

# Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems

Stefan Røpke

DTU Transport

June 13th, 2012

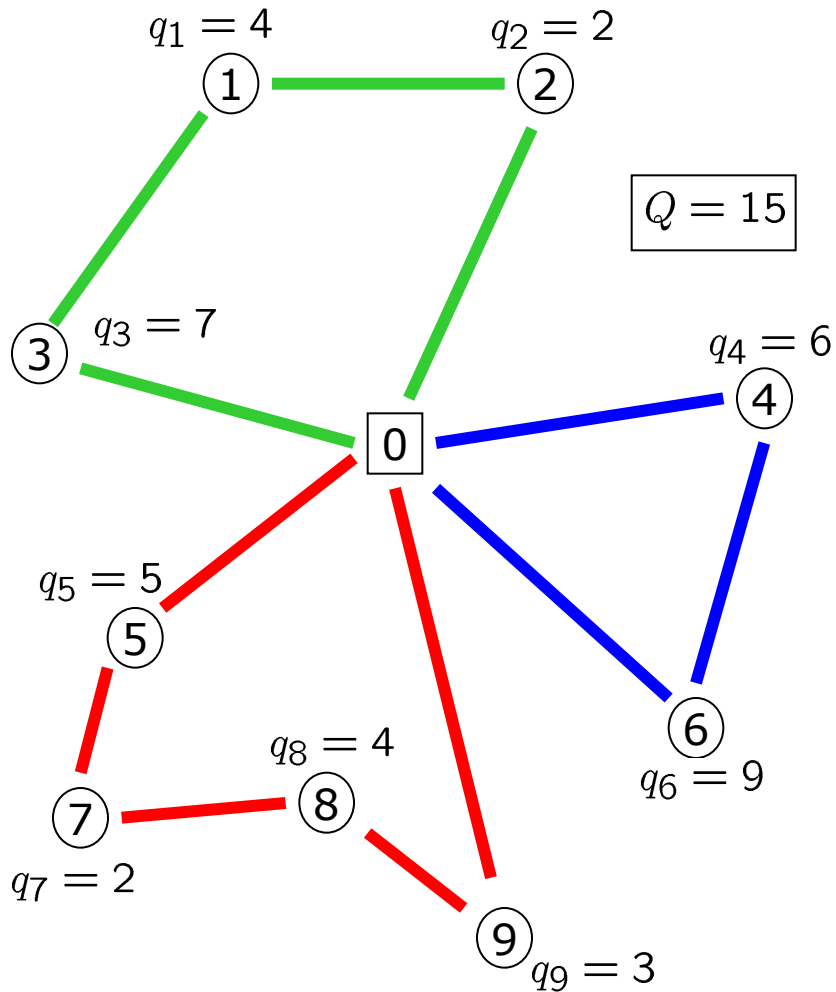
# **The Solomon instances are solved!**

Stefan Røpke

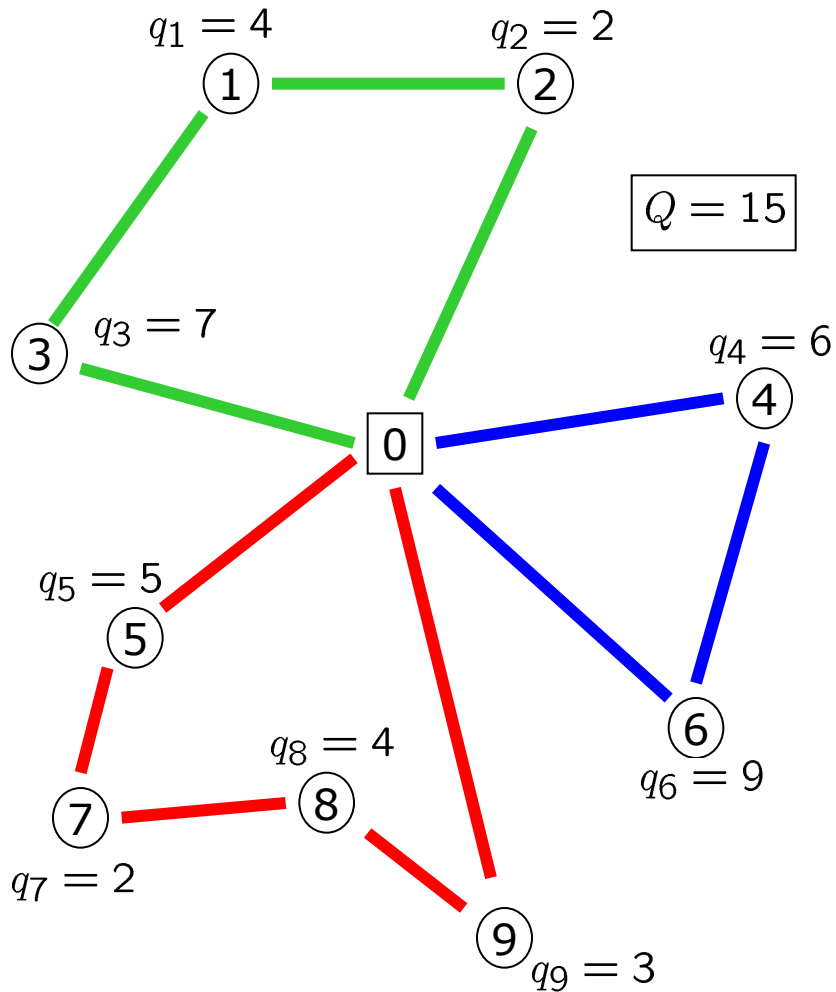
DTU Transport

June 13th, 2012

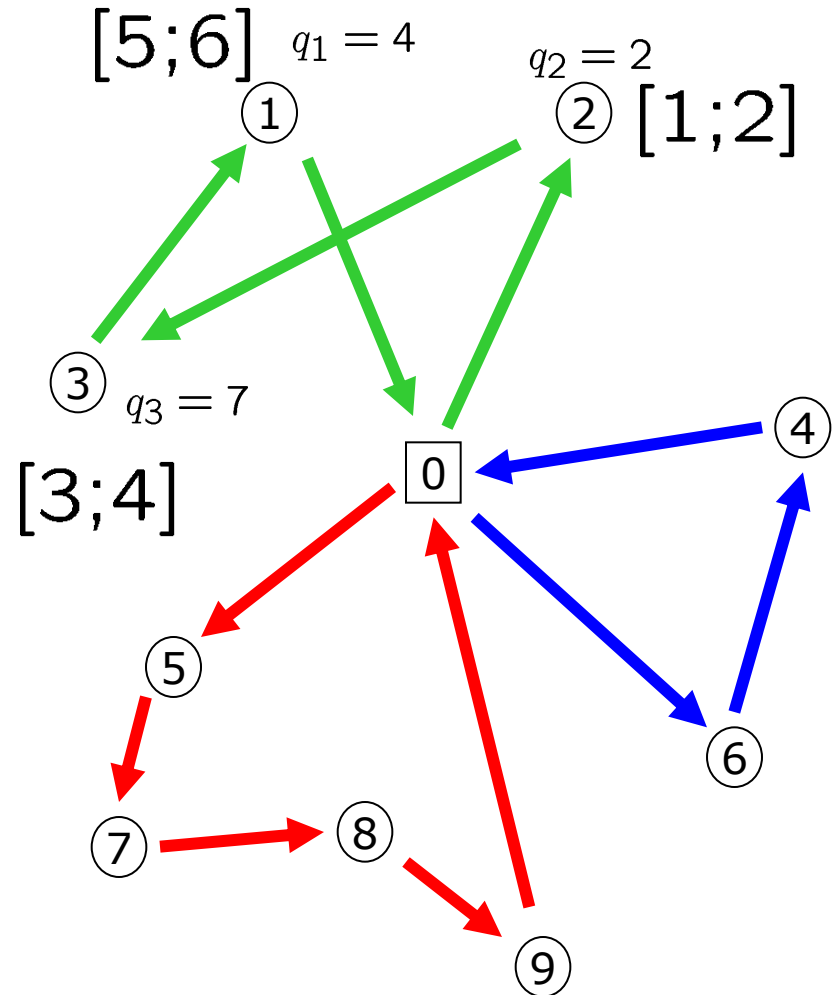
# Capacitated vehicle routing problem (CVRP)



# Capacitated vehicle routing problem (CVRP)



# Vehicle routing problem with time windows (VRPTW)



# The solomon instances

- A challenging set of instances for the VRPTW proposed in 1987. Consists of 56 instances each with 100 customers.
- Have gained enormous popularity.











[Algorithms for the vehicle routing and scheduling problems with time window constraints](#)

[MM Solomon - Operations research, 1987 - JSTOR](#)

This paper considers the design and analysis of algorithms for vehicle routing and scheduling problems with time window constraints. Given the intrinsic difficulty of this problem class, approximation methods seem to offer the most promise for practical size ...

[Citeret af 1594 - Relaterede artikler - Alle 13 versioner](#)



Authors	Nationality	Year	#Solved
Desrochers, Desrosiers, Solomon		1992	7
Kohl, Desrosiers, Madsen, Solomon, Soumis		1999	14
Larsen		1999	17
Irnich, Villeneuve		2003 (2006)	29
Feillet, Dejax, Gendreau, Gueguen		2004	17
Salani (Righini)		2004	11
Kallehauge, Larsen, Madsen		2006	25
Jepsen, Petersen, Spoorendonk, Pisinger		2008	45
Desaulniers, Lessard, Hadjar		2008	51
Baldacci, Mingozzi, Roberti		2011	55

# VRPTW set-partitioning model

- $V_c = \{1, \dots, n\}$ . Set of customers.
- $\Omega$  : set of all feasible VRPTW routes.
- $c_p$  : cost of route  $p \in \Omega$ .
- $a_{ip}$  : constant that is 1 if customer  $i$  is visited by route  $p$  and 0 otherwise.
- $y_p$  : binary variable that is 1 if and only if path  $p \in \Omega$  is used in the solution.

