Introduction to Column Generation

CPAIOR Master Class – 2016 - Banff

Louis-Martin Rousseau
Canada Research Chair in
Healthcare Analytics and Logistics

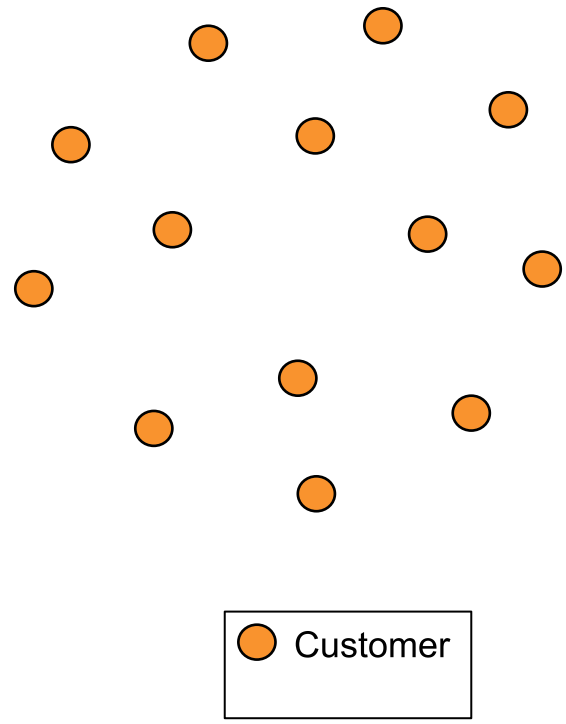
CIRRELT – POLYTECHNIQUE MTL

An example

Vehicle routing problem

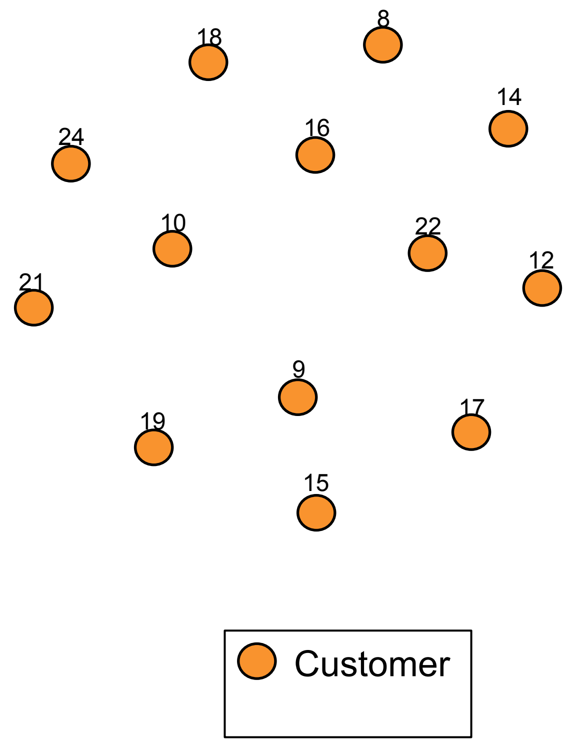
Vehicle routing problem

- Customers



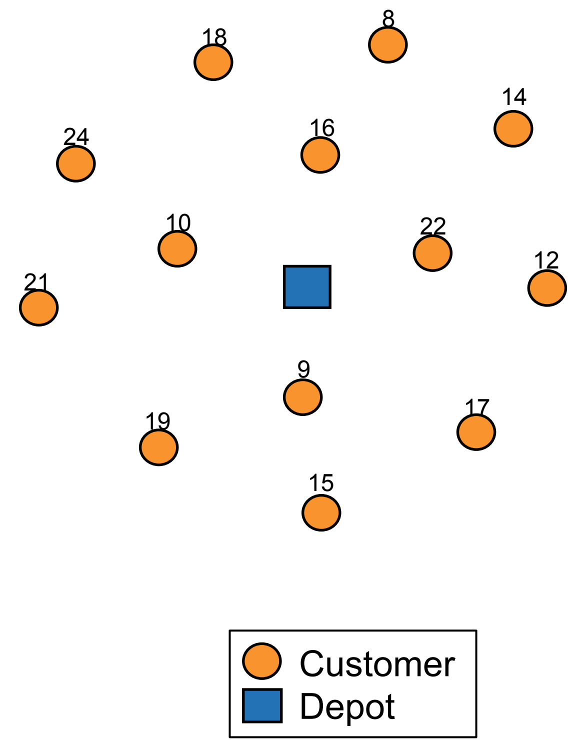
Vehicle routing problem

- Customers
  - Demand constraints



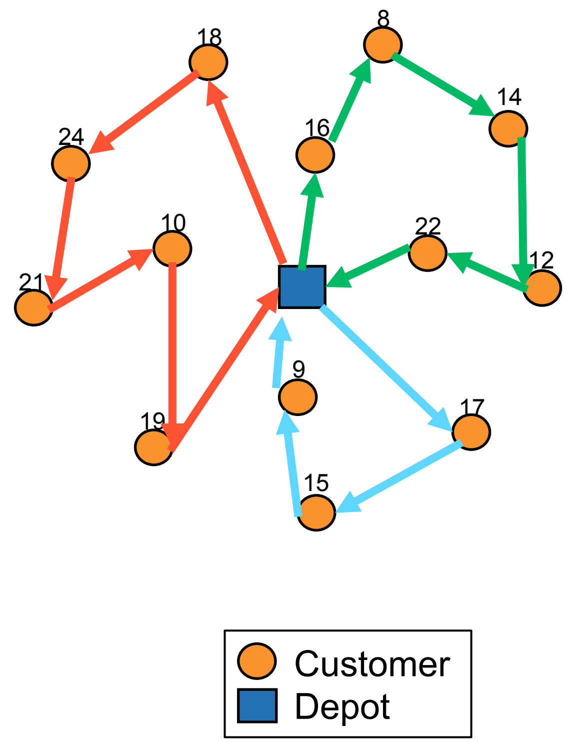
Vehicle routing problem

- Customers
  - Demand constraints
- Vehicles
  - Capacity constraints
  - Flow conservation constraints



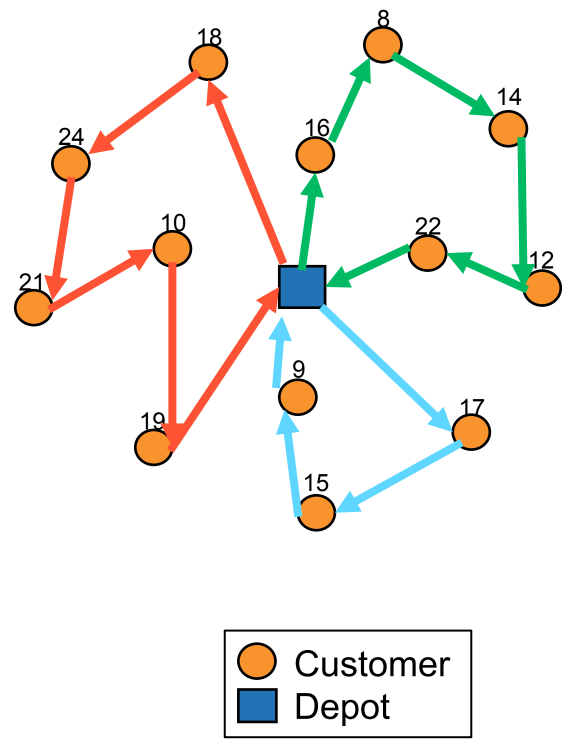
Vehicle routing problem

- Customers
  - Demand constraints
- Vehicles
  - Capacity constraints
  - Flow conservation constraints
- Objective:
  - Find routes that minimize total distance



Vehicle routing problem

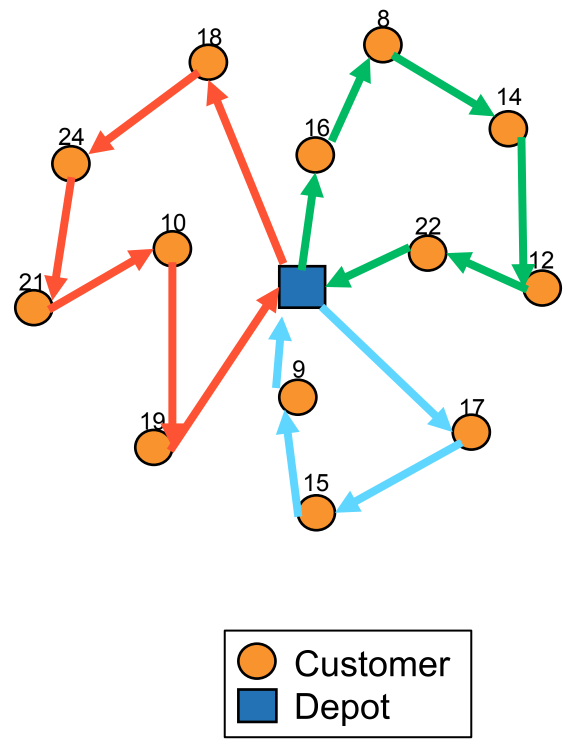
- Standard mip formulation:
  - Scaling issues
  - Symmetry
  - More complex constraints add even more complexity
  - Some constraints can lead to bad linear relaxations.



Vehicle routing problem

- Standard mip formulation:
  - Scaling issues
  - Symmetry
  - More complex constraints add even more complexity
  - Some constraints can lead to bad linear relaxations.

- Enumerate all possible routes
  - Much simpler formulation
  - Vehicle constraints are implicitly considered in route enumeration
  - Better Linear Relaxation



Vehicle routing problem

- Enumerate all possible routes

Minimize $\sum_{p \in \Omega} c_p \theta_p$

subject to: $\sum_{p \in \Omega} v_{ip} \theta_p = 1 \quad \forall i \in N$

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

Vehicle routing problem

CIRRELT

- Enumerate all possible routes

Minimize $\sum_{p\in\Omega} c_p \theta_p$

subject to: $\sum_{p\in\Omega} v_{ip} \theta_p = 1 \quad \forall i \in N$

Set of customers

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

Set of routes

Vehicle routing problem

- Enumerate all possible routes

Minimize $\sum_{p\in\Omega} c_p \theta_p$

subject to: $\sum_{p\in\Omega} v_{ip} \theta_p = 1 \quad \forall i \in N$

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

$= \begin{cases} 0,1, & \textit{if route p is used} \\ & \textit{otherwise} \end{cases}$

Vehicle routing problem

- Enumerate all possible routes

Cost of route p

Minimize $\sum_{p\in\Omega} c_p \theta_p$

$v_{ip} = \begin{cases} 1, & \text{if route } p \text{ visits customer } i \\ 0, & \text{otherwise} \end{cases}$

subject to: $\sum_{p\in\Omega} v_{ip} \theta_p = 1 \quad \forall i \in N$

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

$\theta_p = \begin{cases} 1, & \text{if route } p \text{ is used} \\ 0, & \text{otherwise} \end{cases}$

Vehicle routing problem

- Enumerate all possible routes

Cost of route p

Minimize $\sum_{p \in \Omega} c_p \theta_p$

$v_{ip} = \begin{cases} 1, & \text{if route } p \text{ visits customer } i \\ 0, & \text{otherwise} \end{cases}$

subject to: $\sum_{p \in \Omega} v_{ip} \theta_p = 1 \qquad \forall i \in N$

$\theta_p \in \{0,1\} \qquad \forall p \in \Omega$

Possibly huge number of routes

$\theta_p = \begin{cases} 1, & \text{if route } p \text{ is used} \\ 0, & \text{otherwise} \end{cases}$

Vehicle routing problem

CIRRELT

- Enumerate all possible routes

Cost of route p

Minimize $\sum_{p\in\Omega} c_p \theta_p$

$v_{ip} = \{ 1, 0,$ *if route p visits customer i*

subject to: $\sum_{p\in\Omega} v_{ip} \theta_p = 1 \qquad \forall i \in N$

$\theta_p \in \{0,1\} \qquad \forall p \in \Omega$

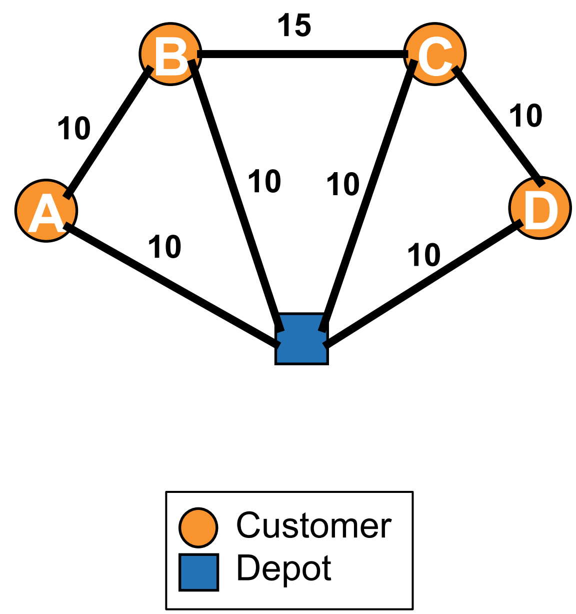
$\theta_p = \{ 1, 0,$ *if route p is used@otherwise*

Possibly huge number of routes

A very small number of routes are interesting

Vehicle routing problem

An example (max 2 clients)



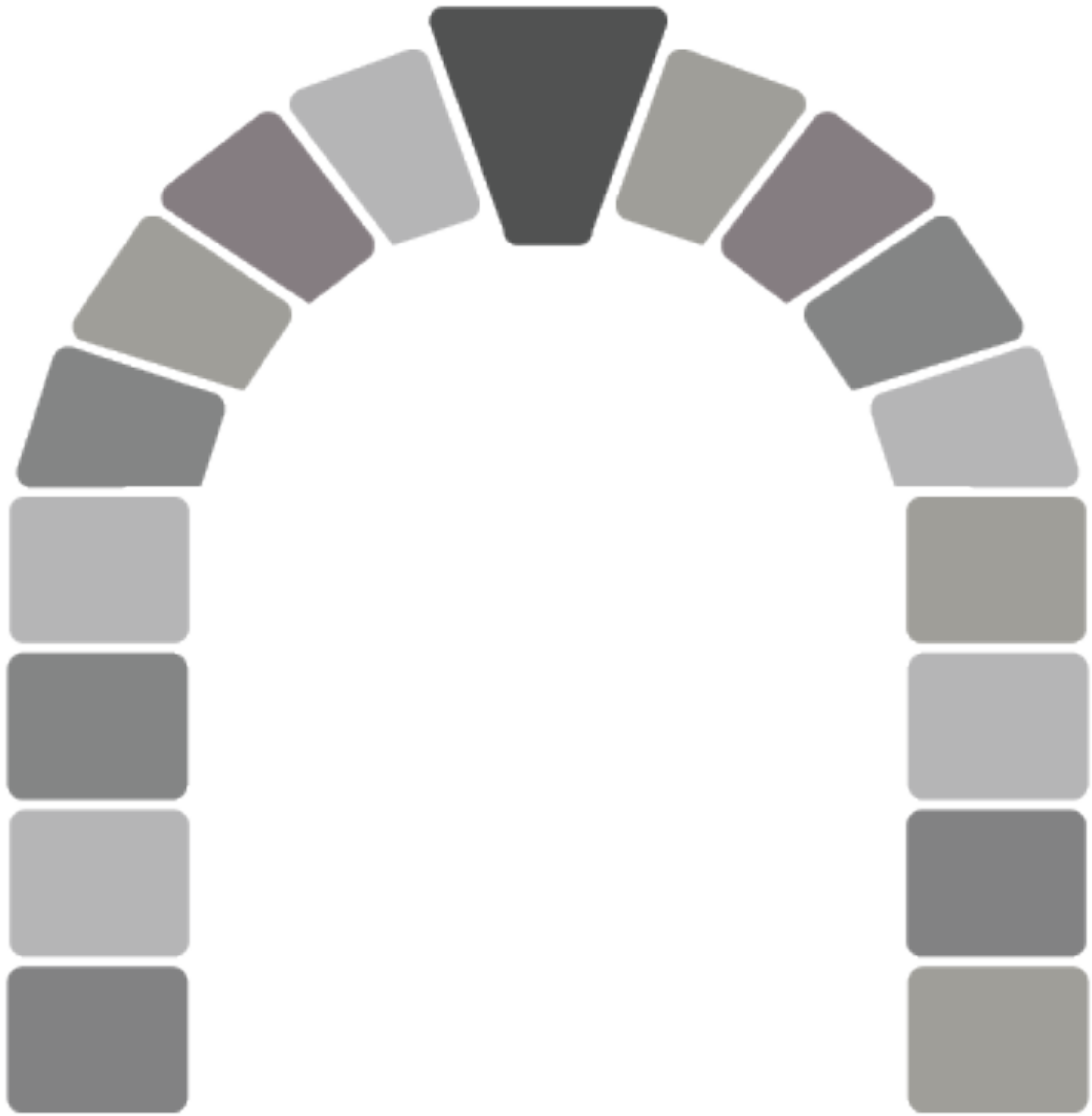
$$\text{Min} \quad 20\,x_1 + 20\,x_2 + 20\,x_3 + 20\,x_4 + 30\,x_5 + 30\,x_6 + 35\,x_7$$

