# Column generation and IP
# From textbook to practice - Part II

Ricardo Fukasawa

Department of Combinatorics & Optimization
University of Waterloo

MIP Workshop 2025
June 2, 2025

UNIVERSITY OF WATERLOO
FACULTY OF MATHEMATICS
Department of Combinatorics
and Optimization

# Goal of this part

- Provide several ideas (mostly on higher level) of what can be done on top of what we have seen
- Not many details provided for some ideas
- Provide takeaways and references for all

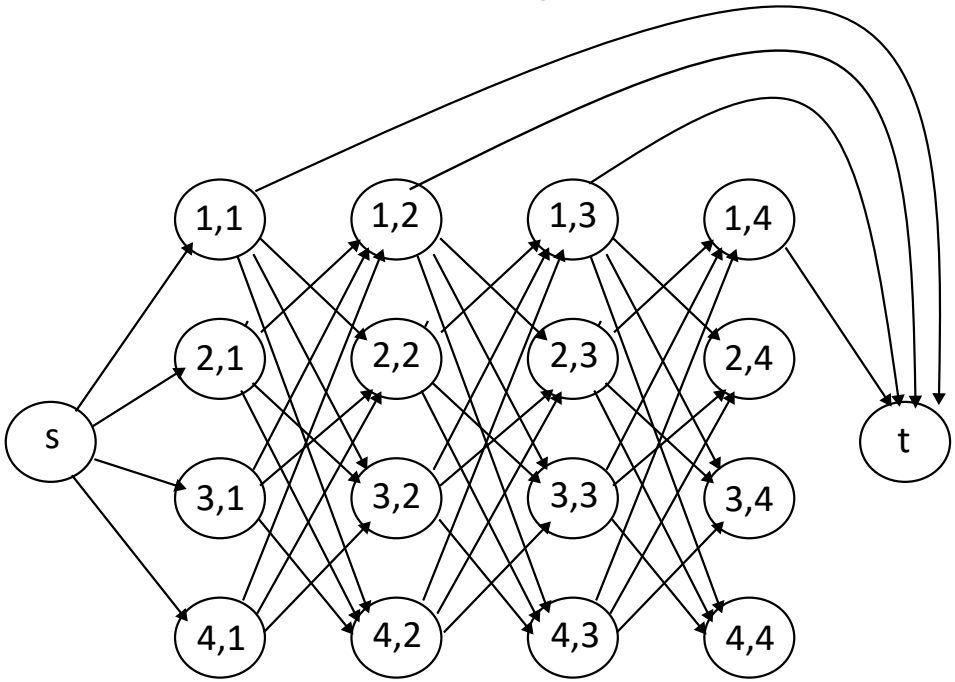
# Color code


- Textbook
- Issues that are somewhat standard
- Research issues

# Shortest path with resource constraints

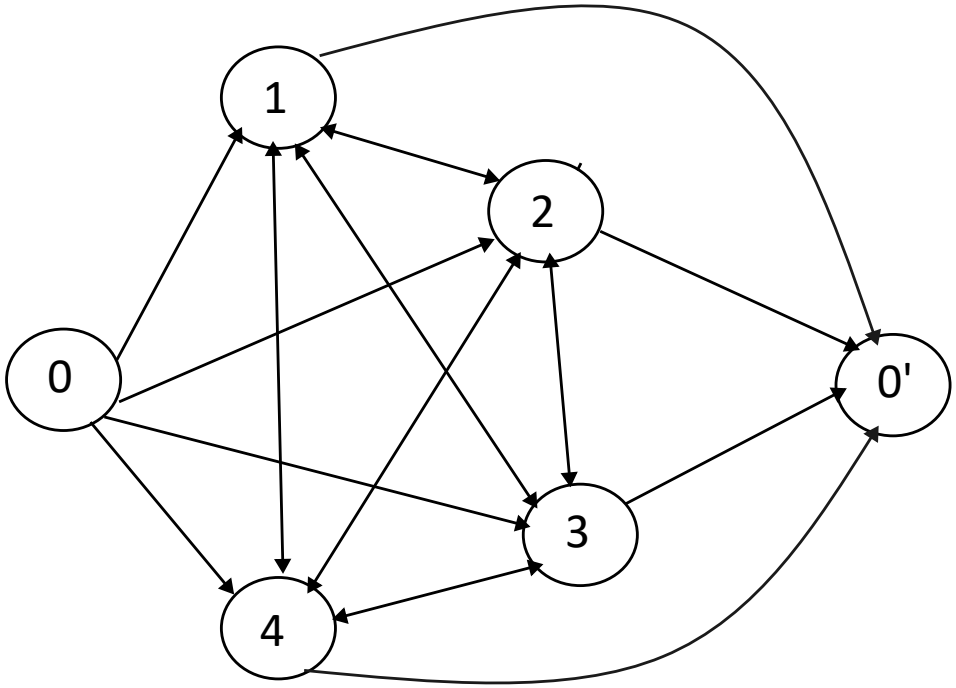
# Shortest path with resource constraints

For the CVRP, the pricing problem is to find a shortest path in:



# Shortest path with resource constraints

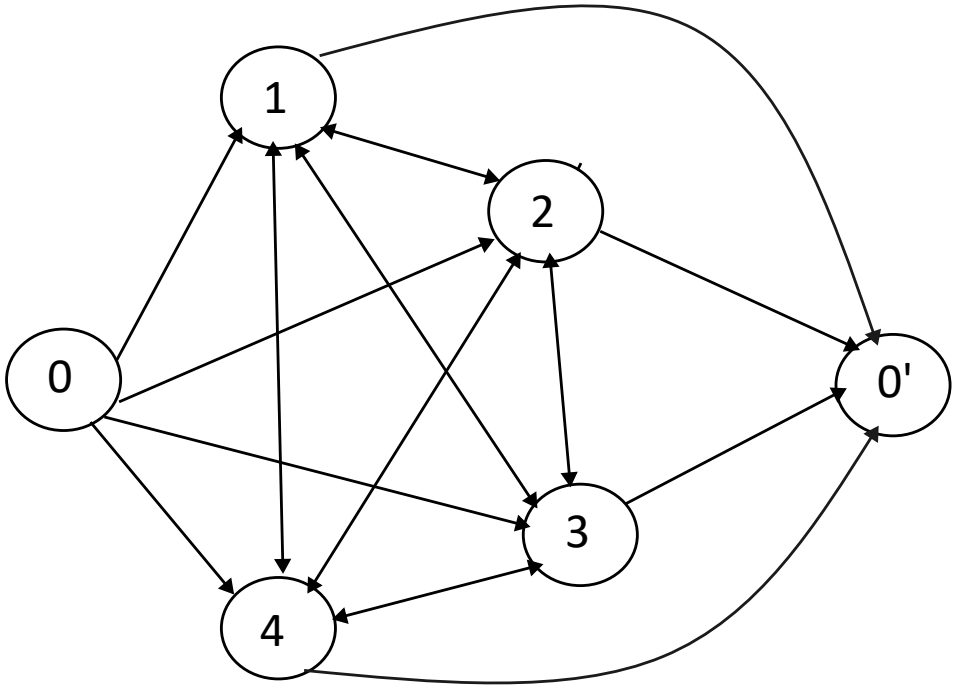
Different perspective. Consider the original graph (with dummy destination depot $0'$):




- Define a resource called "load"

# Shortest path with resource constraints

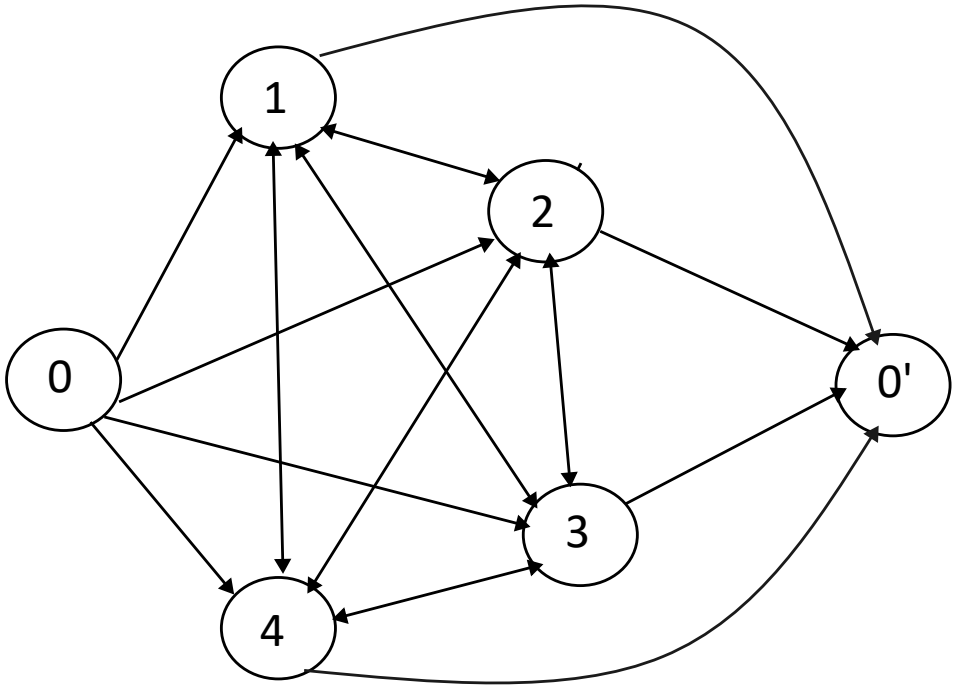
Different perspective. Consider the original graph (with dummy destination depot $0'$):




- Define a resource called "load"
- Everytime we visit a client $i$, the resource load increases by $d_i$.

# Shortest path with resource constraints

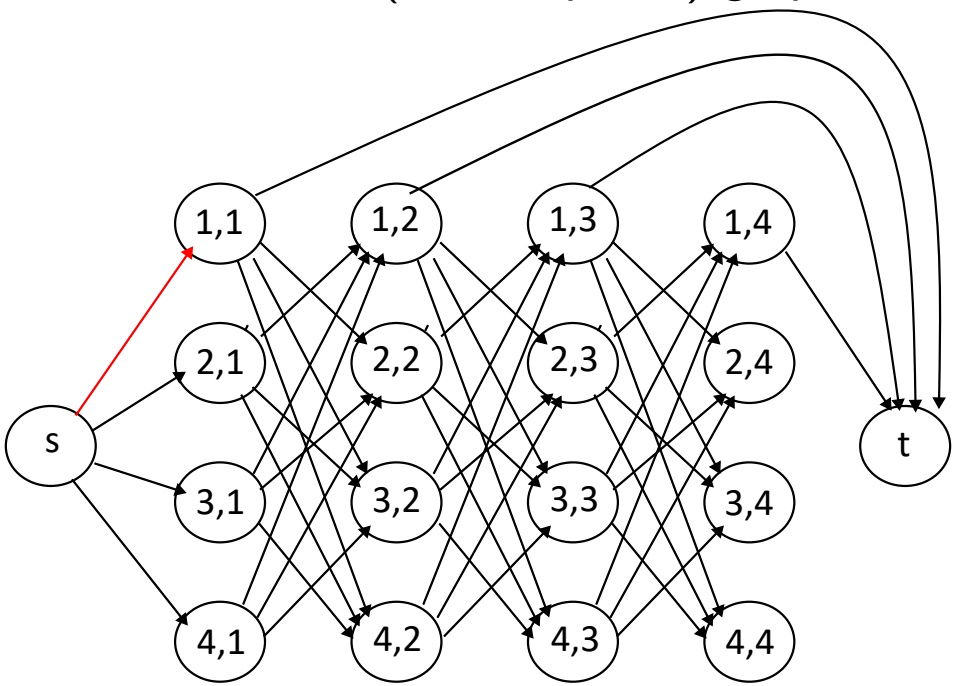
Different perspective. Consider the original graph (with dummy destination depot $0'$):



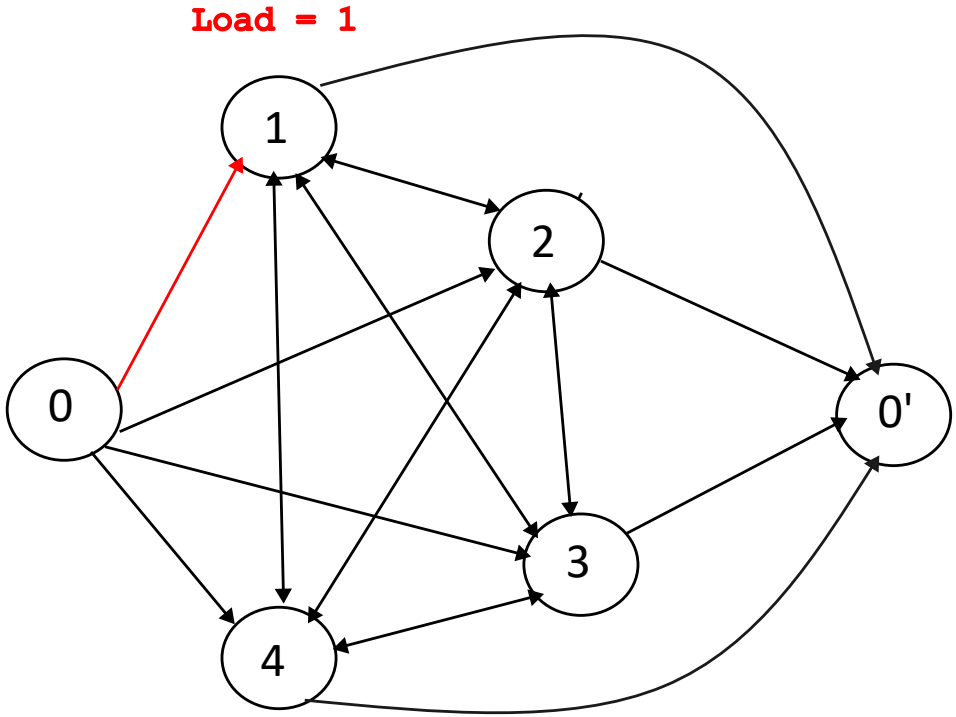

- Define a resource called "load"
- Everytime we visit a client $i$, the resource load increases by $d_i$.
- Find minimum cost $0 - 0'$ walk such that total load is $\leq C$