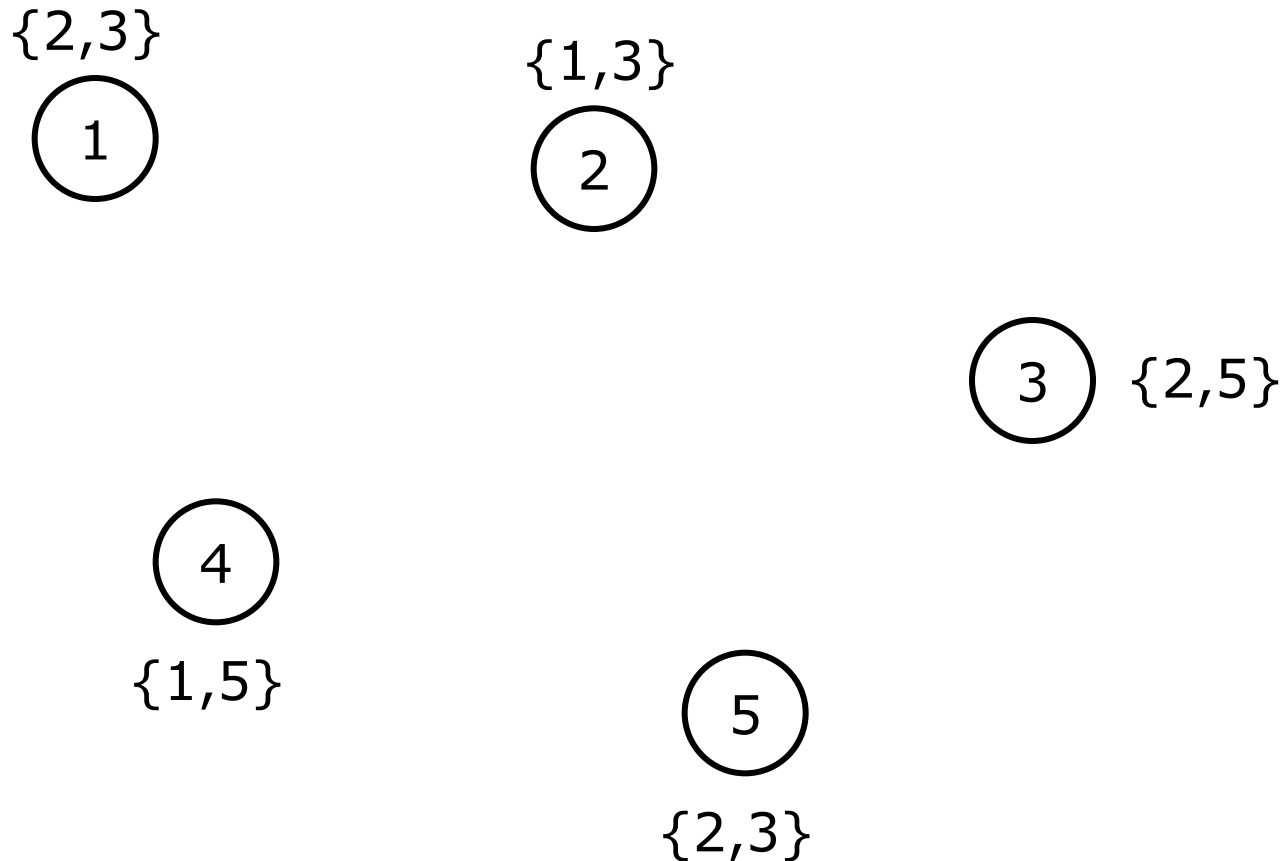
$$\min \sum_{p \in \Omega} c_p y_p$$

Subject to:

$$\sum_{p \in \Omega} a_{ip} y_p = 1 \quad \forall i \in V_c$$

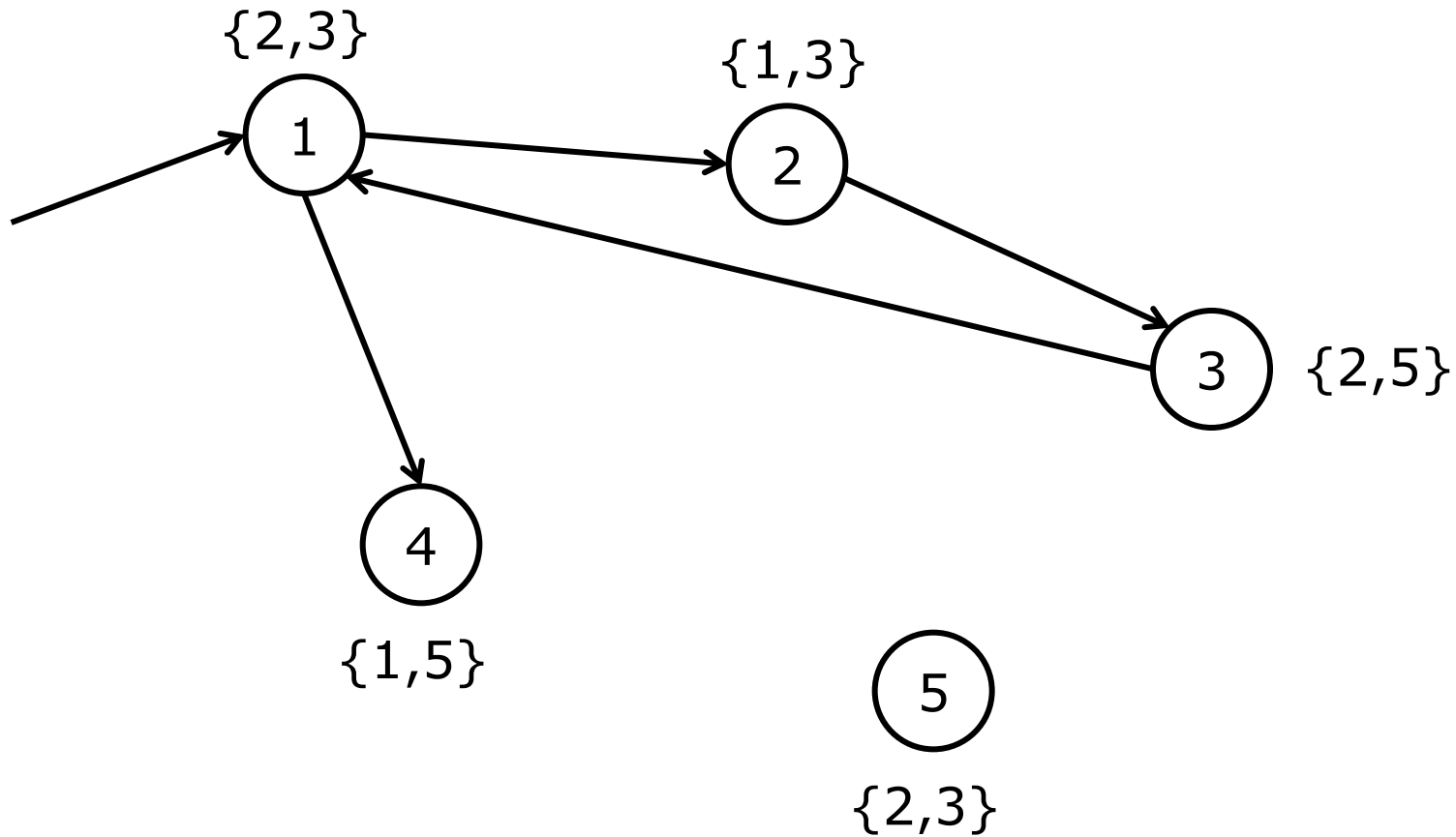
$$y_p \in \{0, 1\} \quad \forall p \in \Omega$$

