Génération de contraintes et prétraitement en programmation mathématique: application à un problème d'ordonnancement avec sélection

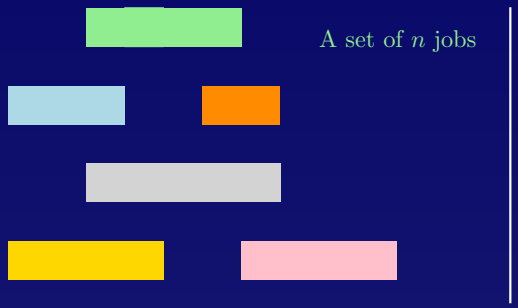
F.Della Croce[1], C. Koulamas[2], V T'kindt[3]

1. D.A.I. Politecnico di Torino, Italy and CNR, IEIIT, Torino, Italy
2. College of Business, Florida International University, USA
3. Université Francois-Rabelais, CNRS, Tours, France

JIRC 2014

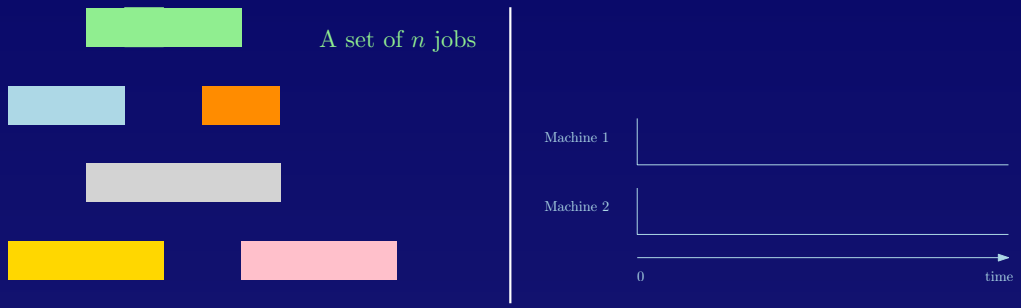
Tours, 16 Avril 2014

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$



- Each job i has proc. times a_i on machine 1 and b_i on machine 2.
- $C_{j,i}$ denotes the compl. time of job i on machine j.
- all jobs have the same due date d which is unknown.

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$



- Each job i has proc. times a_i on machine 1 and b_i on machine 2.
- $C_{j,i}$ denotes the compl. time of job i on machine j.
- all jobs have the same due date d which is unknown.

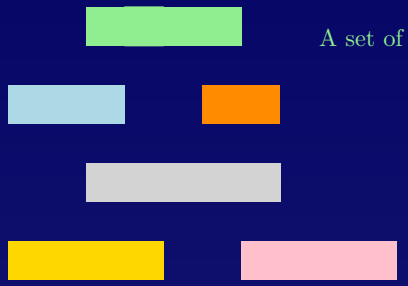
Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

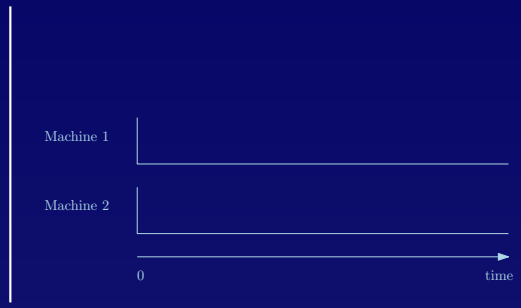
- Each job i has proc. times a_i on machine 1 and b_i on machine 2.
- $C_{j,i}$ denotes the compl. time of job i on machine j.
- all jobs have the same due date d which is unknown.

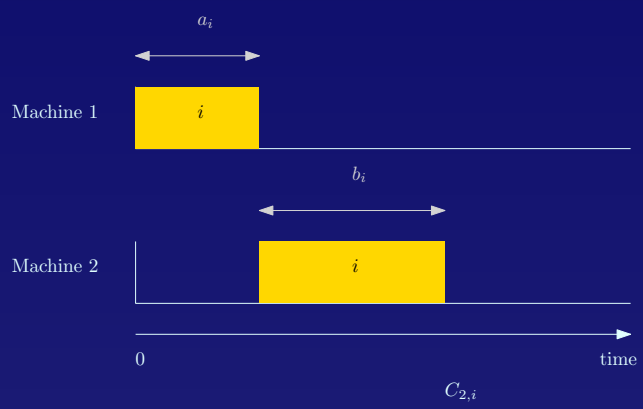
Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- Each job i has proc. times a_i on machine 1 and b_i on machine 2.
- $C_{j,i}$ denotes the compl. time of job i on machine j.
- all jobs have the same due date d which is unknown.

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$







- Each job i has proc. times a_i on machine 1 and b_i on machine 2.
- $C_{j,i}$ denotes the compl. time of job i on machine j.
- all jobs have the same due date d which is unknown.

