Handling difficult linking constraints in CG

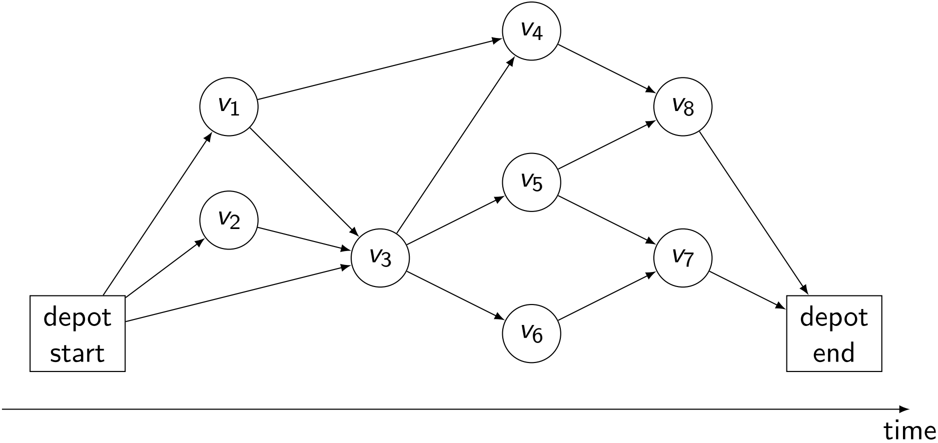
CERMICS

Axel Parmentier

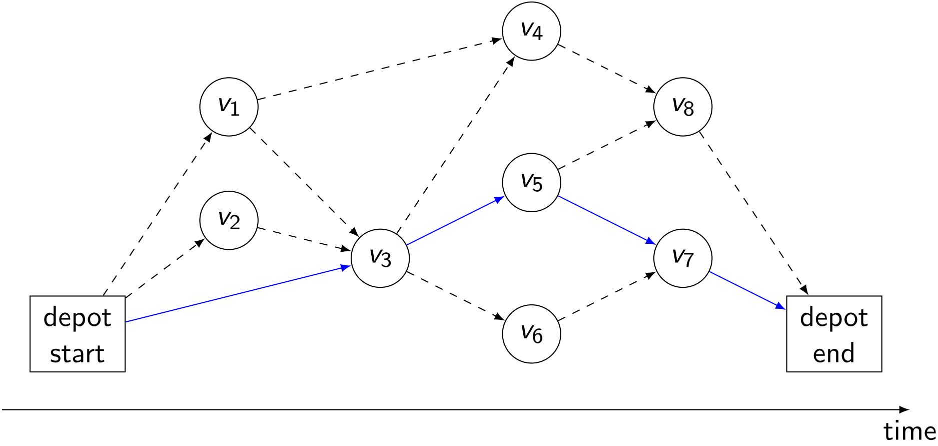
Tel Aviv, April 2018

How fixed point theorems in ordered algebraic structures enable to design practically efficient algorithms for industrial routing problems.

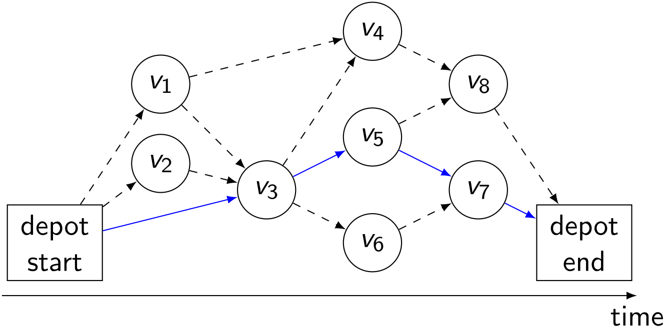
Problem settings



Build sequences of tasks operated



Column generation



$$\min \sum_{P \in \mathcal{P}} c_P x_P$$

$$\sum_{P \ni v} x_P = 1 \qquad \forall v$$

$$x_P \in \{0, 1\}$$

- Path cost not linear in arc costs
- Path must satisfy constraints

Constraint example

Limited number of arcs in P

Restricted master problem $\mathcal{P}' \subset \mathcal{P}$, with $|\mathcal{P}'| \ll |\mathcal{P}|$

$$\begin{aligned} \min_{x} \quad & \sum_{P \in \mathcal{P}} c_r x_r \\ \mathrm{st} \quad & \sum_{P \ni v} x_v = 1 \quad \forall \ell \in \mathcal{L} \\ & x_r \geq 0 \end{aligned}$$

Restricted master problem $\mathcal{P}' \subset \mathcal{P}$, with $|\mathcal{P}'| \ll |\mathcal{P}|$

