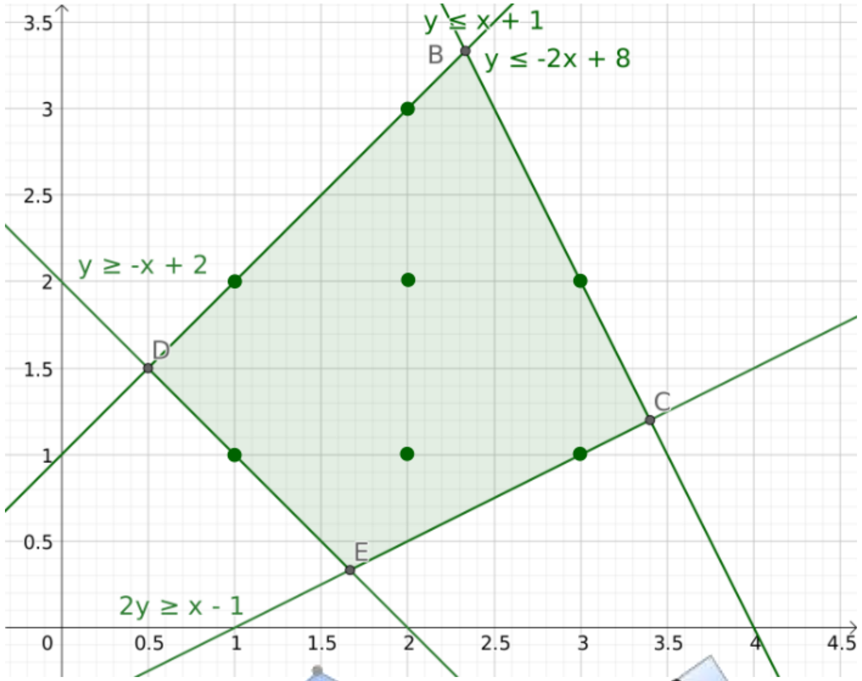
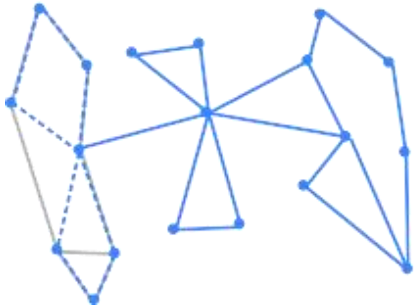
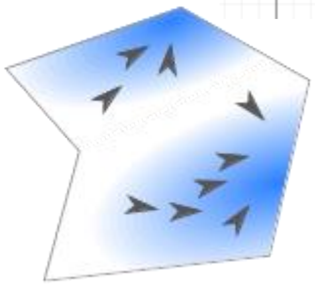
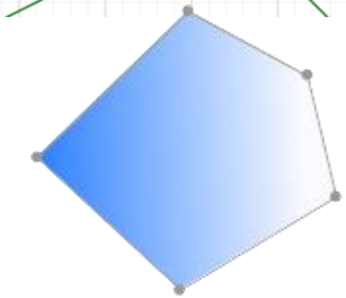
Decomposition 1

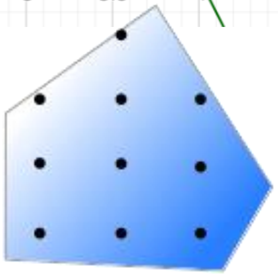
COMP4691/8691

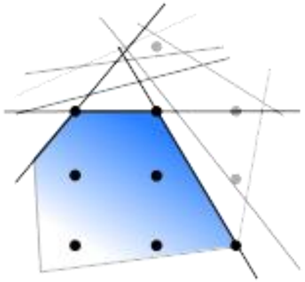


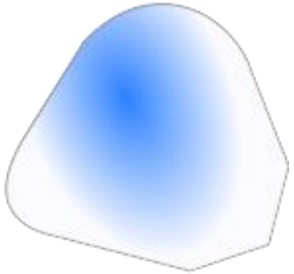












Decompositions Methods

- Huge Linear Programs can now be solved by commercial systems
  - Billions of (potential) entries in the constraint matrix
  - 100s of thousands of variables **and** 100s of thousands of constraints
  - (but the matrix is usually sparse)

- However ILPs are still problematic
  - They are $\mathcal{NP}$ hard, so size is going to bite sometime

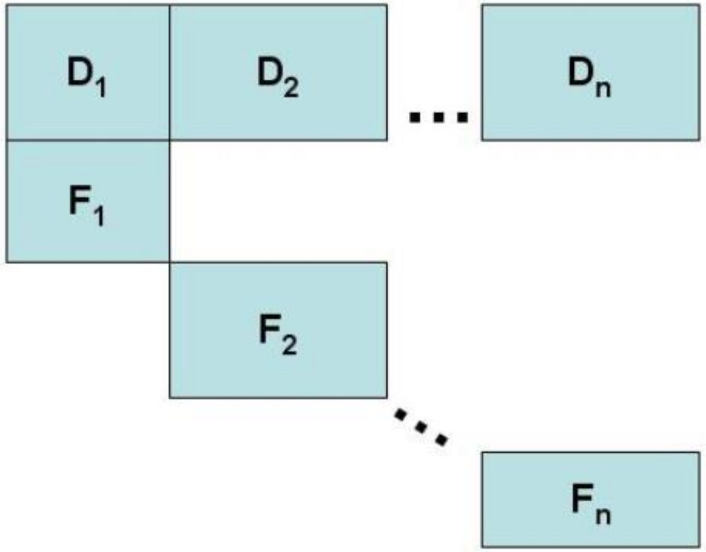
- *Decomposition* methods offer a way of solving larger problems
  - Break up the problem
  - Remove complicating bits
  - Look at the problem in a different way

Benders' decomposition

Column Generation

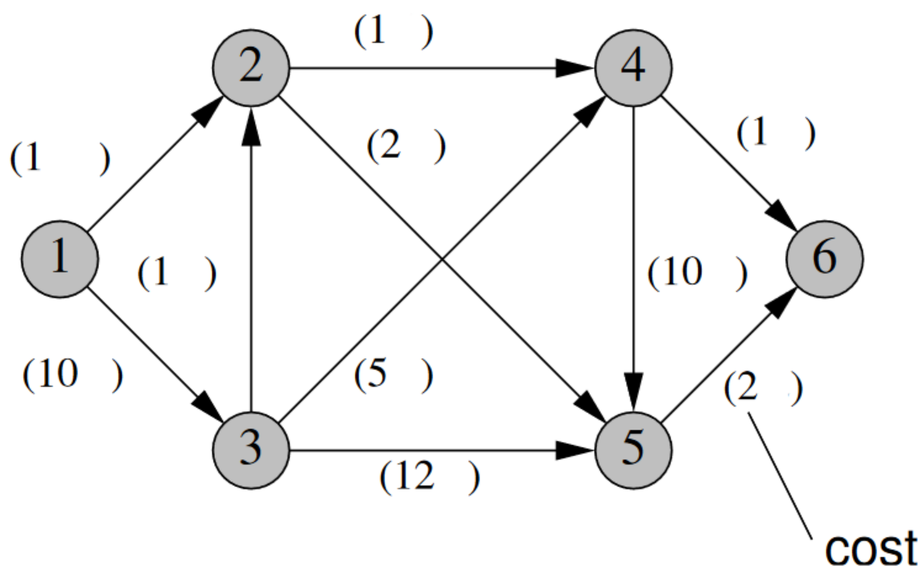
Decomposition Topic Outline

- **Column Generation**
  - Constrained Shortest Path
  - Cutting Stock
  - Dantzig-Wolfe decomposition
- Bender's Decomposition



