Universidad de los Andes
Colombia

Solving Hard Shortest Path Problems with the Pulse Framework

Lecture 3: Applications and perspectives

Andrés Medaglia, Ph.D.
(amedagli@uniandes.edu.co)

Joint work with:
L. Lozano, Ph.D. (U. of Cincinnati); D. Duque, Ph.D.(c) (Northwestern U.);
N. Cabrera, M.Sc. & D. Yamín (U. de los Andes); M. Bolívar, M.Sc. (Glovo)

Departamento de Ingeniería Industrial
Centro para la Optimización y Probabilidad Aplicada
Universidad de los Andes (Colombia)
Universidad de la República; Montevideo (Uruguay), Marzo 9-13

Actualizado: 2020/03/12

http://copa.uniandes.edu.co/

Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
- Part V: perspectives

Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
  - Multi-activity shift scheduling problem
  - Bus rapid transit route design problem
  - Interdiction problem with fortification
  - Green VRP
  - Other applications
- Part V: perspectives

Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
  - **Multi-activity shift scheduling problem**
  - Bus rapid transit route design problem
  - Interdiction problem with fortification
  - Green VRP
  - Other applications
- Part V: perspectives

Multi-activity shift scheduling problem

A primer in shift scheduling: BixiPizza

- BixiPizza es una cadena de comida rápida.
- Promesa de servicio:

  “Si no recibes tu BixiPizza caliente en 45 minutos, te la regalamos”
- Se determinó la zona de cobertura para un nivel de servicio del 95% para cada punto de venta.

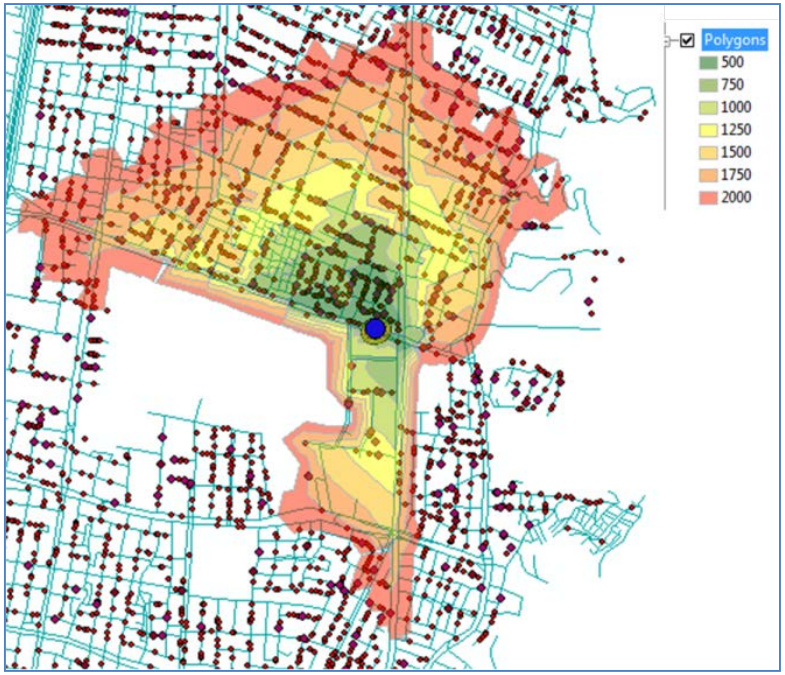
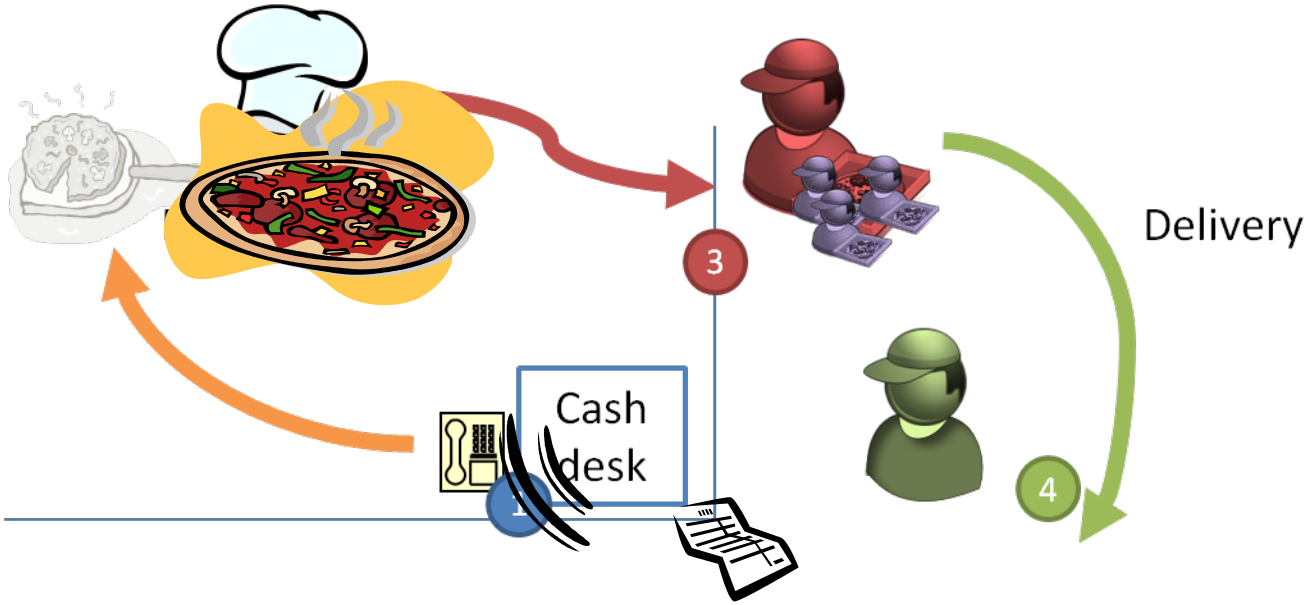


Figura 1: Zona de cobertura

Multi-activity shift scheduling problem

BixiPizza: Situación



Multi-activity shift scheduling problem

BixiPizza: Situación

Tabla 1: Mínimo número de domiciliarios requeridos en cada franja horaria del día sábado para cumplir la promesa de servicio de BixiPizza.

Franja Horaria (hh:mm- hh:mm)	Domiciliarios Requeridos
11:00 a. m. - 11:30 a. m.	2
11:30 a. m. - 12:00 p. m.	2
12:00 p. m. - 12:30 p. m.	5
12:30 p. m. - 01:00 p. m.	5
01:00 p. m. - 01:30 p. m.	5
01:30 p. m. - 02:00 p. m.	5
02:00 p. m. - 02:30 p. m.	5
... ...	...
07:00 p. m. - 07:30 p. m.	8
07:30 p. m. - 08:00 p. m.	8
08:00 p. m. - 08:30 p. m.	8
08:30 p. m. - 09:00 p. m.	2
09:00 p. m. - 09:30 p. m.	2
09:30 p. m. - 10:00 p. m.	2
10:00 p. m. - 10:30 p. m.	2
10:30 p. m. - 11:00 p. m.	2

Multi-activity shift scheduling problem

BixiPizza: Malla de turnos

Universidad de los Andes Colombia

Tiempo completo

- **Duración:**
  8 horas
- **Descanso:**
  1 hora
  - Descanso programado en la 4° hora.
- **Inicio:**
  Horas en punto
- **Costo/hora:**
  - $ 7,950 pesos

Hora	Turno 1	Turno 2	Turno 3	Turno 4	Turno 5
11:00-11:30					
11:30-12:00					
12:00-12:30					
12:30-13:00					
13:00-13:30					
13:30-14:00					
14:00-14:30					
14:30-15:00					
15:00-15:30					
15:30-16:00					
16:00-16:30					
16:30-17:00					
17:00-17:30					
17:30-18:00					
18:00-18:30					
18:30-19:00					
19:00-19:30					
19:30-20:00					
20:00-20:30					
20:30-21:00					
21:00-21:30					
21:30-22:00					
22:00-22:30					
22:30-23:00					

Multi-activity shift scheduling problem

BixiPizza: Turno tiempo completo

Universidad de los Andes Colombia

Tiempo completo

- **Duración:**
  8 horas
- **Descanso:**
  1 hora
  - Descanso programado en la 4ta hora.
- **Inicio:**
  Horas en punto
- **Costo/hora:**
  - $ 7,950 pesos

Hora	Turno 1	Turno 2	Turno 3	Turno 4	Turno 5
11:00-11:30					
11:30-12:00					
12:00-12:30					
12:30-13:00					
13:00-13:30					
13:30-14:00					
14:00-14:30					
14:30-15:00					
15:00-15:30					
15:30-16:00					
16:00-16:30					
16:30-17:00					
17:00-17:30					
17:30-18:00					
18:00-18:30					
18:30-19:00					
19:00-19:30					
19:30-20:00					
20:00-20:30					
20:30-21:00					
21:00-21:30					
21:30-22:00					
22:00-22:30					
22:30-23:00					

	Disponible
	No disponible

Multi-activity shift scheduling problem

BixiPizza: Turno tiempo parcial

Tiempo completo

- **Duración:**
  8 horas
- **Descanso:**
  1 hora
  - Descanso programado en la 4ta hora.
- **Inicio:**
  Horas en punto
- **Costo/hora:**
  - $ 7,950 pesos

Tiempo parcial

- **Duración:**
  6 horas
- **Descanso:**
  0.5 horas
  - Descanso al inicio de la 3ra hora.
- **Inicio:**
  Cada media-hora
- **Costo/hora:**
  - $ 9,710 pesos

Multi-activity shift scheduling problem

BixiPizza: Turno por horas (subcontratado)

Tiempo completo

- **Duración:** 8 horas
- **Descanso:** 1 hora
  - Descanso programado en la 4ta hora.
- **Inicio:** Horas en punto
- **Costo/hora:**
  - $ 7,950 pesos

Tiempo parcial

- **Duración:** 6 horas
- **Descanso:** 0.5 horas
  - Descanso al inicio de la 3ra hora.
- **Inicio:** Cada media-hora
- **Costo/hora:**
  - $ 9,710 pesos

Subcontratado

- **Duración:** 3 horas
- **Descanso:** No aplica
- **Inicio:** Horas en punto
- **Costo/hora:**
  - $ 12,000 pesos

Multi-activity shift scheduling problem

BixiPizza: Modelo de programación matemática

Universidad de los Andes Colombia

Objetivo

- Minimizar costos de contratación

Decisiones

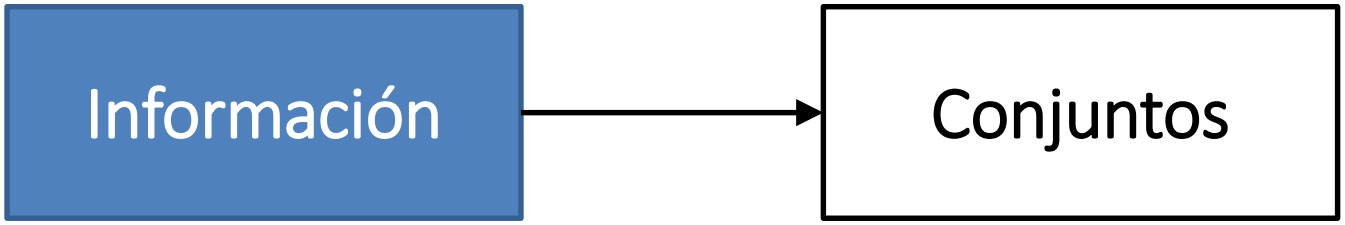
- ¿Cuántos domiciliarios contratar?
- ¿Qué franjas horarias deben cubrir?

Restricciones

- Se debe garantizar la disponibilidad de domiciliarios para cumplir la meta de nivel de servicio.
- No se puede contratar unidades fraccionales de personas.

Multi-activity shift scheduling problem

BixiPizza: Conjuntos



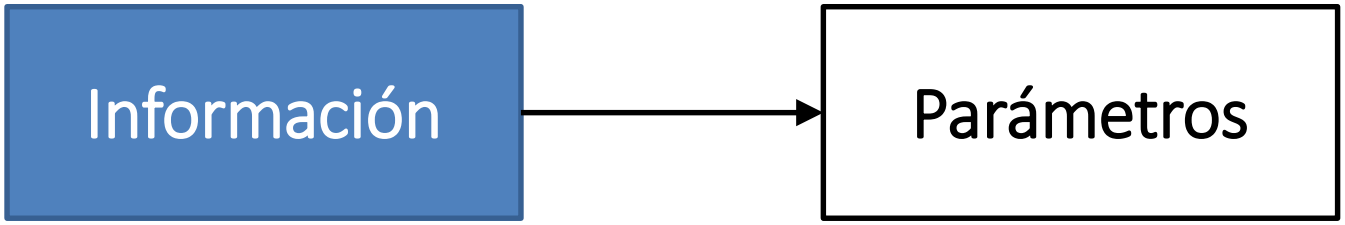
Conjuntos

Ω: Conjunto de turnos de trabajo.

T: Conjunto de franjas horarias de una jornada laboral.

Multi-activity shift scheduling problem

BixiPizza: Parámetros



Parámetros

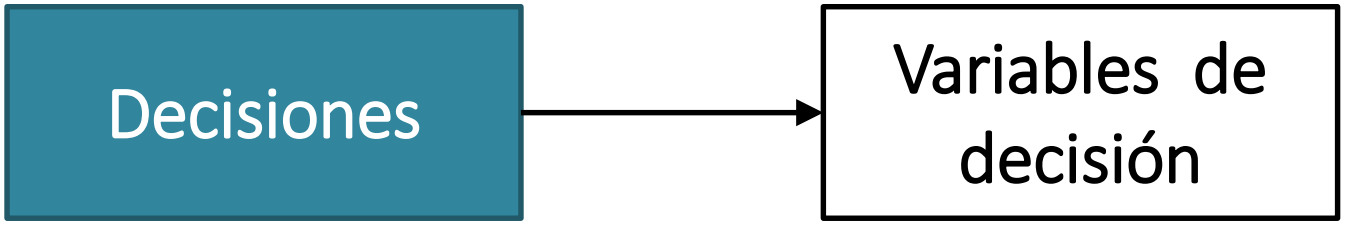
c_j: Costo diario por domiciliario contratado con el turno $j \in \Omega$.

d_t: Requerimientos de domiciliarios durante la franja horaria $t \in T$.

a_{tj}: Parámetro binario que toma el valor de 1 si un domiciliario con el turno $j \in \Omega$ está disponible durante la franja horaria $t \in T$, o toma el valor de 0 en caso contrario.

Multi-activity shift scheduling problem

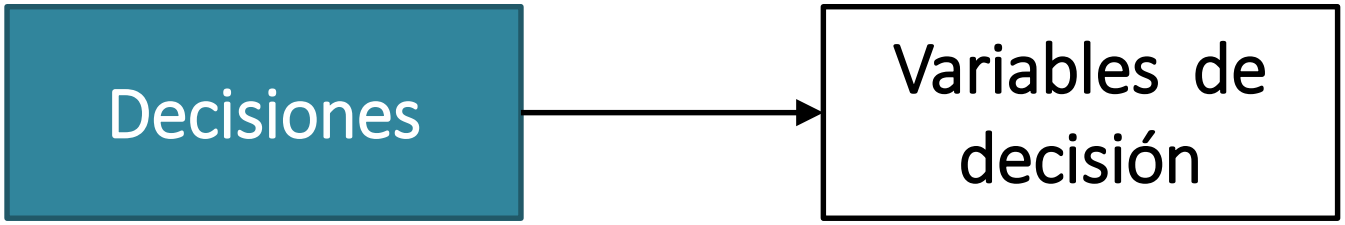
BixiPizza: Formulación del modelo



Variables de decisión (¿Qué debe decidirse?)

