Non-disjoint multi-agent scheduling problem on identical parallel processors

Van Ut TRAN[1,2], Ameur SOUKHAL[1], Tuong Nguyen HUYNH[3],

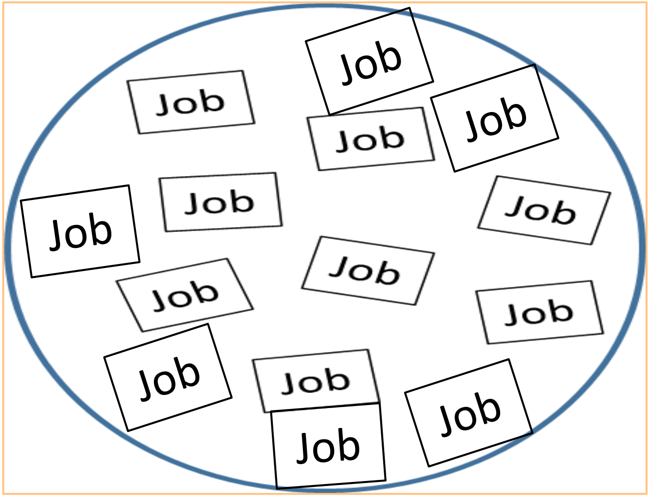
[1]Laboratory of Computer Science, François Rabelais Tours University
[2]Can Tho University of Technology , Vietnam
[3]Ho Chi Minh City University of Technology, Vietnam
vanut.tran@etu.univ-tours.fr

Content of presentation

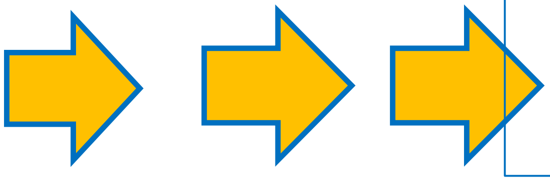
Motivation

Classical Scheduling:

- Set of jobs to be scheduled on one (several) machine(s)
- Each job has a set of characteristics
- One objective function to be optimized



Several models and efficient methods



One Objective function:

Minimize:

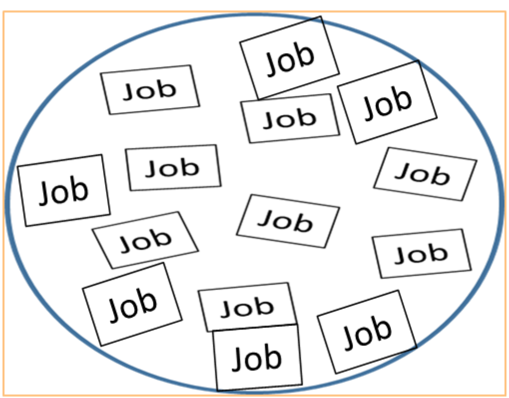
Makespan (C_{max})

Or Tardy job ($\sum Uj$)

Motivation

Multicriteria scheduling problems *(T'kindt & Billaut 2005)*

- Only one objective function is not always sufficient
- Good solutions with respect to one objective may be bad with respect to other objectives
- Finding solutions of good compromise



Several objective functions:

Makespan (C_{max})

And **Tardy job** ($\sum U_j$) **and** …

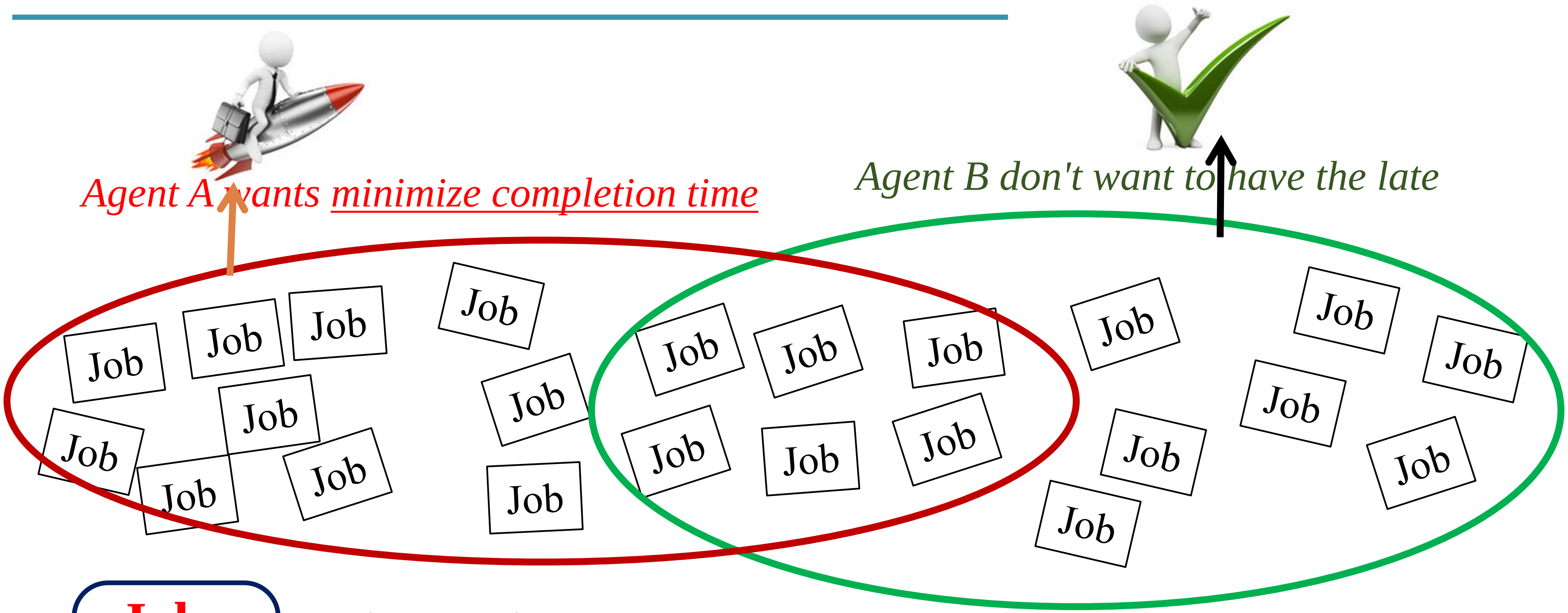
Motivation

Sometimes… Multi-agent Scheduling

- ✓ Jobs are not equivalent and applying the same measure to all jobs is not useful.
- ✓ Each subset of jobs is assessed according to objective function, where jobs are in competition for the use of the machines.
- ✓ This is a multi-criteria, multi-agent scheduling problem, where a new type of compromise has to be obtained.

