A Branch-and-Cut-and-Price algorithm for a large class of parallel machine scheduling problems

Teobaldo Bulhões² Ruslan Sadykov¹ Eduardo Uchoa²
Anand Subramanian³

₁ Inria Bordeaux and Univ. Bordeaux, France
₂ Univ. Federal Fluminense, Brazil
₃ Univ. Federal da Paraíba, Brazil

Journées GOTHa/Bermudes
Tours, September 26, 2017
Contents

Introduction

Set covering formulation and Branch-and-Price

Subset-row cuts

Computational results
The scheduling problem we want to solve

- Set $M$ of unrelated machines
- $n$ jobs, each job $j \in J = \{1, \ldots, n\}$ has
  - processing time $p^k_j$, dependent on the machine
  - release and due dates $r_j$ and $d_j$
  - earliness and tardiness unitary penalties $\alpha_j$ and $\beta_j$
- Given completion time $C_j$ of job $j \in J$ in the schedule, its cost is
  \[
  \alpha_j E_j + \beta_j T_j = \alpha_j \cdot \max\{0, d_j - C_j\} + \beta_j \cdot \max\{0, C_j - d_j\}
  \]
- There is a sequence-dependent setup time $s_{i,j}^k$ if job $j$ is scheduled immediately after job $i$ on machine $k$.
- The objective is to minimize the total earliness/tardiness cost.
- Problem’s notation:
  \[
  R|r_j, s_{i,j}^k| \sum_j \alpha_j E_j + \beta_j T_j
  \]
Heterogeneous Vehicle Routing with Time Windows

Set $I$ of customers, each $i \in I$ with demand $d_i$, service time $s_i$ and time window $[r_i, d_i]$.

Set $M$ of vehicle types, each $k \in M$ with a depot $i_{|I|+k}$ with $U_k$ vehicles of capacity $Q_k$, with fixed cost $f_u$, travel costs $c_{ij}^k$ and travel distances $d_{ij}^k$ for each pair $(i, j) \in I \cup M$ of customers/depots.

Objective: minimize the total fixed and travel cost.
## Similarities between problems

<table>
<thead>
<tr>
<th>HVRP</th>
<th>UMSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous vehicles</td>
<td>Unrelated machines</td>
</tr>
<tr>
<td>Vehicle routes</td>
<td>Machine schedules</td>
</tr>
<tr>
<td>Capacity resource</td>
<td>One job at a time</td>
</tr>
<tr>
<td>Time resource</td>
<td>Time resource</td>
</tr>
<tr>
<td>Service times</td>
<td>Job processing times</td>
</tr>
<tr>
<td>Distances between customers</td>
<td>Setup times</td>
</tr>
<tr>
<td>Minimizing vehicle cost and total travelled distance</td>
<td>Minimizing “just-in-time” penalty (sort of “soft” time windows)</td>
</tr>
</tbody>
</table>
State-of-the-art exact algorithms for Vehicle Routing

Instances with 100–150 customers are routinely solved to optimality


Existing exact approaches in the literature for scheduling on parallel machines with sum criteria

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>Constraints</th>
<th>Formulations</th>
<th>Machines</th>
<th>Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R</td>
<td>s_{ij}^k</td>
<td>\sum \alpha_j E_j + \beta_j T_j$</td>
<td>Only MIP formulations, up to 5 machines and 12 jobs.</td>
<td>5</td>
</tr>
<tr>
<td>$R</td>
<td></td>
<td>\sum T_j$</td>
<td>A branch-and-bound [Shim and Kim, 2007], up to 5 machines and 20 jobs.</td>
<td>5</td>
</tr>
<tr>
<td>$R</td>
<td></td>
<td>\sum w_j T_j$</td>
<td>A branch-and-bound [Liaw et al., 2003], up to 4 machines and 18 jobs.</td>
<td>4</td>
</tr>
<tr>
<td>$Q</td>
<td>s_{ij}^k</td>
<td>\sum E_j + T_j$</td>
<td>A MIP and a Benders decomposition [Balakrishnan et al., 1999], up to 20 jobs.</td>
<td>5</td>
</tr>
<tr>
<td>$P</td>
<td>s_f</td>
<td>\sum T_j$</td>
<td>A branch-and-bound [Schaller, 2014], up to 3 machines and 14 jobs.</td>
<td>3</td>
</tr>
<tr>
<td>$P</td>
<td>r_j</td>
<td>\sum w_j T_j$</td>
<td>A branch-and-bound [Jouglet and Savourey, 2011], up to 5 machines and 20 jobs</td>
<td>5</td>
</tr>
<tr>
<td>$P</td>
<td></td>
<td>\sum w_j T_j$</td>
<td>A Branch-Cut-and-Price [Pessoa et al., 2010], up to 4 machines and 100 jobs.</td>
<td>4</td>
</tr>
<tr>
<td>$R</td>
<td>a_k, r_j, s_{ij}^k</td>
<td>\sum w_j T_j$</td>
<td>A branch-and-price [Lopes and de Carvalho, 2007], up to 50 machines and 150 jobs</td>
<td>50</td>
</tr>
</tbody>
</table>
Set covering (master) formulation