# Shortest path with resource constraints

Path in extended ("state-space") graph

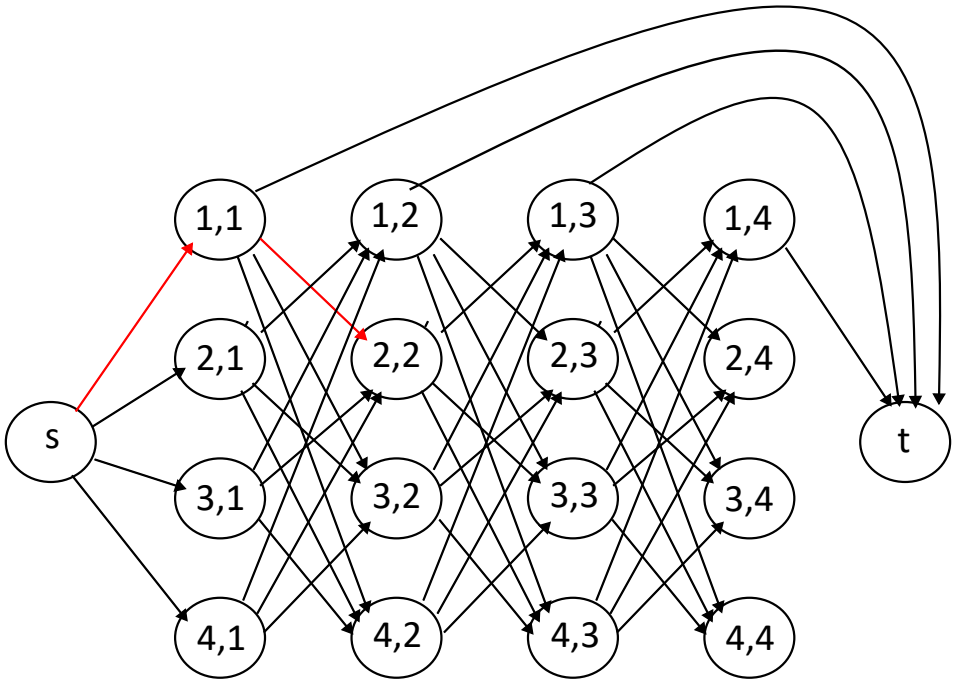



"Path" with resource constraints in original graph (actually a walk)

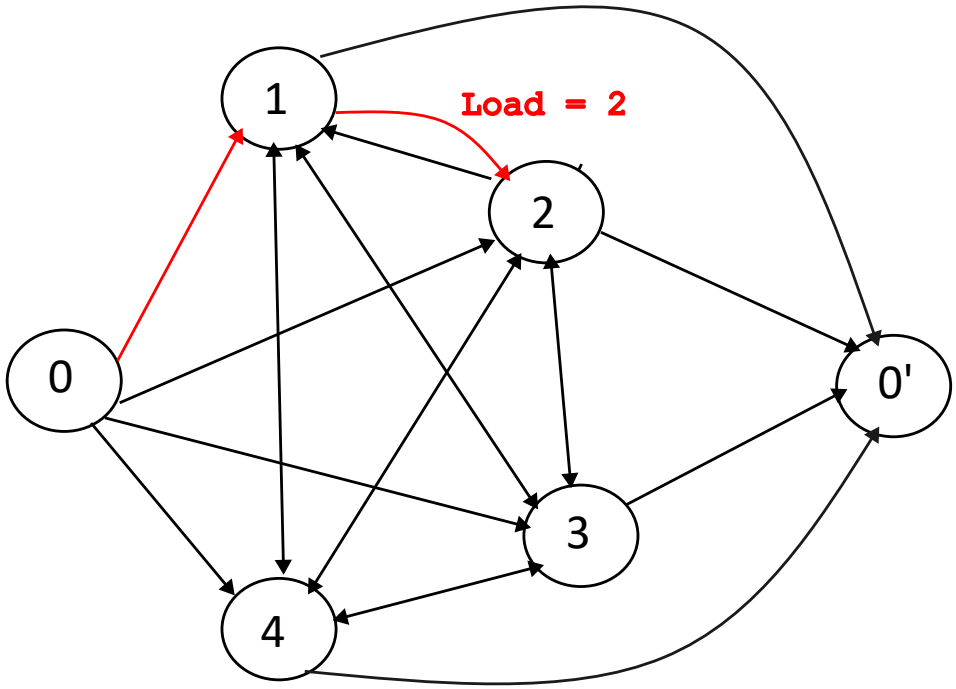


# Shortest path with resource constraints

Path in extended ("state-space") graph

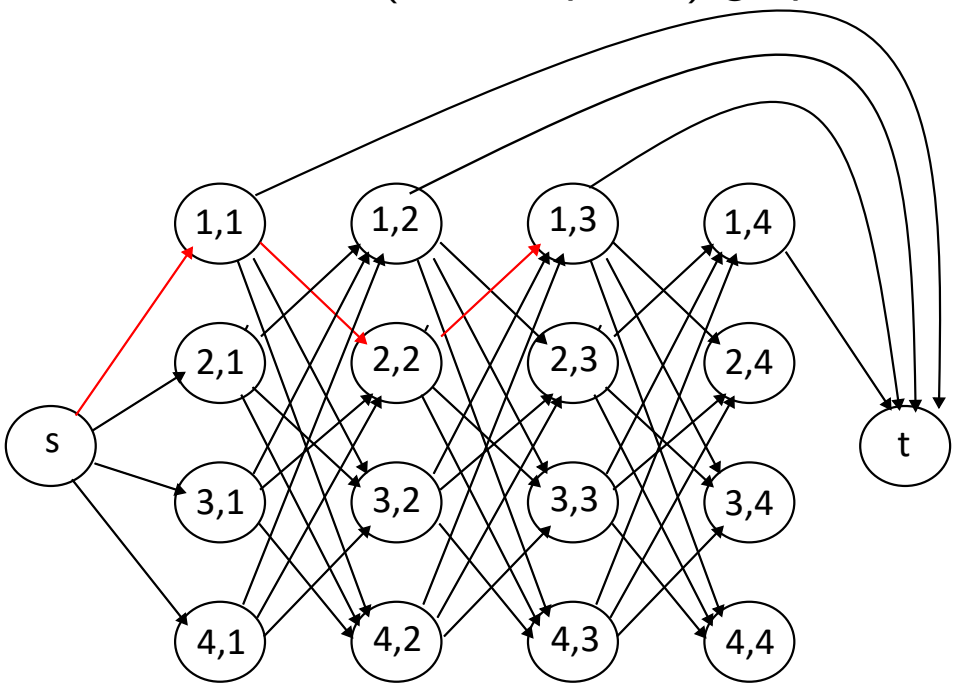



"Path" with resource constraints in original graph (actually a walk)

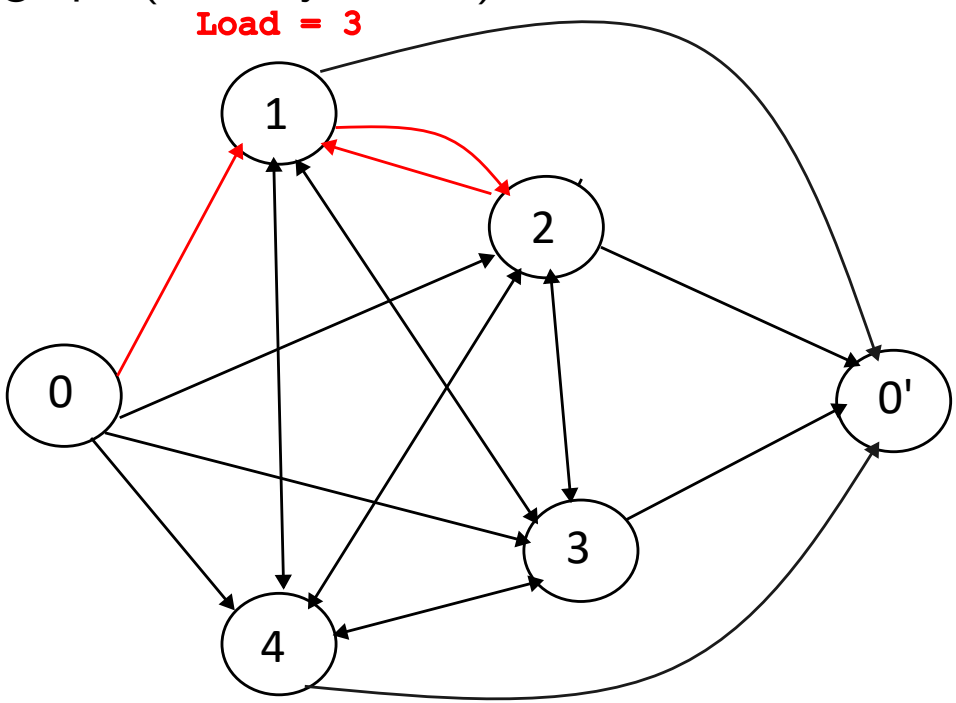


# Shortest path with resource constraints

Path in extended ("state-space") graph

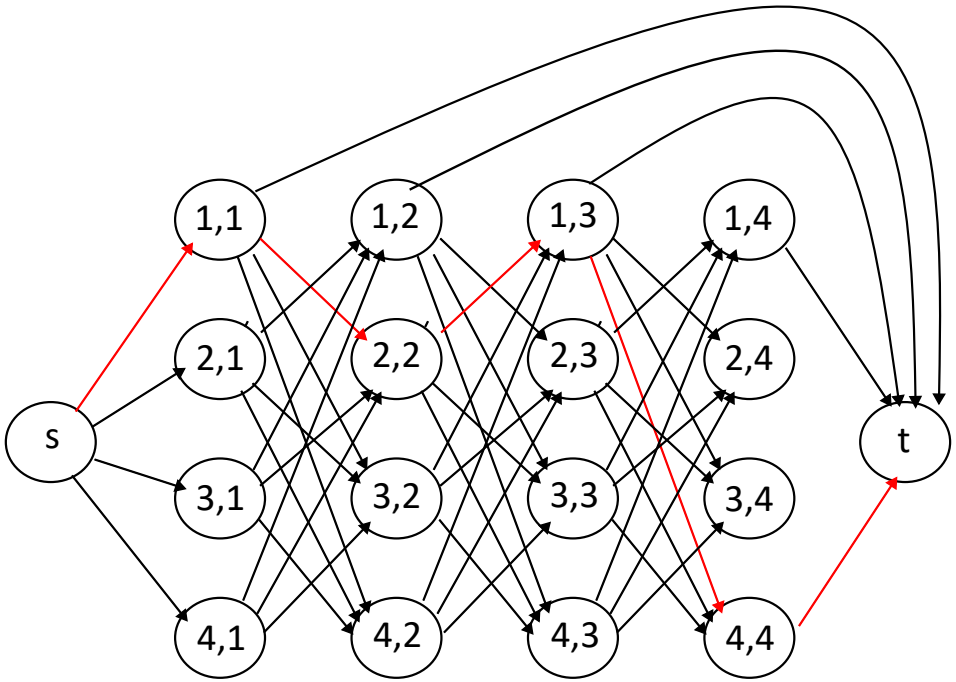



"Path" with resource constraints in original graph (actually a walk)