Multi-activity shift scheduling problem

BixiPizza: Variables de decisión

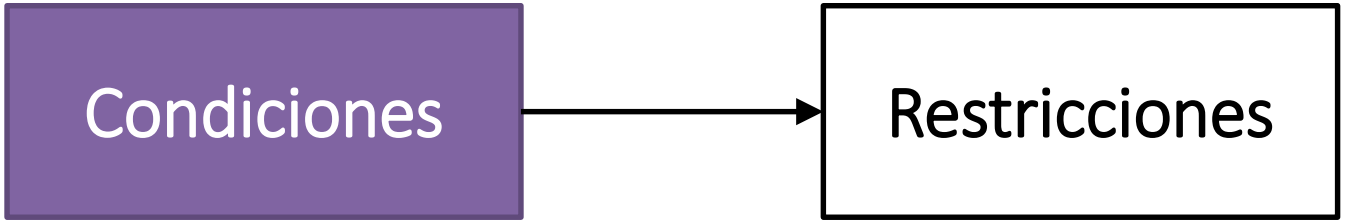


Variables de decisión (¿Qué debe decidirse?)

x_j: Cantidad de domiciliarios a contratar en el turno $j \in \Omega$

Multi-activity shift scheduling problem

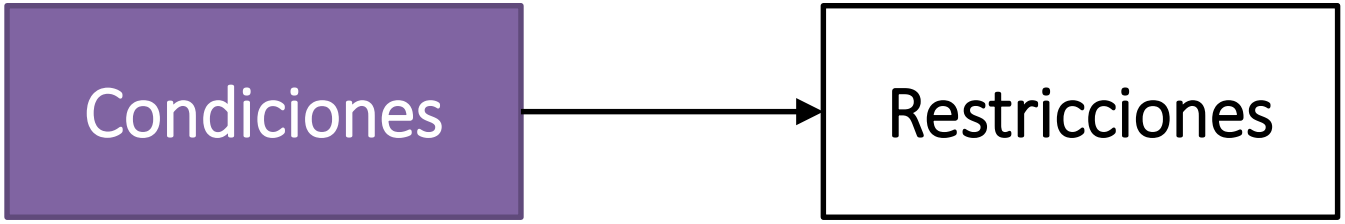
BixiPizza: Restricciones



Restricciones (¿Qué limita la decisión?)

Multi-activity shift scheduling problem

BixiPizza: Restricciones

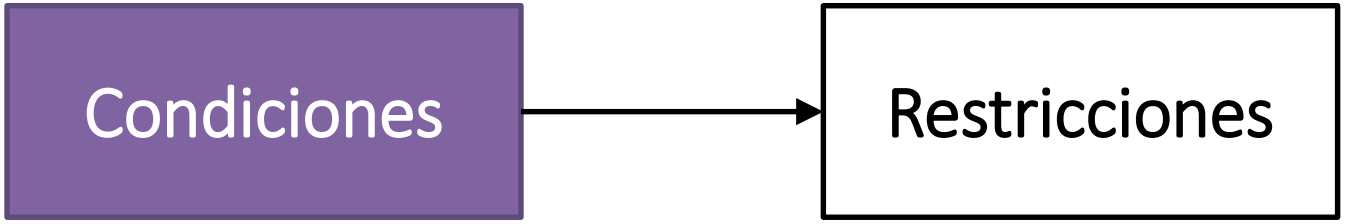


Restricciones (¿Qué limita la decisión?)

- Se debe garantizar la disponibilidad de domiciliarios para cumplir la meta de nivel de servicio.
- No se puede contratar unidades fraccionales de personas.

Multi-activity shift scheduling problem

BixiPizza: Restricciones



Restricciones (¿Qué limita la decisión?)

- Se debe garantizar la disponibilidad de domiciliarios para cumplir la meta de nivel de servicio.

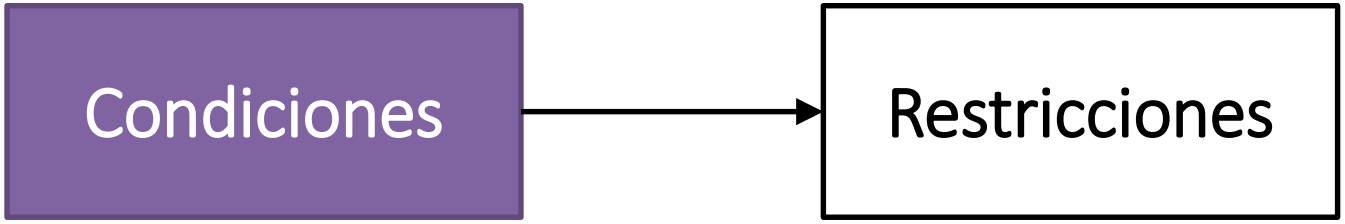
$$\sum_{j \in \Omega} a_{tj} x_j \geq d_t, \quad \forall\, t \in T$$

La restricción debe cumplirse para cada franja horaria

Contabiliza los domiciliarios disponibles en $t \in \mathbf{T}$.

Multi-activity shift scheduling problem

BixiPizza: Restricciones



Restricciones (¿Qué limita la decisión?)

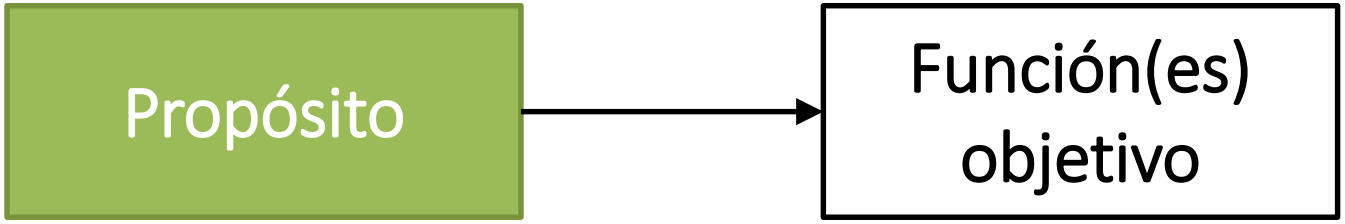
- No se puede contratar fracciones de personas.

Las variables de decisión son enteras no negativas

$$x_j \in \mathbb{Z}_+^1, \forall\, j \in \Omega$$

Multi-activity shift scheduling problem

BixiPizza: Función objetivo



Función Objetivo (¿Cómo cuantificar el impacto de una decisión?)

$$\min z = \sum_{j \in \Omega} c_j x_j$$

Costo de la política de personal

Multi-activity shift scheduling problem

BixiPizza: Formulación del modelo

$$\min z = \sum_{j \in \Omega} c_j x_j$$

$$s.a.$$

$$\sum_{j \in \Omega} a_{tj} x_j \geq d_t \quad \forall\, t \in T$$

$$x_j \in \mathbb{Z}_+^1 \quad \forall\, j \in \Omega$$

Multi-activity shift scheduling problem

BixiPizza: Implementación del modelo (turnos completos)

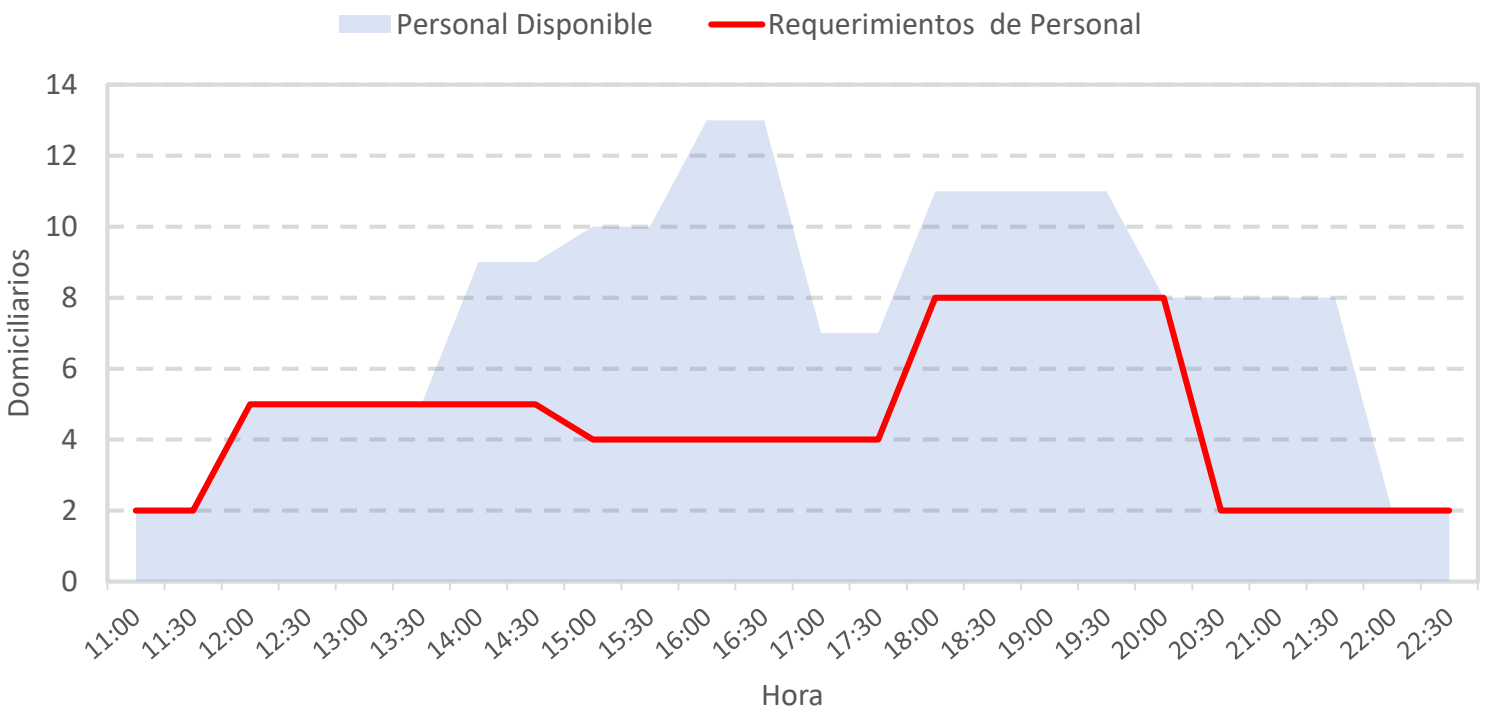
Hora Inicio	Hora Fin	Turno 1	Turno 2	Turno 3	Turno 4	Turno 5	Requerimientos de Personal		Personal Disponible
11:00 a. m.	11:30 a. m.	1	0	0	0	0	2	≤	2
11:30 a. m.	12:00 p. m.	1	0	0	0	0	2		2
12:00 p. m.	12:30 p. m.	1	1	0	0	0	5		5
12:30 p. m.	01:00 p. m.	1	1	0	0	0	5		5
01:00 p. m.	01:30 p. m.	1	1	1	0	0	5		5
01:30 p. m.	02:00 p. m.	1	1	1	0	0	5		5
02:00 p. m.	02:30 p. m.	0	1	1	1	0	5		9
02:30 p. m.	03:00 p. m.	0	1	1	1	0	5		9
03:00 p. m.	03:30 p. m.	1	0	1	1	1	4		10
03:30 p. m.	04:00 p. m.	1	0	1	1	1	4		10
04:00 p. m.	04:30 p. m.	1	1	0	1	1	4		13
04:30 p. m.	05:00 p. m.	1	1	0	1	1	4		13
05:00 p. m.	05:30 p. m.	1	1	1	0	1	4		7
05:30 p. m.	06:00 p. m.	1	1	1	0	1	4		7
06:00 p. m.	06:30 p. m.	1	1	1	1	0	8		11
06:30 p. m.	07:00 p. m.	1	1	1	1	0	8		11
07:00 p. m.	07:30 p. m.	0	1	1	1	1	8		11
07:30 p. m.	08:00 p. m.	0	1	1	1	1	8		11
08:00 p. m.	08:30 p. m.	0	0	1	1	1	8		8
08:30 p. m.	09:00 p. m.	0	0	1	1	1	2		8
09:00 p. m.	09:30 p. m.	0	0	0	1	1	2		8
09:30 p. m.	10:00 p. m.	0	0	0	1	1	2		8
10:00 p. m.	10:30 p. m.	0	0	0	0	1	2		2
10:30 p. m.	11:00 p. m.	0	0	0	0	1	2		2

	Turno 1	Turno 2	Turno 3	Turno 4	Turno 5	Total
Domiciliarios	2	3	0	6	2	13
Duración de turno	8	8	8	8	8	
Costo por hora	7950	7950	7950	7950	7950	
Costo por turno	$ 127,200	$ 190,800	$ -	$ 381,600	$ 127,200	
Costo Total	min $ 826,800					

Multi-activity shift scheduling problem

BixiPizza: Solución con domiciliarios tiempo completo

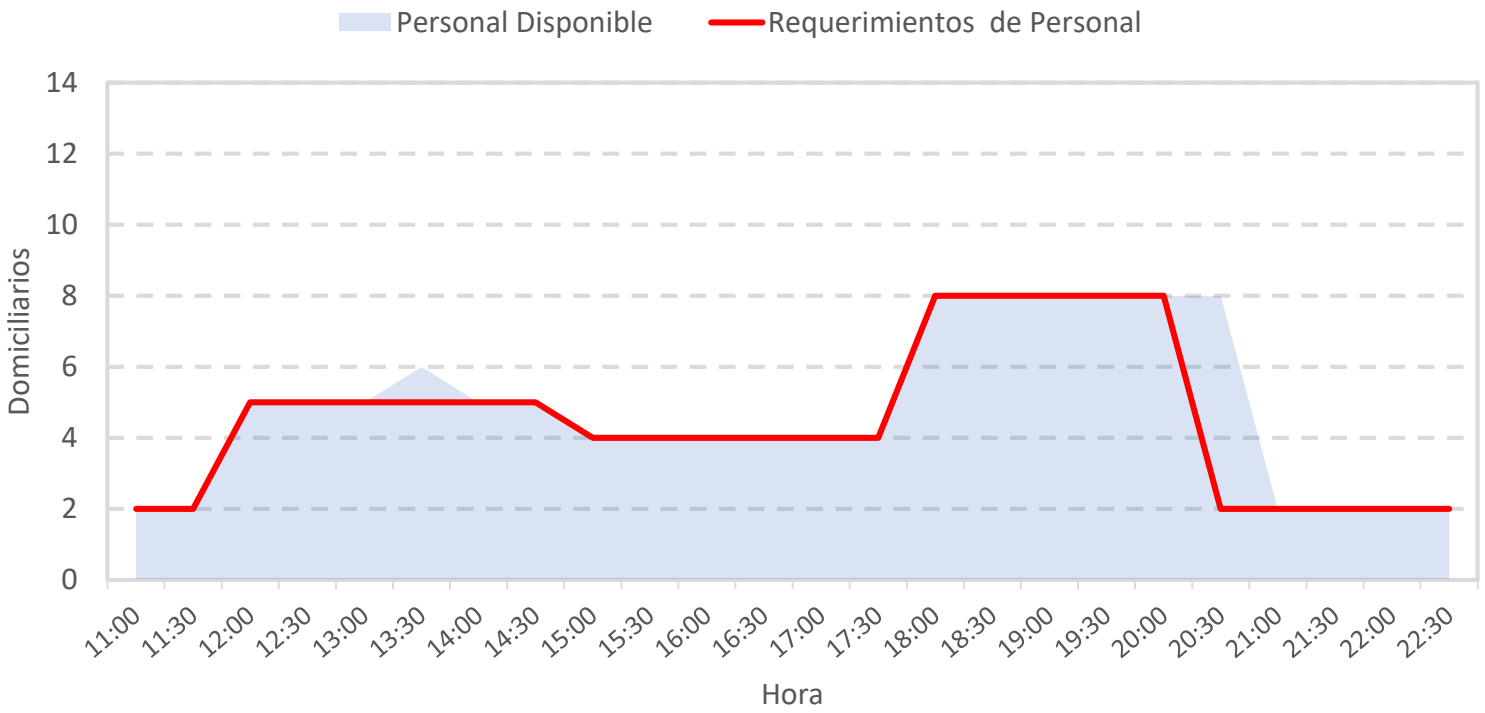
Costo total: $826,800



Multi-activity shift scheduling problem

BixiPizza: Solución con turnos tiempo completo, parcial y subcontratado

Costo total: $600,660



Multi-activity shift scheduling problem

BixiPizza: ¿Cómo lo resolvemos si hay muchos turnos posibles?