- \( \Omega_k \) — set of pseudo-schedules for machine \( k \in M \)
- \( a^\omega_j \) — number of times that job \( j \) appears in pseudo-schedule \( \omega \).
- \( c_\omega \) — cost of pseudo-schedule \( \omega \).
- **Binary variable** \( \lambda^\omega_k = 1 \) if and only if pseudo-schedule \( \omega \) is assigned to machine \( k \in M \)

\[
\min \sum_{k \in M} \sum_{\omega \in \Omega_u} c_\omega \lambda_s \\
\sum_{k \in M} \sum_{\omega \in \Omega_u} a^\omega_j \lambda^\omega = 1, \quad \forall j \in J, \\
\sum_{\omega \in \Omega_k} \lambda^\omega \leq 1, \quad \forall k \in M, \\
\lambda^\omega \in \{0, 1\}, \quad \forall \omega \in \Omega_k, \forall k \in M.
\]
Pricing subproblem for machine \( k \in M \)

Extended graph \( G_k \)

Arc \((i, j, t)\) — setup time between job \( i \) and \( j \) is started at time \( t \), and job \( j \) is started at time \( t + s_{ij}^k \)

Variable \( x_{ij}^t \) — arc \((i, j, t)\) in the solution or not

\[
\begin{align*}
J &= \{1, 2, 3\}, \quad T = 8 , \quad p_1 = 4, \quad p_2 = 1, \quad p_3 = 3, \quad s_{ij} = 1, \quad \forall i, j \in J \\
\text{Pseudo-schedules 0-2-3-2-0 and 0-2-1-0 are shown}
\end{align*}
\]
Pricing subproblem: the labelling algorithm

Given dual solution $\pi$ of the restricted master problem, the pricing subproblem is

$$\min_{\omega \in \Omega_k} \bar{c}_\omega = c_\omega - \sum_{j \in J} a^\omega_j \pi_j = \sum_{i,j \in J, i \neq j} \left( c_j^{t+s_{ij}+p_j} - \pi_j \right) \cdot x_{ij}$$

i.e. the shortest path problem in the extended graph.

Labels
Each label $L = (\bar{c}^L, j^L, t^L)$ represents a partial pseudo-schedule. It dominates another label $L'$ if

$$j^L = j^{L'}, t^L = t^{L'}, \bar{c}^L \leq \bar{c}^{L'}$$
The labeling algorithm

\[ t = 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \]

\[ 5 \]
\[ 4 \]
\[ 3 \]
\[ 2 \]
\[ 1 \]

\[ j = 0 \]

Initial label
The labeling algorithm

![Diagram showing label expansion over time](image)
The labeling algorithm

\[
\begin{array}{cccccc}
  t = 0 & 1 & 2 & 3 & 4 & 5 \\
  5 & & & & & \\
  4 & & & & & \\
  3 & & & & & \\
  2 & & & & & \\
  1 & & & & & \\
  j = 0 & & & & & \\
\end{array}
\]

label expansion
The labeling algorithm

\[ t = 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \]

\[
\begin{array}{ccccccc}
5 & & & & & & \\
4 & & & & & & \\
3 & & & & & & \\
2 & & & & & & \\
1 & & & & & & \\
j = 0 & & & & & & \\
\end{array}
\]

Do both labels need to be kept in bucket (4,4)?
The labeling algorithm

The labels represent partial paths 0-1-4 and 0-3-4.
Fixing of arc variables by reduced costs

- $Z_{RM}$ — optimal value of the current restricted master.
- $Z_{sub}^k$ — minimum reduced cost for machine $k \in M$.
- Lagrangian lower bound: $Z_{RM} + \sum_{k \in M} Z_{sub}^k$.
- $Z_{inc}$ — value of the best known integer solution.
- $Z_{sub}^k(a)$ — current minimum reduced cost of a path containing arc $a \in G_k$.

