Improved Lower Bounds using lm-SRCs for the Capacitated Arc Routing Problem

Rafael Martinelli[†] Marcelo Malta[‡] Marcus Poggi[‡]

[†]Departamento de Engenharia Industrial – PUC-Rio

[‡]Departamento de Informática – PUC-Rio

Column Generation 2016

The Capacitated Arc Routing Problem

- Connected undirected graph $G = (V, E)$
- Costs $c : E \to \mathbb{Z}^+$
- Demands $d : E \to \mathbb{Z}^+$
- Set I containing k identical vehicles with capacity Q
- Depot vertex labeled 0
- Set $E_R = \{e \in E \,|\, d_e > 0\}$ of **required** edges

The Capacitated Arc Routing Problem

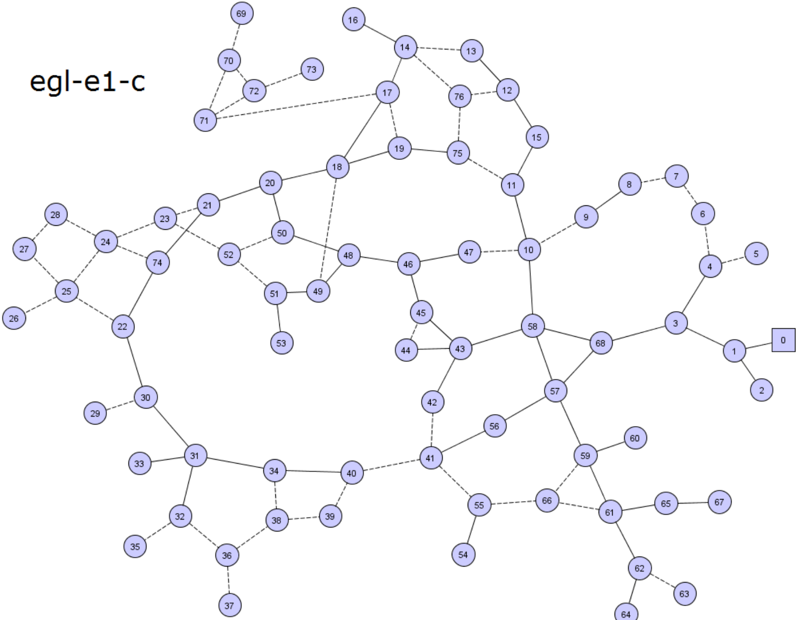
A set F of closed routes starting and ending at the depot is a feasible CARP solution if:

- Each required edge is serviced by exactly one route in F;
- The sum of demands of the serviced edges in each route in F does not exceed the vehicle capacity.

Edges in a route can be either *serviced* or *deadheaded*.

Goal

Find a feasible F solution minimizing the sum of the costs of the routes. Golden and Wong, 1981 egl-e1-c



69
16
14
13
70
73
72
17
76
12
71
15
19
75
18
11
20
8
7
21
6
28
23
50
9
24
52
48
47
10
27
74
46
4
5
25
51
49
45
3
26
22
58
0
53
44
43
68
1
57
2
42
30
29
56
60
31
59
33
34
40
41
55
66
65
67
61
32
38
39
35
36
54
62
63
37
64

The Capacitated Arc Routing Problem

- Golden and Wong, 1981
- Strongly NP-Hard

- Applications:
  - Garbage collection
  - Street sweeping
  - Winter gritting
  - Electric meter reading
  - Airline scheduling

- Hard to solve for more than 30 required edges

The Set Partitioning Approach

- The number of possible routes is exponentially large
- Dantzig-Wolfe decomposition of flow formulation
- This decomposition does not enforce the routes to be elementary

Mathematical Notation

- Ω – Set containing all possible routes
- λ_r – Binary variable, 1 if route r is used
- a_r^e – The number of times edge e is serviced by route r
- b_r^e – The number of times edge e is deadheaded by route r

The Set Partitioning Approach

Mathematical Formulation

$$
\begin{aligned}
\text{MIN} \quad & \sum_{r \in \Omega} c_r \lambda_r \\
\text{s.t.} \quad & \sum_{r \in \Omega} \lambda_r = k \\
& \sum_{r \in \Omega} a_r^e \lambda_r = 1 \quad \forall e \in E_R \\
& \lambda_r \in \{0, 1\} \quad \forall r \in \Omega
\end{aligned}
$$

