Time-indexed formulations for earliness-tardiness scheduling

Francis Sourd

LIP6
CNRS – Université Paris 6

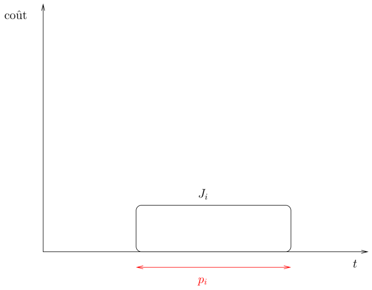
Séminaire ID
Grenoble – November 23rd 2006

Ordonnancement Avance-Retard

- n **tâches à ordonnancer sur une machine**
- Chaque tâche J_i a une durée p_i
- Chaque tâche J_i a une date d'échéance d_i
- Objectif : déterminer les dates d'exécution C_i des tâches afin d'optimiser le critère avance-retard; pénalités par unité de temps: α_i et β_i

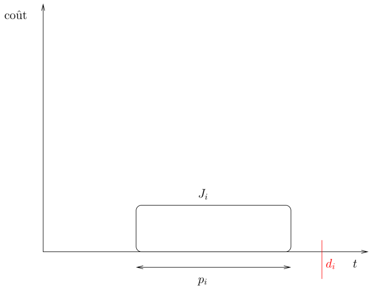
Modélisation Juste-à-Temps

- n tâches à ordonnancer sur une machine
- **Chaque tâche J_i a une durée p_i**
- Chaque tâche J_i a une date d'échéance d_i
- Objectif : déterminer les dates d'exécution C_i des tâches afin d'optimiser le critère avance-retard; pénalités par unité de temps: α_i et β_i



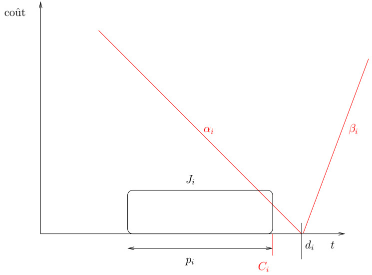
Modélisation Juste-à-Temps

- n tâches à ordonnancer sur une machine
- Chaque tâche J_i a une durée p_i
- **Chaque tâche J_i a une date d'échéance d_i**
- Objectif : déterminer les dates d'exécution C_i des tâches afin d'optimiser le critère avance-retard; pénalités par unité de temps: α_i et β_i



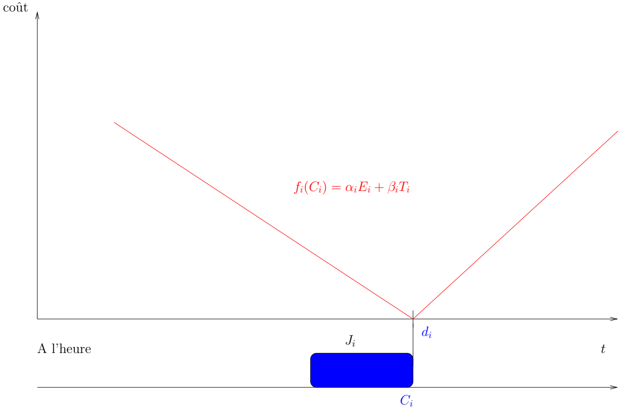
Modélisation Juste-à-Temps

- n tâches à ordonnancer sur une machine
- Chaque tâche J_i a une durée p_i
- Chaque tâche J_i a une date d'échéance d_i
- **Objectif : déterminer les dates d'exécution C_i des tâches afin d'optimiser le critère avance-retard; pénalités par unité de temps: α_i et β_i**