$$\min z = \sum_{j \in \Omega} c_j x_j$$

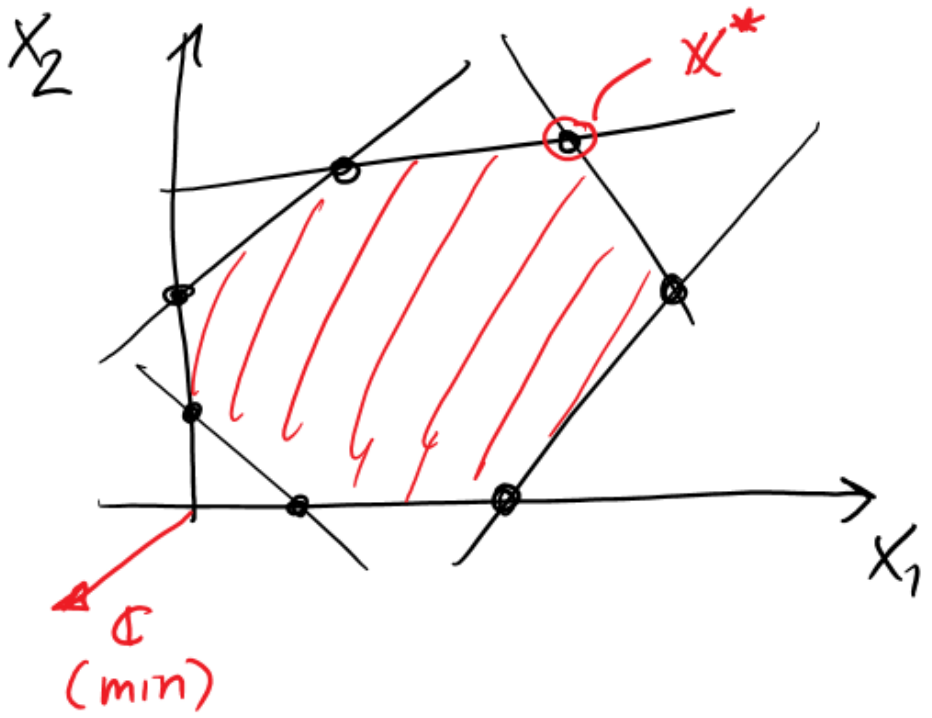
$$s.a.$$

$$\sum_{j \in \Omega} a_{tj} x_j \geq d_t \quad \forall\, t \in T$$

$$x_j \in \mathbb{Z}_+^1 \quad \forall\, j \in \Omega$$

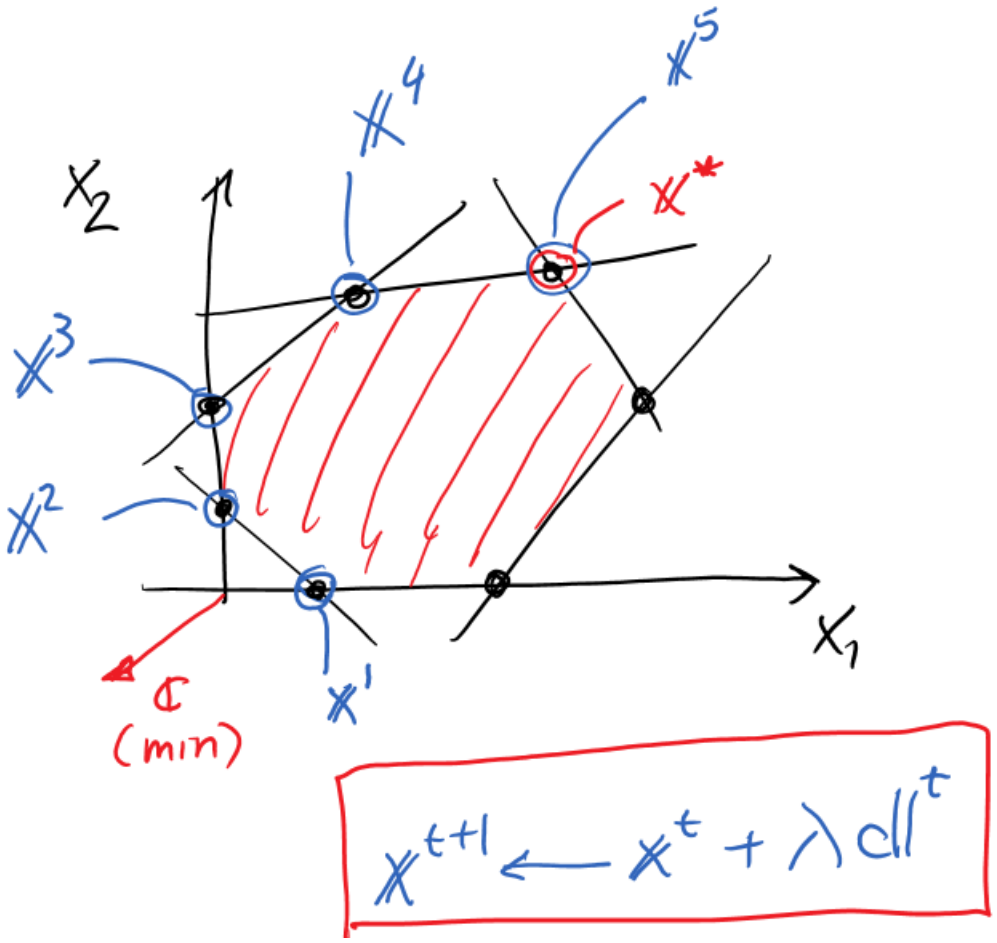
Multi-activity shift scheduling problem

BixiPizza: ¿Cómo lo resolvemos?



Multi-activity shift scheduling problem

BixiPizza: ¿Cómo lo resolvemos?



Multi-activity shift scheduling problem

BixiPizza: ¿Cómo lo resolvemos?

$$\min z = \sum_{j \in \Omega} c_j x_j$$

$$s.a.$$

$$\sum_{j \in \Omega} a_{tj} x_j \geq d_t \quad \forall\, t \in T$$

$$x_j \in \mathbb{Z}^1_+ \quad \forall\, j \in \Omega$$

$$\min \; c^T x$$

$$s.a.$$

$$Ax = b$$

$$x \geq \vec{0}$$

$$\min z = c^T x$$

$$s.a.$$

$$Ax - \vec{s} = b$$

$$x \geq \vec{0}$$

$$s \geq \vec{0}$$

$$c^T = [c_1 \cdots c_j \cdots c_n]$$

$$x^T = [x_1 \cdots x_j \cdots x_n]$$

$$A = [\vec{a}_1 | \cdots | \vec{a}_j | \cdots | \vec{a}_n]$$

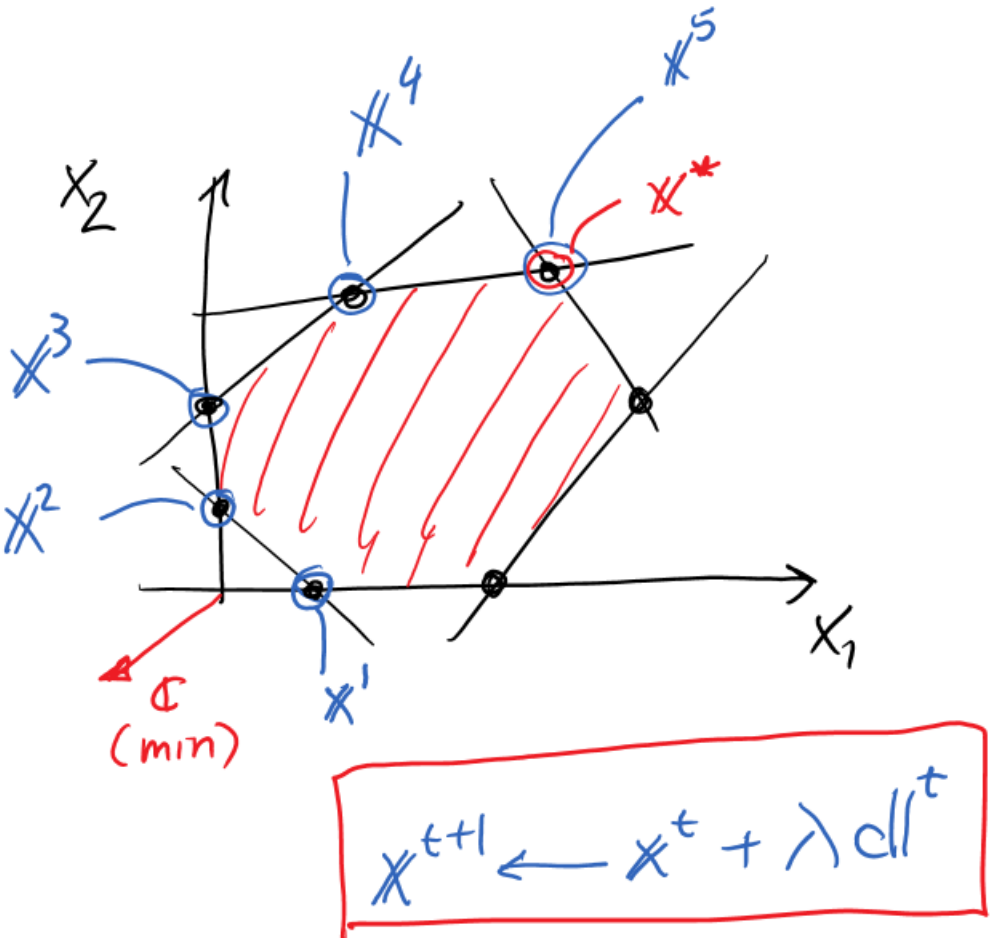
$$[a_{tj}] \rightarrow \{0,1\}$$

$$b = [d_t]$$

$$x \in \mathbb{Z}^n_+ \rightarrow x \geq \vec{0}$$

Multi-activity shift scheduling problem

BixiPizza: ¿Cómo lo resolvemos?

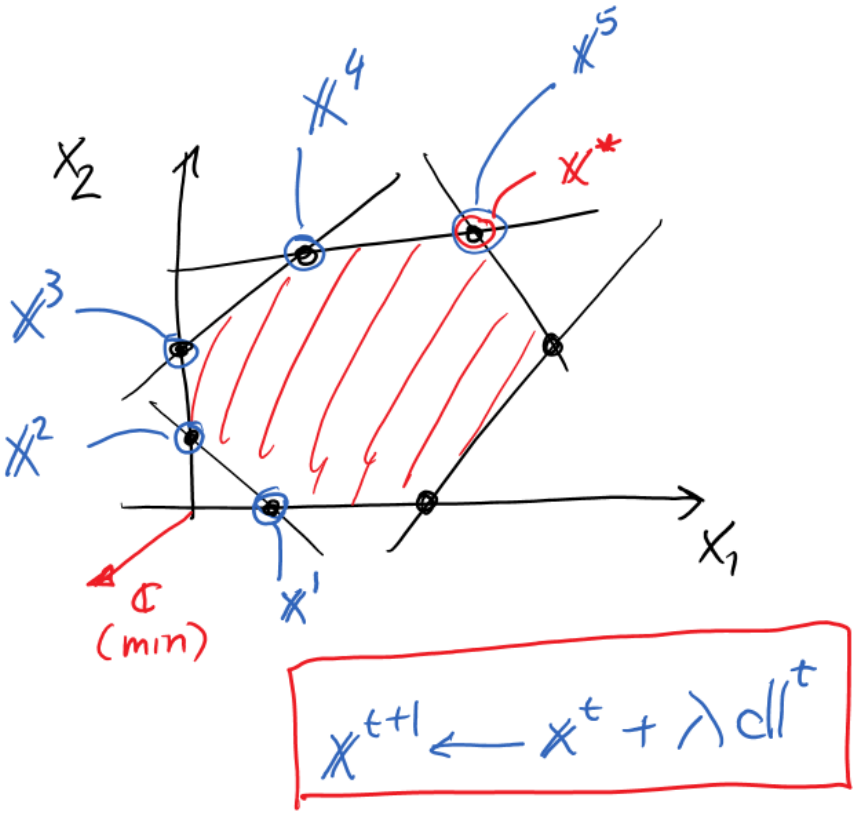


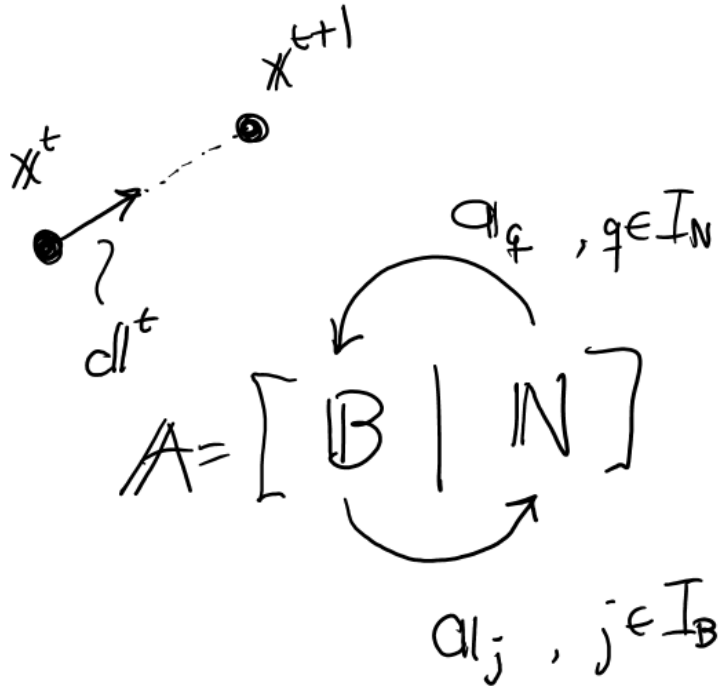
$$A = [B \mid N], \quad I_B, I_N$$

$$x = \begin{bmatrix} x_B \\ \hline x_N \end{bmatrix} = \begin{bmatrix} B^{-1}b \\ \hline \vec{0} \end{bmatrix}$$

Multi-activity shift scheduling problem

BixiPizza: ¿Cómo lo resolvemos?





