Machine scheduling by column-and-row generation on the time-indexed formulation

Ruslan Sadykov[1] François Vanderbeck[1,2]

[1] INRIA Bordeaux — Sud-Ouest, France [2] University Bordeaux I, France

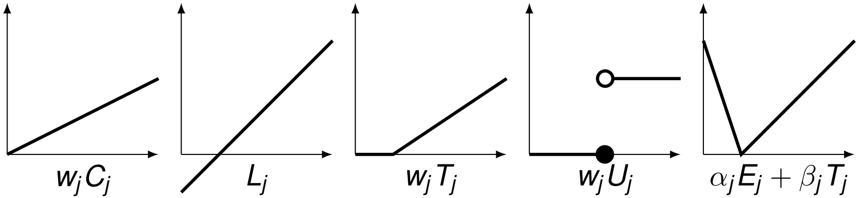
MAPSP 2011
Nymburk, Czech Republic, June 20

Contents

The problem

- m identical machines
- n non-preemptive jobs with discrete processing times p_j
- Arbitrary cost functions $f_j(t)$.
- The objective is to find a schedule $(C_1, \ldots, C_n)$ minimizing the total cost $\sum_j f_j(C_j)$.

Classical cost functions:



Time-indexed formulation

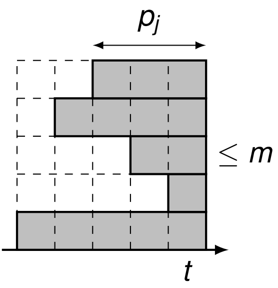
$x_{jt} = 1$ iff job j is started at time t

$$\min \sum_{j\in N} \sum_{t=0}^{T-p_j} f_j(t) x_{jt}$$

$$s.t. \quad \sum_{t=0}^{T-p_j} x_{jt} = 1, \quad j \in N,$$

$$\sum_{j\in N} \sum_{s=t-p_j+1}^{t} x_{js} \le m, \quad t \in [0, T-1],$$

$$x_{jt} \in \{0,1\}, \quad j \in N, \ t \in [0, T-p_j].$$



Very tight LP bounds, but
Solution of LP relaxation takes a lot of time ← our motivation

A brief (and non-complete) history

- First usages: **[Bowman, OR59]**, **[Pritsker, Watters, Wolfe, MS69]**, **[Redwine, Wismer, JOTA74]**.
- Polyhedral studies: **[Dyer, Wolsey, DAM90]**, **[Sousa, Wolsey, MP92]**, **[van den Akker, Hurkens, Savelsbergh, MP99]**
- Column generation: **[van den Akker, Hoogeveen, van de Velde, OR99]**, **[van den Akker, van Hoesel, Savelsbergh, IJoC00]**, **[Bigras, Gamache, Savard, IJoC08]**
- Computationally successful algorithms based on time-indexed formulations: **[Pan, Shi, MP07]**, **[Tanaka, Fujikuma, Araki, JSch09]**, **[Pessoa, Uchoa, Poggi de Aragão, Rodrigues, MPC10]**

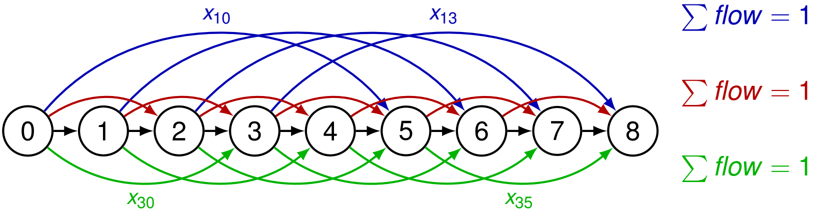
Contents

Sparse (flow) reformulation [R]

x_{0t} — number of idle machines at time t

$$\sum_{j \in N \cup \{0\}} x_{j0} = m,$$

$$\sum_{j \in N} x_{jt} - \sum_{j \in N: t-p_j \geq 0} x_{j,t-p_j} + x_{0t} - x_{0,t-1} = 0, \quad t \in [1, T-1],$$



We need to find a min cost flow of m units from 0 to T.

Path reformulation [M]

$\lambda_p = 1$ iff a unit of flow takes path p

$q_{jt}^p = 1$ iff path p contains arc representing variable x_{jt}

$$
\begin{aligned}
\min \quad & \sum_{p \in P} \sum_{j,t} f_j(t) q_{jt}^p \lambda_p \\
s.t. \quad & \sum_{p \in P} \sum_{t} q_{jt}^p \lambda_p = 1, \quad \forall j \in N, \\
& \sum_{p \in P} \lambda_p = m, \\
& \lambda_p \geq 0, \quad \forall p \in P, \\
& \sum_{p \in P} q_{jt}^p \lambda_p \in \mathbb{Z}_+, \quad \forall j, t.
\end{aligned}
$$

Standard column generation applied to solve $[M_{LP}]$ has severe convergence problems (especially when $m = 1$).

Contents

Column-and-row generation: a hybrid approach

Alternative to direct resolution of $[R_{LP}]$ by a MIP solver

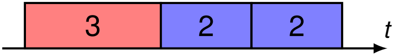
- Dynamic generation of the variables of $[R_{LP}]$:
  generated in bunch by solving the pricing subproblem.
- Adding rows that become active.