Tâche à l'heure

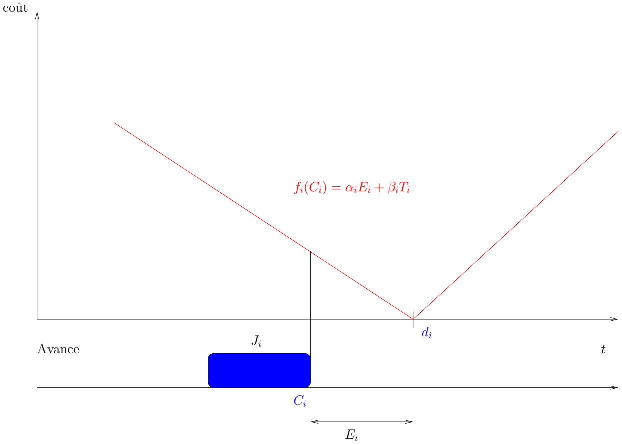
La tâche J_i est à l'heure $C_i = d_i$
Coûts provoqués par la tâche i: 0



Tâche en avance

La tâche J_i est en avance $E_i = d_i - C_i$

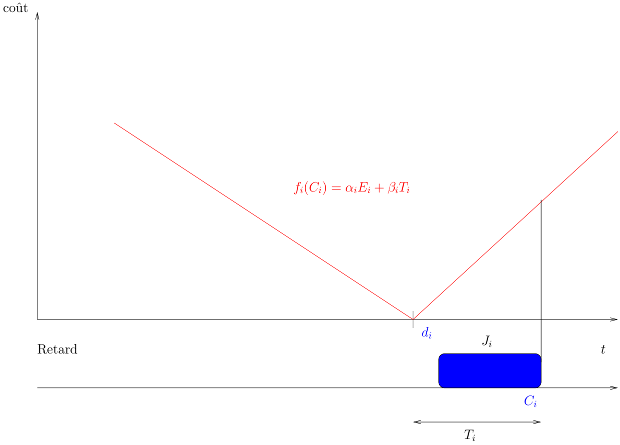
Coûts provoqués par la tâche i: $\alpha_i E_i$



Tâche en retard

La tâche J_i est en retard $T_i = C_i - d_i$

Coûts provoqués par la tâche i: $\beta_i T_i$



One-machine problem with completion costs

- n jobs and one machine with a time horizon T
  - processing time p_i
  - cost c_{it} of job i if it completes at t.
  - earliness-tardiness case: $c_{it} = f_i(t)$
- Size of the input is $O(nT)$
- Find a **one-machine schedule** that minimizes the total cost.

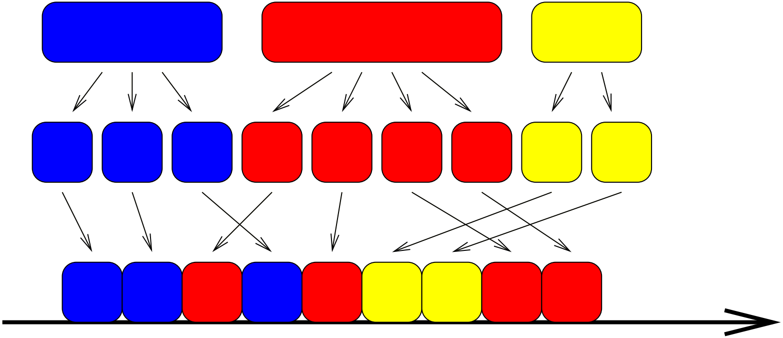
Theoretical results

- **NP-complete** even if $\alpha_i = 0$
- Polynomial cases
  - $p_i = p$, $\alpha_i = \alpha$ and $\beta_i = \beta$
    - Garey, Tarjan and Wilfong (1988)
    - Verma and Dessouky (1998)
  - Large common due date and $\alpha_i = \alpha$ and $\beta_i = \beta$
    - Kanet (1981)
    - Hall and Posner (1991)
  - Sequenced tasks $C_1 < C_2 < \cdots < C_n$
    - Garey, Tarjan and Wilfong (1988)
    - Sourd (2005) for non-convex piecewise linear cost functions