$$\text{A}: \quad x_1 + x_5 = 1$$

$$\text{B}: \quad + x_2 + x_5 + x_7 = 1$$

$$\text{C}: \quad + x_3 + x_6 + x_7 = 1$$

$$\text{D}: \quad + x_4 + x_6 = 1$$

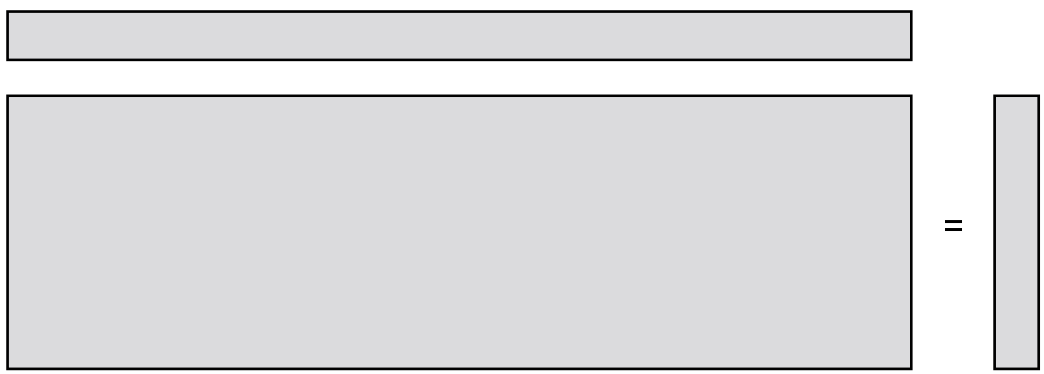


An intuitive view of

Column Generation

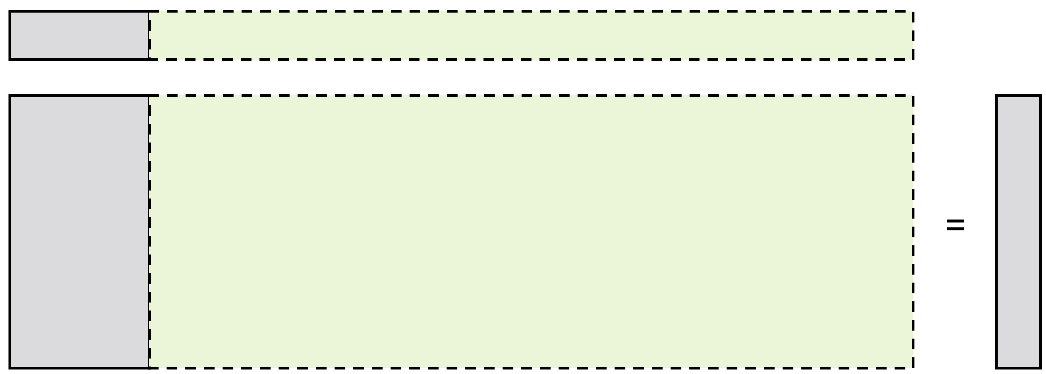
Column Generation

- Solve linear programs with a lot of variables



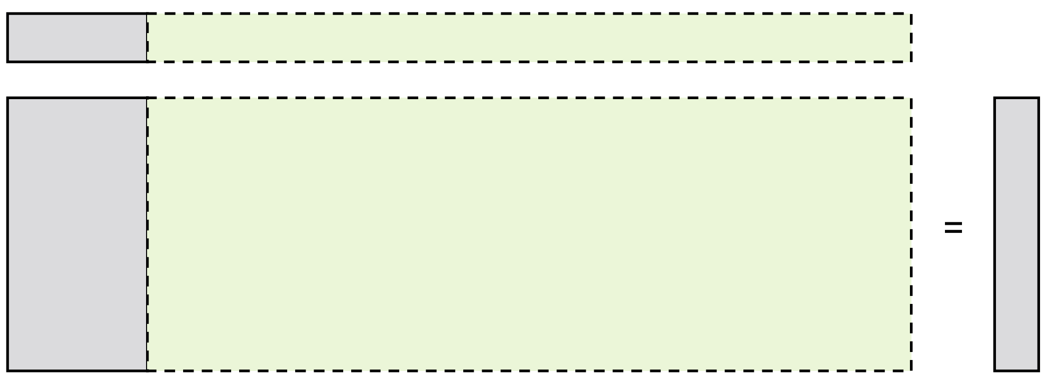
Column Generation

- Solve linear programs with a lot of variables
  - Solve with a subset of variables



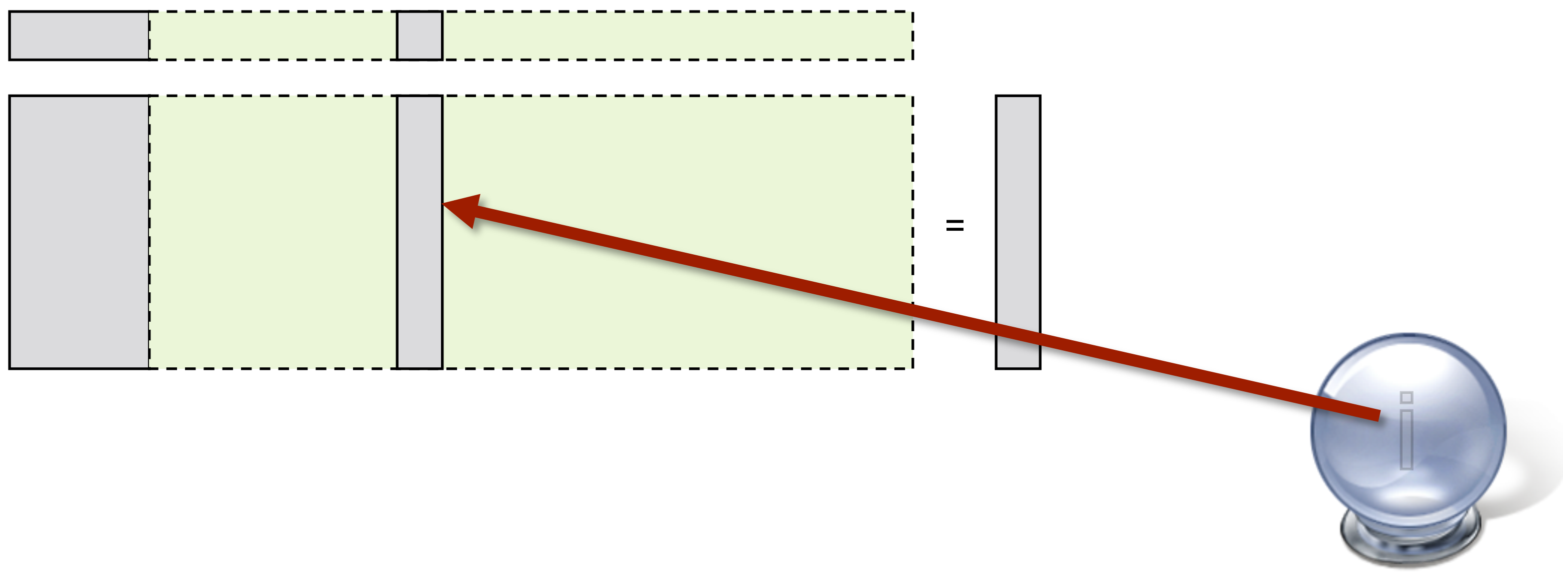
Column Generation

- Solve linear programs with a lot of variables
  - Solve with a subset of variables



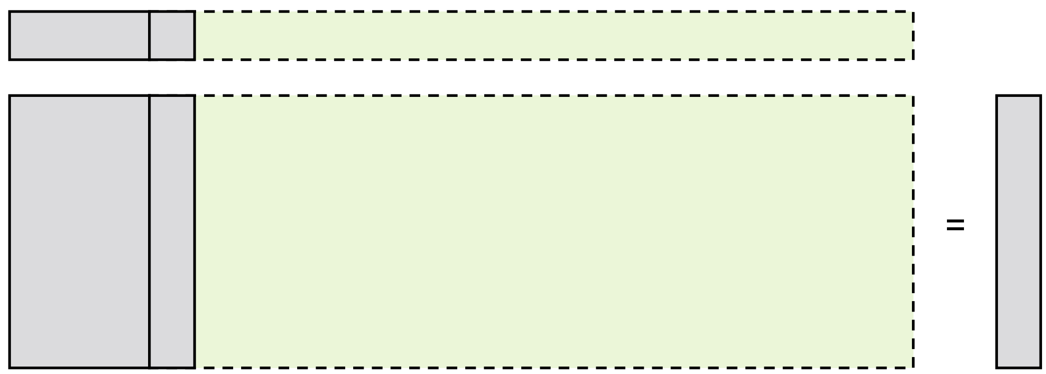
Column Generation

- Solve linear programs with a lot of variables
  - Solve with a subset of variables



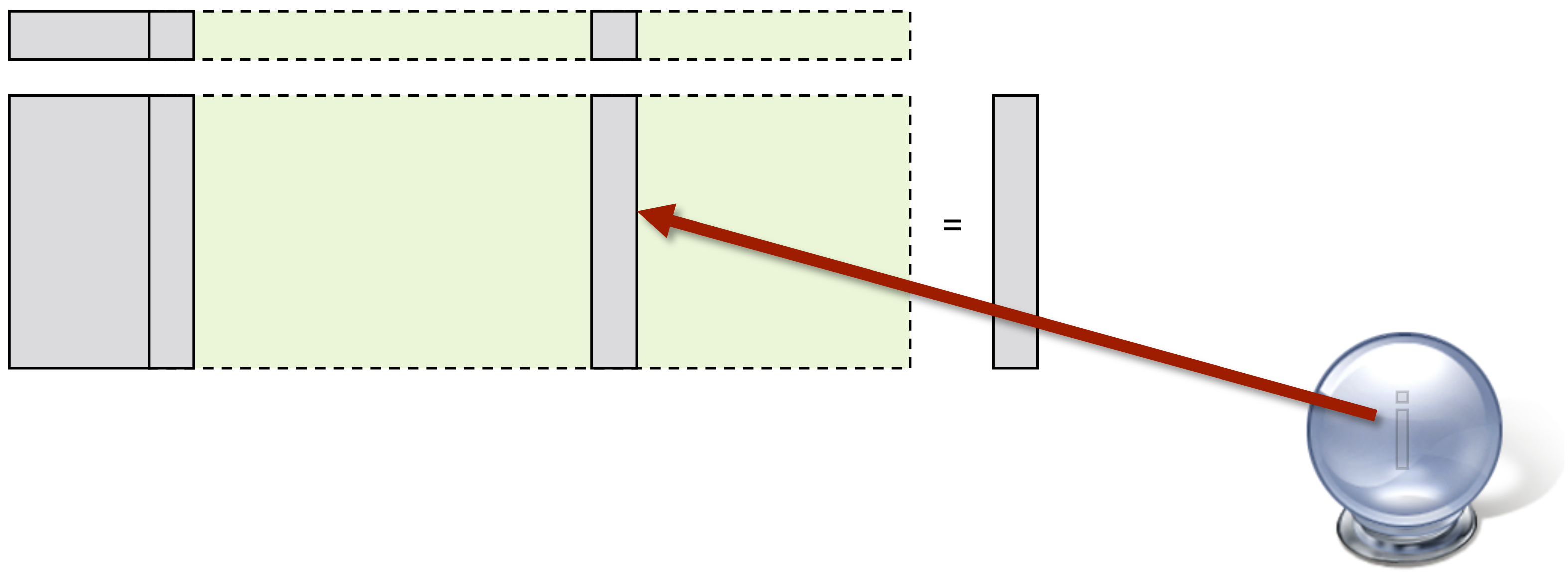
Column Generation

- Solve linear programs with a lot of variables
  - Solve with a subset of variables



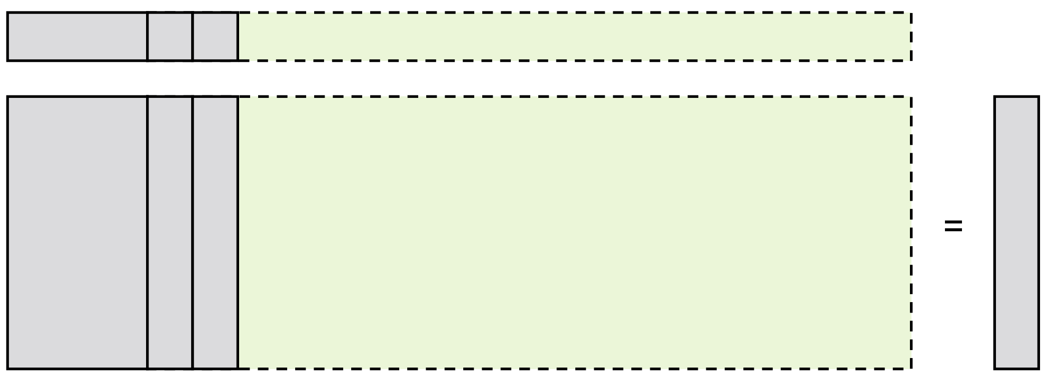
Column Generation

- Solve linear programs with a lot of variables
  - Solve with a subset of variables



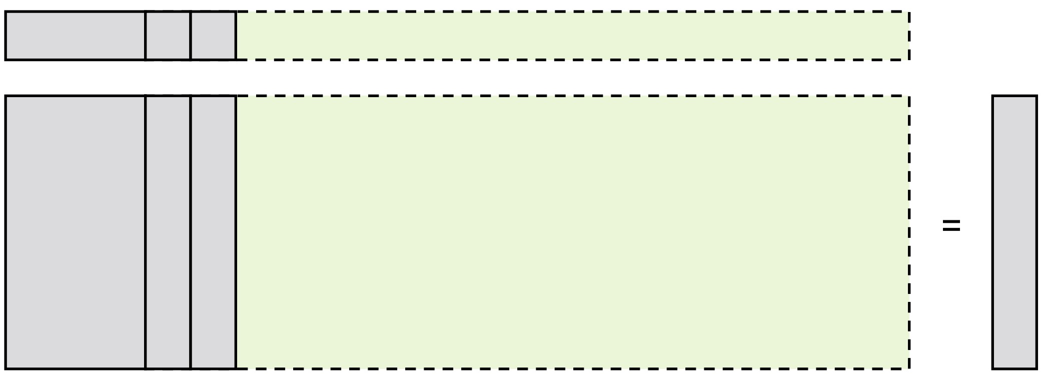
Column Generation

- Solve linear programs with a lot of variables
  - Solve with a subset of variables



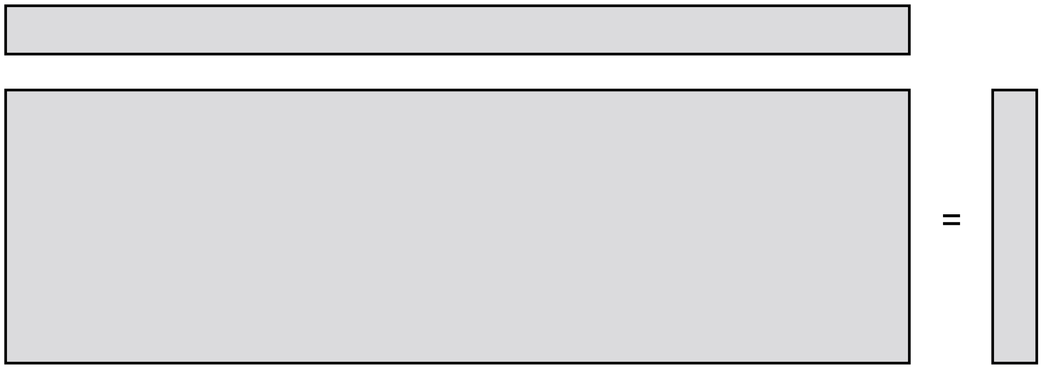
Column Generation

- Solve linear programs with a lot of variables
  - Solve with a subset of variables



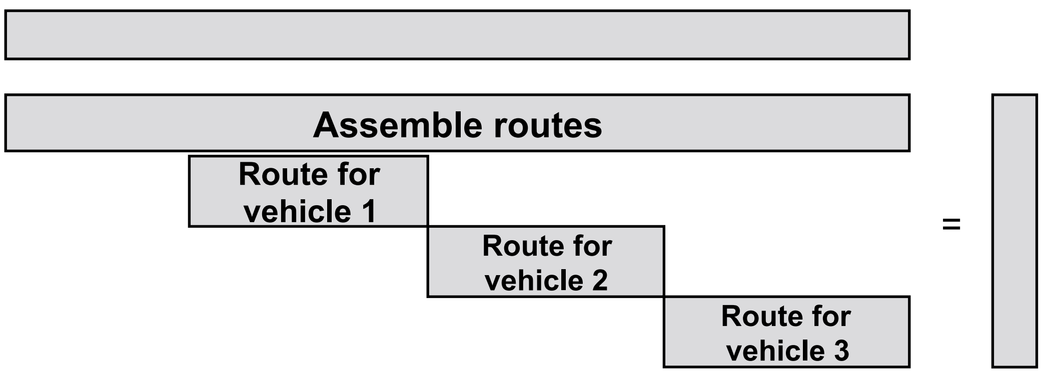
Column Generation

- When to use column generation?



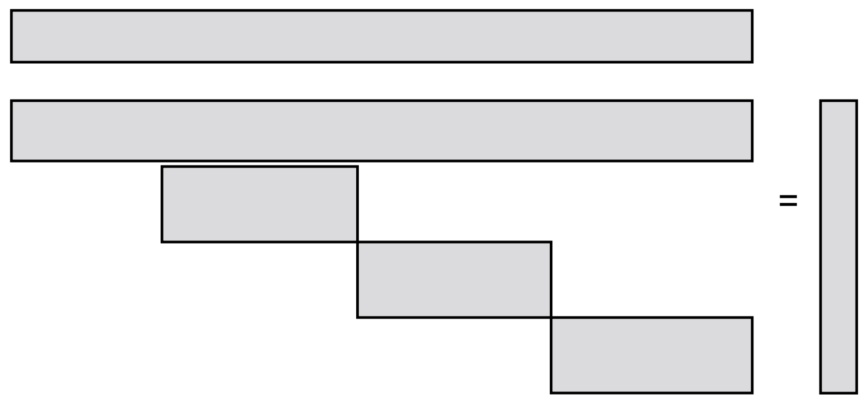
Column Generation

- When to use column generation?



