Multi-agent scheduling problems

Alessandro Agnetis, Università di Siena
Dipartimento di Ingegneria dell'Informazione

Toulouse, LAAS-CNRS 13/5/2011

INTRODUCTION: MULTI-AGENT SCHEDULING

Centralized decision making

- Most classical scheduling models assume a single decision maker, having total control over his/her processes
- The focus is on the *optimal* way of allocating processing resources

Multi-agent scheduling problems

- Various *agents* compete for a limited set of (production, logistic etc…) resources
- The agents must *negotiate* resource usage
- Possibly, a central entity may *coordinate* or facilitate the bargaining process

Multi-agent scheduling problems

- A set of m agents
- Each agent owns a set of jobs, requiring certain processing resources for a given time
- Each agent i has a *cost function* $f^{(i)}(\sigma)$ depending only on its own jobs' schedule

Multi-agent scheduling problems

- A "globally optimal" schedule (e.g. maximizing total utility) may not be the "best" solution, because it may be unfair
- Any solution must result from a decision process in which all agents are involved
- How to define and find a "good" compromise schedule?

Information exchange - I

- The agents accept to share their information (possibly with a coordinator), if this can help reaching a satisfactory resource allocation
- Among all possible solutions, some have properties that make them special candidates for the outcome of negotiation
- Combinatorial models

Information exchange - II

- The agents not only accept to share information, but even allow *side payments* to compensate the agents who are less favored in the schedule
- A solution is therefore a schedule complemented by a payment scheme
- Cooperative games (sequencing games…)

Information exchange - III

- The agents are not willing/able to disclose complete information concerning their respective jobs and objectives
- Resource allocation is carried out by means of a decentralized protocol
- Auction-type mechanisms (market oriented programming, automated negotiation)

MOTIVATING EXAMPLES

Aircraft arrival planning

[Soomer and Franx 2008]

- Aircraft land on a single runway of an airport
- Airlines (agents) have costs related to the delay of their flights (jobs)
- The runway can be used by one aircraft at a time, and there are minimum separation times between landings

Telecommunications - Packet scheduling [Peha 1995, Meiners and Torng 2009]

- Radio resources are used by various types of services (agents), such as voice, ftp, web browsing…
- Packets (jobs) have therefore different costs related to their transmission schedule
- Some voice-packets can go lost but most of them must arrive within a deadline, ftp can be slower but no packet must be lost

Manufacturing - maintenance, rescheduling

[Kubzin and Strusevich 2006, Leung, Pinedo, Wan 2010]

- Maintenance and manufacturing are two distinct activities (agents), each consisting of several operations (jobs) that compete for resource usage
- As new jobs arrive, some other jobs have to be re-scheduled, and hence hold a different objective from the other operations

Multi-agent scheduling problems

- A set of k agents, each owning a set of *jobs*
- Each job has a certain processing time p_j and requires some processing resource
- Each agent i wants to minimize a cost function $f^i(\sigma)$ which only depends on the schedule of its jobs
- What is the "best" way of allocating jobs to the resources?

COMPLEXITY

Multi-objective problems

1. Minimize one agent's cost function with a constraint on the other agents' cost functions (*ε-constrained approach*):

$$\min f^k(\sigma)$$
$$f^1(\sigma) \leq Q_1$$
$$f^2(\sigma) \leq Q_2$$
$$\ldots\ldots$$
$$f^{k-1}(\sigma) \leq Q_{k-1}$$

For each Q_i, this approach allows to compute one Pareto optimal schedule

$1 \mid \Sigma_i C_i \leq Q \mid T_{max}$ - Example

Agent 1

$f^1 = \Sigma_i C_i$

$Q = 43$

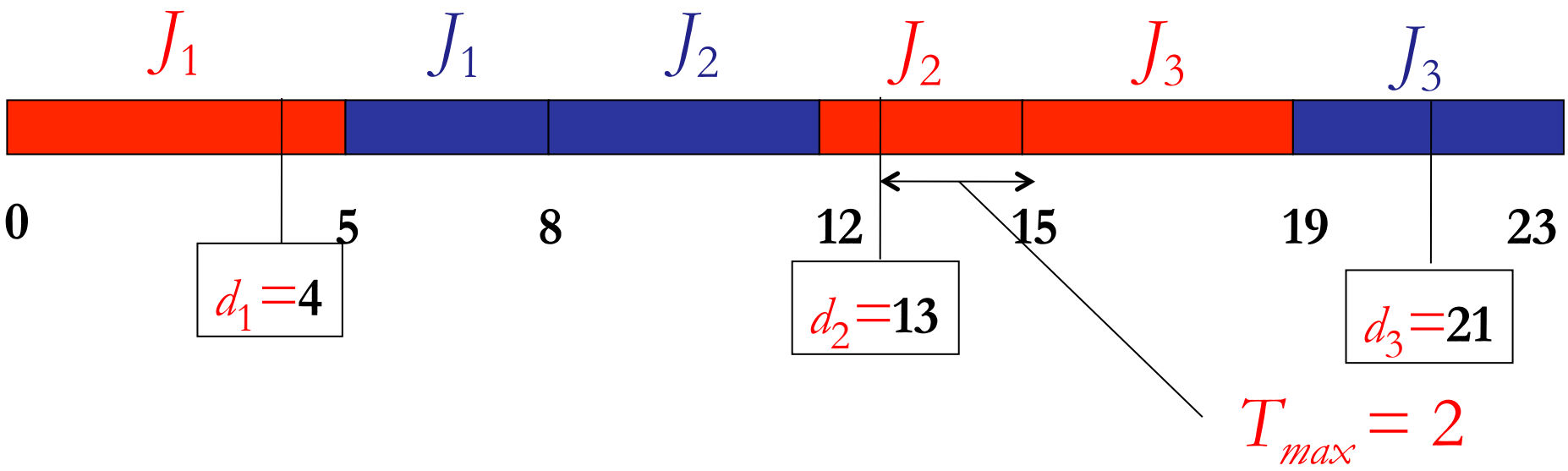
J_i	p_i
1	3
2	4
3	4

Agent 2

$f^2 = T_{max}$

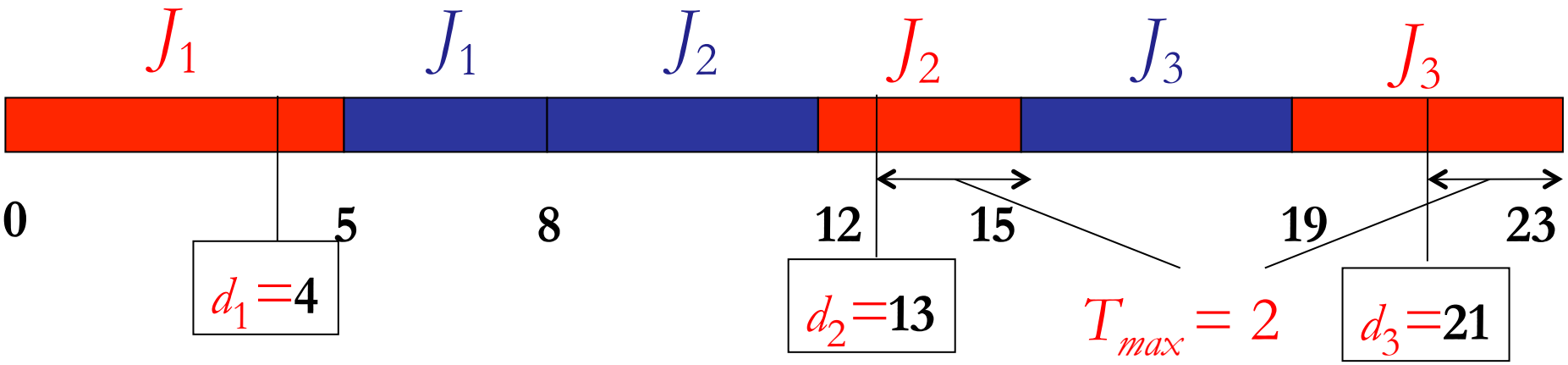
J_i	p_i	d_i
1	5	4
2	3	13
3	4	21

Schedule σ



$$\Sigma_i C_i = 8 + 12 + 23 = 43$$

Schedule σ'