Alternative to the standard column generation

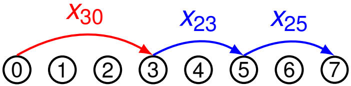
- Perform the column generation for $[M_{LP}]$
- "Project" the master program in $[R_{LP}]$

Column-and-row generation procedure for $[R_{LP}]$

1. Solve the pricing subproblem (obtain a pseudo schedule), using the dual values associated to the “covering” constraints



2. Disaggregate the subproblem solution in variables x.



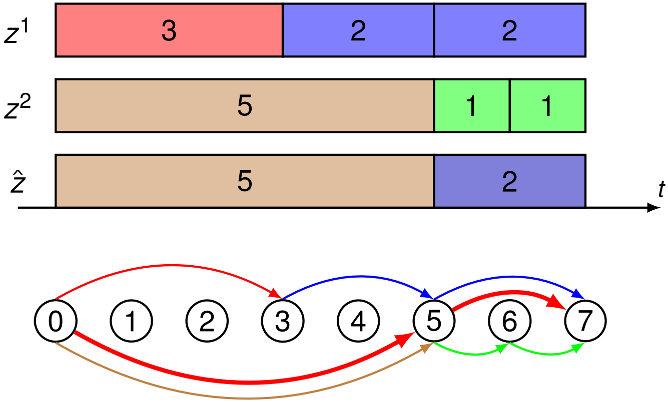
3. Add them to the restricted formulation $[\overline{R}_{LP}]$ along with the associated flow conservation constraints.

4. Resolve $[\overline{R}_{LP}]$ and update dual prices.

Proposition

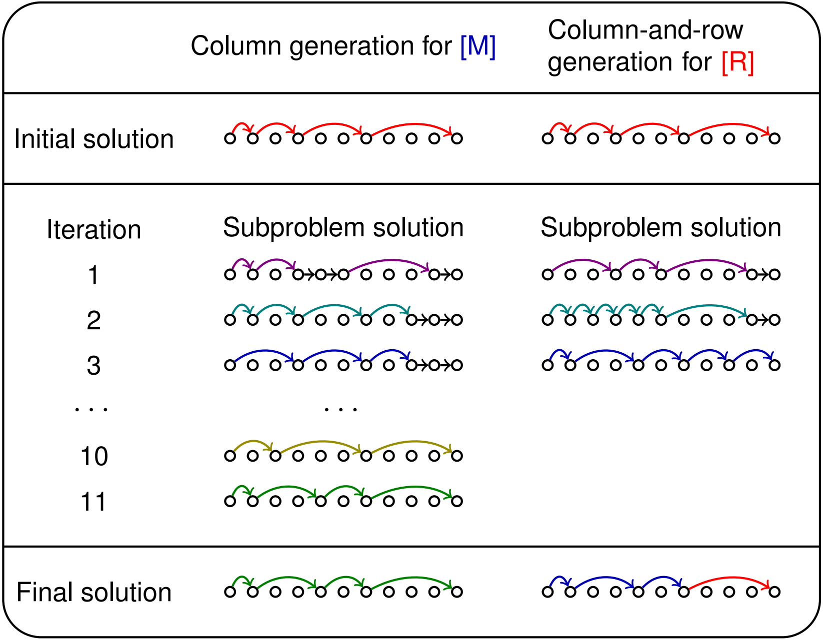
Either the solution of the restricted formulation $[\overline{R}_{LP}]$ is optimal for $[R_{LP}]$ or some variables x forming the pricing subproblem solution have negative reduced cost in $[\overline{R}_{LP}]$.

Recombination property



Column-and-row generation converges faster than the standard column generation.

Example of convergence



Column-and-row generation in general

Assumption

There exists a polyhedron which describes the convex hull of the subproblem.
Special case is the Dantzig-Wolfe reformulation: one variable per feasible solution.

Other applications in the literature

- Multi-commodity capacitated network design **[Frangioni, Gendron DAM09, WP10]**,
- Multi-commodity network flow **[Jones, Lustig, Farvoden MP93]**
- Bin-packing **[Valerio de Carvalho, AOR99]**
- Vehicle routing with split delivery **[Feillet, Dejax, Gendreau, Gueguen WP06]**

Contents

Numerical experiments setup

Instances

- Weighted total tardiness criterion
- Generated the same way as the instances from the OR-library
- 25 instances for each size
- Trivial instances are removed

Statistics

#sp number of pricing subproblems solved

vars percentage of variables x generated

cpu solution time, in seconds

Numerical results: one machine

		Cplex 12.1 for [R_{LP}]	Col. gen. for [M_{LP}]		Col.-and-row gen. for [R_{LP}]		
$p_{\max}$	n	*cpu*	*#sp*	*cpu*	*#sp*	*vars*	*cpu*
			1 machine				
50	25	2.9	352	1.7	54	8.9%	0.8
50	50	34.8	1559	41.7	82	6.7%	5.9
50	100	381.7	9723	2531.0	112	6.1%	47.3
100	25	11.3	378	2.3	75	5.9%	1.6
100	50	155.4	1418	44.3	114	4.6%	18.4
100	100	2039.6	10375	3436.3	155	4.5%	182.3

Numerical results: parallel machines

		Cplex 12.1 for [R_{LP}]	Col. gen. for [M_{LP}]		Col.-and-row gen. for [R_{LP}]		
p_{max}	*n*	*cpu*	*#sp*	*cpu*	*#sp*	*vars*	*cpu*
2 machines							
100	25	7.2	208	0.7	62	5.4%	0.5
100	50	198.6	641	10.0	93	4.5%	3.1
100	100	4038.4	2697	198.2	115	4.5%	30.5
4 machines							
100	50	35.1	441	3.4	90	4.4%	1.5
100	100	726.1	1353	47.0	113	4.3%	10.7
100	200	22441.6	4306	684.7	151	3.1%	80.2

Implementation alternatives to try

- Enumeration
- Additional stabilization
- Removing unnecessary variables
- Disaggregation to sub-paths instead of arcs
- Arc-indexed formulation ($x_{ijt} = 1$ iff job j is immediately preceded by job i at time t)