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

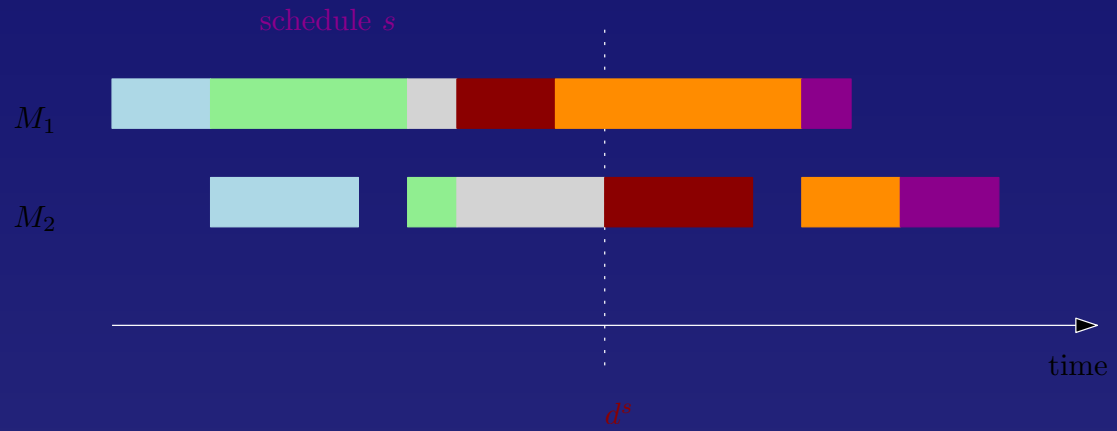
- $\bar{U} = \sum_{i=1}^{n} U_i$ denotes the number of late jobs, where $U_i = 1$ if job i is late ($C_{2,i} > d$), otherwise $U_i = 0$.
- The objective is to minimize the number of late jobs $\bar{U}$ as well as the common due date d ;
- A solution of the problem is a couple (s, d^s), where s is a schedule and d^s its common due date, to which a criteria vector $[\bar{U}(s); d^s]$ is associated.

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- $\bar{U} = \sum_{i=1}^{n} U_i$ denotes the number of late jobs, where $U_i = 1$ if job i is late ($C_{2,i} > d$), otherwise $U_i = 0$.
- The objective is to minimize the number of late jobs $\bar{U}$ as well as the common due date d ;
- A solution of the problem is a couple (s, d^s), where s is a schedule and d^s its common due date, to which a criteria vector $[\bar{U}(s); d^s]$ is associated.

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- $\bar{U} = \sum_{i=1}^{n} U_i$ denotes the number of late jobs, where $U_i = 1$ if job i is late ($C_{2,i} > d$), otherwise $U_i = 0$.
- The objective is to minimize the number of late jobs $\bar{U}$ as well as the common due date d ;
- A solution of the problem is a couple (s, d^s), where s is a schedule and d^s its common due date, to which a criteria vector $[\bar{U}(s); d^s]$ is associated.



Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- What are we looking for ?
- Pareto optimal solutions (s, d^s),

  A solution (s, d^s) is a strict Pareto optimum iff there does not exist another solution $(s', d^{s'})$ such that $\bar{U}(s') \leq \bar{U}(s)$ and $d^{s'} \leq d^s$ with at least one strict inequality.

  $\Rightarrow$ The enumeration problem is weakly $\mathcal{NP}$-hard.
- Computing one Pareto optimum by the ϵ-constraint approach is also weakly $\mathcal{NP}$-hard,
  - Problem 1 : $F2|d_i = d, unknown\ d|\epsilon(d/\bar{U})$ is equivalent to $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$
  - Problem 2 : $F2|d_i = d, unknown\ d|\epsilon(\bar{U}/d)$ is equivalent to $F2|d_i = \epsilon|\bar{U}$

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- What are we looking for ?
- Pareto optimal solutions (s, d^s),

  A solution (s, d^s) is a strict Pareto optimum iff there does not exist another solution $(s', d^{s'})$ such that $\bar{U}(s') \leq \bar{U}(s)$ and $d^{s'} \leq d^s$ with at least one strict inequality.

  $\Rightarrow$ The enumeration problem is weakly $\mathcal{NP}$-hard.
- Computing one Pareto optimum by the ϵ-constraint approach is also weakly $\mathcal{NP}$-hard,
  - Problem 1 : $F2|d_i = d, unknown\ d|\epsilon(d/\bar{U})$ is equivalent to $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$
  - Problem 2 : $F2|d_i = d, unknown\ d|\epsilon(\bar{U}/d)$ is equivalent to $F2|d_i = \epsilon|\bar{U}$

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- What are we looking for ?
- Pareto optimal solutions (s, d^s),

  A solution (s, d^s) is a strict Pareto optimum iff there does not exist another solution $(s', d^{s'})$ such that $\bar{U}(s') \leq \bar{U}(s)$ and $d^{s'} \leq d^s$ with at least one strict inequality.

  $\Rightarrow$ The enumeration problem is weakly $\mathcal{NP}$-hard.
- Computing one Pareto optimum by the ϵ-constraint approach is also weakly $\mathcal{NP}$-hard,
  - Problem 1 : $F2|d_i = d, unknown\ d|\epsilon(d/\bar{U})$ is equivalent to $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$
  - Problem 2 : $F2|d_i = d, unknown\ d|\epsilon(\bar{U}/d)$ is equivalent to $F2|d_i = \epsilon|\bar{U}$

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- What are we looking for ?
- Pareto optimal solutions (s, d^s),

  A solution (s, d^s) is a strict Pareto optimum iff there does not exist another solution $(s', d^{s'})$ such that $\bar{U}(s') \leq \bar{U}(s)$ and $d^{s'} \leq d^s$ with at least one strict inequality.

  $\Rightarrow$ The enumeration problem is weakly $\mathcal{NP}$-hard.
- Computing one Pareto optimum by the ϵ-constraint approach is also weakly $\mathcal{NP}$-hard,
  - Problem 1 : $F2|d_i = d, unknown\ d|\epsilon(d/\bar{U})$ is equivalent to $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$
  - Problem 2 : $F2|d_i = d, unknown\ d|\epsilon(\bar{U}/d)$ is equivalent to $F2|d_i = \epsilon|\bar{U}$

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- Why are they selection problems ?
- Problem 1 : I give you a number ϵ of jobs to be processed $\Rightarrow$ compute the minimum makespan,
- Problem 2 : I give you a time horizon $[0; \epsilon] \Rightarrow$ try to process as much jobs as possible,
- ... can be usefull for deriving lower bounds for other scheduling problems...