Multi-activity shift scheduling problem

BixiPizza: ¿Cómo lo resolvemos?

Columna entrante:

$x_q = 0 \rightarrow x_q = 1$

$\bar{C}_q = C_q - C_B^T B^{-1} a_q \quad , \quad q \in I_N$

- Si $\bar{C}_q < 0 \Rightarrow a_q$ es una buena columna
- Si $\bar{C}_q \geq 0, \forall q \in I_N \Rightarrow$ no existe una buena columna

Multi-activity shift scheduling problem

BixiPizza: ¿Cómo lo resolvemos?

¿Qué pasa si no conocemos todas las columnas de forma explícita?

$$\min \bar{C}_q$$

s.a.,

a_q sea una columna factible para mi problema

$$\min \; C_q - \underbrace{C_B^T B^{-1}}_{\pi^T} a_q$$

s.a.,

a_q factible

Subproblema en generación de columnas

Multi-activity shift scheduling problem

BixiPizza: ¿Cómo lo resolvemos?

$$\min \; c_q - \pi_B^T B^{-1} a_{1q}$$

$$\text{s.a.,} \quad a_{1q} \text{ factible}$$

Subproblema en generación de columnas

Tiempo completo

- **Duración:** 8 horas
- **Descanso:** 1 hora
  - Descanso programado en la 4ta hora.
- **Inicio:** Horas en punto
- **Costo/hora:**
  - $ 7,950 pesos

Hora	Turno 1	Turno 2	Turno 3	Turno 4	Turno 5
11:00-11:30					
11:30-12:00					
12:00-12:30					
12:30-13:00					
13:00-13:30					
13:30-14:00					
14:00-14:30					
14:30-15:00					
15:00-15:30					
15:30-16:00					
16:00-16:30					
16:30-17:00					
17:00-17:30					
17:30-18:00					
18:00-18:30					
18:30-19:00					
19:00-19:30					
19:30-20:00					
20:00-20:30					
20:30-21:00					
21:00-21:30					
21:30-22:00					
22:00-22:30					
22:30-23:00					

	Disponible
	No disponible

$$\vec{a}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Multi-activity shift scheduling problem

Case study: A parking lot operator

- Restrepo, Lozano & Medaglia (2012)

Multi-activity shift scheduling problem

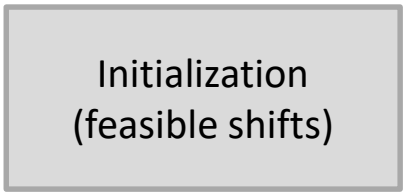
Case study: A parking lot operator

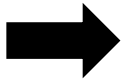
Universidad de los Andes Colombia

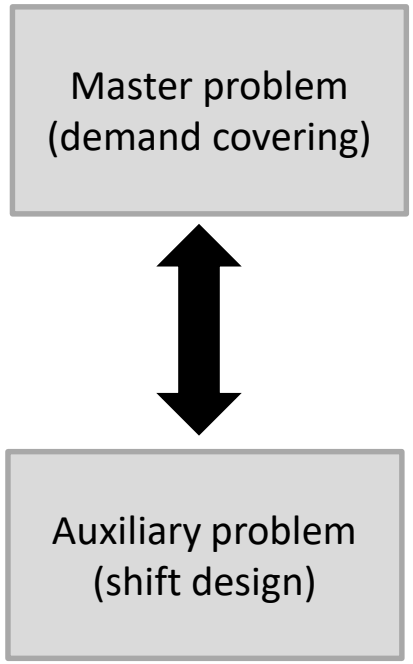
- One of the largest parking operators in Bogotá (Colombia).
- The company has more than 400 employees working at 120 parking lots.
- The parking lots are grouped by zones according to their proximity.
- The company wants a system to schedule their security staff and cashiers.
- The current staff scheduling policy (baseline) does not allow staff movements and is planned over a week and updated every four months.
- One of the goals is to evaluate the impact of a new policy that allows staff movements between parking lots.

Multi-activity shift scheduling problem

Column generation approach for MASSP





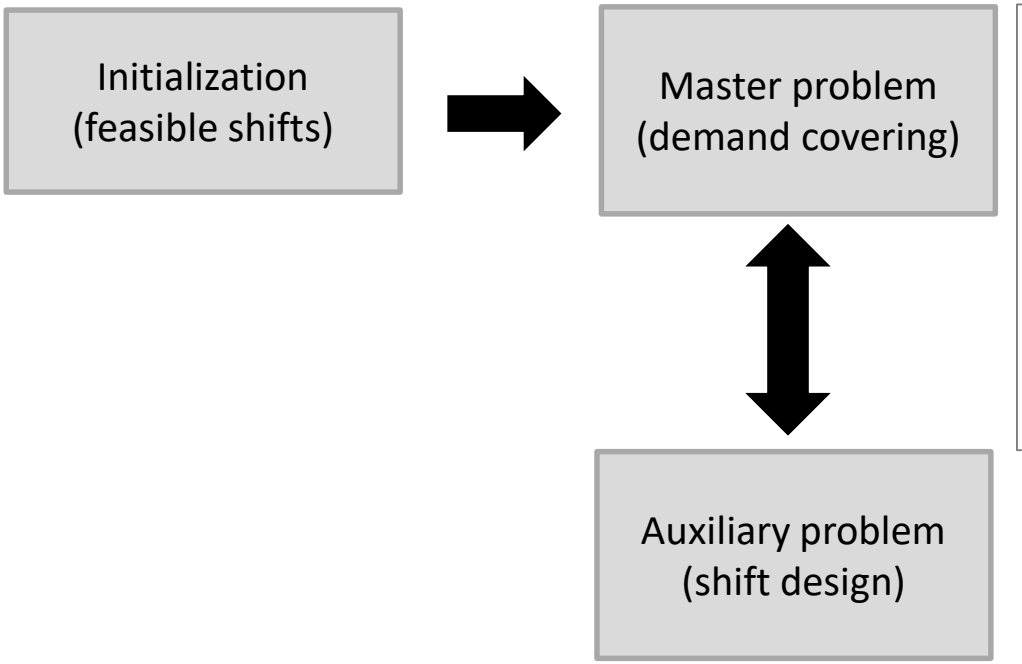


WORK RULES

1. Shift duration
2. Minimum shift duration per lot
3. Minimum shift duration per day
4. Maximum number of parking lots (per shift)
5. Maximum number of days (per shift)
6. Number of breaks
7. Travel distances between lots
8. Consecutive time intervals
9. Consecutive days

Multi-activity shift scheduling problem

Column generation for shift scheduling



WORK RULES

1. Shift duration
2. Minimum shift duration per lot
3. Minimum shift duration per day
4. Maximum number of parking lots (per shift)
5. Maximum number of days (per shift)
6. Number of breaks
7. Travel distances between lots
8. Consecutive time intervals
9. Consecutive days

Multi-activity shift scheduling problem

Column generation: master problem

$$\min \quad Z_{RMP(\Omega')} = \sum_{j \in \Omega'} c_j x_j$$

subject to

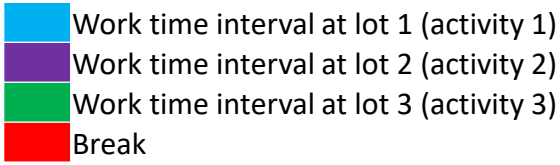
$$\sum_{j \in \Omega'} a_{tlj} x_j \geq d_{tl}; \quad t \in \mathcal{T},\ l \in \mathcal{K} \longrightarrow \pi_{tl}$$

$$x_j \geq 0, \quad j \in \Omega'$$

Multi-activity shift scheduling problem

Column generation: auxiliary problem

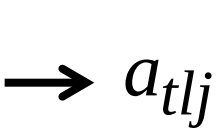
Time interval	Activity (lot)
7:00-8:00	2
8:00-9:00	2
9:00-10:00	-
10:00-11:00	1
11:00-12:00	-
12:00-13:00	1
13:00-14:00	3
14:00-15:00	3
15:00-16:00	3



Multi-activity shift scheduling problem

Column generation: auxiliary problem

Time interval	Activity (lot)
7:00-8:00	2
8:00-9:00	2
9:00-10:00	-
10:00-11:00	1
11:00-12:00	-
12:00-13:00	1
13:00-14:00	3
14:00-15:00	3
15:00-16:00	3

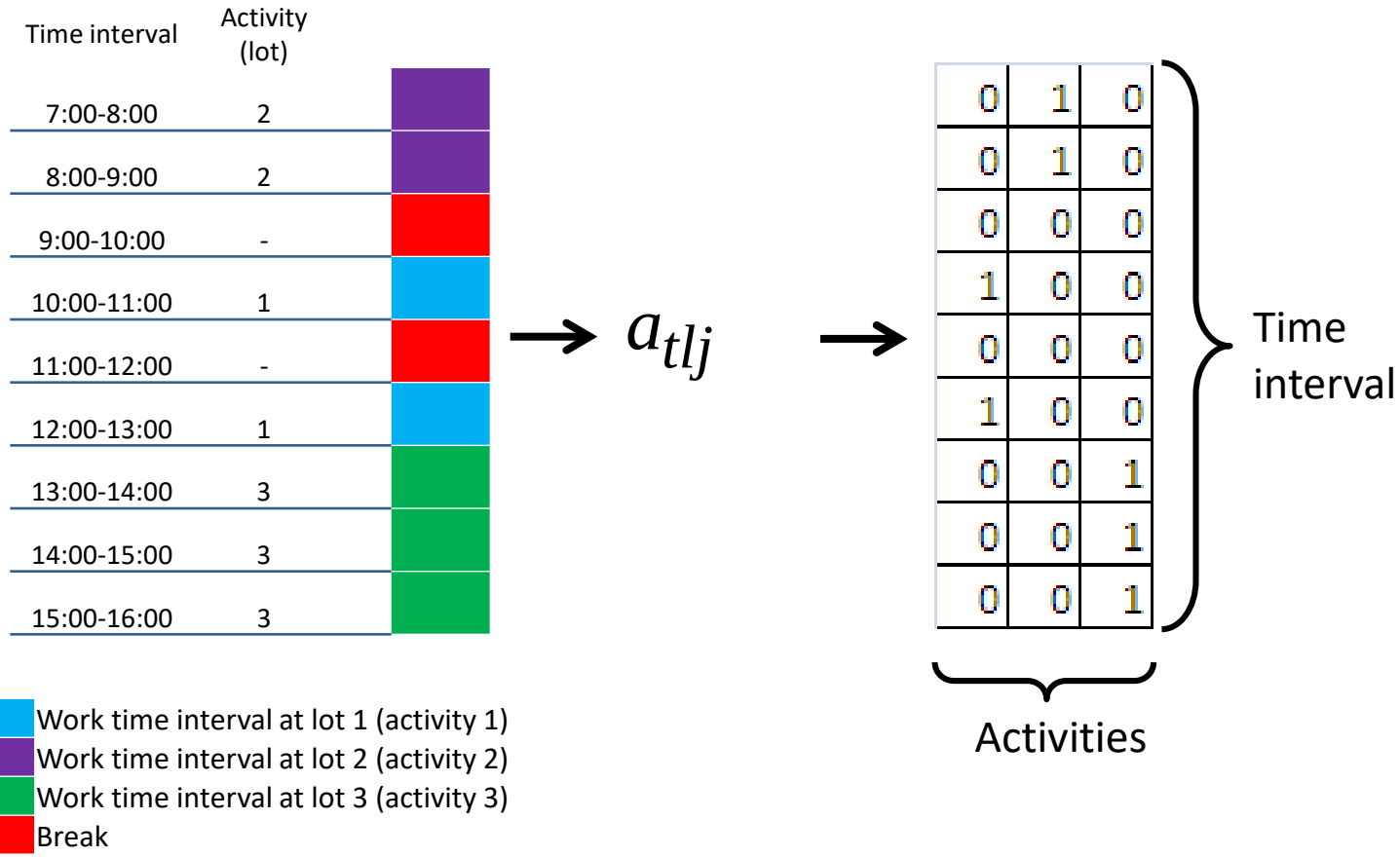


a_{tlj}

Work time interval at lot 1 (activity 1)
Work time interval at lot 2 (activity 2)
Work time interval at lot 3 (activity 3)
Break

