldots < l_n$

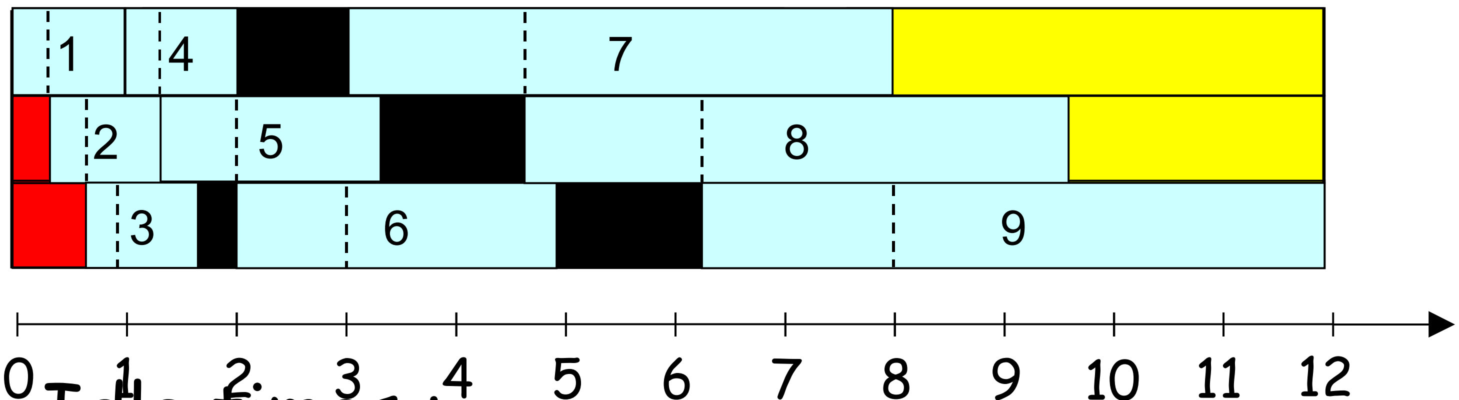
  Task (i+1) starts when 1/m of task i has been executed.

- Example: (m=3)



Algorithm SPTδ

- **Thm:** SPTδ is (2-1/m)-approximate.
- **Idea of the proof:** (m=3)



- Idle times :

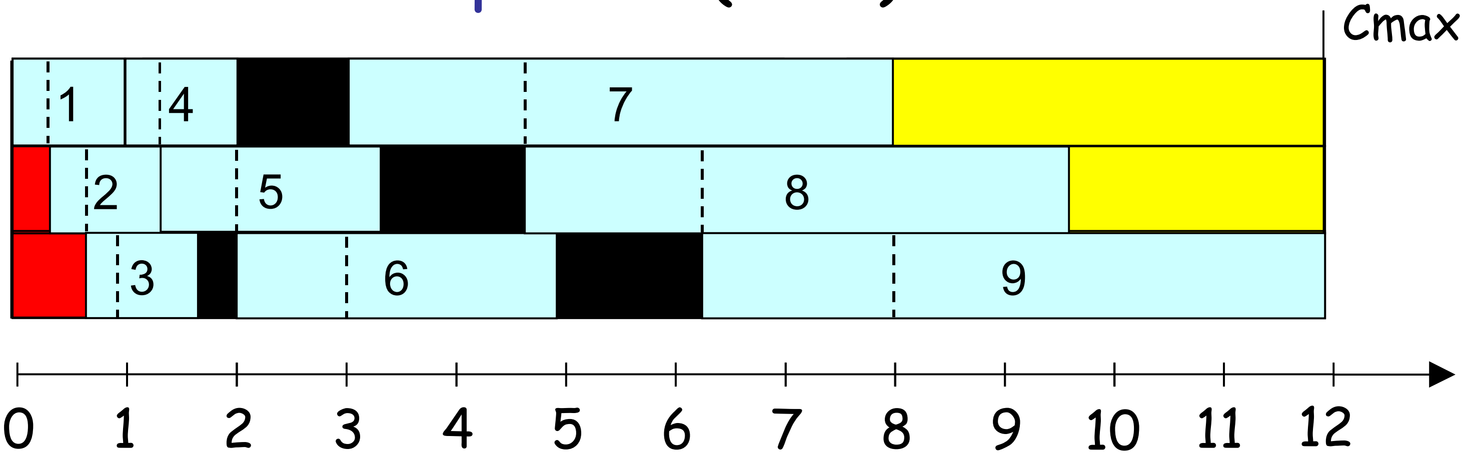
  idle_beginning(i) = $\sum (1/3\ l_j)$

  idle_middle(i) = $1/3\ (\ l_{i-3} + l_{i-2} + l_{i-1}\ ) - l_{i-3}$

  idle_end(i) = $l_{i+1} - 2/3\ l_i$ + idle_end(i+1)

Algorithm SPTδ

- Thm: SPTδ is (2-1/m)-approximate.
- Idea of the proof: (m=3)



Cmax = (Σ(idle times) + Σ(li)) / m

Σ(idle times) ≤ (m-1) l_n and l_n ≤ OPT

⇒ Cmax ≤ (2 - 1/m) OPT

A truthful algorithm: SPTδ-LPT

- Algorithm SPTδ-LPT:
  - With a proba. $m/(m+1)$, returns SPTδ.
  - With a proba. $1/(m+1)$, returns LPT.

- The expected approx. ratio of SPTδ - LPT is smaller than the one of SPT: e.g. for $m=2$, ratio(SPTδ-LPT) < 1.39, ratio(SPT)=1.5

- Thm: SPTδ-LPT is truthful.

A truthful algorithm: SPTδ-LPT

- **Thm:** SPTδ-LPT is truthful.

  Idea of the proof:

- Suppose that task i bids $b > l_i$. It is now larger than tasks 1,…, x, smaller than task x+1.

$$l_1 < \ldots < \cancel{l_i} < l_{i+1} < \ldots < l_x < b < l_{x+1} < \ldots < l_n$$

- LPT: decrease of $C_i(lpt) \leq (l_{i+1} + \ldots + l_x)$
- SPTδ: increase of $C_i(spt\delta) = 1/m\ (l_{i+1} + \ldots + l_x)$
- SPTδ-LPT:

  change = $- m/(m+1)\ C(spt\delta) + 1/(m+1)\ C(spt\delta) > 0$

AT model: Truthful algorithms

Monotonicity: Increasing the speed of exactly one machine does not make the algorithm decrease the work assigned to that machine.

Thm [AT01]: *A* mechanism *M*=(*A*,*P*) is truthful iff *A* is monotone.

An example

The greedy algorithm is not monotone.

Instance: 1, ε, 1, 2-3 ε, for $0<\varepsilon<1/3$

Speeds (s1,s2)	M1	M2
(1,1)	1, ε	1, 2-3 ε
(1,2)	ε, 2-3 ε	1,1

Results for the AT model

3-approx randomized mechanism [AT01]

(2+ε)-approx mechanism for divisible speeds and integer and bounded speeds [ADPP04]

(4+ε)-approx mechanism for fixed number of machines [ADPP04]

12-approx mechanism for any number of machines [AS05]

Conclusion

- Future work:

-Links between LS and game theory

-Many variants of scheduling problems

-Repeated games

…