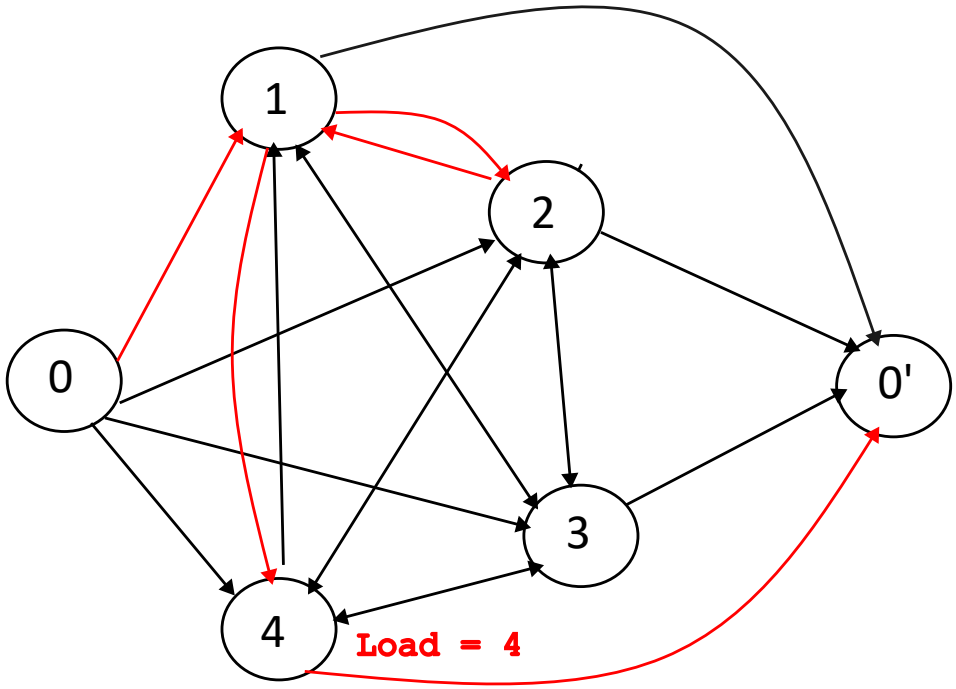


# Shortest path with resource constraints

Path in extended ("state-space") graph



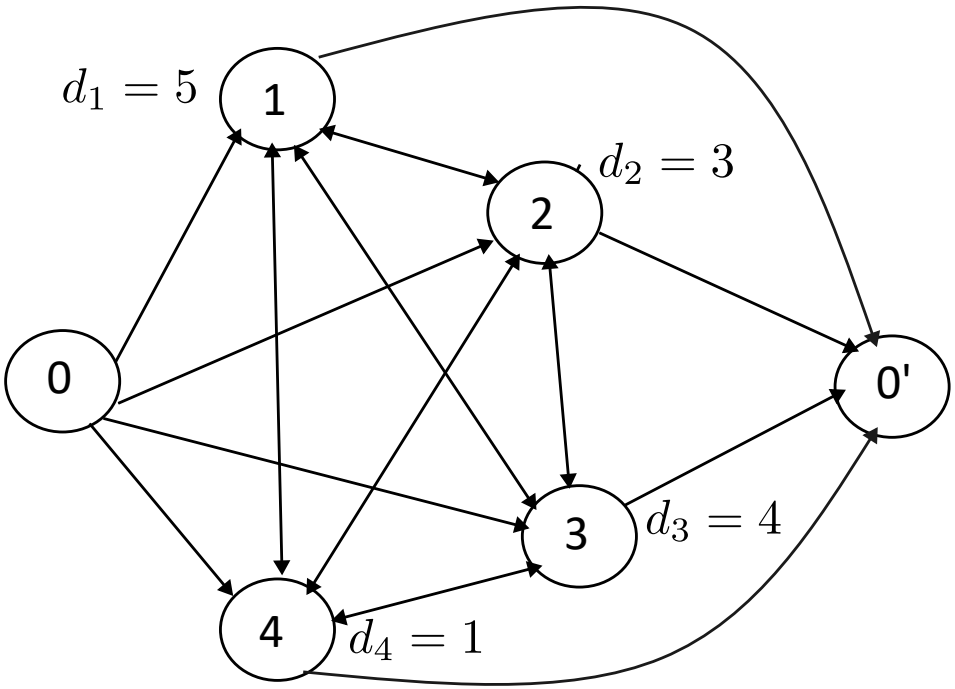

"Path" with resource constraints in original graph (actually a walk)



# Shortest path with resource constraints (SPPRC)

Why does it matter?

- Lima et al. (2022): Several applications with pseudopolynomial arc-flow formulations, linked to DW over them
- 
- (SPPRC) can be solved more efficiently

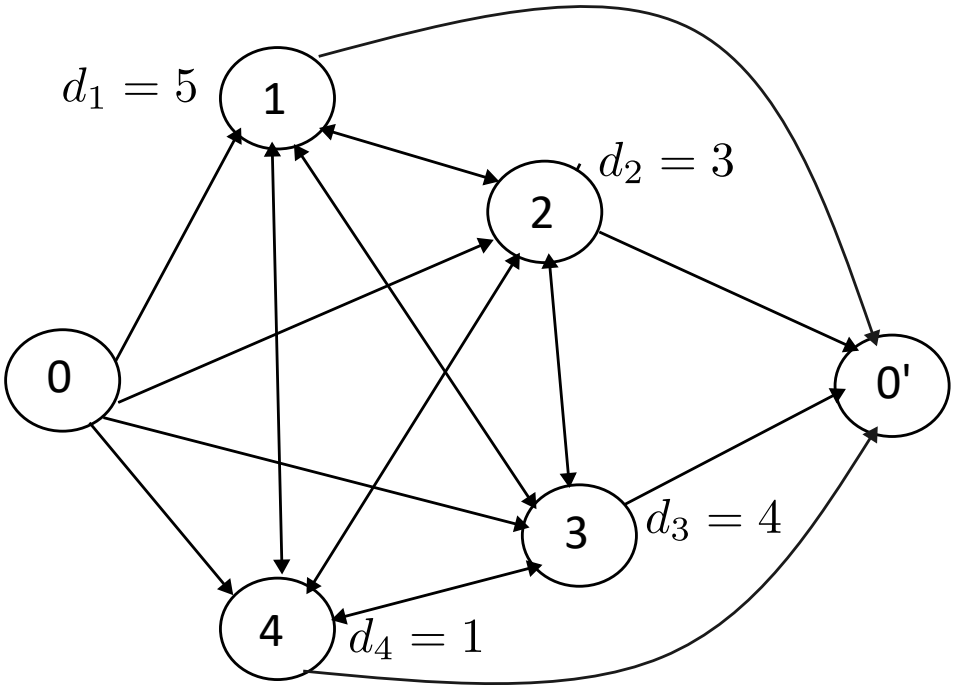


# Shortest path with resource constraints (SPPRC)

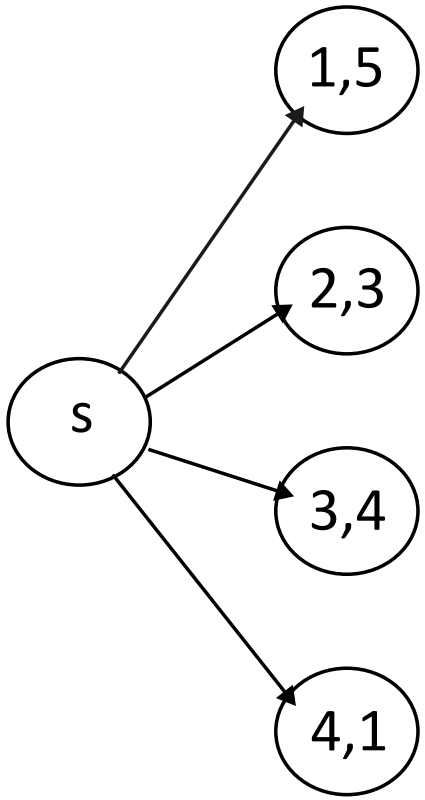
Why does it matter?

- Lima et al. (2022): Several applications with pseudopolynomial arc-flow formulations, linked to DW over them
- 
- (SPPRC) can be solved more efficiently
  - States are generated only as needed

Original graph (with demands)




Path in extended ("state-space") graph

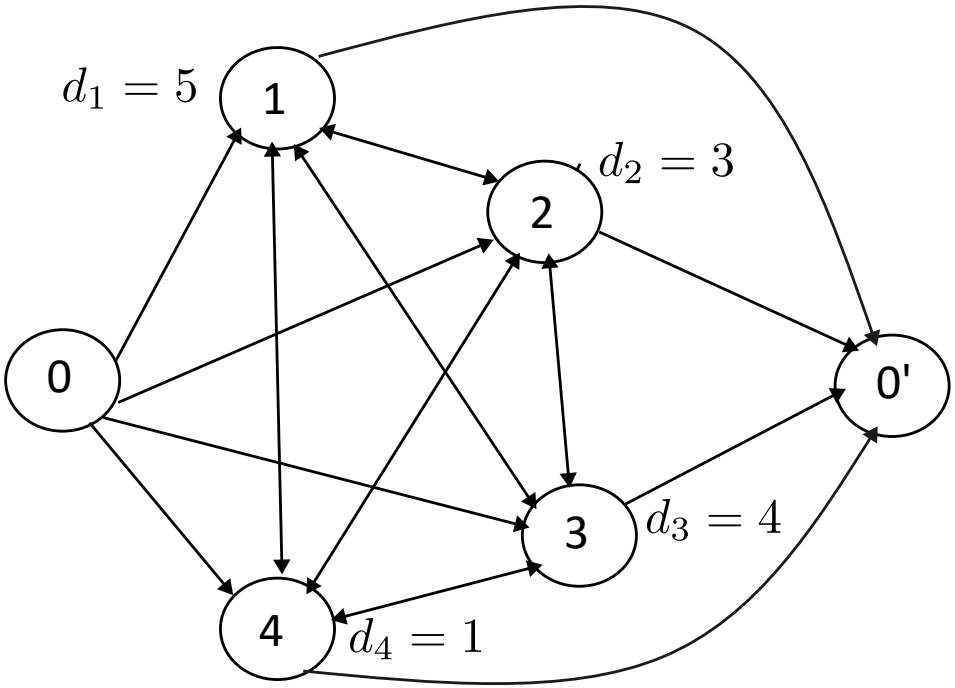


# Shortest path with resource constraints (SPPRC)

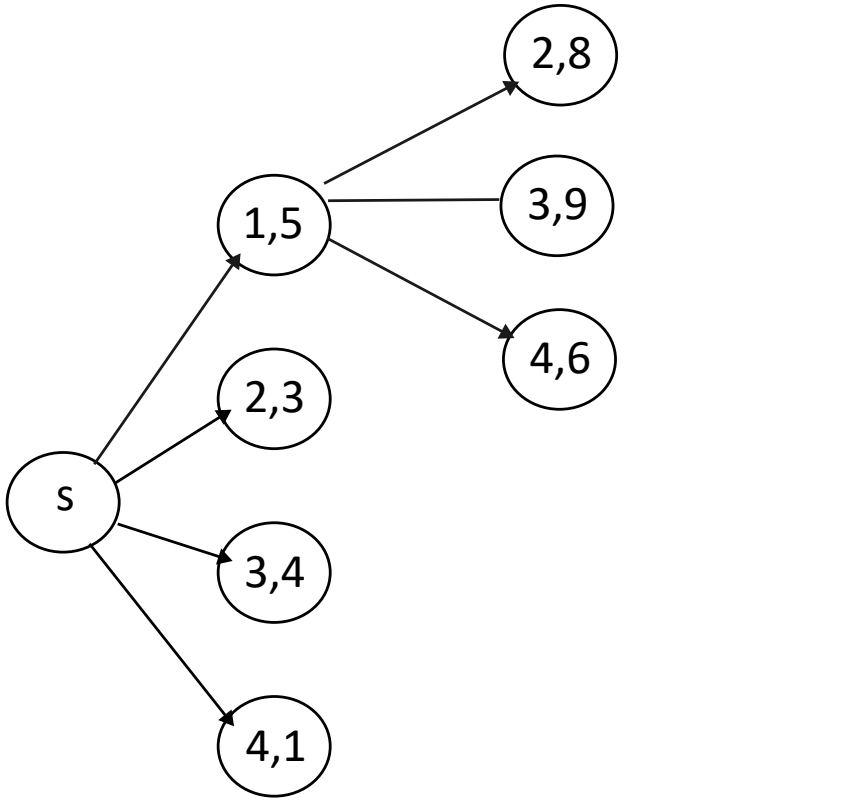
Why does it matter?