Shortest Path Problem

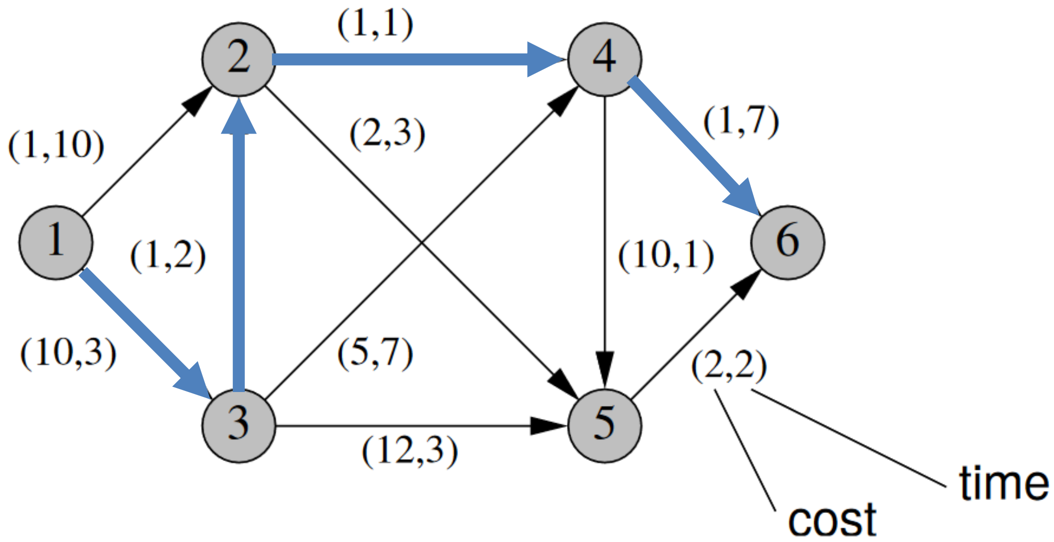
- We have seen several ways to solve Shortest Path Problems in AI



- Initial state/node: 1
- Goal state/node: 6

Constrained Shortest Path Problem

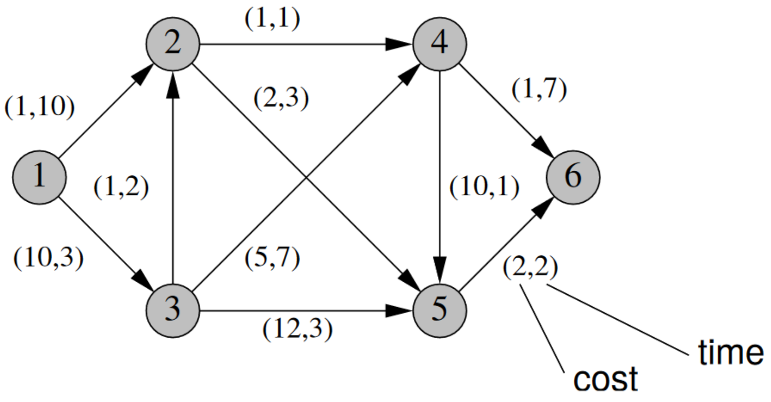
- Let's add the following constraint:
  - We cannot exceed 14 units of time to reach the goal



Optimal solution:
1 → 3 → 2 → 4 → 6
cost: 13
time: 13

- Can you use A* to solve this problem?
  - Not as it is. We would need to change the **problem representation** to distinguish reaching states at different times, e.g., reaching state 2 using
  - 1 unit of cost and 10 units of time (1 → 2)
  - 11 units of cost and 5 units of time (1 → 3 → 2) ← **Should you prune this option?**
- Complexity: Constrained SPs are $\mathcal{NP}$-hard

Constrained SP as an ILP



$$z^\star := \min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j:(1,j)\in A} x_{1j} = 1$$

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 \qquad i = 2,3,4,5$$

$$\sum_{i:(i,6)\in A} x_{i6} = 1$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij} \le 14$$

$$x_{ij} \in \{0,1\} \quad (i,j) \in A$$

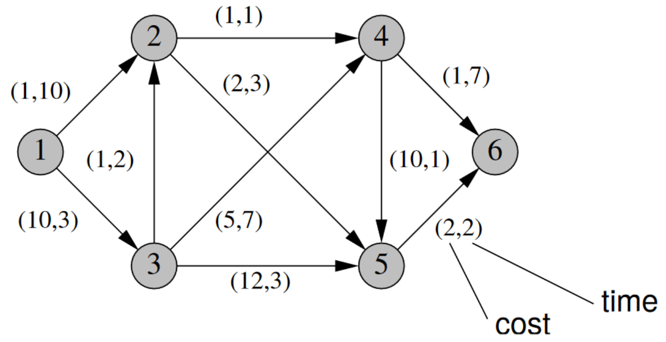
- Flow model
  - push 1 unit of flow into node 1
  - preservation of flow
  - extract 1 unit of flow from node 6
- Each arc (i,j) ∈ A is either used or not
- We can solve with Branch and Bound

Reformulation

- The "first step" of Column Generation (CG) is to reformulate the problem such that we separate the "hard" part from the "easy" one
- In CG, the **hard part is usually the complicating constraints**
- In the Constrained Shortest Path:
  - **Hard part:** we cannot exceed 14 units of time to reach the goal
  - **Easy part:** solving **unconstrained** shortest path problems
- Let's reformulate the problem using paths as variables

Path Reformulation

- Let's enumerate all paths and write an ILP that chooses the optimal one
  - y_{1246} is a binary variable representing the path 1→2→4→6
  - Object function coefficient is the cost of the path
  - Coefficient in time constraint is the time used by this path



Variables (Columns) used by Column Generation

$$
\begin{array}{rl}
\min & 3y_{1246} + 14y_{12456} + 5y_{1256} + 13y_{13246} + 24y_{132456} + 15y_{13256} + 16y_{1346} + 27y_{13456} + 24y_{1356} \\
\text{s.t.} & 18y_{1246} + 14y_{12456} + 15y_{1256} + 13y_{13246} + 9y_{132456} + 10y_{13256} + 17y_{1346} + 13y_{13456} + 8y_{1356} \leq 14 \\
& y_{1246} + y_{12456} + y_{1256} + y_{13246} + y_{132456} + y_{13256} + y_{1346} + y_{13456} + y_{1356} = 1 \\
& y_{1246},\ y_{12456},\ y_{1256},\ y_{13246},\ y_{132456},\ y_{13256},\ y_{1346},\ y_{13456},\ y_{1356} \in \{0, 1\}
\end{array}
$$