$$\Sigma_i C_i = 8 + 12 + 19 = 39$$

$1 \mid C_{max} \leq Q \mid \sum w_i C_i$ - Example

Agent 1

$f^1 = C_{max}$

J_i	p_i
1	10

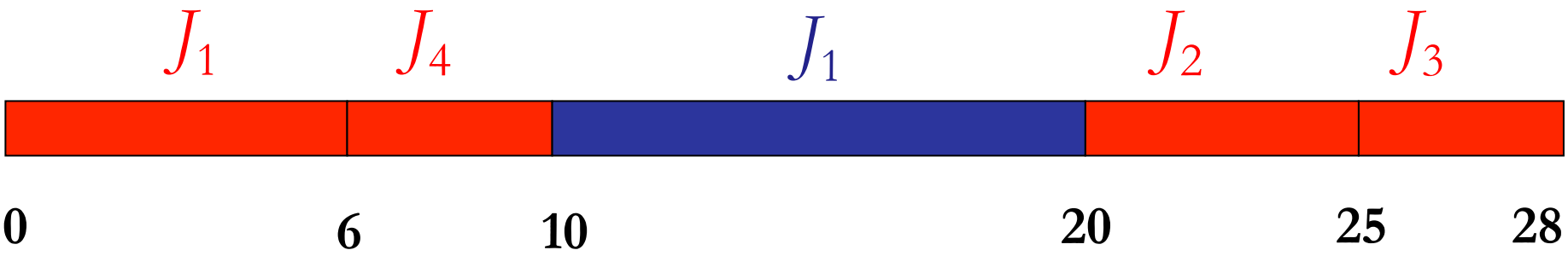
$Q = 20$

Agent 2

$f^2 = \sum_i w_i C_i$

J_i	p_i	w_i
1	6	9
2	5	7
3	3	4
4	4	5

Optimal solution σ^*



$$\sum_i w_i C_i(\sigma) = 9*6+5*10+7*25+4*28$$

J_i	p_i	w_i
1	6	9
2	5	7
3	3	4
4	4	5

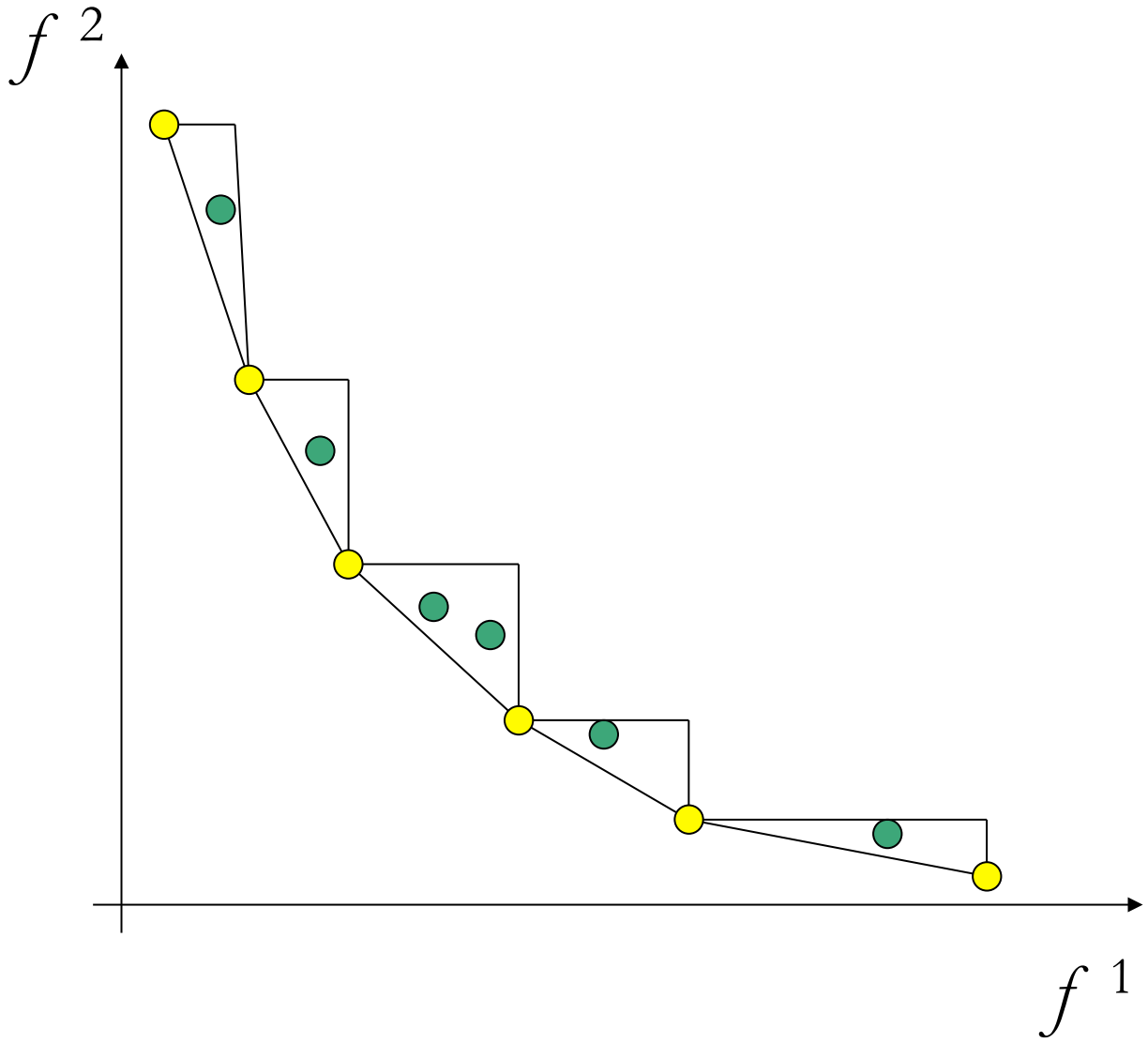
$$C_{max}(\sigma^*) = 20$$

Multi-objective problems

2. Minimize a convex combination of the agents' cost functions:

$$\min \Sigma_i \lambda_i f^i(\sigma)$$

This allows to compute all *extreme schedules* (subset of Π)


f 2
f 1

Multi-objective problems

3. Compute *all* Pareto optimal schedules

   The complexity of this task depends on the complexity of the ε-constrained problem and on the size of Π

Complexity results

[Baker and Smith 2003, Hoogeveen 2002]

[A., Mirchandani, Pacciarelli and Pacifici 2004]

[A., De Pascale and Pacciarelli 2009, Sourd 2008]

[Leung, Pinedo and Wan 2010]

[Yuan, Shan and Feng 2005]

[Cheng, Ng and Yuan 2006]

1	2	ε-constr.	$\lvert\Pi\rvert$	$\lambda f^1+(1-\lambda)f^2$
f_{max}	f_{max}	$O(n^2)$	$O(n_1 n_2)$	$O(n^4)$
$\Sigma w_j C_j$	C_{max}	Bin. NP-hard	Pseudopol.	$O(n \log n)$
$\Sigma w_j C_j$	L_{max}	Str. NP-hard	Pseudopol.	NP-hard
ΣC_j	f_{max}	$O(n \log n)$	$O(n_1 n_2)$	$O(n^3 \log n)$
ΣU_j	f_{max}	$O(n \log n)$	$O(n_1)$	$O(n^2 \log n)$
ΣU_j	ΣU_j	$O(n^3)$	$O(n_1)$	$O(n^4)$
$\Sigma w_j U_j$	$\Sigma w_j U_j$	Bin. NP-hard	Pseudopol.	NP-hard
ΣC_j	ΣU_j	NP-hard	$O(n_2)$	NP-hard
$\Sigma w_j C_j$	ΣU_j	Str. NP-hard	$O(n_2)$	NP-hard
ΣC_j	ΣC_j	Bin. NP-hard	Pseudopol.	$O(n \log n)$
$\Sigma w_j C_j$	$\Sigma w_j C_j$	NP-hard	Pseudopol.	$O(n \log n)$

Release dates

- $1 \mid r_j \mid \lambda C_{max}^{1} + (1-\lambda) C_{max}^{2}$ is binary NP-hard [Ding and Sun 2010]
- This motivates addressing problems with release dates *and* preemption
- Some results consider *mixed preemption:* one agent's jobs can be preempted, the other agent's cannot (e.g. maintenance cannot be preempted by "true" jobs)

Parallel machines, *preemptive* problems (two agents) [Leung, Pinedo and Wan 2010]

1	2	*No. of machines*		ε-constrained problem
f_{max}	f_{max}	m		$O((\log Q)\ n \log mn)$
ΣC_j	f_{max}	2		$O(n \log n)$
ΣU_j	f_{max}	fixed m		$O(n^{3m-5}\ n_1^2 n_2^2)$
ΣU_j	f_{max}	m not fixed		Binary NP-hard
f_{max}	f_{max}	m	r_j	$O((\log Q)\ n^3)$
ΣC_j	f_{max}	fixed m	r_j	Binary NP-hard
ΣU_j	ΣU_j	fixed m	r_j	Binary NP-hard

Makespan minimization in a flow shop with two agents

[A., Mirchandani, Pacciarelli and Pacifici 2004, Huynh Tuong, Soukhal and Billaut 2008, Huynh Tuong, Soukhal 2009]

1	2	M	ε-constrained problem
$M1 \rightarrow M2$	$M1 \rightarrow M2$	2	Binary NP-hard
$M1 \rightarrow M2$	$M1 \rightarrow M3$	3	Strongly NP-hard
$M1 \rightarrow M2$	$M2 \rightarrow M3$	3	Binary NP-hard
$M1 \rightarrow M3$	$M2 \rightarrow M3$	3	Strongly NP-hard

k agents [A., Pacciarelli and Pacifici 2007]