- Lima et al. (2022): Several applications with pseudopolynomial arc-flow formulations, linked to DW over them
- 
- (SPPRC) can be solved more efficiently
  - States are generated only as needed

Original graph (with demands)




Path in extended ("state-space") graph

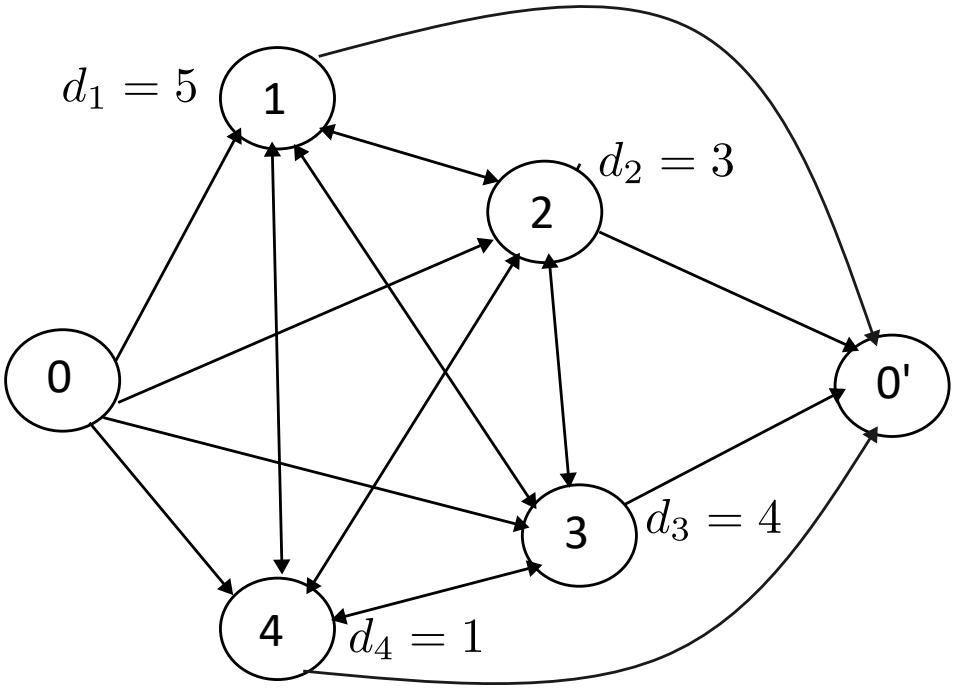


# Shortest path with resource constraints (SPPRC)

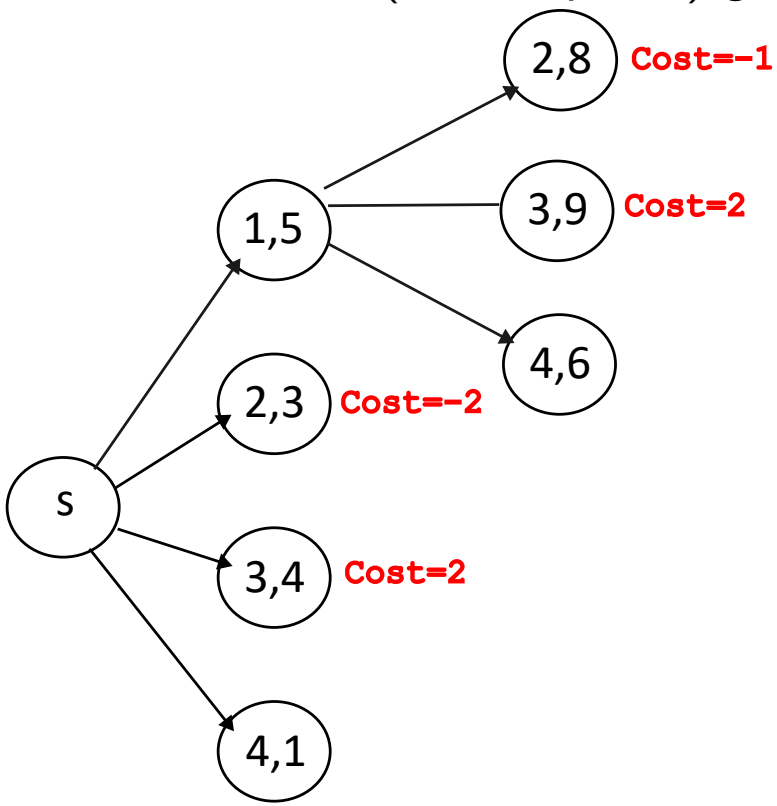
Why does it matter?

- Lima et al. (2022): Several applications with pseudopolynomial arc-flow formulations, linked to DW over them
- 
- (SPPRC) can be solved more efficiently
  - States are generated only as needed
  - Dominated states are eliminated (leads to so-called "domination rules")

Original graph (with demands)




Path in extended ("state-space") graph

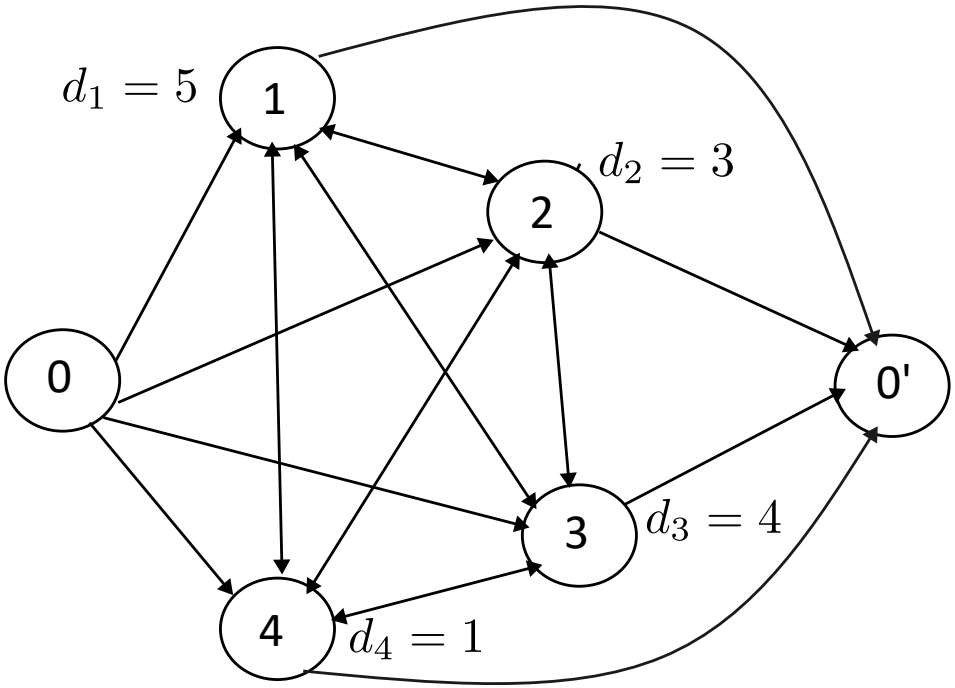


# Shortest path with resource constraints (SPPRC)

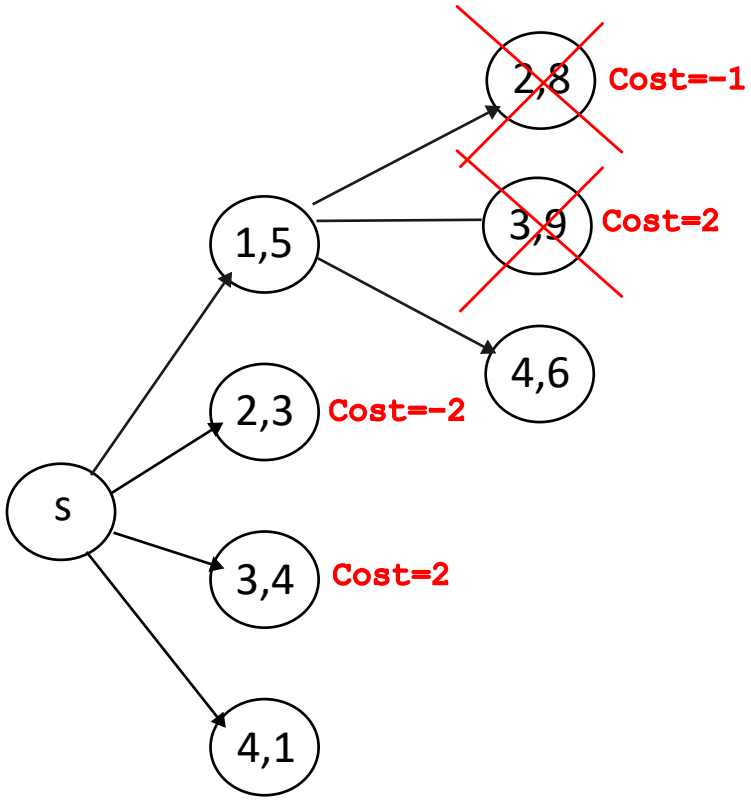
Why does it matter?

- Lima et al. (2022): Several applications with pseudopolynomial arc-flow formulations, linked to DW over them
-
- (SPPRC) can be solved more efficiently
  - States are generated only as needed
  - Dominated states are eliminated (leads to so-called "domination rules")

Original graph (with demands)




Path in extended ("state-space") graph



## Takeaway

Several successful DW/CG based formulations based on DP.

## Takeaway

Several successful DW/CG based formulations based on DP.

## Takeaway 2

Solving DP with SPPRC is more efficient.

# Non-robust cuts (?)

# Non-robust cuts (?)

$$\begin{array}{lll} \min & \sum_{r\in\mathcal{R}} c_r \cdot \lambda_r & \\ \text{s.t.} & \sum_{r\in\mathcal{R}} \textsc{count}(v,r) \cdot \lambda_r = 1, & \forall v \in V_+, \\ & \sum_{r\in\mathcal{R}} \lambda_r = K, & \\ & \lambda_r \in \{0,1\}, & \forall r \in \mathcal{R}. \end{array} \tag{SP}$$

Non-robust cuts:

- Directly on the $\lambda$ variables (not translated to original space)
- Dates back to Nemhauser and Park (1991) - for a different problem and not with this name
- Example: Subset-row cuts (Jepsen et al., 2008): Pick $S \subseteq V$, and add degree constraints with certain multipliers:

$$\sum_{r\in\mathcal{R}} \sum_{v\in S} \frac{\textsc{count}(v,r)}{2} \cdot \lambda_r = \frac{|S|}{2}$$

- Apply CG

$$\sum_{r\in\mathcal{R}} \left\lfloor \sum_{v\in S} \frac{\textsc{count}(v,r)}{2} \right\rfloor \cdot \lambda_r \leq \left\lfloor \frac{|S|}{2} \right\rfloor$$

# Non-robust cuts (?)

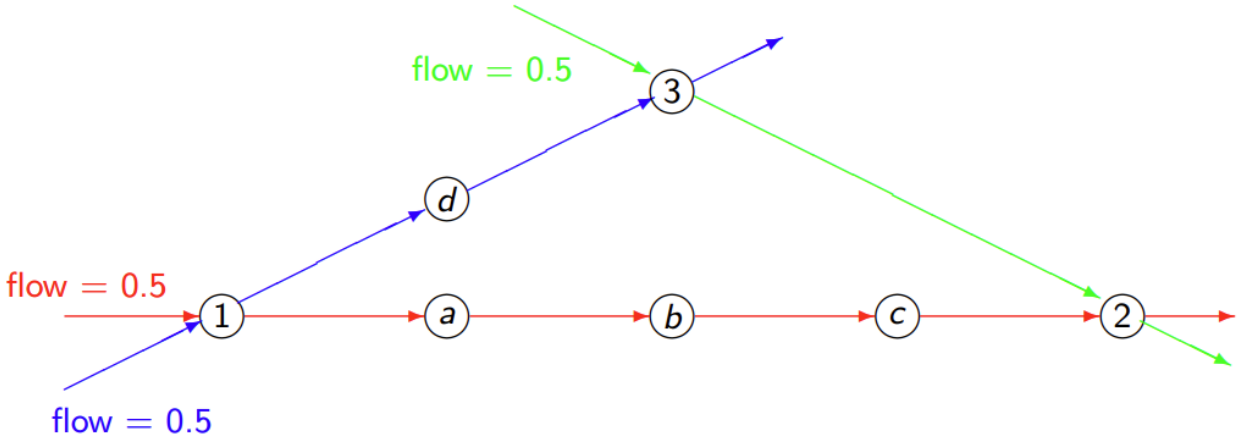
Subset-row cuts (SRC)

- Must keep track of the number of visited customers in $S$, for every $S$ used in a SRC
- Becomes expensive, though gives strong bounds

Limited-memory SRC (Pecin et al., 2017).

- Consider $|S| = 3$
- Weaken the cut
- $\lambda_r$ receives coefficient 1 if route $r$ visits at least two customers $i_1$ and $i_2$ in $S$ AND nodes between $i_1$ and $i_2$ belong to a pre-specified subset of customers (the memory)
- All other $\lambda_r$ have a zero coefficient

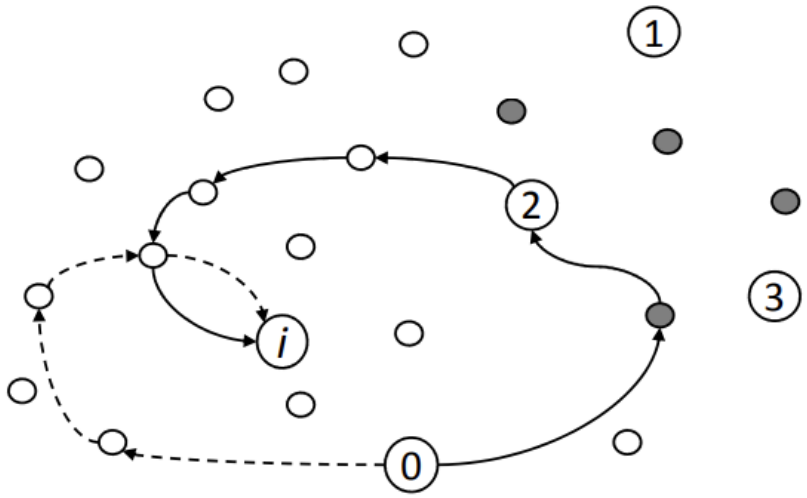
# Non-robust cuts (?)




Set $S = \{1, 2, 3\}$
Memory: $\{a, b, c, d\}$

(Figures taken from Pecin's presentation at ColGen2016 workshop
`https://www.gerad.ca/colloques/ColumnGeneration2016/PDF/Pecin.pdf`)

# Non-robust cuts (?)




Set $S = \{1, 2, 3\}$
Memory: $\{a, b, c, d\}$

(Figures taken from Pecin's presentation at ColGen2016 workshop
`https://www.gerad.ca/colloques/ColumnGeneration2016/PDF/Pecin.pdf`)

# Non-robust cuts (?)

| | Gap(%) |
|---|---|
| Only CG (elementary routes) | 2.63 |
| + robust cuts | 0.98 |
| + 3SRCs | 0.35 |
| + 4SRCs + 5SRCs | 0.24 |
| Rank 1 Cuts up to 5 rows | 0.17 |

(Table taken from Pecin's presentation at ColGen2016 workshop
`https://www.gerad.ca/colloques/ColumnGeneration2016/PDF/Pecin.pdf`)

## Takeaway

Non-robust cuts are good, but must be used with caution.