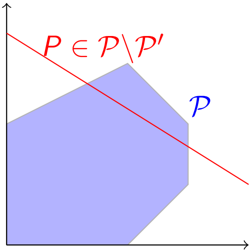
$$\begin{aligned} \min_x \quad & \sum_{P \in \mathcal{P}} c_r x_r \\ \text{st} \quad & \sum_{P \ni v} x_v = 1 \quad \forall \ell \in \mathcal{L} \\ & x_r \geq 0 \end{aligned}$$

Restricted dual problem

$$\begin{aligned} \max \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & \sum_{v \in P} y_P \leq c_P \quad \forall P \in \mathcal{P}' \end{aligned}$$

Pricing subproblem

$$\min_{P \in \mathcal{P}} c_P - \sum_{v \in P} y_P$$



Algorithm:

- solve on $\mathcal{P}'$
- solve pricing subproblem
- add violated dual constraint to $\mathcal{P}'$

Column generation primer

Restricted master problem $\mathcal{P}' \subset \mathcal{P}$, with $|\mathcal{P}'| \ll |\mathcal{P}|$

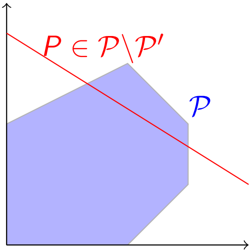
$$\begin{aligned} \min_x \quad & \sum_{P \in \mathcal{P}} c_r x_r \\ \text{st} \quad & \sum_{P \ni v} x_v = 1 \quad \forall \ell \in \mathcal{L} \\ & x_r \geq 0 \end{aligned}$$

Restricted dual problem

$$\begin{aligned} \max \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & \sum_{v \in P} y_P \leq c_P \quad \forall P \in \mathcal{P}' \end{aligned}$$

Pricing subproblem

$$\min_{P \in \mathcal{P}} c_P - \sum_{v \in P} y_P$$



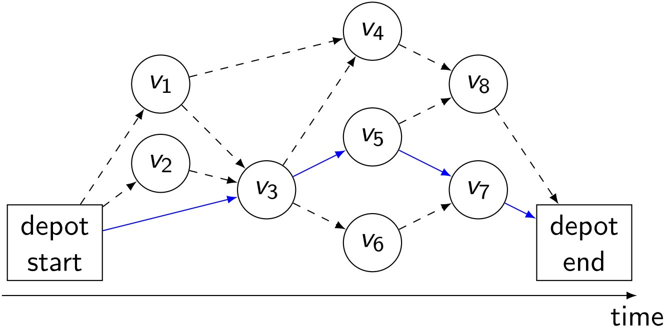
Algorithm:

- solve on $\mathcal{P}'$
- solve pricing subproblem
- add violated dual constraint to $\mathcal{P}'$

Key element in the performance: pricing subproblem algorithm

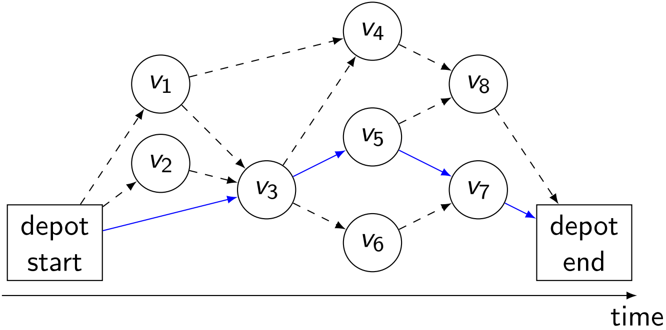
Resource constrained shortest path algorithm

$$\min_{P \in \mathcal{P}} c_P - \sum_{v \in P} y_P$$



Resource constrained shortest path algorithm

$$\min_{P \in \mathcal{P}} c_P - \sum_{v \in P} y_P$$



Pricing subproblem is a resource constrained shortest path algorithm

What a good pricing algorithms changes – Airline crew pairing

Instance	$\|V\|$	Alg	RCSP time av (mm:ss)	Pricing time	Total time (hh:mm:ss)
CP50	290	LS	00:00.560	97.55%	00:04:37.5
		LC	00:01.275	97.38%	00:11:36.9
		Our A*	00:00.016	59.87%	00:00:17.2
CP70	408	LS	00:11.489	99.52%	05:07:05.0
		LC	00:17.157	99.56%	07:28:22.2
		Our A*	00:00.039	58.48%	00:01:12.1
CP90	516	LS	00:40.707	Stopped after 48h	
		LC	01:42.864	Stopped after 48h	
		Our A*	00:00.340	81.86%	00:12:36.3
A318	669	LS	00:53.009	Stopped after 48h	
		LC	01:36.035	Stopped after 48h	
		Our A*	00:01.651	86.97%	01:32:49.6

Application of the method to Air France crew pairing problem (joint work with F. Meunier)

Part content

Shortest Path in an Ordered Monoid

For each arc a a resource $q_a \in \mathcal{R}$

- Associative binary operator $\oplus$: path resources
- Neutral element 0: empty path

$$\circ \xrightarrow{q_1} \circ \xrightarrow{q_2} \circ \xrightarrow{q_3} \circ \qquad q_P = q_1 \oplus q_2 \oplus q_3$$

$(\mathcal{R}, \oplus)$ is a monoid.