- The ε-constrained problem is polynomially solvable whenever:

$f^i(\sigma) = f_{max}$	$i=1,\ldots,k$

$f^i(\sigma) = f_{max}$	$i=1,\ldots,k\text{-}1$
$f^k(\sigma) = \Sigma_i C_i$	

$f^i(\sigma) = f_{max}$	$i=1,\ldots,p$
$f^i(\sigma) = \Sigma_i U_i$	$i=p+1,\ldots,k$

k agents

- The ε-constrained problem is strongly NP-hard when [Cheng, Ng and Yuan 2006]

$f^i(\sigma) = \sum_i w_i U_i \quad i=1,\ldots,k$ $\qquad$ k not fixed

- The convex combination problem is NP-hard when [Cheng, Ng and Yuan 2008]

$f^i(\sigma) = L_{max} \quad i=1,\ldots,k$ $\qquad$ k not fixed

Research issues

Recently, the complexity of more multi-agent problems is being studied, including:

- Controllable/compressible processing times
- Deteriorating jobs
- Multi-agent project scheduling

BRANCH AND BOUND

Only few papers on branch and bound

Solution algorithms for some of the hard cases (*ε-constrained approach*):

$$\min \Sigma_i w_i^1 C_i^1$$

$$\{\Sigma_i w_i^2 C_i^2, C_{max}^2, L_{max}^2\} \leq Q$$

in all these cases, the Lagrangian dual can be solved very efficiently

[A., De Pascale, Pacciarelli 2009]

$$\min \sum_i w_i^1 C_i^1(\sigma)$$

$$\sum_i w_i^2 C_i^2(\sigma) \leq Q$$

$$\sigma \in S$$

- If we relax the constraint, we get the Lagrangian problem:

$$L(\lambda) = \min_{\sigma \in S} \sum_i w_i^1 C_i^1(\sigma) + \lambda \left(\sum_i w_i^2 C_i^2(\sigma) - Q\right)$$

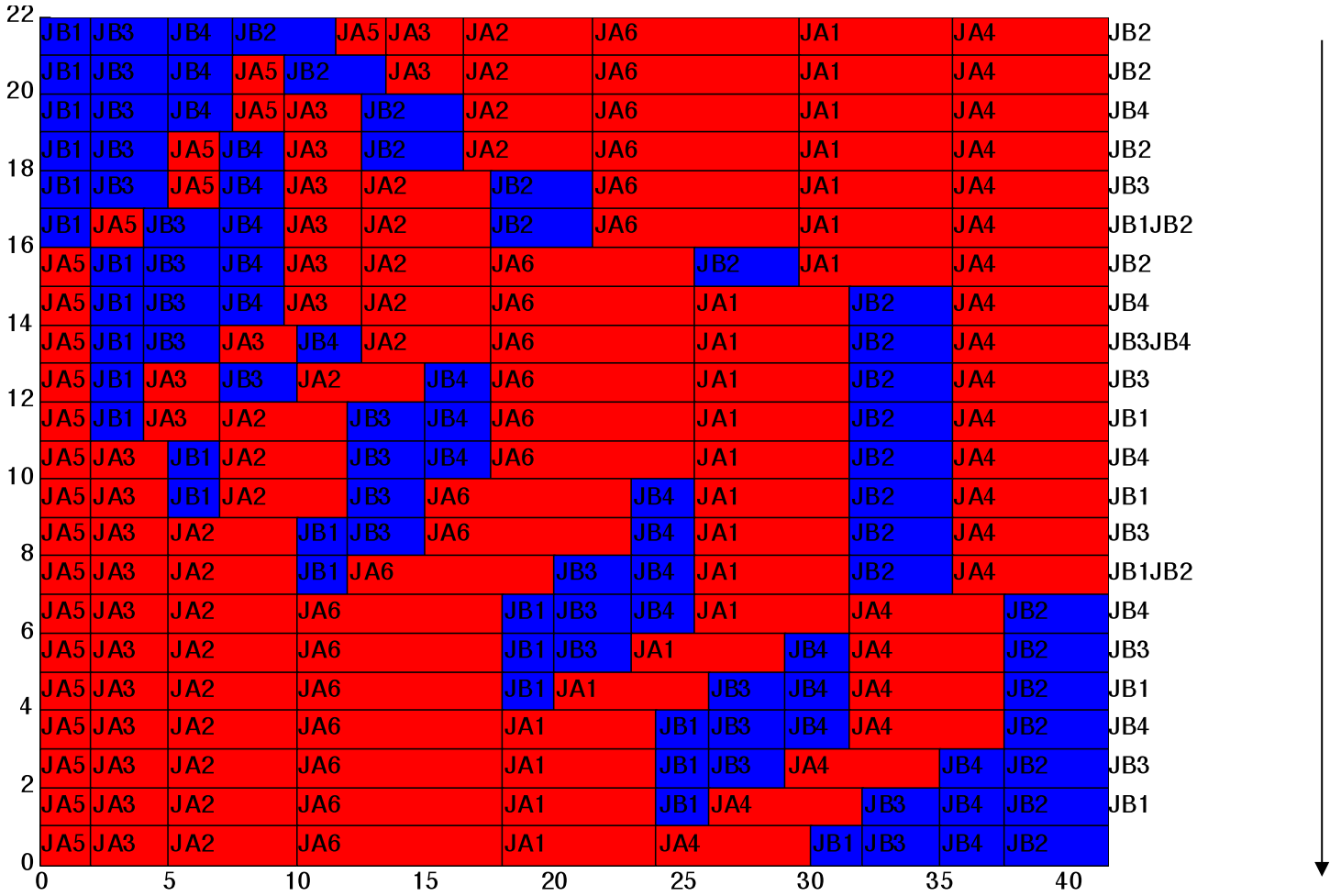
Lagrangian relaxation

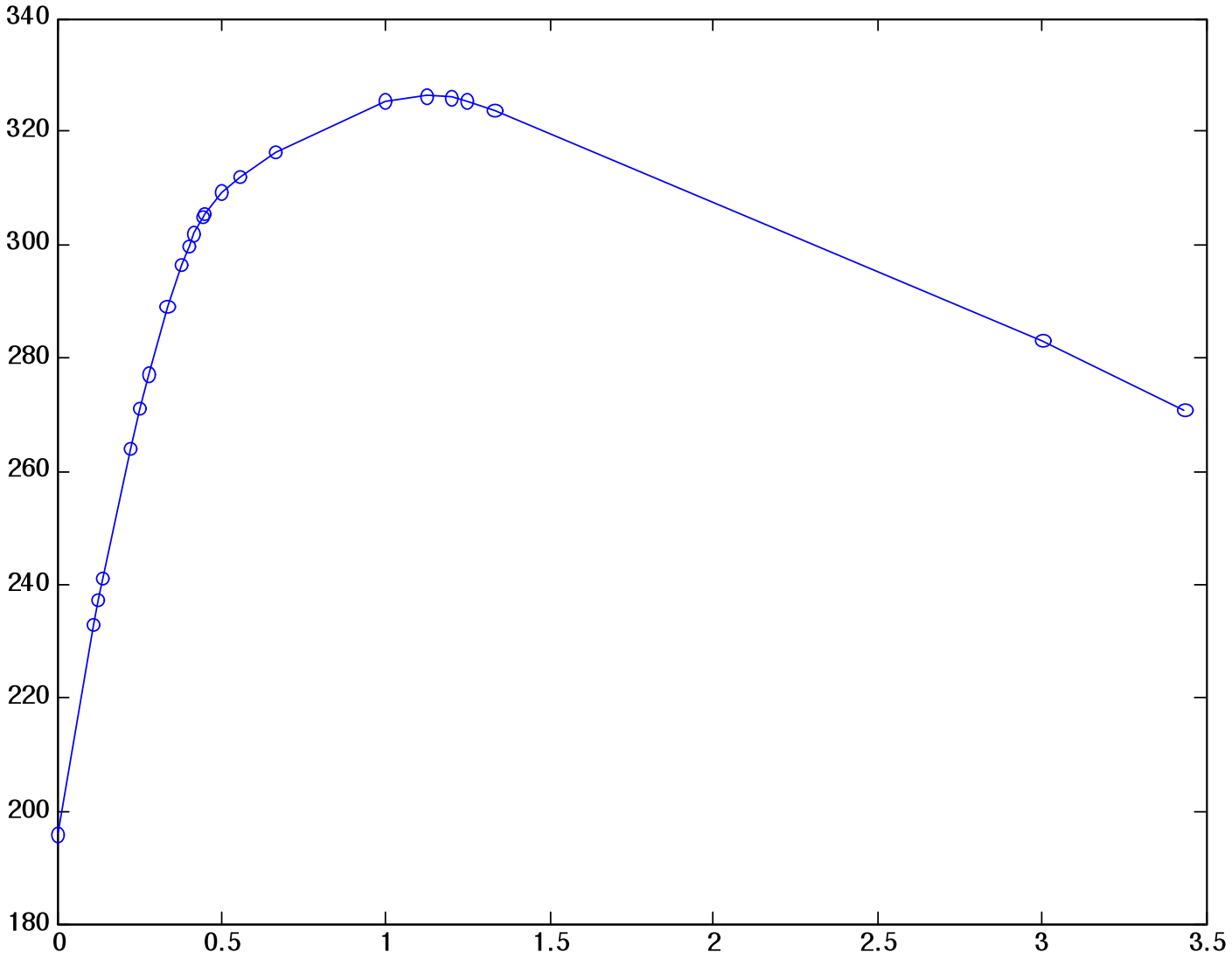
- The Lagrangian problem is simply solved by ranking the jobs in nondecreasing order of δ_k where