Problems noted "multi-agent scheduling problems " (Agnetis et al. 2014).

Multi-agent scheduling



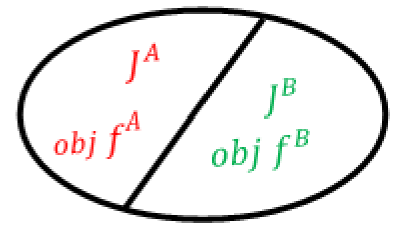
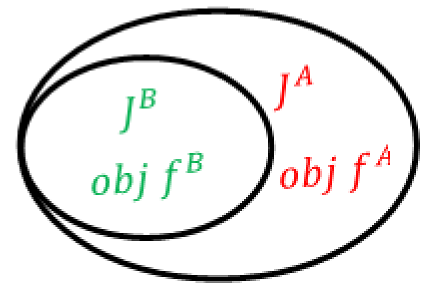
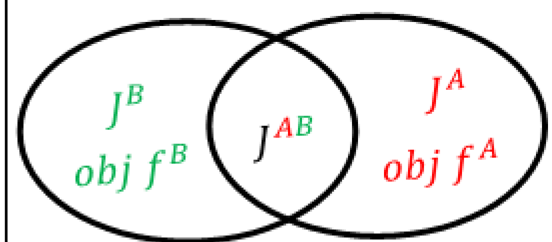
Jobs
Jobs
Jobs

Objective: schedule jobs on machines to

Min(f¹, f²)

Scenarios for multiagent

(Agnetis et al. 2014)

COMPETING (CO)	$J^A \cap J^B = \varnothing$	
INTERFERING (IN)	$J^B \supseteq J^A = J^G$	
NON-DISJOINT (ND)	$J^A \cap J^B \neq \varnothing$ $J^G = J^A \cup J^B$	

Multiagent: definition and notations

The book "multi-agent scheduling problems " (Agnetis et al. 2014)

- ✓ Groups of jobs are identified by their owned 'agents' k, $\forall k \in \{1,2,\ldots K\}$
- ✓ Agent k have the jobs $J_1^k,\ldots,J_{n_k}^k \in J^{\,k}$, objective function $f^{\,k}$, $\forall k \in \{1,2,\ldots K\}$,
- ✓ J_j^k : the job number j of agent k
- ✓ p_j^k : processing time of job number j of agent k
- ✓ d_j^k : due date of job number j of agent k
- ✓ C_j^k : completion time of job number j of agent k
- ✓ C_{max}^k : makespan of agent k
- ✓ U_j^k : number of tardy jobs in position job j of agent k.
- ✓ The tardiness penalty $U_j^k = 0\ if\ \ C_j^k < d_j^k(early),\ \ 1\ otherwise$ (*tardy*)

Problem definition and notations

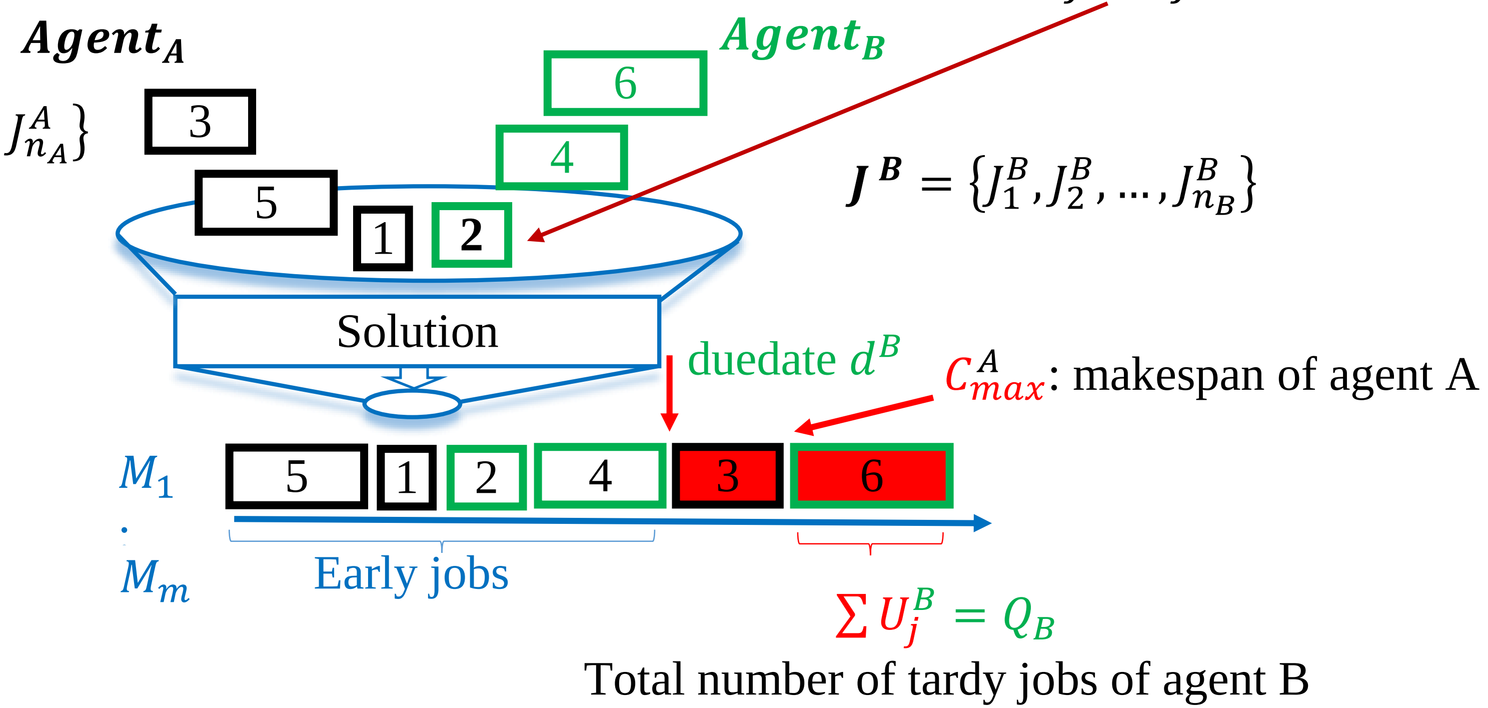
$1(P_m)|ND, d^B, \sum U_j^B \leq 1| \, C_{max}^A$