Column Generation

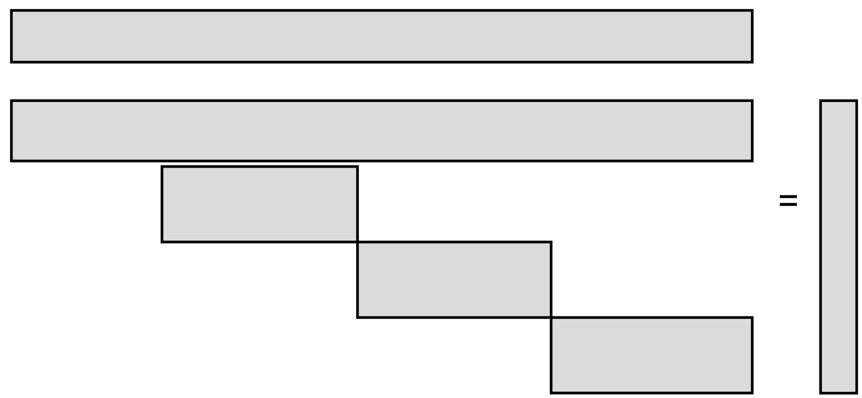
- When to use column generation?
- Works well generally on:
  - Vehicle routing
  - Airline Scheduling
  - Shift Scheduling
  - Jobshop Scheduling
  - …



- Multi-level of assembly

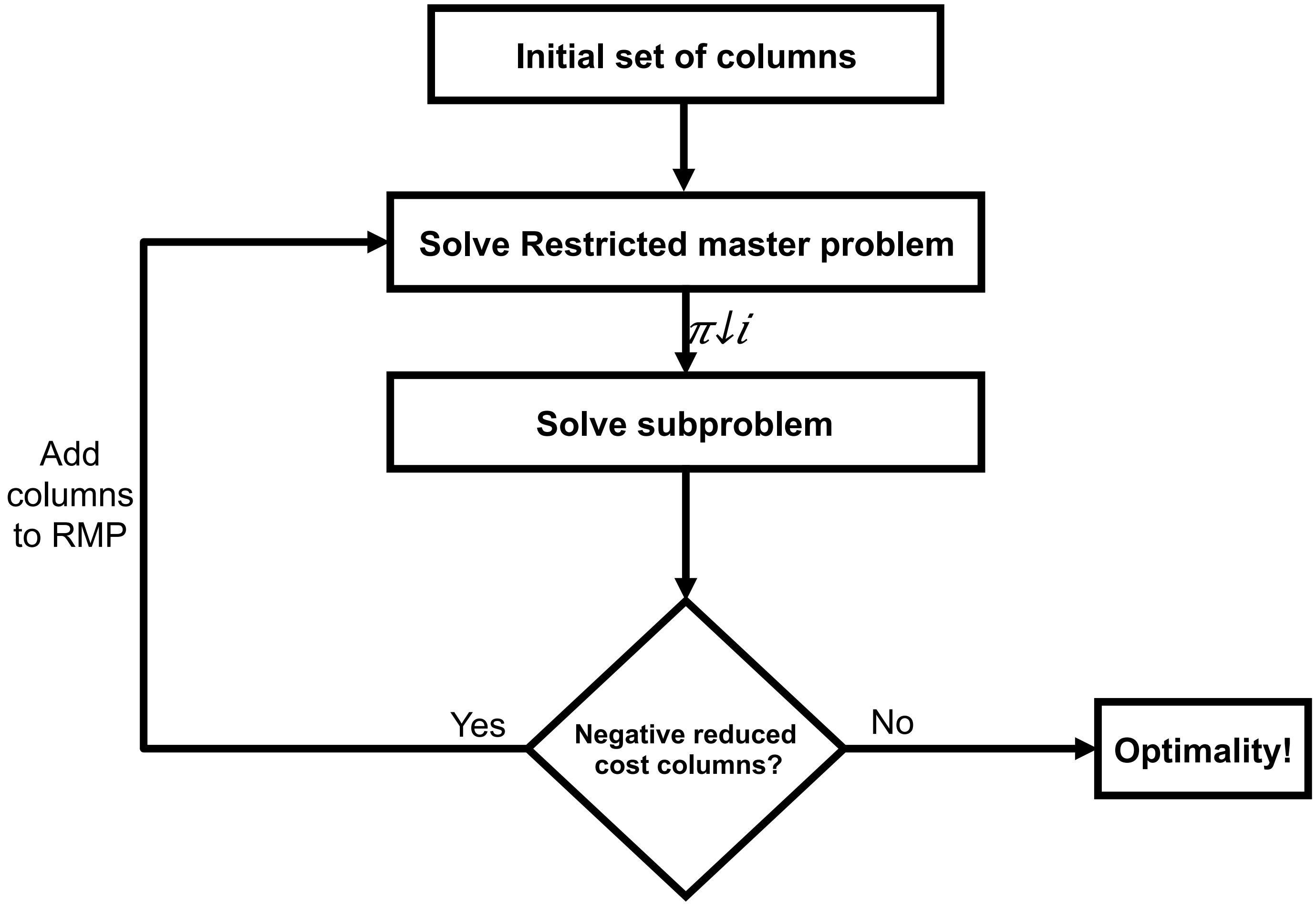
Column Generation

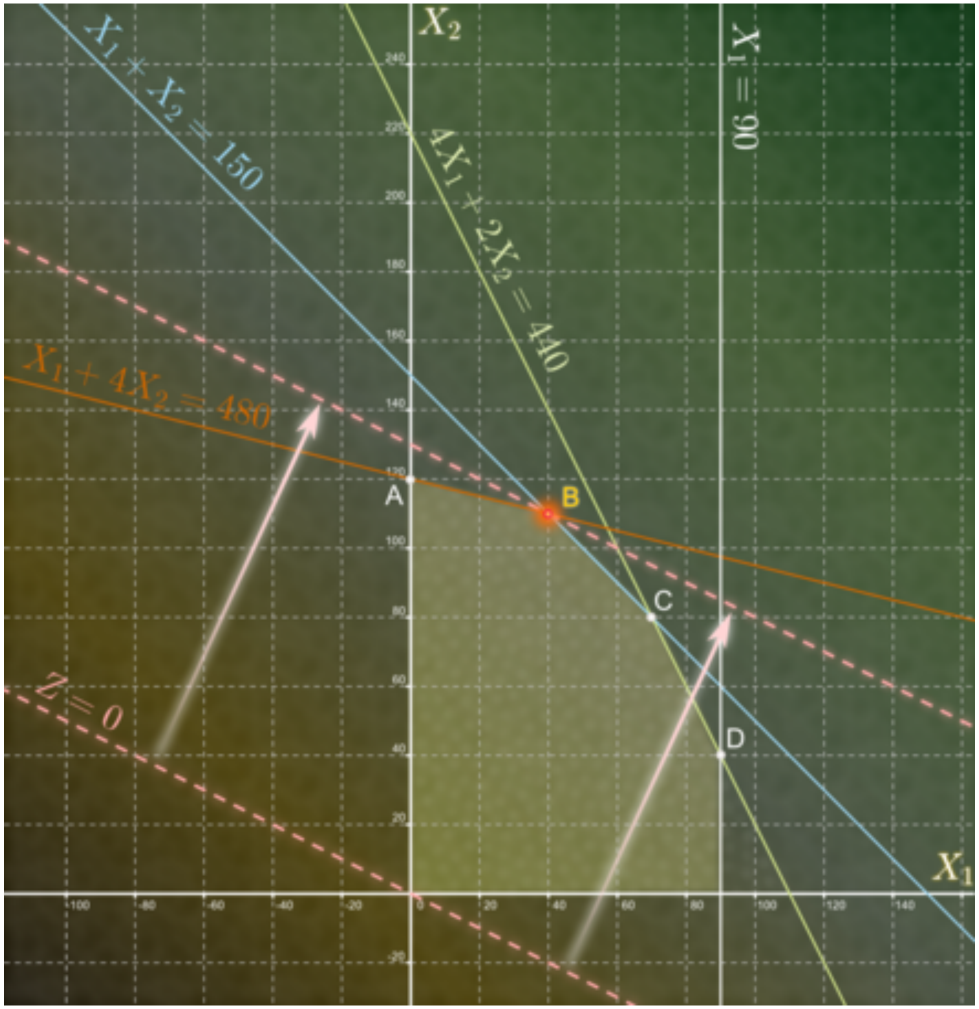
- When to use column generation?
- Works well generally on:
  - Vehicle routing
  - Airline Scheduling
  - Shift Scheduling
  - Jobshop Scheduling
  - …



- Multi-level of assembly
- Worked the best when part of the problem has an underlying structure: Network, Hypergraph, knapsack, etc…

Column Generation



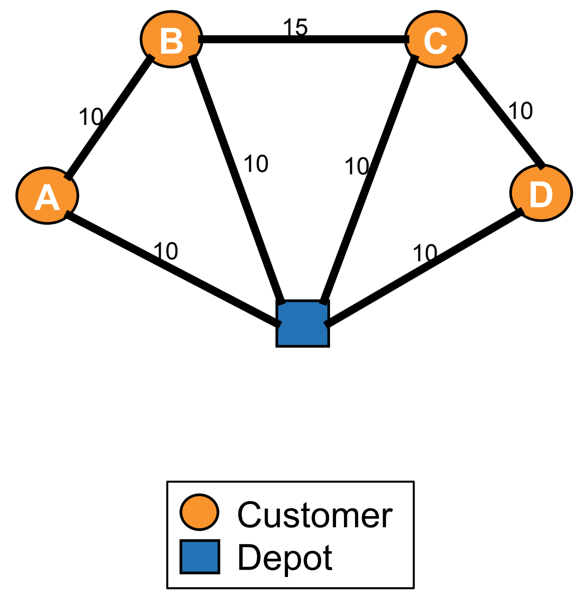


Master Probelm for the

Vehicle routing problem

Vehicle routing problem

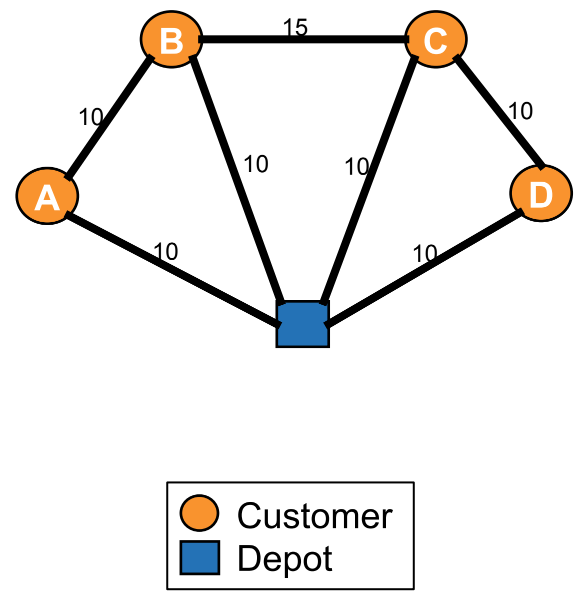
- An example (max 2 clients)



Vehicle routing problem

- An example (max 2 clients)

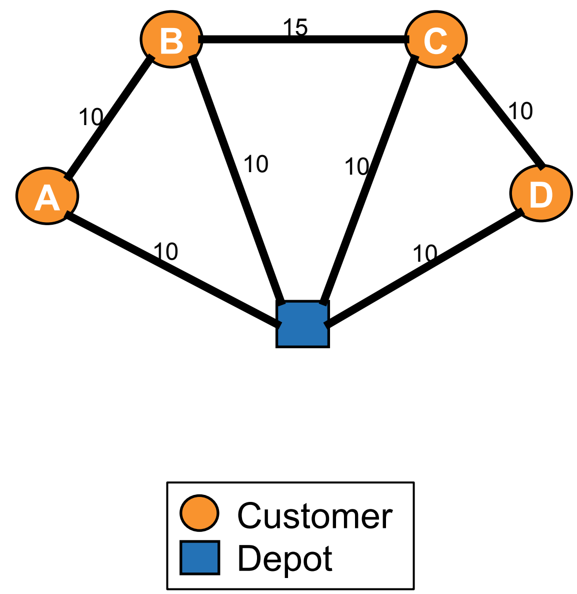
$$
\begin{array}{lccccl}
\text{Min} & 20\,x_1 & +20\,x_2 & +20\,x_3 & +20\,x_4 & \\
\text{A :} & x_1 & & & & = 1 \\
\text{B :} & & x_2 & & & = 1 \\
\text{C :} & & & x_3 & & = 1 \\
\text{D :} & & & & x_4 & = 1
\end{array}
$$



Vehicle routing problem

- An example (max 2 clients)

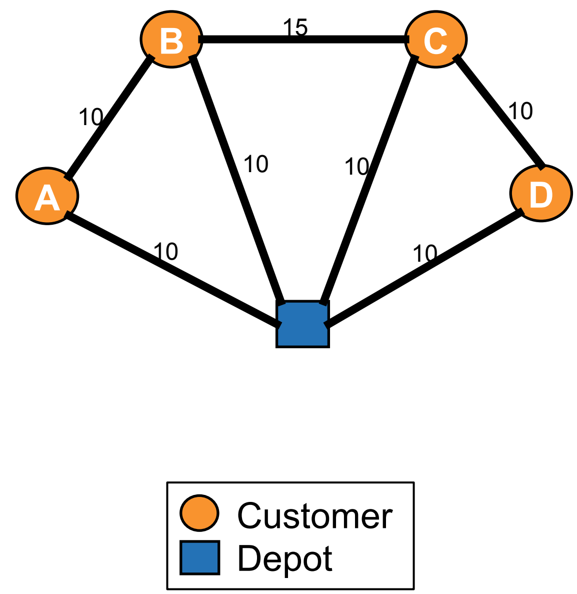
	x_1	x_2	x_3	x_4	
Min	20	20	20	20	
A :	1				= 1
B :		1			= 1
C :			1		= 1
D :				1	= 1



Vehicle routing problem

- An example (max 2 clients)

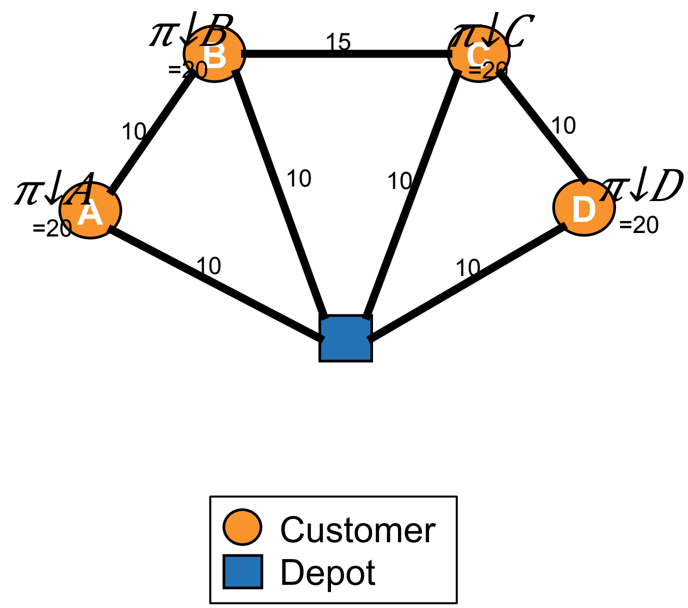
	x_1	x_2	x_3	x_4		
ĉ	0	0	0	0		π_i
A :	1				= 1	20
B :		1			= 1	20
C :			1		= 1	20
D :				1	= 1	20
	1	1	1	1	80	



Vehicle routing problem

- An example (max 2 clients)

	x_1	x_2	x_3	x_4		π_i
ĉ	0	0	0	0		
A :	1				= 1	20
B :		1			= 1	20
C :			1		= 1	20
D :				1	= 1	20
	1	1	1	1	80	

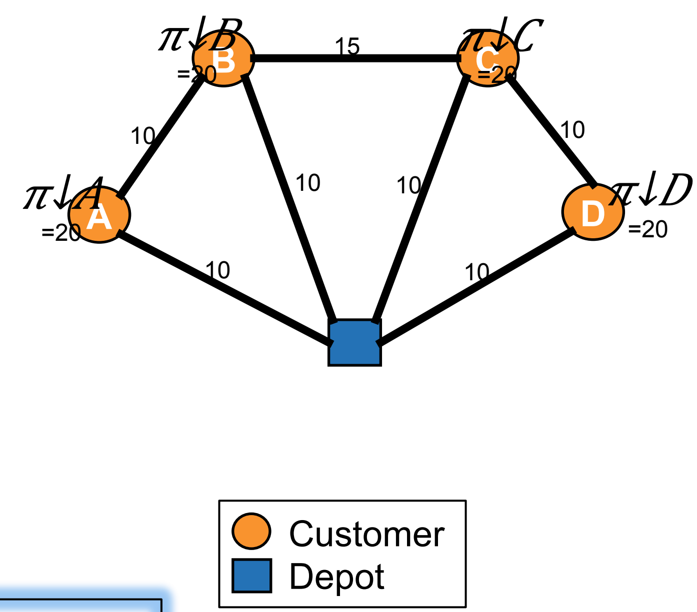


π_i : Marginal price of visiting customer *I*

Vehicle routing problem

- An example (max 2 clients)

	x_1	x_2	x_3	x_4		
ĉ	0	0	0	0		π_i
A :	1				= 1	20
B :		1			= 1	20
C :			1		= 1	20
D :				1	= 1	20
	1	1	1	1	80	



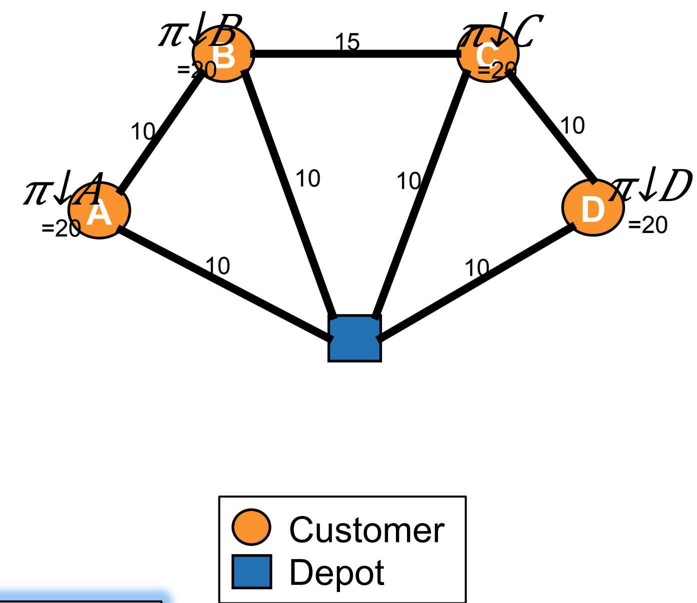
π_i : Marginal price of visiting customer *i*