$$\delta_k^1 = w_k^1 / p_k^1 \qquad \text{if } k \in 1$$

$$\delta_k^2 = \lambda\, w_k^2 / p_k^2 \qquad \text{if } k \in 2$$

Optimal schedules for decreasing λ




340
320
300
280
260
240
220
200
180
0
0.5
1
1.5
2
2.5
3
3.5

Lower bounds

- The problem $1 \mid L_{max}^{2} \leq Q \mid \Sigma w_j^{1} C_j^{1}$ is a *special case* of

$$1 \mid d_j \mid \Sigma_j w_j C_j$$

- Pan (2003) solves instances with up to 100 jobs, based on a bounding approach by Posner (1995)
- T'Kindt, Della Croce and Esswein (2004) use an improved search strategy to solve instances up to 130 jobs

Results

- A branch and bound algorithm based on the Lagrangian bounds allows to solve problems with up to 80 jobs, without the enhancements due to the dominance rule
- "Two-agent" instances apparently are harder than the average instances of

$$1 \mid d_j \mid \Sigma_j w_j C_j$$

$F2 \mid L_{max}^{2} \leq 0 \mid \Sigma C_j^{1}$ [Lee, Chen, Chen and Wu 2010]

- The algorithm is based on several combinatorial bounds and solves problems with up to 20 jobs
- A simulated annealing approach produces an average error less than 1% on small instances (no hint on larger instances)
- Instances are harder as the fraction of jobs in J^2 grows (same as in the single machine case)

Research issues

- Refinement and extension of the Lagrangian approach (more than two agents, other objective functions…)
- Improvement of the enumeration procedure, deeper understanding of the problems

FAIRNESS

Fairness vs. efficiency

- Among all Pareto optimal solutions, some treat the agents very differently
- When various agents have conflicting objectives, an external coordinator may help reaching a mutually acceptable solution, ensuring a satisfactory level of efficiency and fairness

Fairness vs. efficiency

- System efficiency is usually measured through the sum of the individual objective functions (supposed homogeneous)
- "Fairness" can be defined in different ways, often not equivalent
- We refer to the problem $1||(\Sigma_j C_j^1, \Sigma_j C_j^2)$

Equitable solutions

- Consider two *Pareto optimal* schedules σ and σ' such that

$$f^1(\sigma) = f^1(\sigma') + \varepsilon$$
$$f^2(\sigma) = f^2(\sigma') - \varepsilon$$

then, if $f^1(\sigma') > f^2(\sigma')$, the schedule σ is *equitably-dominated* by σ'

Bargaining games

- A *bargaining game* is a particular game defined by:
  - a set X of possible *agreements*, each agreement σ having a certain *utility* $u^1(\sigma)$ and $u^2(\sigma)$ to each of the two agents
  - A *disagreement point* $D=(u^1(\sigma)$ and $u^2(\sigma))$, representing what the agents get the bargaining process fails
- The disagreement point is dominated by any other point in X

Solution of a bargaining game

- The *solution* of a bargaining game is a *function* that, given:
  - the bargaining set X
  - the disagreement point D
  - the utility functions $u^1(\sigma)$, $u^2(\sigma)$ of the two agents

  returns a solution $x^* \in X$ as a possible result of bargaining

Scheduling bargaining games

- Let σ^1 and σ^2 indicate the *best* schedule for agents 1 and 2 respectively (as if the other agent is not present)
- Clearly, σ^1 and σ^2 are also the *worst* schedule for agents 2 and 1 respectively
- We can set: $u^1(\sigma) = f^1(\sigma^2) - f^1(\sigma)$

$$u^2(\sigma) = f^2(\sigma^1) - f^2(\sigma)$$

Nash bargaining solution (NBS)

- Nash (1950) proposed a solution as an agreement x^* with respect to which no player has "enough motivation" to deviate
- It depends on the agents' risk attitudes
- It can be characterized in axiomatic terms, but also in terms of players' preferences over the set of *lotteries* among elements of the bargaining set X

Nash bargaining solution

- An agreement x^* such that if, for some agreement x and probability p, player A prefers