In case two agents: A and B

In case Non − disjoint: $J^A \cup J^B \neq \emptyset$

$J^A = \{J_1^A, J_2^A, \dots, J_{n_A}^A\}$

$\boldsymbol{J^B} = \{J_1^B, J_2^B, \dots, J_{n_B}^B\}$



The studied problem is denoted by $P_m|ND, d^B, \sum U_j^B \leq Q_B| \, C_{max}^A$.

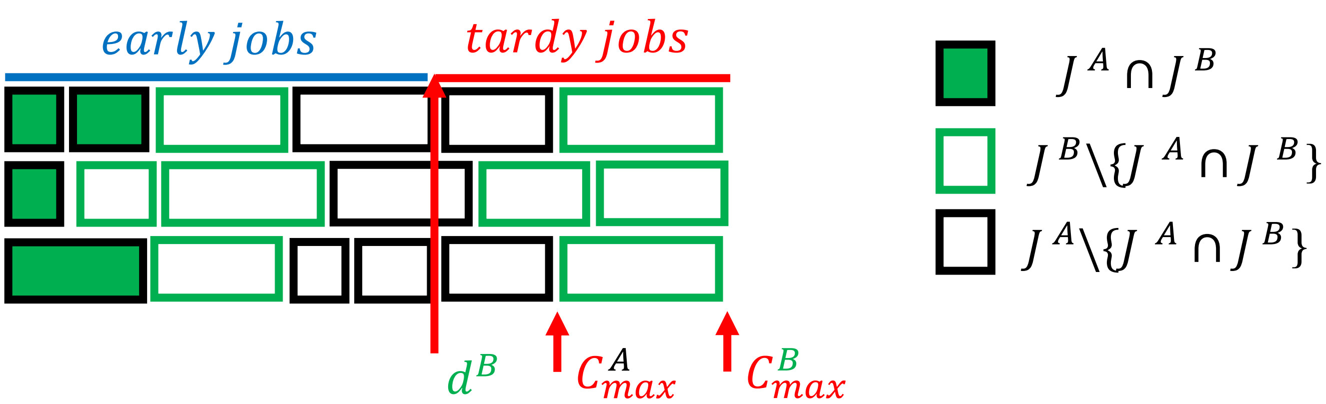
This problem is NP-hard (*Sadi et al. 2014*)

Structure of non-dominated solution

Proposition:

If problem $P_m|ND, d^B, \sum U_j^B \leq Q_B| C_{max}^A$ admits a feasible solution, then it is possible to build an optimal solution such that on each machine we have:

1. *Jobs of agent B appear in SPT order.*
2. *Tardy job J_j within $J^B \backslash \{J^A \cap J^B\}$ is scheduled after the jobs of agent A.*



Integer programming formulation 1

The first one, is based on precedence decision variables

- $x_{i,j}$= 1 if job J_j is scheduled on machine M_i; 0 otherwise.
- $y_{j,k}$ = 1 if job J_j is executed before job J_k on the same machine; 0 otherwise.
- z_j = 1 if job J_j is scheduled after its due date d^B; 0 otherwise.

$$(MILP-Assign) \quad Min \quad C^A_{max}$$

$$s.t.\left\{\begin{array}{llll}
\sum_{i=0}^{m} x_{i,j} & = 1, & \forall J_j \in \mathcal{J} & (1)\\
C_j - \sum_{k=1}^{n} p_k \times y_{j,k} & \geq p_j, & \forall J_j \in \mathcal{J} & (2)\\
y_{j,k} + y_{k,j} & \leq 1, & \forall J_j, J_k \in \mathcal{J} & (3)\\
x_{i,j} + x_{i,k} - y_{j,k} - y_{k,j} & \leq 1, & \forall J_j, J_k \in \mathcal{J} & (4)\\
 & & i = 1, \ldots, m & \\
x_{i,j} + y_{j,k} - x_{i,k} & \leq 1, & \forall J_j \in \mathcal{J} & (5)\\
 & & i = 1, \ldots, m & \\
y_{j,k} + y_{k,l} - y_{j,k} & \leq 1, & \forall J_j, J_k, J_l \in \mathcal{J} & (6)\\
C_j - d_B - HV z_j & \leq 0, & \forall J_j \in \mathcal{J}^B & (7)\\
\sum_{J_j \in \mathcal{J}^B} z_j & \leq Q_B & & (8)\\
x_{i,j}, y_{j,k}, z_j \in \{0,1\}, C_j \geq p_j & & \forall J_j, J_k \in \mathcal{J} & (9)\\
 & & i = 1, \ldots, m &
\end{array}\right.$$

Integer programming formulation 2

The second, is based on time indexing decision variables

❖ $s_{j,t}$ *takes as new binary variables that are time indexed.* $s_{j,t}$ *takes as a value 1 if job* J_j *starts its processing at time t; 0 otherwise.*
Thereby, we have $n \times (T+1)$ *binary variables, which is pseudo polynomial.*

$$(MILP-Time) \quad Min \quad C^A_{max}$$

$$s.t. \begin{cases}
\sum_{t=0}^{T} s_{j,t} & = 1, & \forall J_j \in \mathcal{J}^A & (10)\\
\sum_{J_j \in \mathcal{J}} \sum_{l=\max\{0,t-p_j+1\}}^{t} s_{j,l} & \leq m, & \forall t = 0,\ldots,T & (11)\\
C^A_{max} - \sum_{t=0}^{T-p_j} (t+p_j) s_{j,t} & \geq 0, & \forall J_j \in \mathcal{J}^A & (12)\\
\sum_{t=0}^{T-p_j} (t+p_j) s_{j,t} - HV z_j & \leq d^B, & \forall J_j \in \mathcal{J}^B & (13)\\
\sum_{J_j \in \mathcal{J}^B} z_j & \leq Q_B, & & (14)\\
s_{j,t} \in \{0,1\}, z_j \in \{0,1\}, & & \forall J_j \in \mathcal{J} \quad t = 0,\ldots,T, & (15)
\end{cases}$$