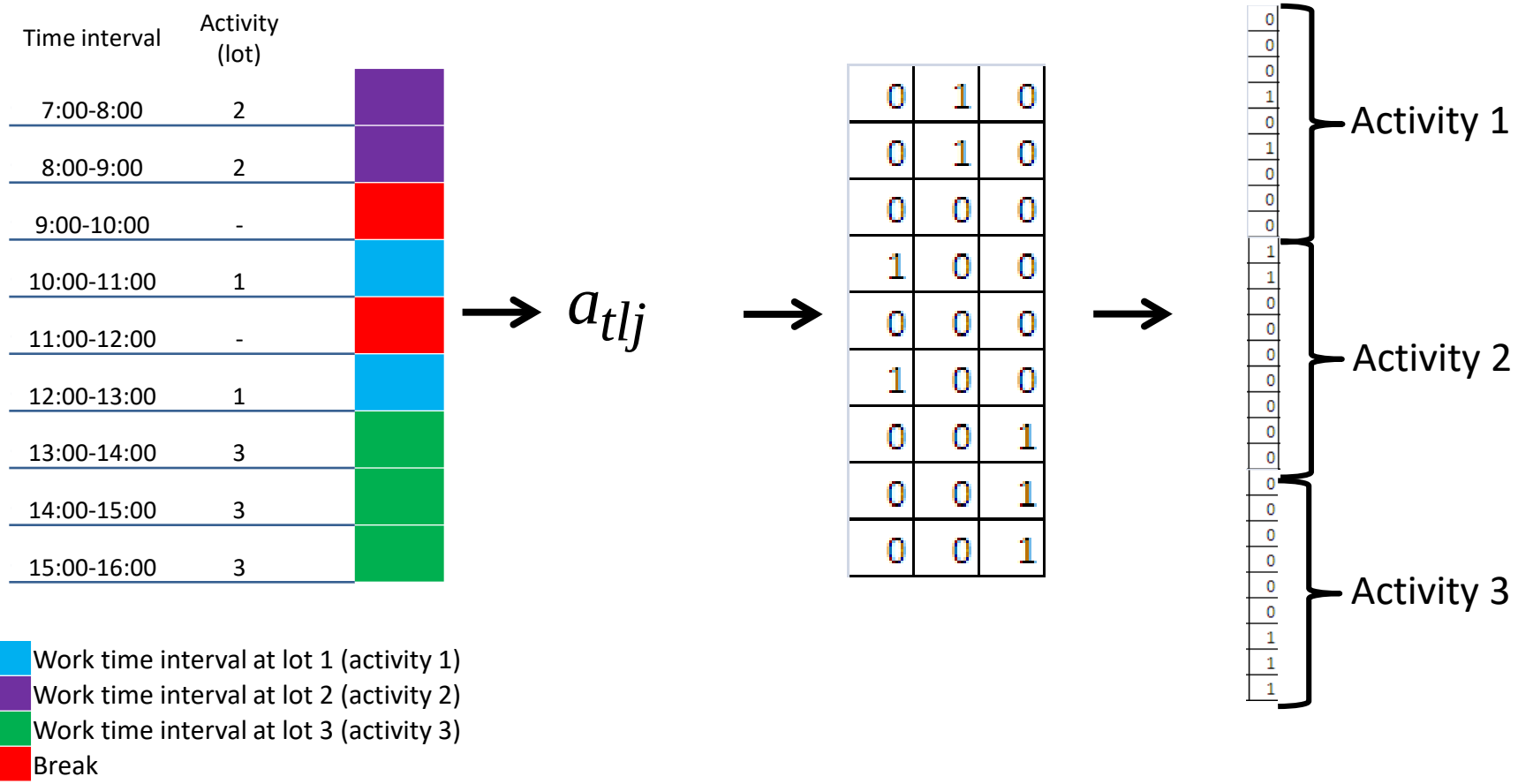
Multi-activity shift scheduling problem

Column generation: auxiliary problem



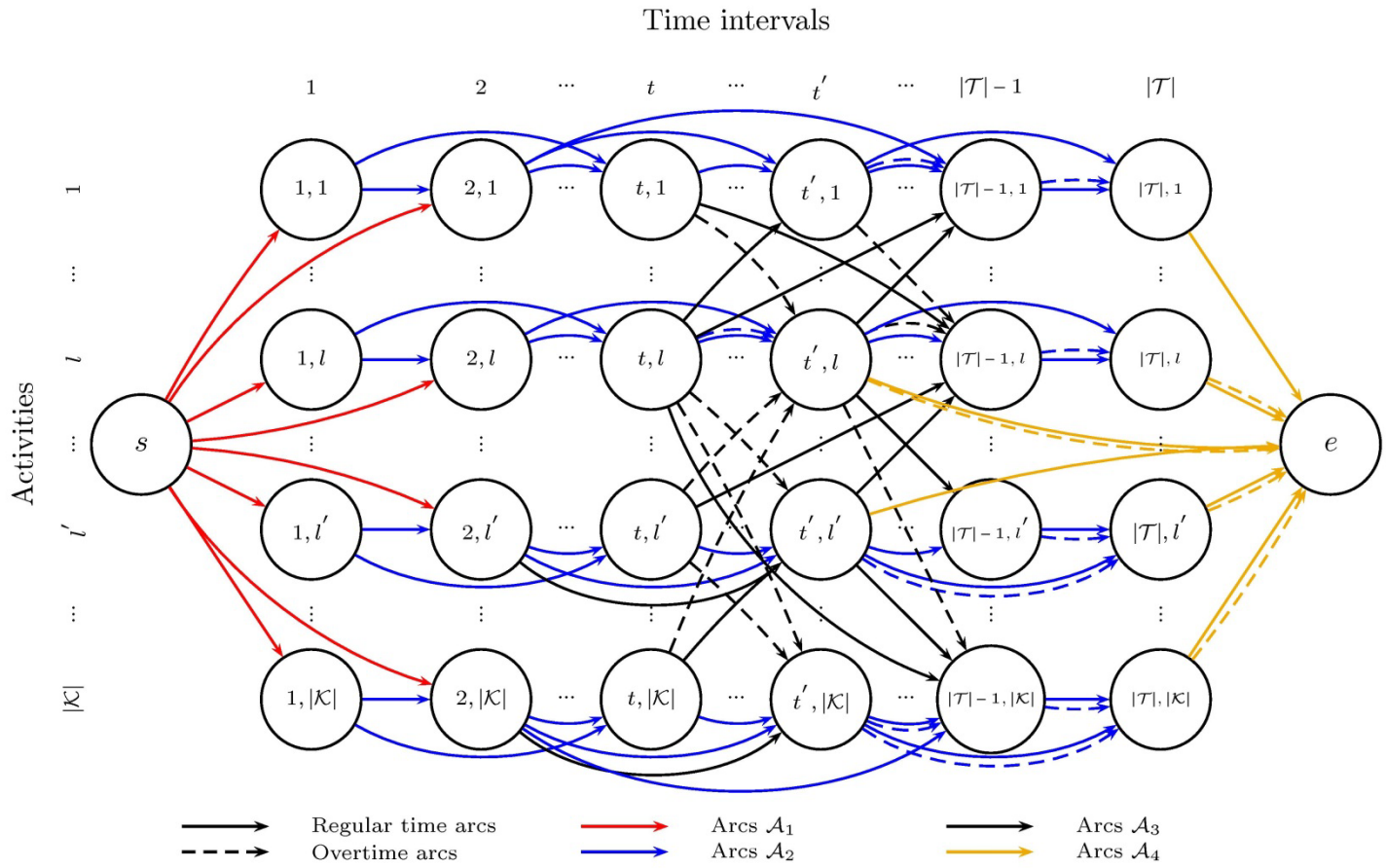
Multi-activity shift scheduling problem

Column generation: auxiliary problem



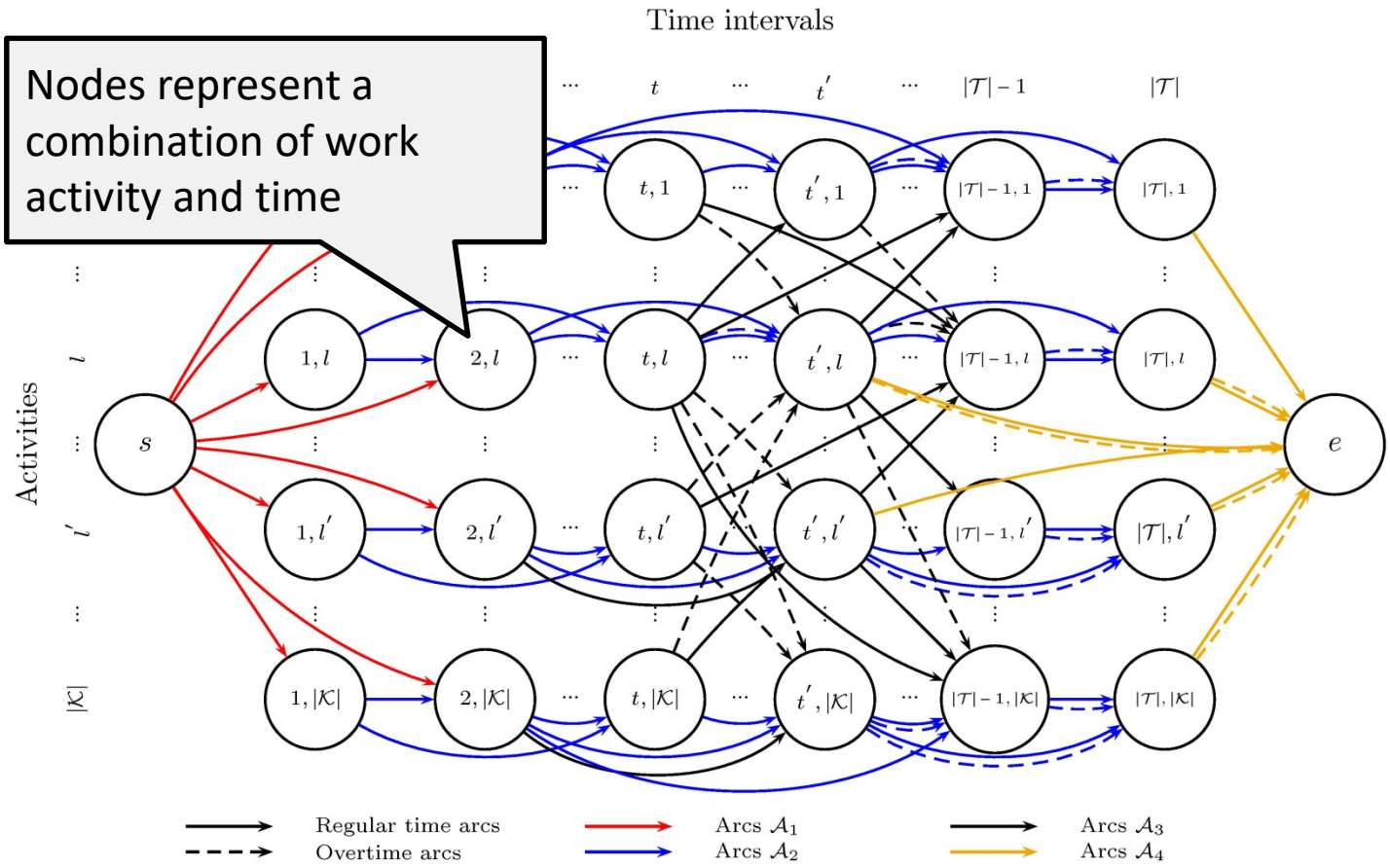
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



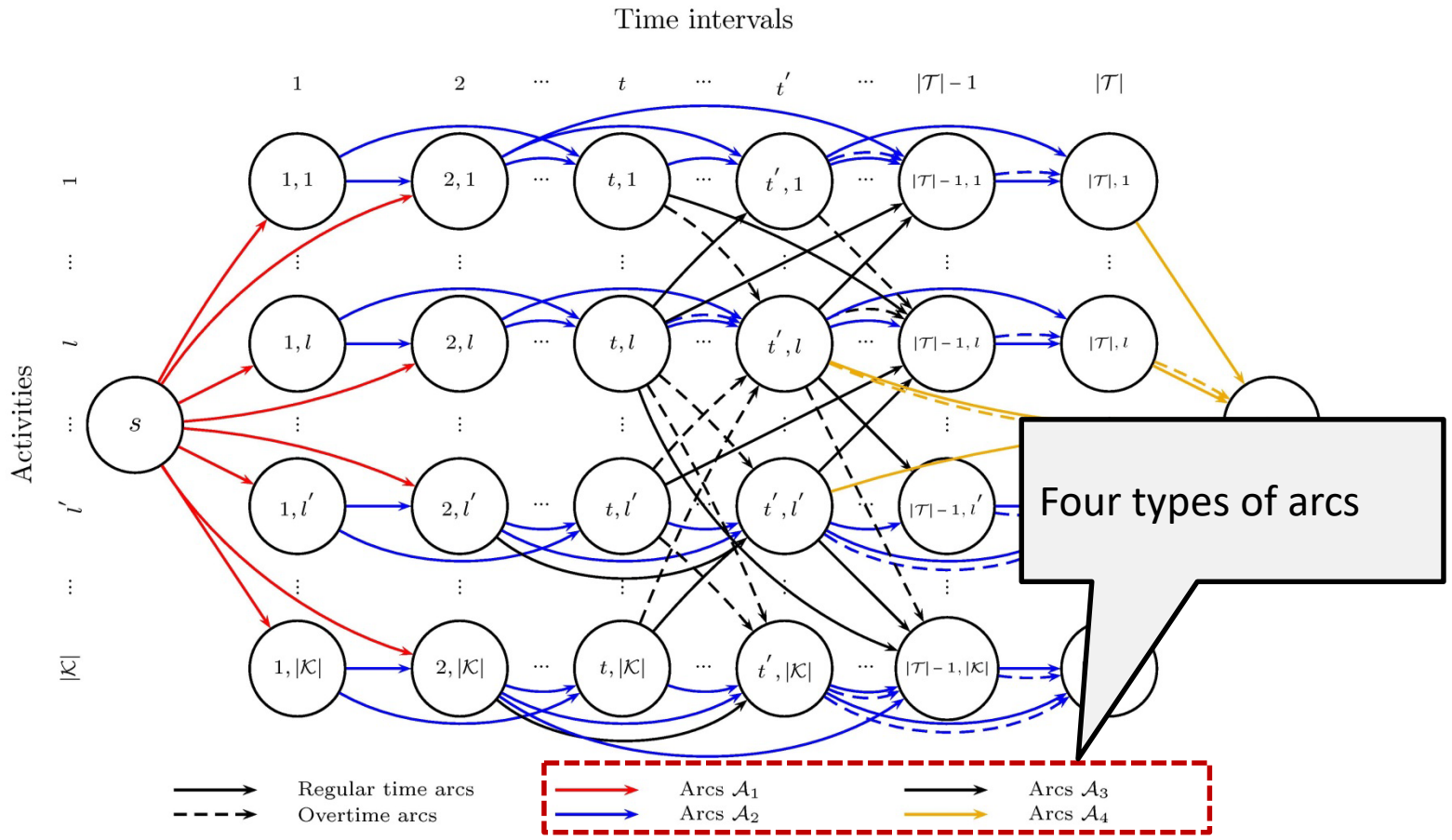
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



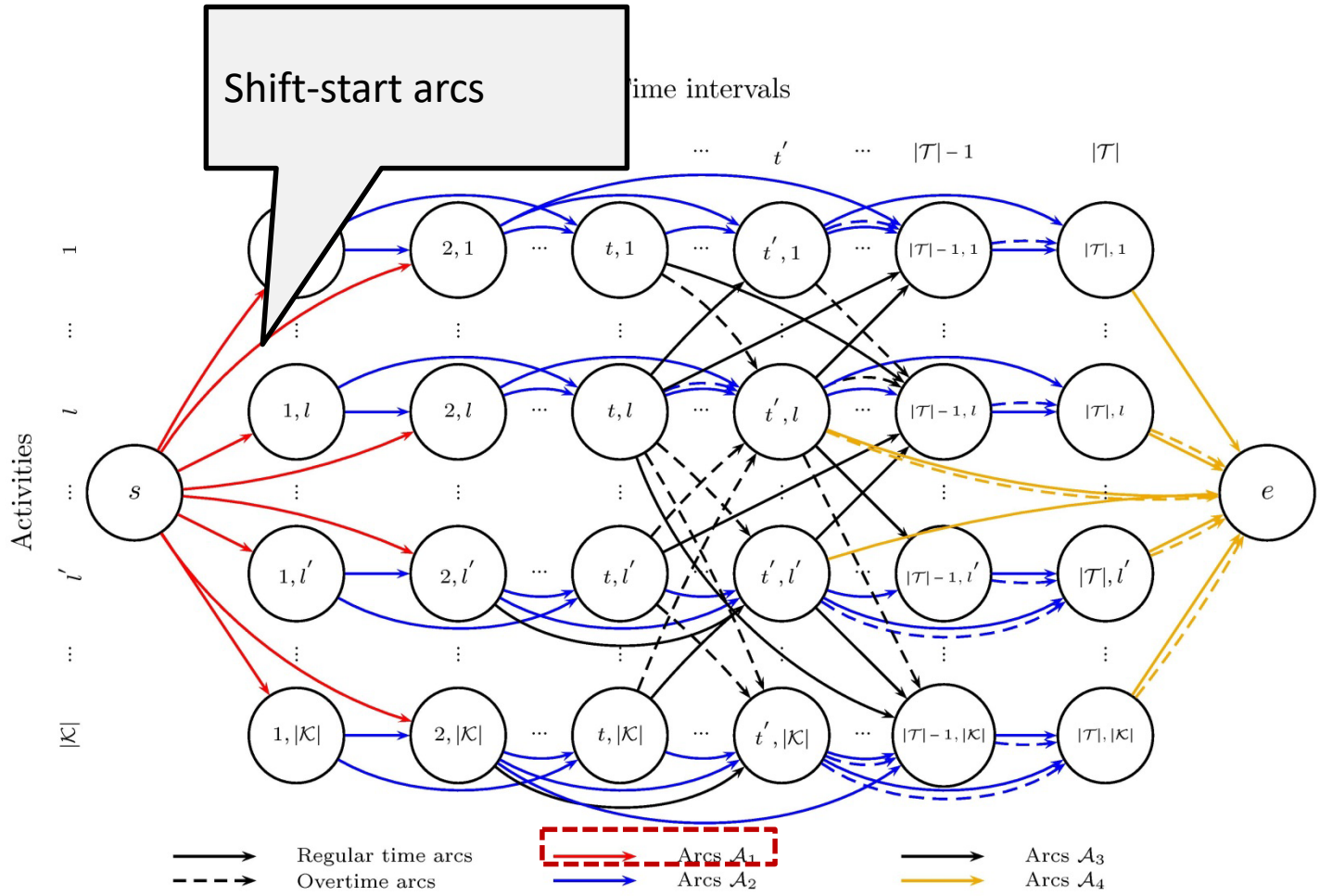
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