- An order $\preceq$ compatible with $\oplus$: $\quad q \preceq \tilde{q} \Rightarrow \begin{cases} r \oplus q \preceq r \oplus \tilde{q} \\ q \oplus r \preceq \tilde{q} \oplus r \end{cases}$

$(\mathcal{R}, \oplus, \preceq)$ is an ordered monoid.

- Non-decreasing cost c and constraint ρ functions.

Given an ordered monoid $(\mathcal{R}, \oplus, \preceq)$

Input:

- Digraph $D = (V, A)$
- Two vertices $o, d \in V$
- Resources $q_a \in \mathcal{R}$
- Two non-decreasing oracles $c : \mathcal{R} \to \mathbb{R}$
  $\rho : \mathcal{R} \to \{0, 1\}$

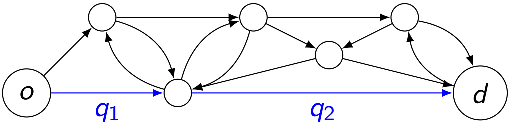
Output:

- An o-d path P such that

$$\rho\left(\bigoplus_{a \in P} q_a\right) = 0$$

which minimizes

$$c\left(\bigoplus_{a \in P} q_a\right)$$



$q_P = q_1 \oplus q_2$ $\rho(q_P) = 0$

Given an ordered monoid $(\mathcal{R}, \oplus, \preceq)$

Input:

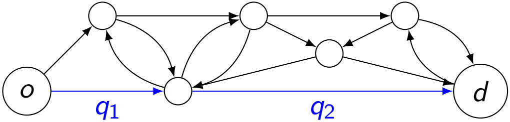
- Digraph $D = (V, A)$
- Two vertices $o, d \in V$
- Resources $q_a \in \mathcal{R}$
- Two non-decreasing oracles $c : \mathcal{R} \to \mathbb{R}$
  $\rho : \mathcal{R} \to \{0, 1\}$

Output:

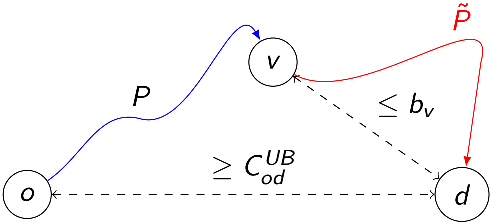
- An o-d path P such that
  $$\rho\left(\bigoplus_{a \in P} q_a\right) = 0$$
  which minimizes
  $$c\left(\bigoplus_{a \in P} q_a\right)$$

Jerusalem – Tel Aviv by car

- $q = (\delta, \tau)$
- Cost: $c(q) = \lambda_1 \delta + \lambda_2 \tau$
- On time arrival:
  $\rho(q) = \mathbb{1}_{(\tau_0, +\infty)}(\tau)$



$q_P = q_1 \oplus q_2$ $\qquad$ $\rho(q_P) = 0$



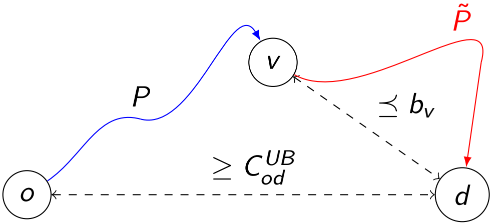
- $q_P \in \mathbb{R}$
- $C_{od}^{UB} \geq \min_{P \in \mathcal{P}_{o,d}} q_P$
- $b_v \leq q_P, \forall P \in \mathcal{P}_{vd}$

A path $P \in \mathcal{P}_{ov}$ satisfying $q_P + b_v > C_{od}^{UB}$ is not the subpath of an optimal path.

- Generate all the paths satisfying

$$q_P + b_v \leq C_{od}^{UB}$$

- Update C_{od}^{UB}



- $q_P \in \mathcal{R}$
- $C_{od}^{UB} \geq \min_{P|\rho(P)=0} c(q_P)$
- $b_v \preceq q_{\tilde{P}}, \forall \tilde{P} \in \mathcal{P}_{vd}$

A path $P \in \mathcal{P}_{ov}$ satisfying $c(q_P \oplus b_v) > C_{od}^{UB}$ or $\rho(q_P \oplus b_v) = 1$ is not the subpath of an optimal path.