Polynomial heuristic H1

Problem $P_m|ND, d^B, \sum U_j^B \leq Q_B|\, C_{max}^A$ is NP-hard (*Sadi et al. 2014*)

Algorithm 1 LPT-FAM $\quad$ *Complexity*: $O(n_A \log(n_A) + (n_B \log(n_B))$

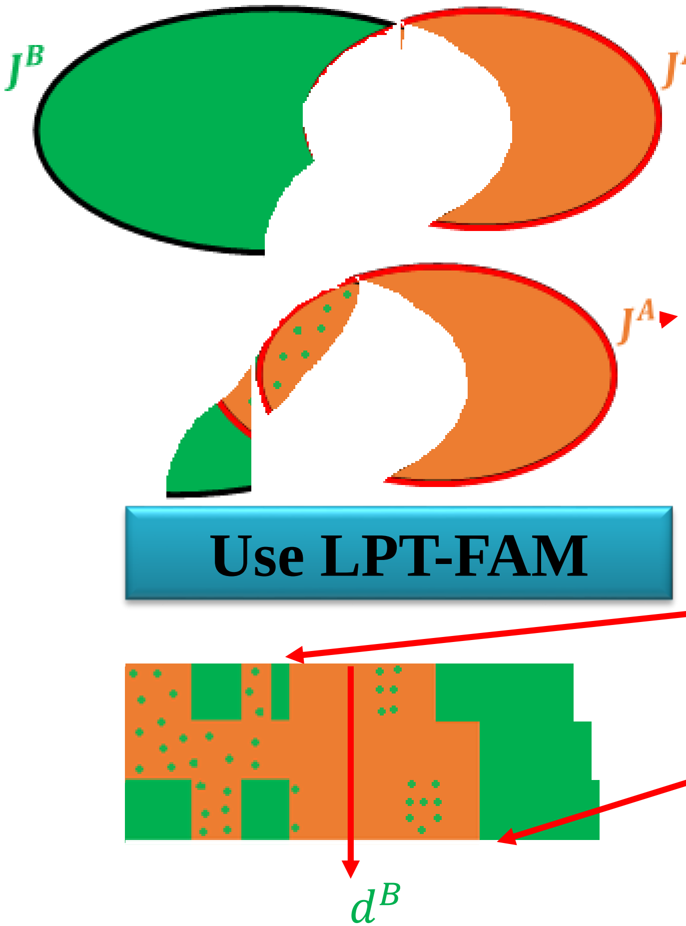
```
Sort jobs in J^B according to SPT rule;
Set E = {J_1^B ..., J_{n_B-Q_B}^B};
Schedule the jobs in E using LPT-FAM;
if at least one job is late then
    Stop; // this heuristic cannot find a feasible solution
else
    Schedule jobs in 𝒥^A\{𝒥^A ∩ E} using LPT-FAM;
    Schedule jobs in 𝒥^B\{𝒥^B ∩ E} using LPT-FAM;
    Return the resulting solution;
```

SPT: *The **S**hortest **P**rocessing **T**ime first*

LPT*: The **L**ongest **P**rocessing **T**ime first*

FAM*: Assign job to the **F**irst **A**vailable **M**achine.*

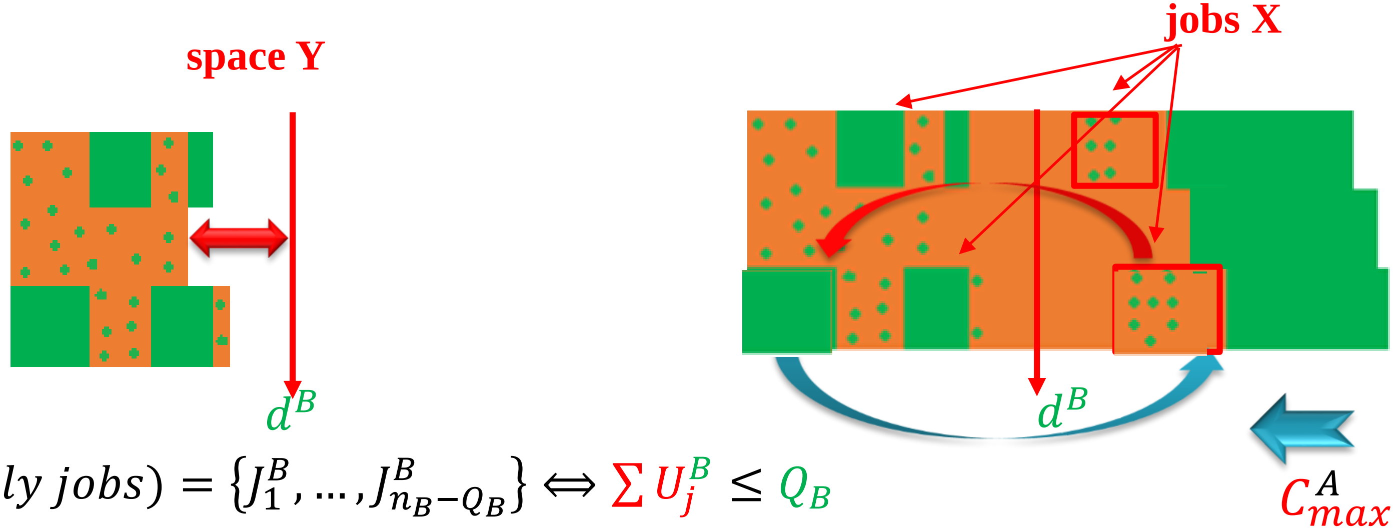
Polynomial heuristic H1



$Set\ E(early\ jobs) = \{J_1^B, \dots, J_{n_B - Q_B}^B\} \Leftrightarrow \sum U_j^B \leq Q_B$

$P_m | ND, d^B, \sum U_j^B \leq Q_B | C_{max}^A$

Polynomial heuristic H1



$Set\ E(early\ jobs) = \{J_1^B, \dots, J_{n_B - Q_B}^B\} \Leftrightarrow \sum U_j^B \leq Q_B$