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- Why are they selection problems ?
- Problem 1 : I give you a number ϵ of jobs to be processed $\Rightarrow$ compute the minimum makespan,
- Problem 2 : I give you a time horizon $[0; \epsilon] \Rightarrow$ try to process as much jobs as possible,
- ... can be usefull for deriving lower bounds for other scheduling problems...

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- Why are they selection problems ?
- Problem 1 : I give you a number ϵ of jobs to be processed $\Rightarrow$ compute the minimum makespan,
- Problem 2 : I give you a time horizon $[0; \epsilon] \Rightarrow$ try to process as much jobs as possible,
- ... can be usefull for deriving lower bounds for other scheduling problems...

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- Why are they selection problems ?
- Problem 1 : I give you a number ϵ of jobs to be processed $\Rightarrow$ compute the minimum makespan,
- Problem 2 : I give you a time horizon $[0; \epsilon] \Rightarrow$ try to process as much jobs as possible,
- ... can be usefull for deriving lower bounds for other scheduling problems...

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- An important remark....
- Problem 1 ($F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$) : in case $\epsilon = 0$, we have the $F2||C_{max}$ problem,
- Polynomially solvable in $O(n\log(n))$ time by Johnson's algorithm (1954),
- Given any subset of $(n - \epsilon)$ jobs the associated value of d can be computed by that algorithm.

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- An important remark....
- Problem 1 ($F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$) : in case $\epsilon = 0$, we have the $F2||C_{max}$ problem,
- Polynomially solvable in $O(n\log(n))$ time by Johnson's algorithm (1954),
- Given any subset of $(n - \epsilon)$ jobs the associated value of d can be computed by that algorithm.

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- An important remark....
- Problem 1 ($F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$) : in case $\epsilon = 0$, we have the $F2||C_{max}$ problem,
- Polynomially solvable in $O(n\log(n))$ time by Johnson's algorithm (1954),
- Given any subset of $(n - \epsilon)$ jobs the associated value of d can be computed by that algorithm.

Introducing the problem $F2|d_i = d, unknown\ d|d, \bar{U}$

- An important remark....
- Problem 1 ($F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$) : in case $\epsilon = 0$, we have the $F2||C_{max}$ problem,
- Polynomially solvable in $O(n \log(n))$ time by Johnson's algorithm (1954),
- Given any subset of $(n - \epsilon)$ jobs the associated value of d can be computed by that algorithm.

Literature

- Hariri and Potts, 1989 : exact approach for problem $F||\bar{U}$.
- Jozefowska et al., 1994 : Complexity issues on various shop problems including problem $F2|d_i = d|\bar{U}^w$
- Gupta and Hariri, 1997 : special cases and polynomial time heuristics for problem $F2||\bar{U}$.
- Della Croce et al., 2000 : exact approach for problem $F2|d_i = d|\bar{U}$ (Problem 2/900 jobs).
- T'kindt et al., 2007 : exact approach for problem $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$ (Problem 1/3000 jobs).
- Della Croce et al., 2014 : exact IP approach for problem $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$ (Problem 1/100000 jobs).

Literature

- Hariri and Potts, 1989 : exact approach for problem $F||\bar{U}$.
- Jozefowska et al., 1994 : Complexity issues on various shop problems including problem $F2|d_i = d|\bar{U}^w$
- Gupta and Hariri, 1997 : special cases and polynomial time heuristics for problem $F2||\bar{U}$.
- Della Croce et al., 2000 : exact approach for problem $F2|d_i = d|\bar{U}$ (Problem 2/900 jobs).
- T'kindt et al., 2007 : exact approach for problem $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$ (Problem 1/3000 jobs).
- Della Croce et al., 2014 : exact IP approach for problem $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$ (Problem 1/100000 jobs).

Literature

- Hariri and Potts, 1989 : exact approach for problem $F||\bar{U}$.
- Jozefowska et al., 1994 : Complexity issues on various shop problems including problem $F2|d_i = d|\bar{U}^w$
- Gupta and Hariri, 1997 : special cases and polynomial time heuristics for problem $F2||\bar{U}$.
- Della Croce et al., 2000 : exact approach for problem $F2|d_i = d|\bar{U}$ (Problem 2/900 jobs).
- T'kindt et al., 2007 : exact approach for problem $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$ (Problem 1/3000 jobs).
- Della Croce et al., 2014 : exact IP approach for problem $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$ (Problem 1/100000 jobs).

Literature

- Hariri and Potts, 1989 : exact approach for problem $F||\bar{U}$.
- Jozefowska et al., 1994 : Complexity issues on various shop problems including problem $F2|d_i = d|\bar{U}^w$
- Gupta and Hariri, 1997 : special cases and polynomial time heuristics for problem $F2||\bar{U}$.
- Della Croce et al., 2000 : exact approach for problem $F2|d_i = d|\bar{U}$ (Problem 2/900 jobs).
- T'kindt et al., 2007 : exact approach for problem $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$ (Problem 1/3000 jobs).
- Della Croce et al., 2014 : exact IP approach for problem $F2|d_i = d, unknown\ d, \bar{U} = \epsilon|d$ (Problem 1/100000 jobs).