Robust Cuts

Odd Degree Cutset Cuts

$$\sum_{r\in\Omega}\sum_{e\in\delta(S)} b_r^e\lambda_r \geq 1 \quad \forall S \subseteq V\backslash\{0\}, |\delta_R(S)| \text{ odd}$$

Capacity Cuts

$$\sum_{r\in\Omega}\sum_{e\in\delta(S)} b_r^e\lambda_r \geq 2k(S) - |\delta_R(S)| \quad \forall S \subseteq V\backslash\{0\}$$

Odd Degree Cutset Cuts Separation

- Use the exact algorithm proposed by Padberg and Rao in 1982
- The algorithm builds a Gomory-Hu tree (Gomory and Hu, 1961)
- It can be done in polynomial time running $|V| - 1$ times any max flow algorithm

Capacity Cuts Separation

- Use the exact algorithm proposed by Martinelli et al. in 2011
- Inspired on the exact separation of Chvátal-Gomory Cuts done by Fischetti and Lodi in 2007
- It uses a mixed-integer formulation to find a violated cut

Column Generation

Reduced Costs

$$\begin{aligned}
\bar{c}_r &= c_r - \gamma - \sum_{e \in E_R} a_r^e \beta_e - \sum_{S \subseteq V \setminus \{0\}} \sum_{e \in \delta(S)} b_r^e \pi_S \\
&= -\gamma + \sum_{e \in E_R} a_r^e \left(c_e - \beta_e\right) + \sum_{e \in E} b_r^e \left(c_e - \sum_{S \subseteq V \setminus \{0\} : e \in \delta(S)} \pi_S \right)
\end{aligned}$$

Column Generation

- Since an optimal solution to the CARP does not include routes which service a required more than once, ideally we want to price elementary routes.
- This corresponds to solve the Elementary Shortest Path Problem with Resource Constraints (ESPPRC) as a pricing subproblem.
- An alternative to deal with this complexity is to relax the elementarily constraint of the path, that means solving the Shortest Path Problem with Resource Constraints (SPPRC), also known as the *q-route problem*.
- The SPPRC can be tackled using a pseudo-polynomial dynamic programming algorithm, as described in the seminal work of Christofides et al.

Column Generation

- Aiming to have a better compromise between pricing efficiency and lower bounds, Baldacci, Mingozzi and Roberti proposed the ng-route relaxation.
- This relaxation defines for each required $e \in E_R$ a subset of requireds $N_e \subseteq E_R$ which have a relationship with the required e.
- A possible representation for this relationship can be a neighborhood relationship, i.e., N_e contains the nearest required edges of e.

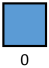
Column Generation

- Given a path $P = (0, \ldots, e_i, \ldots, e_p)$, let $E_R(P)$ be the set of required edges visited by P. A function $\Pi(P)$ of prohibited extensions for the path P can be defined as

$$\Pi(P) = \left\{ e_i \in E_R(P) : e_i \in \bigcap_{s=i+1}^{p} N_s, i = 1, \ldots, p-1 \right\} \cup \{e_p\}$$

Example 1

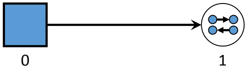
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$



$$\pi_0 = \{\}$$

Example 1

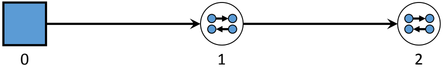
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$



$\pi_0 = \{\}$
$\pi_1 = \pi_0 \cap N_1 \cup \{1\}$
$\pi_1 = \{1\}$

Example 1

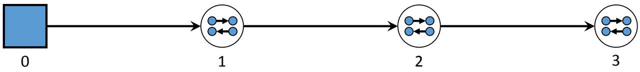
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$



$\pi_1 = \{1\}$
$\pi_2 = \pi_1 \cap N_2 \cup \{2\}$
$\pi_2 = \{1, 2\}$

Example 1

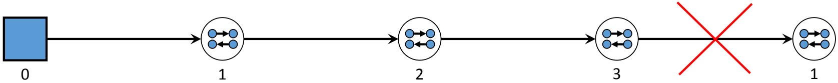
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$



$\pi_2 = \{1, 2\}$
$\pi_3 = \pi_2 \cap N_3 \cup \{3\}$
$\pi_3 = \{1, 3\}$

Example 1

$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$

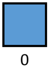


$\pi_3 = \{1, 3\}$

The extension is **not** allowed!