To improve this heuristic with the main idea:
"we try to swap the jobs if possible"

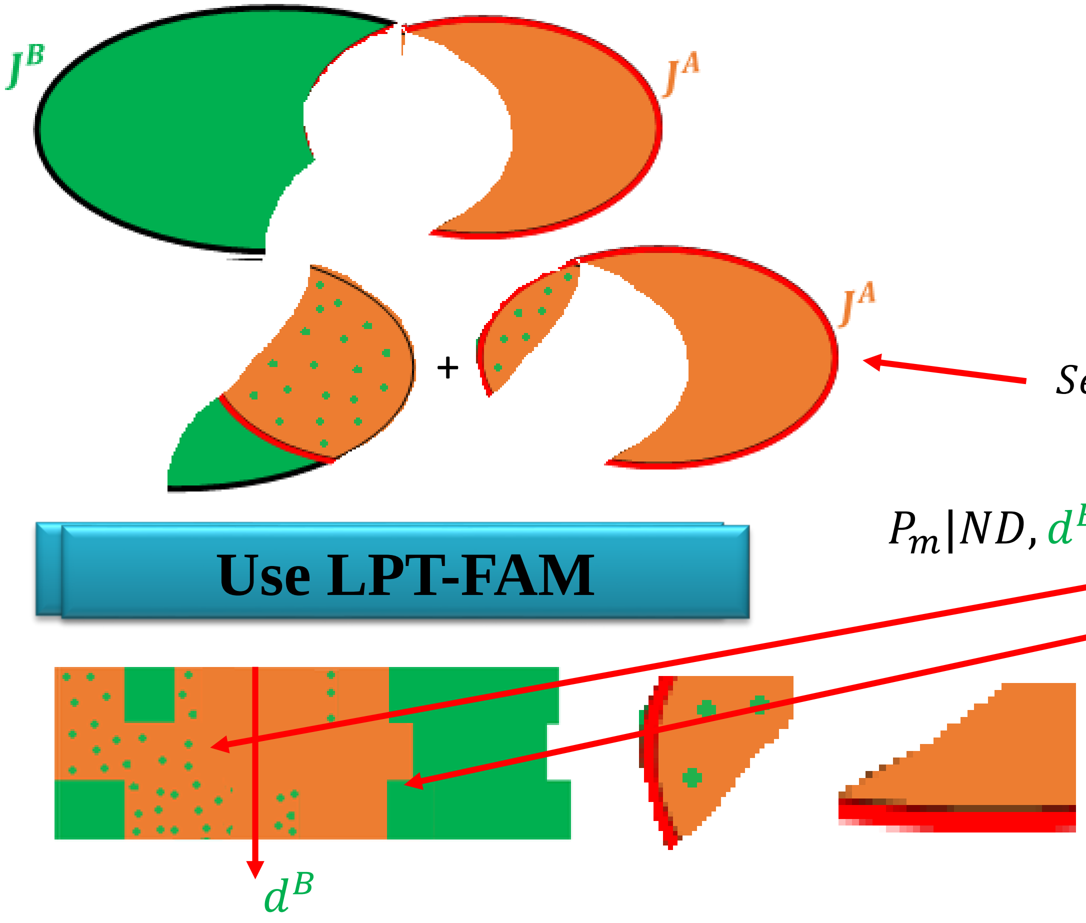
Polynomial heuristic H2

$Complexity: O(n_A \log(n_A) + (n_B \log(n_B))$

Algorithm 2 LPT-FAM with jobs rescheduling

Sort the jobs in J^B according to SPT rule;
Set $E = \{J_1^B \ldots, J_{n_B - Q_B}^B\}$;
Set $S = E \cup \mathcal{J}^A$ in LPT order;
Set $E^B = 0$; // the number of early jobs;
while $S \neq \emptyset$ and $E^B < n_B - Q_B$ **do**
    Schedule J_j using *LPT-FAM*; improve
    if J_j is late **then**
        Remove largest job J_k already scheduled, $J_k \notin E$;
        Put $S = S \backslash \{J_k\}$
        Reschedule J_j using *LPT-FAM*;
    else
        Set $E^B = E^B + 1$;
if $E^B = n_B - Q_B$ **then**
    Schedule jobs of $\mathcal{J}^A$ not already scheduled using *LPT-FAM*;
    Return the resulting solution;
else
    Stop; // This heuristic cannot find a feasible solution;

Polynomial heuristic H2



$Set\ S = \{J_1^B, \dots, J_{n_B - Q_B}^B\} \cup J^A$

$P_m | ND, d^B, \sum U_j^B \leq Q_B | C_{max}^A$

Pseudo-polynomial heuristic

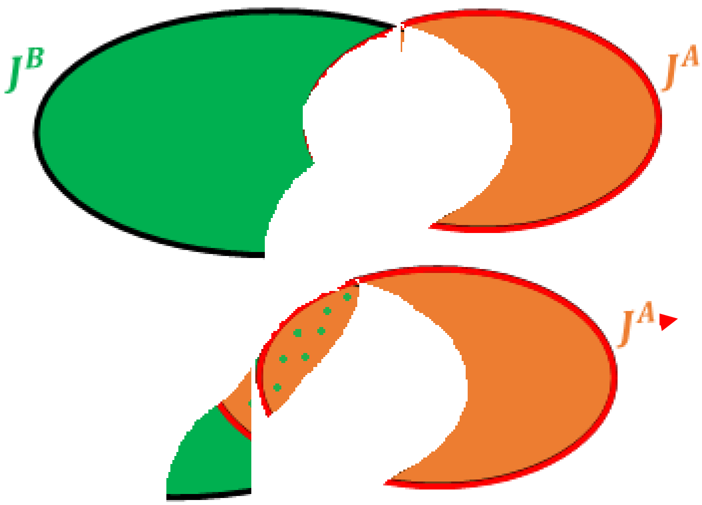
These heurictis base on Dynamic Programming algorithm (DP) proposed to slove optimally classical scheduling problem $P_m||C_{max}$ *(Blaziwicz et al. 2007)*