ILP formulation of problem $F2|d_i = d, unknown\ d|\epsilon(d/\bar{U})$

Data :

n, number of jobs

a_i, $1 \leq i \leq n$, proc. time on machine 1
of the job in ith position in Johnson's schedule

b_i, $1 \leq i \leq n$, proc. time on machine 2
of the job in ith position in Johnson's schedule

ϵ, number of late jobs

Variables :

x_i, $1 \leq i \leq n$, is equal to 1 if job i is early, 0 otherwise

d, the common due date

ILP formulation of problem $F2|d_i = d, unknown\ d|\epsilon(d/\bar{U})$

Problem A :

Minimize d

Constraints :

$$\sum_{i=1}^{n} x_i = n - \epsilon$$

$$\sum_{i=1}^{u} a_i x_i + \sum_{i=u}^{n} b_i x_i \leq d,\ \forall u = 1, ..., n$$

$$x_i \in \{0; 1\},\ \forall i = 1, ..., n$$

Example for problem A with $\epsilon = 1$

J_i	a_i	b_i
1	25	70
2	40	40
3	50	20
4	65	15

Min d

$x_1 + x_2 + x_3 + x_4 = 3$

$25x_1 + 70x_1 + 40x_2 + 20x_3 + 15x_4 \leq d$ constraint c_1

$25x_1 + 40x_2 + 40x_2 + 20x_3 + 15x_4 \leq d$

$25x_1 + 40x_2 + 50x_3 + 20x_3 + 15x_4 \leq d$

$25x_1 + 40x_2 + 50x_3 + 65x_4 + 15x_4 \leq d$ constraint c_n

$x_i \in \{0; 1\}, \forall i = 1, ..., 4$

Let us go back to 2007 : preprocessing

- Goal : reduce the size of the IP formulation,
- How ?... by exploiting simple properties between IP and its linear relaxation LP,
- Notations :
  - x_j^{LP} : values of the x_j variables of the LP-relaxation,
  - z_{LP}^* : optimal value of the objective function of the LP-relaxation,
  - B^* : be the associated basis,
  - z_{IP}^* : optimal value of the objective function of the IP,

Let us go back to 2007 : preprocessing

- Goal : reduce the size of the IP formulation,
- How ?... by exploiting simple properties between IP and its linear relaxation LP,
- Notations :
  - x^{LP} : values of the x variables of the LP-relaxation,
  - z^*_{LP} : optimal value of the objective function of the LP-relaxation,
  - B^* : be the associated basis,
  - z^*_{IP} : optimal value of the objective function of the IP,

Let us go back to 2007 : preprocessing

- Goal : reduce the size of the IP formulation,
- How ?... by exploiting simple properties between IP and its linear relaxation LP,
- Notations :
  - x^{LP} : values of the x variables of the LP-relaxation,
  - x^{IP} : values of the x variables of the IP,
  - z^*_{LP} : optimal value of the objective function of the LP-relaxation,
  - B^* : be the associated basis,
  - z^*_{IP} : optimal value of the objective function of the IP,

Let us go back to 2007 : preprocessing

- Goal : reduce the size of the IP formulation,
- How ?... by exploiting simple properties between IP and its linear relaxation LP,
- Notations :
  - x^{LP} : values of the x variables of the LP-relaxation,
  - x^{IP} : values of the x variables of the IP,
  - z^*_{LP} : optimal value of the objective function of the LP-relaxation,
  - B^* : be the associated basis,
  - z^*_{IP} : optimal value of the objective function of the IP,

Let us go back to 2007 : preprocessing

- Goal : reduce the size of the IP formulation,
- How ?... by exploiting simple properties between IP and its linear relaxation LP,
- Notations :
  - x^{LP} : values of the x variables of the LP-relaxation,
  - x^{IP} : values of the x variables of the IP,
  - z^*_{LP} : optimal value of the objective function of the LP-relaxation,
  - B^* : be the associated basis,
  - z^*_{IP} : optimal value of the objective function of the IP,

Let us go back to 2007 : preprocessing

- Goal : reduce the size of the IP formulation,
- How ?... by exploiting simple properties between IP and its linear relaxation LP,
- Notations :
  - x^{LP} : values of the x variables of the LP-relaxation,
  - x^{IP} : values of the x variables of the IP,
  - z^*_{LP} : optimal value of the objective function of the LP-relaxation,
  - B^* : be the associated basis,
  - z^*_{IP} : optimal value of the objective function of the IP,

Let us go back to 2007 : preprocessing

- Goal : reduce the size of the IP formulation,
- How ?... by exploiting simple properties between IP and its linear relaxation LP,
- Notations :
  - x^{LP} : values of the x variables of the LP-relaxation,
  - x^{IP} : values of the x variables of the IP,
  - z^*_{LP} : optimal value of the objective function of the LP-relaxation,
  - B^* : be the associated basis,
  - z^*_{IP} : optimal value of the objective function of the IP,

Let us go back to 2007 : preprocessing

- Goal : reduce the size of the IP formulation,
- How ?... by exploiting simple properties between IP and its linear relaxation LP,
- Notations :
  - x^{LP} : values of the x variables of the LP-relaxation,
  - x^{IP} : values of the x variables of the IP,
  - z^*_{LP} : optimal value of the objective function of the LP-relaxation,
  - B^* : be the associated basis,
  - z^*_{IP} : optimal value of the objective function of the IP,

Let us go back to 2007 : preprocessing

- Concerning non-basic variables $x_i^{LP} \notin B^*$,
- We know that :

$$z^*_{MIP} = z^*_{LP} + \sum_{x_i^{LP} \notin B^*} r_i x_i^{LP}, \quad (1)$$

with r_i the reduced cost associated to variable x_i^{LP}.

- Let UB be an upper bound to z^*_{MIP}. Then, we can write :

$$\sum_{x_i^{LP} \notin B^*} r_i x_i^{LP} \leq UB - z^*_{LP}. \quad (2)$$

- Fixing rule : $\forall x_i^{LP} \notin B^*$, if $r_i > UB - z^*_{LP}$ then in any optimal solution of the IP formulation, $x_i^{IP} = 0$. By reasoning on the slack variables s_i associated to the constraints $x_i \leq 1$, we can deduce when $x_i^{IP} = 1$.