Generalized A* Algorithm: a Branch & Bound

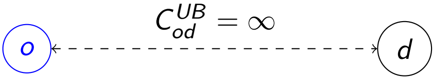
- Generate all the paths satisfying

$$c(q_P \oplus b_v) \leq C_{od}^{UB} \quad \text{and} \quad \rho(q_P \oplus b_v) = 0 \tag{Low}$$

- Update C_{od}^{UB}

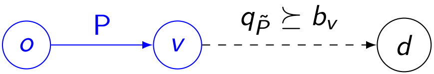
Generalized A* algorithm (2/2)

Initially: $L \leftarrow$ empty path in o

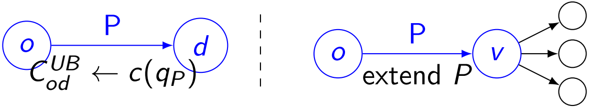


While L is not empty:

- extract $\min_{P \in L} c\,(q_P \oplus b_v)$



- If (Low) is satisfied, $\begin{cases} \rho(q_P \oplus b_v) = 0 \\ c(q_P \oplus b_v) < C_{od}^{UB} \end{cases}$



L: list of paths to be considered
C_{od}^{UB}: upper bound on optimal solution cost

Preprocessing: b_v lower bound on v-d paths resources

Key: $c(q_P \oplus b_v)$
Test: (Low)

Theorem

Under general assumptions (corresponding to the absence of negative cycles), A* *converges* after a finite number of iterations and

- if $C_{od}^{UB} = \infty$, then there is no feasible *o*-*d* paths,
- otherwise, C_{od}^{UB} is the cost of an optimal solution.

Instance	$\|V\|$	Alg	RCSP iter av. nb.	Cut Dom.	RCSP time av (mm:ss)
CP50	290	LS	1.020e+04	–	00:00.560
		LC	1.308e+04	–	00:01.275
		Our A*	4.914e+02	4.01%	00:00.016
CP70	408	LS	5.644e+04	–	00:11.489
		LC	7.730e+04	–	00:17.157
		Our A*	1.994e+03	4.28%	00:00.039
CP90	516	LS	9.779e+04	–	00:40.707
		LC	2.007e+05	–	01:42.864
		Our A*	9.966e+03	5.88%	00:00.340
A318	669	LS	1.319e+05	–	00:53.009
		LC	3.802e+05	–	01:36.035
		Our A*	2.549e+04	3.72%	00:01.651

Bound Computation

École des Ponts ParisTech

Definition: *lattice*

A partially ordered set $(\mathcal{R}, \preceq)$ is a lattice if any pair $(q, \tilde{q})$ admits:

A greatest lower bound or *meet* denoted $q \wedge \tilde{q}$

$$\left.\begin{array}{l} b \preceq q \\ b \preceq \tilde{q} \end{array}\right\} \Leftrightarrow b \preceq q \wedge \tilde{q}$$

$$\begin{tikzcd}[column sep=small] & q \vee \tilde{q} \arrow[dl, dash] \arrow[dr, dash] & \\ q \arrow[dr, dash] & & \tilde{q} \arrow[dl, dash] \\ & q \wedge \tilde{q} & \end{tikzcd}$$

A least upper bound or *join* denoted $q \vee \tilde{q}$

$$\left.\begin{array}{l} b \succeq q \\ b \succeq \tilde{q} \end{array}\right\} \Leftrightarrow b \succeq q \vee \tilde{q}$$

Example:

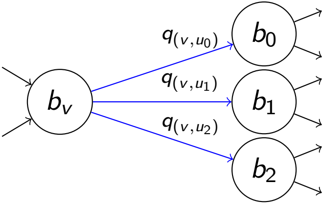
$(\mathbb{R}^2, \leq)$ endowed $\leq$ with the product order

- $q \wedge \tilde{q} = (\min(q_1, \tilde{q}_1), \min(q_2, \tilde{q}_2))$
- $q \vee \tilde{q} = (\max(q_1, \tilde{q}_1), \max(q_2, \tilde{q}_2))$

Ford-Bellman algorithm for usual shortest path problem

Minimum costs b_v of v-d paths satisfy the dynamic programming equation:

$$\begin{cases} b_d = 0, \\ b_{v \neq d} = \min\left(b_v, \min\limits_{u \in N^+(v)} \left(q_{(v,u)} + b_u\right)\right) \end{cases}$$



(b_v) is a fixed point of:

$$F : (b_v)_v \mapsto (b'_v)_v \text{ s.t.: } \begin{cases} b'_d = 0 \\ b'_{v \neq d} = \min\left(b_v, \min\limits_{u \in N^+(v)} \left(q_{(v,u)} + b_u\right)\right) \end{cases}$$

Usual Ford-Bellman algorithm

$(b_v^k) = F^k(\infty)$ is the cost of a shortest v-d path with at most k arcs.