Example 2

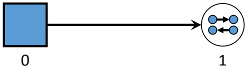
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 2\}$$



$$\pi_0 = \{\}$$

Example 2

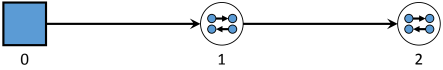
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 2\}$$



$$\pi_0 = \{\}$$
$$\pi_1 = \pi_0 \cap N_1 \cup \{1\}$$
$$\pi_1 = \{1\}$$

Example 2

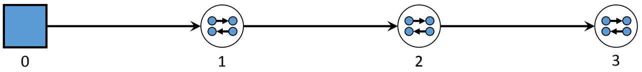
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 2\}$$



$$\pi_1 = \{1\}$$
$$\pi_2 = \pi_1 \cap N_2 \cup \{2\}$$
$$\pi_2 = \{1, 2\}$$

Example 2

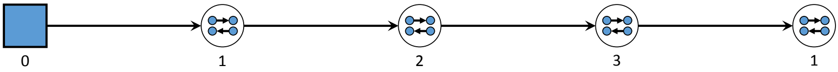
$$N_1 = \{1,2\},\ N_2 = \{2,1\},\ N_3 = \{3,2\}$$



$$\pi_2 = \{1,2\}$$
$$\pi_3 = \pi_2 \cap N_3 \cup \{3\}$$
$$\pi_3 = \{2,3\}$$

Example 2

$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 2\}$$



$\pi_3 = \{2, 3\}$

The extension is allowed.

Pricing

- This pricing is solved exactly using a forward dynamic programming algorithm
- The complexity is still pseudo-polynomial when the size of NGs is bounded by a constant factor
- But it still needs some speed up techniques:
  - Simple heuristic to find ng-routes with negative reduced cost
  - Dominance rules (pricing with elementary routes)

Heuristic Pricing

- Simple but effective heuristic very similar to the basic dynamic programming algorithm to compute ng-routes.
- During the dynamic programming algorithm, we store just the best possible partial ng-route for each capacity c and end required edge e.
- The resulting complexity of this algorithm is $\mathcal{O}(n^2Q)$.

Dominance Rule

- Given two paths P_1 and P_2, we say that P_1 dominates P_2 if:
  - $v(P_1) \leq v(P_2)$
  - $d(P_1) \leq d(P_2)$
  - $\bar{c}(P_1) \leq \bar{c}(P_2)$
  - $\Pi(P_1) \subseteq \Pi(P_2)$

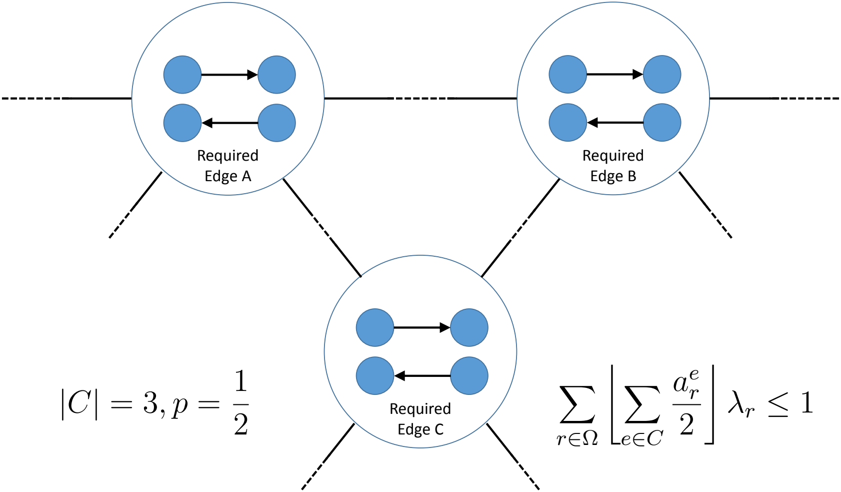
Subset-Row Cuts

- Introduced in 2008 by Jepsen et al.
- Lead to good improvements on lower bounds
- Their general form are Chvátal-Gomory Rank-1 Cuts