Algorithm 3 Heuristic based on dynamic programming

1: Sort the jobs in J^B according to SPT rule;
2: Set $E = \{J_1^B \ldots, J_{n_B - Q_B}^B\}$;
3: Optimally solve problem $P2||C_{max}$ considering only jobs of E by the DP
4: **if** $C_{max}(E) > d^B$ **then**
5: Stop; // This problem has no feasible solution;
6: Optimally solve problem $P2||C_{max}$ considering only jobs of $(J^A \backslash \{J^A \cap E\})$ by the DP and taking into account no-availability machines at time zero (jobs of E have been already scheduled)
7: Optimally solve problem $P2||C_{max}$ considering only jobs of $(\mathcal{J}^B \backslash \{\mathcal{J}^A \cup E\})$ by the DP and taking into account no-availability machines at time zero (previous jobs have been already scheduled)
8: Try to schedule tardy jobs of agent B earlier without increasing makespan value by moving them to the left before d^B
9: **Return** the resulting solution;

A solution is given in $O(n^2 + n(UB)^2$, where UB is the upper bound of the makespan of agent A.

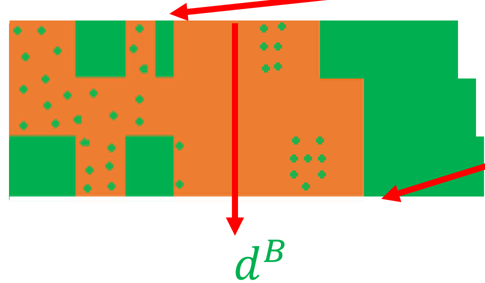
Pseudo-polynomial heuristic H3



$Set\ E(early\ jobs) = \{J_1^B, \ldots, J_{n_B - Q_B}^B\} \Leftrightarrow \sum U_j^B \leq Q_B$

$P_m | ND, d^B, \sum U_j^B \leq Q_B | \, C_{max}^A$

Use Dynamic programming

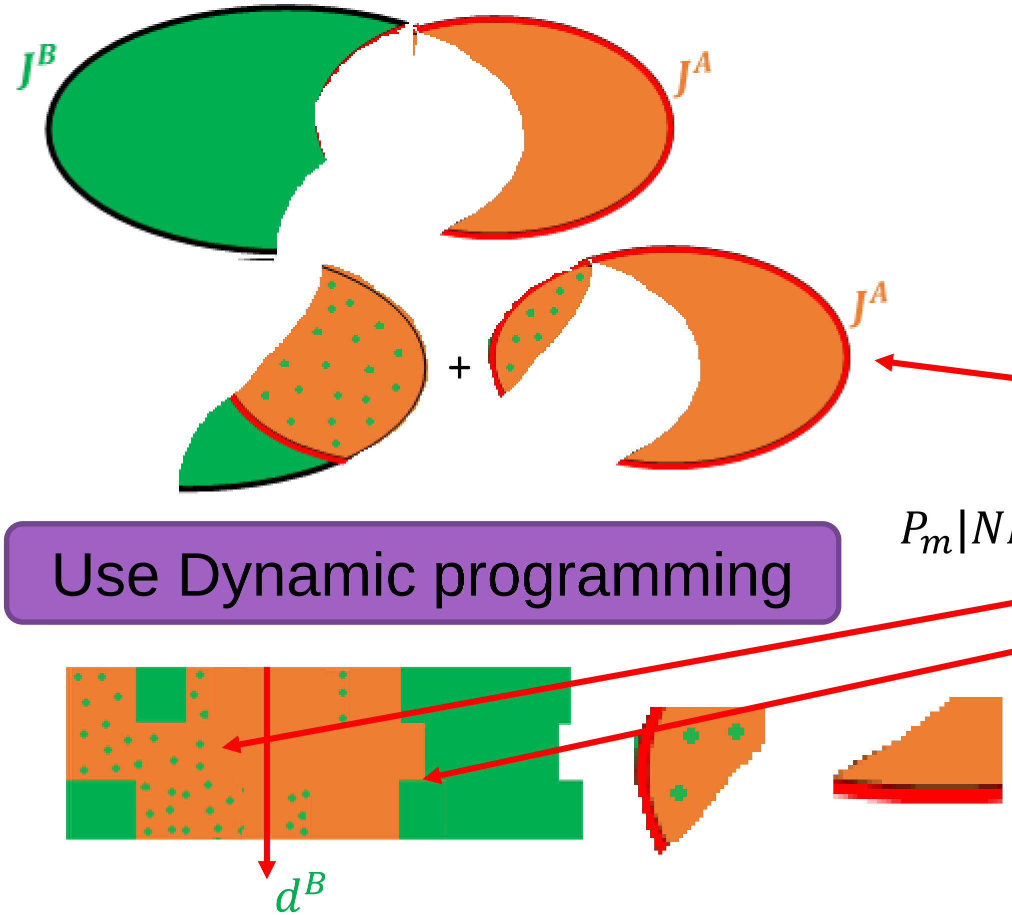


Pseudo-polynomial heuristic H4

Algorithm 4 LPT-FAM-Dynamic programming $Complexity: O(nlog(n) + n(UB)^2$

Sort the jobs in J^B in SPT order;
Set $E = \{J_1^B \ldots, J_{n_B - Q_B}^B\}$;
Set $S = E \cup \mathcal{J}^A$ in LPT order;
Set $E^B = 0$; // the number of early jobs;
while $S \neq \emptyset$ and $E^B < n_B - Q_B$ **do**
Schedule J_j using $LPT\text{-}FAM$;
if J_j is late **then**
Remove J_k the largest job already scheduled;
Put $S = S \backslash \{J_k\}$
Reschedule J_j using $LPT\text{-}FAM$;
else
Set $E^B = E^B + 1$;
if $E^B = n_B - Q_B$ **then**
if some jobs of E^B are late **then**
Stop; // This problem has no feasible solution;
else
Use DP to schedule the jobs of $\mathcal{J}^A$ not already scheduled;
Use DP to schedule the jobs of $\mathcal{J}^B$ not already scheduled;
Return the resulting solution;