If there is no cycles of negative costs, $(b_v) = F^n(\infty)$ satisfies the dynamic programming equation. $n = |V|$.

Generalized dynamic programming equation

$$\begin{cases} b_d = 0, \\ b_{v \neq d} = \bigwedge \left(q_v, \bigwedge_{u \in N^+(v)} \left(q_{(v,u)} \oplus b_u \right) \right) \end{cases}$$

Admits a greatest solution $b_v^\dagger$ (Knaster-Tarski fixed-point theorem)

$$F : (b_v)_v \mapsto (b'_v)_v \text{ st: } \begin{cases} b'_d = 0 \\ b'_{v \neq o} = \bigwedge \left(b_v, \bigwedge_{u \in N^+(v)} \left(q_{(v,u)} \oplus b_u \right) \right) \end{cases}$$

Generalized Ford-Bellman algorithm

$(b_v^k) = F^k(\infty) \preceq q_P$ for of any v-d path P with at most k arcs.

Generalized dynamic programming (2/2)

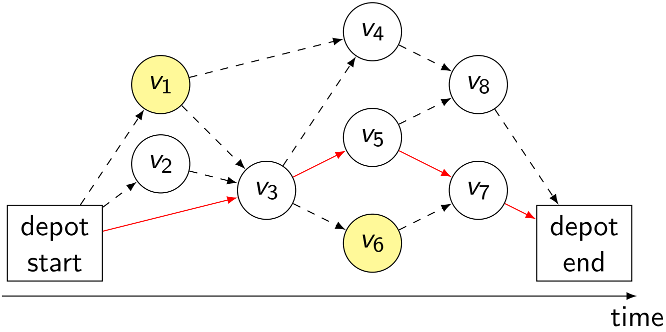
$$F : (b_v)_v \mapsto (b'_v)_v \text{ st: } \begin{cases} b'_d = 0 \\ b'_{v \neq o} = \bigwedge \left(b_v, \bigwedge_{u \in N^+(v)} \left(q_{(v,u)} \oplus b_u \right) \right) \end{cases}$$

- $b_v^k = F^k(b_v)$
- $b_v^\infty = \bigwedge_{k \in \mathbb{Z}_+} b_v^k$
- $b_v^\dagger = F(b_v^\dagger)$
- $b_v^{\mathrm{opt}} = \bigwedge_{p \in \mathcal{P}_{vd}} q_P$
- ℓ^*: nb arcs in longest elem. path

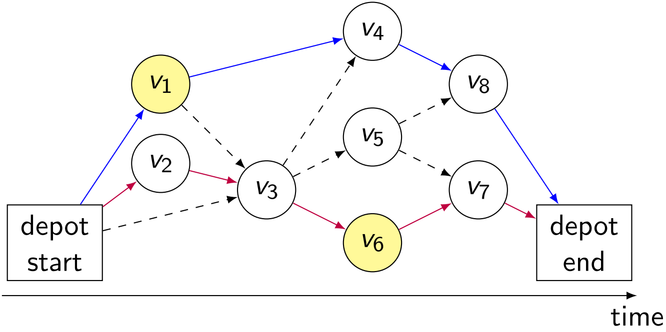
Theorem

$$b_v^\dagger \preceq b_v^\infty \preceq b_v^{\ell^*} \preceq b_v^{\mathrm{opt}} \preceq q_P \quad \text{for all } P \text{ in } \mathcal{P}_{vd}.$$

Part content


v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
depot start
depot end
time

Battery charge



$$\min \sum_{P \in \mathcal{P}} c_P x_P$$

$$\sum_{P \ni v} x_P = 1 \qquad \forall v$$

$$x_P \in \{0, 1\}$$

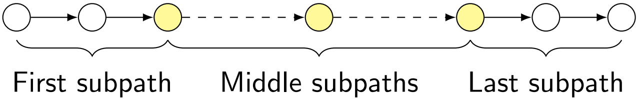
- Additional constraints on $\mathcal{P}$

Modeling the eletricity consumption constraint

Constraint easily modeled by a monoid if on full path

$$(\mathcal{R}, \oplus, \leqslant) = (\mathbb{Z}_+, +, \leq) \quad \rho(z) = \mathbb{1}_{z > \text{capacity}}$$