# ng routes (Baldacci, Mingozzi, Roberti, 2011)

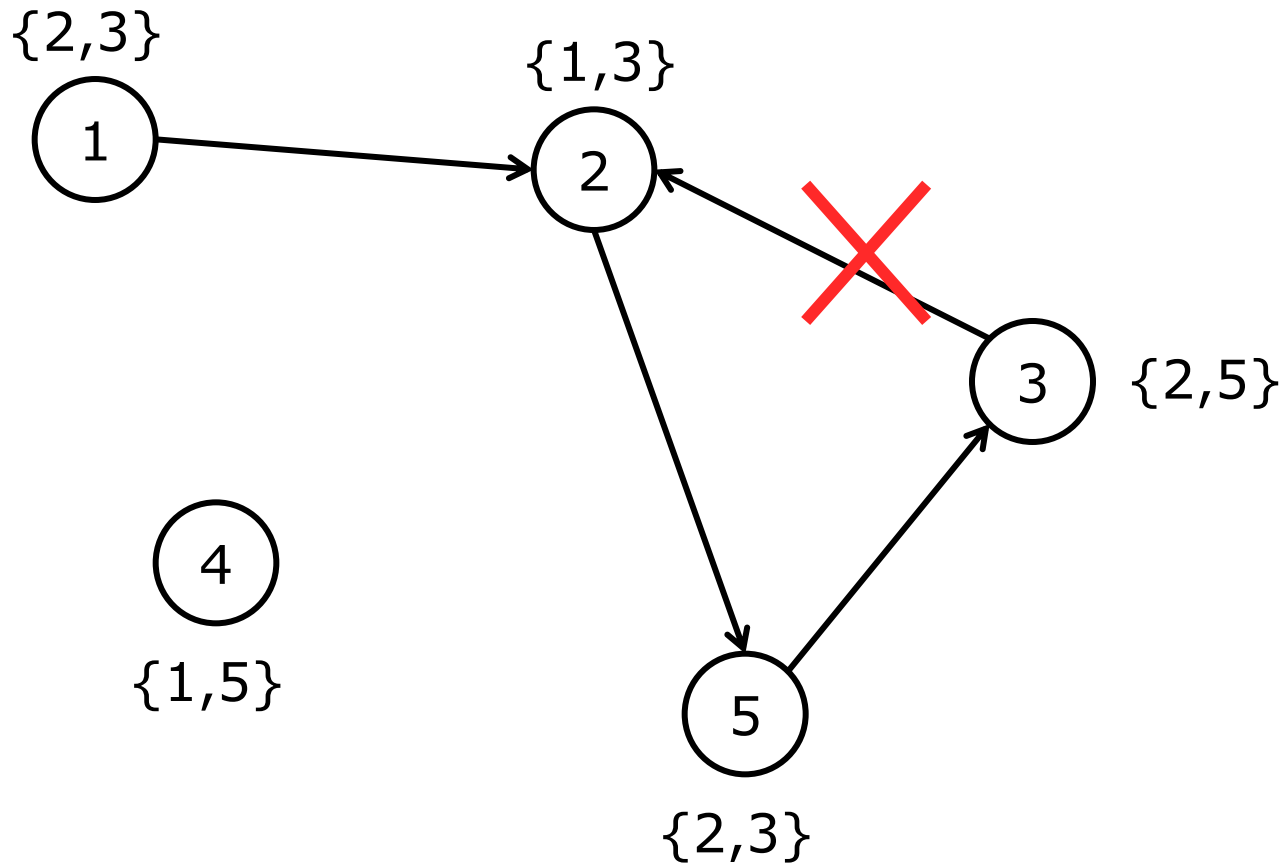




# ng routes



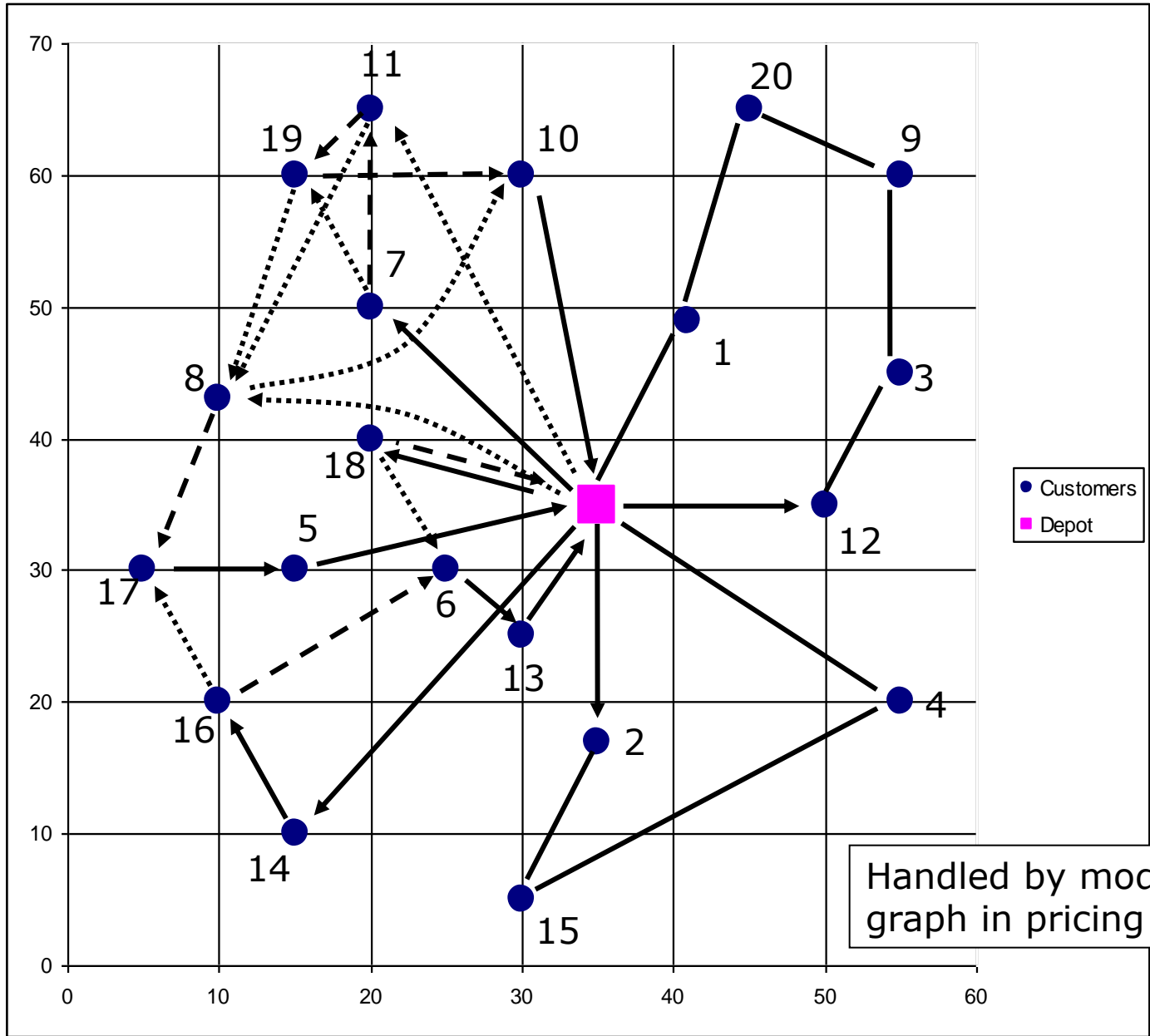
# ng routes



# ng routes

- Exact: Mono directional algorithm
- Heuristics:
  - Truncated version of the exact algorithm.
  - Simple tabu search algorithm.
- “Dirty trick” for VRPTW: Throw away capacity resource in the pricing problem and handle the capacity constraint in the master problem instead. Works because capacity constraints rarely are binding in VRPTW instances.