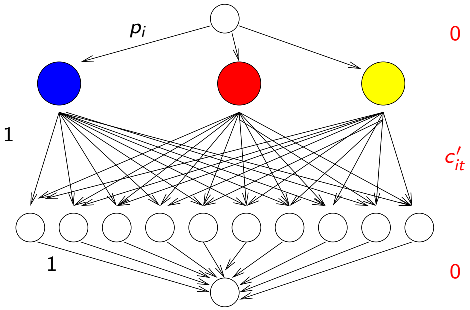
Main lower bounds

- Unsuccessful combinatorial lower bounds
- Linear Programming based lower bounds
  - $x_{it} = 1$ when J_i completes at t
    - Relaxing the resource constraint
    - Relaxing the number of occurence of a job
  - $y_{it} = 1$ when J_i is in process at t
    - Preemptive lower bound
    - Transportation problem - Pseudopolynomial
    - Continuous variant - Polynomial but slow convergence

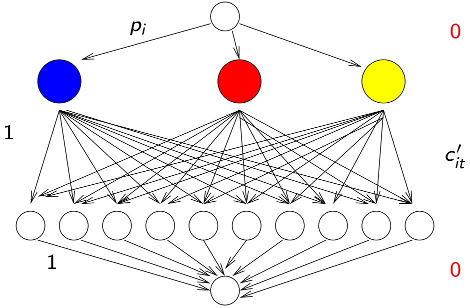
Assignment-based lower bound



Assignment through a network flow problem



Assignment through a network flow problem

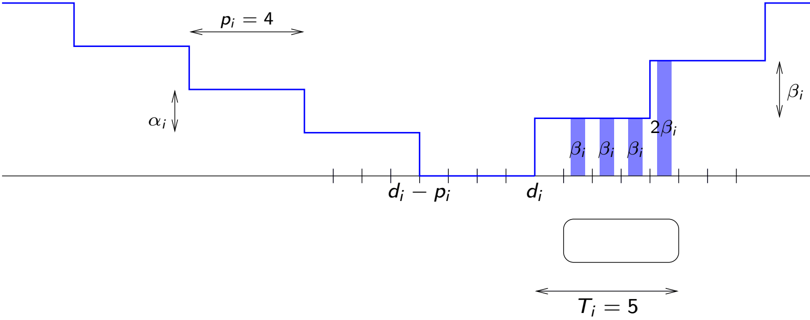


Assignment costs

Assignment costs c'_{it} have to be defined so that we have a **lower bound**.

Defining assignment costs

Sourd and Kedad-Sidhoum (J. Sched., 2003)



$$\sum_{t'=t-p_i+1}^{t} c'_{it} \leq c_{it}$$

Solving the assignment problem

- Number of time points?
  - Time horizon $T = \max d_i + \sum p_i$
  - Pseudo-polynomial w.r.t. the input
- $O(nT)$ assignment arcs
- $n << T$: unbalanced assignment
- $O(n^2 T)$ algorithms instead of $O(T^3)$
- Polynomial continuous variant *[Sourd, INFORMS JoC, 2004]*

IP with end time variables

- $x_{it} = 1$ when J_i completes at time t

$$
\begin{aligned}
\min \quad & \sum_j \sum_{t=p_j}^{T} c_{jt} x_{jt} \\
\text{s.t.} \quad & \sum_{t=p_j}^{T} x_{jt} = 1 && \forall j \\
& \sum_j \sum_{s=t}^{t+p_j} x_{js} \leq 1 && \forall t \\
& x_{jt} \in \{0,1\} && \forall j, \forall t \in [p_j, T]
\end{aligned}
$$

IP with end time variables

- $x_{it} = 1$ when J_i completes at time t

$$
\begin{aligned}
\text{min} \quad & \textstyle\sum_j \sum_{t=p_j}^{T} c_{jt} x_{jt} \\
\text{s.t.} \quad & \textstyle\sum_{t=p_j}^{T} x_{jt} = 1 && \forall j \\
& \textstyle\sum_j \sum_{s=t}^{t+p_j} x_{js} \leq 1 && \forall t \\
& x_{jt} \in \{0, 1\} && \forall j, \forall t \in [p_j, T]
\end{aligned}
$$

- Continuous relaxation
  - Very good lower bound
  - Very large LP. Column generation.