Then constraint on subpath modeled using



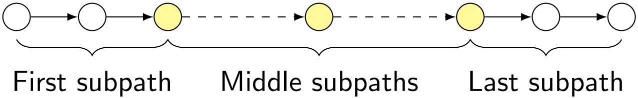
- use pairs $(r^{\mathrm{b}}, r^{\mathrm{e}})$ of resources in $\mathcal{R}^2$,
- turn them into an ordered monoid

Modeling the eletricity consumption constraint

Constraint easily modeled by a monoid if on full path

$$(\mathcal{R}, \oplus, \leqslant) = (\mathbb{Z}_+, +, \leq) \quad \rho(z) = \mathbb{1}_{z > \text{capacity}}$$

Then constraint on subpath modeled using

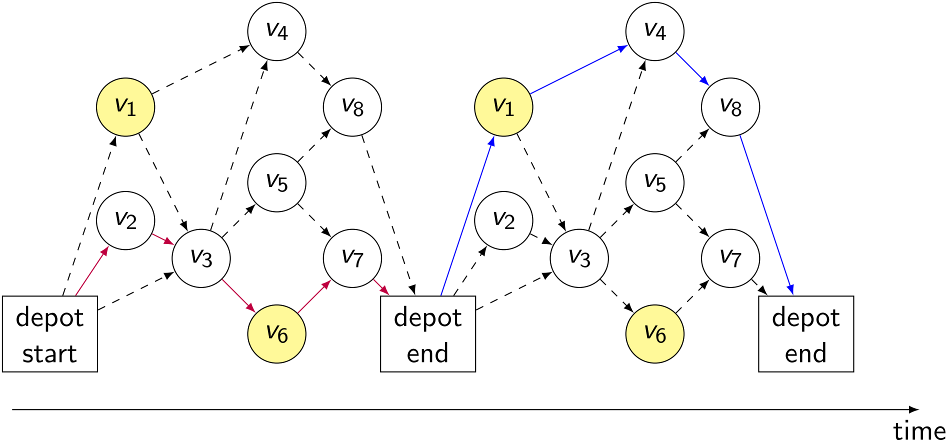


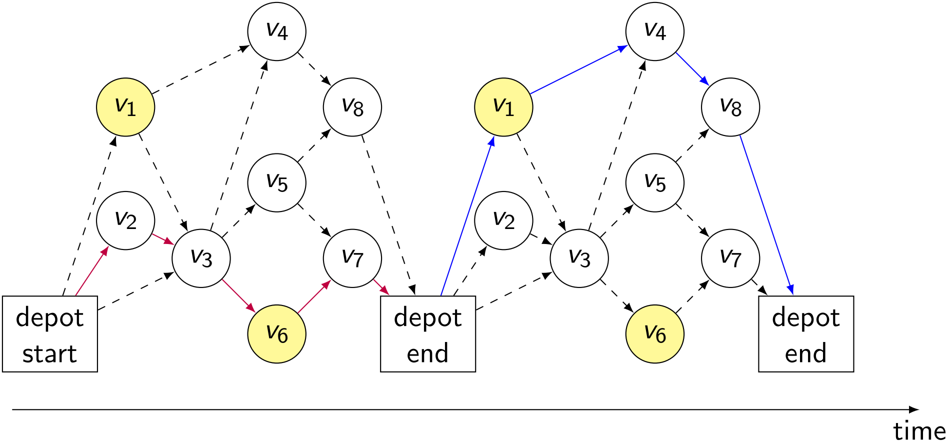
Ordered monoid $\mathcal{S} = \mathcal{R}^2 \cup \mathcal{R} \cup \{\infty\}$

$$\begin{aligned}
q \boxplus \infty = \infty \boxplus q &= \infty, \quad \forall q \in \mathcal{S} \\
(r_1) \boxplus (r_2^{\mathrm{b}}, r_2^{\mathrm{e}}) &= (r_1 \oplus r_2^{\mathrm{b}}, r_2^{\mathrm{e}}) \\
(r_1^{\mathrm{b}}, r_1^{\mathrm{e}}) \boxplus (r_2) &= (r_1^{\mathrm{b}}, r_1^{\mathrm{e}} \oplus r_2) \\
(r_1) \boxplus (r_2) &= (r_1 \oplus r_2) \\
(r_1^{\mathrm{b}}, r_1^{\mathrm{e}}) \boxplus (r_1^{\mathrm{b}}, r_2^{\mathrm{e}}) &= \begin{cases} \infty & \text{if } \rho(r_1^{\mathrm{e}} \oplus r_2^{\mathrm{e}}) = 1, \\ (r_1^{\mathrm{b}}, r_2^{\mathrm{e}}) & \text{otherwise.} \end{cases}
\end{aligned}$$

