Extended Formulations for the Scheduling Problem: Column Generation

Artur Pessoa and Eduardo Uchoa
Engenharia de Produção, Universidade Federal Fluminense, Brazil.
{artur,uchoa}@producao.uff.br.

Marcus Poggi de Aragão
Departamento de Informática, PUC Rio de Janeiro, Brazil.
poggi@inf.puc-rio.br.

Rosiane Rodrigues
COPPE - Sistemas, Universidade Federal do Rio de Janeiro.
rosiane@cos.ufrj.br.
1. Introduction
   - Problem Definition
   - Earlier work

2. Formulations and Cuts
   - Extended Formulations
   - Proposed Formulation
   - Cuts

3. Column Generation
   - Pricing
   - Fixing by reduced costs
   - Stabilization
   - Branch-cut-and-price (robust)

4. Computational Experiments
The Scheduling Problem:

- $J = \{1, \ldots, n\}$ - set of jobs to be processed
- $M = \{1, \ldots, m\}$ - set of parallel identical machines
- $p_j$ - job $j$ positive integral processing time
- $f_j(C_j)$ - cost function over completion time of job $j$
- $C_j$ - completion time of job $j$

Find the machines and instants in time for all jobs to start such that:

i. No preemption

ii. Each machine can process at most one job at a time

iii. Machines can stay idle

iv. Minimizes $\sum_{j=1}^{n} f_j(C_j)$
Special cases of this cost function:

- $1|| \sum w_j T_j$ single machine weighted tardiness
- $P|| \sum w_j T_j$ multiple machine weighted tardiness
  
  $d_j$ - due date of job $j$
  $T_j = \max\{0, C_j - d_j\}$ - tardiness of job $j$
  $w_j$ - weight of tardiness of job $j$

**Strongly NP-Hard**

Models any cost function based on penalties for

- job earliness or tardiness
- time window for start and/or completion of jobs
  penalties can be infinity
Exact approaches use two distinct kinds of formulations (Queyranne, Schulz 97):

- MIP formulations where job sequence is represented by binary variables and completion times by continuous variables;
- IP time indexed formulations, where the completion time of each job is represented by binary variables indexed over a discretized time horizon

Latter formulations are known to yield better bounds
Pseudo-polynomially large number of variables $\Rightarrow$ difficulty
Avella, Boccia and D’Auria (2005):
- near-optimal solutions (gaps below 3%)
- instances with $n$ up to 400, using Lagrangean relaxation to approximate the time indexed formulation bound

Pan and Shi (2007):
- showed that the classical time indexed bound can be exactly computed by solving a cleverly crafted transportation problem.
- branch-and-bound for the $1|| \sum w_j T_j$ that consistently solved all the OR-Library instances with up to 100 jobs

Bigras, Gamache and Savard (2008):
- proposed obtaining the same bound by column generation
- branch-and-price somehow less efficient
- could not solve some of those instances

The last two algorithms may need to explore large enumeration trees
Time indexed formulation for the single machine scheduling problem.

Dyer, Wolsey 1990
Sousa, Wolsey 1992
Van der Akker et al. 1999, 2000
...

- all jobs must be processed in a given time horizon ranging from 0 to $T$
- binary variables $y_{jt}$ indicate that job $j$ starts at time $t$ on some machine
Time-Indexed Formulation

Minimize
\[ \sum_{j \in J} \sum_{t=0}^{T-p_j} f_j(t + p_j) y_j^t \]  \quad (1a)

S.t.
\[ \sum_{t=0}^{T-p_j} y_j^t = 1 \quad \text{for} \quad j \in J \]  \quad (1b)

\[ \sum_{j \in J, \ t \ + \ p_j \ \leq \ T} \sum_{s=\max\{0, t-p_j+1\}}^{t} y_j^s \leq 1 \quad (t = 0, \ldots, T - 1), \quad (1c) \]

\[ y_j^t \in \{0, 1\} \quad j \in J; \ t = 0, \ldots, T - p_j \]  \quad (1d)

Parallel machines: right-hand side \[ 1 \rightarrow m. \]
Proposed Formulation: Arc-Time-Indexed

- Uses an even larger number of variables:
  - one for each pair of jobs and
  - each possible completion time.
- Also assumes an execution time horizon from 0 to $T$
- Machines are idle at time 0 and after time $T$
Binary variables $x_{ij}^t$, $i \neq j$,
indicate that job $i$ completes and job $j$
starts at time $t$ on the same machine.

