Column Generation for Extended Formulations

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

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

ISMP 2012
Berlin, August 23

Contents

Motivation

Methodology

Interest of the approach

Numerical results and conclusions

Extended formulations

Reformulation involving extra variables

$\Downarrow$

tighter relations between variables

Ways to obtain

- Variable Splitting (binary or unary expansion)
- Network Flow (Multi-Commodity)
- Dynamic Programming Solver **[Martin et al]**
- Union of Polyhedra **[Balas]**
- Polyhedral Branching Systems **[Kaibel, Loos]**
- ...

Ways to exploit extended formulations

1. Use a direct MIP-solver approach: size is an issue.

2. Use projection tools: Benders' cuts.
   $\rightarrow$ dynamic outer approximation of the formulation

3. Use of an approximation **[Van Vyve & Wolsey MP06]**
   - Drop some of the constraints
   - Aggregate commodities
   - Partial reformulation

   $\rightarrow$ static outer approximation of the formulation

4. Use (delayed) column generation.
   $\rightarrow$ dynamic inner approximation of the formulation

Column-and-row generation

It is a generalization of the standard column generation (based on the Dantzig-Wolfe reformulation).

Our contributions

- Reviewing of the methodology of the column-and-row generation and presenting it as a generic approach
- Analysis of the interest of the column-and-row generation approach: its good performance is explained by a stabilization effect
- New computational results

Contents

Motivation

Methodology

Interest of the approach

Numerical results and conclusions

Extended formulation for a subsystem

Original formulation

$$[\mathrm{F}] \equiv \min \left\{ c\,x \quad \begin{array}{rcl} A\,x & \geq & a \\ B\,x & \geq & b \\ x & \in & \mathbb{Z}^n_+ \end{array} \right\}$$

Subsystem

$$\mathrm{P} \equiv \left\{ \begin{array}{rcl} B\,x & \geq & b \\ x & \in & \mathbb{R}^n_+ \end{array} \right\}$$

$$X = \mathrm{P} \cap \mathbb{Z}^n$$

Main assumption

There exists a polyhedron

$$Q = \left\{ Hz \geq h, z \in \mathbb{R}^e_+ \right\}$$

and transformation T s.t. Q defines an **extended formulation** for X:

$$\mathrm{conv}(X) = \mathrm{proj}_x Q = \left\{ x = Tz : Hz \geq h, z \in \mathbb{R}^e_+ \right\}$$

Extended reformulation

Original formulation

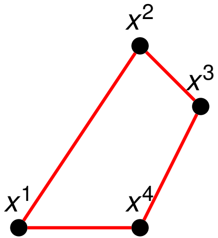
$$[F] \equiv \min\left\{ c\,x \quad \begin{array}{rcl} A\,x & \geq & a \\ B\,x & \geq & b \\ x & \in & \mathbb{Z}^n_+ \end{array} \right\}$$

Extended reformulation

$$[R] \equiv \min\left\{ c\,T\,z \quad \begin{array}{rcl} A\,T\,z & \geq & a \\ H\,z & \geq & h \\ z & \in & \mathbb{Z}^e_+ \end{array} \right\}$$

Special case: Dantzig-Wolfe reformulation

$$[M] \equiv \min\left\{ \sum_{g \in G} c\,x^g\,\lambda_g \quad \begin{array}{rcl} \sum_{g \in G} A\,x^g\,\lambda_g & \geq & a \\ \sum_{g \in G} \lambda_g & = & 1 \\ \lambda & \in & \{0,1\}^{|G|} \end{array} \right\}$$



Column-and-row generation: a hybrid approach

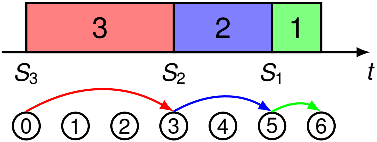
Alternative to direct resolution by a MIP solver

- Dynamic generation of the variables of [R]:
  generated in bunch by optimizing over X.
- Adding rows that become active.