$$L = < p, x; 1\text{-}p, D >$$

to x^*,

then player B prefers

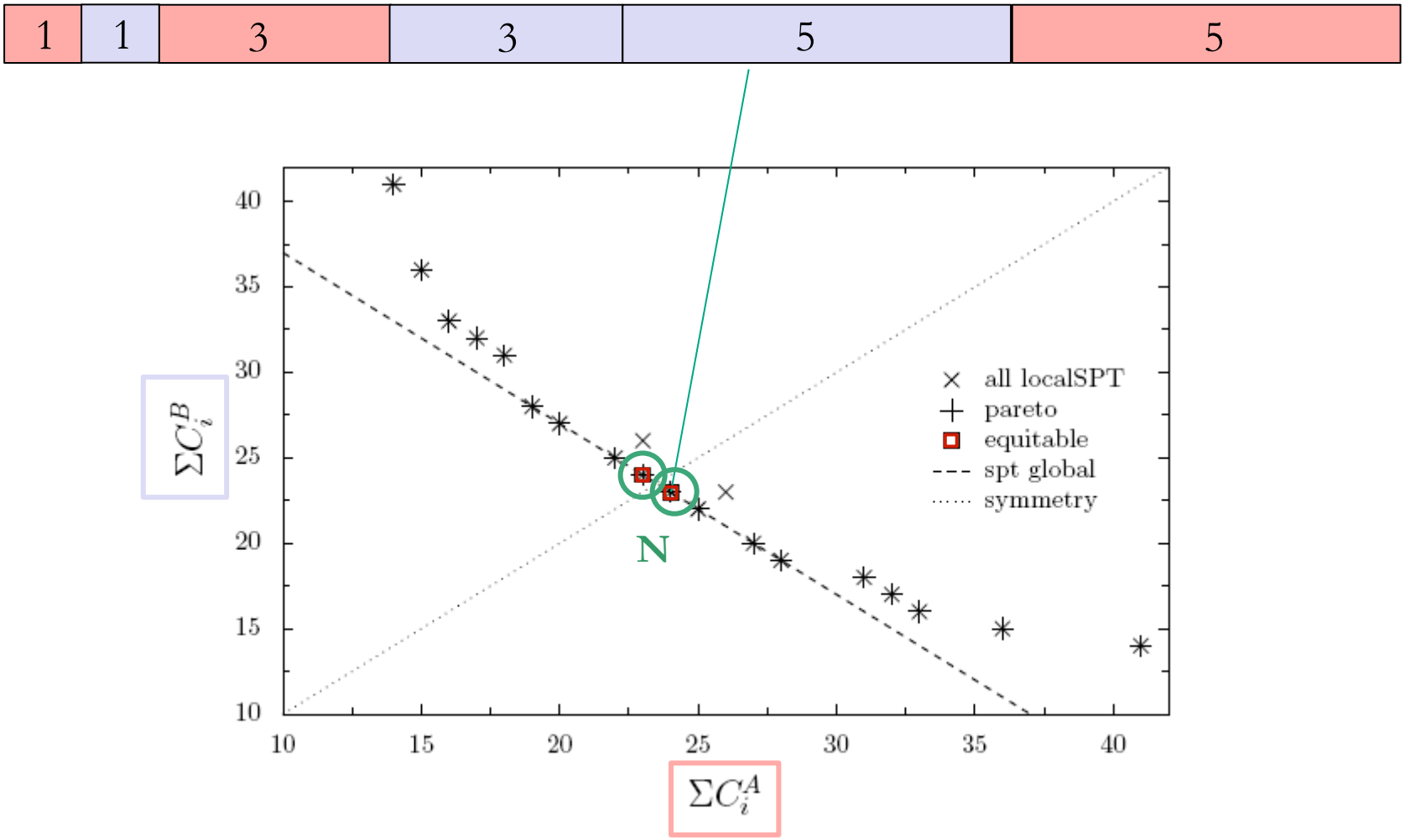
$$L = < p, x^*; 1\text{-}p, D >$$

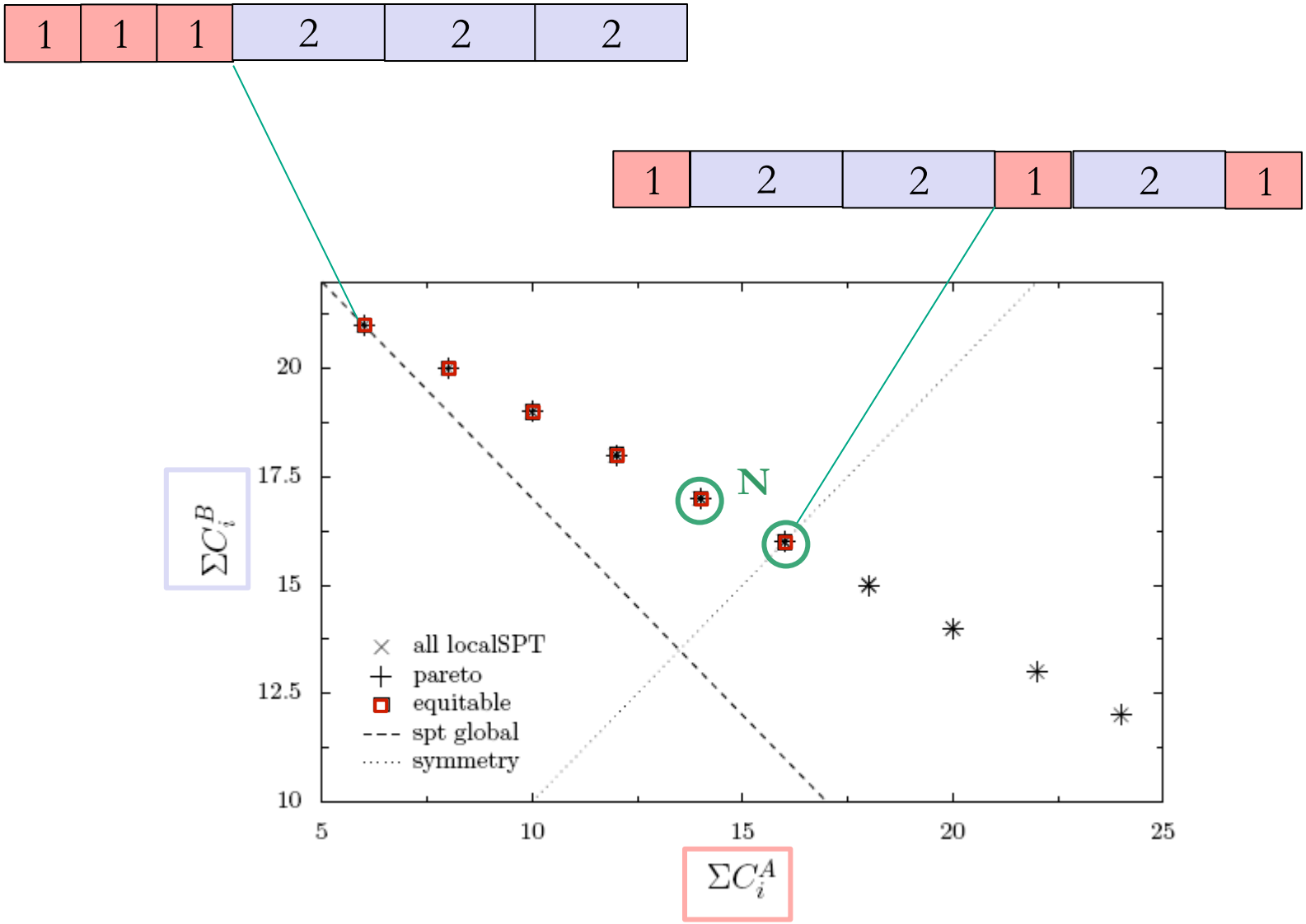
to x

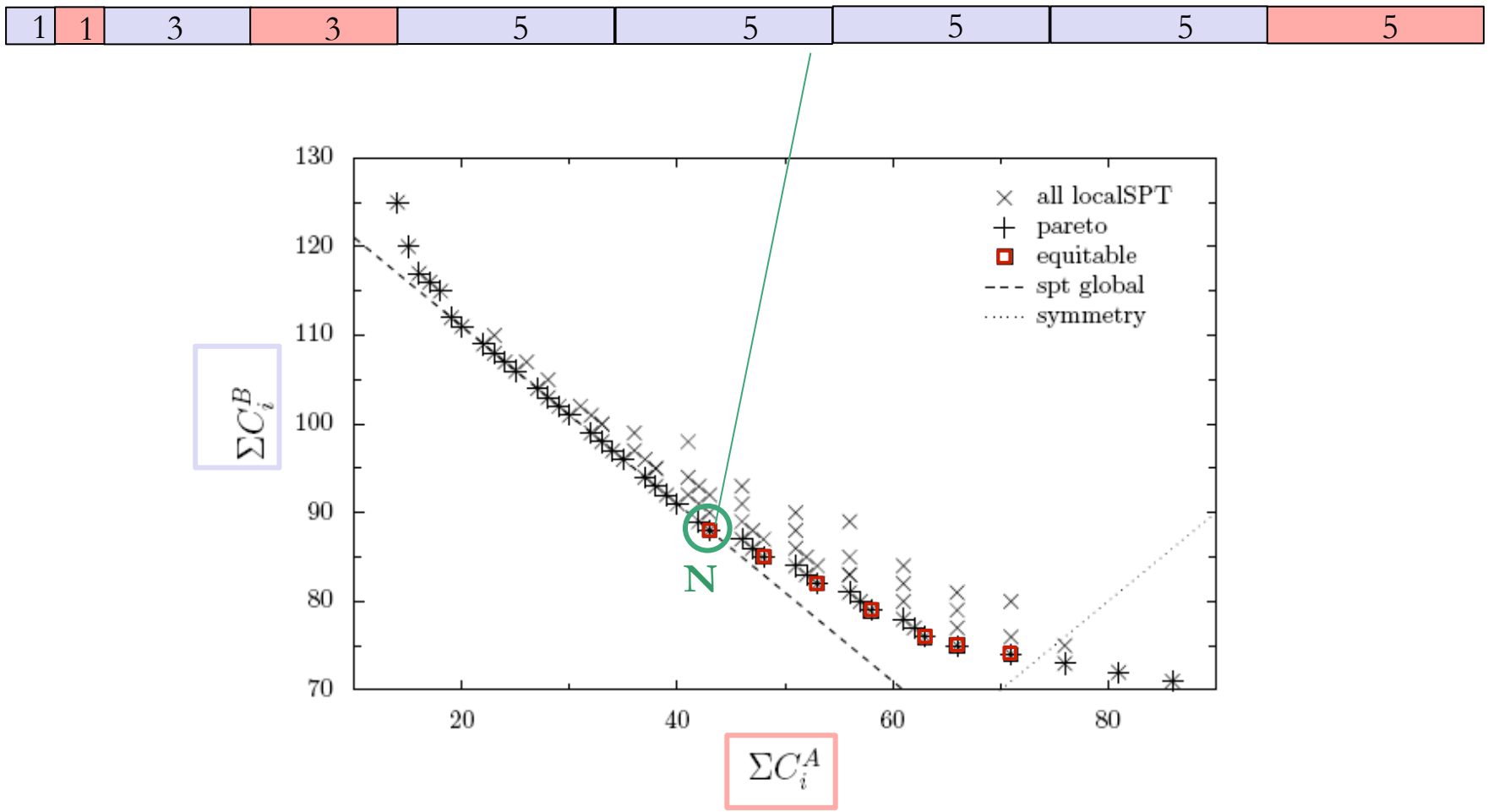
Nash bargaining solution

- The NBS can be conveniently characterized as the Pareto optimal schedule that maximizes the *product* of the two agents' utilities:

$$N(\sigma^*) = \max_{\sigma} [u^1(\sigma)\, u^2(\sigma)]$$


1
1
3
3
5
5
40
35
30
25
20
15
10
10
15
20
25
30
35
40
$\sum C_i^B$
$\sum C_i^A$
all localSPT
pareto
equitable
spt global
symmetry
N


1 1 1 2 2 2
1 2 2 1 2 1
N
20
17.5
15
12.5
10
5
10
15
20
25
ΣC_i^B
ΣC_i^A
all localSPT
pareto
equitable
spt global
symmetry


1
1
3
3
5
5
5
5
5
130
120
110
100
90
80
70
20
40
60
80
$\sum C_i^B$
$\sum C_i^A$
all localSPT
pareto
equitable
spt global
symmetry
N