Can I find a route such that:
$c < \sum \pi_i$

Vehicle routing problem

- An example (max 2 clients)

	x_1	x_2	x_3	x_4		
ĉ	0	0	0	0		π_i
A :	1				= 1	20
B :		1			= 1	20
C :			1		= 1	20
D :				1	= 1	20
	1	1	1	1	80	



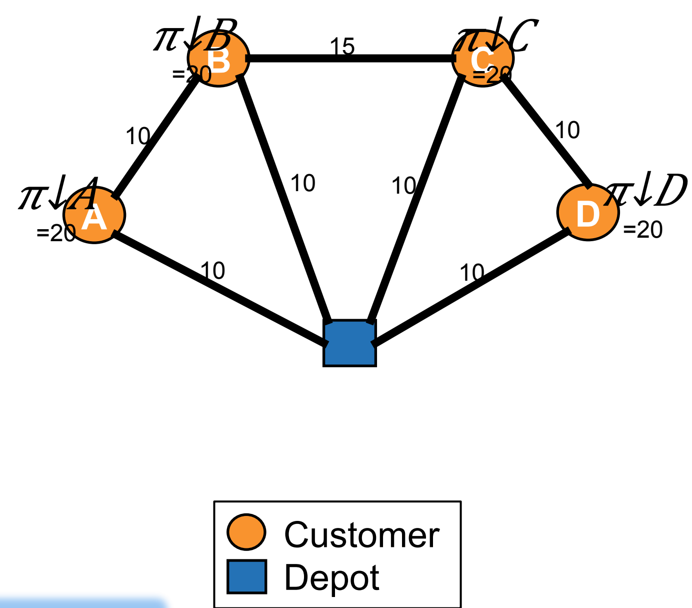
π_i : Marginal price of visiting customer *i*

Can I find a route such that:
$c - \sum \pi_i < 0$

Vehicle routing problem

- An example (max 2 clients)

	x_1	x_2	x_3	x_4		π_i
ĉ	0	0	0	0		
A :	1				= 1	20
B :		1			= 1	20
C :			1		= 1	20
D :				1	= 1	20
	1	1	1	1	80	



π_i : Marginal price of visiting customer *I*

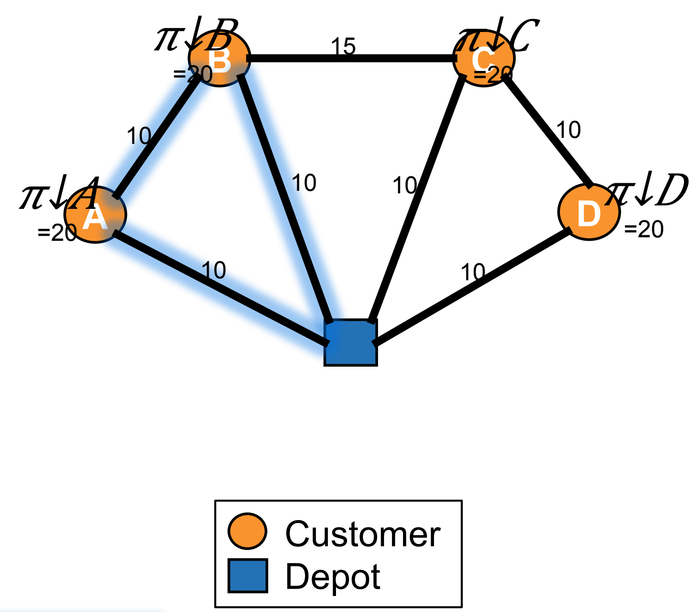
Can I find a route such that:
$c - \sum \pi_i < 0$

Reduced cost!

Vehicle routing problem

- An example (max 2 clients)

	x_1	x_2	x_3	x_4		
ĉ	0	0	0	0		π_i
A :	1				= 1	20
B :		1			= 1	20
C :			1		= 1	20
D :				1	= 1	20
	1	1	1	1	80	



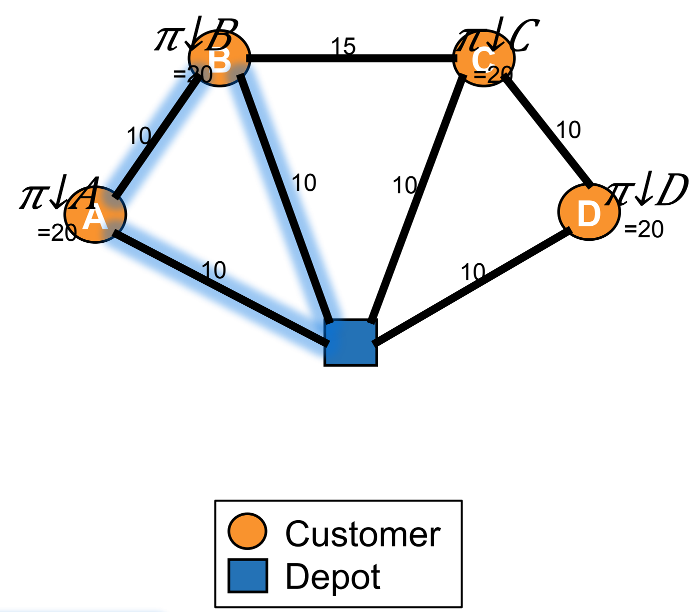
π_i : Marginal price of visiting customer *I*

Can I find a route such that:
$c - \sum \pi_i < 0$

Vehicle routing problem

- An example (max 2 clients)

	x_1	x_2	x_3	x_4	x_5		
ĉ	0	0	0	0	-10		π_i
A :	1				1	= 1	20
B :		1			1	= 1	20
C :			1			= 1	20
D :				1		= 1	20
	1	1	1	1		80	



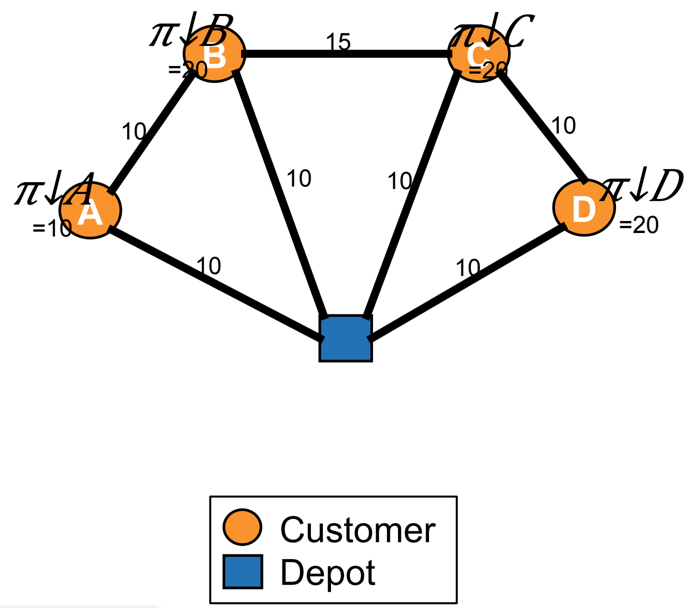
π_i : Marginal price of visiting customer *I*

Can I find a route such that:
$c - \sum \pi_i < 0$

Vehicle routing problem

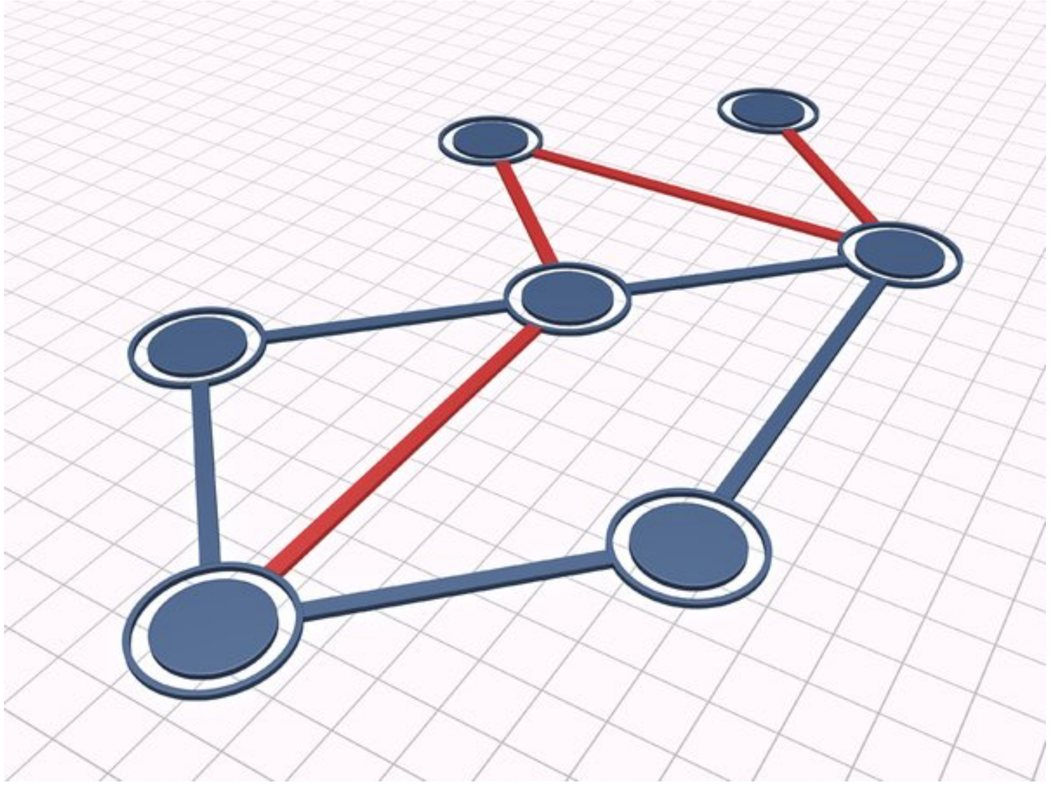
- An example (max 2 clients)

	x_1	x_2	x_3	x_4	x_5		π_i
ĉ	10	0	0	0	0		
A :	1				1	= 1	10
B :		1			1	= 1	20
C :			1			= 1	20
D :				1		= 1	20
		0	1	1	1	70	



π_i : Marginal price of visiting customer *i*

Can I find a route such that:
$c - \sum \pi_i < 0$



Sub Probelm for the

Vehicle routing problem

General Subproblem

- Implicit representation of all variables
  - Every possible solution to the subproblem is a variable

- Optimization objective:

→ **find variable with (the most) negative reduced c**

Min $\hat{c} = c - \sum_i a_i \pi_i \quad a_i = \begin{cases} 1, & \text{if customer } i \text{ is visited} \\ 0, & \text{otherw} \end{cases}$

General Subproblem

- Implicit representation of all variables
  - Every possible solution to the subproblem is a variable

- Optimization objective:

→ **find variable with (the most) negative reduced cost**

$$\text{Min } \hat{c} = c - \sum_i a_i \pi_i \qquad a_i = \begin{cases} 1, & \textit{if customer } i \textit{ is visited} \\ 0, & \textit{otherwise} \end{cases}$$

$$c = \sum_x c_x x$$

Subproblem

- Implicit representation of all variables
  - Every possible solution to the subproblem is a variable

- Optimization objective:

find variable with (the most) negative reduced cost

Min $\hat{c} = \sum_{x} c_x x - \sum_i a_i \pi_i$, $a_i = \begin{cases} 1, & \text{if customer } i \text{ is visited} \\ 0, & \text{otherw…} \end{cases}$

Subproblem

- Implicit representation of all variables
  - Every possible solution to the subproblem is a variable
- Optimization objective:

→ **find variable with (the most) negative reduced cost**

Min $\hat{c} = \sum_{x} c_x x - \sum_{i} \pi_i a_i$ $\quad a_i = \begin{cases} 1, & \text{if customer } i \text{ is visited} \\ 0, & \text{otherwise} \end{cases}$

Subject to: Capacity constraints

Flow conservation constraints

Shortest-path problem with resource constraints: Dynamic programming

Resources Constraint SPP

- Resource r = 1,…,R
- Resource consumption $t^r_{ij} > 0$ on each arc.
- Resources window$[a^r_i,b^r_i]$ at each node
  - Resources level cannot go above b^r_i when node v_i is reached
  - If t^r_{ij} is below a^r_i when node path reaches v_i then is it set to a^r_i

Resources Constraint SPP - DP

- Dynamic Programming Algorithm

  - L_i : list of labels associated with node v_i

  - label $l = (c, T^1, \ldots, T^R)$ where
    - c is the cost of the label
    - T^r is the consumption level of resource r
    - a label represents a partial path from v_0 to v_i
    - $v(l)$ is the node which to which l is associated

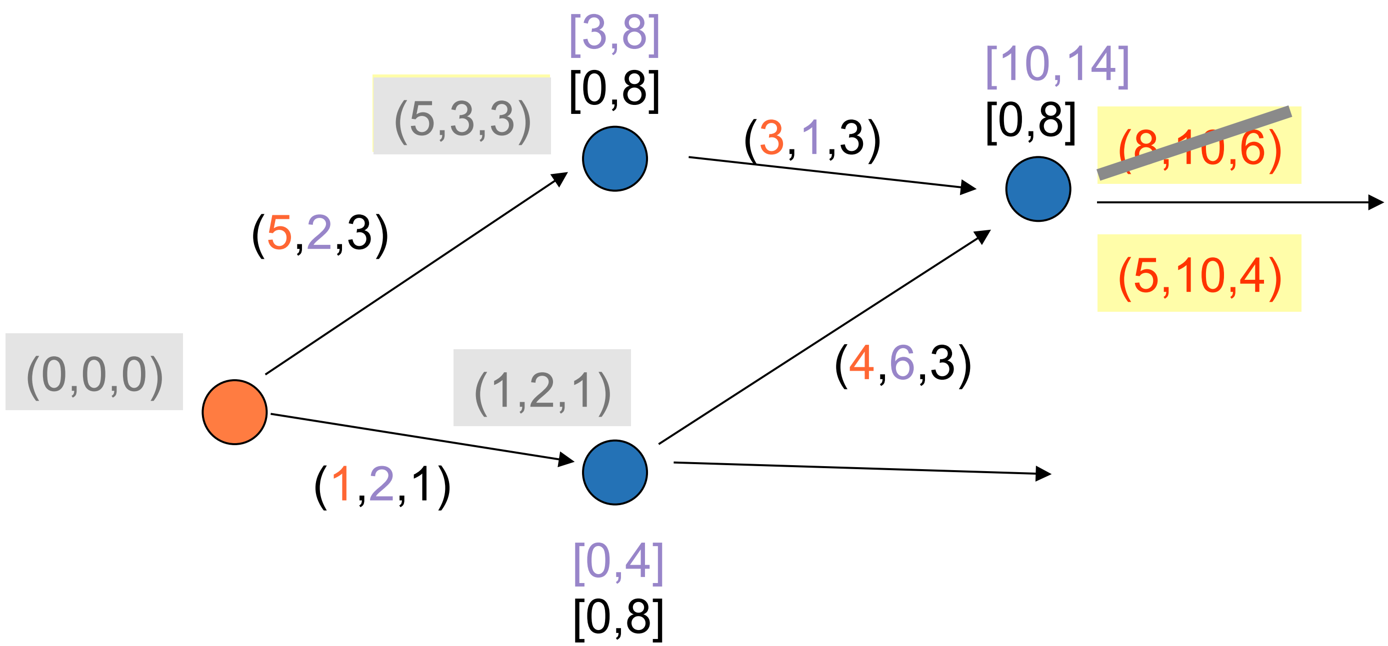
Resources Constraint SPP - DP

- Extending a label $l = (c, T^1_i, \ldots, T^R_i)$ from v_i to v_j
  - Create a label $(c + c_{ij}, T^1 + t^1_{ij}, \ldots, T^R + t^R_{ij})$
    - Making sure we respect $[a^1_j, b^1_j], \ldots, [a^R_j, b^R_j]$
  - Insert the label in the list of labels associated with v_j
  - Apply **Dominance Rules**
    - Without such rules, the algorithm would enumerates all possible paths
  - Resources constraints make sure the algorithm terminates

Resources Constraint SPP - DP

- Dominance Rules: l_1 dominates l_2 iff :
  - $c(l_1) <= c(l_2)$
  - Every feasible **future** extension of l_2 will be feasible for l_1
    - *Most often* we check that $T^r(l_1) <= T^r(l_2)$ for all r

Dominance: an example



Subproblem – Constraint Programming

CIRRELT

- "Arc Flow" model

- Objectives:
  - Minimize: $\sum_i (\text{ReducedCost}(i, S_i))$

- Variables:

– $S_i \in N$	Successor of node i
– $V_i \in \{\text{False}, \text{True}\}$	Node i visited by current path
– $l_i \in [0..\text{Capacity}]$	Truck load after visit of node i

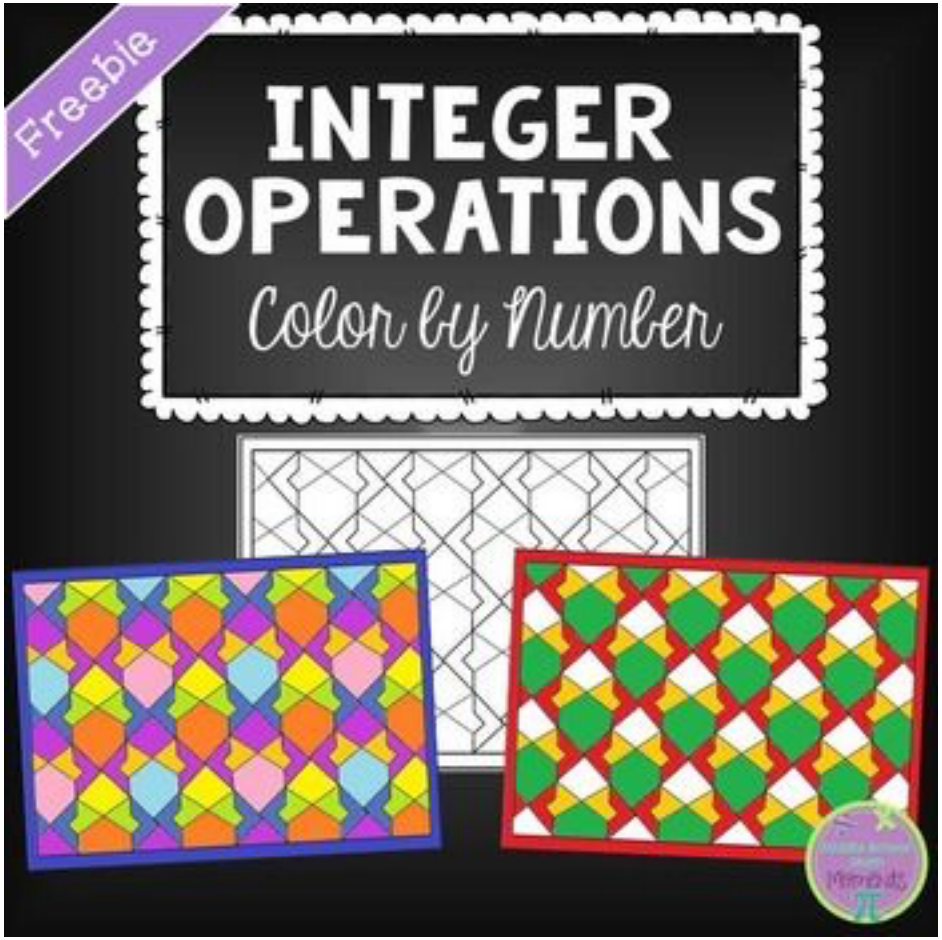
- Constraints:

– $S_i = i \rightarrow V_i = \text{False}$	S-V Coherence constraints
– AllDiff(S)	Conservation of flow
– NoSubTour(S)	SubTour elimination constraint
– $S_i = j \rightarrow l_i + D_j = l_j$	Capacity constraints

- + Redundant Constraints from work on TSP(TW)

Subproblem – Constraint Programming

- ”Position” model
- Objectives:
  - Minimize: $\sum_k$ (ReducedCost(P_k, P_{k+1}))
- Variables:
  - $P_k \in N$ — Node visited a position k
  - $L_k \in [0..Capacity]$ — Truck load after visiting position k
- Constraints:
  - AllDiff(P) — Elementarity of the path
  - $L_{k+1} = L_k + D_{Pk}$ — Capacity constraints
  - P_k = depot → P_{k+1} = depot — Padding at the end of path



Branch-and-price

Obtaining integer solutions

Branch-and-price

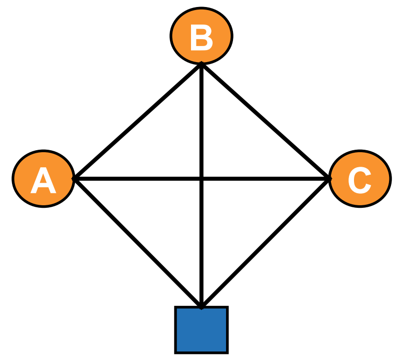
- Column generation + MIP : Branch-and-price
  - How to obtain integer solutions?

Branch-and-price

- Column generation + MIP : Branch-and-price
  - How to obtain integer solutions?
    - Branch-and-bound -> solve LP relaxation at each node
    - Branch-and-price -> column generation to solve LP relaxation at each node

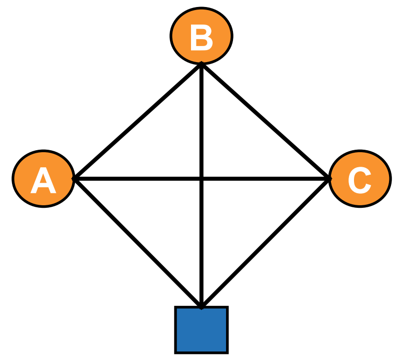
Branch-and-price

- Vehicle routing problem
  - Max 2 customers
  - Cost of all arc : 1



Branch-and-price

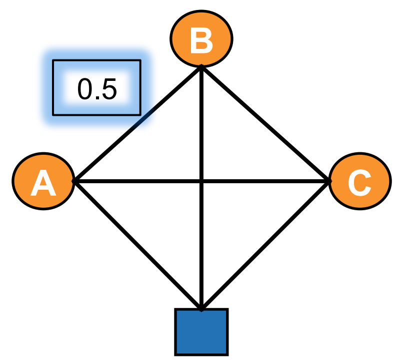
- Vehicle routing problem
  - Max 2 customers
  - Cost of all arc : 1



	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1
OptSol:	0.5	0.5	0.5	4.5

Branch-and-price

- Vehicle routing problem
  - Max 2 customers
  - Cost of all arc : 1

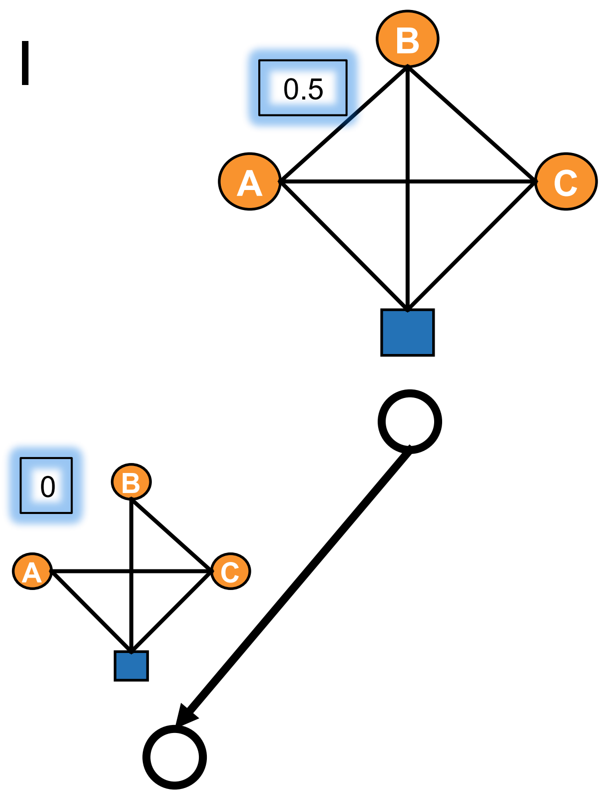


	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1
OptSol:	0.5	0.5	0.5	4.5

Branch-and-price

CIRRELT

- Vehicle routing problem
  - Max 2 customers
  - Cost of all arc : 1

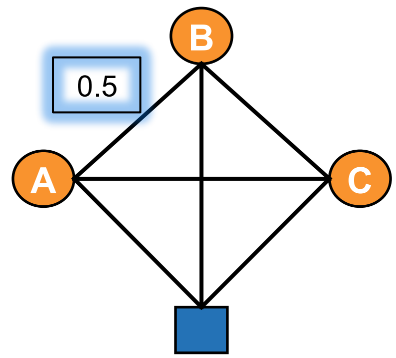


	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1
OptSol:	0.5	0.5	0.5	4.5

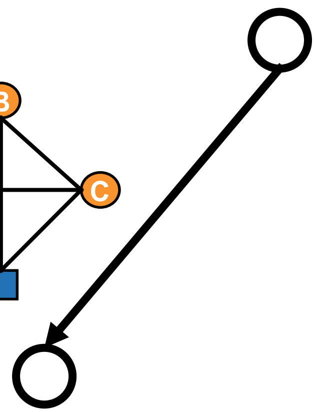
	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1

Branch-and-price

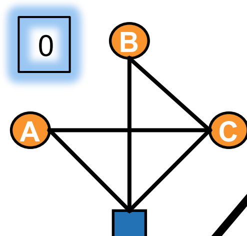
- Vehicle routing problem
  - Max 2 customers
  - Cost of all arc : 1



	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1
OptSol:	0.5	0.5	0.5	4.5



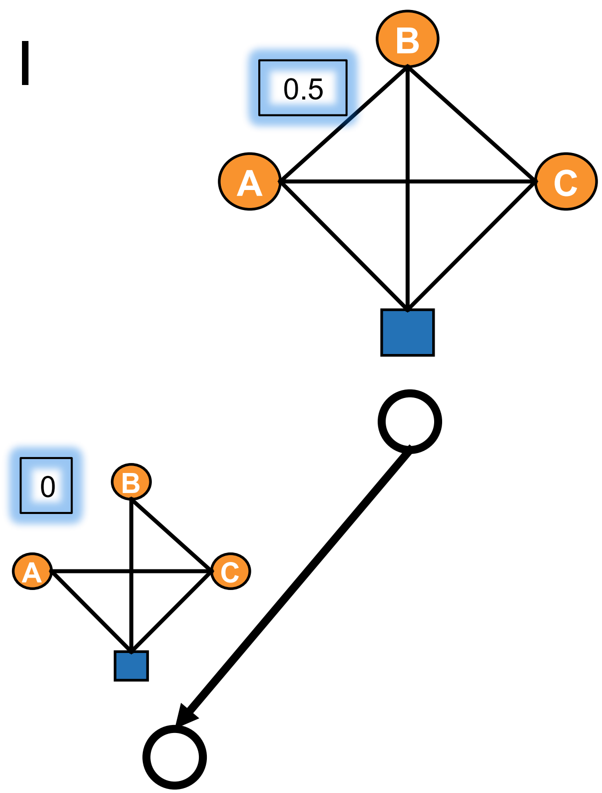
	x_4
	2
A :	1
B :	
C :	



	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1

Branch-and-price

- Vehicle routing problem
  - Max 2 customers
  - Cost of all arc : 1



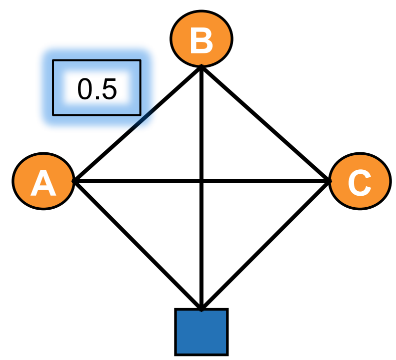
	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1
OptSol:	0.5	0.5	0.5	4.5

	x_4
	2
A :	1
B :	
C :	

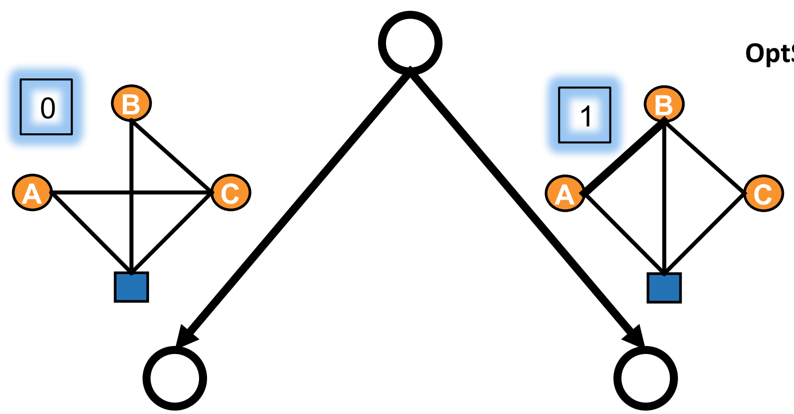
	x_1	x_2	x_3	x_4	
Min	3	3	3	2	
A :	1	1		1	= 1
B :	1		1		= 1
C :		1	1		= 1

Branch-and-price

- Vehicle routing problem
  - Max 2 customers
  - Cost of all arc : 1



	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1
OptSol:	0.5	0.5	0.5	4.5

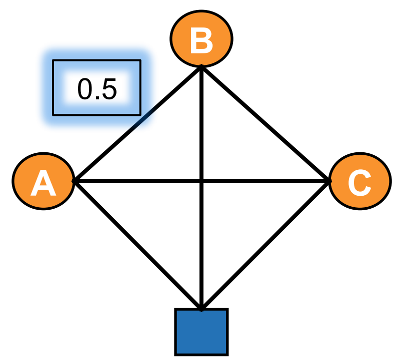


	x_1	x_2	x_3	x_4	
Min	3	3	3	2	
A :	1	1		1	= 1
B :	1		1		= 1
C :		1	1		= 1

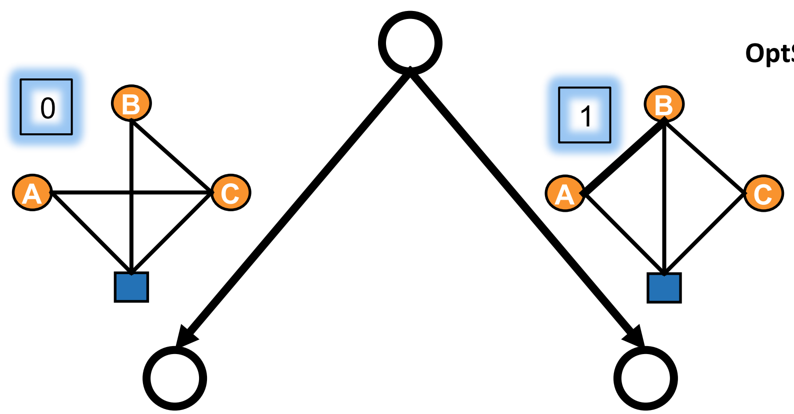
	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1

Branch-and-price

- Vehicle routing problem
  - Max 2 customers
  - Cost of all arc : 1



	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1
OptSol:	0.5	0.5	0.5	4.5



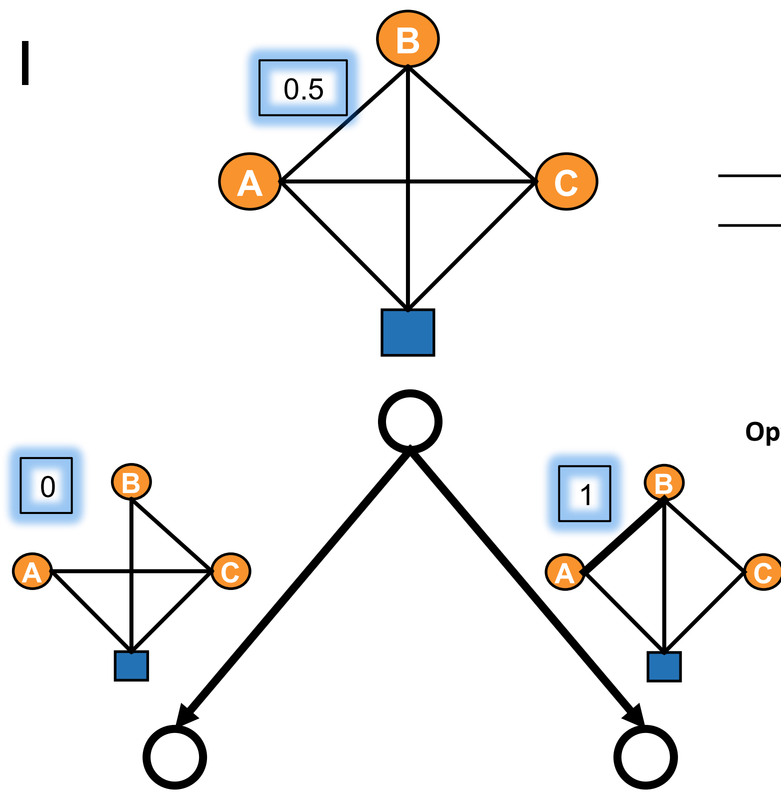
	x_1	x_2	x_3	x_4	
Min	3	3	3	2	
A :	1	1		1	= 1
B :	1		1		= 1
C :		1	1		= 1

	x_1	x_2	x_3	x_5	
Min	3	3	3	2	
A :	1	1			= 1
B :	1		1		= 1
C :		1	1	1	= 1

Branch-and-price

- Vehicle routing problem
  - Max 2 customers
  - Cost of all arc : 1

Why branch on arc-flow variables?



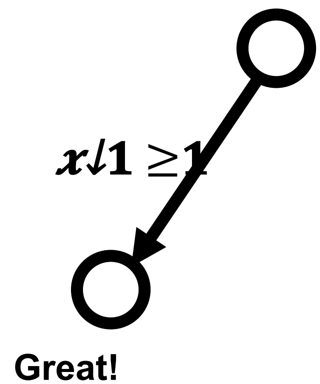
	x_1	x_2	x_3	
Min	3	3	3	
A :	1	1		= 1
B :	1		1	= 1
C :		1	1	= 1
OptSol:	0.5	0.5	0.5	4.5

	x_1	x_2	x_3	x_4	
Min	3	3	3	2	
A :	1	1		1	= 1
B :	1		1		= 1
C :		1	1		= 1

	x_1	x_2	x_3	x_5	
Min	3	3	3	2	
A :	1	1			= 1
B :	1		1		= 1
C :		1	1	1	= 1

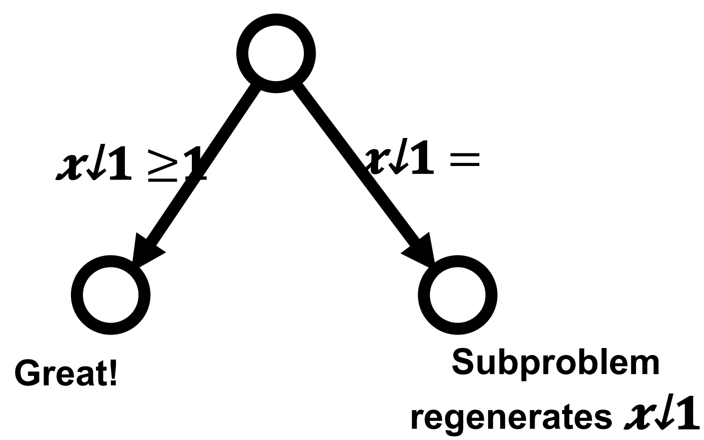
Branch-and-price

- Branching possibilities
  - Branch on master variables



Branch-and-price

- Branching possibilities
  - Branch on master variables



Branch-and-price

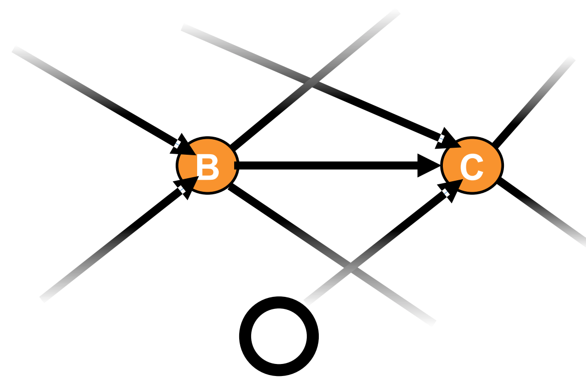
- Branching possibilities
  - Branch on master variables… NO!