**Takeaway**

Non-robust cuts are good, but must be used with caution.

**Takeaway 2**

Sometimes it is worth to use weaker formulations, if they are more efficient.

# Dealing with nonlinearity

## Nonlinear DW (Ceselli, Létocart, and Traversi, 2022)

$$\begin{array}{ll} \min & x^T Q x + c^T x \\ \text{s.t.} & Dx \geq f \\ & Gx \geq h \\ & x \in \{0,1\}^n \end{array} \qquad \text{(BQP)}$$

# Nonlinear DW (Ceselli, Létocart, and Traversi, 2022)

$$\begin{array}{ll} \min & x^T Q x + c^T x \\ \text{s.t.} & Dx \geq f \\ & Gx \geq h \\ & x \in \{0,1\}^n \end{array} \qquad \text{(BQP)}$$

$$\begin{array}{lll} \min & x^T Q x + c^T x & \\ \text{s.t.} & Dx \geq f & \\ & x & -\sum_{i \in \mathcal{R}} v^i \lambda_i = 0 \\ & & \sum_{i \in \mathcal{R}} \lambda_i = 1 \\ & x \in [0,1]^n & \\ & & \lambda_i \in [0,1] \end{array} \qquad \text{(Q-QM)}$$

$$\begin{array}{ll} \min & \sum_{i \in \mathcal{R}} c^i \lambda_i \\ \text{s.t.} & \sum_{i \in \mathcal{R}} D v^i \lambda_i \geq f \\ & \sum_{i \in \mathcal{R}} \lambda_i = 1 \\ & \lambda_i \in [0,1] \end{array} \qquad \text{(Q-QP)}$$

# Nonlinear DW (Ceselli, Létocart, and Traversi, 2022)

$$\begin{array}{ll} \min & x^T Q x + c^T x \\ \text{s.t.} & Dx \geq f \\ & Gx \geq h \\ & x \in \{0,1\}^n \end{array} \qquad \text{(BQP)}$$

$$\begin{array}{ll} \min & x^T Q x + c^T x \\ \text{s.t.} & Dx \geq f \\ & x \\ & \\ & \\ & x \in [0,1]^n \end{array} \qquad \begin{array}{l} \\ \\ -\sum_{i \in \mathcal{R}} v^i \lambda_i = 0 \\ \sum_{i \in \mathcal{R}} \lambda_i = 1 \\ \\ \lambda_i \in [0,1] \end{array} \qquad \text{(Q-QM)}$$

$$\begin{array}{ll} \min & \sum_{i \in \mathcal{R}} c^i \lambda_i \\ \text{s.t.} & \sum_{i \in \mathcal{R}} D v^i \lambda_i \geq f \\ & \sum_{i \in \mathcal{R}} \lambda_i = 1 \\ & \lambda_i \in [0,1] \end{array} \qquad \text{(Q-QP)}$$

- Ceselli, Létocart, and Traversi (2022): No dominance between the two relaxations
- Ceselli, Létocart, and Traversi (2022): Study (theoretical and empirical) strength of several variants

# Nonlinear DW (Ceselli, Létocart, and Traversi, 2022)

Variants considered:

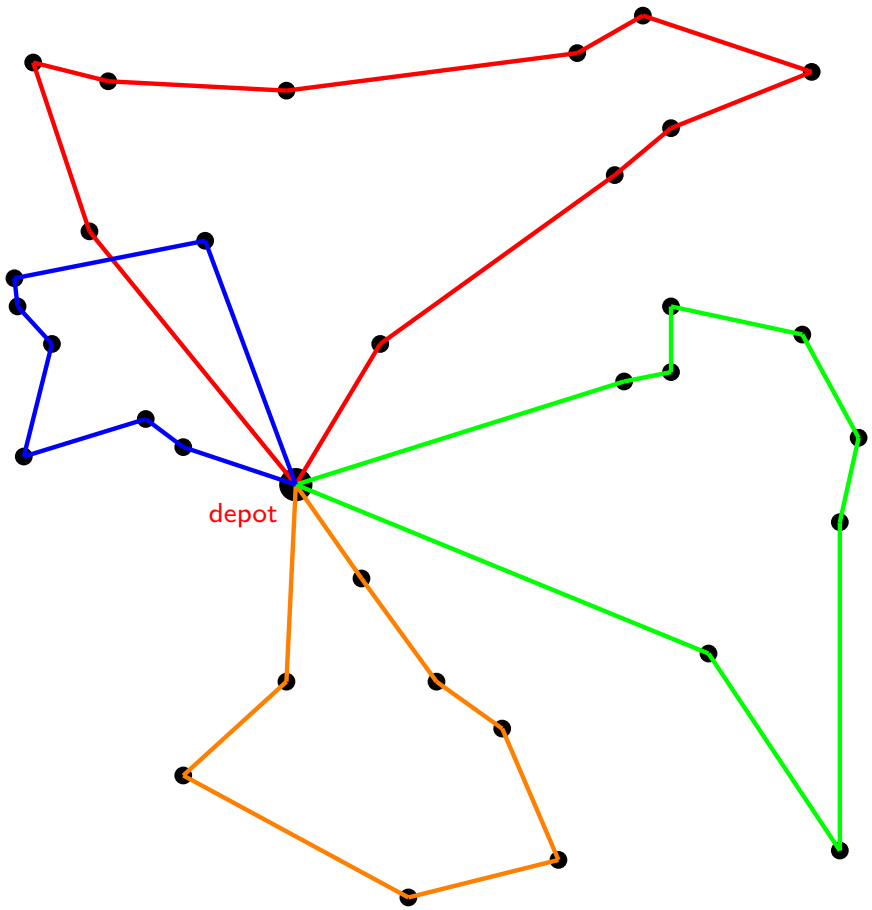
- (Q-QM), (Q-QP)
- Corresponding versions with addition of Lagrangian terms in the objective for:
  - $(x_i^2 - x_i)$
  - Implicit equalities
  - $(z_{ij} - x_i x_j)$ (after adding extra binary variables $z_{ij}$)
- A variant of quadratic pricing provides strongest bounds (nearly 100% gap closed) in a reasonable time for cardinality constrained quadratic knapsack.
- Variants of quadratic pricing provides strongest theoretical bounds.

## Takeaway

CG with NLP $\rightarrow$ opens many more questions/challenges/opportunities

# Nonlinear DW (F. et al., 2018)

Recall CVRP:




- $G = (V, E)$
- $V = \{0\} \cup V_+$
- Edge lengths $\ell_e$, $e \in E$
- $K$ vehicles, capacity $C$
- Client demands $d_i, \forall i \in V_+$.
- Let $S_j$ be the set of clients served by route $j$.
  Then $d(S_j) := \sum_{u \in S_j} d_u \leq C$
- Goal: Find minimum cost set of $K$ routes that start/end at depot, serves all customers and respect capacity constraint

# Nonlinear DW (F. et al., 2018)

Problem studied:

- Same data as CVRP (directed version)
- Each customer has a time window $[a_v, b_v]$ when it must be visited
- Early arrival is allowed (must wait until beginning of time window)
- Decision variable: average speed $v_{ij} \in [l, u]$ under which to travel $ij$
- $t_{ij} = \frac{\ell_{ij}}{v_{ij}}$
- Objective: $\min \sum_{ij} (\ell_{ij} + f(v_{ij}))$, where $f$ is a strictly convex function
- Motivation: Minimize pollution, calculated as function of speed

## Nonlinear DW (F. et al., 2018)

Key result:

**Theorem (F. et al., 2018)**

*If time window is not hit at customer $i$, then optimal speed for arc entering and leaving $i$ is the same.*

# Nonlinear DW (F. et al., 2018)

Key result:

**Theorem (F. et al., 2018)**

*If time window is not hit at customer $i$, then optimal speed for arc entering and leaving $i$ is the same.*

Pricing

- $q$-routes: Walks that start/end at depot and satisfy capacity
- Pricing: Finding minimum cost $q$-route
- Shortest path in a "state-space" graph
- States are $(v, \delta, t, ltw)$.
  - $v$: Last visited client
  - $\delta$: Total accumulated demand
  - $t$: Total accumulated time in route
  - $ltw$: Last customer for which time window was hit

## Takeaway

Nonlinear pricing: Better bounds, but solving pricing requires much more work.

# Pricing with other oracles

# Pricing with other oracles (Lubke, F., and Ricardez-Sandoval, 2024)

*Disclosure: Similar approach has been done in several other papers (e.g. Jaumard, Semet, and Vovor, 1998)*

# Pricing with other oracles (Lubke, F., and Ricardez-Sandoval, 2024)

*Disclosure: Similar approach has been done in several other papers (e.g. Jaumard, Semet, and Vovor, 1998)*

Context:

- Schedule analysis, that must be done in a particular sequence
- Analysis must be done by personnel, whose schedules have several requirements (e.g. break time, lunch, shift characteristics, etc)

# Pricing with other oracles (Lubke, F., and Ricardez-Sandoval, 2024)

*Disclosure: Similar approach has been done in several other papers (e.g. Jaumard, Semet, and Vovor, 1998)*

Context:

- Schedule analysis, that must be done in a particular sequence
- Analysis must be done by personnel, whose schedules have several requirements (e.g. break time, lunch, shift characteristics, etc)

Basic model idea:

- Time-discretized model (e.g. every 15 minutes)
- Analysis Schedule variables/constraints (precedence, machine capacity, etc.)
- Personnel constraints (break time, lunch, etc.):
  - Variables $z_{ept} \in \{0, 1\}$: Whether employee $e$ (or type $e$) is working process $p$ at time $t$
- Constraints linking both

# Pricing with other oracles (Lubke, F., and Ricardez-Sandoval, 2024)

Problem:

- Expressing personnel constraints in terms $z_{ept}$ variables may be tricky
- May require Big-M and/or additional binary variables
- Examples:
  - If an employee started working at time $t$, they must stay working for 10 periods
  - An employee must take lunch break of 1h between 11:30am and 2:30pm
  - Employees must take "regular" breaks (e.g. one 15 minute break every 4h)

# Pricing with other oracles (Lubke, F., and Ricardez-Sandoval, 2024)

Problem:

- Expressing personnel constraints in terms $z_{ept}$ variables may be tricky
- May require Big-M and/or additional binary variables
- Examples:
  - If an employee started working at time $t$, they must stay working for 10 periods
  - An employee must take lunch break of 1h between 11:30am and 2:30pm
  - Employees must take "regular" breaks (e.g. one 15 minute break every 4h)

Idea:

- Use columns $\lambda_{ei}$ to represent a possible shift $i$ of employee $e$
- Link $z_{ept}$ with $\lambda_{ei}$

# Pricing with other oracles (Lubke, F., and Ricardez-Sandoval, 2024)

Problem:

- Expressing personnel constraints in terms $z_{ept}$ variables may be tricky
- May require Big-M and/or additional binary variables
- Examples:
  - If an employee started working at time $t$, they must stay working for 10 periods
  - An employee must take lunch break of 1h between 11:30am and 2:30pm
  - Employees must take "regular" breaks (e.g. one 15 minute break every 4h)

Idea:

- Use columns $\lambda_{ei}$ to represent a possible shift $i$ of employee $e$
- Link $z_{ept}$ with $\lambda_{ei}$
- Generate columns using constraint programming
  - Scheduling constraints and logical constraints are easier to implement
  - More flexible and efficient than IP-based (for this problem)
  - Reasonably fast in practice

# Pricing with other oracles (Lubke, F., and Ricardez-Sandoval, 2024)

Problem:

- Expressing personnel constraints in terms $z_{ept}$ variables may be tricky
- May require Big-M and/or additional binary variables
- Examples:
  - If an employee started working at time $t$, they must stay working for 10 periods
  - An employee must take lunch break of 1h between 11:30am and 2:30pm
  - Employees must take "regular" breaks (e.g. one 15 minute break every 4h)

Idea:

- Use columns $\lambda_{ei}$ to represent a possible shift $i$ of employee $e$
- Link $z_{ept}$ with $\lambda_{ei}$
- Generate columns using constraint programming
  - Scheduling constraints and logical constraints are easier to implement
  - More flexible and efficient than IP-based (for this problem)
  - Reasonably fast in practice
- Pure scheduling (without employees) is solved better by IP-techniques

## Takeaway

Pricing: We can use diverse tools even if they don't have good worst-case guarantees.

## Takeaway

Pricing: We can use diverse tools even if they don't have good worst-case guarantees.