Given a subset of required edges $C \subseteq E_R$ and a multiplier $p \in \mathbb{R}$, $0 < p < 1$:

$$\sum_{r \in \Omega} \left\lfloor p \sum_{e \in C} a_r^e \right\rfloor \lambda_r \leq \lfloor p \, |C| \rfloor$$

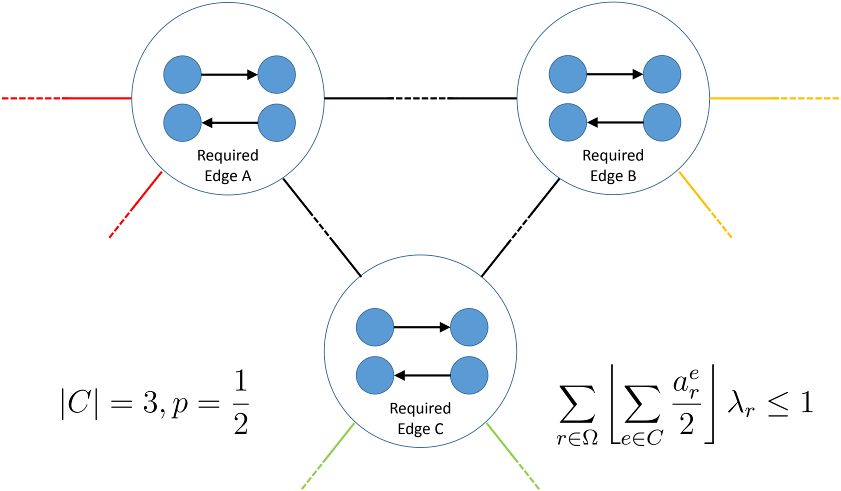
Subset-Row Cuts



$$|C| = 3, p = \frac{1}{2}$$

$$\sum_{r \in \Omega} \left\lfloor \sum_{e \in C} \frac{a_r^e}{2} \right\rfloor \lambda_r \leq 1$$

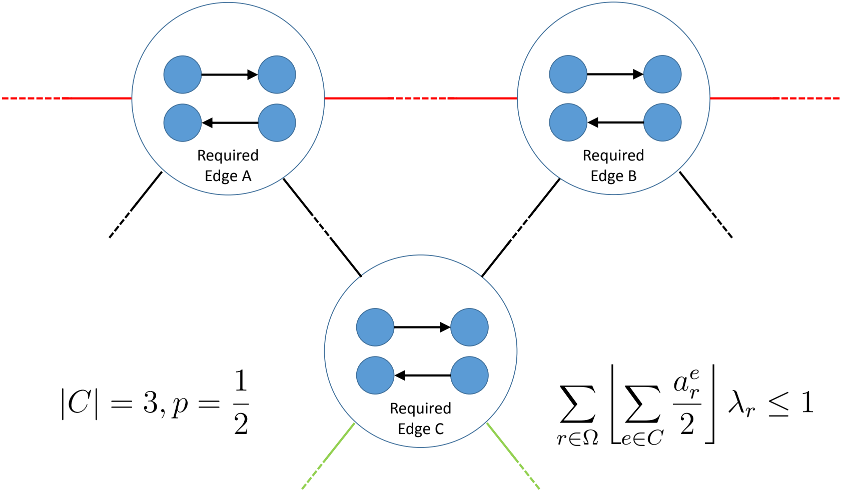
Subset-Row Cuts



$$|C| = 3, p = \frac{1}{2}$$

$$\sum_{r \in \Omega} \left\lfloor \sum_{e \in C} \frac{a_r^e}{2} \right\rfloor \lambda_r \leq 1$$