Alternative to the standard column generation

- Perform the column generation for [M]
- “Project” the master program in [R]
  (we “split” generated columns into individual variables)

Example: machine scheduling with a sum criterion



$$\min \left\{ \sum_j c(S_j) \right.$$

$$\left. \begin{array}{rl} & S_j + p_j \leq S_i \\ \text{or} & S_i + p_i \leq S_j \end{array} \quad \forall (i,j) \right\}$$

$$[\text{R}] \equiv \min \left\{ \sum_{jt} c_{jt}\, z_{jt} \right.$$

$$\sum_{t=0}^{T-p_j} z_{jt} = 1 \quad \forall j \in J$$

$$\sum_{j \in J} z_{j0} = 1$$

$$\sum_{j \in J} z_{jt} - \sum_{j \in J} z_{j,t-p_j} = 0 \quad \forall t \geq 1$$

$$\left. z_{jt} \in \{0,1\} \quad \forall j,t \right\}$$

$$[\text{M}] \equiv \min \left\{ \sum_{g \in G} c^g \lambda_g \right.$$

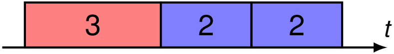
$$\sum_{g \in G} \sum_{t=0}^{T-p_j} z_{jt}^g \lambda_g = 1 \quad \forall j \in J$$

$$\sum_{g \in G} \lambda_g = 1$$

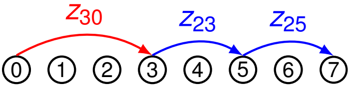
$$\left. \lambda_g \in \{0,1\} \ \forall g \in G \right\}$$

Machine scheduling: column-and-row generation

1. Solve the restricted extended formulation $[\overline{R}_{LP}]$ (start from a feasible one) and update dual prices.
2. Solve the pricing subproblem (obtain a pseudo schedule)



3. Disaggregate the subproblem solution in arc variables z.



4. If some of these variables z are not in $[\overline{R}_{LP}]$, add them to it along with the associated flow conservation constraints, then go to step 1.
5. Otherwise stop (the current solution of $[\overline{R}_{LP}]$ is optimal for [R]).

Restricted reformulations

$Z = \{z^s\}_{s \in S}$ — a set of integer solutions of Q, $\overline{S} \subset S$
$\overline{z}$ — restriction of z to the components of $\bigcup_{s \in \overline{S}} \mathrm{supp}(z^s)$
$\overline{G} = G(\overline{S}) = \{g \in G : x^g = T\, z^s, s \in \overline{S}\}$

$$[\overline{R}_{LP}] \equiv \min \left\{ c\, \overline{T}\, \overline{z} \quad \begin{array}{rcl} A\, \overline{T}\, \overline{z} & \geq & a \\ \overline{H}\, \overline{z} & \geq & \overline{h} \\ \overline{z} & \in & \mathbb{R}_+^{\overline{e}} \end{array} \right\}$$

$$[\overline{M}_{LP}] \equiv \min \left\{ \sum_{g \in \overline{G}} c\, x^g\, \lambda_g \quad \begin{array}{rcl} \sum_{g \in \overline{G}} A\, x^g\, \lambda_g & \geq & a \\ \sum_{g \in \overline{G}} \lambda_g & = & 1 \\ \lambda & \in & \mathbb{R}_+^{|\overline{G}|} \end{array} \right\}$$

Proposition

$$v^{[M_{LP}]} = v^{[R_{LP}]} \leq v^{[\overline{R}_{LP}]} \leq v^{[\overline{M}_{LP}]}.$$

Column-and-row generation procedure

Step 0: Initialize the dual bound, $\beta := -\infty$, and a subset $\overline{S}$ so that $[\overline{R}_{LP}]$ is feasible.

Step 1: Solve $[\overline{R}_{LP}]$ and collect its dual solution $\overline{\pi}$ associated to constraints $A\,\overline{T}\,\overline{z} \geq a$.

Step 2: Obtain a solution z^* of the pricing problem:

$$\min\{(c-\overline{\pi}A)Tz :\ z \in Z\} = \min\{(c-\overline{\pi}A)x :\ x \in X\}.$$

Step 3: Compute the Lagrangian dual bound: $L(\overline{\pi}) \leftarrow \overline{\pi}\,a + (c - \overline{\pi}A)\;T\;z^*$, and update $\beta \leftarrow \max\{\beta, L(\overline{\pi})\}$. If $v^{[\overline{R}_{LP}]} \leq \beta$, STOP.

Step 4: Update the current bundle $\overline{S}$ by adding solution z^* and update $[\overline{R}_{LP}]$. Go to Step 1.

Proposition

Either $v^{[\overline{R}_{LP}]} \leq \beta$ (stopping condition), or some of the components of z^* have negative reduced cost in $[\overline{R}_{LP}]$.