Computing fair solutions

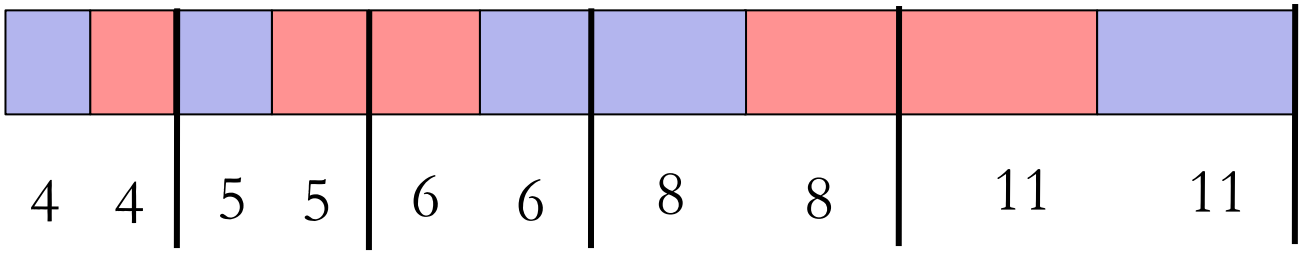
- For problem $1||(\Sigma_j C_j^1, \Sigma_j C_j^2)$, a globally *efficient* solution can be found by scheduling all jobs in SPT
- Finding a *fair* solution or set of solutions is in general binary NP-hard
- (note that showing that a schedule σ is a NBS is not in NP)

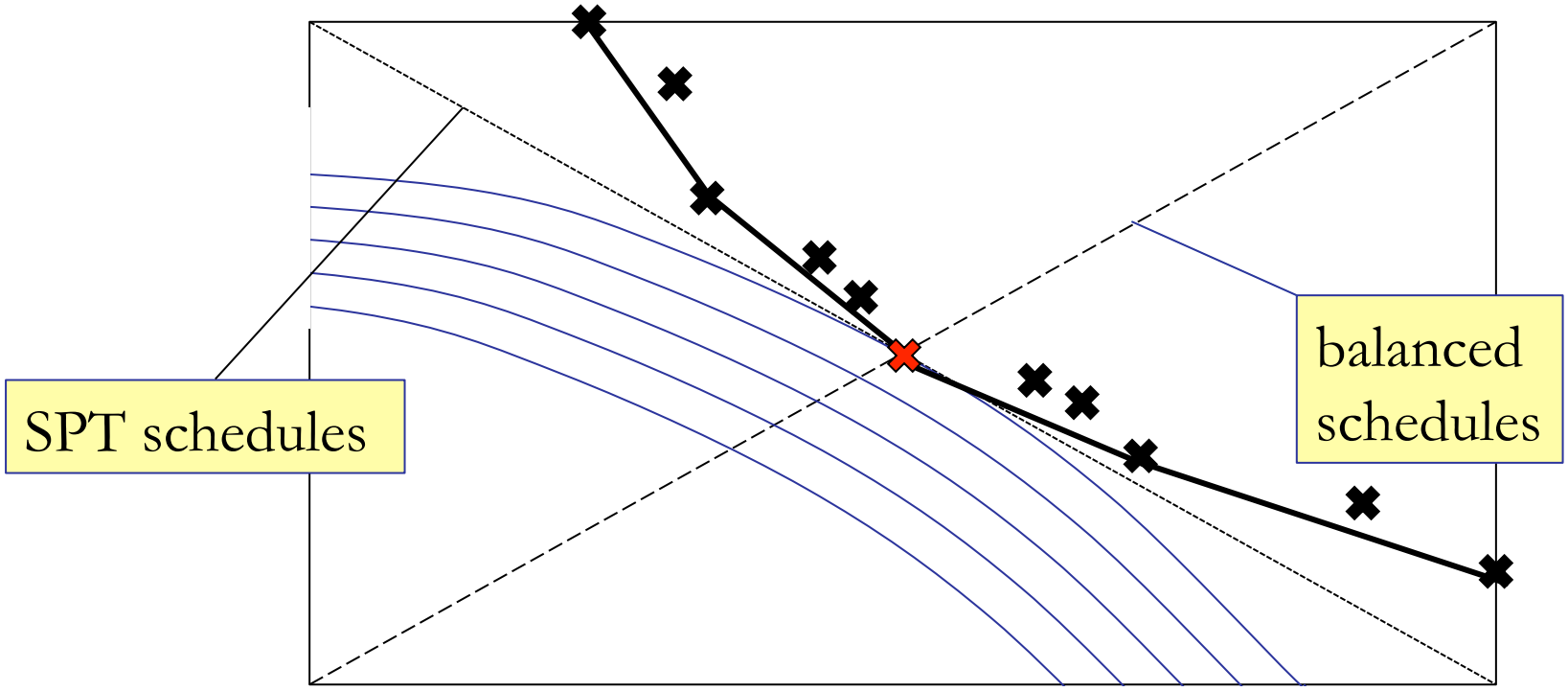
Computing fair solutions

- Consider an instance of PARTITION, e.g.

  $\{4, 5, 6, 8, 11\}$

- Build an instance of $1||(\Sigma_j C_j^1, \Sigma_j C_j^2)$ in which the agents have identical jobs, of length equal to the objects of PARTITION
- If and only if it is a yes-instance, an *ideal* solution exists (therefore it coincides with the only NBS, equitably-nondominated etc)

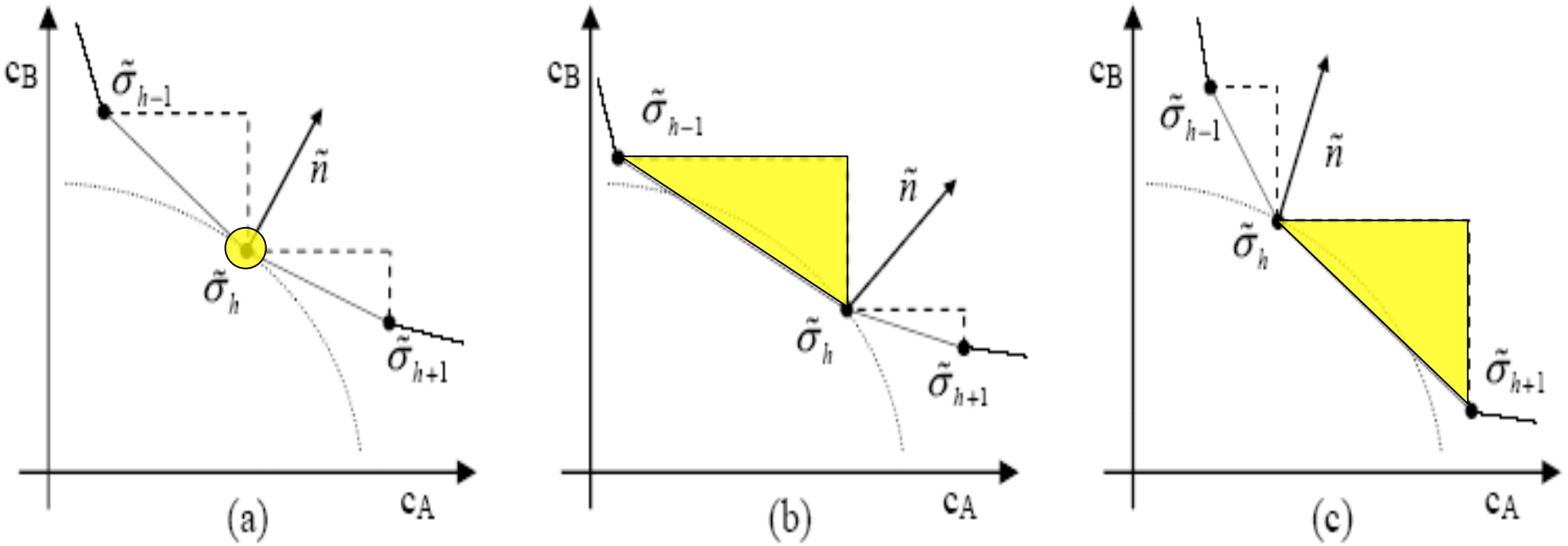

4 4 5 5 6 6 8 8 11 11


SPT schedules
balanced schedules

PO schedules and NBS

[A., De Pascale and Pranzo 2009]

- Even if an ideal solution does not exist, the NBS can be efficiently found by the following approach:
  - Generate all extreme schedules
  - Locate the *triangle* containing the NBS
  - Enumerate Pareto optimal solutions in the triangle


cB
$\tilde{\sigma}_{h-1}$
$\tilde{n}$
$\tilde{\sigma}_h$
$\tilde{\sigma}_{h+1}$
cA
(a)
cB
$\tilde{\sigma}_{h-1}$
$\tilde{n}$
$\tilde{\sigma}_h$
$\tilde{\sigma}_{h+1}$
cA
(b)
cB
$\tilde{\sigma}_{h-1}$
$\tilde{n}$
$\tilde{\sigma}_h$
$\tilde{\sigma}_{h+1}$
cA
(c)