Let us go back to 2007 : preprocessing

- Concerning non-basic variables $x_i^{LP} \notin B^*$,
- We know that :

$$z^*_{MIP} = z^*_{LP} + \sum_{x_i^{LP} \notin B^*} r_i x_i^{LP}, \quad (1)$$

with r_i the reduced cost associated to variable x_i^{LP}.

- Let UB be an upper bound to z^*_{MIP}. Then, we can write :

$$\sum_{x_i^{LP} \notin B^*} r_i x_i^{LP} \leq UB - z^*_{LP}. \quad (2)$$

- Fixing rule : $\forall x_i^{LP} \notin B^*$, if $r_i > UB - z^*_{LP}$ then in any optimal solution of the IP formulation, $x_i^{IP} = 0$. By reasoning on the slack variables s_i associated to the constraints $x_i \leq 1$, we can deduce when $x_i^{IP} = 1$.

Let us go back to 2007 : preprocessing

- Concerning non-basic variables $x_i^{LP} \notin B^*$,
- We know that :

$$z^*_{MIP} = z^*_{LP} + \sum_{x_i^{LP} \notin B^*} r_i x_i^{LP}, \quad (1)$$

with r_i the reduced cost associated to variable x_i^{LP}.

- Let UB be an upper bound to z^*_{MIP}. Then, we can write :

$$\sum_{x_i^{LP} \notin B^*} r_i x_i^{LP} \leq UB - z^*_{LP}. \quad (2)$$

- Fixing rule : $\forall x_i^{LP} \notin B^*$, if $r_i > UB - z^*_{LP}$ then in any optimal solution of the IP formulation, $x_i^{IP} = 0$. By reasoning on the slack variables s_i associated to the constraints $x_i \leq 1$, we can deduce when $x_i^{IP} = 1$.

Let us go back to 2007 : preprocessing

- Concerning non-basic variables $x_i^{LP} \notin B^*$,
- We know that :

$$z^*_{MIP} = z^*_{LP} + \sum_{x_i^{LP} \notin B^*} r_i x_i^{LP}, \quad (1)$$

with r_i the reduced cost associated to variable x_i^{LP}.

- Let UB be an upper bound to z^*_{MIP}. Then, we can write :

$$\sum_{x_i^{LP} \notin B^*} r_i x_i^{LP} \leq UB - z^*_{LP}. \quad (2)$$

- Fixing rule : $\forall x_i^{LP} \notin B^*$, if $r_i > UB - z^*_{LP}$ then in any optimal solution of the IP formulation, $x_i^{IP} = 0$. By reasoning on the slack variables s_i associated to the constraints $x_i \leq 1$, we can deduce when $x_i^{IP} = 1$.

Let us go back to 2007 : preprocessing

- Concerning basic variables $x_i^{LP} \in B^*$,
- All reduced costs are equal to 0,
- Pseudo-costs l_i and u_i (Driebeek's penalties),
- Fixing rule :
  1. $\forall x_i^{LP} \in B^*$, if $(l_i x_{jt}^{LP}) > (UB - z_{LP}^*)$ then $x_i^{IP} = 1$,
  2. $\forall x_i^{LP} \in B^*$, if $(u_i(1 - x_{jt}^{LP})) > (UB - z_{LP}^*)$ then $x_i^{IP} = 0$.

Let us go back to 2007 : preprocessing

- Concerning basic variables $x_i^{LP} \in B^*$,
- All reduced costs are equal to 0,
- Pseudo-costs l_i and u_i (Driebeek's penalties),
- Fixing rule :
  1. $\forall x_i^{LP} \in B^*$, if $(l_i x_{jt}^{LP}) > (UB - z_{LP}^*)$ then $x_i^{IP} = 1$,
  2. $\forall x_i^{LP} \in B^*$, if $(u_i(1 - x_{jt}^{LP})) > (UB - z_{LP}^*)$ then $x_i^{IP} = 0$.

Let us go back to 2007 : preprocessing

- Concerning basic variables $x_i^{LP} \in B^*$,
- All reduced costs are equal to 0,
- Pseudo-costs l_i and u_i (Driebeek's penalties),
- Fixing rule :
  1. $\forall x_i^{LP} \in B^*$, if $(l_i x_{jt}^{LP}) > (UB - z_{LP}^*)$ then $x_i^{IP} = 1$,
  2. $\forall x_i^{LP} \in B^*$, if $(u_i(1 - x_{jt}^{LP})) > (UB - z_{LP}^*)$ then $x_i^{IP} = 0$.

Let us go back to 2007 : preprocessing

- Concerning basic variables $x_i^{LP} \in B^*$,
- All reduced costs are equal to 0,
- Pseudo-costs l_i and u_i (Driebeek's penalties),
- Fixing rule :
  1. $\forall x_i^{LP} \in B^*$, if $(l_i x_{jt}^{LP}) > (UB - z_{LP}^*)$ then $x_i^{IP} = 1$,
  2. $\forall x_i^{LP} \in B^*$, if $(u_i(1 - x_{jt}^{LP})) > (UB - z_{LP}^*)$ then $x_i^{IP} = 0$.

Let us go back to 2007 : preprocessing

- Fixing rules are very simple... and known for a long time (since 1966),
- Fixing non-basic variable is yet implemented in commercial solvers (like CPLEX),
- What is the interest of doing preprocessing ?