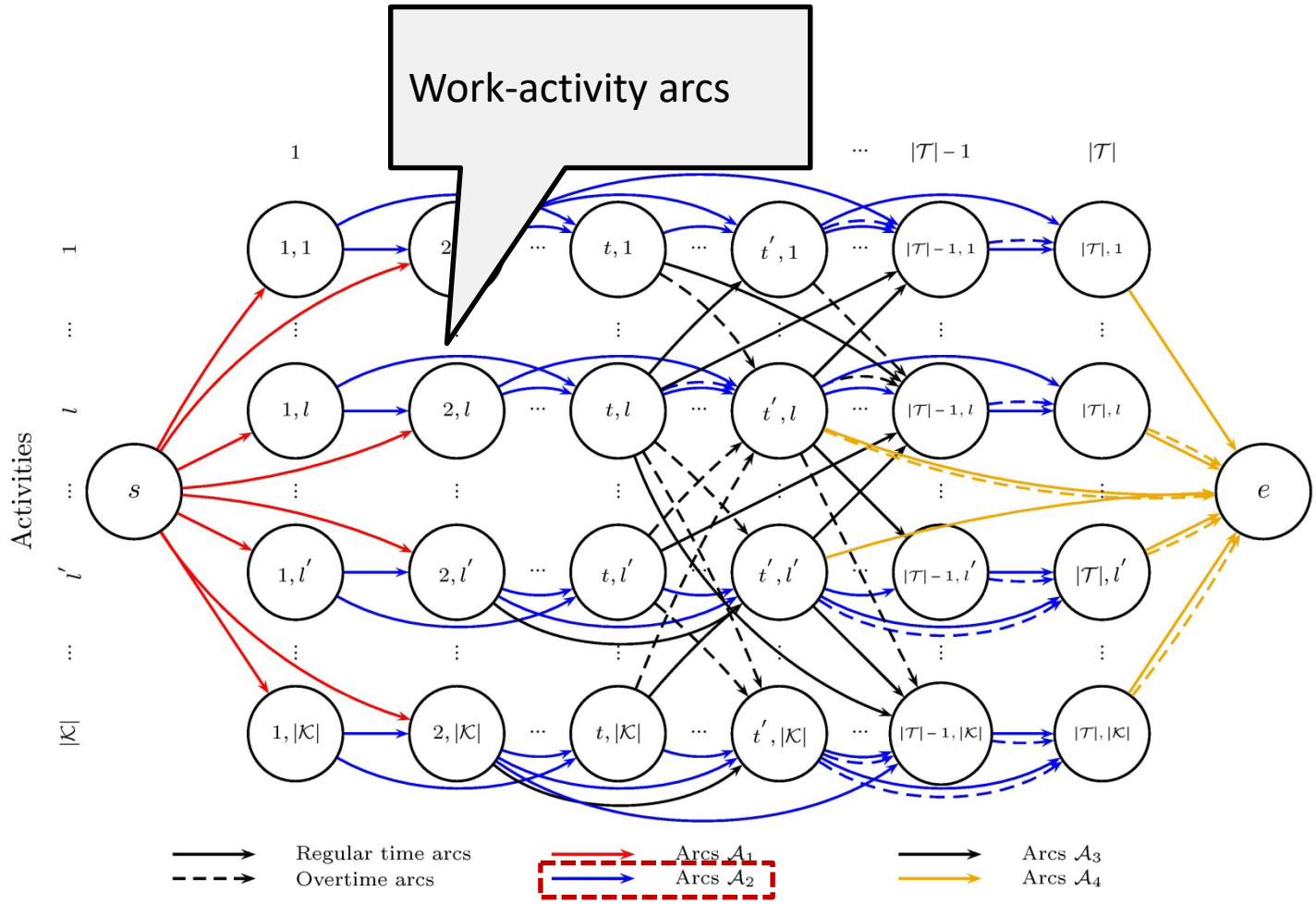
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



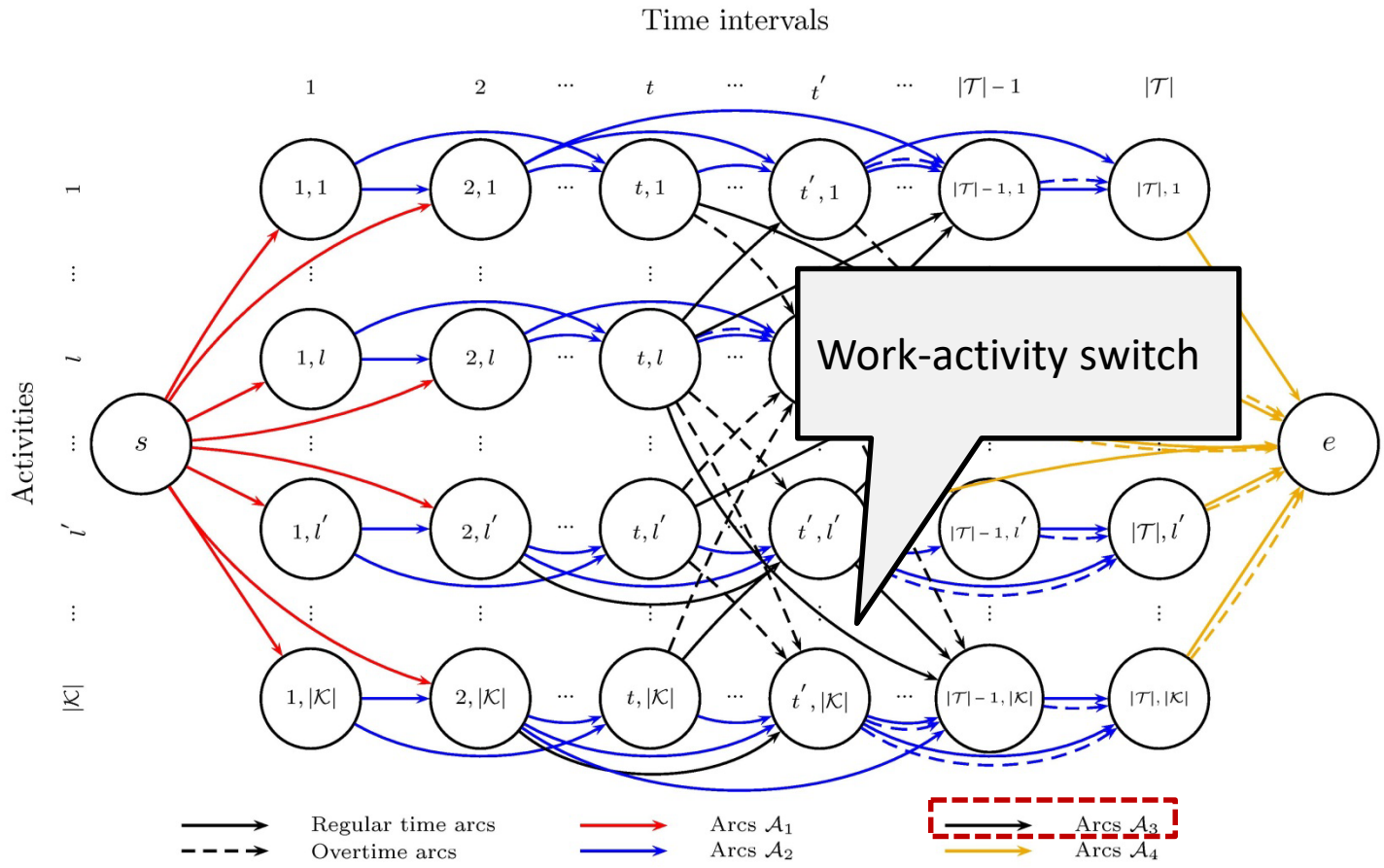
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



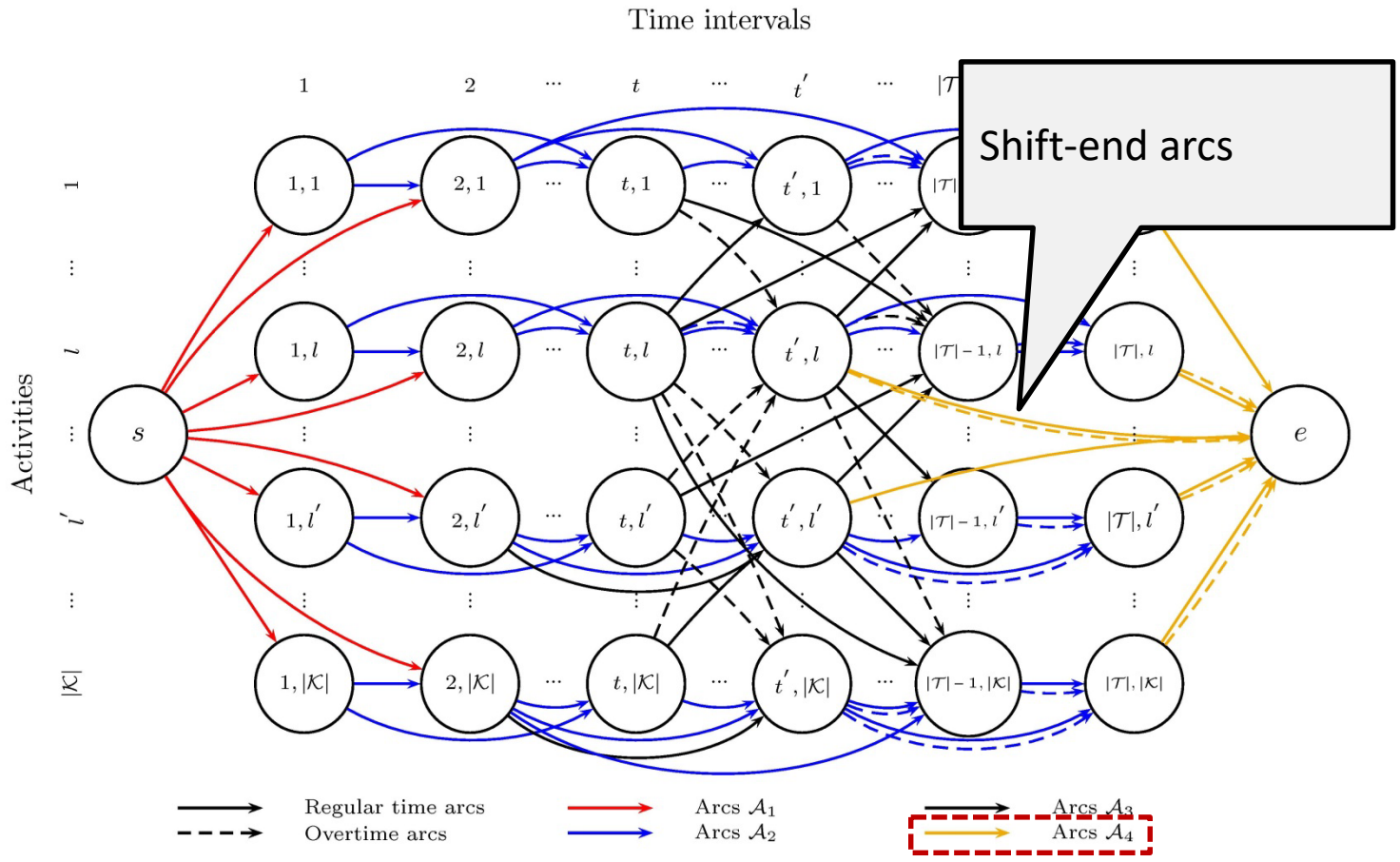
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



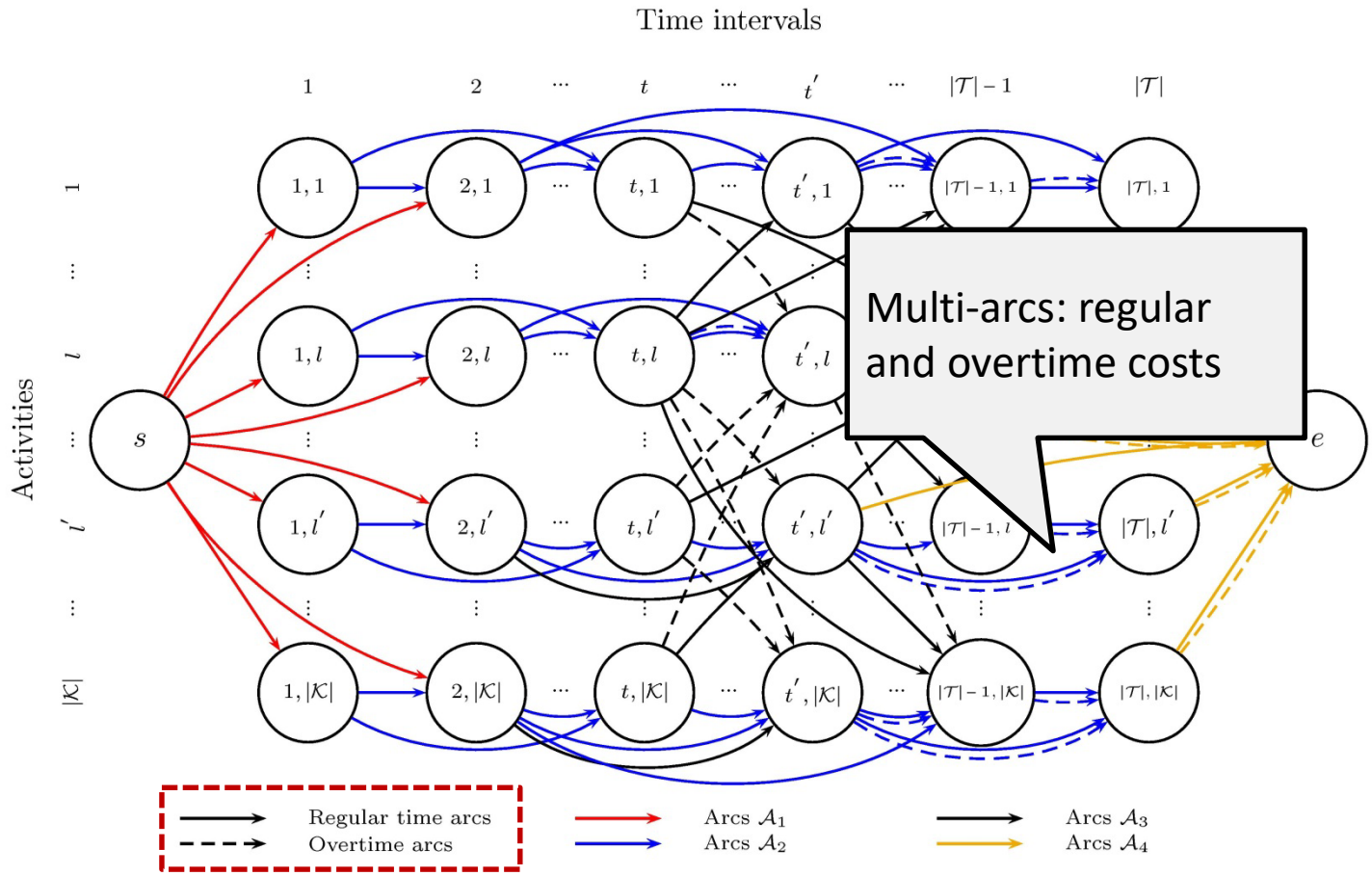
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