$x_{0j}^t$ indicate that job $j$ starts at time $t$ in a machine that was idle from time $t - 1$ to $t$

in particular, $x_{0j}^0$ indicate that $j$ starts on some machine at time 0

$x_{i0}^t$ indicate that job $i$ finishes at time $t$ at a machine that will stay idle from time $t$ to $t + 1$

in particular, variables $x_{i0}^T$ indicate that $i$ is the last job at a machine

integral variables $x_{00}^t$ indicate the number of machines that were idle from time $t - 1$ to $t$ that will remain idle from time $t$ to $t + 1$

$J_+ = \{0, 1, \ldots, n\}$
Arc-Time-Indexed Formulation

\[
\text{Min } \sum_{i \in J_+} \sum_{j \in J \setminus \{i\}} \sum_{t = p_i}^{T - p_j} f_j(t + p_j) x_{ij}^t \\
\sum_{i \in J_+ \setminus \{j\}} \sum_{t = p_i}^{T - p_j} x_{ij}^t = 1 \quad \forall j \in J
\]

\[
\sum_{j \in J_+ \setminus \{i\}} x_{ji}^t - \sum_{j \in J_+ \setminus \{i\}} x_{ij}^{t + p_i} = 0 \quad \forall i \in J; \ t = 0, \ldots, T - p_i
\]

\[
\sum_{j \in J_+} x_{j0}^t - \sum_{j \in J_+, \ t + p_j + 1 \leq T} x_{0j}^{t + 1} = 0 \quad t = 0, \ldots, T - 1
\]
\[
\sum_{j \in J_+} x_{0j}^0 = m \quad (3a)
\]
\[
x_{ij}^t \in \mathbb{Z}_+ \quad \forall i \in J_+; \forall j \in J_+ \setminus \{i\} \quad (3b)
\]
\[
t = p_i, \ldots, T - p_j), \quad (3c)
\]
\[
x_{00}^t \in \mathbb{Z}_+ \quad t = 0, \ldots, T - 1 \quad (3d)
\]

... and the redundant equation:

\[
\sum_{i \in J_+} x_{i0}^T = m \quad (4)
\]

It defines a network flow of \( m \) units over an acyclic layered graph \( G = (V, A) \).
Network $m$ units sent from source to sink

Example: $m = 2$, $n = 4$ $p_1 = 2$, $p_2 = 1$, $p_3 = 2$, $p_4 = 4$ $T = 6$

An integral solution of the arc-time indexed formulation:
paths in the layered network.
Proposition

The Arc-Time-Indexed formulation dominates the Time-Indexed formulation.

Proof.

Let $\bar{x}$ be a linear relaxation solution of Arc-Time-Indexed with cost $z$. $\bar{x}$ can be converted into $\bar{y}$ a linear relaxation solution of the Time-Indexed formulation with same cost:

$$\bar{y}_j^t = \sum_{i \in J+ \setminus \{j\}} \bar{x}_{ij}^t, \quad j \in J, \quad t = 0, \ldots, T - p_j.$$ 

Arc-Time-Indexed formulation can be strictly better than the Time-Indexed formulation: Example

$1|| \sum w_j T_j$ problem where $n = 3$;
$p_1 = 100, p_2 = 300, p_3 = 200$; $d_1 = 200, d_2 = 300, d_3 = 400$;
$w_1 = 6, w_2 = 3, w_3 = 2$; and $T = 600$. 
If the Arc-Time-Indexed formulation is weakened by adding $x^t_{jj}$ variables:

- it becomes equivalent to the Time-Indexed formulation
- i.e., it is only slightly better