$$\begin{aligned}
q &\preceq \infty \quad \forall q \in \mathcal{S}, \\
(r_1) \sqsubseteq (r_2) \quad &\text{if} \quad r_1 \preceq \tilde{r}_2, \\
(r_1^{\mathrm{b}}, r_1^{\mathrm{e}}) \sqsubseteq (r_2) \quad &\text{if} \quad \begin{cases} r_1^{\mathrm{b}} \preceq r_2 \\ r_1^{\mathrm{e}} \preceq r_2 \end{cases} \\
(r_1^{\mathrm{b}}, r_1^{\mathrm{e}}) \sqsubseteq (r_2^{\mathrm{b}}, r_2^{\mathrm{e}}) \quad &\text{if} \quad \begin{cases} r_1^{\mathrm{b}} \preceq r_2^{\mathrm{b}}, \\ r_1^{\mathrm{e}} \preceq r_2^{\mathrm{e}}. \end{cases}
\end{aligned}$$

Coupling constraints

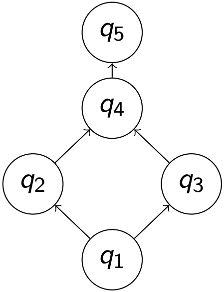




How to handle such constraints in column generation?

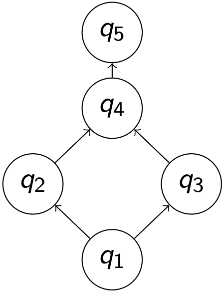
Handling coupling constraints

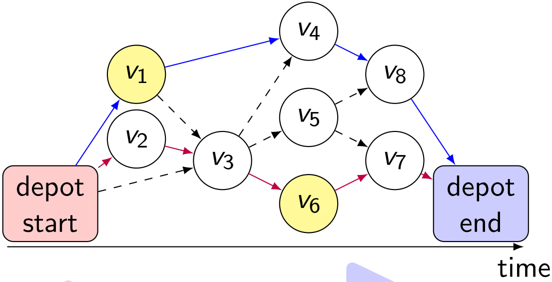
Lattice ordered monoid for constraints on subpath

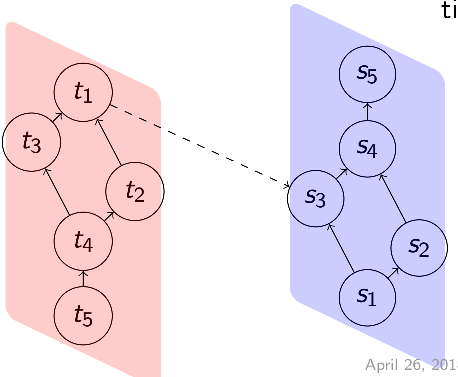


Handling coupling constraints

Lattice ordered monoid for constraints on subpath







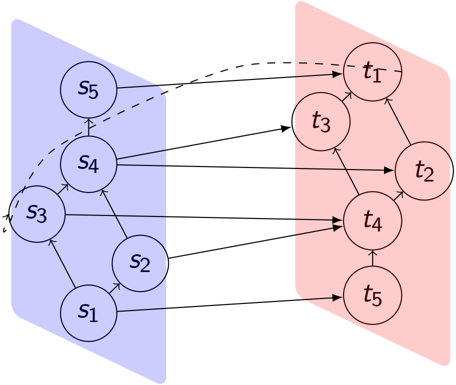
Define graph H on $U = S^2$ by adding

- Hasse diagrams
- (t_i, s_j) if there is $P \in \mathcal{P}$ with resources (q_j, q_i)
- arcs (s_i, t_j) if

$$\rho(q_i \oplus q_j) = 0$$

and

$$\rho(q_i \oplus q) = 1, \forall q > q_j$$

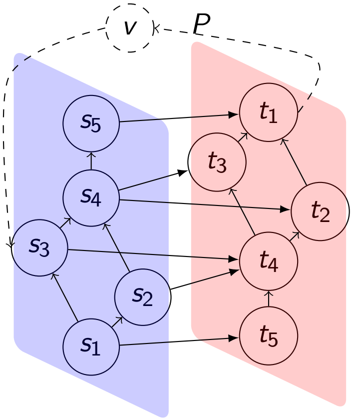


Given P and P' in $\mathcal{P}$, then P ends in s_i, and P' starts in t_j,

P and P' can be operated in a sequence $\Leftrightarrow$ there is an s_i-t_j path