Example: multi-item multi-echelon lot sizing

y_{et}^{k} — setup for item k at echelon e in period t
x_{et}^{k} — production for item k at echelon e in period t

$$
[F] \equiv \quad \min \Big\{ \quad \sum_{ket} (c_{et}^{k}\, x_{et}^{k} + f_{et}^{k}\, y_{et}^{k}) :
$$

$$
\sum_{k} y_{et}^{k} \le 1 \quad \forall e, t
$$

$$
\sum_{\tau=1}^{t} x_{e\tau}^{k} \ge \sum_{\tau=1}^{t} x_{e+1,\tau}^{k} \quad \forall k, e < E, t
$$

$$
\sum_{\tau=1}^{t} x_{E\tau}^{k} \ge D_{1t}^{k} \quad \forall k, t
$$

$$
x_{et}^{k} \le D_{tT}^{k}\, y_{et}^{k} \quad \forall k, e, t
$$

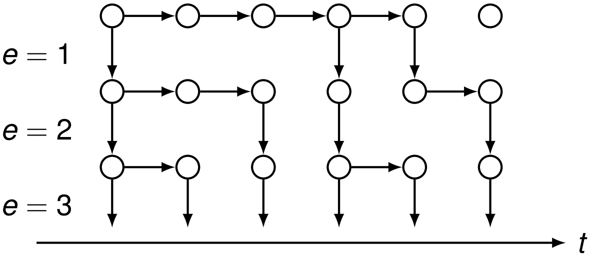
$$
x_{et}^{k} \ge 0 \quad \forall k, e, t
$$

$$
y_{et}^{k} \in \{0, 1\} \quad \forall k, e, t \Big\}
$$

Multi-echelon lot sizing: extended formulation

Dominance property

There exists an optimal solution in which $x_{et} \cdot s_{et} = 0 \ \forall k, e, t \Rightarrow$ production plan for every item k is a directed tree:



Dynamic programming

State (e, t, a, b) corresponds to accumulating at echelon e in period t a production covering exactly the demand of periods $a, \dots, b$. Extended formulation follows from **[Martin et al]**.

A generalization

Relaxed assumption

There exists a polyhedron

$$Q = \left\{ Hz \geq h, z \in \mathbb{R}^e_+ \right\}$$

and transformation T s.t. Q defines a **tighter formulation** for X:

$$\mathrm{conv}(X) \subset \mathrm{proj}_x Q = \left\{ x = Tz : Hz \geq h, z \in \mathbb{R}^e_+ \right\} \subset \mathsf{P}$$

Consequences

- Column-and-row procedure is still valid
- However, in general, the dual bound is not as tight as $v^{[M_{LP}]}$.

Contents

Motivation

Methodology

Interest of the approach

Numerical results and conclusions

Column-and-row generation vs. column generation

Proposition reminder

$$v^{[M_{LP}]} = v^{[R_{LP}]} \leq v^{[\overline{R}_{LP}]} \leq v^{[\overline{M}_{LP}]}.$$

Remark

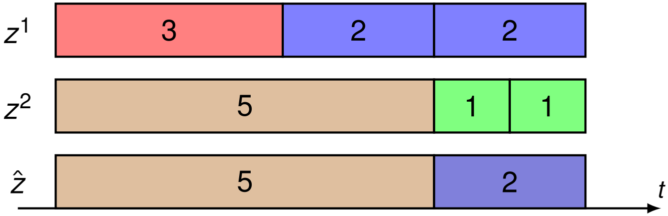
Column-and-row generation can converge faster than the standard column generation.
But when (and why) this happens?

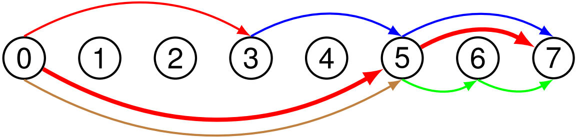
Recombination property

Given $\overline{S}$, subproblem solutions $z^1, \ldots, z^k \in Z(\overline{S})$ can be recombined in a new solution $\hat{z} \in [\overline{R}_{LP}]$ such that $\hat{z} \notin \mathrm{conv}(Z(\overline{S}))$.