Branch-and-price

- Branching possibilities
  - Branch on master variables… NO!
  - Branch on subproblem variables

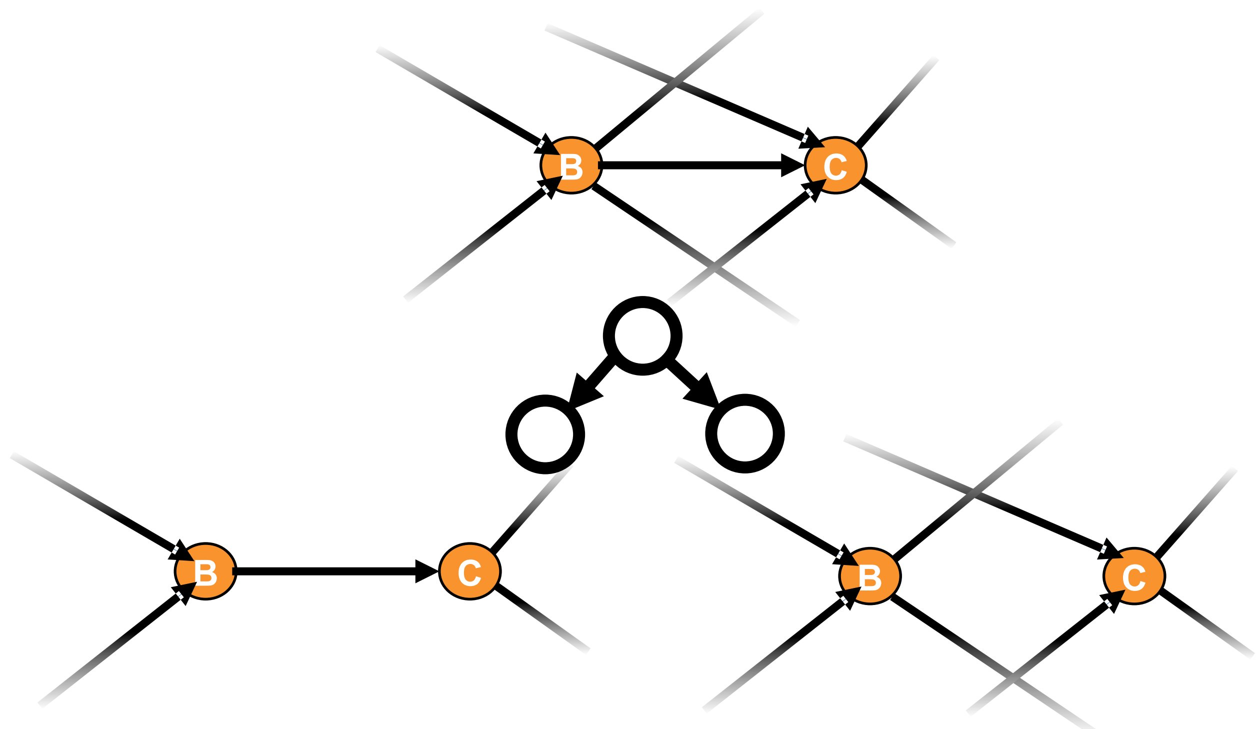
Branch-and-price

- Branching possibilities
  - Branch on master variables… NO!
  - Branch on subproblem variables



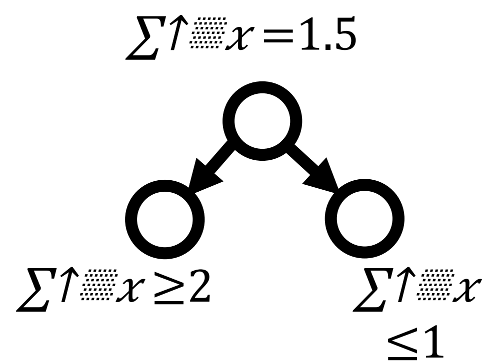
Branch-and-price

- Branching possibilities
  - Branch on master variables… NO!
  - Branch on subproblem variables



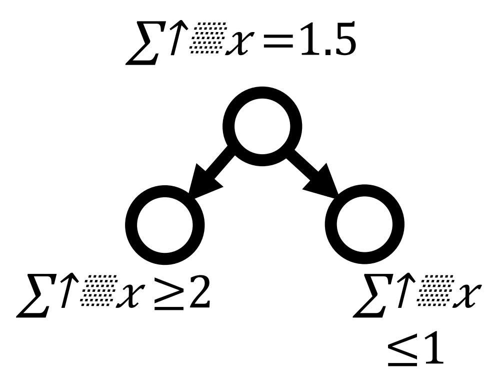
Branch-and-price

- Branching possibilities
  - Branch on master variables… NO!
  - Branch on subproblem variables
  - Branch on the master problem constraints
    - Add constraints c-> π_c must be added to the subproblems
    - Example: Branch on the total number of vehicle used



Branch-and-price

- Branching possibilities
  - Branch on master variables… NO!
  - Branch on subproblem variables
  - Branch on the master problem constraint
    - Add constraints -> π_i to add to the subproblems
    - Example: Branch on the total number of vehicle used



Best branching for shift scheduling problem

Branch-and-price

- Branching possibilities
  - Branch on master variables… NO!
  - Branch on subproblem variables
  - Branch on the master problem constraints
  - Cuts
    - Again dual variable $\pi\downarrow$ must be added to add to the subproblems

Applied column generation

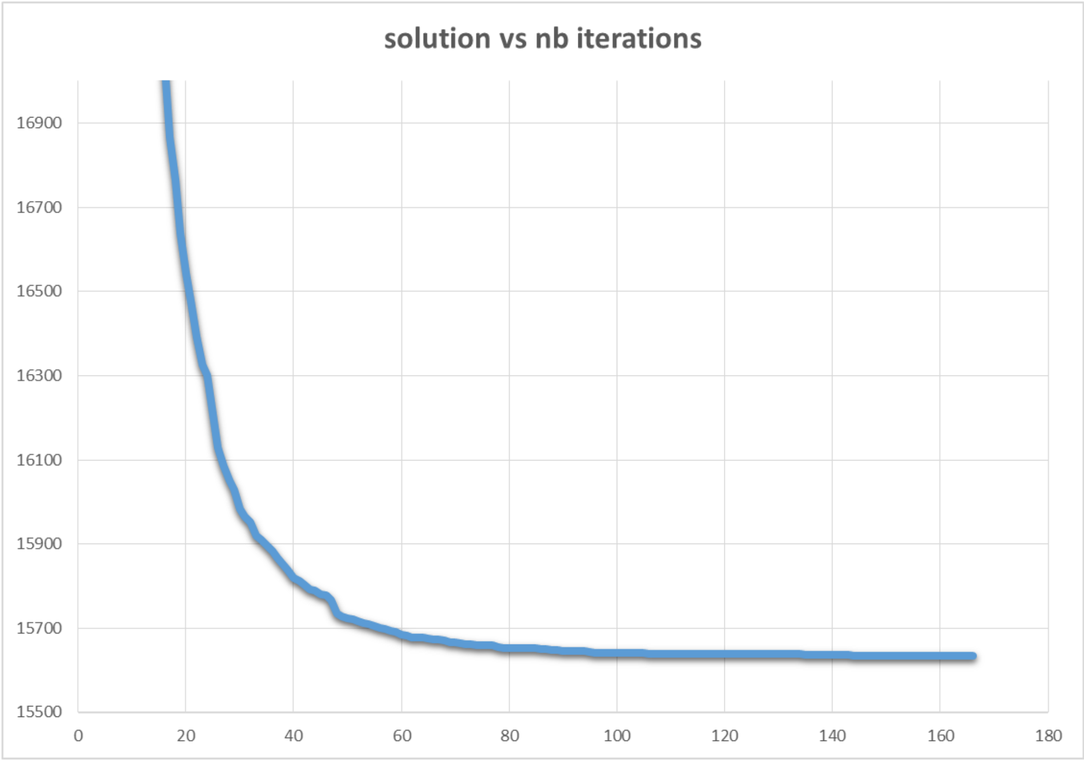
Main Challenges

Applied column generation

- Evolution of costs
  - Long convergence time

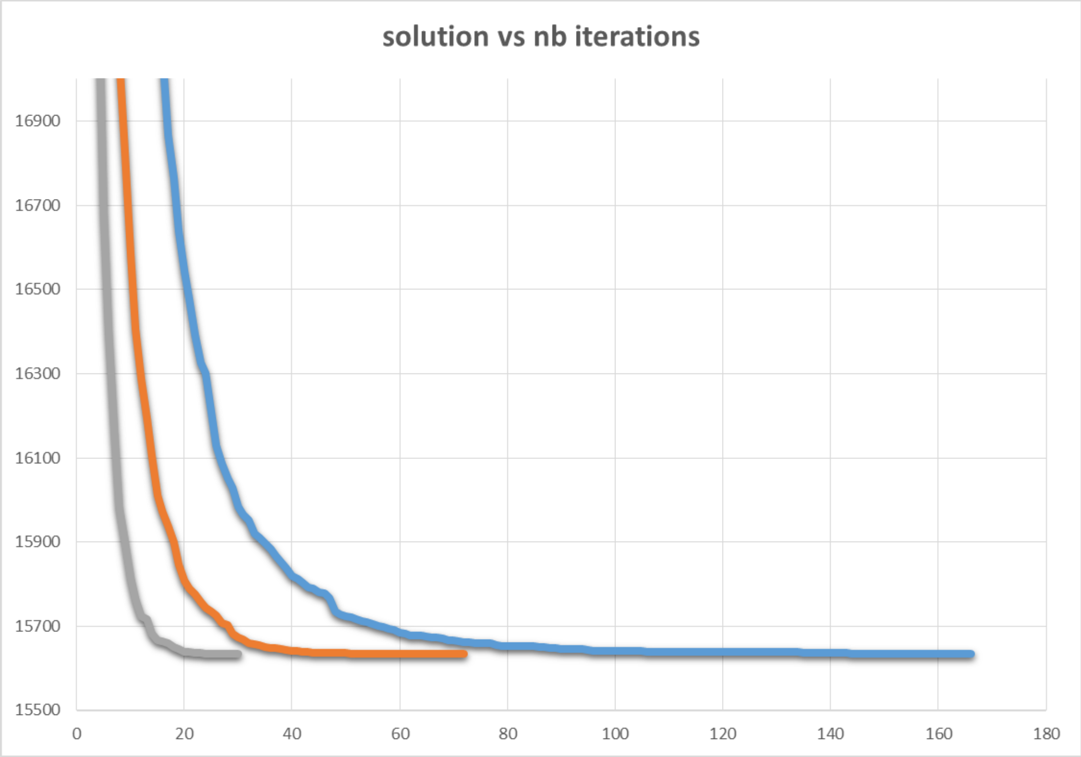
Applied column generation

- Evolution of costs
  - Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns



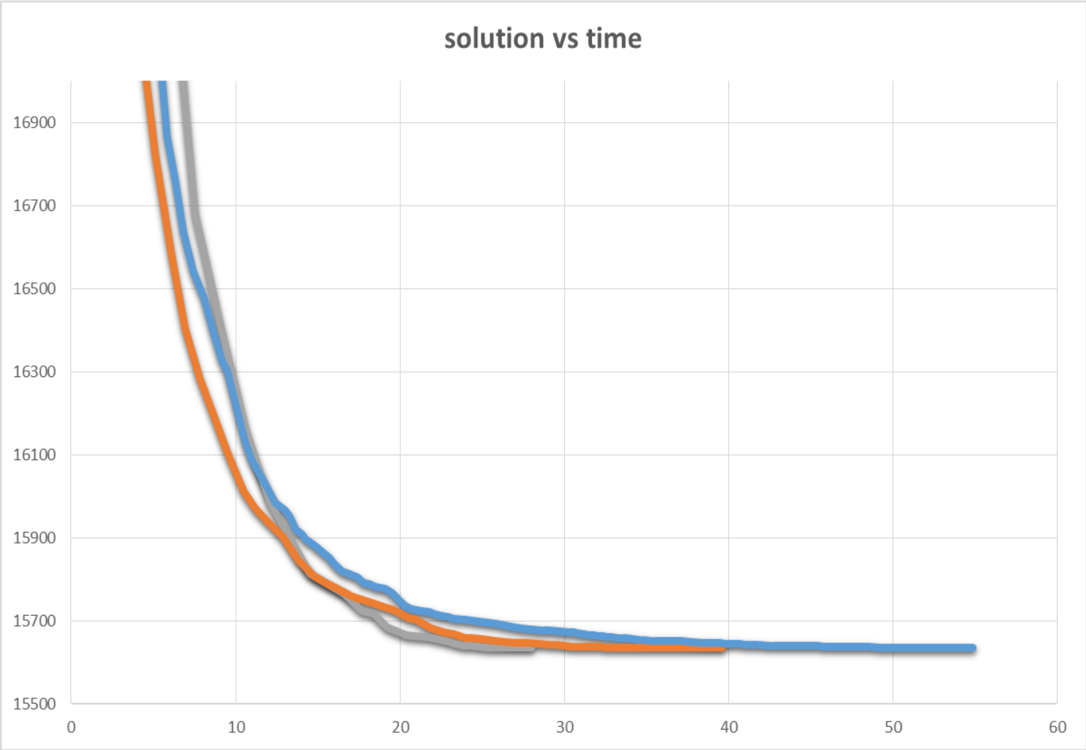
Applied column generation

- Evolution of costs
  - Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns



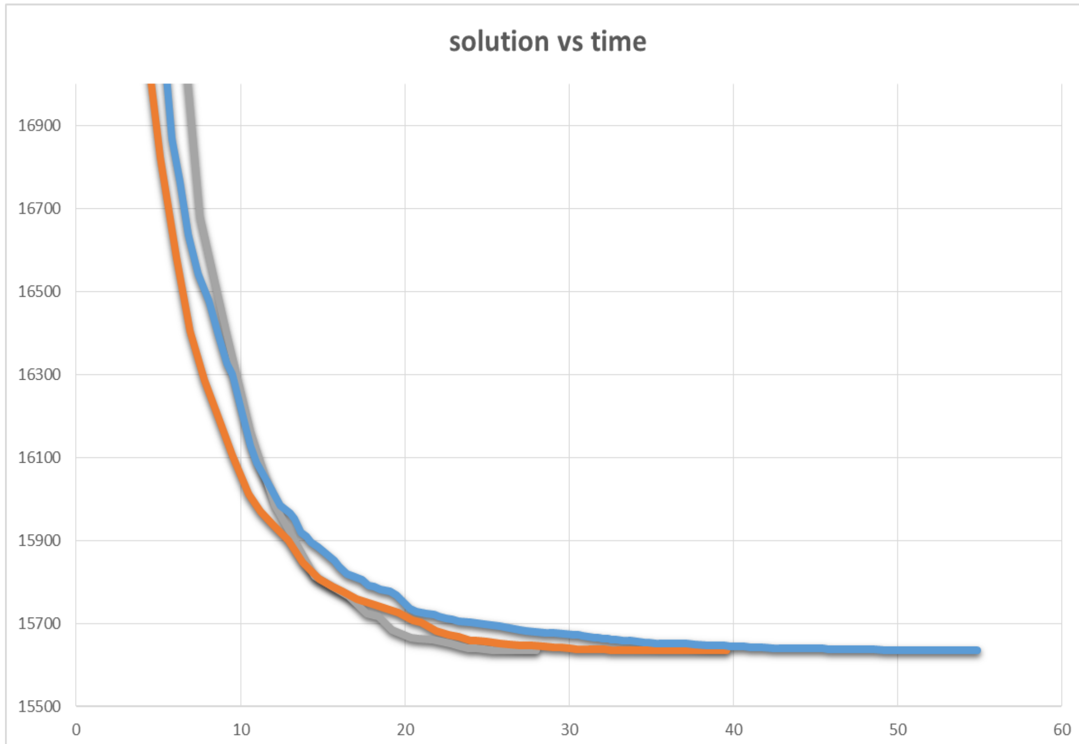
Applied column generation

- Evolution of costs
  - Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns



Applied column generation

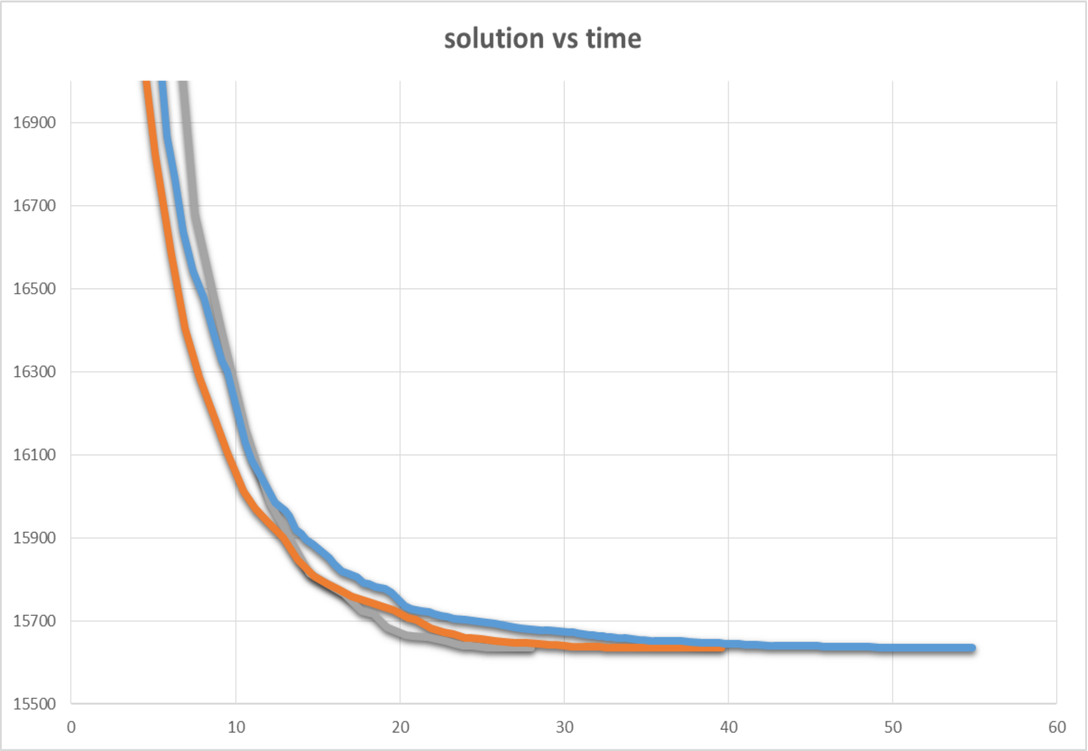
- Evolution of costs
  - Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns
  - Delete variables in RMP