IP with end time variables

- $x_{it} = 1$ when J_i completes at time t

$$
\begin{array}{llll}
\min & \sum_j \sum_{t=p_j}^{T} c_{jt} x_{jt} & & \\
\text{s.t.} & \sum_{t=p_j}^{T} x_{jt} = 1 & \forall j & \times \mu_j \\
 & \sum_j \sum_{s=t}^{t+p_j} x_{js} \le 1 & \forall t & \\
 & x_{jt} \in \{0,1\} & \forall j, \forall t \in [p_j, T] &
\end{array}
$$

- Continuous relaxation
  - Very good lower bound
  - Very large LP. Column generation.
- Lagrangean relaxation
  - of the number of occurences *[Péridy, Pinson and Rivreau, EJOR, 2003]*
  - of the capacity constraints *[Fisher, Math. Prog., 1976]*

IP with end time variables

- $x_{it} = 1$ when J_i completes at time t

$$\begin{array}{lll} \min & \sum_j \sum_{t=p_j}^{T} c_{jt} x_{jt} & \\ \text{s.t.} & \sum_{t=p_j}^{T} x_{jt} = 1 & \forall j \\ & \sum_j \sum_{s=t}^{t+p_j} x_{js} \leq 1 & \forall t \\ & x_{jt} \in \{0,1\} & \forall j, \forall t \in [p_j, T] \end{array}$$

- Continuous relaxation
  - Very good lower bound
  - Very large LP. Column generation.
- Lagrangean relaxation
  - of the number of occurences *[Péridy, Pinson and Rivreau, EJOR, 2003]*
  - of the capacity constraints *[Fisher, Math. Prog., 1976]*
  - **Integrity property**

Relaxing the number of occurences

$$
\begin{array}{lll}
\min & \sum_{jt} c_{jt} x_{jt} & \\
\text{s.t.} & \sum_{t} x_{jt} = 1 & \forall j \\
& \sum_{j\, s \in [t-p_j, t]} x_{js} \le m & \forall t \\
& x_{jt} \le 1 & \forall j\ t
\end{array}
$$

Relaxing the number of occurences

$$
\begin{array}{llll}
\min & \sum_{jt} c_{jt} x_{jt} & & \\
\text{s.t.} & \sum_{t} x_{jt} = 1 & \forall j & \times \mu_j \\
& \sum_{j} \sum_{s \in [t-p_j, t]} x_{js} \leq m & \forall t & \\
& x_{jt} \leq 1 & \forall j\ t &
\end{array}
$$

Relaxing the number of occurences

$$
\begin{aligned}
L_{occ}(\mu) = \min \quad & \textstyle\sum_{jt} (c_{jt} - \mu_j) x_{jt} + \mu_j \\
\text{s.t.} \quad & \\
& \textstyle\sum_{j} \sum_{s \in [t-p_j, t]} x_{js} \leq m \quad \forall t \\
& x_{jt} \leq 1 \quad \forall j\, t
\end{aligned}
$$

Relaxing the number of occurences

$$L_{occ}(\mu) = \min \quad \sum_{jt} (c_{jt} - \mu_j) x_{jt} + \mu_j$$

$$\text{s.t.}$$

$$\sum_{j\, s \in [t-p_j, t]} x_{js} \leq m \quad \forall t$$

$$x_{jt} \leq 1 \quad \forall j\, t$$

Optimization of the Lagrangean problem

- $L_{occ}(\mu)$ can be computed as a shortest path problem.
- $\max_\mu L_{occ}(\mu)$ is a lower bound.
- L_{occ} is a concave non-smooth function.

Relationship between the two models

Pan & Shi, Math. Prog., 2006.

- The assignment-based LB is weaker than the linear relaxation of the end-time based LB.
- Assignment costs are free subject to $\sum_{t'=t-p_i+1}^{t} c'_{it} \leq c_{it}$.
- Optimizing the choice of c'_{it} gives an assignment LB equal to the end-time LB.