  .... exploit the knowledge of a good UB dedicated to our problem,

  .... adequately make use of cuts to strengthen the LP relaxation,

Let us go back to 2007 : preprocessing

- Fixing rules are very simple... and known for a long time (since 1966),
- Fixing non-basic variable is yet implemented in commercial solvers (like CPLEX),
- What is the interest of doing preprocessing ?

  .... exploit the knowledge of a good UB dedicated to our problem,

  .... adequately make use of cuts to strengthen the LP relaxation,

Let us go back to 2007 : preprocessing

- Fixing rules are very simple... and known for a long time (since 1966),
- Fixing non-basic variable is yet implemented in commercial solvers (like CPLEX),
- What is the interest of doing preprocessing ?

  .... exploit the knowledge of a good UB dedicated to our problem,

  .... adequately make use of cuts to strengthen the LP relaxation,

Let us go back to 2007 : preprocessing

- Fixing rules are very simple... and known for a long time (since 1966),
- Fixing non-basic variable is yet implemented in commercial solvers (like CPLEX),
- What is the interest of doing preprocessing ?

  .... exploit the knowledge of a good UB dedicated to our problem,

  .... adequately make use of cuts to strengthen the LP relaxation,

Let us go back to 2007 : preprocessing

- Fixing rules are very simple... and known for a long time (since 1966),
- Fixing non-basic variable is yet implemented in commercial solvers (like CPLEX),
- What is the interest of doing preprocessing ?

  .... exploit the knowledge of a good UB dedicated to our problem,

  .... adequately make use of cuts to strengthen the LP relaxation,

Let us go back to 2007 : cuts

- Logical cuts ($\forall i = 1...n$) :

$$2a_1x_1+...+2a_{i-1}x_{i-1}+(a_i+b_i)x_i+(a_{i+1}+b_{i+1})x_{i+1}+b_{i+2}x_{i+2}+...+b_nx_n \leq 2d-\tau \quad (3)$$

where $\tau = \min_{i-\epsilon+1 \leq h < i, i+\epsilon \geq l > i+1}\{a_{i+1}, b_i, |a_{i+1} - b_i|, 2b_h, 2a_l\}$.

- We also generated 1-cuts (problem independent),
- These cuts have been added to the LP relaxation and then preprocessing is applied,

Let us go back to 2007 : cuts

- Logical cuts ($\forall i = 1...n$) :

$$2a_1x_1 + ... + 2a_{i-1}x_{i-1} + (a_i + b_i)x_i + (a_{i+1} + b_{i+1})x_{i+1} + b_{i+2}x_{i+2} + ... + b_nx_n \leq 2d - \tau \quad (3)$$

where $\tau = \min_{i-\epsilon+1 \leq h < i, i+\epsilon \geq l > i+1}\{a_{i+1}, b_i, |a_{i+1} - b_i|, 2b_h, 2a_l\}$.

- We also generated 1-cuts (problem independent),
- These cuts have been added to the LP relaxation and then preprocessing is applied,

Let us go back to 2007 : cuts

- Logical cuts ($\forall i = 1...n$) :

$$2a_1x_1 + ... + 2a_{i-1}x_{i-1} + (a_i + b_i)x_i + (a_{i+1} + b_{i+1})x_{i+1} + b_{i+2}x_{i+2} + ... + b_nx_n \leq 2d - \tau \quad (3)$$

where $\tau = \min_{i-\epsilon+1 \leq h < i, i+\epsilon \geq l > i+1} \{a_{i+1}, b_i, |a_{i+1} - b_i|, 2b_h, 2a_l\}$.

- We also generated 1-cuts (problem independent),
- These cuts have been added to the LP relaxation and then preprocessing is applied,

Let us go back to 2007 : experimentation (preprocessing)

	LB_{cut}/UB						
n	G_{avg}	G_{max}	D_{avg}	D_{max}	t_{avg}	t_{max}	Fix
100	2.06	5.37	37.85	80.00	0.03	1	83.90
200	1.11	2.76	30.69	66.66	0.06	1	80.00
300	0.60	2.05	30.46	50.00	0.23	1	78.87
400	0.53	1.49	26.14	50.00	0.26	1	79.46
500	0.38	1.34	25.98	50.00	0.43	1	76.40
1000	0.16	0.40	29.82	50.00	3.10	4	88.41
1500	0.08	0.27	35.97	50.00	9.23	10	88.82
2000	0.07	0.15	29.16	50.00	23.26	32	89.84
2500	0.05	0.14	28.33	33.33	43.83	48	85.97
3000	0.02	0.10	34.44	50.00	96.10	115	86.68

with $G = 1000\frac{UB-LB}{Opt}$ and $D = 100\frac{Opt-LB}{UB-LB}$.

Let us go back to 2007 : experimentation (exact solution)

	IP (CPLEX 8.0)				*BaB*			
n	t_{avg}	t_{max}	nd_{avg}	nd_{max}	t_{avg}	t_{max}	nd_{avg}	nd_{max}
100	0.13	1	44.80	121	0.03	1	40.06	181
200	0.43	1	64.50	280	0.23	1	85.23	231
300	1.23	2	114.63	386	0.40	1	97.43	329
400	2.10	5	63.86	326	0.80	2	117.40	331
500	3.50	7	84.73	592	1.53	4	134.53	383
1000	21.20	52	84.56	428	9.36	28	171.33	921
1500	65.70	143	117.40	630	21.46	55	136.30	481
2000	136.53	303	79.80	537	49.36	210	153.13	1383
2500	Out of time				91.50	177	208.73	987
3000	Out of time				156.36	341	179.93	1089
3500	Out of time				Out of time			

time limit = 180s.