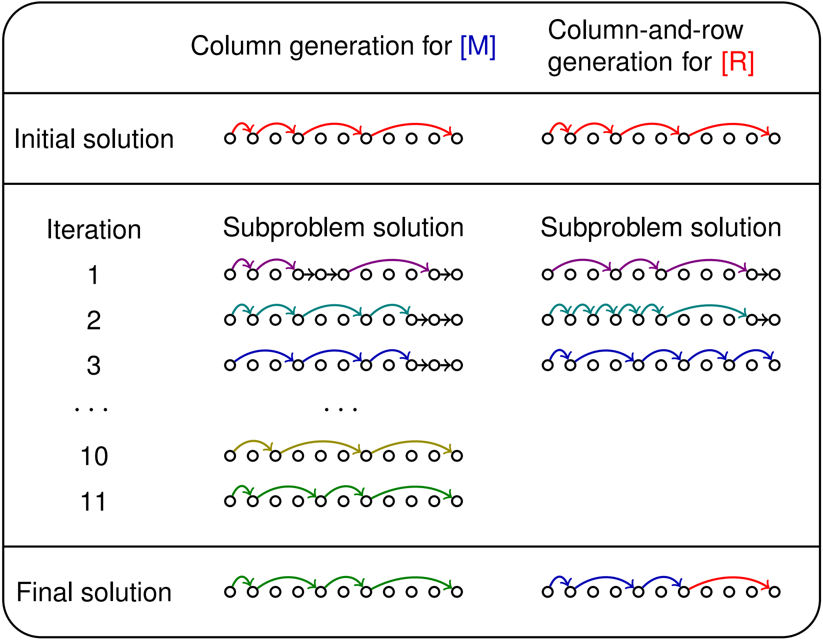
Machine scheduling: recombination property

$$Z(\overline{S}) = \{z^1, z^2\}, \qquad \hat{z} \in [\overline{R}_{LP}]$$



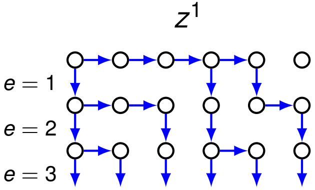


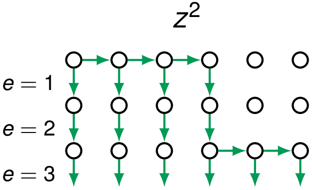
Machine scheduling: example of convergence

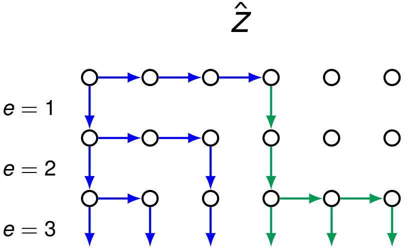


Multi-echelon lot-sizing: recombination property

$$Z(\overline{S}) = \{z^1, z^2\}, \qquad \hat{z} \in [\overline{R}_{LP}]$$







Contents

Motivation

Methodology

Interest of the approach

Numerical results and conclusions

Machine Scheduling: numerical results

- Generated similarly to the instances from the OR-library
- Averages for 25 instances are given
- Processing times are in $[1, \ldots, 100]$.

		Cplex 12.1 for [R_{LP}]	Colomn gen. for [M_{LP}]		Column-and-row generation for [R_{LP}]		
m	*n*	*cpu*	*#it*	*cpu*	*#it*	*vars*	*cpu*
1	25	7.1	337	0.9	124	3.8%	0.8
1	50	132.6	1274	24.2	246	2.7%	8.6
1	100	2332.0	8907	1764.4	455	1.9%	61.3
2	25	4.1	207	0.3	97	3.9%	0.2
2	50	109.2	645	5.7	173	2.8%	1.9
2	100	3564.4	2678	115.5	319	2.1%	14.9
4	50	18.7	433	1.5	167	3.0%	0.7
4	100	485.7	1347	27.9	295	2.2%	5.2
4	200	$>$2h	4315	409.4	561	1.5%	39.4

#it number of column generation iterations

vars percentage of variables z generated

cpu solution time, in seconds

Machine Scheduling: results with smoothing

Both column and column-and-row generation are stabilized with smoothing: pricing problem is solved for the vector of dual values which is a linear combination of current dual solution and the stability center (smoothing parameter α is the best possible).