Computational experiments

- The approach has been run on several instances of various sizes
- $J^A = \{10, 20, 30, 40\}$
- $J^B = \{10, 20, 30, 40\}$
- All weights and processing times uniformly distributed in [1,25]

n_A	n_B	$\|E\|$	$\|\Pi\|$	T_1	T_Π	T_{NASH}
10	**10**	**60**	**603**	**0.01**	**4**	**0.06**
10	**20**	**119**	**4595**	**0.05**	**217**	**1.72**
10	**30**	**178**	**15383**	**0.14**	**2185**	**12.30**
10	**40**	**232**	**74771**	**0.32**	**24061**	**102.27**
20	**10**	**118**	**4698**	**0.05**	**227**	**1.86**
20	**20**	**239**	**15601**	**0.14**	**2110**	**8.96**
20	**30**	**345**	**74547**	**0.33**	**24912**	**69.21**
20	**40**	**451**	*220000*	**0.65**	*146400*	**322.87**
30	**10**	**178**	**15413**	**0.14**	**2120**	**12.48**
30	**20**	**346**	**75225**	**0.31**	**22883**	**65.24**
30	**30**	**510**	*219000*	**0.64**	*140700*	**282.58**
30	**40**	**662**	*950000*	**1.12**	*1056000*	**1571.23**
40	**10**	**233**	**73952**	**0.33**	**24427**	**110.23**
40	**20**	**452**	*220000*	**0.63**	*138000*	**311.06**
40	**30**	**653**	*947000*	**1.09**	*1062000*	**1551.55**
40	**40**	**862**	*2397000*	**1.81**	*4218000*	**4974.83**

Future research

- Derive appropriate fairness concepts also for nonhomogeneous objective functions
- Design algorithms to actually compute fair solutions in several settings

SEQUENCING GAMES

Transferable utility

- A different situation is that in which agents, besides communicating, can also *transfer utility* among them, i.e., side payments are possible
- This applies to situations in which, starting from an initial solution, an overall better solution exists, and payments can help reaching it (compensating the agents which can be disfavored)

Sequencing games

- *Sequencing games* are a class of cooperative games, analyzing the possibilities that various agents have to change their scheduling and so that each of them is better-off
- N is the set of all agents
- Agents of a subset $S \subseteq N$ may form a *coalition*, if they can rearrange themselves so to improve their situation

Coalitions

- A subset $S \subseteq N$ of agents can form a *coalition*, if this leads to an improvement of their situation, regardless of the other agents, $N \setminus S$
- In general, some restrictions apply on how can coalitions be formed and the actions they can take (typically, no harm must be done to the other players)

Value function v

- For each $S \subseteq N$, $v(S)$ is the maximum utility the agents in S can attain by forming a coalition
- $v(N)$ is the overall utility attained if all the agents form a single coalition (*value of the game*)

Example: the gloves

- 4 players: a and b own a left glove each, c and d a right glove each
- $N = \{a,b,c,d\}$
- $v(a) = v(b) = v(c) = v(d) = 0$
- $v(a,b) = v(c,d) = 0$
- $v(S) = 1$ for all other S with 2 or 3 players
- $v(N) = 2$

Allocations

- The value of the game can be *allocated* among the agents according to a vector $x \in R^{|N|}_{+}$ such that

$$\sum_{j \in S} x_j \geq v(S)$$

$$x_i \geq v(i)$$

Allocation and coalitions

- If, for a given allocation x, one has that for *each* $S \subset N$:

$$\sum_{j \in S} x_j \geq v(S)$$

then no proper subset of agents has interest in forming coalitions, but everyone is happy with the "grand coalition" N

Core of a game

- An allocation x of this type belongs to the *core of the game*
- Some games have a nonempty core for any value of the parameters defining the game, others only in some cases

Computing the core

- In order to find, if it exists, an x belonging to the core of a game, one can solve the LP

$$\min \sum_{j \in N} x_j$$

$$\sum_{j \in S} x_j \geq v(S) \qquad \forall S \subset N$$

$$x_j \geq v(j)$$

The gloves (cont.)

- The following x is in the core

  $x = [1/2 \quad 1/2 \quad 1/2 \quad 1/2]$

  as well as

  $x = [2/3 \quad 2/3 \quad 1/3 \quad 1/3]$

  and all vectors $(0 \leq \varepsilon \leq 1)$

  $x = [1-\varepsilon \quad 1-\varepsilon \quad \varepsilon \quad \varepsilon]$

Sequencing games

[Curiel, Pederzoli, Tijs 1989]

- A class of cooperative games
- Given n agents, each holding a set of jobs, an initial sequence σ_0, and a (monetary) objective function for each agent, how will the agents form coalitions to maximize their own profit?

Sequencing games

- Single machine
- Each agent j holds exactly one job
- Each job requires the machine for time p_j
- Initially, the agents are ordered by σ_0
- If $C_j(\sigma_0)$ is the completion time of job j, the agent incurs cost $\alpha_j C_j(\sigma_0)$

Value of the game

- If a supervisor enforces the WSPT sequence σ^*, the maximum overall profit would be attained (*value of the game*)

$$v(N) = \sum_{j \in N} \alpha_j (C_j(\sigma_0) - C_j(\sigma^*))$$

Coalitions

- A subset S of agents can form a coalition, if this leads to an improvement of their situation, regardless of (and with no harm for) the other agents, $N \backslash S$
- A schedule σ is *feasible* for a coalition S (with respect to σ_0) if no agent in $N \backslash S$ is overtaken by an agent in S, i.e., S must be formed by consecutive agents in σ_0

Value of a coalition

- Let Σ_S the set of feasible schedules for the agents in S
- The maximum profit the agents in S can get is

$$v(S) = \max_{\sigma \in \Sigma_S} \{ \sum_{j \in S} \alpha_j (C_j(\sigma_0) - C_j(\sigma)) \}$$

Value of a coalition

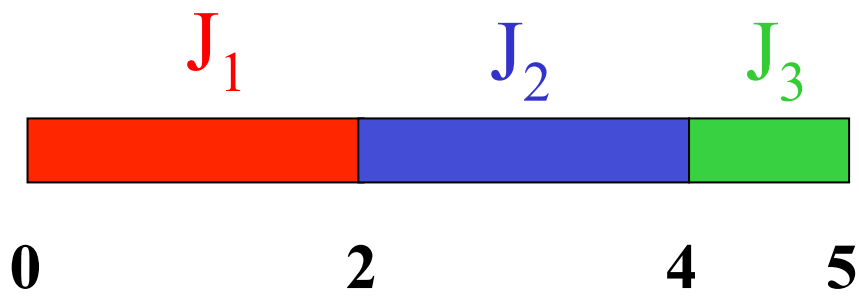
- If j precedes i, and they exchange their position, we get a *saving* g_{ij}

$$g_{ij} = \max\{0, \alpha_j p_i - \alpha_i p_j\}$$

- It is easy to verify that:

$$v(S) = \sum_{i,j \in S : \sigma_0(i) < \sigma_0(j)} g_{ij}$$

Example - schedule σ_0



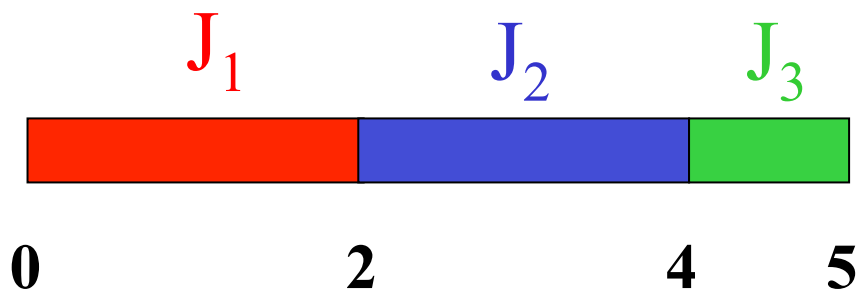
$\alpha_1 = 4$

$\alpha_2 = 6$

$\alpha_3 = 5$

$4*2+6*4+5*5=57$

Example - schedule σ_0



$\alpha_1 = 4$

$\alpha_2 = 6$

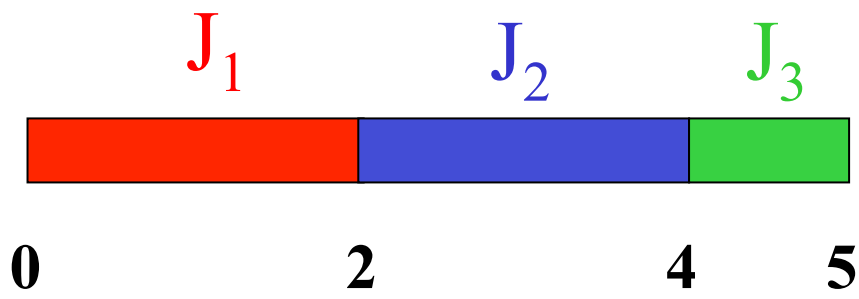
$\alpha_3 = 5$

$g_{12} = \max\{0, 6*2-4*2\} = 4$

$g_{23} = \max\{0, 5*2-6*1\} = 4$

$g_{13} = \max\{0, 5*2-4*1\} = 6$

Example - schedule σ_0



$\alpha_1 = 4$

$\alpha_2 = 6$

$\alpha_3 = 5$

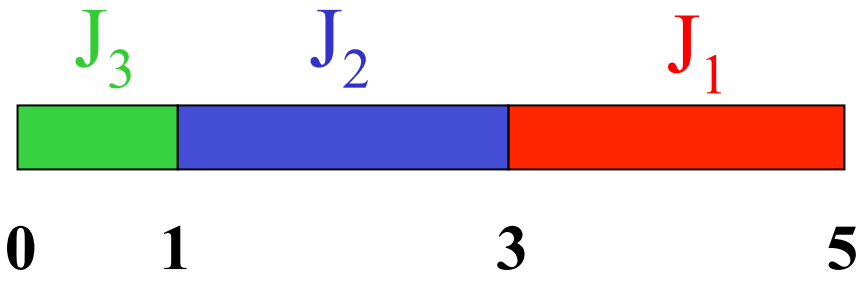
$v(\{1\})=0 \quad v(\{2\})=0 \quad v(\{3\})=0$