Column generation formulation with coupling constraints

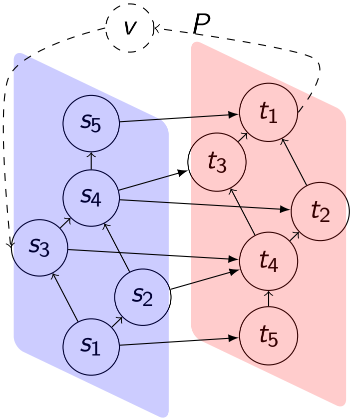
$$H = (U, A')$$



Primal

$$
\begin{aligned}
\min \quad & \sum_{P \in \mathcal{P}} c_P x_P & \\
& \sum_{P \ni v} x_P = 1 & \forall v \in V \\
& \sum_{a \in \delta^-(u)} x_a = \sum_{a \in \delta^+(u)} x_a & \forall u \in U \\
& x_a \geq 0 & \forall a \in A'
\end{aligned}
$$

$$H = (U, A')$$

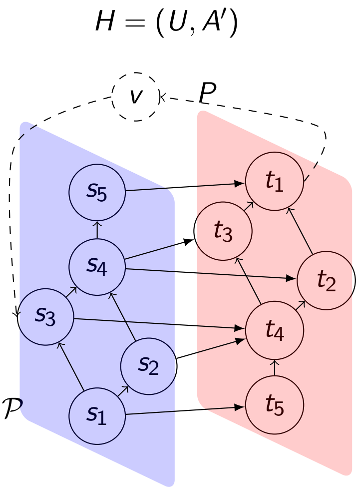


Primal

$$\begin{aligned}
\min \quad & \sum_{P \in \mathcal{P}} c_P x_P & \\
& \sum_{P \ni v} x_P = 1 & \forall v \in V \\
& \sum_{a \in \delta^-(u)} x_a \leq \sum_{a \in \delta^+(u)} x_a & \forall u \in U \\
& x_a \geq 0 & \forall a \in A'
\end{aligned}$$

Primal

$$\min \sum_{P\in\mathcal{P}} c_P x_P$$
$$\sum_{P\ni v} x_P = 1 \qquad \forall v \in V$$
$$\sum_{a\in\delta^-(u)} x_a \leq \sum_{a\in\delta^+(u)} x_a \qquad \forall u \in U$$
$$x_a \geq 0 \qquad \forall a \in A'$$



Dual

$$\max \sum_{v} y_v$$
$$\text{s.t.} \quad c_P - \lambda_{t(p)} + \lambda_{s(P)} - \sum_{v\in P} y_v \geq 0, \;\; \forall P \in \mathcal{P}$$
$$\lambda_u \leq \lambda_{u'}, \;\; \forall (u,u') \in A' \backslash \mathcal{P}$$
$$\lambda \geq 0$$

Pricing subproblem solved using the same monoid

$$\begin{aligned}
\max \quad & \sum_{v} y_v \\
\text{s.t.} \quad & c_P - \lambda_{t(p)} + \lambda_{s(P)} - \sum_{v \in P} y_v \geq 0, \quad \forall P \in \mathcal{P} \\
& \lambda_u \leq \lambda_{u'}, \quad \forall (u,u') \in A' \backslash \mathcal{P} \\
& \lambda \geq 0
\end{aligned}$$

$$q_i \preceq q_j \quad \text{implies} \quad \begin{cases} -\lambda_{t_i} \leq -\lambda_{t_j} \\ \lambda_{s_i} \leq \lambda_{s_j} \end{cases} \quad \text{, hence}$$

$$q_P \preceq q_Q \quad \text{implies} \quad c_P - \lambda_{t(P)} + \lambda_{s(P)} - \sum_{v \in P} y_v \leq c_Q - \lambda_{t(Q)} + \lambda_{s(Q)} - \sum_{v \in Q} y_v$$

Border constraints do not change (too much) the pricing subproblem

Pricing subproblem solved using the same monoid

$$\begin{aligned}
\max \quad & \sum_v y_v \\
\text{s.t.} \quad & c_P - \lambda_{t(p)} + \lambda_{s(P)} - \sum_{v \in P} y_v \geq 0, \quad \forall P \in \mathcal{P} \\
& \lambda_u \leq \lambda_{u'}, \quad \forall (u, u') \in A' \backslash \mathcal{P} \\
& \lambda \geq 0
\end{aligned}$$

$$q_i \preceq q_j \quad \text{implies} \quad \begin{cases} -\lambda_{t_i} \leq -\lambda_{t_j} \\ \lambda_{s_i} \leq \lambda_{s_j} \end{cases} \quad \text{, hence}$$

$$q_P \preceq q_Q \quad \text{implies} \quad c_P - \lambda_{t(P)} + \lambda_{s(P)} - \sum_{v \in P} y_v \leq c_Q - \lambda_{t(Q)} + \lambda_{s(Q)} - \sum_{v \in Q} y_v$$

Border constraints do not change (too much) the pricing subproblem

Numerical experiments in progress. Works well if no heuristic branching. Not that well if heuristic branching.