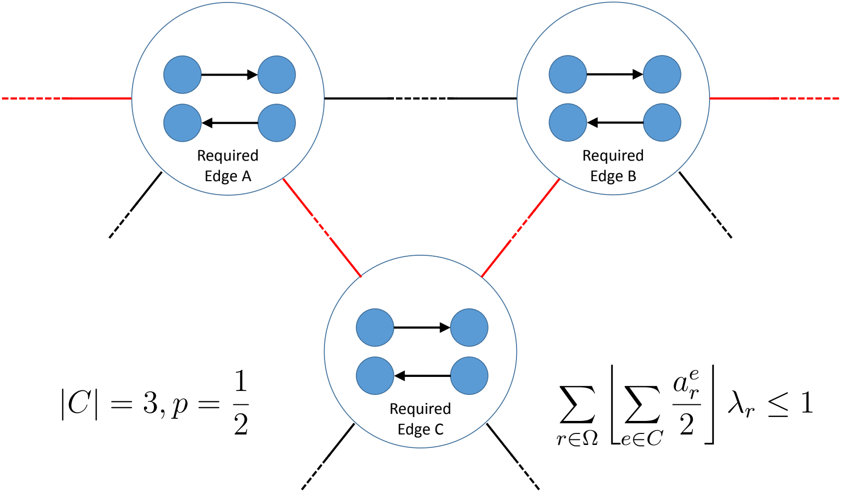
Subset-Row Cuts



$$|C| = 3, p = \frac{1}{2}$$

$$\sum_{r \in \Omega} \left\lfloor \sum_{e \in C} \frac{a_r^e}{2} \right\rfloor \lambda_r \leq 1$$

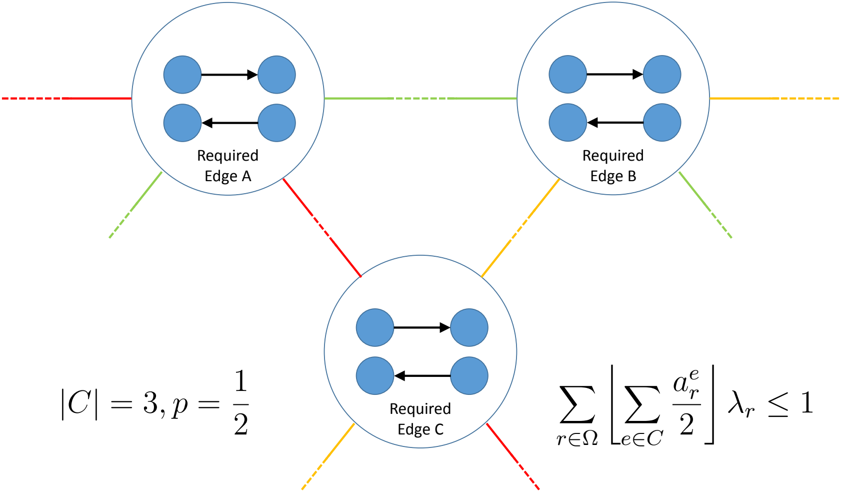
Subset-Row Cuts



$$|C| = 3, p = \frac{1}{2}$$

$$\sum_{r \in \Omega} \left\lfloor \sum_{e \in C} \frac{a_r^e}{2} \right\rfloor \lambda_r \leq 1$$

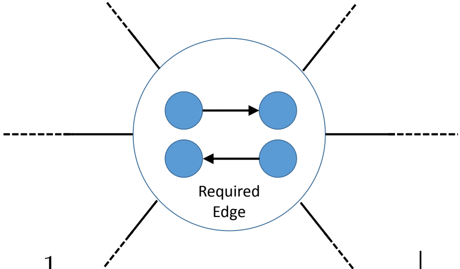
Subset-Row Cuts



$$|C| = 3, p = \frac{1}{2}$$

$$\sum_{r \in \Omega} \left\lfloor \sum_{e \in C} \frac{a_r^e}{2} \right\rfloor \lambda_r \leq 1$$

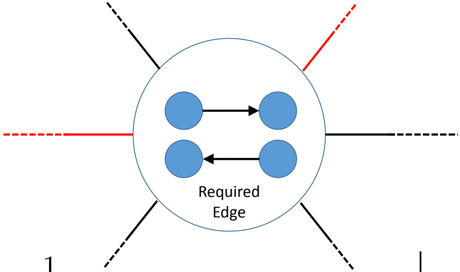
Subset-Row Cuts



$$|C| = 1, p = \frac{1}{2}$$

$$\sum_{r \in \Omega} \left\lfloor \sum_{e \in C} \frac{a_r^e}{2} \right\rfloor \lambda_r \leq 0$$

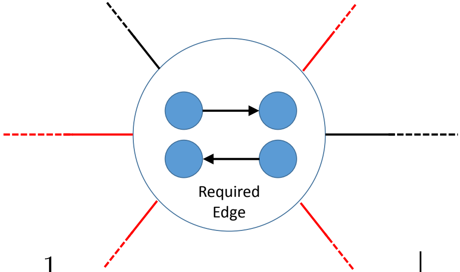
Subset-Row Cuts



$$|C| = 1, p = \frac{1}{2}$$

$$\sum_{r \in \Omega} \left\lfloor \sum_{e \in C} \frac{a_r^e}{2} \right\rfloor \lambda_r \leq 0$$