• Arc $a$ can be removed (it cannot take part of any improving solution) if

$$Z_{sub}^k(a) + \sum_{k' \in M \setminus \{k\}} Z_{sub}^{k'} + Z_{RM} \geq Z_{inc}.$$

• A good heuristic is very important!
Computing $Z_{sub}^k(a)$ [Ibaraki and Nakamura, 1994]

How to compute the shortest path passing through arc $a = (i, j, t) \in G_k$?

\[ Z_{sub}^k(a = (i, j, k)) = F(i, t) + B(j, t + s_{ij}^k + p_j^k) + c_{j}^{t+s_{ij}^k+p_j^k} \]

1. $F(i, t)$ — the value of the shortest path from $s$ to node $(i, t)$
2. $B(k, t + s_{ij}^k + p_j^k)$ — the value of the shortest path from $d$ to node $(j, t + s_{ij}^k + p_j^k)$

---

Dual price smoothing stabilization

- $\overline{\pi}$ — current dual solution of the restricted master
- $\pi^*$ — dual solution giving the best Lagrangian bound so far
- We solve the pricing problem using the dual vector

$$\pi' = (1 - \alpha) \cdot \overline{\pi} + \alpha \cdot \pi^*,$$

where $\alpha \in [0, 1)$.

- Parameter $\alpha$ is automatically adjusted in each column generation iteration using the sub-gradient of the Lagrangian function at $\pi'$ [Pessoa et al., 2017].

---

Branching

- Branching on aggregated arc variables

\[ \sum_{0 \leq t \leq T} x_{ij}^{tk} \in \{0, 1\}, \]

i.e. job \( i \) immediately precedes job \( j \) on machine \( k \) or not

- Multi-phase strong branching is used

- Branching history is kept is used through pseudo-costs
Contents

Introduction

Set covering formulation and Branch-and-Price

Subset-row cuts

Computational results
Subset-Row Cuts (SRCs) [Jepsen et al., 2008]

Given $C \subseteq J$ and a multiplier $\rho$, the $(C, \rho)$-Subset Row Cut is:

$$\sum_{k \in M} \sum_{\omega \in \Omega_k} \left[ \rho \sum_{i \in C} a_{i\omega} \right] \lambda_{\omega} \leq \lfloor \rho |C| \rfloor$$

Special case of Chvátal-Gomory rank-1 cuts obtained by rounding of $|C|$ set-packing constraints in the master

Here we use only 1-row and 3-row cuts with $\rho = \frac{1}{2}$. We separate them by enumeration.


Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows.

Impact on the pricing problem

Given dual value $\nu_\gamma < 0$ for each active subset row cut $\gamma \in \Gamma$, defined for subset $C_\gamma$ of jobs, modified reduced cost of pseudo-schedule $\omega \in \Omega_k$ is:

$$
\bar{c}_\omega = \sum_{i, j \in J, t \in T} \left( c_j^t + s_{ij} + p_j - \pi_j \right) \cdot x_{ij}^t - \sum_{\gamma \in \Gamma} \left[ \frac{1}{2} \cdot \sum_{j \in C_\eta, i \in J, \ i \neq j, \ t \in T} x_{ij}^t \right].
$$

Each cut adds to labels an additional binary state $S_L^\gamma$ (parity of the number of times jobs in $C_\gamma$ appear in the partial schedule $L$), resulting in a weaker domination:

$$
\bar{c}^L - \sum_{\gamma \in \Gamma: S_L^\gamma > S_{L'}^\gamma} \nu_\gamma \leq \bar{c}'^L \quad \text{instead of} \quad \bar{c}^L \leq \bar{c}'^L
$$
Limited memory cuts [Pecin et al., 2017]

For each active cut $\gamma \in \Gamma$, define a memory $\mathcal{M}_\gamma$ of vertices (jobs) which “remember” state $S_\gamma$.

If $j^L \notin \mathcal{M}_\gamma$, then $S^L_\gamma \leftarrow 0$.

Vectors $S^L$ are sparser $\Rightarrow$ stronger domination

---

Contents

Introduction

Set covering formulation and Branch-and-Price

Subset-row cuts

Computational results
Results for $R | r_j, s_{ij}^k | \sum \alpha_j E_j + \beta_j T_j$, small setup times

Initial heuristic and instances by [Kramer and Subramanian, 2017]

<table>
<thead>
<tr>
<th>Size</th>
<th>With cuts</th>
<th>BKS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n</td>
<td>m</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>60/60</td>
</tr>
<tr>
<td>60</td>
<td>2</td>
<td>60/60</td>
</tr>
<tr>
<td>60</td>
<td>3</td>
<td>60/60</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>60/60</td>
</tr>
<tr>
<td>80</td>
<td>4</td>
<td>48/60</td>
</tr>
</tbody>
</table>

Size Without cuts

<table>
<thead>
<tr>
<th>Size</th>
<th>Without cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>3</td>
</tr>
</tbody>
</table>