Polynomial heuristic H4



$Set\ S = \{J_1^B, \dots, J_{n_B - Q_B}^B\} \cup J^A$

Use Dynamic programming

$P_m | ND, d^B, \sum U_j^B \leq Q_B | C_{max}^A$

Computational results

- ✓ $n \in \{10,20,30,40,50,60,70\}$
- ✓ Fixe time limit is 1h for each value Q
- ✓ M = 2 identical machines
- ✓ 30 instances are generated for each n
- ✓ For each instance, the jobs are assigned randomly to the agents
  - $a_j = 1 \; if J_j \in J^{B} \backslash \{J^{A} \cup J^{B}\}$
  - $a_j = 2 \; if J_j \in \{J^{A} \cup J^{B}\}$
  - $a_j = 3 \; if J_j \in J^{A} \backslash \{J^{A} \cup J^{B}\}$
  - Processing time $p_j^k = \{1; 10\}$

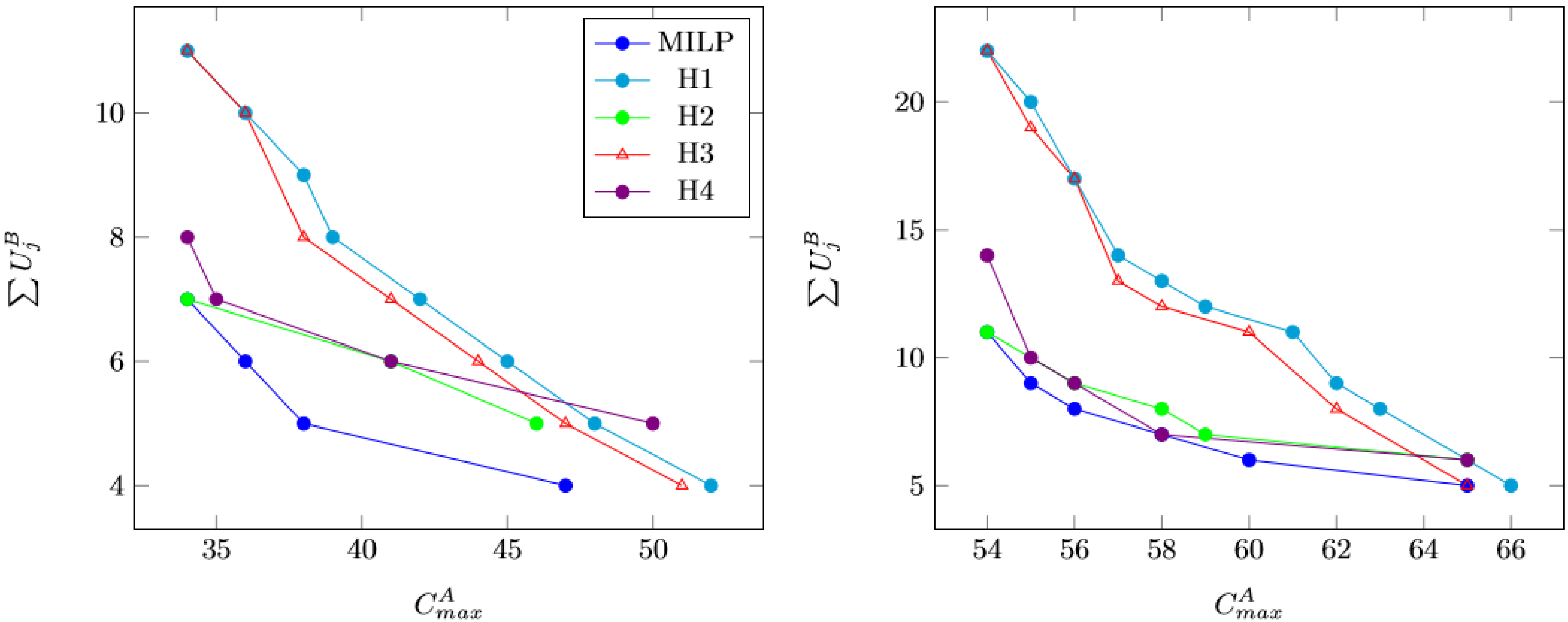
	MILP-Time		*MILP-Assign*	
n	CPU	$\lvert S^*\rvert$	CPU	$\lvert S^*\rvert$
10	0.01	2.37	1.281	2.37
20	0.69	4.07	708.39	4.07
30	2.65	4.87	-	-
40	12.20	6.13	-	-
50	78.00	7.50	-	-
70	4664.16	9.766		

Table 1: Comparison of the performances of the MILPs.

|S|: the cardinality of the exact Pareto front*

Coded with C, Cplex 12.6.2, run in CPU Intel Core i5 2.4Ghz 8GB RAM
The time indexed formulation is better than the assigned formulation, since its solves instances with 70 jobs in 1 hour and 18 minutes on average