- This ILP is correct and will find the opt. solution. What is the issue with it?
  - It has **a lot of variables!**
- But do we need all the path variables? **No**

Path Reformulation ILP

$$z^\star = \min \sum_{p \in P} \left(\sum_{(i,j) \in A} c_{ij} x_{pij} \right) y_p$$

$$\text{s.t.} \quad \sum_{p \in P} \left(\sum_{(i,j) \in A} t_{ij} x_{pij} \right) y_p \leq 14$$

$$\sum_{p \in P} y_p = 1$$

$$y_p \geq 0 \quad p \in P$$

$$\sum_{p \in P} x_{pij} y_p = x_{ij} \quad (i,j) \in A$$

$$x_{ij} \in \{0,1\} \quad (i,j) \in A$$

Inner summation is the total time consume by path p

We want a convex combination of paths.
Together with $x_{ij} \in \{0,1\}$, only one path will be selected

Relax the path variables to be continuous

x_{pij} is a **binary constant** and it is 1 if i→j in path p
This constraints links x_{ij} and y_p

Each arc can be used at most once

- Column Generation idea: the constraint matrix is so large (and implicit) that we will generate it on demand
  - In the Constraint SP, we will generate the paths on demand

Solving the Linear Relaxation

- Let's focus on the linear relaxation first, then the combine it with B&B

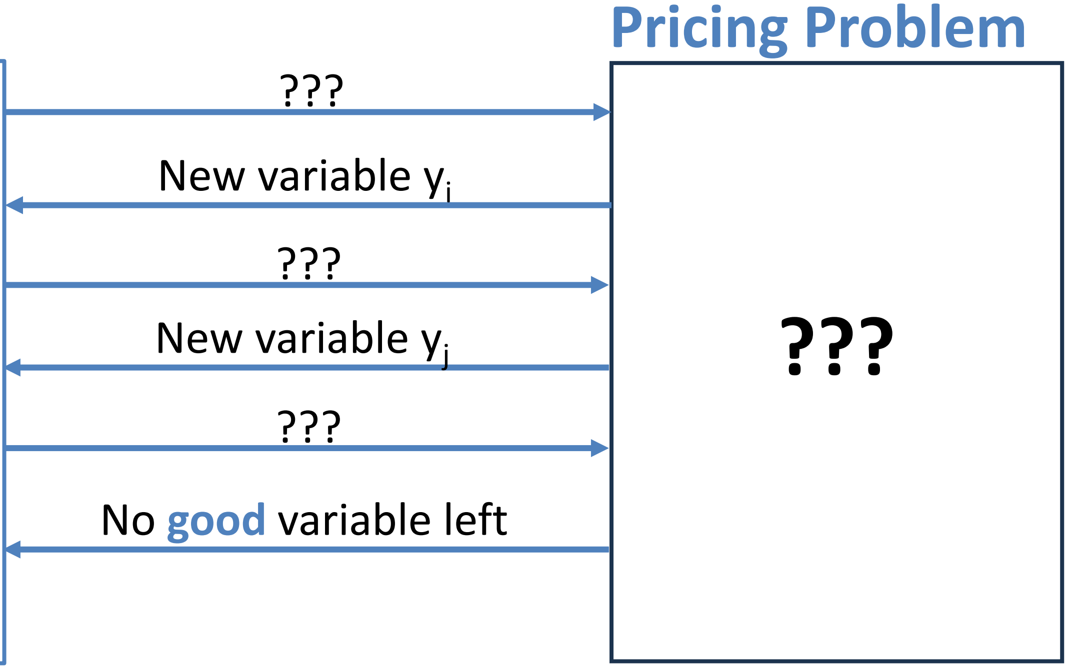
$$
\begin{aligned}
z^\star = \min \sum_{p \in P} \left(\sum_{(i,j) \in A} c_{ij} x_{pij} \right) y_p & \\
\text{s.t.} \quad \sum_{p \in P} \left(\sum_{(i,j) \in A} t_{ij} x_{pij} \right) y_p & \leq 14 \\
\sum_{p \in P} y_p & = 1 \\
y_p & \geq 0 \quad p \in P \\
\cancel{\sum_{p \in P} x_{pij} y_p} & \cancel{= x_{ij} \quad (i,j) \in A} \\
\cancel{x_{ij}} & \cancel{\in \{0,1\} \quad (i,j) \in A}
\end{aligned}
$$

Column Generation: Road Map

- We have two optimization problems:
  - **Reduced Master Problem** (RMP): original problem with just a subset of variables
  - **Pricing problem**: the problem that will generate new variables (columns) for the RMP
  - The original reformulated problem is referred as the **Master Problem**
- For the Linear Relaxation of Constrained Shortest Path:

Reduced Master Problem (RMP)

$$
\begin{aligned}
z^\star = \min \sum_{p \in P'} \left(\sum_{(i,j) \in A} c_{ij} x_{pij} \right) y_p & \\
\text{s.t.} \quad \sum_{p \in P'} \left(\sum_{(i,j) \in A} t_{ij} x_{pij} \right) y_p \leq 14 & \\
\sum_{p \in P'} y_p = 1 & \\
y_p \geq 0 \quad p \in P' \subseteq P &
\end{aligned}
$$



Pricing Problem

- Goal: find a variable to be added to the LP to improve its objective function
- Does it sound familiar? **Revised Simplex**

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\intercal x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Basic Variables: $x_j \geq 0 \quad \forall j \in B_k$ $\qquad \bar{A}\bar{x} = b$

Non-Basic Variables: $x_j = 0 \quad \forall j \notin B_k$ $\qquad \tilde{x} = 0$

Reduced Cost: the marginal increase in objective for a marginal increase in each **non-basic variable**

Pick entering variable: $\bar{A}^\intercal \lambda = \bar{c} \Rightarrow \mu = \tilde{c} - \tilde{A}^\intercal \lambda \Rightarrow u \in \{u | \mu_u < 0\}$

(otherwise optimal)

Dual variables for the **basic variables**

Column Generation: Road Map (2)