		Colomn gen. for [M_{LP}], $\alpha = 0.9$		Column-and-row gen. for [R_{LP}], $\alpha = 0.5$		
m	*n*	*#it*	*cpu*	*#it*	*vars*	*cpu*
1	25	150	0.2	96	2.6%	0.4
1	50	354	3.8	172	1.7%	4.0
1	100	781	39.5	299	1.3%	31.1
2	25	142	0.2	87	3.3%	0.2
2	50	323	1.7	158	2.2%	1.6
2	100	715	17.3	275	1.6%	11.3
4	50	287	0.6	154	2.6%	0.6
4	100	638	8.7	264	1.8%	4.6
4	200	1553	87.7	481	1.2%	33.4

Multi-echelon lot sizing: results with smoothing

Averages for 10 instances are given

			Colomn gen. for $[M_{LP}]$, $\alpha = 0.85$		Column-and-row gen. for $[R_{LP}]$, $\alpha = 0.4$		
E	*K*	*T*	*#it*	*cpu*	*#it*	*vars*	*cpu*
2	10	50	126	1.7	29	0.57%	1.6
2	20	50	79	1.8	27	0.44%	3.1
2	10	100	332	38.0	43	0.15%	8.1
2	20	100	232	31.5	38	0.14%	20.0
3	10	50	187	11.8	38	0.16%	5.5
3	20	50	112	12.0	33	0.12%	9.8
3	10	100	509	454.5	49	0.02%	36.4
3	20	100	362	520.4	48	0.02%	103.1
5	10	50	296	62.6	48	0.10%	16.3
5	20	50	223	66.8	42	0.07%	34.3
5	10	100	882	4855.9	61	0.01%	134.0
5	20	100	362	4657.8	56	0.01%	386.1

Conclusions

1. Column generation for an extended formulation is to be considered when:
   - The extended formulation is obtained using a decomposition.
   - SP solutions can be recombined into alternative ones.

2. The approach can be interpreted as a stabilization method for column generation:
   - disaggregation helps,
   - related to the use of exchange vectors,
   - combined effect with other stabilization techniques (e.g. smoothing).

3. Computational results (ours and in the literature) show that this can be a competitive approach.

Bin Packing: results with smoothing

- Bin capacity is 4000
- Item sizes are generated uniformly in intervals $[1000, 3000]$ ("a2"), $[1000, 1500]$ ("a3"), and $[800, 1300]$ ("a4")
- Averages for 5 instances are given

		Cplex 12.1 for [F]			Col. gen. for [M], $\alpha = 0.85$		Col-and-row gen. for [R], $\alpha = 0.85$		
class	*n*	*gap*	*%gap*	*cpu*	*#it*	*cpu*	*#it*	*cpu*	*vars*
"a2"	200	5.6	5.2	0.1	439	0.3	281	0.5	0.21
	400	8.6	4.0	0.8	1001	1.2	599	2.0	0.15
	800	6.6	1.6	10.4	2725	6.8	1331	12.2	0.13
"a3"	200	4.0	6.0	0.1	158	0.2	124	0.2	0.16
	400	8.6	6.4	0.6	298	0.7	192	0.8	0.10
	800	17.4	6.5	7.7	596	5.5	297	4.8	0.08
"a4"	200	0.8	1.5	0.1	400	0.8	253	1.0	0.27
	400	1.8	1.7	0.6	841	5.4	414	4.5	0.17
	800	2.8	1.3	5.8	1662	38.6	602	16.3	0.13

Generalized Assignment: results with smoothing

Instances from the OR-Library (class D)

		Cplex 12.1 for $[F_{LP}]$		Col. gen. for $[M_{LP}]$, $\alpha = 0.85$			Col-and-row gen for $[R_{LP}]$, $\alpha = 0.5$			
m	*n*	*%gap*	*cpu*	*#it*	*%gap*	*cpu*	*#it*	*%gap*	*cpu*	*vars*
20	100	1.17	0.05	201	0.09	1.4	31	0.40	1.3	2.1
10	100	0.55	0.03	229	0.10	1.2	33	0.35	1.1	1.9
5	100	0.26	0.01	295	0.05	2.2	35	0.20	1.1	1.6
20	200	0.28	0.10	358	0.02	11.9	37	0.17	8.1	1.2
10	200	0.17	0.05	448	0.04	24.6	38	0.14	7.7	1.0
5	200	0.07	0.02	637	0.02	70.5	34	0.07	6.8	0.9
40	400	0.15	0.51	591	0.03	131.1	41	0.11	80.9	0.8
20	400	0.09	0.23	696	0.03	407.1	41	0.08	65.9	0.6
10	400	0.04	0.11	909	0.01	1338.8	41	0.04	58.8	0.5