On the other hand, the Arc-Time-Indexed formulation can be strengthened

**Proposition**

For jobs $i$ and $j$ in $J$, $i < j$, let $x^t_{ij}$ and $x^{t-p_i+p_j}_{ji}$ be a pair of variables defined in Arc-Time-Indexed and let

$$\Delta = (f_i(t) + f_j(t + p_j)) - (f_j(t - p_i + p_j) + f_i(t + p_j))$$

- If $\Delta \geq 0$ variable: $x^t_{ij}$ can be removed
- (Else) If $\Delta < 0$, $x^{t-p_i+p_j}_{ji}$ can be removed.
Job i and j are processed consecutively on some machine

Swap jobs

Completion times are known for each job in both cases

Just compare the resulting $f(C_i) + f(C_j)$

A similar reasoning shows that:

**Proposition**

For job $j$ in $J$, let $x_{j0}^t$ and $x_{0j}^{t-p_j+1}$ be a pair of variables defined in the Arc-Time-Indexed formulation. Let $\Delta = f_j(t) - f_j(t+1)$.

- If $\Delta > 0$ variable $x_{j0}^t$ can be removed
- If $\Delta \leq 0$, $x_{0j}^{t-p_j+1}$ can be removed.

**Result of this Preprocessing**

Exact 50% of the arcs are removed
The Arc-Time-Indexed Formulation has none of the eliminated arcs.
The acyclic network $G = (V, A)$ also has none of these arcs in $A$.
The LP relaxation accepts pseudo-schedules: jobs may repeat (although this preprocessing eliminates many).

**New Formulation**

To the best of our knowledge, this formulation is NEW.

- Picard and Queyranne (1978): three-index formulation for the $1\| \sum w_j T_j$ variables $x^k_{ij}$, meaning that job $j$ follows job $i$ and is the $k$-th job to be scheduled.
  - it has $O(n^3)$ variables and $O(n^2)$ constraints
- Arc-Time-Indexed formulation has $O(n^2 T)$ variables and $O(nT)$ constraints
Pseudo-polynomially large number of variables and constraints makes the direct use of this formulation prohibitive.

We can rewrite it in terms of variables associated to the pseudo-schedules.

The pseudo-schedules are source-destination paths in $G = (V, A)$.

Let $P$ be the set of all source-destination paths in $G = (V, A)$.

- $\lambda_p$: 0-1 variable associated to pseudo-schedule $p$
- $q_{a}^{tp}$: 0-1 coefficient indicating if arc $a^t$ is in the path of pseudo-schedule $p$
- $q_{a}^{tp}$ is associated to variable $x_a^t$ in the Arc-Time-Indexed formulation
- Define $f_0(t)$ as zero for all $t$. 

\[ \lambda_p \] \[ q_{a}^{tp} \]
We may write the Explicit Master:

\[
\begin{align*}
\text{Minimize} & \quad \sum_{(i,j)^t \in A} f_j(t + p_j)x_{ij}^t \\
\text{S.t.} & \quad \sum_{p \in P} q^{tp}_a \lambda_p - x_a^t = 0 \quad (\forall a^t \in A), \\
& \quad \sum_{(j,i)^t \in A} x_{ji}^t = 1 \quad (\forall i \in J), \\
& \quad \sum_{(0,j)^0 \in A} x_{0j}^0 = m \\
& \quad \lambda_p \geq 0 \quad (\forall p \in P), \\
& \quad x_a^t \in \mathbb{Z}_+ \quad (\forall a^t \in A).
\end{align*}
\]
We eliminate the $x$ variables and relax integrality to obtain the Dantzig-Wolfe Master (DWM) LP:

\[
\begin{align*}
\text{Minimize} \quad & \sum_{p \in P} \left( \sum_{(i,j)^t \in A} q_{ij}^{tp} f_j(t + p_j) \right) \lambda_p \\
\text{S.t.} \quad & \sum_{p \in P} \left( \sum_{(j,i)^t \in A} q_{ji}^{tp} \right) \lambda_p = 1 \quad (\forall i \in J), \\
& \sum_{p \in P} \left( \sum_{(0,j)^0 \in A} q_{0j}^{0p} \right) \lambda_p = m \\
& \lambda_p \geq 0 \quad (\forall p \in P). \\
\end{align*}
\]

- $\sum_{(0,j)^0 \in A} q_{0j}^{0p} = 1$ for any $p \in P$
• Cuts on the $x$ variables, $\sum_{a^t \in A} \alpha^t_{al} x^t_a \geq b_l$ take the form $\sum_{p=1}^{P} (\sum_{a^t \in A} \alpha^t_{al} q^{tp}) \lambda_p \geq b_l$ in the DWM.