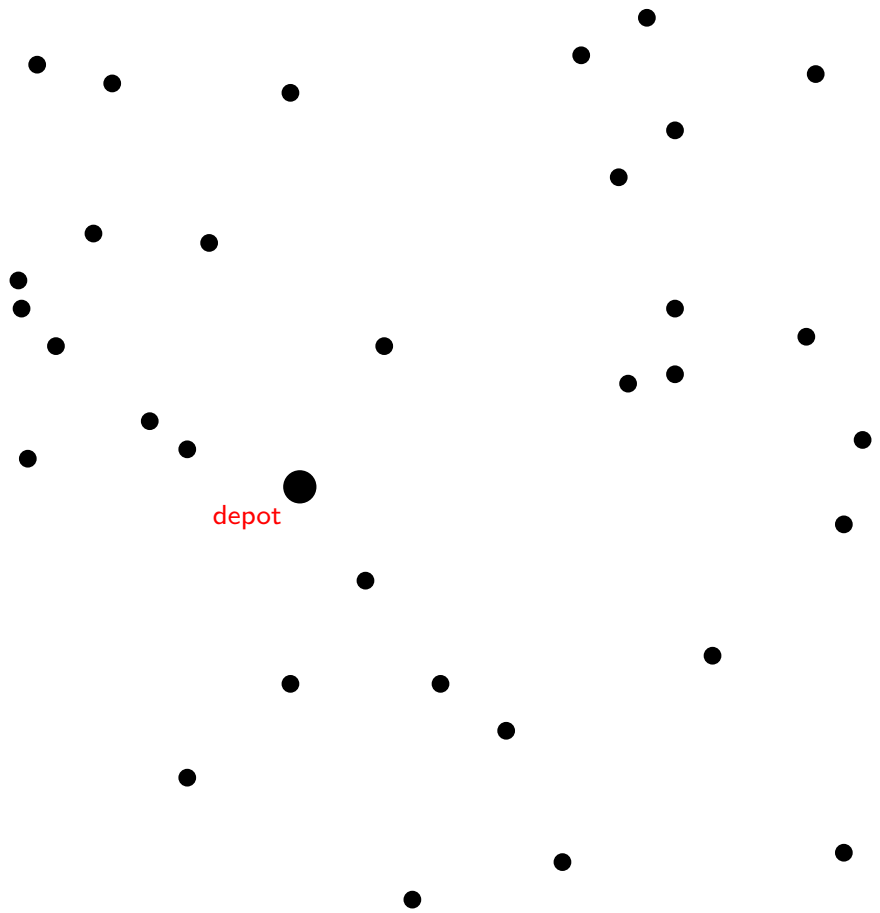
## Takeaway 2

CG allows to combine IP-based approach with other techniques that may work better for a different part of the problem.

## Issue: Dealing with hard pricing

# When pricing is too hard (Dinh, F., and Luedtke, 2018)

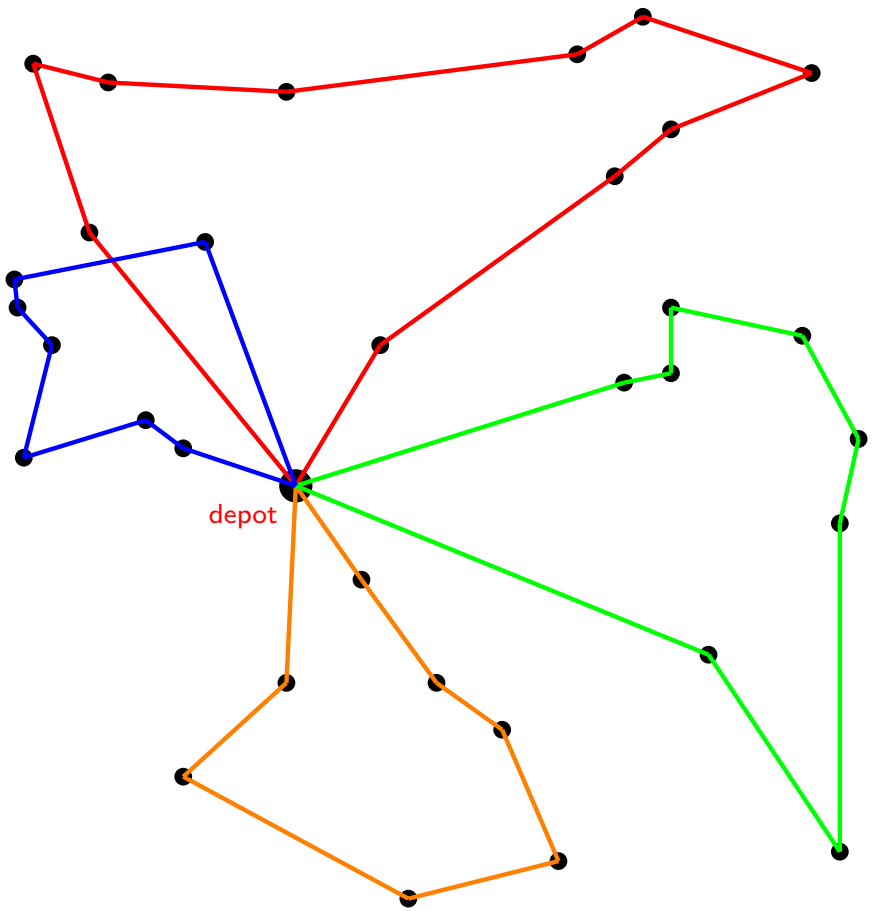
Determinstic VRP




- $G = (V, E)$
- $V = \{0\} \cup V_+$
- Edge lengths $\ell_e$, $e \in E$
- $K$ vehicles, capacity $C$
- Find a set of $K$ routes with minimum total length
- Client demands $d_i, \forall i \in V_+$

# When pricing is too hard (Dinh, F., and Luedtke, 2018)

Determinstic VRP

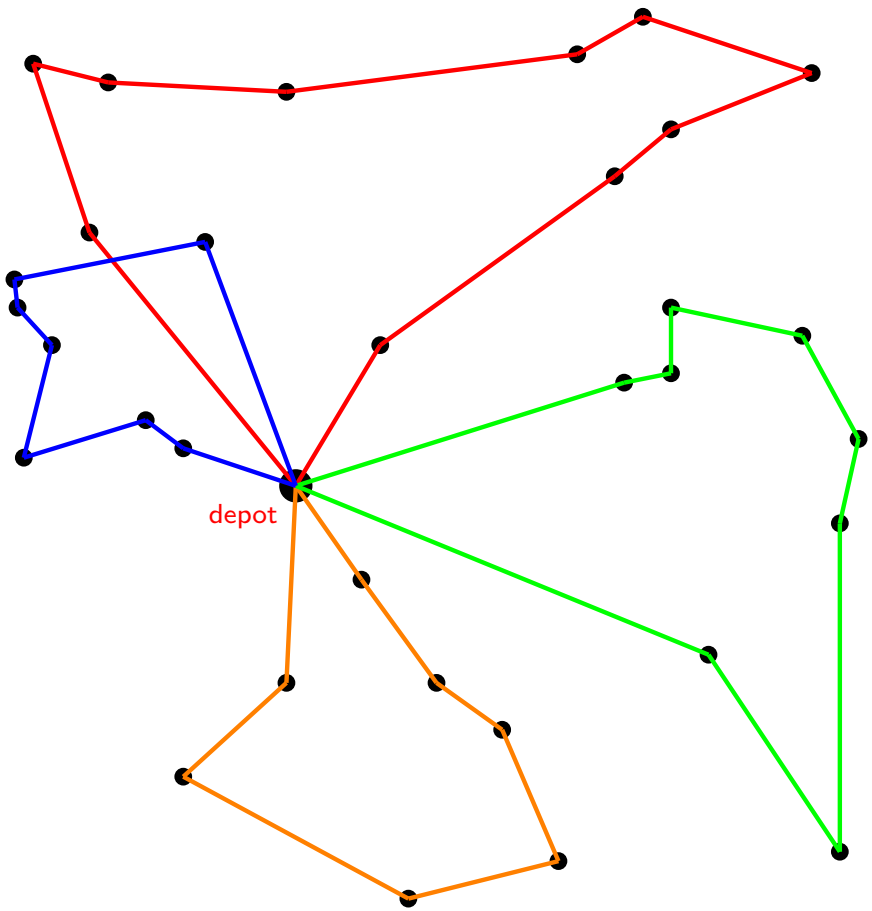



- $G = (V, E)$
- $V = \{0\} \cup V_+$
- Edge lengths $\ell_e$, $e \in E$
- $K$ vehicles, capacity $C$
- Find a set of $K$ routes with minimum total length
- Client demands $d_i, \forall i \in V_+$

- Let $S_j$ be the set of clients served by route $j$.
  Then $d(S_j) := \sum_{i \in S_j} d_i \leq C$

# When pricing is too hard (Dinh, F., and Luedtke, 2018)

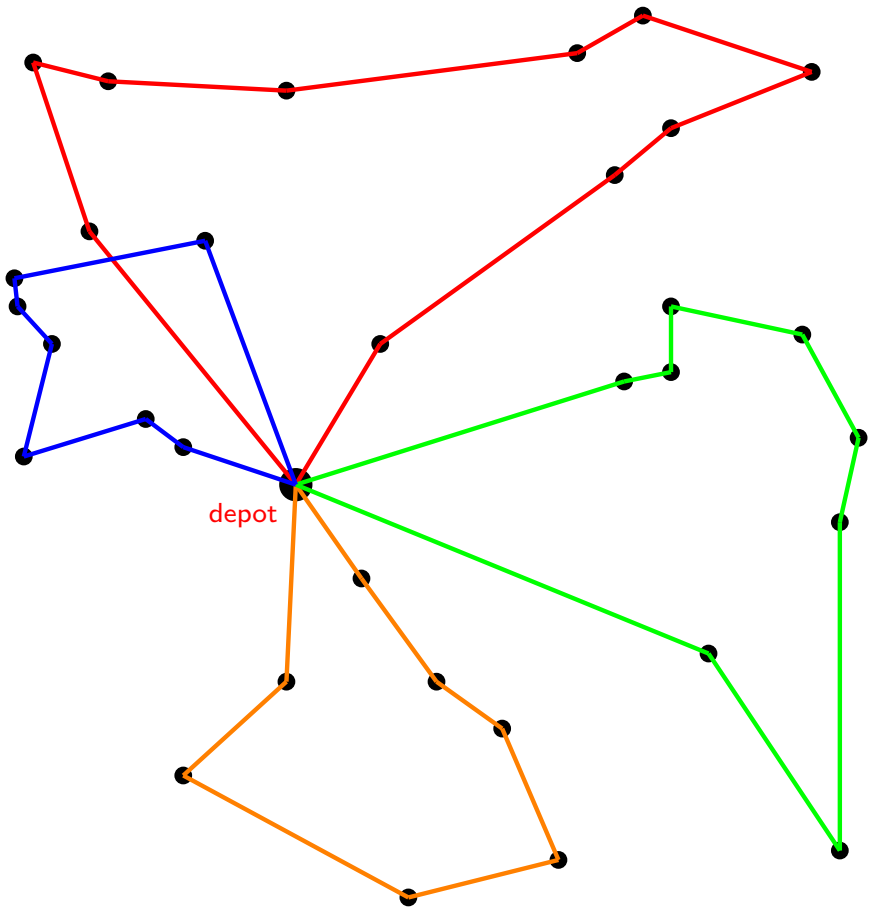
## Chance-constrained VRP




- $G = (V, E)$
- $V = \{0\} \cup V_+$
- Edge lengths $\ell_e$, $e \in E$
- $K$ vehicles, capacity $C$
- Find a set of $K$ routes with minimum total length
- ~~Client demands $d_i, \forall i \in V_+$~~
- Demands $D_i, \forall i \in V_+$: random variables that only get realized after routes have been decided
- Let $S_j$ be the set of clients served by route $j$.
  ~~Then $d(S_j) := \sum_{i \in S_j} d_i \leq C$~~

# When pricing is too hard (Dinh, F., and Luedtke, 2018)

## Chance-constrained VRP




- $G = (V, E)$
- $V = \{0\} \cup V_+$
- Edge lengths $\ell_e$, $e \in E$
- $K$ vehicles, capacity $C$
- Find a set of $K$ routes with minimum total length
- ~~Client demands $d_i, \forall i \in V_+$~~
- Demands $D_i, \forall i \in V_+$: random variables that only get realized after routes have been decided
- Let $S_j$ be the set of clients served by route $j$.
  Then $\mathbb{P}\{D(S_j) \leq C\} \geq 1 - \epsilon$

## When pricing is too hard (Dinh, F., and Luedtke, 2018)

Column generation for stochastic VRP:

Find a walk $0, v_1, \ldots, v_k, 0$ such that $\mathbb{P}\left\{\sum_{i=1}^{k} D_{v_i} \leq C\right\} \geq 1 - \epsilon$.

**Theorem (Dinh, F., and Luedtke, 2018)**

*Finding the least cost walk (q-route) in a graph that respects the capacity chance constraint under the finite distribution model or independent normal is strongly NP-hard.*

# When pricing is too hard (Dinh, F., and Luedtke, 2018)

Column generation for stochastic VRP:

Find a walk $0, v_1, \ldots, v_k, 0$ such that $\mathbb{P}\left\{\sum_{i=1}^{k} D_{v_i} \leq C\right\} \geq 1 - \epsilon$.