Applied column generation

- Evolution of costs
  - Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns
  - Delete variables in RMP

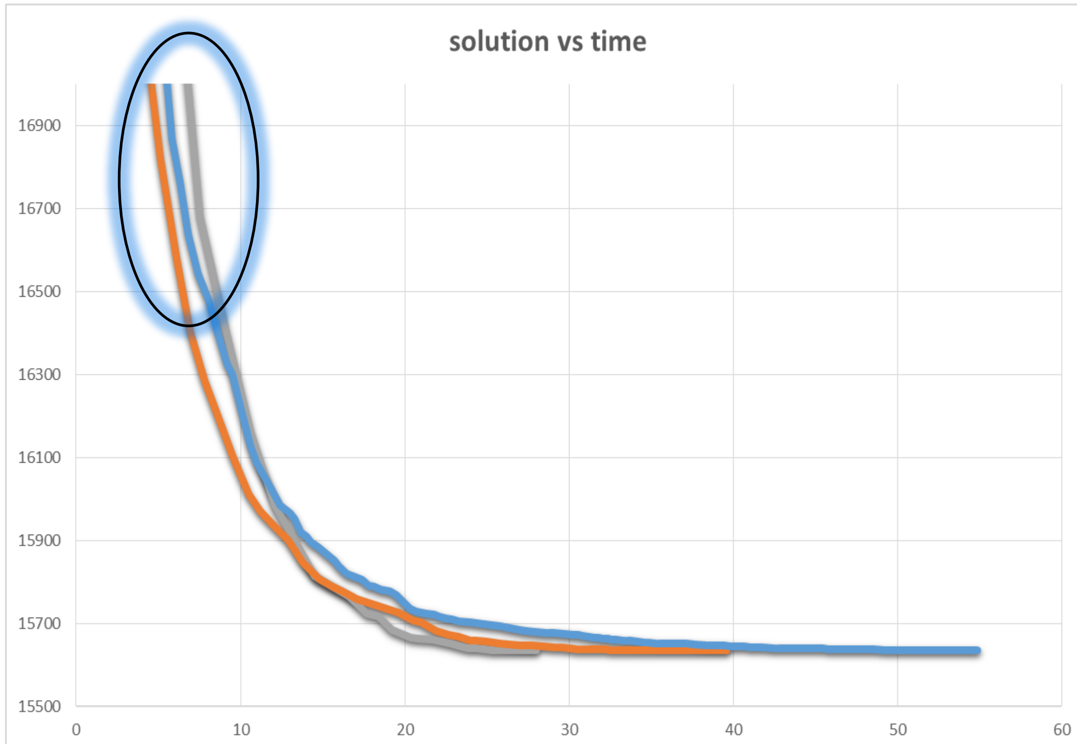
*Balance between subproblems and master problem



Applied column generation

- Evolution of costs
  - Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns
  - Delete variables in RMP

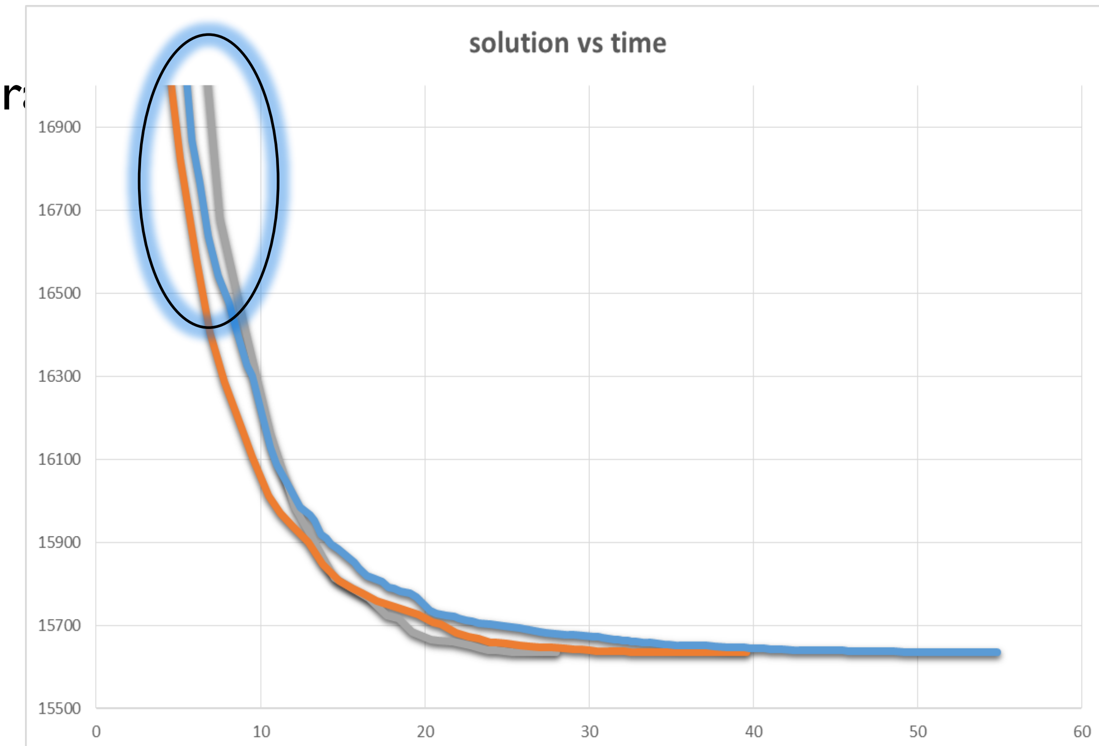
*Balance between subproblems and master problem



Applied column generation

- Evolution of costs
  - Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns
  - Delete variables in RMP
  - Gradually increase computation effort

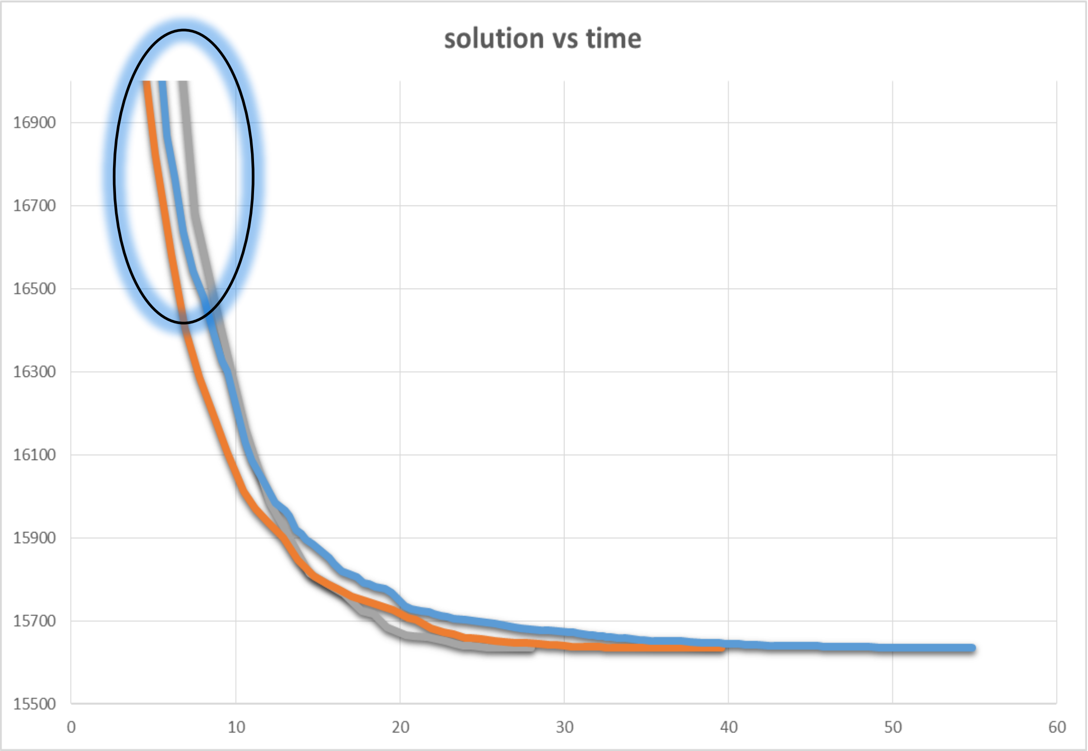
*Balance between subproblems and master problem



Applied column generation

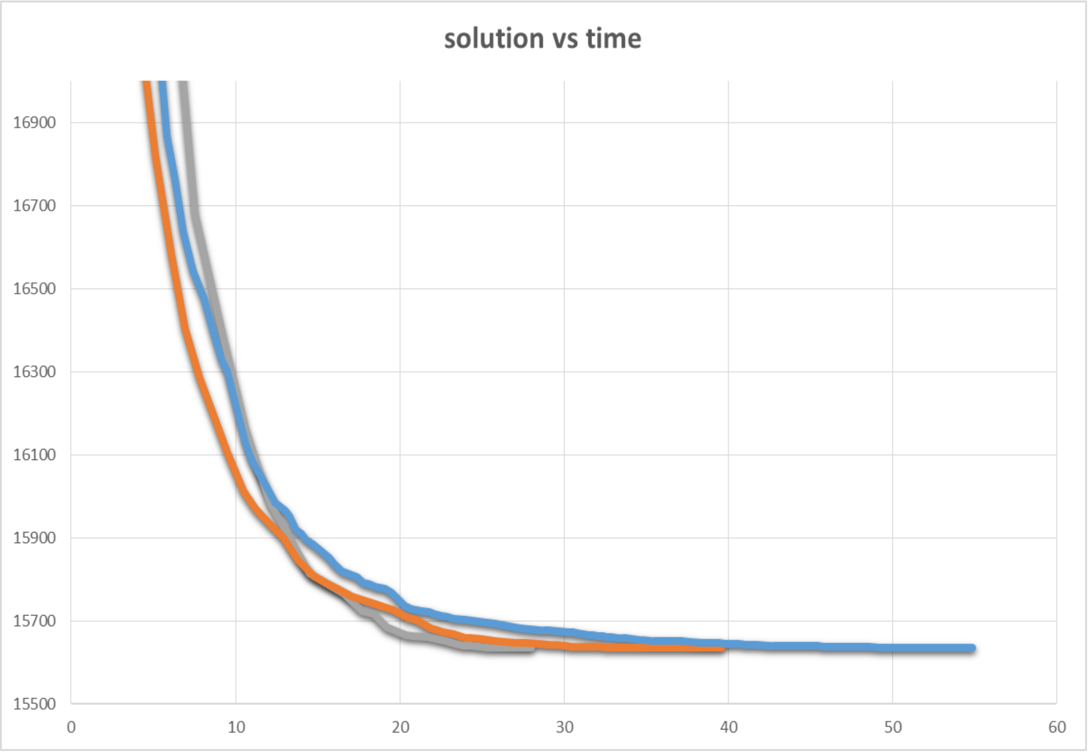
- Evolution of costs
  – Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns
  - Delete variables in RMP
  - Gradually increase computation effort
  - Heuristic pricing

*Balance between subproblems and master problem



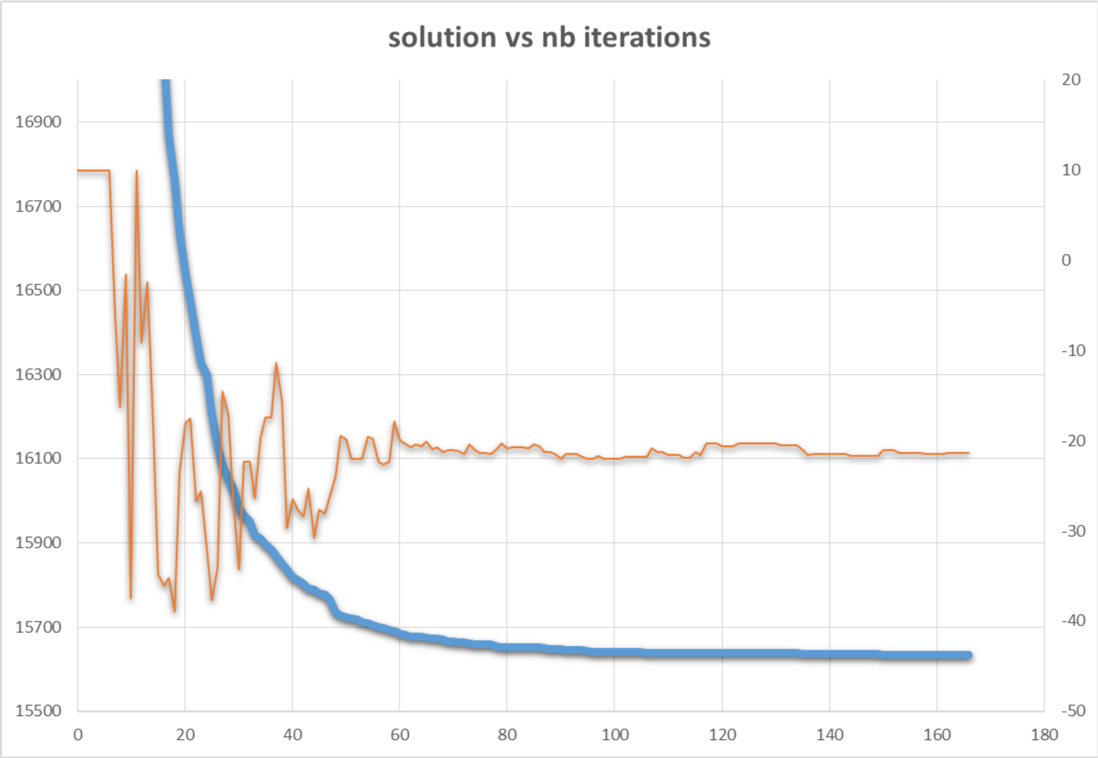
Applied column generation

- Evolution of costs
  - Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns
  - Delete variables in RMP
  - Gradually increase computation effort
  - Heuristic pricing
  - **Stabilization**



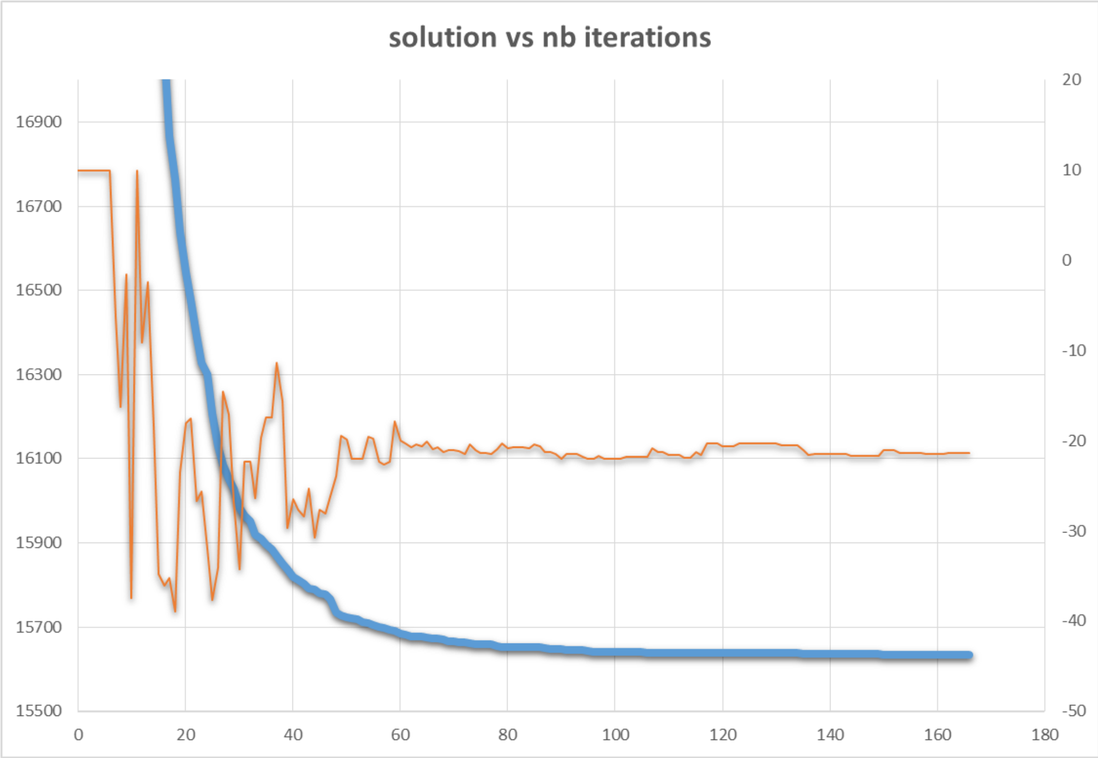
Applied column generation

- Evolution of costs
  - Long convergence time
- Speed-up techniques
  - Spend more time to generate new columns
  - Delete variables in RMP
  - Gradually increase computation effort
  - Heuristic pricing
  - **Stabilization**



Applied column generation

- Stabilization
  - Duals are extreme points
  - Master problem is degenerated
  - Tail-off effect is due to difficulty finding the right dual vector