• Suppose we have $r$ constraints where $\alpha$ is the coefficients for it in the $x$ format.

• $\pi$ are the dual variables associated to these constraints

The reduced cost of an arc $a^t = (i, j)^t$ is then:

$$\bar{c}^t_a = f_j (t + p_j) - \sum_{l=0}^{r} \alpha^t_{al} \pi_l.$$ (7)
We separated the Extended Capacity Cuts:

- Generic family of cuts (Uchoa 2005)
- Effective on the capacitated minimum spanning tree (Uchoa et al. 2008)
- Also effective on many vehicle routing problem variants (Pessoa et al. 2008)
Cuts over the time-indexed variables are derived.

For each vertex $i \in V_+$ the following balance equation is valid:

$$\sum_{a^t \in \delta^-(i)} t x^t_a - \sum_{a^t \in \delta^+(i)} t x^t_a = p_i \quad (8)$$

Let $S \subseteq V_+$ be a set of vertices. Summing the equalities corresponding to each $i \in S$, we get the time-balance equation over $S$:

**Definition**

An *Extended Capacity Cut* (ECC) over $S$ is any inequality valid for $P(S)$, the polyhedron given by the convex hull of the 0-1 solutions of

$$\sum_{a^t \in \delta^-(S)} t x^t_a - \sum_{a^t \in \delta^+(S)} t x^t_a = p(S)$$

It can be noted that those equations are always satisfied by the solutions (DWM) (translated to the $x^t$ space by $\sum_{j=1}^{p} q^t_a \lambda_j - x^t_a$).
HECCs: aggregated variables $v^t$ and $z^t$

$$v^t = \sum_{a^t \in \delta^+(S)} x^t_a \quad (t = 1, \ldots, T), \quad (9)$$

$$z^t = \sum_{a^t \in \delta^-(S)} x^t_a \quad (t = 1, \ldots, T). \quad (10)$$

The balance equation over those variables is:

$$\sum_{t=1}^{T} t v^t - \sum_{t=1}^{T} t z^t = p(S). \quad (11)$$

For each possible pair of values of $T$ and $D = p(S)$, a polyhedron $P(T, D)$

• HECCs are facets of $P(T, D)$
Pricing subproblem:

- Shortest path in the acyclic network $G = (V, A)$
  - Takes $\Theta(|A|)$
  - $|A| = \Theta(n^2 T)$
  - $T = \Omega(np_{avg}/m)$, where $p_{avg}$ is the average job processing time

Time consuming:
For $m = 1$, $n = 100$ and $p_{avg} = 50$, $|A|$ is more than 25 million.
i extreme degeneracy, in fact, when $m = 1$ it can happen that any optimal basis has just one variable with a positive value

ii extreme variable symmetry, in the sense that there is usually many alternative solutions with the same cost

iii an expensive pricing with complexity $\Omega(n^3 p_{avg}/m)$, where $p_{avg}$ is the average job processing time.
Lagrangian subproblem:

\[
L(\pi) = \text{Min} \sum_{a^t \in A} \bar{c}_a^t x_a^t + \sum_{l=0}^{r} b_l \pi_l \quad (12a)
\]

S.t.

\[
\sum_{p \in P} q_{a^t}^p \lambda_p - x_{a^t} = 0 \quad (\forall a^t \in A), \quad (12b)
\]

\[
\sum_{(0,j)^0 \in A} x_{0j}^0 = m \quad (12c)
\]

\[
\lambda_p \geq 0 \quad (\forall p \in P), \quad (12d)
\]

\[
x_{a^t}^t \in Z_+ \quad (\forall a^t \in A). \quad (12e)
\]