Reduced Master Problem (RMP)

$$z^\star = \min \sum_{p \in P'} \left(\sum_{(i,j)\in A} c_{ij} x_{pij} \right) y_p$$

$$\text{s.t.} \quad \sum_{p \in P'} \left(\sum_{(i,j)\in A} t_{ij} x_{pij} \right) y_p \leq 14 \qquad (\lambda_1)$$

$$\sum_{p \in P'} y_p = 1 \qquad (\lambda_0)$$

$$y_p \geq 0 \quad p \in P' \subseteq P$$

Dual variables λ

New variable y_j

Dual variables λ

New variable y_j

Dual variables λ

No **good** variable left

Pricing Problem

Minimize the reduced cost

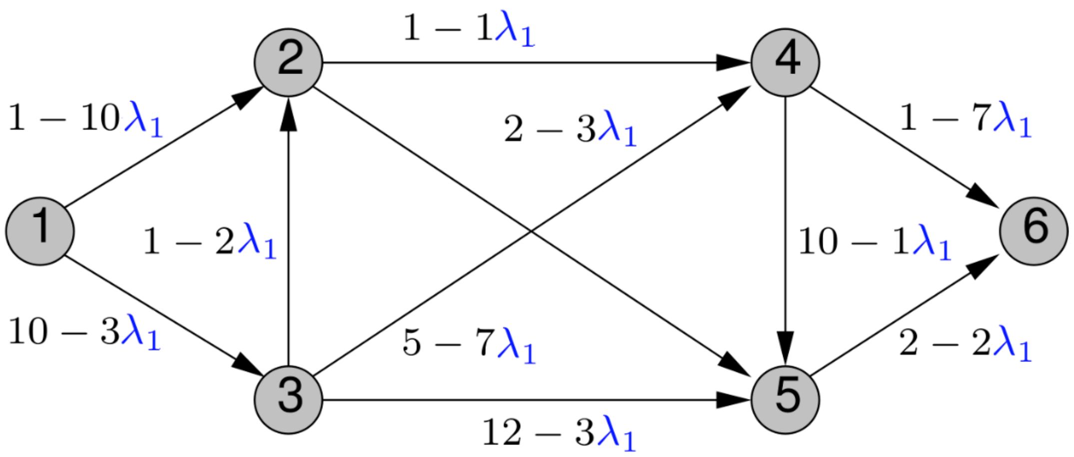
$$\mu = \tilde{c} - \tilde{A}^\intercal \lambda$$

- The pricing problem handles the search over the really large set of columns
- But why is this good?
  - The pricing problem does not have the "hard"/complicating constraint!
- What is the reduced cost of a path p, i.e., μ_p ?

$$\mu_p = \sum_{(i,j)\in A} c_{ij} x_{pij} - \left(\sum_{(i,j)\in A} t_{ij} x_{pij} \right) \lambda_1 - 1\lambda_0$$

Pricing Problem for Constrained Shortest Path

$$\min_p \mu_p = \min_p \sum_{(i,j)\in A} c_{ij} x_{pij} - \left(\sum_{(i,j)\in A} t_{ij} x_{pij} \right) \lambda_1 - 1\lambda_0 = \min_p \sum_{(i,j)\in A} (c_{ij} - t_{ij}\lambda_1)\, x_{pij} - \lambda_0$$



- We have seen this problem today. What is it?
  - (unconstrained) Shortest Path problem!
  - Cost to go from i to j now is $c_{ij} - t_{ij}\lambda_1$
  - This was the motivation of CG: to exploit a class of problems we can easily solve
  - We can use A* or anything else to solve the pricing problem now

Column Generation: Algorithm

1. Start with some columns
2. Solve the problem using those columns (**Restricted Master Problem**)
3. Use the dual variables to define a new subproblem (**Pricing Problem**)
4. Solve the pricing problem to obtain the column with **minimum reduced cost**
5. If the reduced cost of the new column is negative, add it to the RMP and go to 2. Otherwise, we have the optimal solution to the linear relaxation of the problem

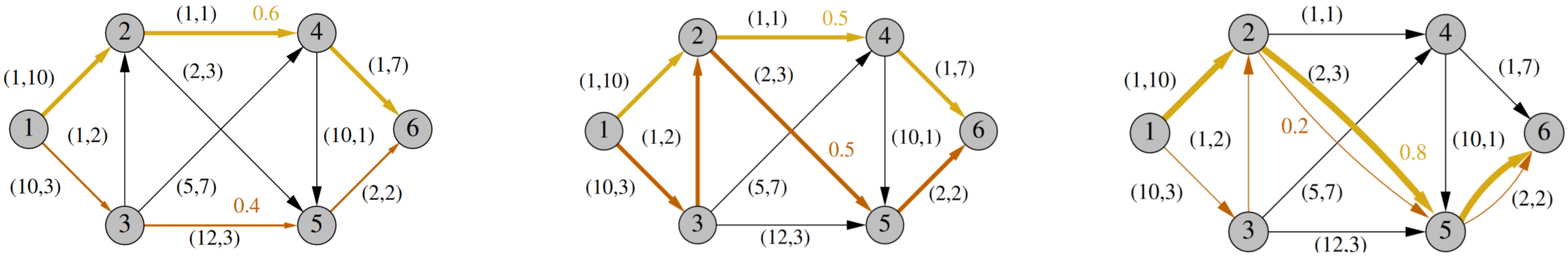
Constrained SP: Linear Relax. Full Example

RMP:

$$
\begin{array}{lcccccccccc}
\min & 3y_{1246} & + & 24y_{1356} & + & 15y_{13256} & + & 5y_{1256} & & & \\
\text{s.t.} & 18y_{1246} & + & 8y_{1356} & + & 10y_{13256} & + & 15y_{1256} & \leq & 14 & ({\color{blue}\lambda_1}) \\
 & y_{1246} & + & y_{1356} & + & y_{13256} & + & y_{1256} & = & 1 & ({\color{magenta}\lambda_0}) \\
 & y_{1246} & , & y_{1356} & , & y_{13256} & , & y_{1256} & \geq & 0 &
\end{array}
$$

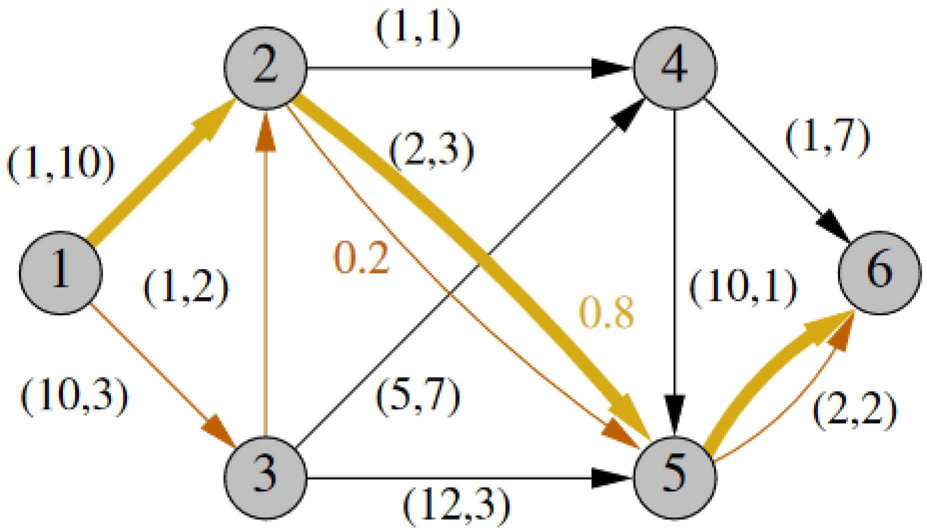
RMP solution	$\bar{z}$	λ_0	λ_1	Pricing Problem $\min_p \mu_p$	p	c_p	t_p
$y_{1246} = 0.6, y_{1356} = 0.4$	11.4	40.80	-2.10	-4.8	13256	15	10
$y_{1246} = y_{13256} = 0.5$	9.0	30.00	-1.50	-2.5	1256	5	15
$y_{13256} = 0.2, y_{1256} = 0.8$	7.0	35.00	-2.00	0	0	0	0