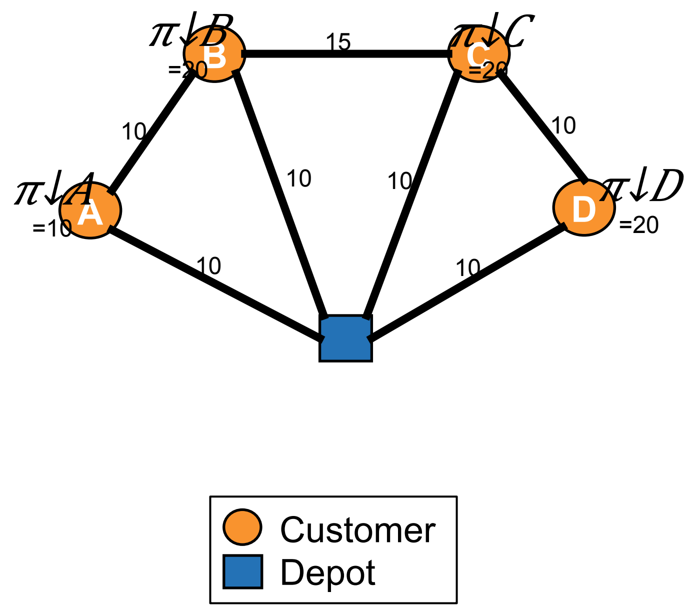
A quick look at

Stabilization issues

Column Generation

- Stabilization

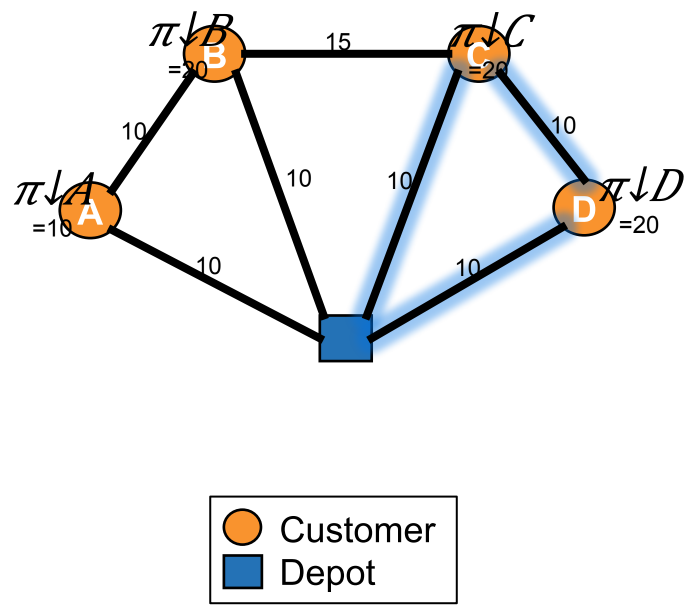
	x_1	x_2	x_3	x_4	x_5		π_i
ĉ	10	0	0	0	0		
A :	1				1	= 1	10
B :		1			1	= 1	20
C :			1			= 1	20
D :				1		= 1	20
		0	1	1	1	70	



Column Generation

- Stabilization

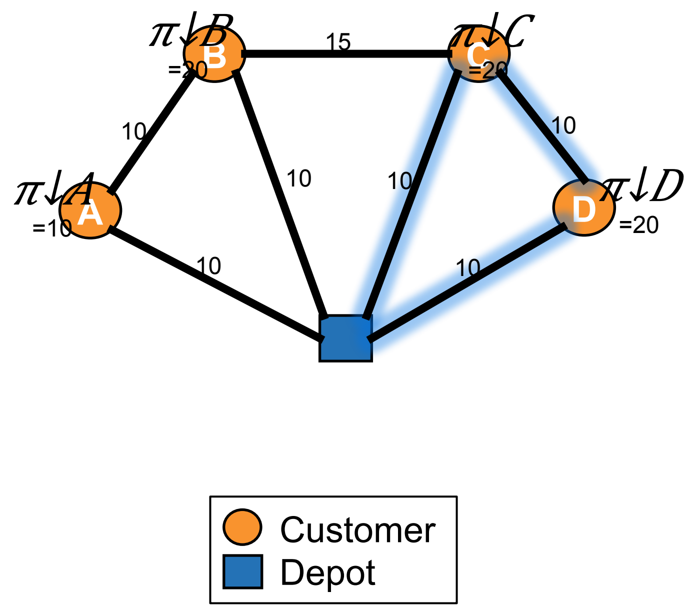
	x_1	x_2	x_3	x_4	x_5		π_i
ĉ	10	0	0	0	0		
A :	1				1	= 1	10
B :		1			1	= 1	20
C :			1			= 1	20
D :				1		= 1	20
		0	1	1	1	70	



Column Generation

- Stabilization

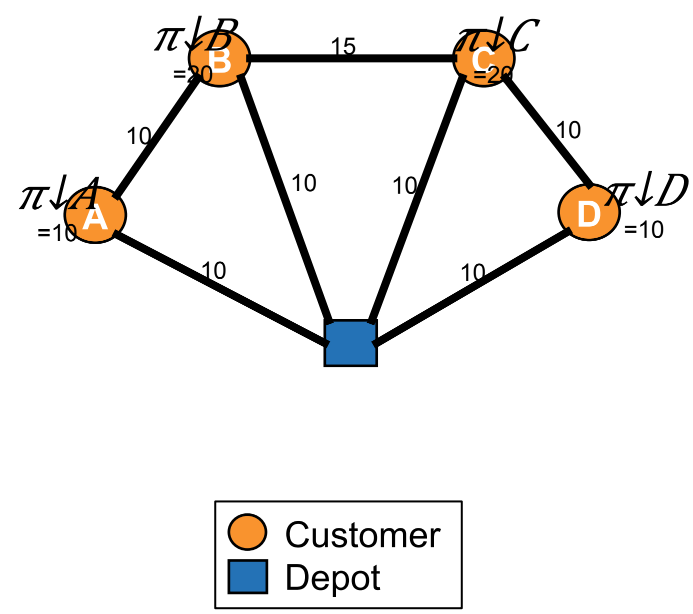
	x_1	x_2	x_3	x_4	x_5	x_6		π_i
ĉ	10	0	0	0	0	-10		
A :	1				1		= 1	10
B :		1			1		= 1	20
C :			1			1	= 1	20
D :				1		1	= 1	20
		0	1	1	1		70	



Column Generation

- Stabilization

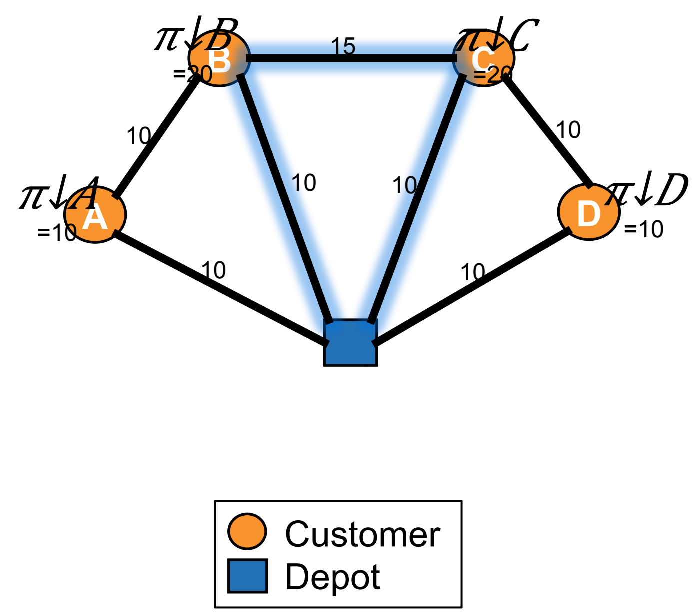
	x_1	x_2	x_3	x_4	x_5	x_6		
ĉ	10	0	0	10	0	0		π_i
A :	1				1		= 1	10
B :		1			1		= 1	20
C :			1			1	= 1	20
D :				1		1	= 1	10
		0	0		1	1	60	



Column Generation

- Stabilization

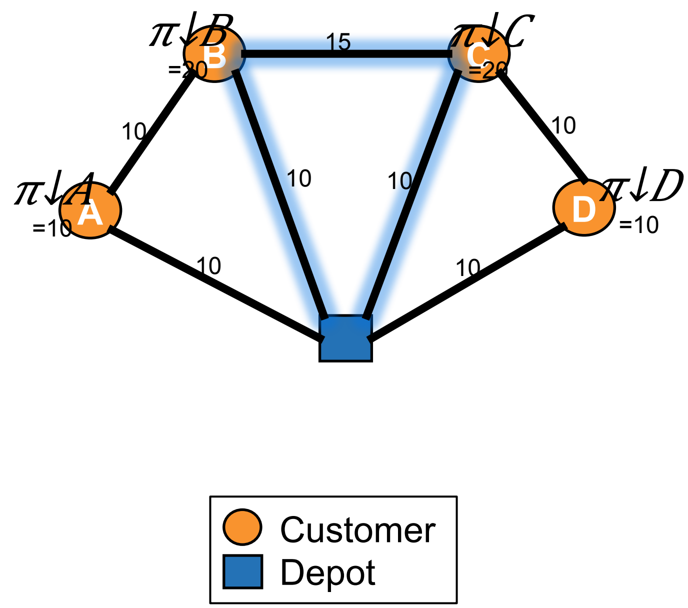
	x_1	x_2	x_3	x_4	x_5	x_6		π_i
ĉ	10	0	0	10	0	0		
A :	1				1		= 1	10
B :		1			1		= 1	20
C :			1			1	= 1	20
D :				1		1	= 1	10
		0	0		1	1	60	



Column Generation

- Stabilization

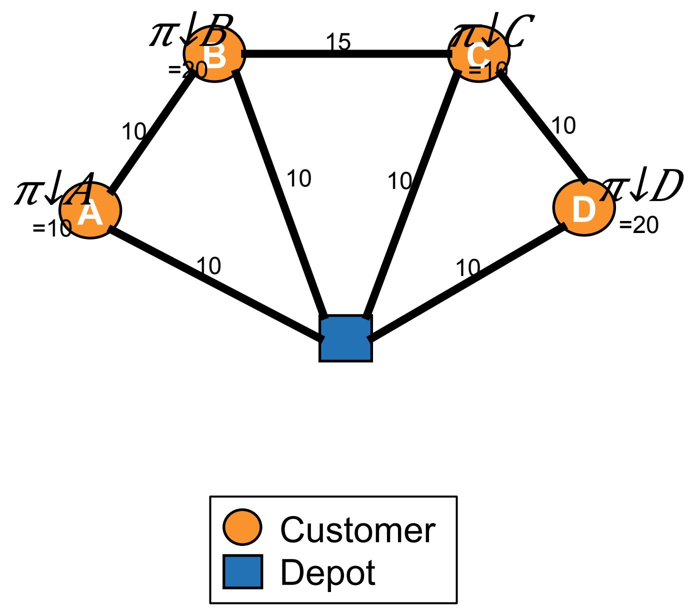
	x_1	x_2	x_3	x_4	x_5	x_6	x_7		
ĉ	10	0	0	10	0	0	-5		π_i
A :	1				1			= 1	10
B :		1			1		1	= 1	20
C :			1			1	1	= 1	20
D :				1		1		= 1	10
		0	0		1	1		60	



Column Generation

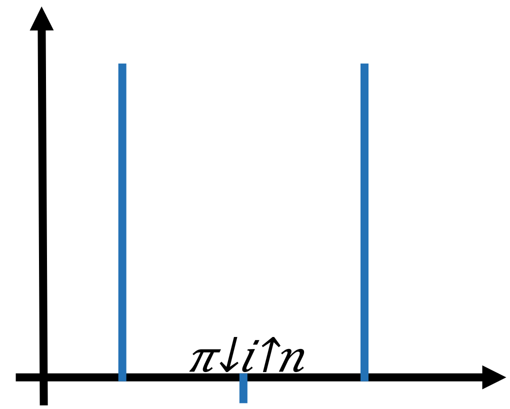
- Stabilization

	x_1	x_2	x_3	x_4	x_5	x_6	x_7		π_i
ĉ	10	0	10	0	0	0	5		
A :	1				1			= 1	10
B :		1			1		1	= 1	20
C :			1			1	1	= 1	10
D :				1		1		= 1	20
		0		0	1	1		60	



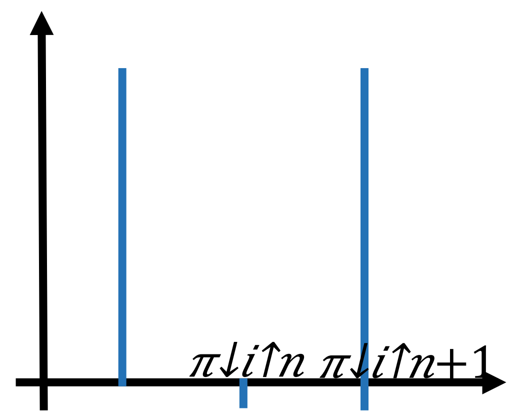
Column Generation

- Stabilization!
  - What to do?
  - Popular technique
    - Box penalization



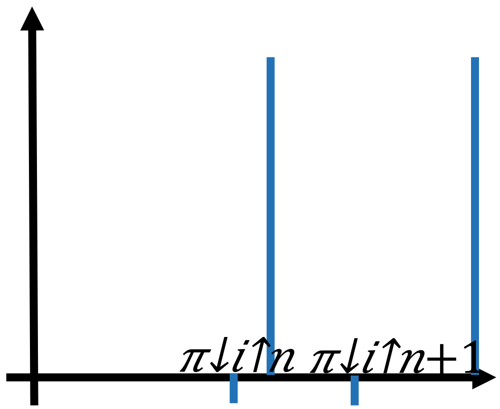
Column Generation

- Stabilization!
  - What to do?
  - Popular technique
    - Box penalization



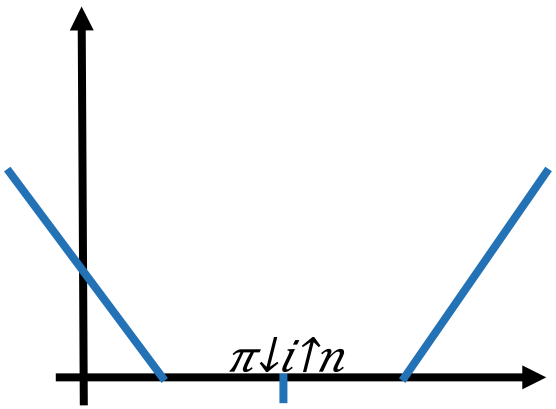
Column Generation

- Stabilization!
  - What to do?
  - Popular technique
    - Box penalization



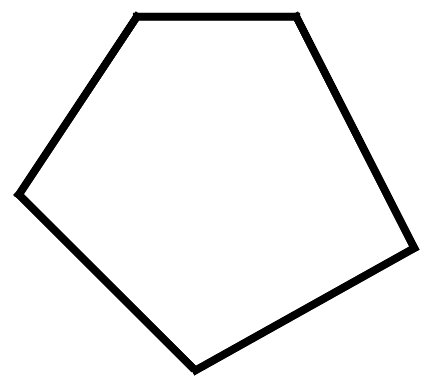
Column Generation

- Stabilization!
  - What to do?
  - Popular technique
    - Box penalization



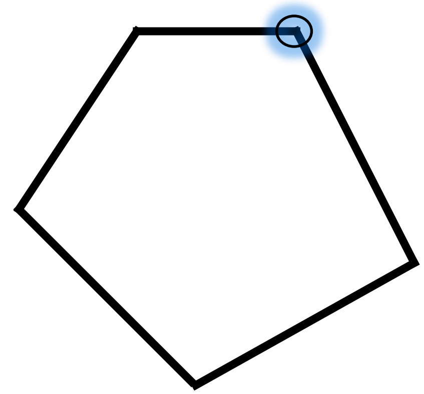
Column Generation

- Stabilization!
  - What to do?
  - Popular technique
    - Box penalization
  - Interior point stabilization



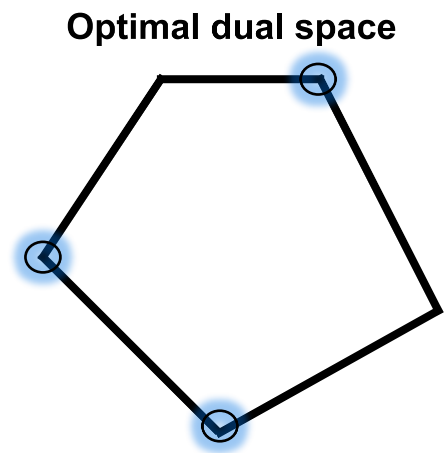
Column Generation

- Stabilization!
  - What to do?
  - Popular technique
    - Box penalization
  - Interior point stabilization
    - Adding a variable to the primal
      is equivalent to adding a cut to the dual



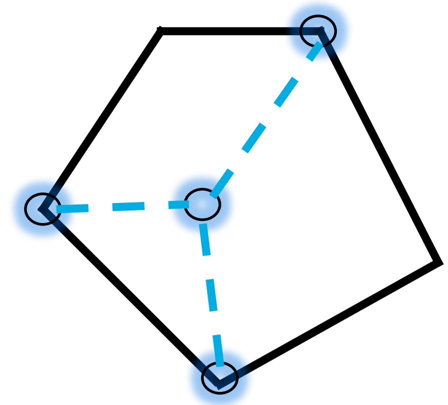
Column Generation

- Stabilization!
  - What to do?
  - Popular technique
    - Box penalization
  - Interior point stabilization
    - Find multiple dual optimal extreme points



Column Generation

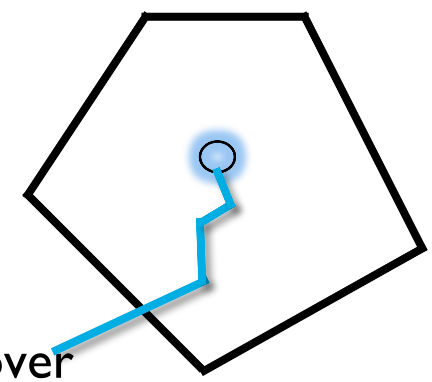
- Stabilization!
  - What to do?
  - Popular technique
    - Box penalization
  - Interior point stabilization
    - Find multiple dual optimal extreme points
      - Do a linear combination



	Average time	Average nb Iterations
Unstabilized	384.4 s	72.6
Box penalization	389.1 s	61.0
IPS	277.9 s	37.1

Column Generation

- Stabilization!
  - What to do?
  - Popular technique
    - Box penalization
  - Interior point stabilization
    - Find multiple dual optimal extreme points
      - Do a linear combination
    - Simple idea: barrier algorithm without crossover



Any Questions ?

Thank you !