IP formulation of the problem

- $x_{it} = 1$ when J_i completes at time t

$$\begin{array}{lll} \min & \sum_j \sum_{t=p_j}^{T} c_{jt} x_{jt} & \\ \text{s.t.} & \sum_{t=p_j}^{T} x_{jt} = 1 & \forall j \\ & \sum_j \sum_{s=t}^{t+p_j} x_{js} \leq 1 & \forall t \\ & x_{jt} \in \{0,1\} & \forall j, \forall t \in [p_j, T] \end{array}$$

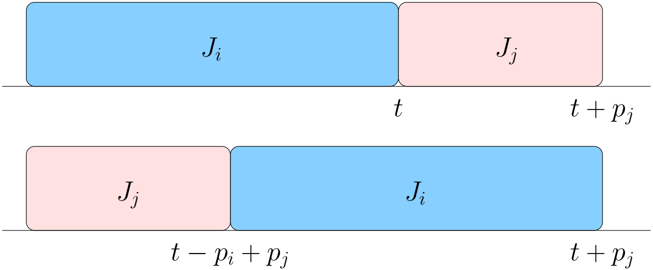
IP formulation of the problem

- $x_{it} = 1$ when J_i completes at time t

$$
\begin{array}{llll}
\min & \sum_j \sum_{t=p_j}^{T} c_{jt} x_{jt} & & \\
\text{s.t.} & \sum_{t=p_j}^{T} x_{jt} = 1 & \forall j & \times \lambda_j \\
& \sum_j \sum_{s=t}^{t+p_j} x_{js} \leq 1 & \forall t & \\
& x_{jt} \in \{0,1\} & \forall j, \forall t \in [p_j, T] &
\end{array}
$$

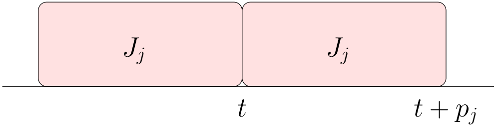
- Our approach:
  - Lagrangean relaxation of the number of occurences
  - Péridy, Pinson and Rivreau (EJOR, 2003)
  - Improving this lower bound even with greater CPU time.

Valid Cut: Swap



$$x_{it} + x_{j,t+p_j} \leq 1 \qquad \text{if } \begin{cases} c_{it} + c_{j,t+p_j} > c_{j,t+p_j-p_i} + c_{i,t+p_j} \\ c_{it} + c_{j,t+p_j} = c_{j,t+p_j-p_i} + c_{i,t+p_j} \\ \qquad \text{and } i \geq j \end{cases}$$

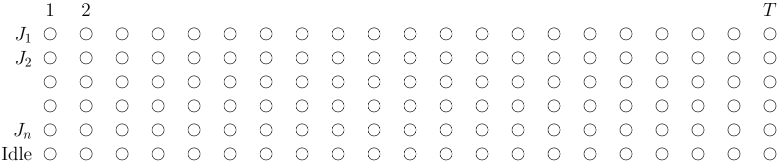
Valid Cut: job repetition



$$x_{jt} + x_{j,t+p_j} \leq 1 \qquad \forall j \, \forall t$$

Lagrangean subproblem

- Each x_{it} is represented by one node



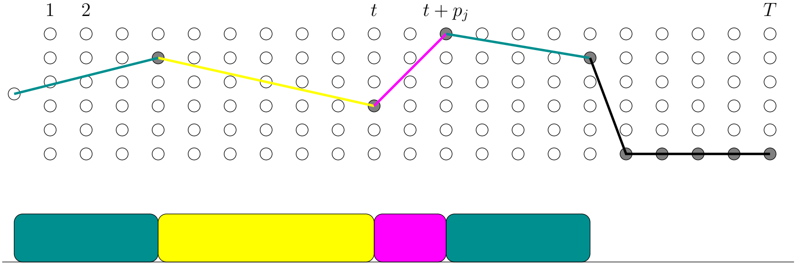
Lagrangean subproblem

Time-indexed formulations for E/T scheduling

Francis Sourd

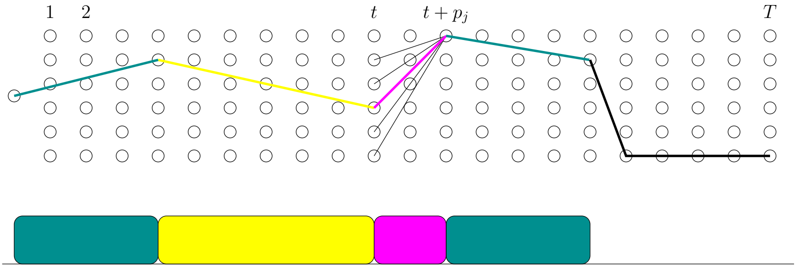
Introduction
Problem definition
Polynomial cases
Lower bounds
IP with assignment variables
IP with end time variables
Relationship between the two models
Improved lower bound
Cut of dominated solutions
Lagrangean subproblem
Applications
Common due date
General due dates