Visualization of the i-th RMP solution:



Integrating with Branch and Bound

- So far we solved the linear relaxation for the root node (no branching)

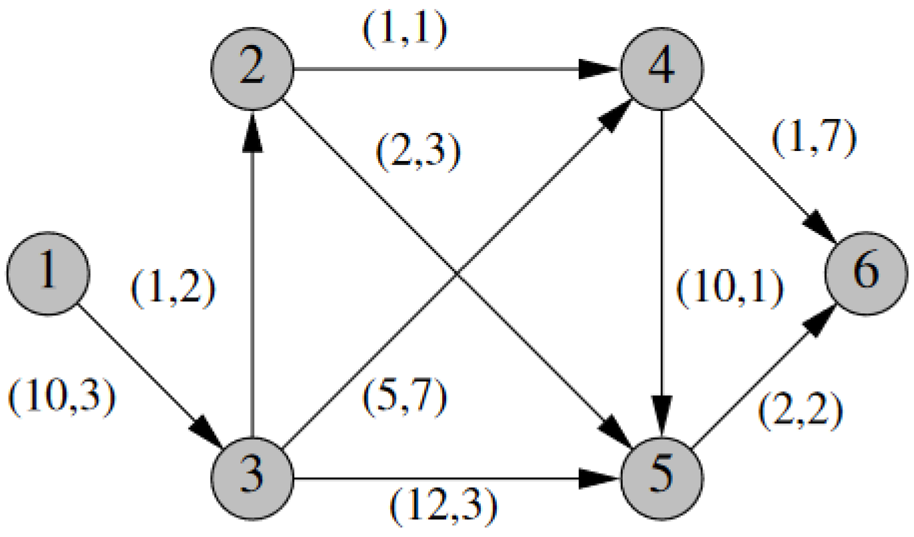


- Now we can branch on the fractional arc variables, e.g., $x_{12} = 0.8$
  - x_{12} == 0 in one branch
  - x_{12} == 1 in the other branch
- Solve each branch with column generation again

Constrained SP: Branching on x_{12}

Branch x_{12} == 0

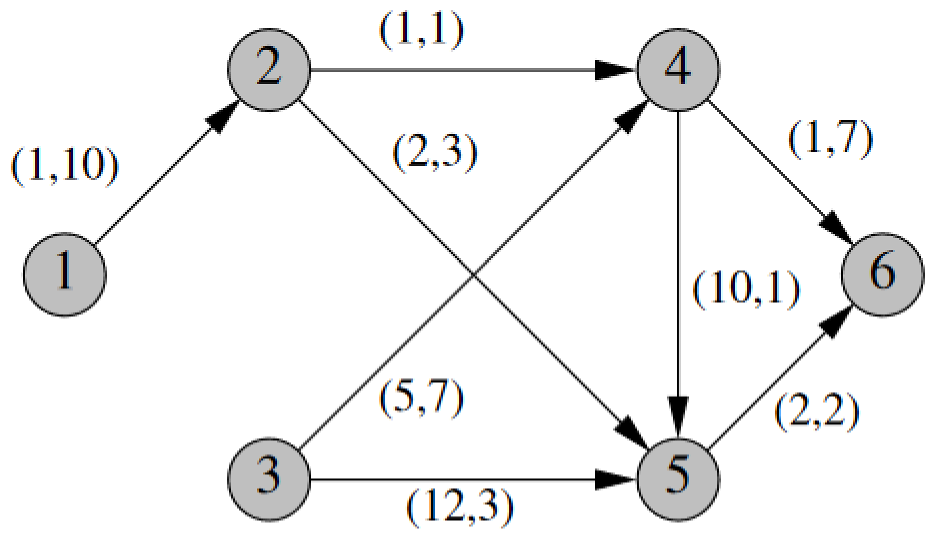
- Remove all y variables that use 1→2, e.g., y_{1246}, y_{1256}, etc.
- Can do so by removing arc 1 → 2 from the graph
- Constrained SP problem for this branch



- Linear Relax. Opt solution: $y_{13246} = 1$
- It's integer feasible!

Branch x_{12} == 1

- Remove all y variables that **do not use** 1→2
- Can do so by removing all arcs leaving 1 and all arcs entering 2 with exception of 1→ 2
- Constrained SP problem for this branch



- This Constrained SP problem is **infeasible**!

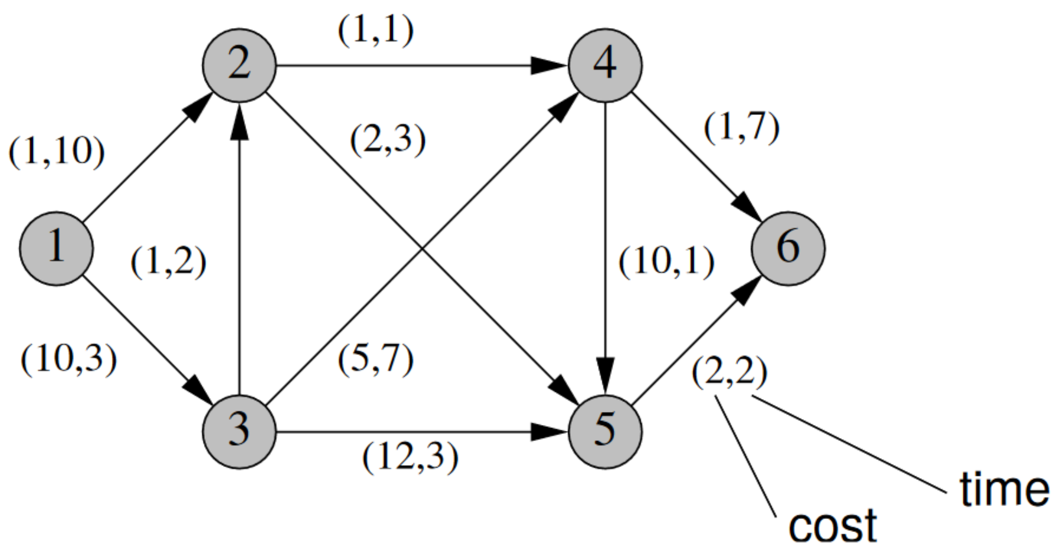
This approach (B&B + CG) is known as branch-and-price

Column Generation Remarks (1)

- We do not need to solve the pricing problem optimally
  - Any **negative** reduced cost column will improve the RMP solution
  - Optimality of the pricing problem is only needed to prove that no column with negative reduce cost exists and therefore the solution to the RMP is optimal
- We can add multiple columns on each iteration
  - Perhaps you have a heuristic algorithm that can quickly compute multiple solutions