# Branch on arcs



- > Weight 1
- - - -> Weight 2/3
- .....> Weight 1/3

Branch on  $x(8,10)$

• Customers  
 ■ Depot

Handled by modifying feasibility graph in pricing problem

# Generalized upper bound (GUB ) branching

- Given a subset of binary variables indexed by  $T$  and a constraint

$$\sum_{i \in T} x_i = 1$$

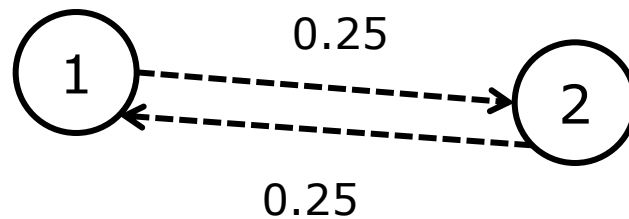
we can create a branch

$$\sum_{i \in T_1} x_i = 0 \quad \vee \quad \sum_{i \in T_2} x_i = 0$$

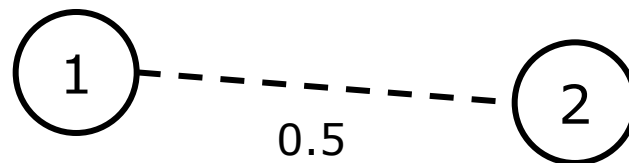
where  $T = T_1 \cup T_2$  and  $T_1 \cap T_2 = \emptyset$ .

- GUB constraints in VRPTW: the number of arcs entering or leaving a customer has to be equal to one.
- Branch is handled by making arcs in  $T_1$  or  $T_2$  infeasible in pricing problem.
- Generalizes branching on arcs.

# Branch on edges

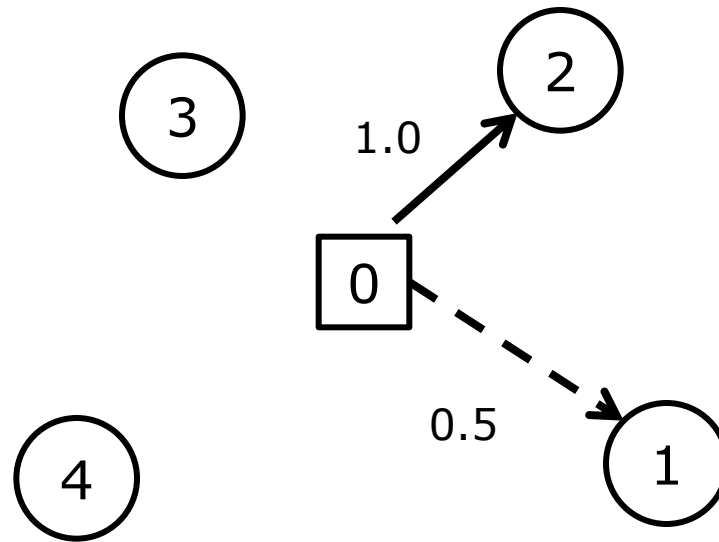


View as an edge



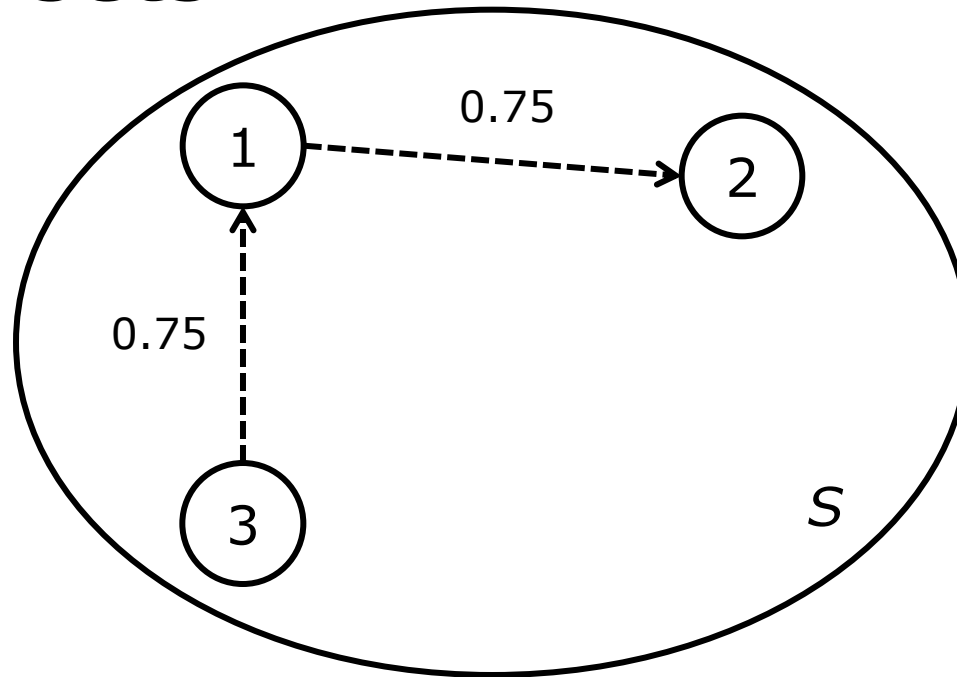
Enforce  $x_{12} + x_{21} = 0$  and  $x_{12} + x_{21} = 1$  in two branches (add as inequalities to the master problem).