Subset-Row Cuts



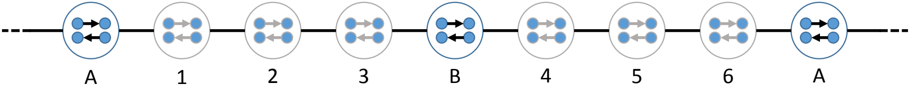
$$|C| = 1, p = \frac{1}{2}$$

$$\sum_{r\in\Omega} \left\lfloor \sum_{e\in C} \frac{a_r^e}{2} \right\rfloor \lambda_r \leq 0$$

Subset-Row Cuts

- They are hard to separate (usually based on enumeration)
- They are also hard to be considered in the pricing sub-problems
- Each SRC active in a SP solution introduces a new resource for the pricing sub-problem
- For the two examples, this is done using a counter sr_c on each label incremented every time a path visits a vertex of the cut
- When the counter reaches 2, the dual variable σ_c must be subtracted to the reduced cost and the counter is reset to 0

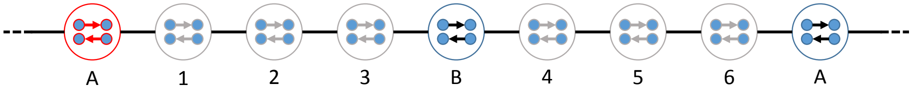
Subset-Row Cuts



$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 0$$

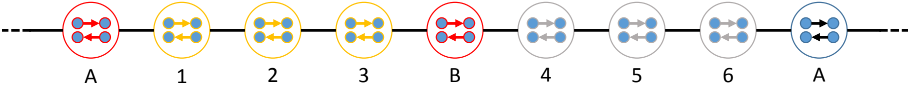
Subset-Row Cuts



$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 1$$

Subset-Row Cuts

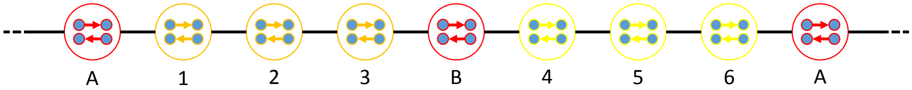


$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 0$$

$$\bar{c}_r = \bar{c}_r - \sigma_c$$

Subset-Row Cuts



$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 1$$

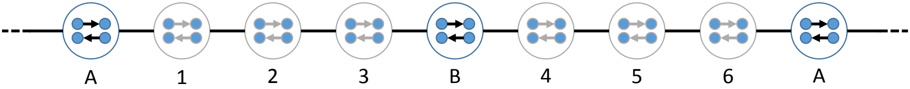
Subset-Row Cuts

- Changes to the dominance rule:
  - $v(P_1) \leq v(P_2)$
  - $d(P_1) \leq d(P_2)$
  - $\Pi(P_1) \subseteq \Pi(P_2)$
  - $\bar{c}(P_1) \leq \bar{c}(P_2) - \sum_{c \in C : sr_c(P_1) > sr_c(P_2)} \sigma_c$

Limited Memory Subset-Row Cuts

- Introduced in 2014 by Pecin et al.
- The intuition is analogous to the ng-route relaxation
- Each required edge has a memory set M_e containing the SRCs it "remembers"
- Every time a path P with $sr_c(P) > 0$ visits a required edge with $c \notin M_e$, the path "forgets" cut c by setting $sr_c = 0$
- For the two examples, the required edges which remember a cut are the ones between an odd visit to set C until the following visit, on any route in the current solution