Column Generation Remarks (2)

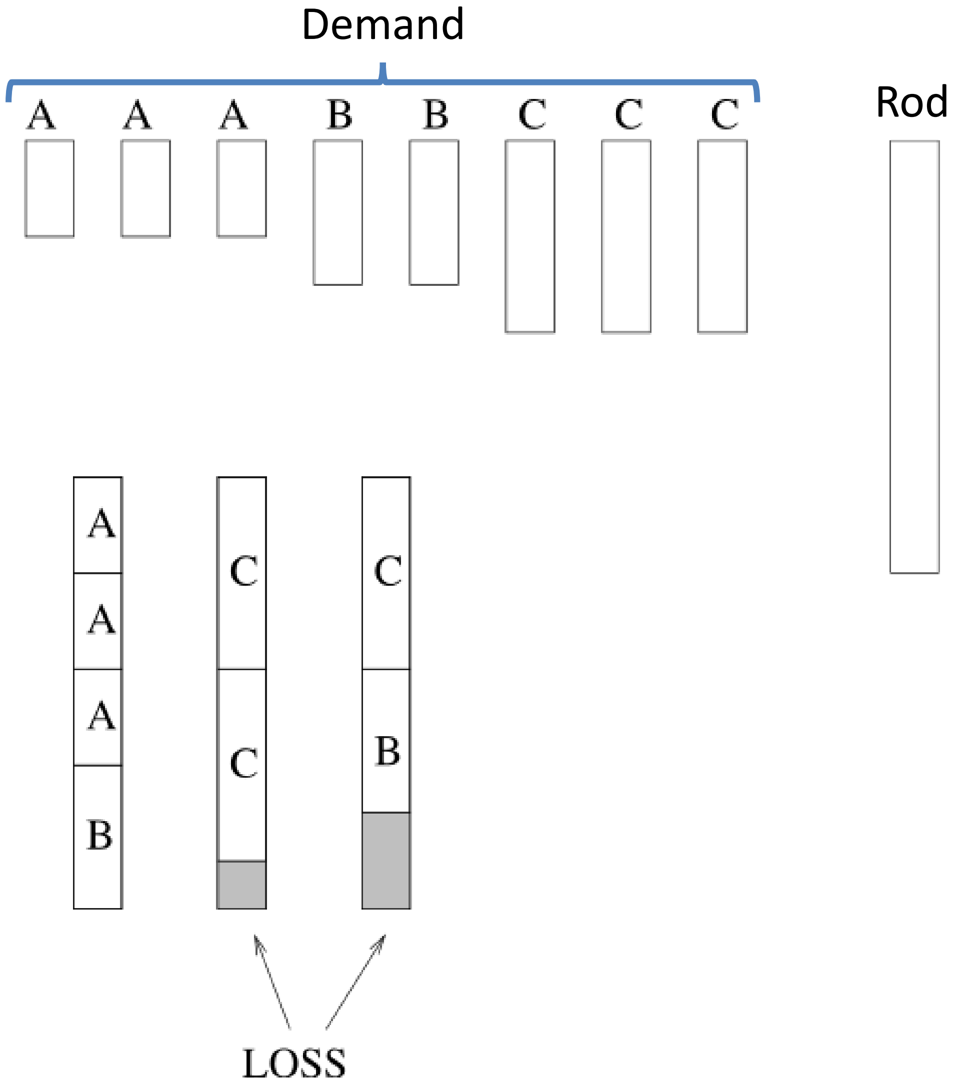
- There are several methods to find the initial columns
  - Penalty-based (pay a large cost for not solving the problem)
  - Farkas-cost
  - Problem-specific: can you think about one for the CSP problem?



We cannot exceed 14 units of time to reach the goal

Cutting Stock Problem

- We have several steel rods and we need to cut them to certain lengths: 22 cm, 45 cm, etc.
- You serve the customers' demands by cutting rods into the right sizes.
- You receive the rods in a particular length, e.g. 200 cm
- What is the minimum number of rods you need?
- How do you cut the customers' lengths while minimising the waste?



Cutting Stock: MIP 1

Given

- d_i = Demand for product *i of length* w_i
- W = length of base stock (rod)

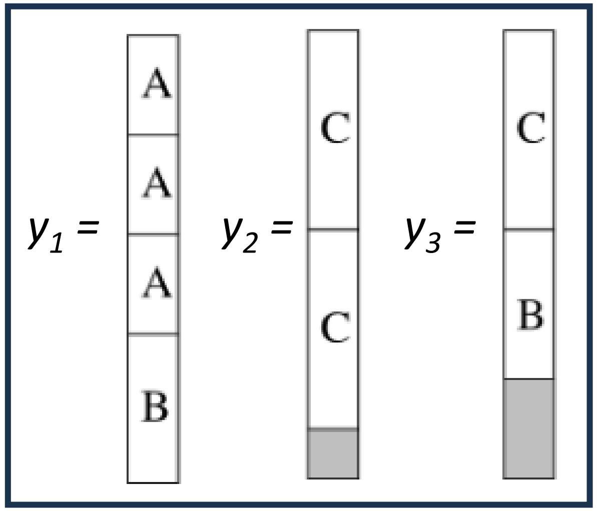
Decision Variables

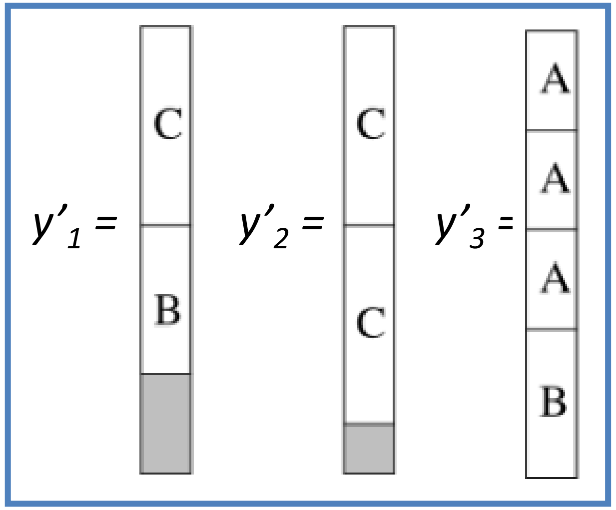
- y_k = 1 if we use rod *k*
  = 0 otherwise
- x_{ik} = Number of product *i* cut from rod *k*

$$
\begin{aligned}
\min \quad & \sum_k y_k \\
\text{s.t.} \quad & \sum_k x_{ik} = d_i && \forall i \\
& \sum_i w_i x_{ik} \le W y_k && \forall k \\
& x_{ik} \in \mathbb{N}_0 \\
& y_k \in \{0, 1\}
\end{aligned}
$$

Cutting Stock

- … but, this has a huge number of symmetric solutions
- I can interchange rods for a different solution:





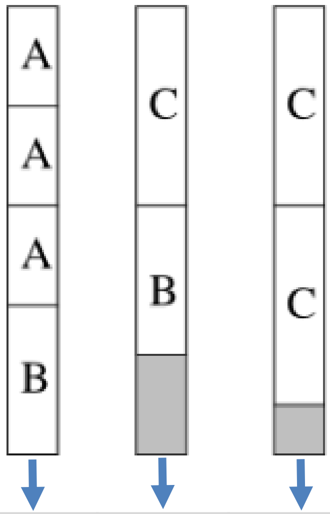
$$\begin{aligned}
\min \quad & \sum_k y_k \\
\text{s.t.} \quad & \sum_k x_{ik} = d_i && \forall i \\
& \sum_i w_i x_{ik} \leq W y_k && \forall k \\
& x_{ik} \in \mathbb{N}_0 \\
& y_k \in \{0, 1\}
\end{aligned}$$

Another way to think about it: *Patterns*

- Don't think about which rod for each item
- Think about the *patterns* for a rod

Cutting Stock: Reformulation (1)

- A *pattern* is a list of the products we will cut from a rod
  - E.g. 3 of product A and 1 of B, or
    1 of product C and 1 of B
- We ensure feasibility *a priori*
  - Don’t allow the pattern if the sum of lengths exceeds maximum



rows: offered lengths
columns: patterns
a_{ij}: number of items of length i produced by pattern j

A	3	0	0	2
B	1	1	0	0
C	0	1	2	0
D	0	0	0	2
E	0	0	0	1

Cutting Stock: Reformulation (2)

Let's say we can list *all possible* patterns

- Now our formulation is easy
- x_j is: how many times do we use pattern *j*?

A	3	0	0	2	…
B	1	1	0	0	…
C	0	1	2	0	…
D	0	0	0	2	…
E	0	0	0	1	…

make it easier by producing a little extra instead of having a pattern that produces a single length

$$\begin{aligned} \min \quad & \sum_j x_j \\ \text{s.t.} \quad & \sum_j a_{ij} x_j = d_i \quad \forall i \\ & x_j \in \mathbb{N}_0 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \min \quad & \sum_j x_j \\ \text{s.t.} \quad & \sum_j a_{ij} x_j \geq d_i \quad \forall i \\ & x_j \in \mathbb{N}_0 \end{aligned}$$

Cutting Stock: Pricing Problem (1)

- Recall the reduced cost is: $\mu = \tilde{c} - \tilde{A}^\intercal \lambda$
- For cutting stock problem, the reduced cost for a pattern j is: $\mu_j = 1 - \sum_i a_{ij} \lambda_i$

$$
\begin{aligned}
\min \quad & \sum_j x_j \\
\text{s.t.} \quad & \sum_j a_{ij} x_j \geq d_i \quad \forall i \quad (\lambda_i) \\
& x_j \in \mathbb{N}_0
\end{aligned}
$$

- We want to find a **feasible** pattern that minimises it:

$$
\begin{aligned}
\min_a 1 - \sum_i a_i \lambda_i \qquad & = 1 - \max_a \sum_i a_i \lambda_i \\
\text{s.t.} \sum_i w_i a_i \leq W \qquad & \text{s.t.} \sum_i w_i a_i \leq W \\
a_i \in \mathbb{Z}_{\geq 0} \qquad & a_i \in \mathbb{Z}_{\geq 0}
\end{aligned}
$$

Cutting Stock: Pricing Problem (2)

$$\max_a \sum_i a_i \lambda_i$$

$$\text{s.t.} \sum_i w_i a_i \leq W$$

$$a_i \in \mathbb{Z}_{\geq 0}$$

- This gives us a knapsack problem
  - Weight: the offered lengths w_i (pick lengths totalling up to W)
  - Value: the *reduced cost* derived from dual variables
- Yes, we have just replaced one $\mathcal{NP}$-hard problem with another, BUT
  - We only have one rod now, not *k* of them
  - Knapsack is an "easy" NP problem
    - The heuristic solution offers very good bounds on the optimal value

Cutting Stock: Column Generation

1. Start with some columns
   – **Cutting stock**: can you think about an easy set of patterns to start with?
2. Solve the **Restricted Master Problem**
   – **Cutting stock**: the problem with the patterns we have so far
3. Use the dual variables to define the **Pricing Problem**
   – **Cutting stock**: knapsack problem with values derived from the dual variables
4. Solve the pricing problem to obtain the column with **minimum reduced cost**
   – **Cutting stock**: solve the knapsack problem
5. If the reduced cost of the new column is negative, include it, and go to 2. Otherwise, we have the optimal solution to the relaxed (non-integer) problem
   – **Cutting stock**: if a pattern with negative reduced cost was found, add it to the RMP

Cutting Stock: Example

- Demand:
  - 44 pieces of length 81 cm
  - 3 pieces of length 70 cm
  - 48 pieces of length 68 cm
- Supply:
  - We have steel rods of length 218 cm of unit cost

	Len	d_i
A	81	44
B	70	3
C	68	48
Rods	218	

Cutting Stock: Initial Patterns

	Len	d_i
A	81	44
B	70	3
C	68	48
Rod	218	

1. Start with some columns

- Make a guess at good columns
- Make something up
  - Must form a “basis”
  - (i.e. we have to be able to create any other column by combining our initial columns)

- We’ll use a simple basis

1	0	0
0	1	0
0	0	1

Cutting Stock: First RMP

	Len	d_i
A	81	44
B	70	3
C	68	48
Rod	218	

Solve the RMP

$$\min x_1 + x_2 + x_3$$

$$\begin{array}{llll} 1\,x_1 & & & \geq 44 \\ & 1\,x_2 & & \geq 3 \\ & & 1\,x_3 & \geq 48 \end{array}$$

- This gives us $\lambda = (1, 1, 1)$

Cutting Stock: First Pricing Problem

	Len	d_i
A	81	44
B	70	3
C	68	48
Rod	218	

Solve the pricing problem, i.e., knapsack problem for $\lambda = (1, 1, 1)$

$$
\begin{aligned}
\max \quad & 1\,a_A + 1\,a_B + 1\,a_C \\
\text{s.t.} \quad & 81\,a_A + 70\,a_B + 68\,a_C \leq 218 \\
& a_A \leq 2 \\
& a_B \leq 3 \\
& a_C \leq 3 \\
& a_A, a_B, a_C \in \mathbb{N}
\end{aligned}
$$

Simple bounds: 218 / 81 = 2.7, so no more than 2 of A can fit on a rod

Gives $(a_A, a_B, a_C) = (0,0,3)$

Cutting Stock: Second RMP

Solve RMP with the new column (a_A, a_B, a_C) = (0,0,3)

$$
\begin{array}{llllll}
\min & x_1 + x_2 + x_3 + x_4 & & & & \\
 & 1\ x_1 & & & & \geq 44 \\
 & & 1\ x_2 & & & \geq 3 \\
 & & & 1\ x_3 & + 3\ x_4 & \geq 48
\end{array}
$$