From 2007 to 2014 : experimentation (exact solution)

n	IP (CPLEX 12.2)				BaB			
	t_{avg}	t_{max}	n_{avg}	n_{max}	t_{avg}	t_{max}	n_{avg}	n_{max}
1000	4.70	10	128.26	512	1.90	4	112.53	347
2000	20.23	45	285.03	663	11.13	17	136.86	379
3000	54.06	151	439.60	978	42.46	189	403.80	4097
4000	Out of Memory				Out of Memory			

- Why *Out of Memory* for 4000 jobs ?

  $\Rightarrow$ Approximately 800 binary variables (not so much)...

  $\Rightarrow$ ...exactly 4001 constraints (not so much)...

  $\Rightarrow$ ... but $16,000,000$ non-zero coefficients.

  Even the LP is not solvable by the solver.

From 2007 to 2014 : experimentation (exact solution)

n	IP (CPLEX 12.2)				BaB			
	t_{avg}	t_{max}	n_{avg}	n_{max}	t_{avg}	t_{max}	n_{avg}	n_{max}
1000	4.70	10	128.26	512	1.90	4	112.53	347
2000	20.23	45	285.03	663	11.13	17	136.86	379
3000	54.06	151	439.60	978	42.46	189	403.80	4097
4000	Out of Memory				Out of Memory			

- Why *Out of Memory* for 4000 jobs ?

  $\Rightarrow$ Approximately 800 binary variables (not so much)...

  $\Rightarrow$ ...exactly 4001 constraints (not so much)...

  $\Rightarrow$ ... but $16,000,000$ non-zero coefficients.

  Even the LP is not solvable by the solver.

From 2007 to 2014 : experimentation (exact solution)

n	IP (CPLEX 12.2)				BaB			
	t_{avg}	t_{max}	n_{avg}	n_{max}	t_{avg}	t_{max}	n_{avg}	n_{max}
1000	4.70	10	128.26	512	1.90	4	112.53	347
2000	20.23	45	285.03	663	11.13	17	136.86	379
3000	54.06	151	439.60	978	42.46	189	403.80	4097
4000	Out of Memory				Out of Memory			

- Why *Out of Memory* for 4000 jobs ?

  $\Rightarrow$ Approximately 800 binary variables (not so much)...

  $\Rightarrow$ ...exactly 4001 constraints (not so much)...

  $\Rightarrow$ ... but $16,000,000$ non-zero coefficients.

  Even the LP is not solvable by the solver.

Relaxing all but two constraints in the ILP formulation of problem $F2|d_i = d, unknown\ d|\epsilon(d/\bar{U})$

Problem A_{rel}

Minimize d

Constraints :

$\sum_{i=1}^{n} x_i = n - \epsilon$

$\sum_{i=1}^{1} a_i x_i + \sum_{i=1}^{n} b_i x_i \leq d,$ constraint c_1

$\sum_{i=1}^{n} a_i x_i + \sum_{i=n}^{n} b_i x_i \leq d,$ constraint c_n

$x_i \in \{0; 1\}, \forall i = 1, ..., n$

Comparing A and A_{rel}

Property : $\frac{OPT_A}{OPT_{A_{rel}}} \leq 2$ and this ratio is asymptotically tight.

- The idea of our CG approach is to start with A_{rel} and iteratively add missing constraints from A,
- Can work if few of such constraints are added since solving A_{rel} is much faster than solving A,
- No more preprocessing on the variables / Preprocessing on the constraints.

Comparing A and A_{rel}

Property : $\frac{OPT_A}{OPT_{A_{rel}}} \leq 2$ and this ratio is asymptotically tight.

- The idea of our CG approach is to start with A_{rel} and iteratively add missing constraints from A,
- Can work if few of such constraints are added since solving A_{rel} is much faster than solving A,
- No more preprocessing on the variables / Preprocessing on the constraints.

Comparing A and A_{rel}

Property : $\frac{OPT_A}{OPT_{A_{rel}}} \leq 2$ and this ratio is asymptotically tight.

- The idea of our CG approach is to start with A_{rel} and iteratively add missing constraints from A,
- Can work if few of such constraints are added since solving A_{rel} is much faster than solving A,
- No more preprocessing on the variables / Preprocessing on the constraints.

Comparing A and A_{rel}

Property : $\frac{OPT_A}{OPT_{A_{rel}}} \leq 2$ and this ratio is asymptotically tight.

- The idea of our CG approach is to start with A_{rel} and iteratively add missing constraints from A,
- Can work if few of such constraints are added since solving A_{rel} is much faster than solving A,
- No more preprocessing on the variables / Preprocessing on the constraints.

Constraint Generation Approach : pseudo-code

1: End=$False$
2: **while** !End **do**
3: Solve the IP of A_{rel} : $\bar{x}$ is its solution with value $OPT_{A_{rel}}$
4: Compute the value OPT_A of the IP of A for solution $\bar{x}$
5: **if** $(OPT_A = OPT_{A_{rel}})$ **then**
6: End=$True$
7: **else**
8: Let c_j be the most violated constraint in A
9: Add c_j to A_{rel}
10: **end if**
11: **end while**
12: **return** $\bar{x}$ as the optimal solution of A

Computational tests

- a_i's and b_i's drawn at random using an uniform distribution between 10 and 100.
- The value ϵ can take $(n+1)$ values. We tested $\epsilon = \lceil \frac{n}{2} \rceil$, $\epsilon = n - 10$ and $\epsilon = n - 5$.
- For each problem size, 30 instances are generated.
- All tests were performed on a PC Computer Intel i5 with 4 cores of 2.6GHz and 8Gb of RAM. Whenever we solved an IP model we used the CPLEX 12.2 mathematical solver.