- For each possible \( \pi \), an optimal solution can be constructed by setting \( \lambda_{p^*} = m - p^* \): path of minimum reduced cost,
- all other \( \lambda \) variables are set to zero,
- the \( x \) variables are set in order to satisfy (12b).
Let $L(\pi, a, t)$ be the solution of the Lagrangean problem with the additional constraint $x_a^t \geq 1$

- $a^t$ can be eliminated if $L(\pi, a, t) \geq Z_{INC} - Z_{INC}$ best known solution

- $L(\pi, a, t)$ can be computed by obtaining the shortest path labels forward and backward, and finally adding $a^t$ reduced cost to its extremeties’ labels

- Amounts to have a pricing 3 times slower

This fixing procedure performed extremely well.
Stabilization

- Presented by Eduardo Uchoa yesterday
- Convex combination of Lagrangean dual and DWM’s current simplex multipliers
- One parameter
- Misprice
- Either dual or primal improvement: exponential convergence
- Hot start with the Volume algorithm
Branch-cut-and-price (robust)

- Primal heuristics
- Root bounds are strong
- Branching on original variables $x^t_a$
- Switch to Branch-and-cut when the number $x^t_a$ is small
Instances for $1|| \sum w_j T_j$

- Experiments taken on the set of 375 instances of the OR Library
- Generated by Potts and Wassenhove (1985) and contains 125 instances for each $n \in \{40, 50, 100\}$.
- Same set used by Pan and Shi (2007) and Bigras, Gamache and Savard (2008)

Instances for $P|| \sum w_j T_j$

- We derived 100 new instances from those in the OR-Library
- For $m \in \{2, 4\}$, $n\{40, 50\}$
  - We pick the first $1|| \sum w_j T_j$ instance in each group (the one ending in 1 or 6)
  - Divided each due date $d_j$ by $m$ (and rounded down)
  - Processing times $p_j$ and weights $w_j$ were kept unchanged
All our experiments were performed in a notebook with processor Intel Core Duo (but using a single core) with a clock of 1.66GHz and 2GB of RAM. The linear program solver was CPLEX 11
### Table: Comparison of the complete BCP algorithm with the best algorithm by Pan and Shi

<table>
<thead>
<tr>
<th>$n$</th>
<th>Alg.</th>
<th>Avg $T(s)$</th>
<th>Max $T(s)$</th>
<th>Avg. Nd</th>
<th>Max Nd</th>
<th>Root Gap %</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>PS</td>
<td>69.0</td>
<td>235</td>
<td>141</td>
<td>293</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td>BCP</td>
<td>12.1</td>
<td>43.6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>PS</td>
<td>142.8</td>
<td>232</td>
<td>416</td>
<td>5623</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>BCP</td>
<td>28.1</td>
<td>123.8</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>PS</td>
<td>1811</td>
<td>32400</td>
<td>18877</td>
<td>&gt;909844</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td>BCP</td>
<td>648.5</td>
<td>8508</td>
<td>2.03</td>
<td>42</td>
<td>0.0013</td>
</tr>
</tbody>
</table>
Table: Detailed results of the complete BCP algorithm over a sample of 25 OR-Library instances with \( n = 100 \).