**Theorem (Dinh, F., and Luedtke, 2018)**

*Finding the least cost walk (q-route) in a graph that respects the capacity chance constraint under the finite distribution model or independent normal is strongly NP-hard.*

Solution

- Derive a valid branch-and-cut approach

# When pricing is too hard (Dinh, F., and Luedtke, 2018)

Column generation for stochastic VRP:

Find a walk $0, v_1, \ldots, v_k, 0$ such that $\mathbb{P}\left\{\sum_{i=1}^{k} D_{v_i} \leq C\right\} \geq 1 - \epsilon$.

**Theorem (Dinh, F., and Luedtke, 2018)**

*Finding the least cost walk (q-route) in a graph that respects the capacity chance constraint under the finite distribution model or independent normal is strongly NP-hard.*

Solution

- Derive a valid branch-and-cut approach
- Define an easier pricing problem, which will allow integer solutions to be infeasible

# When pricing is too hard (Dinh, F., and Luedtke, 2018)

Column generation for stochastic VRP:

Find a walk $0, v_1, \ldots, v_k, 0$ such that $\mathbb{P}\left\{\sum_{i=1}^{k} D_{v_i} \leq C\right\} \geq 1 - \epsilon$.

**Theorem (Dinh, F., and Luedtke, 2018)**

*Finding the least cost walk (q-route) in a graph that respects the capacity chance constraint under the finite distribution model or independent normal is strongly NP-hard.*

Solution

- Derive a valid branch-and-cut approach
- Define an easier pricing problem, which will allow integer solutions to be infeasible
- Branch-and-price is NOT a valid approach

# When pricing is too hard (Dinh, F., and Luedtke, 2018)

Column generation for stochastic VRP:

Find a walk $0, v_1, \ldots, v_k, 0$ such that $\mathbb{P}\left\{\sum_{i=1}^{k} D_{v_i} \leq C\right\} \geq 1 - \epsilon$.

**Theorem (Dinh, F., and Luedtke, 2018)**

*Finding the least cost walk (q-route) in a graph that respects the capacity chance constraint under the finite distribution model or independent normal is strongly NP-hard.*

Solution

- Derive a valid branch-and-cut approach
- Define an easier pricing problem, which will allow integer solutions to be infeasible
- Branch-and-price is NOT a valid approach
- Must be combined with cuts to yield a valid approach

# When pricing is too hard (Dinh, F., and Luedtke, 2018)

Column generation for stochastic VRP:

Find a walk $0, v_1, \ldots, v_k, 0$ such that $\mathbb{P}\left\{\sum_{i=1}^{k} D_{v_i} \leq C\right\} \geq 1 - \epsilon$.

**Theorem (Dinh, F., and Luedtke, 2018)**

*Finding the least cost walk (q-route) in a graph that respects the capacity chance constraint under the finite distribution model or independent normal is strongly NP-hard.*

Solution

- Derive a valid branch-and-cut approach
- Define an easier pricing problem, which will allow integer solutions to be infeasible
- Branch-and-price is NOT a valid approach
- Must be combined with cuts to yield a valid approach
- Noteworthy: Typically both branch-and-price and branch-and-cut approaches are valid

## Takeaway

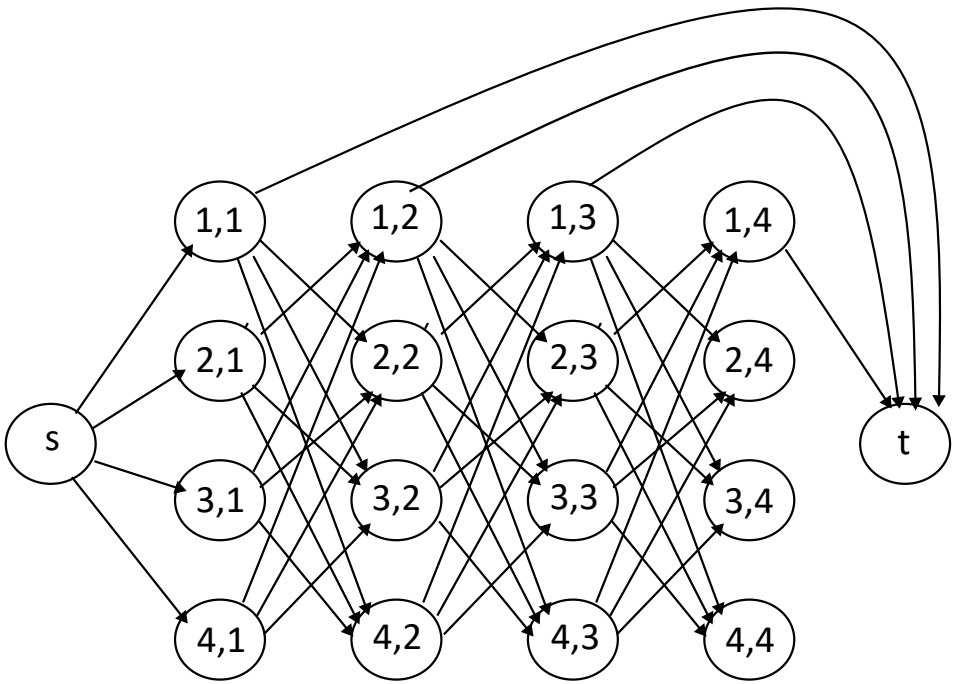
Combining pricing and cutting is advantageous.
For BPC, only ONE of them must yield a valid formulation.
The other one may be chosen carefully to strengthen, but not be too expensive.

# Dynamically refining CG

# Refining pricing dynamically (Cire et al., 2025)

Column elimination

- Start with $q$-route formulation



# Refining pricing dynamically (Cire et al., 2025)

Column elimination

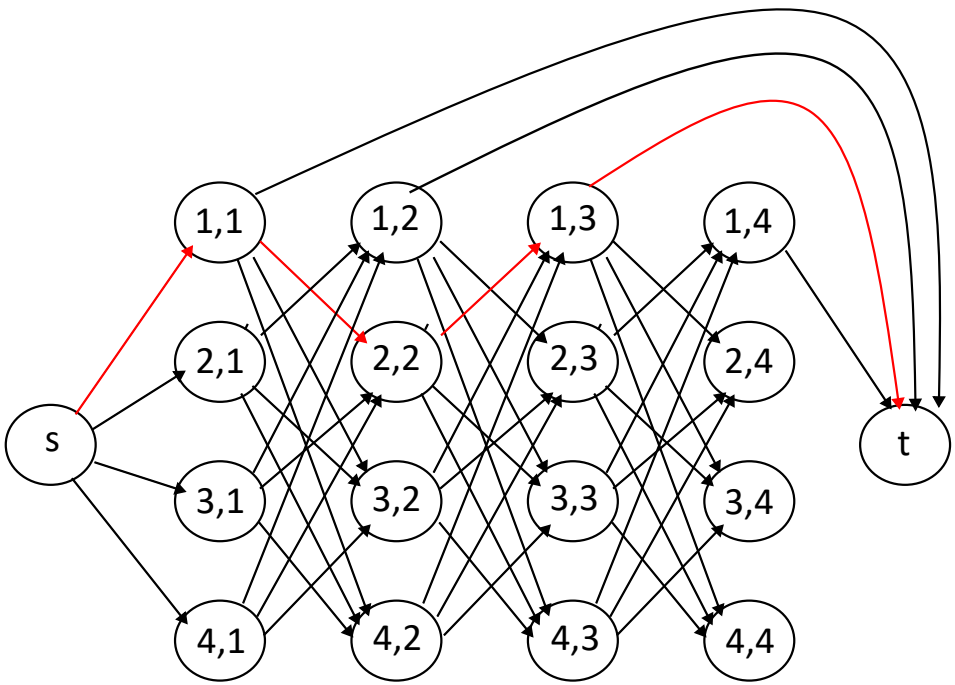
- Start with $q$-route formulation
- Formulate problem as a network flow over the state-space graph

$$
\begin{aligned}
\min \quad & \sum_{a \in \mathcal{A}} c_a y_a \\
\text{s.t.} \quad & \sum_{a \in \delta^-(u)} y_a - \sum_{a \in \delta^+(u)} y_a = 0, \forall u \in \mathcal{N} \setminus \{s, t\} \\
& \sum_{a \in \mathcal{A}^i} y_a = 1, \forall i \in V_+ \\
& \sum_{a \in \delta^+(s)} y_a = K
\end{aligned}
$$

# Refining pricing dynamically (Cire et al., 2025)

Column elimination

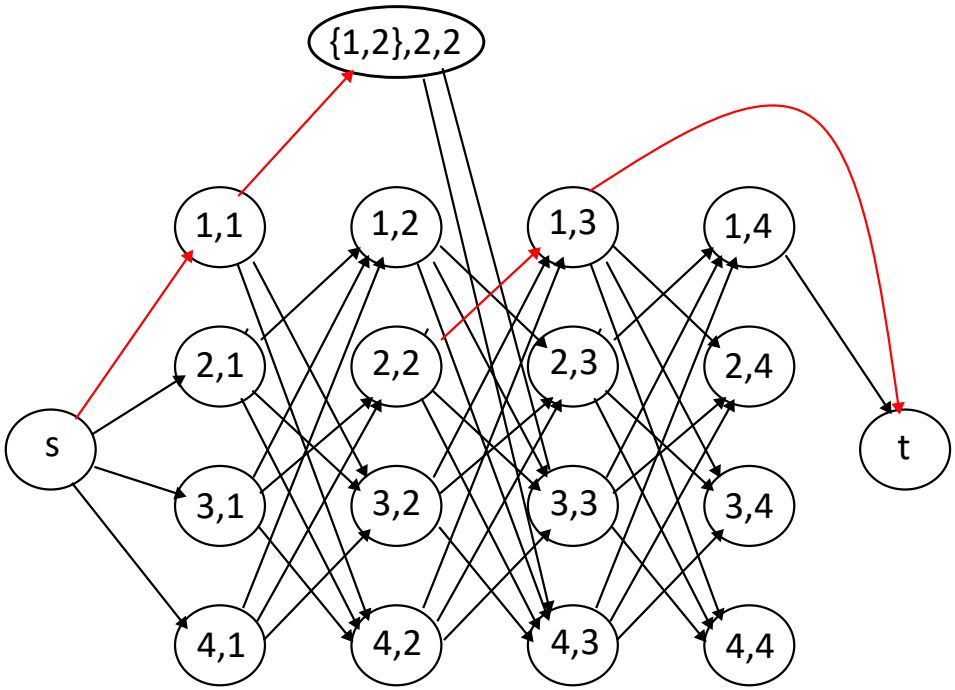
- Start with $q$-route formulation
- Formulate problem as a network flow over the state-space graph
- Iteratively refine it "as needed"



# Refining pricing dynamically (Cire et al., 2025)

Column elimination

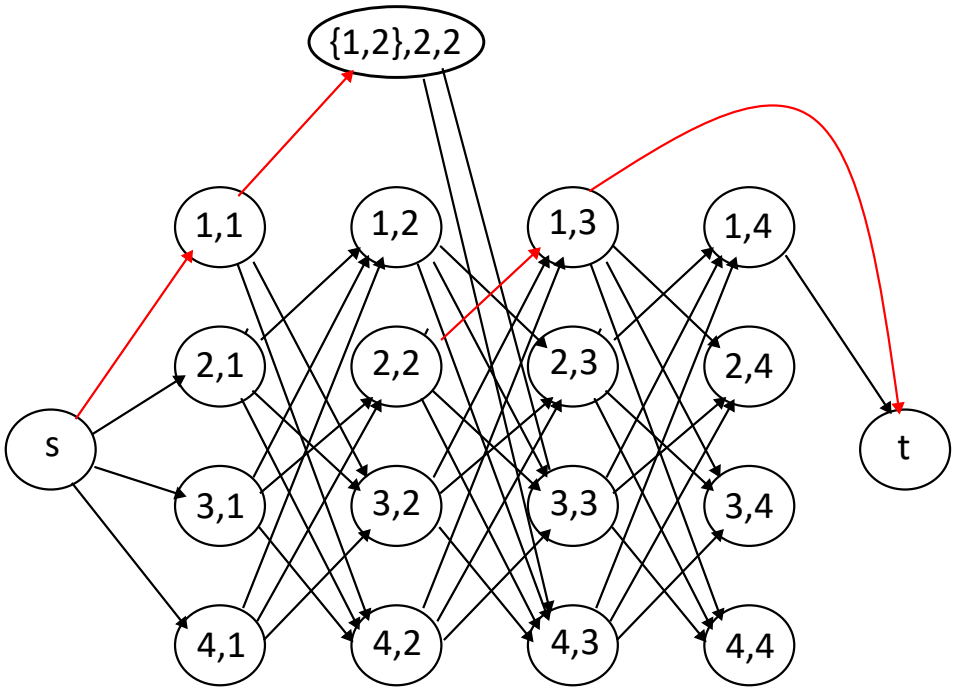
- Start with $q$-route formulation
- Formulate problem as a network flow over the state-space graph
- Iteratively refine it "as needed"