- Each x_{it} is represented by one node
- A solution of the Lagrangean subproblem is a path that traverses the nodes with $x_{it} = 1 \rightarrow$ **pseudo-schedule**



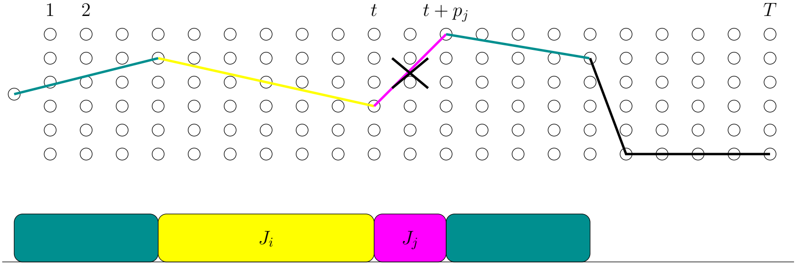
Lagrangean subproblem

- Each x_{it} is represented by one node
- A solution of the Lagrangean subproblem is a path that traverses the nodes with $x_{it} = 1 \rightarrow$ **pseudo-schedule**
- Arcs $(i, t) \rightarrow (i, t + p_j)$ with cost $c_{j,t+p_j} - \lambda_j$
- $O(nT)$ nodes and $O(n^2 T)$ arcs



Cuts in the Lagrangean subproblems

- Assume we have the cut $x_{it} + x_{j,t+p_j} \leq 1$
- Arc $(i, t) \rightarrow (j, t + p_j)$ is removed.



Computing the lower bound

- For some λ the shortest path in the graph gives a lower bound.
  - Computed in $O(n^2 T)$ time
- Multipliers λ are to be adjusted
  - to maximize the lower bound
  - subgradient method / SolvOpt

Computing the lower bound

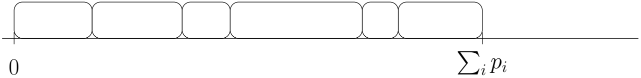
- For some λ the shortest path in the graph gives a lower bound.
  - Computed in $O(n^2T)$ time
- Multipliers λ are to be adjusted
  - to maximize the lower bound
  - subgradient method / SolvOpt
- **Speed up**: Arcs can be removed using reduced costs and the upper bound.
- Very efficient in practice.

Earliness-tardiness common due date problem

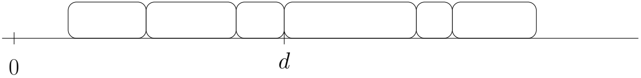
$$d_i = d$$

Properties of the dominating schedules

- either



- or



Lower bound

- A lower bound for each case, we keep the min of both bounds.
- In each case, the graph of the Lagrangean subproblem is simplified.
- With these simplification the Lagrangean problem can be solved
  - by dynamic programming
  - in $O(nT)$.
- Similar to the approach of van den Akker et al (2002) (which only consider the case where $d \geq \sum_i p_i$.

Instances

- Instances by Biskup and Feldmann (2001)
- Available at OR-Library (J.E. Beasley)
- $n = 50$, 100, 200, 500 and 1000 jobs
- Processing times of at most 20 units
- More or less restrictive due dates (factor h)

$$d = \left\lfloor h \sum_i p_i \right\rfloor$$

.