# Branch on number of vehicles



Let  $x_{ij}^*$  be arc variables corresponding to current LP solution and let  $V_c$  be the set of customer nodes. Enforce  $\sum_{i \in V_c} x_{0i} \leq \lfloor \sum_{i \in V_c} x_{0i}^* \rfloor$  and  $\sum_{i \in V_c} x_{0i} \geq \lceil \sum_{i \in V_c} x_{0i}^* \rceil$  in two branches (add as inequalities to the master problem).

# Branch on sets



Enforce  $\sum_{i,j \in S, i \neq j} x_{ij} \leq \left\lfloor \sum_{i,j \in S, i \neq j} x_{ij}^* \right\rfloor$  and  $\sum_{i,j \in S, i \neq j} x_{ij} \geq \left\lceil \sum_{i,j \in S, i \neq j} x_{ij}^* \right\rceil$  in two branches (add as inequalities to the master problem). Generalizes both branching on edges and branching on vehicles.



Authors	Year	Branching
Desrochers, Desrosiers, Solomon	1992	Arcs, vehicles
Kohl, Desrosiers, Madsen, Solomon, Soumis	1999	Arcs, vehicles
Larsen	1999	Arcs, vehicles, time windows
Irnich, Villeneuve	2003 (2006)	Arcs, vehicles
Salani	2004	GUB
Kallehauge, Larsen, Madsen	2006	Arcs
Jepsen, Petersen, Spoorendonk, Pisinger	2008	Sets
Desaulniers, Lessard, Hadjar	2008	Arcs
Baldacci, Mingozzi, Roberti	2011	-

# Strong branching

- Assume branching on arcs.
- Which arc should we branch on?
- Strong branching: Select  $k$  candidates and evaluate all  $k$  branches. Go on with the best candidate.
- How to do this in a column generation algorithm
  - How much effort when evaluating candidates?
    1. Solve LP with existing columns and cuts?
    2. Also generate columns with heuristic pricing?
    3. Also generate cuts?
    4. Solve pricing problem to optimality?

# Strong branching

- How to choose among the  $k$  candidates?
- Let  $\Delta_1$  and  $\Delta_2$  be increase in lower bound in the two child nodes.
- $\text{score} = \alpha \min\{\Delta_1, \Delta_2\} + (1 - \alpha) \max\{\Delta_1, \Delta_2\}$  (Linderoth and Savelsbergh, 1999) or
- $\text{score} = \max\{\Delta_1, \epsilon\} \cdot \max\{\Delta_2, \epsilon\}$  (Achterberg, 2007)
- select branch with highest score.

# Speeding up strong branching

1. Evaluate all candidates by solving LPs using existing cuts and columns. This establishes upper bounds on  $\Delta_1$  and  $\Delta_2$ .
2. Calculate score for all candidates based on quick evaluation above.
3. Sort candidates in decreasing order according to score.
4. Perform full evaluation of candidate with best score. This establishes a lower bound  $s^*$  for the score that can be obtained.
5. For each remaining candidate calculate lower bounds  $\underline{\Delta}_1, \underline{\Delta}_2$  on  $\Delta_1$  and  $\Delta_2$  based based on  $s^*$  and upper bound on  $\Delta_1$  and  $\Delta_2$  obtained in step 1.
6. If  $\Delta_i < \underline{\Delta}_i$  for  $i$  equal 1 or 2 then we can skip candidate. If not we try to improve  $\underline{\Delta}_i$  by generating columns and stop if the value falls below  $\underline{\Delta}_i$ .

$$\text{score} = \alpha \min\{\Delta_1, \Delta_2\} + (1 - \alpha) \max\{\Delta_1, \Delta_2\}$$

# Strong branching

## Evaluating 30 candidates

	<b>Naive</b>	<b>Improved</b>	<b>Speedup</b>
RC108	19.1s	7.9s	2.4
R112	18.2s	4.2s	4.3
R206	39.6s	18.7s	2.1

# Strong branching in branch-and-price algorithms

Authors	Notes
Ralphs, 2003	7 candidates.
Fukasawa, Longo, Lysgaard, Poggi de Aragão, Reis, Uchoa, Werneck, 2006	Between 5 and 10 candidates
Irnich, 2010	Number of candidates depend on current lower bound
Martinelli, Pecin, Poggi de Aragão, Longo, 2011	3 candidates.

# Reliability branching

- Proposed by Achterberg, Koch, Martin, 2005
- Try to avoid expensive evaluations.
- For each variable, keep an estimate for change in lower bound when branching up and down.
- Estimates are initially uninitialized.
- When a variable needs to be evaluated we check how many full evaluations we have done on it (initially 0).
- If this number larger than a parameter (reliability parameter) then we use the estimated changes to calculate score. Otherwise we fully evaluate the branch and update estimates.