# Refining pricing dynamically (Cire et al., 2025)

Column elimination

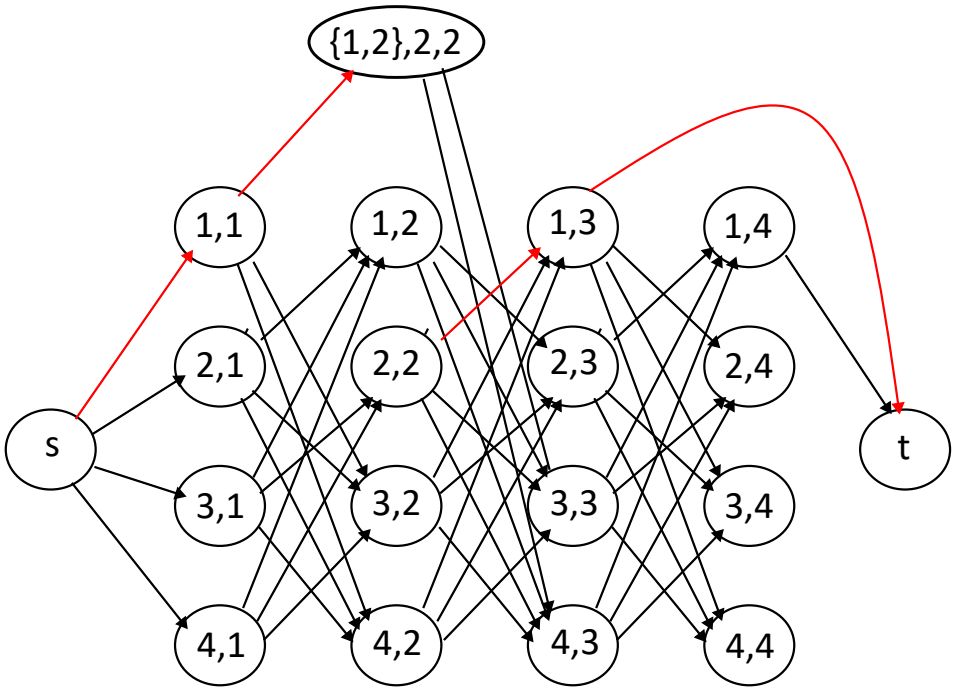
- Start with $q$-route formulation
- Formulate problem as a network flow over the state-space graph
- Iteratively refine it "as needed"
- Similar to dynamic ng-routes, decremental state-space relaxation approaches to eliminate cycles (works in the full DP-state space)



# Refining pricing dynamically (Cire et al., 2025)

Column elimination

- Start with $q$-route formulation
- Formulate problem as a network flow over the state-space graph
- Iteratively refine it "as needed"
- Similar to dynamic ng-routes, decremental state-space relaxation approaches to eliminate cycles (works in the full DP-state space)
- Infeasibility in column elimination does not need to be because of cycles



# Refining pricing dynamically (Cire et al., 2025)

Basic idea:

- Define a network $\mathcal{N}$ on the original graph
- Work with basic CG approach on top of $\mathcal{N}$
- For all CG-based variables $\lambda_r$ that are $> 0$ and NOT feasible:
  - Do CE refinement on $\mathcal{N}$ to remove $r$
- Repeat

# Refining pricing dynamically (Cire et al., 2025)

Basic idea:

- Define a network $\mathcal{N}$ on the original graph
- Work with basic CG approach on top of $\mathcal{N}$
- For all CG-based variables $\lambda_r$ that are $> 0$ and NOT feasible:
  - Do CE refinement on $\mathcal{N}$ to remove $r$
- Repeat

- When applied to CCVRP:
  - Allows to eliminate chance-constraint infeasible columns
  - Avoid having to do strongly np-hard pricing
  - Produces state-of-the-art results

## Takeaway

Dynamically refining CG can be good idea.
(see poster by Matheus Ota)

## Takeaway

Dynamically refining CG can be good idea.
(see poster by Matheus Ota)

## Vague thought 2

Decision diagrams are similar to DP algorithms that are used in CG.
Using them in/with CG make sense.

# Replacing CG with cuts

## Replacing CG with cuts

Consider our extended formulation:

$$
\begin{array}{lllll}
z_{BPC} = & \min & c^T x & & \\
& \text{s.t.} & x & - \sum_{i \in \mathcal{R}} v^i \lambda_i & = 0 \\
& & Dx & & \geq f \\
& & & \sum_{i \in \mathcal{R}} \lambda_i & = 1 \\
& & x \in \mathbb{Z}^n & & \\
& & & \lambda_i \in [0,1], \forall i \in \mathcal{R} &
\end{array}
\qquad \text{(BPC)}
$$

- Produces good bounds.
- May be expensive to solve (particularly at every branch-and-bound node)

## Replacing CG with cuts

Consider our extended formulation:

$$
\begin{array}{lllll}
z_{BPC} = & \min & c^T x & & \\
& \text{s.t.} & x & - \sum_{i \in \mathcal{R}} v^i \lambda_i & = 0 \\
& & Dx & & \geq f \\
& & & \sum_{i \in \mathcal{R}} \lambda_i & = 1 \\
& & x \in \mathbb{Z}^n & & \\
& & & \lambda_i \in [0,1], \forall i \in \mathcal{R} &
\end{array}
\qquad \text{(BPC)}
$$

- Produces good bounds.
- May be expensive to solve (particularly at every branch-and-bound node)
- Question: Can we get the same bound by using only $x$ variables and cuts?

## Replacing CG with cuts

Consider our extended formulation:

$$
\begin{array}{lllll}
z_{BPC} = & \min & c^T x & & \\
& \text{s.t.} & x - \sum_{i \in \mathcal{R}} v^i \lambda_i & = 0 & (\pi) \\
& & Dx & \geq f & \\
& & \sum_{i \in \mathcal{R}} \lambda_i & = 1 & (\pi_o) \\
& & x \in \mathbb{Z}^n & & \\
& & \lambda_i \in [0,1], \forall i \in \mathcal{R} & &
\end{array}
\qquad \text{(BPC)}
$$

- Produces good bounds.
- May be expensive to solve (particularly at every branch-and-bound node)
- Question: Can we get the same bound by using only $x$ variables and cuts?
- Why?

# Replacing CG with cuts

Consider our extended formulation:

$$
\begin{array}{lllll}
z_{BPC} = & \min & c^T x & & \\
& \text{s.t.} & x - \sum_{i \in \mathcal{R}} v^i \lambda_i & = 0 & (\pi) \\
& & Dx & \geq f & \\
& & \sum_{i \in \mathcal{R}} \lambda_i & = 1 & (\pi_o) \\
& & x \in \mathbb{Z}^n & & \\
& & \lambda_i \in [0,1], \forall i \in \mathcal{R} & &
\end{array}
\qquad \text{(BPC)}
$$

- Produces good bounds.
- May be expensive to solve (particularly at every branch-and-bound node)
- Question: Can we get the same bound by using only $x$ variables and cuts?
- Why?
  - Can be used in branch-and-cut (implemented in commercial solvers)

# Replacing CG with cuts

Consider our extended formulation:

$$
\begin{array}{lllll}
z_{BPC} = & \min & c^T x & & \\
& \text{s.t.} & x - \sum_{i \in \mathcal{R}} v^i \lambda_i & = 0 & (\pi) \\
& & Dx & \geq f & \\
& & \sum_{i \in \mathcal{R}} \lambda_i & = 1 & (\pi_o) \\
& & x \in \mathbb{Z}^n & & \\
& & \lambda_i \in [0,1], \forall i \in \mathcal{R} & &
\end{array}
\qquad \text{(BPC)}
$$

- Produces good bounds.
- May be expensive to solve (particularly at every branch-and-bound node)
- Question: Can we get the same bound by using only $x$ variables and cuts?
- Why?
  - Can be used in branch-and-cut (implemented in commercial solvers)
  - Easier to intersect multiple relaxations

## Replacing CG with cuts

Consider our extended formulation:

$$
\begin{array}{lllll}
z_{BPC} = & \min & c^T x & & \\
& \text{s.t.} & x - \sum_{i \in \mathcal{R}} v^i \lambda_i & = 0 & (\pi) \\
& & Dx & \geq f & \\
& & \sum_{i \in \mathcal{R}} \lambda_i & = 1 & (\pi_o) \\
& & x \in \mathbb{Z}^n & & \\
& & \lambda_i \in [0,1], \forall i \in \mathcal{R} & &
\end{array}
\qquad \text{(BPC)}
$$

- Produces good bounds.
- May be expensive to solve (particularly at every branch-and-bound node)
- Question: Can we get the same bound by using only $x$ variables and cuts?
- Why?
  - Can be used in branch-and-cut (implemented in commercial solvers)
  - Easier to intersect multiple relaxations
- Trivial: $c^T x \geq z_{BPC} \rightarrow$ Terrible performance

Chen, Günlük, and Lodi (2024) and Ota, F., and Kazachkov (2025):
Better ways to do this: Benders/Fenchel cuts.

## Takeaway

There are good ways to derive cuts from DW formulation and recover same bound.

# Some other issues

# CG based heuristics

How to obtain good primal solutions?

- Strategy 1: Solve LP relaxation with CG
  Then solve the IP (without generating any more columns at BB nodes

# CG based heuristics

How to obtain good primal solutions?

- Strategy 1: Solve LP relaxation with CG
  Then solve the IP (without generating any more columns at BB nodes
- Strategy 2: Use formulation in "natural" variable space ($x$) combined with CG, and then do general MIP techniques based on the $x$ values

# CG based heuristics

How to obtain good primal solutions?

- Strategy 1: Solve LP relaxation with CG
  Then solve the IP (without generating any more columns at BB nodes
- Strategy 2: Use formulation in "natural" variable space ($x$) combined with CG, and then do general MIP techniques based on the $x$ values
- Strategy 3: Exploit special CG structure to guide your heuristic
  Diving heuristic: Joncour et al. (2010),
  Feasibility pump: Pesneau, Sadykov, and Vanderbeck (2012)

# Software and implementation

- Branch-and-price not supported by commercial solvers
- Open source supported:
  - SCIP
  - Highs
  - Coluna.jl
  - SYMPHONY
  - COIN
  - idol
- VRP specific: VRPsolver (also been used to solve non-routing problems)

# Final thoughts

# Final thoughts

- CG can be FUN

# Final thoughts

- CG can be FUN
- CG can be USEFUL

# Final thoughts

- CG can be FUN
- CG can be USEFUL
- CG can be HARD

# Final thoughts

- CG can be FUN
- CG can be USEFUL
- CG can be HARD
- CG can be INTERESTING

# Final thoughts

- CG can be FUN
- CG can be USEFUL
- CG can be HARD
- CG can be INTERESTING

My hope:

# Final thoughts

- CG can be FUN
- CG can be USEFUL
- CG can be HARD
- CG can be INTERESTING

My hope:

- That you learned something

# Final thoughts

- CG can be FUN
- CG can be USEFUL
- CG can be HARD
- CG can be INTERESTING

My hope:

- That you learned something
- That you remember what you learned in the future

# Final thoughts

- CG can be FUN
- CG can be USEFUL
- CG can be HARD
- CG can be INTERESTING

My hope:

- That you learned something
- That you remember what you learned in the future
- That it perhaps may be useful to you

# Final thoughts

- CG can be FUN
- CG can be USEFUL
- CG can be HARD
- CG can be INTERESTING

My hope:

- That you learned something
- That you remember what you learned in the future
- That it perhaps may be useful to you

THANK YOU!

# Bibliography I


Ceselli, A., L. Létocart, and E. Traversi (2022). "Dantzig–Wolfe reformulations for binary quadratic problems". In: *Mathematical Programming Computation* 14, 85—120.

Chen, Rui, Oktay Günlük, and Andrea Lodi (2024). "Recovering Dantzig–Wolfe Bounds by Cutting Planes". In: *Operations Research* 73.2, pp. 1128–1142.

Cire, A. et al. (2025). "Combining column elimination and column generation".

Dinh, T., R. F., and J. Luedtke (2018). "Exact algorithms for the chance-constrained vehicle routing problem". In: *Mathematical Programming Series B* 172.1–2, 105—138.

F., R. et al. (2018). "A joint vehicle routing and speed optimization problem". In: *INFORMS Journal on Computing* 30.4, 694—709.

Jaumard, B., F. Semet, and T. Vovor (1998). "A generalized linear programming model for nurse scheduling". In: *European Journal of Operational Research* 107.1, 1—18.

Jepsen, M. et al. (2008). "Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows". In: *Operations Research* 56.2, 497—511.

Joncour, C. et al. (2010). "Column Generation based Primal Heuristics". In: *Electronic Notes in Discrete Mathematics* 36, pp. 695–702.

Lima, V. L. et al. (2022). "Arc flow formulations based on dynamic programming: Theoretical foundations and applications". In: *European Journal of Operational Research* 296, pp. 3–21.

# Bibliography II


Lubke, D., R. F., and L. Ricardez-Sandoval (2024). "Integration of machine scheduling and personnel allocation for an industrial-scale analytical services facility using column generation".

Nemhauser, G.L. and S. Park (1991). "A polyhedral approach to edge coloring". In: *Operations Research Letters* 10.6, pp. 315–322.

Ota, M., R. F., and A. Kazachkov (2025). "Approximating value functions via corner Benders' cuts".

Pecin, D. et al. (2017). "Improved branch-cut-and-price for capacitated vehicle routing". In: *Mathematical Programming Computation* 9, 61—100.

Pesneau, P, R. Sadykov, and F. Vanderbeck (2012). "Feasibility Pump Heuristics for Column Generation Approaches". In: *Experimental Algorithms*. Ed. by R. Klasing. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 332–343.