A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems.
Technical report, Universidad Federal da Paraíba, Brazil.
Results for $R \mid r_j, s_{ij}^k \mid \sum \alpha_j E_j + \beta_j T_j$, larger setup times

Initial heuristic and instances by [Kramer and Subramanian, 2017]

<table>
<thead>
<tr>
<th>Size</th>
<th>With cuts</th>
<th>BKS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Solved</td>
<td>Root Gap (%)</td>
<td>Gap (%)</td>
</tr>
<tr>
<td>40</td>
<td>60/60</td>
<td>0.43</td>
<td>0.00</td>
</tr>
<tr>
<td>60</td>
<td>58/60</td>
<td>2.22</td>
<td>0.06</td>
</tr>
<tr>
<td>60</td>
<td>45/60</td>
<td>4.29</td>
<td>1.21</td>
</tr>
<tr>
<td>80</td>
<td>28/60</td>
<td>2.89</td>
<td>1.32</td>
</tr>
<tr>
<td>80</td>
<td>10/60</td>
<td>5.17</td>
<td>3.91</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Size</th>
<th>Without cuts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Solved</td>
<td>Root Gap (%)</td>
</tr>
<tr>
<td>40</td>
<td>60/60</td>
<td>4.08</td>
</tr>
<tr>
<td>60</td>
<td>43/60</td>
<td>4.71</td>
</tr>
<tr>
<td>60</td>
<td>37/60</td>
<td>5.99</td>
</tr>
</tbody>
</table>

A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems.
Technical report, Universidad Federal da Paraíba, Brazil.
Results for $R \mid \sum \alpha_j E_j + \beta_j T_j$

| Size | Our Branch-Cut-and-Price | | | | | | BKS | [Şen and Bülbül, 2015] |
|------|---------------------------|--------|--------|--------|--------|--------|--------|--------|-----------------------|
| $n$  | $m$ | Solv. | Root Gap(%) | Gap (%) | Root Time | Total Time | Nod. num. | Impr. (%) | New | Solv. | Gap (%) | Time |
| 40   | 2  | 60/60 | 0.04 | 0.00 | 2m | 5m | 3.4 | 0.00 | 0 | 26/60 | 0.16 | 1m |
| 60   | 2  | 60/60 | 0.04 | 0.00 | 9m | 12m | 3.3 | 0.00 | 1 | 7/60 | 0.89 | 2m |
| 60   | 3  | 60/60 | 0.05 | 0.00 | 6m | 7m | 2.9 | 0.01 | 5 | 7/60 | 0.82 | 2m |
| 80   | 2  | 59/60 | 0.02 | 0.00 | 28m | 40m | 5.4 | 0.00 | 3 | 2/60 | 0.90 | 2m |
| 80   | 4  | 60/60 | 0.11 | 0.00 | 15m | 16m | 3.9 | 0.07 | 15 | 0/60 | 4.54 | 4m |
| 90   | 3  | 60/60 | 0.05 | 0.00 | 29m | 34m | 4.7 | 0.03 | 20 | 1/60 | 2.52 | 3m |
| 100  | 5  | 59/60 | 0.20 | 0.02 | 31m | 57m | 26.7 | 0.10 | 27 | 0/60 | 8.83 | 5m |
| 120  | 3  | 56/60 | 0.16 | 0.04 | 1h54m | 3h00m | 16.7 | 0.07 | 22 | 0/60 | 4.12 | 3m |
| 120  | 4  | 58/60 | 0.23 | 0.01 | 1h24m | 2h12m | 17.7 | 0.17 | 31 | 0/60 | 6.98 | 4m |

With subset row cuts, root gap is 6 times smaller (40 and 60 jobs instances).

In 30 minutes, CPLEX solved 49/60 inst. with 40 jobs, 36/120 inst. with 60 jobs, 3/120 inst. with 80 jobs, 2/60 inst. with 90 jobs.

Final remarks

- First use of non-robust cuts for scheduling problems
- Significant computational improvement over the existing exact approaches for the problem
  - scales up to 4 machines and 80 jobs for “generic” instances with setup times
  - solves 532/540 instances without setup times with up to 4 machines and 120 jobs
- Need more testing on “less generic” instances
- Ways to improve results:
  - A better heuristic for generic instances is needed!
  - First convergence is very slow
  - More balanced branching
  - Separation for rank-1 cuts with 4 and more rows
  - Enumeration [Baldacci et al., 2008]
  - Avoid discretisation


References II


Minimizing total tardiness for scheduling identical parallel machines with family setups.

Minimizing total tardiness in an unrelated parallel-machine scheduling problem.