Computational tests for the $F2|d_j = d,\ unknown\ d\ |\epsilon(d/\sum U_j)$ problem

- Comparing BaB (the exact algorithm of T'kindt et al, 2007), IP (the ILP model solved by CPLEX 12.2) and $IPCG$ (the constraint generation approach) for $\epsilon = n/2$.

n	IP		BaB		$IPCG$	
	t_{avg}	t_{max}	t_{avg}	t_{max}	t_{avg}	t_{max}
1000	4.70	10	1.90	4	0.03	1
2000	20.23	45	11.13	17	0.06	1
3000	54.06	151	42.46	189	0.10	1
4000	Out of Memory		Out of Memory		0.10	1

Solving instances for $\epsilon = \frac{n}{2}$

Computational tests

n	$IPCG$					
	t_{avg}	t_{max}	n_{avg}	n_{max}	it_{avg}	it_{max}
10000	0.46	1	36.06	174	1.0	1
20000	0.76	2	33.96	228	1.0	1
30000	1.36	5	35.66	331	1.0	1
40000	3.03	8	75.26	330	1.0	1
50000	6.13	16	181.86	600	1.0	1
60000	5.96	12	141.03	450	1.0	1
70000	6.26	10	160.20	527	1.0	1
80000	8.26	16	161.80	442	1.0	1
90000	9.33	17	144.66	400	1.0	1
100000	11.00	27	198.46	934	1.0	1

Solving large instances for $\epsilon = \frac{n}{2}$

Computational tests

n	$IPCG$					
	t_{avg}	t_{max}	n_{avg}	n_{max}	it_{avg}	it_{max}
10000	0.40	1	117.30	696	14.76	27
20000	1.46	4	372.63	1478	29.90	43
30000	3.80	8	678.76	2119	47.30	70
40000	8.73	16	1320.23	5011	68.80	89
50000	16.83	26	1872.10	6793	87.46	115
60000	36.10	62	3639.26	22673	116.70	149
70000	69.06	105	4513.16	26700	138.50	176
80000	107.66	140	5398.03	30913	167.63	194
90000	140.23	264	6657.63	15460	194.66	227
100000	208.33	311	9512.46	15310	219.80	253

Solving large instances for $\epsilon = n - 10$

Computational tests

n	$IPCG$					
	t_{avg}	t_{max}	n_{avg}	n_{max}	it_{avg}	it_{max}
10000	0.43	1	141.13	606	23.66	38
20000	1.23	2	383.13	842	41.43	55
30000	3.40	5	599.10	1150	63.26	84
40000	8.10	11	1108.20	2065	85.60	101
50000	16.36	23	1468.93	2842	110.66	130
60000	32.16	44	2021.63	3455	137.06	156
70000	54.73	73	2797.33	4621	160.63	182
80000	86.20	112	4168.40	7088	183.33	203
90000	133.43	190	6139.33	10678	206.76	232
100000	196.86	254	9663.96	16279	230.90	258

Solving large instances for $\epsilon = n - 5$

Computational tests

n	$IPCG$					
	t_{avg}	t_{max}	n_{avg}	n_{max}	it_{avg}	it_{max}
500	32.30	50	8475.16	10557	600.70	789
1000	62.86	115	22413.10	40688	1131.03	1697
1500	152.50	269	33811.70	39205	1647.36	1976

Enumeration of the strict Pareto optima for the flow shop problem

Conclusions and extensions

- A constraint generation approach applied to the IP formulation of the problem has been proposed.
- The approach is capable of solving problems with up to 100000 jobs.
- The approach has been extended to job shop and open shop with similar results.
- The presence of jobs weights does not affect the strength of the solution approach.
- Possible extensions of this approach may occur in all scheduling problems that can be expressed by means of multidimensional knapsack problems, as, for instance, the $1||\sum w_j U_j$ problem with or without deadline constraints.

Conclusions and extensions

- A constraint generation approach applied to the IP formulation of the problem has been proposed.
- The approach is capable of solving problems with up to 100000 jobs.
- The approach has been extended to job shop and open shop with similar results.
- The presence of jobs weights does not affect the strength of the solution approach.
- Possible extensions of this approach may occur in all scheduling problems that can be expressed by means of multidimensional knapsack problems, as, for instance, the $1||\sum w_j U_j$ problem with or without deadline constraints.

Conclusions and extensions

- A constraint generation approach applied to the IP formulation of the problem has been proposed.
- The approach is capable of solving problems with up to 100000 jobs.
- The approach has been extended to job shop and open shop with similar results.
- The presence of jobs weights does not affect the strength of the solution approach.
- Possible extensions of this approach may occur in all scheduling problems that can be expressed by means of multidimensional knapsack problems, as, for instance, the $1||\sum w_j U_j$ problem with or without deadline constraints.

Conclusions and extensions

- A constraint generation approach applied to the IP formulation of the problem has been proposed.
- The approach is capable of solving problems with up to 100000 jobs.
- The approach has been extended to job shop and open shop with similar results.
- The presence of jobs weights does not affect the strength of the solution approach.
- Possible extensions of this approach may occur in all scheduling problems that can be expressed by means of multidimensional knapsack problems, as, for instance, the $1||\sum w_j U_j$ problem with or without deadline constraints.

Conclusions and extensions

- A constraint generation approach applied to the IP formulation of the problem has been proposed.
- The approach is capable of solving problems with up to 100000 jobs.
- The approach has been extended to job shop and open shop with similar results.
- The presence of jobs weights does not affect the strength of the solution approach.
- Possible extensions of this approach may occur in all scheduling problems that can be expressed by means of multidimensional knapsack problems, as, for instance, the $1||\sum w_j U_j$ problem with or without deadline constraints.