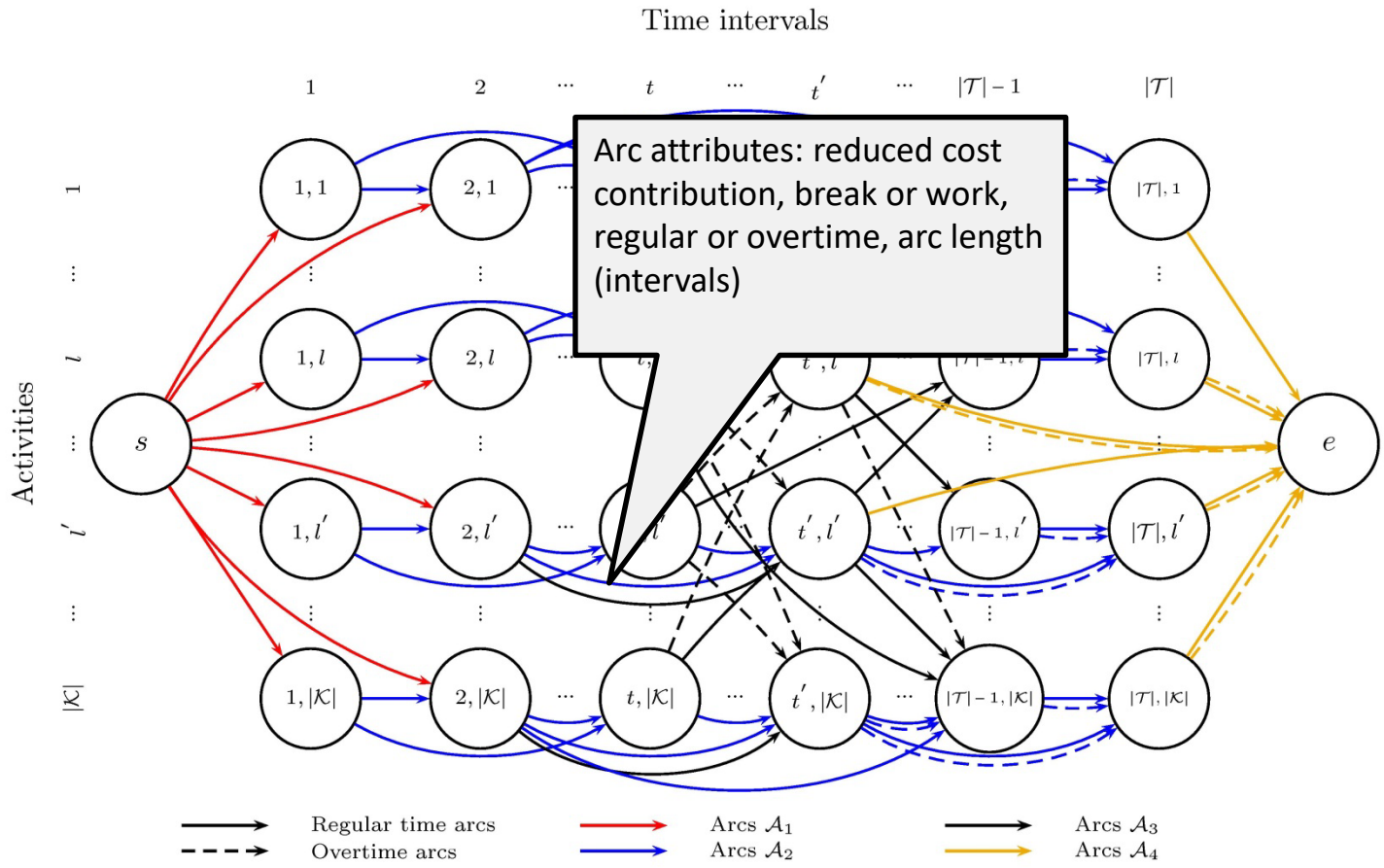
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



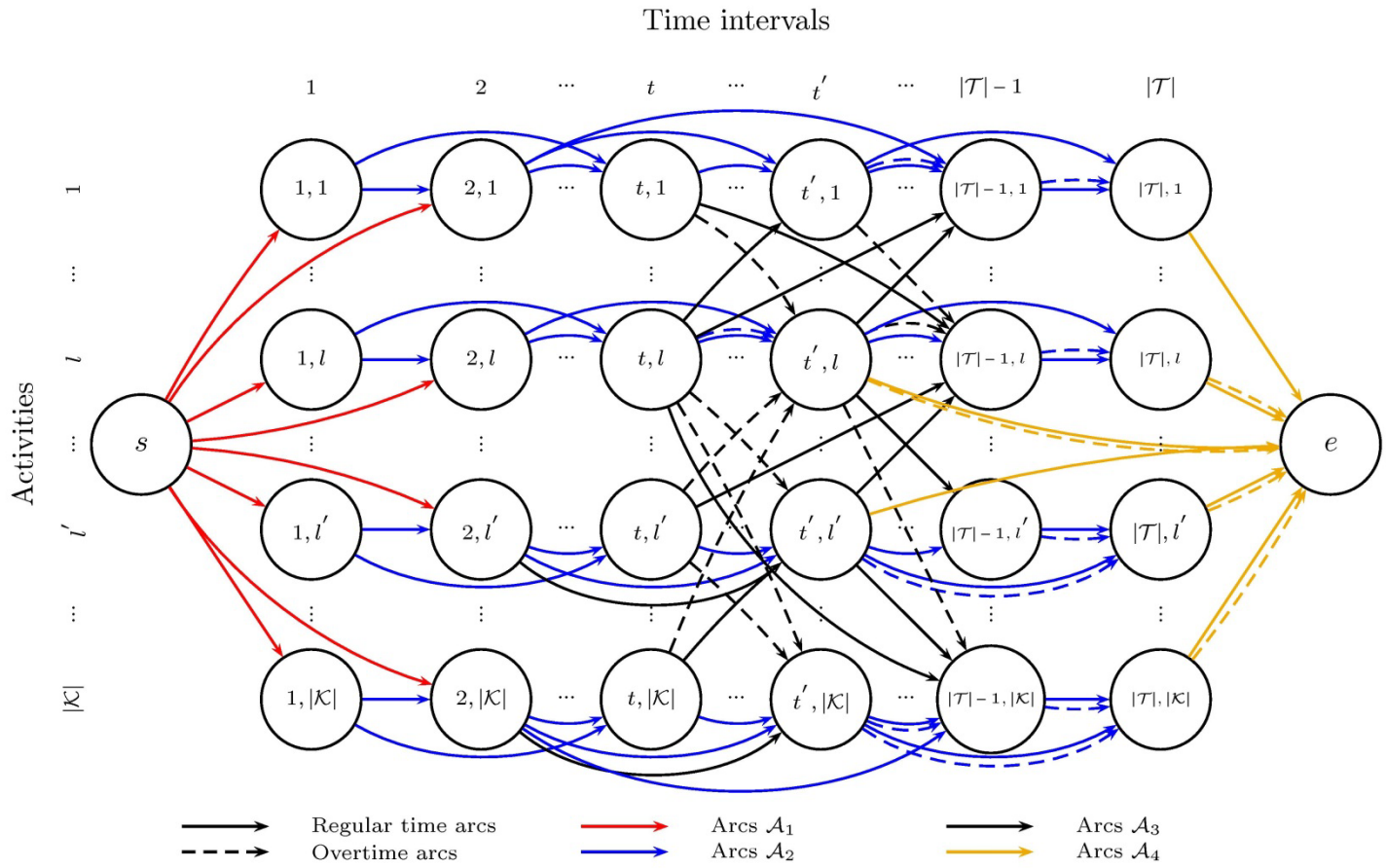
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



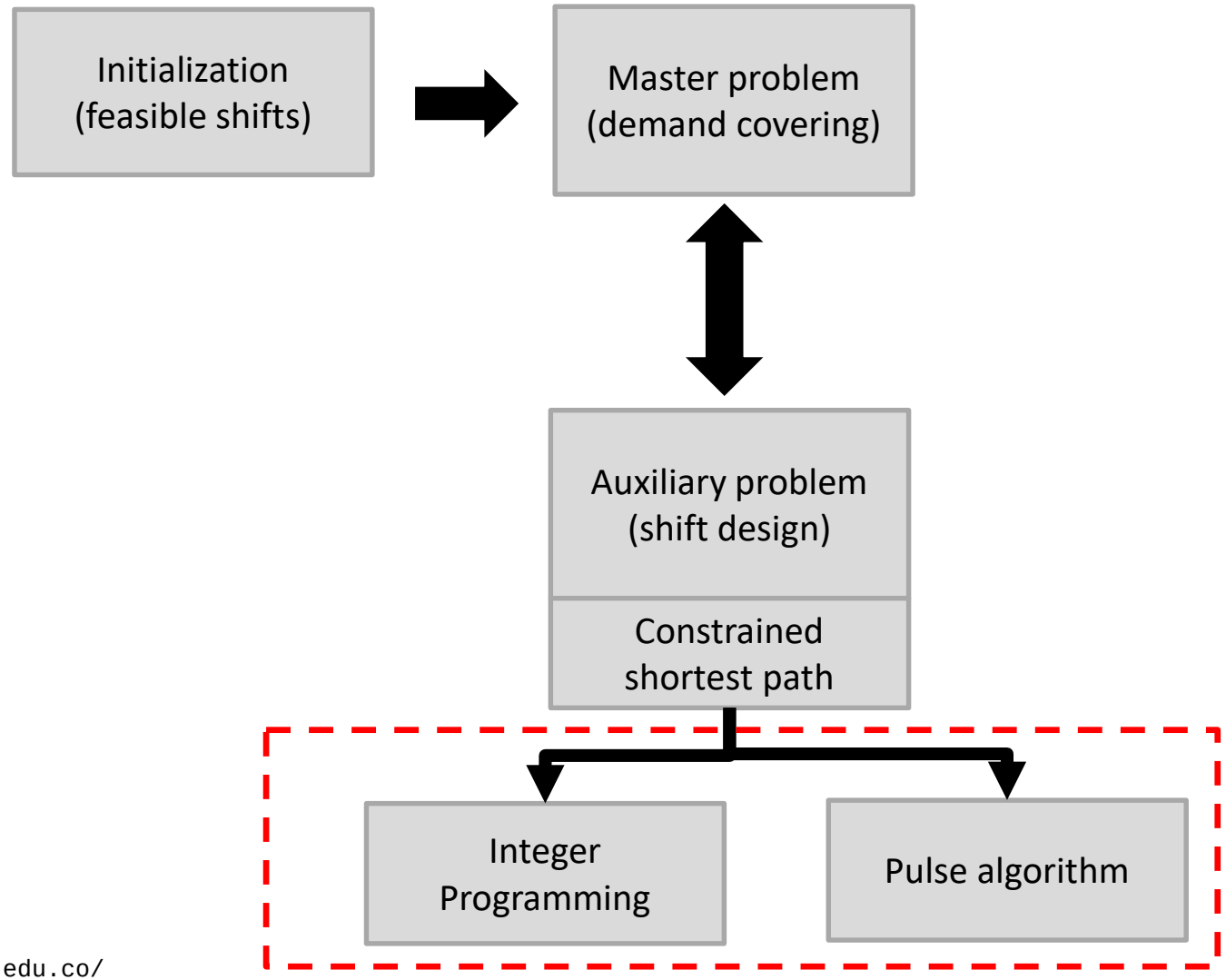
Multi-activity shift scheduling problem

Auxiliary problem: a network-oriented formulation



Multi-activity shift scheduling problem

Column generation approach for MASSP



Multi-activity shift scheduling problem

Computational experiments: IP vs. pulse

- The auxiliary problem is a variant of the Shortest Path Problem with Resource Constraints (SPPRC).
- Defined over an acyclic graph with negative costs, several problem-specific side constraints and up to 2,000 nodes and 50,000 arcs.
- Modifications (over the pulse for CSP)
  - Acyclic graph (no cycle pruning)
  - Dominance pruning (multiple resources)
  - Infeasibility extended to multiple resource constraints

Lozano & Medaglia (2013):

Table 4
Speeding up the solution of real-world MASSP instances with the pulse algorithm.

Instance	Time (s)	
	IP[a]	Pulse
A5-TIL30-D7	3549	17
A7-TIL30-D7	71,216	83
A16-TIL30-D7	848,402	629

[a] Xpress-MP optimizer version 19.00.00.

Multi-activity shift scheduling problem

Computational experiments: case study

Universidad de los Andes Colombia

Table 1
Computational performance on instances with a one-day planning horizon.

Instance	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{A}\rvert$	LR-RMP (s)	IP-RMP (s)	Gap (%)	Iter.	Columns
A5-TIL30-D1-V1	242	3087	2.39	2.41	0	123	123
A5-TIL30-D1-V2	242	2247	0.42	0.44	0	17	17
A7-TIL30-D1-V1	338	5047	16.61	16.67	0	388	388
A7-TIL30-D1-V2	338	3815	2.81	2.88	0	96	96
A13-TIL30-D1-V1	626	14,704	404.47	404.70	0	1909	1909
A13-TIL30-D1-V2	626	12,359	38.52	38.61	0	396	396
A16-TIL30-D1-V1	770	12,516	131.56	131.70	0	997	997
A16-TIL30-D1-V2	770	10,205	13.53	13.59	0	200	200
A21-TIL30-D1-V1	1010	28,447	3650.10	3654.83	0	3776	3776
A21-TIL30-D1-V2	1010	24,639	200.42	202.31	0	727	727

Table 2
Computational performance on instances with a one-week planning horizon.