	Len	d_i
A	81	44
B	70	3
C	68	48
Rod	218	

- This gives us λ = (1.0, 1.0, 0.33)

Cutting Stock: Second Pricing Problem

Solve the pricing problem, i.e.,
knapsack problem for λ = (1. 1. 0.33)

$$\begin{aligned} \max \quad & 1\, a_A + 1\, a_B + 0.33\, a_C \\ \text{s.t.} \quad & 81\, a_A + 70\, a_B + 68\, a_C \leq 218 \\ & a_A \leq 2 \\ & a_B \leq 3 \\ & a_C \leq 3 \\ & a_A, a_B, a_C \in \mathbb{N} \end{aligned}$$

	Len	d_i
A	81	44
B	70	3
C	68	48
Rod	218	

Gives $(a_A, a_B, a_C) = (0,3,0)$

Cutting Stock: Third RMP

Solve RMP with the new column $(a_A, a_B, a_C) = (0,3,0)$

	Len	d_i
A	81	44
B	70	3
C	68	48
Rod	218	

$$\min\ x_1 + x_2 + x_3 + x_4 + x_5$$

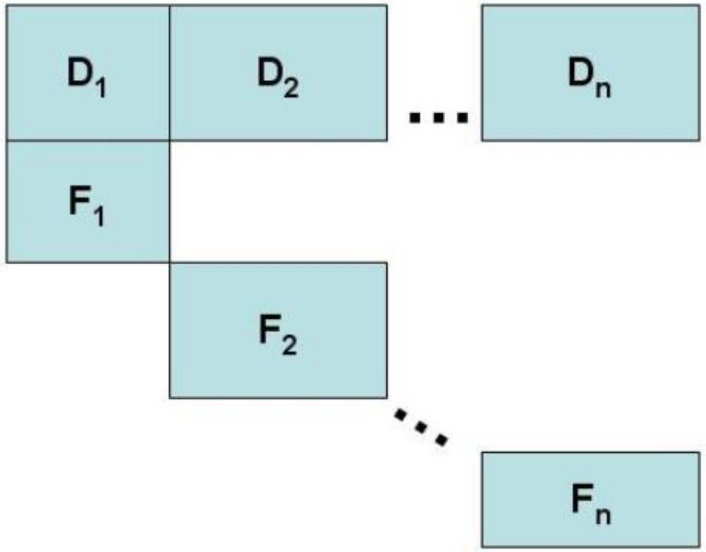
$$\begin{array}{lllllll}
1\ x_1 & & & & & \geq 44 \\
 & 1\ x_2 & & & + 3\ x_5 & \geq 3 \\
 & & 1\ x_3 & + 3\ x_4 & & \geq 48
\end{array}$$

… and so on until we find the optimal solution to the linear relaxation

Then we do Branch and Bound on the fractional usage of patterns

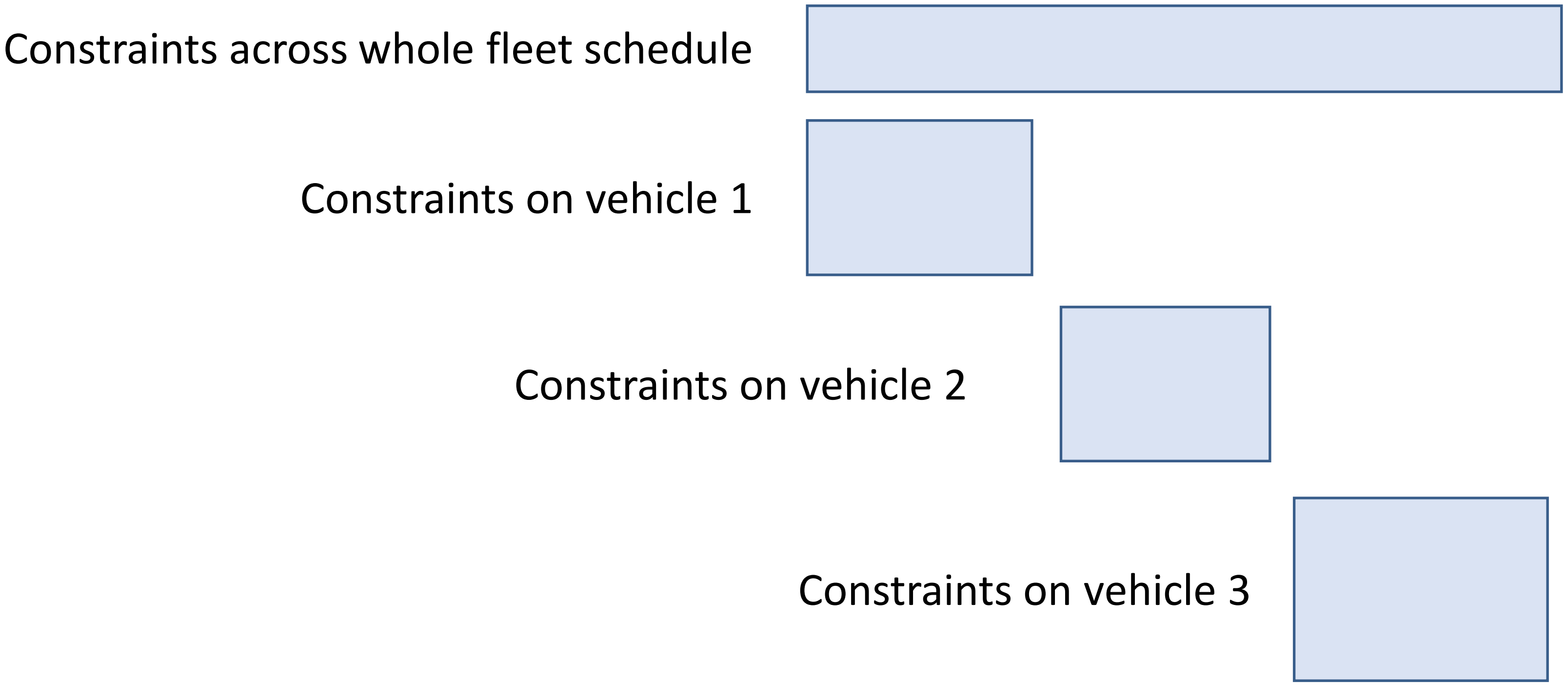
Dantzig-Wolfe Decomposition

- Dantzig-Wolfe Decomposition is **Column Generation on steroids**,
  - For problems with a block structure
  - Column Generation is done for each block
  - Solutions are brought together in a Master
  - New constraints on the subproblems are generated
  - Back to CG



Dantzig-Wolfe Decomposition

- Column Generation helps with complicating constraints
- Dantzig-Wolfe takes that to the next level
- E.g. scheduling problem



Attributions

- Dantzig-Wolfe decomposition image from Wikipedia
- Constrained Shortest Path example and images based on the “A Primer in Column Generation” by Jacques Desrosiers and Marco E. Lubbecke and it respective slides
- Cutting Stock worked example and images adapted from Phil Kilby and Thomas Stidsen