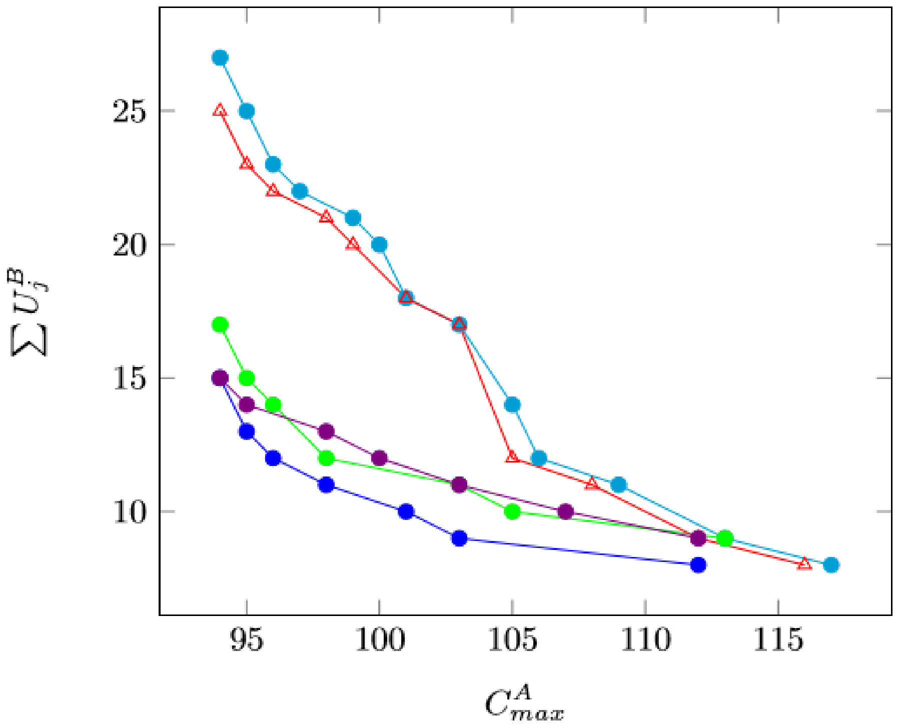
Computational results



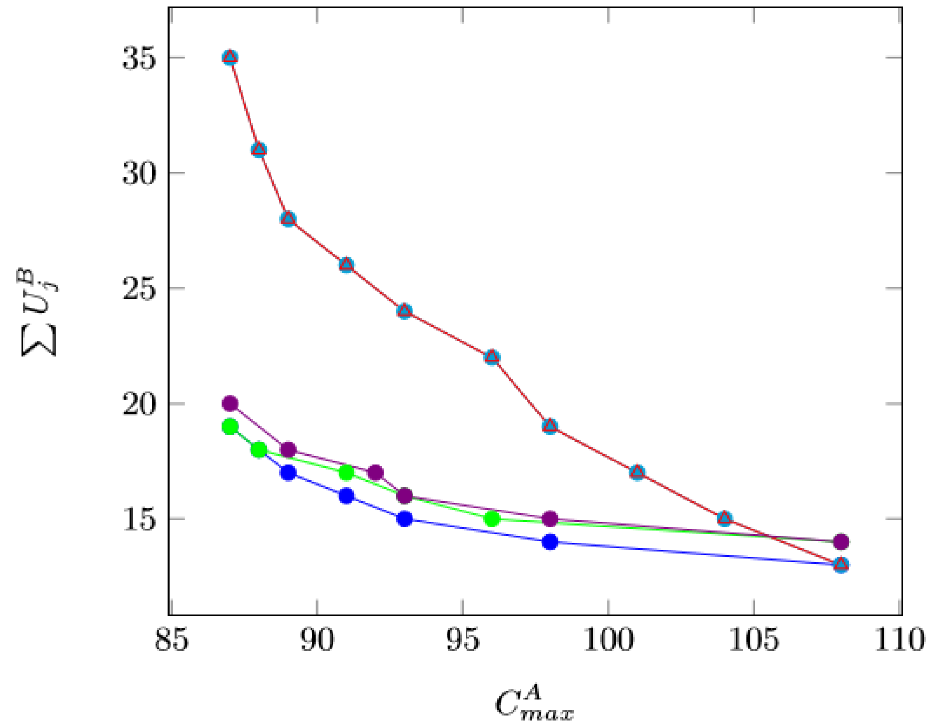
(a) 20 jobs (b) 30 jobs

Example of the obtained Pareto fronts with instances with 20, 30 jobs.

Computational results



(c) 40 jobs



(d) 50 jobs

Example of the obtained Pareto fronts with instances with 40, 50

Computational results

|S|: (the cardinality of the near Pareto front)

	H1		H2		H3		H4	
n	CPU	$\|S\|$	CPU	$\|S\|$	CPU	$\|S\|$	CPU	$\|S\|$
10	0.00	2.87	0.00	2.43	0.000	2.97	0.000	2.53
20	0.00	5.63	0.00	4.07	0.000	5.40	0.000	4.03
30	0.00	7.50	0.00	4.90	0.001	7.13	0.000	4.87
40	0.00	9.67	0.00	6.20	0.002	9.30	0.001	5.97
50	0.00	11.53	0.00	7.27	0.006	11.40	0.003	6.77
70	0.00	15.23	0.00	9.60	0.019	15.13	0.005	8.63

Table 2: Performance comparison using the CPU and $|S|$.

Coded with Python 3.5, Cplex 12.6.3 and run in CPU Intel Core i5 2.4Ghz 8GB RAM
CPU'time of H1 and H2 always nearly zero.
CPU'time of H4 < H3, because UB(upper bound) of H4 < H3 (LPT-FAM improve).
Value |S| : H1 > H3 > H2 > H4 (H4 use exact method)

Computational results

	H1			H2			H3			H4		
n	$\%S$	GD	$\mathcal{H}$	$\%S$	GD	$\mathcal{H}$	$\%S$	GD	$\mathcal{H}$	$\%S$	GD	$\mathcal{H}$
10	29.83	0.86	24.21	36.94	0.68	17.90	33.28	0.82	25.66	28.33	0.89	21.11
20	14.40	1.39	37.39	28.85	0.91	11.52	18.85	1.39	36.31	12.08	1.24	19.95
30	6.09	1.86	40.94	25.19	1.06	10.37	7.08	1.89	40.54	9.82	1.44	21.23
40	1.78	2.02	41.13	27.04	1.12	7.86	1.84	2.06	41.58	9.66	1.43	20.07
50	1.01	2.47	44.52	37.08	1.01	5.65	1.21	2.48	44.50	14.50	1.53	17.69
70	0.70	3.09	45.30	35.74	0.96	4.77	0.70	3.10	45.29	14.13	1.39	10.51

Table 3: Performance comparison using the $\%S$, GD and $\mathcal{H}$.

- **%S**: this metric calculates the number of exact solutions generated given by $|S \cap S*|/|S|$.
- **GD**: generational distance.
- **H**: Hypervolume calculates the area dominated by some front.

Conclusions and Perspectives

- Multi-agent scheduling problems with common due date to minimize both tardy jobs and makespan: $P_m|ND, d^B, \sum U_j^B \leq Q_B| C_{max}^A$
  - Two types of mathematical programming formulation based on : precedence and time indexing decision variables.
  - Proposed four heuristics for this NP-hard problem:
    - ✓ Polynomial heuristics: Algorithm H1, H2
    - ✓ Pseudo-polynomial heuristics: Algorithm H3, H4 base on dymamic programming
- Perspectives :

  For further research, we will propose a genetic algorithm starting from the solutions obtained by the heuristics. It would be also interesting to seek for a pseudo-polynomial time algorithm.

Thank you
for your attention!