Limited Memory Subset-Row Cuts

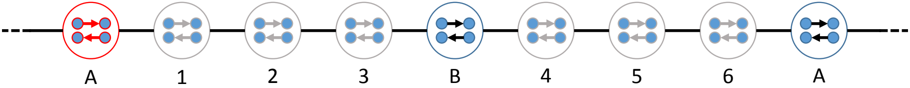


$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 0$$

$$cand = \{\}$$

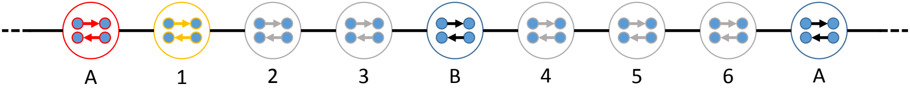
Limited Memory Subset-Row Cuts



$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 1$$
$$cand = \{\}$$

Limited Memory Subset-Row Cuts

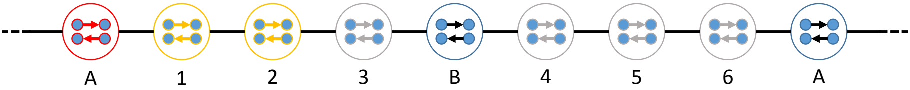


$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 1$$

$$cand = \{1\}$$

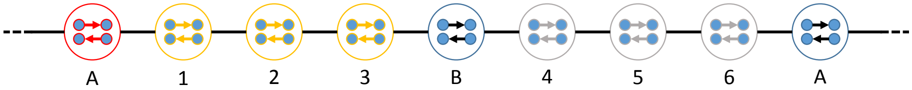
Limited Memory Subset-Row Cuts



$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 1$$
$$cand = \{1, 2\}$$

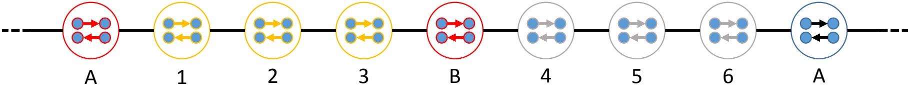
Limited Memory Subset-Row Cuts



$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 1$$
$$cand = \{1, 2, 3\}$$

Limited Memory Subset-Row Cuts



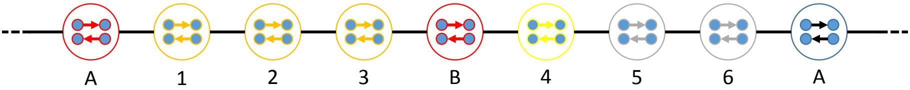
$$C = \{A, B, C\}, p = \frac{1}{2}$$

$sr_c = 0$

$cand = \{\}$

Add c to the memory of 1, 2 and 3

Limited Memory Subset-Row Cuts

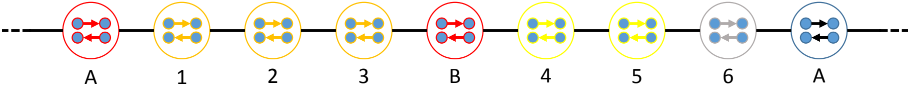


$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 0$$

$$cand = \{\}$$

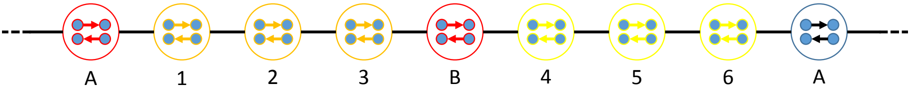
Limited Memory Subset-Row Cuts



$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 0$$
$$cand = \{\}$$

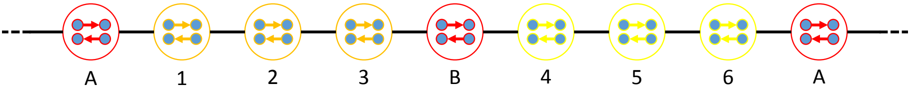
Limited Memory Subset-Row Cuts



$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 0$$
$$cand = \{\}$$

Limited Memory Subset-Row Cuts



$$C = \{A, B, C\}, p = \frac{1}{2}$$

$$sr_c = 1$$
$$cand = \{\}$$

Limited Memory Subset-Row Cuts

- Another example:
  - $C = \{1, 2, 3\}$
  - $r_1 = (0, 1, 4, 5, 3, 7, 4, 0)$
  - $r_2 = (0, 8, 2, 8, 6, 2, 8, 0)$

Limited Memory Subset-Row Cuts

- Another example:
  - $C = \{1, 2, 3\}$
  - $r_1 = (0, 1, 4, 5, 3, 7, 4, 0)$
  - $r_2 = (0, 8, 2, 8, 6, 2, 8, 0)$
- The cut will the added to the memory set M_e of required edges $e \in \{1, 2, 3\} \cup \{4, 5\} \cup \{8, 6\}$