<table>
<thead>
<tr>
<th>Inst</th>
<th>Volume</th>
<th>1st.LP</th>
<th>Remain Root</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>407703</td>
<td>160</td>
<td>279.9</td>
</tr>
<tr>
<td>21</td>
<td>898925</td>
<td>125</td>
<td>212.7</td>
</tr>
<tr>
<td>26</td>
<td>8</td>
<td>1</td>
<td>69.4</td>
</tr>
<tr>
<td>31</td>
<td>24202</td>
<td>20</td>
<td>90.2</td>
</tr>
<tr>
<td>36</td>
<td>108293</td>
<td>93</td>
<td>209.3</td>
</tr>
<tr>
<td>41</td>
<td>462117</td>
<td>401</td>
<td>922.7</td>
</tr>
<tr>
<td>46</td>
<td>829771</td>
<td>340</td>
<td>498.0</td>
</tr>
<tr>
<td>51</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>56</td>
<td>9046</td>
<td>20</td>
<td>76.6</td>
</tr>
<tr>
<td>61</td>
<td>86793</td>
<td>227</td>
<td>480.4</td>
</tr>
<tr>
<td>66</td>
<td>243637</td>
<td>555</td>
<td>1164.7</td>
</tr>
<tr>
<td>71</td>
<td>640799</td>
<td>351</td>
<td>735.1</td>
</tr>
<tr>
<td>76</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>81</td>
<td>1400</td>
<td>30</td>
<td>80.2</td>
</tr>
<tr>
<td>86</td>
<td>66850</td>
<td>186</td>
<td>463.5</td>
</tr>
<tr>
<td>91</td>
<td>248284</td>
<td>401</td>
<td>1027.7</td>
</tr>
<tr>
<td>96</td>
<td>495358</td>
<td>376</td>
<td>918.7</td>
</tr>
<tr>
<td>101</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>106</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>111</td>
<td>158962</td>
<td>454</td>
<td>1554.2</td>
</tr>
<tr>
<td>116</td>
<td>370435</td>
<td>445</td>
<td>1288.2</td>
</tr>
<tr>
<td>121</td>
<td>471166</td>
<td>392</td>
<td>957.4</td>
</tr>
</tbody>
</table>
**Table:** Comparison of different bounding methods for multi-machine instances.