- 280 instances

Results

- The 280 instances are **all** solved...
- ...without any branching!
- Computational times are significantly faster than the approach of van den Akker *et al* (although d is **not** large).

n	$h = 0.4$			$h = 0.6$		
	% solved	Avg time	Max time	% solved	Avg time	Max time
50	100%	0.15	0,28	100%	0.14	0.21
100	100%	0.85	0.99	100%	1.08	1.92
200	100%	7.19	8.72	100%	7.83	10.9
500	100%	90.4	105	100%	98.9	139
1000	100%	794	1027	100%	915	1321

Distinct due dates

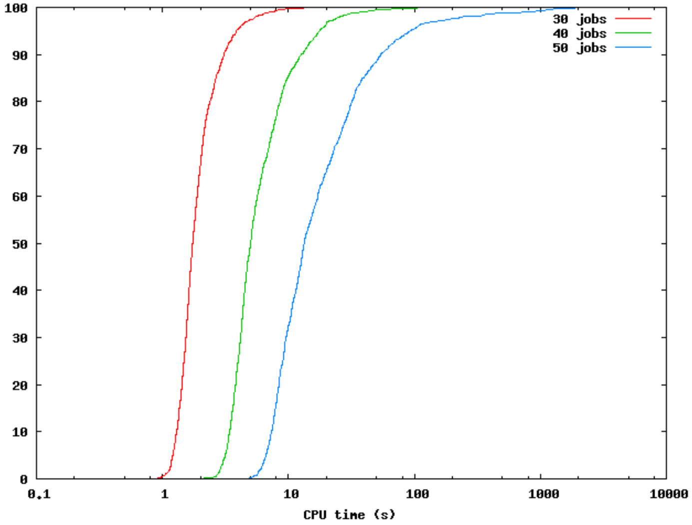
Algorithm

- Branching scheme
  - Pseudo-schedule is usually **not** a schedule
  - Reparing the violated constraints
  - **Example**:
    - A job J_i processed several time may have different predecessors
    - Branch on the choice of the predecessor of J_i
- Heuristic for the initial upper bound
  - Iterative improvement procedure
  - Fast neighborhood search (Hendel & Sourd, to appear in EJOR)
  - Run the descent procedure 10 times from random initial sequence

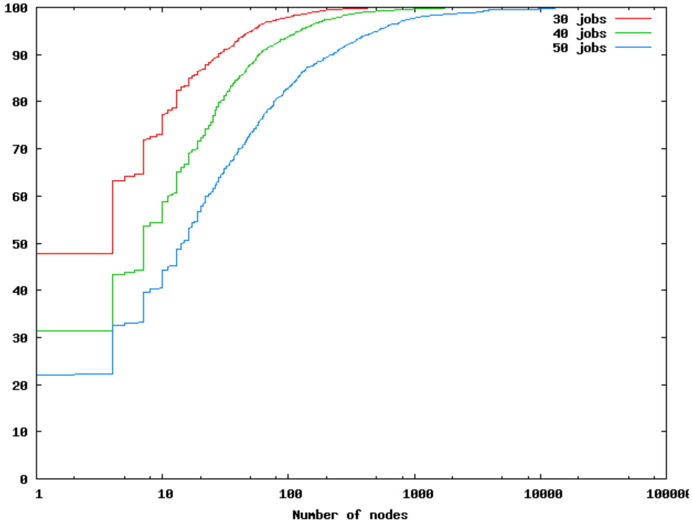
Instances

- $n = 20, 30, 40$ and 50 jobs
- Processing times between 10 and 100
- Due date generation
  - Tardiness factor τ
  - Range factor ρ
  - Due date in $\tau P \pm \rho P/2$ with $P = \sum_i p_i$
- ρ and τ in 0.2; 0.3 ... 0.8
- 26 instances for each (n, ρ, τ)
- **5096** instances

Run-time distribution



Number of nodes



Conclusion

- Conclusion
  - The performance are significantly better than previous algorithms
    - Common due date
    - Distinct due dates
  - Good behaviour in presence of release dates
- Further work
  - Improving the lower bound
    - moves other than swap
    - CP techniques: Shaving / Edge-finding
  - New problems and constraints:
    - No idle time
    - Precedence graph, setups
    - Difficult instances for $1|r_i|\sum w_i T_i$