Computational Experiments

- The algorithms were implemented in C++ using Microsoft Visual C++ 2013
- IBM ILOG CPLEX Optimizer 12.6.2 was used for solving the formulations
- The experiments were conducted on an Intel Core i7-3960X 3.30GHz with 64GB RAM running Ubuntu Linux Server 14.04
- We used the eglese instance set (Li and Eglese, 1992), containing 24 instances having from 51 to 190 required edges
- 12 instances are still open

Results

			ng=8		elem		1src		3src		1+3src	
Instance	LB	UB	value	time	value	time	value	time	value	time	value	time
egl-e1-A	**3548**	**3548**	3548.0	3.9	3548.0	3.8	3548.0	3.7	3548.0	3.8	3548.0	3.5
egl-e1-B	**4498**	**4498**	4472.5	1.8	4473.4	2.1	4473.3	3.3	4484.2	17.2	4484.5	16.3
egl-e1-C	**5595**	**5595**	5541.8	2.2	5544.8	3.3	5544.8	4.4	5562.0	324.4	5563.8	1547.3
egl-e2-A	**5018**	**5018**	5016.6	15.3	5016.9	441.4	5016.9	49.2	5018.0	32.2	5018.0	30.4
egl-e2-B	**6317**	**6317**	6291.7	4.9	6299.3	22.1	6299.2	25.9	6302.0	564.5	6302.1	329.5
egl-e2-C	**8335**	**8335**	8270.6	3.4	8274.4	5.0	8274.4	5.0	8302.7	842.9	8302.6	398.9
egl-e3-A	**5898**	**5898**	5896.1	12.6	5896.1	79.8	5896.1	35.0	5896.4	7200.0	5896.4	7200.0
egl-e3-B	7744	7775	7696.9	10.8	7704.6	21.1	7704.5	59.0	7734.7	7200.0	7735.0	7200.0
egl-e3-C	10244	10292	10180.0	4.6	10184.1	5.7	10183.8	13.2	10231.3	108.0	10232.7	174.3
egl-e4-A	6408	6444	6389.7	38.9	6394.3	546.5	6393.7	182.4	6406.9	7200.0	6406.8	7200.0
egl-e4-B	8935	8961	8884.8	13.0	8890.4	30.6	8890.1	31.8	8925.0	636.9	8926.8	1794.9
egl-e4-C	11512	11529	11465.3	8.0	11467.5	8.5	11467.2	10.3	11493.6	127.6	11493.5	101.3
egl-s1-A	**5018**	**5018**	5014.5	13.2	5014.5	84.9	5014.5	31.6	5018.0	36.2	5018.0	32.5
egl-s1-B	**6388**	**6388**	6377.4	7.0	6378.0	21.0	6377.6	9.4	6387.0	42.0	6386.8	41.3
egl-s1-C	**8518**	**8518**	8484.4	4.1	8487.2	9.3	8487.1	4.7	8504.4	8.9	8504.5	8.7
egl-s2-A	9838	9875	9800.4	96.6	9801.6	306.9	9801.2	190.5	9823.6	7200.0	9823.7	7200.0
egl-s2-B	13017	13057	12965.9	46.4	12978.6	182.6	12978.5	156.8	13009.7	451.6	13008.9	467.8
egl-s2-C	**16425**	**16425**	16347.9	18.9	16358.4	34.9	16358.2	28.6	16387.3	99.1	16387.8	83.7
egl-s3-A	10165	10201	10140.8	138.2	10147.9	5158.8	10147.6	753.0	10167.9	7200.0	10168.7	7086.8
egl-s3-B	13648	13682	13618.7	46.3	13623.5	113.5	13623.1	98.1	13643.2	2757.1	13643.2	2191.1
egl-s3-C	**17188**	**17188**	17096.8	22.9	17113.7	41.2	17113.2	40.2	17140.9	133.2	17142.7	138.6
egl-s4-A	12159	12216	12127.4	151.4	12137.1	889.7	12136.2	477.3	12158.7	4279.1	12159.4	7200.0
egl-s4-B	16114	16214	16065.5	85.8	16078.6	290.5	16078.2	383.7	16126.8	3545.0	16126.6	2805.2
egl-s4-C	20430	20461	20384.8	43.9	20397.2	77.3	20396.8	66.9	20429.5	284.2	20429.1	228.6

Obrigado!

Questions?