# Cuts applied

- CVRP:
  - capacity inequalities, strengthened comb inequalities, 2 edges hypo tour constraints, Homogeneous multistar constraints (Lysgaard, Letchford, Eglese, 2004).
- VRPTW:
  - capacity inequalities, strengthened comb inequalities (CVRP, Lysgaard, Letchford, Eglese, 2004)
  - 2-path inequalities (Kohl, Desrosiers, Madsen, Solomon, Soumis, 1999)
  - Tournament inequalities (ATSPTW, Ascheuer, Fischetti, Grötschel, 2001)
  - Generalized odd-cat inequalities (ATSP, Balas, 1989)
- Only cuts on the variables of the original formulation. **No cuts on master problem variables.**



# Impact of some of the options presented.

- "Tuning" test set (VRPTW), 50 instances. 20 are from the Solomon data set, 30 are randomly generated.
- Time limit 1800 seconds

# Branching on arcs

Candidates: 15/10/10

	<b>Normal branching</b>	<b>Strong branching</b>
Solved (of 50)	9	32
Average gap after branch and bound	0.99%	0.5%
Avg. time to opt (*)	849 s	229 s
BB Nodes (*)	796	91

(\*) only for instances that both configurations can solve.

# Branching on arcs

Candidates:

15/10/10

	<b>Strong branching <math>\alpha=0.99</math></b>	<b>Full strong branching <math>\alpha=0.99</math></b>	<b>Strong branching <math>\alpha=5/6</math></b>	<b>Strong branching <math>\alpha=3/4</math></b>	<b>Strong branching <math>\alpha=3/4</math> Candidates* 2</b>
Solved (of 50)	32	28	32	33	33
Average gap after branch and bound	0.5%	0.54 %	0.47%	0.47%	0.47%
Avg. time to opt (*)	476 s	668 s	383 s	370 s	392 s
BB Nodes (*)	191	109	168	160	140

(\*) only for instances that all configurations can solve.

# Branching on ...

Candidates:

15/10/10

	<b>Arcs</b>	<b>GUB</b>	<b>Edges</b>	<b>Branch on sets</b>	<b>Arcs reliability branching</b>
Solved (of 50)	32	20	28	28	29
Average gap after branch and bound	0.5%	0.78	0.54%	0.51 %	0.44 %
Avg. time to opt (*)	453	-	502	600	575
BB Nodes (*)	184	-	195	220	268

(\*) only for instances that all configurations can solve.

# Results on Solomon instances

	#instances	# Solved				Time (s)			
		R12	BMR	JPSP	DLH	R12	BMR	JPSP	DLH
C1	9	<b>9</b>	9	9	9	<b>15</b>	25	468	18
RC1	8	<b>8</b>	8	8	8	<b>2907</b>	276	11004	2150
R1	12	<b>12</b>	12	12	12	<b>2040</b>	251	27412	2327
C2	8	<b>8</b>	8	7	8	<b>209</b>	40	2795	2093
RC2	8	<b>8</b>	8	5	6	<b>2205</b>	3767	3204	15394
R2	11	<b>11</b>	10	4	8	<b>30592</b>	28680	35292	63068

BMR: Baldacci, Mingozzi, Roberti. Xeon X7350, 2.93GHz

JPSP: Jepsen, Petersen, Spoorendonk, Pisinger. P-IV, 3 GHz

DHL: Desaulniers, Lessard, Hadjar. Opteron 2.6 GHz

R12: Core i7-2620M 2.7GHz

# Solomon selected instances

	<b>BMR time (s)</b>	<b>R12 time (s)</b>
R204	216367	7159
R208	-	283835
R210	39171	9800

## R208 statistics

Root LB	691.7
Final bound (opt)	701
BB Nodes	2068

<b>Component</b>	<b>Share</b>
Book keeping	11.99%
LP	6.55%
Pricing heuristic	61.05%
Exact pricing	18.18%

# CVRP instances

Class #	R12		BMR		BCM		FLL				LLE	
	Opt	Time	Opt	Time	Opt	Time	Opt	OptBCP	OptBC	Time	Opt	Time
A	22	<b>22 44</b>	22	30	22	118	22	20	2	1961	15	6638
B	20	<b>20 181</b>	20	67	20	417	20	6	14	4763	19	8178
E-M	12	<b>9 1856</b>	9	303	8	1025	9	7	2	126987	3	39592
F	3	<b>3 2163</b>	2	164			3	0	3	2398	3	1046
P	24	<b>24 280</b>	24	85	22	187	24	16	8	2892	16	11219
Tot	81	<b>78</b>	77		72		78	49	29		56	

BMR: Baldacci, Mingozzi, Roberti. Xeon X7350, 2.93GHz

BCM: Baldacci, Christofides, Mingozzi, Pentium 4 2.6-GHz

FLL: Fukasawa et al., Pentium 4 2.4-GHz

LLE: Lysgaard, Letchford, Eglese, Intel Celeron 700-MHz

R12: Core i7-2620M 2.7GHz

# M-n151-k12 statistics

Root LB	1001.54
Current LB	1013.49
Best known upper bound	1015
CPU time so far	3-4 days



# Conclusion

- Strong branching pays off! Consider implementing it in your branch-and-price-(and-cut) algorithm!
- Branching on arcs, edges, sets or GUB does not seem to matter too much. However, arc branching seems best.
- Very good results on the VRPTW. It seems to be worthwhile to go for a simpler pricing problem (no master variable cuts).
- Now it's time to solve the 200 customer VRPTW instances ... 😊

Thank you for your attention!