$v(\{1,2\})=4 \qquad v(\{2,3\})=4$

$v(\{1,3\}) = v(\{1\})+v(\{3\})=0$

$v(\{1,2,3\})=v(N)= 4+4+6=14$

Schedule σ^*



$\alpha_1 = 4$

$\alpha_2 = 6$

$\alpha_3 = 5$

$5*1+6*3+4*5=43$

$v(N) = 57 - 43 = 14$

Allocations

- The value of the game ($v(N)=14$) can be distributed among the agents by a suitable allocation
- An allocation is defined by a vector $x = (x_1, x_2,\ldots, x_N)$ such that $x_j \geq 0$ and

$$\sum_{j=1}^{N} x_j = v(N)$$

Split core

- In the single-machine problem, consider, for each pair i,j such that $\sigma_0(i)<\sigma_0(j)$, a number λ_{ij} between 0 and 1, and let

$$x_i = \sum_{k:\sigma_0(k)<\sigma_0(i)} (1-\lambda_{ki}) g_{ki} + \sum_{j:\sigma_0(i)<\sigma_0(j)} \lambda_{ij} g_{ij}$$

Split core

- In other words, given the saving g_{ij}, a fraction λ_{ij} is allocated to agent j and the remaining part (1- λ_{ij}) to agent i
- For different values of λ_{ij}, one obtains infinite allocations
- All these allocations belong to the core of the game [Hamers et al. 1996]

Sequencing games: summary of results

- An agent may be overtaken by other agents, if this does not increase its completion time
- Slikker (2006) showed that even in this case the core is nonempty

Sequencing games: summary of results

- Even adding release dates, the core is still nonempty
- The m-machine problem has a nonempty core for $m=2$, while it may have empty core for $m=3$ [Slikker 2003]
- Most results extend to the case of agents owning more than one job, and holding a max-type cost function [Estévez-Fernández, Borm, Calleja and Hamers 2008]

AUTOMATED NEGOTIATION

Automated negotiation [Fink 2007]

- One further scenario is when agents are able to compare solutions, but not to explicitly assign a monetary value to each of them
- No side payments are possible
- *Automated negotiation* is a mechanism that allows agents to interact without disclosing any information and without using monetary evaluations

Automated negotiation

- The coordinator randomly generates a *contract* (solution)
- At each iteration, the coordinator submits a new contract to all the agents
- Each agent decides whether to accept it or not

Automated negotiation

- If (and only if) all agents accept it, the newly submitted contract becomes the current contract
- When a stopping criterion is met, the current contract is taken as final solution
- The contracts are generated in the very same way a neighborhood is explored in a metaheuristic algorithm

Automated negotiation

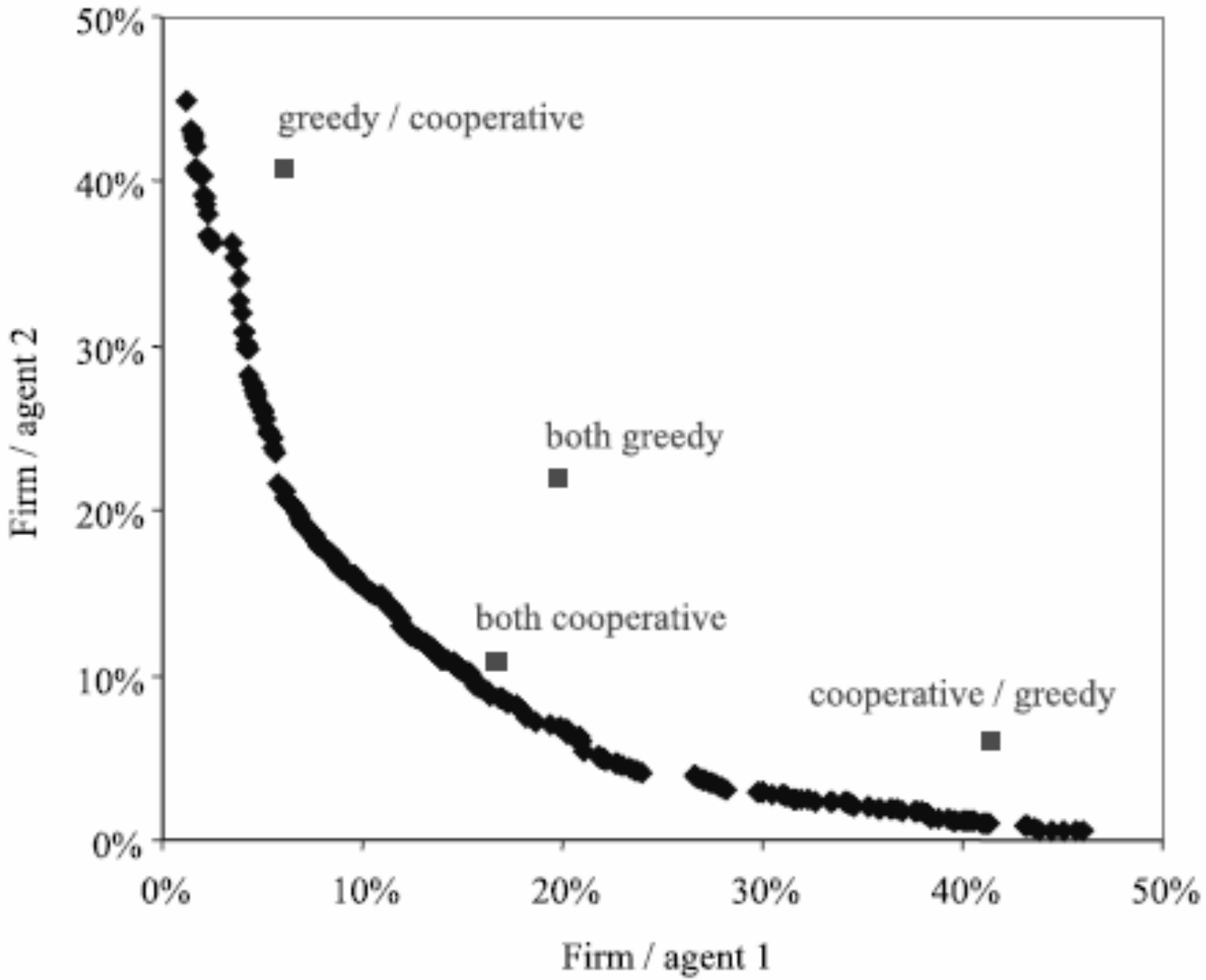
- Fink (2007) presents results on a two-agent problem, in which the agents must agree upon a permutation of the jobs
- Agent 1 owns the first facility, and wants to minimize the total setup cost
- Agent 2 owns the second facility, and wants to minimize the total completion time

Automated negotiation

- In this example, the only quantity the agents must agree upon is a permutation of the jobs
- Therefore, there is no need to disclose further information for the agents to reach an agreement

Greedy vs. cooperative behavior

- If each agent follows a purely greedy strategy, the resulting solution is typically poor
- The coordinator can establish a probabilistic acceptance rate (similar to simulated annealing) for each agent, thus forcing them to partially cooperate
- This leads to improved quality for the final solution


50%
40%
30%
20%
10%
0%
Firm / agent 2
greedy / cooperative
both greedy
both cooperative
cooperative / greedy
0%
10%
20%
30%
40%
50%
Firm / agent 1

Automated negotiation

Pros

- Very flexible and modular
- No real information exchange among the agents (automated generation)
- Fair, no way to “cheat”

Cons

- May not produce a Pareto solution
- Complex to enforce cooperation

Future research

- Devise specific automated negotiation protocols for particular problems, in which partial information has to be disclosed to the coordinator
- Comparison among the solutions produced by different mechanisms and protocols

Conclusions…

- Multi-agent problems are being studied in several, increasingly complex contexts
- While several complexity results have been established, enumeration algorithms, approximation algorithms, fairness and protocols are still largely unexplored
- Possibly, new application areas will benefit from multi-agent models (grids…)