| n | m | Time indexed | |
|---|---|---|---|---|---|---|---|---|---|---|---|
|   |   | Av Gap % | M Gap% | T(s) | Av Gap % | M Gap% | T(s) | Av Gap % | M Gap% | T(s) |
| 40 | 2 | 1.533 | 21.016 | 85.6 | 1.243 | 20.840 | 32.2 | 0.053 | 0.853 | 295.9 |
|   | 4 | 0.544 | 4.787 | 32.2 | 0.406 | 3.390 | 14.1 | 0.105 | 0.841 | 63.9 |
| 50 | 2 | 0.535 | 4.074 | 182.2 | 0.487 | 4.074 | 88.3 | 0.078 | 1.051 | 2298.7 |
|   | 4 | 0.529 | 5.614 | 79.5 | 0.489 | 5.614 | 36.8 | 0.266 | 5.088 | 262.5 |
Table: Detailed results of the complete BCP algorithm over the instances with $m = 2$ and $n = 40$.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>584</td>
<td>89</td>
<td>18.0</td>
<td>156948</td>
<td>606</td>
<td>7</td>
<td>325.4</td>
<td>0</td>
<td>1</td>
<td>343.4</td>
<td>606</td>
</tr>
<tr>
<td>6</td>
<td>3875</td>
<td>141</td>
<td>25.5</td>
<td>82838</td>
<td>3886</td>
<td>3</td>
<td>119.6</td>
<td>0</td>
<td>1</td>
<td>145.1</td>
<td>3886</td>
</tr>
<tr>
<td>11</td>
<td>9592</td>
<td>189</td>
<td>34.5</td>
<td>66999</td>
<td>9617</td>
<td>2</td>
<td>94.6</td>
<td>0</td>
<td>1</td>
<td>129.1</td>
<td>9617</td>
</tr>
<tr>
<td>16</td>
<td>38279</td>
<td>292</td>
<td>45.2</td>
<td>59225</td>
<td>38351</td>
<td>2</td>
<td>515.8</td>
<td>59225</td>
<td>3</td>
<td>561.0</td>
<td>38356</td>
</tr>
<tr>
<td>21</td>
<td>41048</td>
<td>384</td>
<td>37.1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>37.1</td>
<td>41048</td>
</tr>
<tr>
<td>26</td>
<td>87</td>
<td>48</td>
<td>12.7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>12.7</td>
<td>87</td>
</tr>
<tr>
<td>31</td>
<td>3758</td>
<td>172</td>
<td>34.0</td>
<td>106263</td>
<td>3812</td>
<td>5</td>
<td>452.0</td>
<td>0</td>
<td>1</td>
<td>486.0</td>
<td>3812</td>
</tr>
<tr>
<td>36</td>
<td>10662</td>
<td>303</td>
<td>44.4</td>
<td>52812</td>
<td>10700</td>
<td>2</td>
<td>1113.6</td>
<td>52812</td>
<td>5</td>
<td>1193.6</td>
<td>10713</td>
</tr>
<tr>
<td>41</td>
<td>30802</td>
<td>387</td>
<td>46.4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>46.4</td>
<td>30802</td>
</tr>
<tr>
<td>46</td>
<td>34146</td>
<td>430</td>
<td>29.8</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>29.8</td>
<td>34146</td>
</tr>
<tr>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>1272</td>
<td>80</td>
<td>16.5</td>
<td>107098</td>
<td>1279</td>
<td>2</td>
<td>72.3</td>
<td>0</td>
<td>1</td>
<td>88.8</td>
<td>1279</td>
</tr>
<tr>
<td>61</td>
<td>11311</td>
<td>269</td>
<td>45.3</td>
<td>72238</td>
<td>11390</td>
<td>2</td>
<td>1754.2</td>
<td>72238</td>
<td>327</td>
<td>9097.3</td>
<td>11488</td>
</tr>
<tr>
<td>66</td>
<td>35130</td>
<td>323</td>
<td>51.9</td>
<td>75499</td>
<td>35196</td>
<td>2</td>
<td>1503.6</td>
<td>75499</td>
<td>196</td>
<td>6451.1</td>
<td>35279</td>
</tr>
<tr>
<td>71</td>
<td>47935</td>
<td>423</td>
<td>42.9</td>
<td>42430</td>
<td>47952</td>
<td>2</td>
<td>19.5</td>
<td>0</td>
<td>1</td>
<td>62.4</td>
<td>47952</td>
</tr>
<tr>
<td>76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>452</td>
<td>150</td>
<td>20.6</td>
<td>71423</td>
<td>571</td>
<td>2</td>
<td>947.2</td>
<td>0</td>
<td>1</td>
<td>967.8</td>
<td>571</td>
</tr>
<tr>
<td>86</td>
<td>5996</td>
<td>302</td>
<td>40.4</td>
<td>47829</td>
<td>6041</td>
<td>2</td>
<td>253.5</td>
<td>47829</td>
<td>6</td>
<td>298.0</td>
<td>6048</td>
</tr>
<tr>
<td>91</td>
<td>26075</td>
<td>388</td>
<td>56.6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>56.6</td>
<td>26075</td>
</tr>
<tr>
<td>96</td>
<td>66110</td>
<td>358</td>
<td>50.9</td>
<td>46481</td>
<td>66116</td>
<td>2</td>
<td>2.9</td>
<td>0</td>
<td>1</td>
<td>53.8</td>
<td>66116</td>
</tr>
<tr>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>106</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>17898</td>
<td>292</td>
<td>50.0</td>
<td>51884</td>
<td>17936</td>
<td>2</td>
<td>46.5</td>
<td>0</td>
<td>1</td>
<td>96.5</td>
<td>17936</td>
</tr>
<tr>
<td>116</td>
<td>25786</td>
<td>317</td>
<td>50.4</td>
<td>54574</td>
<td>25870</td>
<td>2</td>
<td>173.7</td>
<td>0</td>
<td>1</td>
<td>224.1</td>
<td>25870</td>
</tr>
<tr>
<td>121</td>
<td>64507</td>
<td>390</td>
<td>50.9</td>
<td>48152</td>
<td>64516</td>
<td>2</td>
<td>3.0</td>
<td>0</td>
<td>1</td>
<td>53.9</td>
<td>64516</td>
</tr>
</tbody>
</table>
Comments

- At the end of this first column generation step, besides having a bound close to the optimal, the number of non-fixed variables is usually quite small. Switching to branch-and-cut was an alternative.

- In almost all $1||\sum w_j T_j$ benchmark instances from the OR-Library, with $n \in \{40, 50, 100\}$, we found that the duality gaps were reduced to zero still in the root node.

- The same algorithm was also tested on the $P||\sum w_j T_j$, a harder problem. We do not know any paper claiming optimal solutions on instances of significant size.

- Our branch-cut-and-price could solve instances derived from those in the OR-Library, with $m \in \{2, 4\}$ and $n \in \{40, 50\}$, consistently. However, the solution of several such multi-machine instances did required a significant amount of branching.
Thank you!

Merci Guy, Jacques and Marco!