Instance	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{A}\rvert$	LR-RMP (s)	IP-RMP (s)	Gap (%)	Iter.	Columns
A5-TIL30-D7-V1	482	6589	2629.15	2634.43	0.01	1517	10,619
A5-TIL30-D7-V2	482	4270	16.54	17.05	0.00	81	567
A7-TIL30-D7-V1	674	5180	4471.20	4476.43	0.02	1812	12,684
A7-TIL30-D7-V2	674	3661	81.90	83.08	0.00	154	1078
A13-TIL30-D7-V1[a]	1250	21,542	28,807.95	28,878.46	–	2120	14,840
A13-TIL30-D7-V2	1250	17,293	1749.50	1755.79	0.00	605	4235
A16-TIL30-D7-V1[a]	1538	19,784	28,805.60	28,861.84	–	3199	22,393
A16-TIL30-D7-V2	1538	15,507	624.12	628.29	0.21	363	2541

[a]8-h limit reached solving the LR-RMP.

Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
  - Multi-activity shift scheduling problem
  - **Bus rapid transit route design problem**
  - Interdiction problem with fortification
  - Green VRP
  - Other applications
- Part V: perspectives

Bus Rapid Transit Systems (BRT)

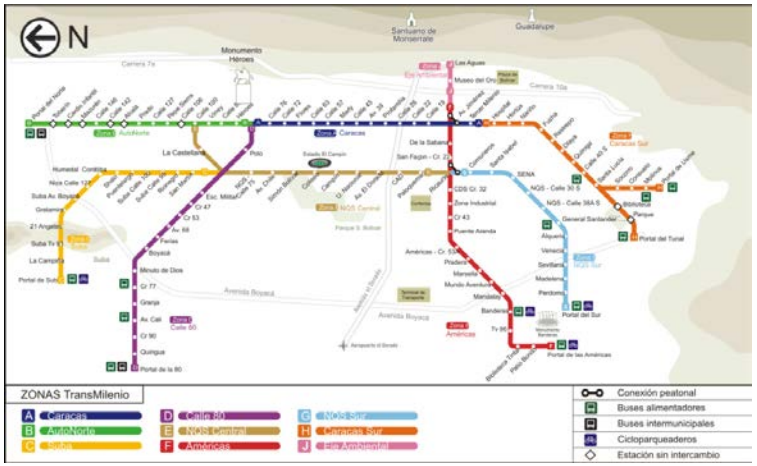
Yinchuan (China)'s Bus Rapid Transit


Bogota (Colombia)'s Bus Rapid Transit

- **BRT systems** are urban transportation solutions that **combine specialized buses, dedicated lanes, stations**, off-vehicle fare collection, and intelligent transportation systems.
- Performance of a BRT system is **equivalent to a railway system, but its cost is lower** (Hodgson et al., 2013).
- **Key characteristics:**
  - Buses can **pass other buses** at stations.
  - **Express routes** stop at fewer stations.
  - **Route transfers** are allowed and sometimes needed.

Bus Rapid Transit Systems (BRT)

Bus Rapid Transit Route Design Problem (BRTRDP)



Bogota's BRT bus corridors

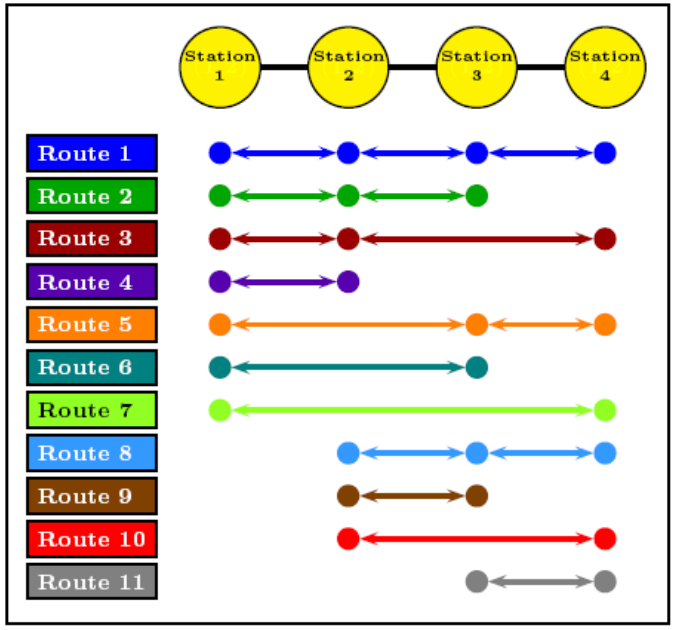
Given the physical network (road/station), the problem consists of **finding a manageable set of routes** that **minimizes total passenger travel time** while **satisfying** system **technical constraints** (OD matrix, min/max bus frequencies), without overcrowding the network (stations, buses, and lanes).

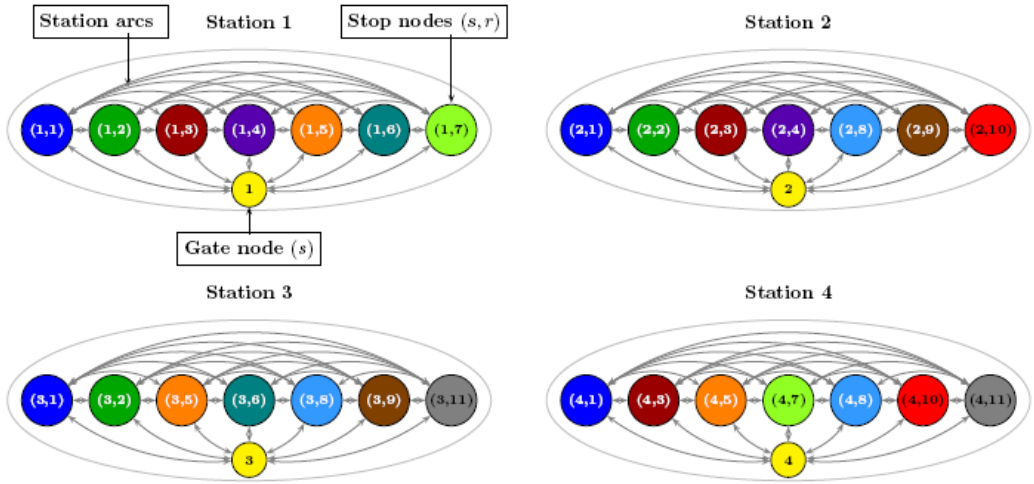
- Walteros, J. L., Medaglia, A. L., Riaño, G. Hybrid Algorithm for Route Design on Bus Rapid Transit Systems (2013), Transportation Science. In Advance.

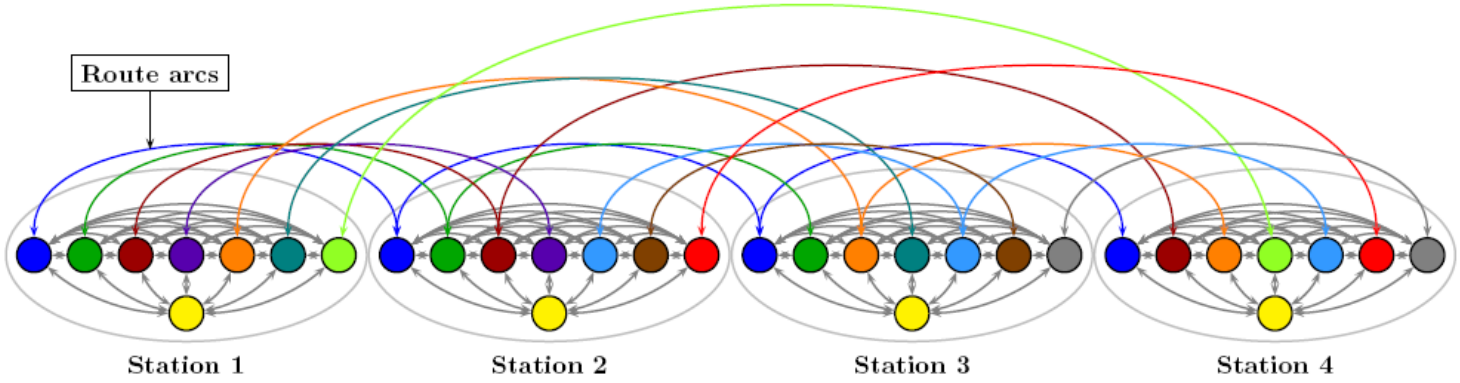
- Feillet, D., Gendreau, M., Medaglia, A. L., Walteros, J. L. A note on branch-and-cut-and-price (2010), Operations Research Letters 38(5): 346-353.

Bus Rapid Transit Systems (BRT)

BRTRDP: network model

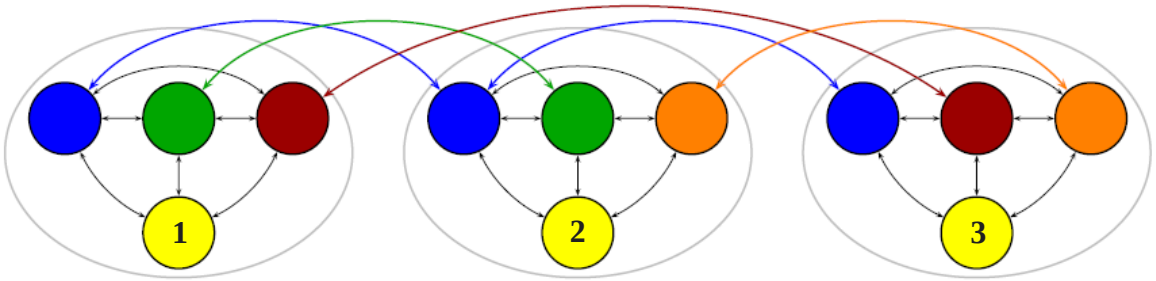




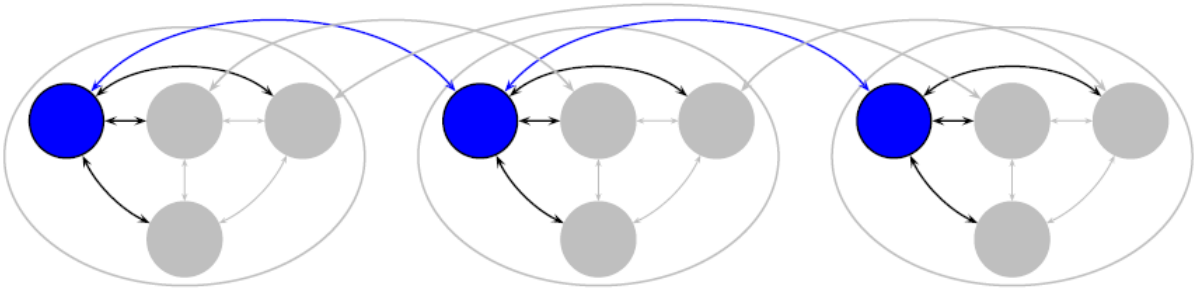


Bus Rapid Transit Systems (BRT)

Network model: a three-station example



Proposed Approach:
Generate the routes systematically



Route =
Nodes and Arcs =
Variables and Constraints

Bus Rapid Transit Systems (BRT)

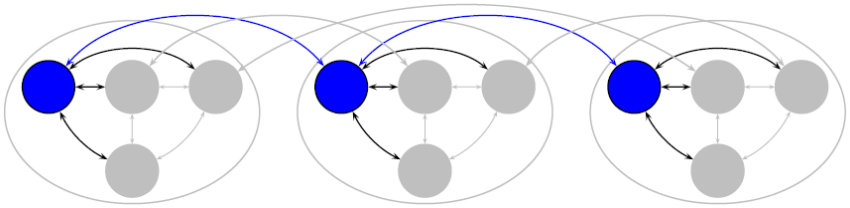
Simultaneous column-row generation

1. Using a solution X_t^* for $MP(\Omega_t)$, build a feasible solution X for $MP(\Omega)$, with cost $z_{MP} = z^*_{MP(\Omega_t)}$.
2. From solution Π_t^* for $D(\Omega_t)$, construct a (not necessarily feasible) solution Π for $D(\Omega)$ with cost $z_D = z^*_{D(\Omega_t)}$ (i.e., $z_D = z_{MP}$).
3. If Π is feasible, the optimality criterion is met and the algorithm stops. Otherwise, at least one constraint of $D(\Omega)$ is being violated, in that case, add at least one route to Ω_t from $\Omega \setminus \Omega_t$, with an associated violated dual constraint and return to step 1.

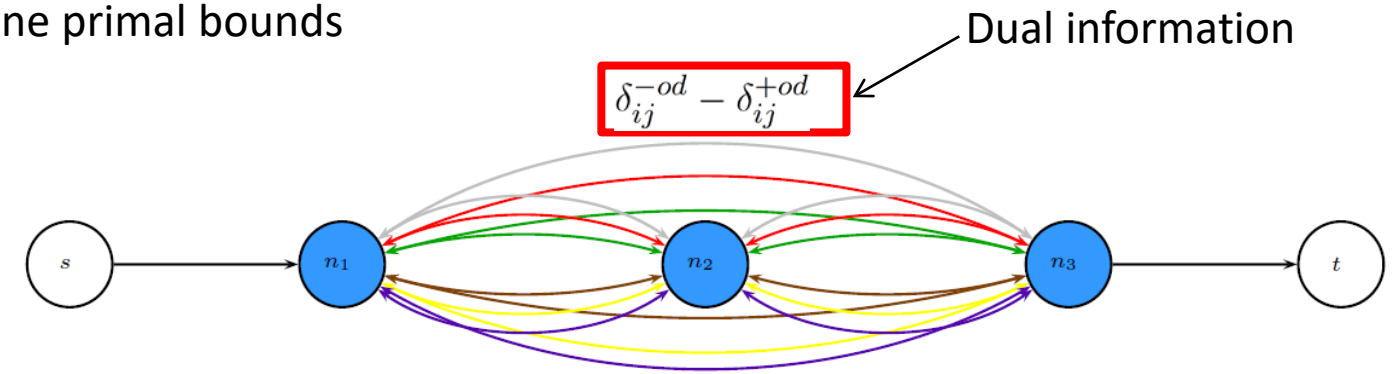
- Feillet, Gendreau, Medaglia & Walteros (2010)

Bus Rapid Transit Systems (BRT)

Acceleration with the pulse algorithm



-**Pruning strategies:**
  - Dominance
  - Infeasibility
- **Acceleration strategies:**
  - Explore several graphs per iteration
  - Find several negative-cost paths
  - Add more than one route per iteration
  - Define primal bounds



Bus Rapid Transit Systems (BRT)

Computational experiments

Comparative results on the single-corridor BRT systems by Feillet et al. (2010)

Instance			Feillet et al. (2010)		SCCG		
Label	Routes	Optimal value	Generated routes	CPU time (s)	Generated routes	CPU time (s)	Speedup
BRT-C1-S3	4	123,900	3	0	2	0	1
BRT-C1-S4	11	249,600	5	0	4	0	1
BRT-C1-S5	26	419,000	8	0	8	0	1
BRT-C1-S6	57	633,000	12	1	13	0	> 1
BRT-C1-S7	120	892,500	17	2	22	1	2.4
BRT-C1-S8	247	1,198,400	23	5	25	1	5.1
BRT-C1-S9	502	1,551,600	30	14	31	2	8.7
BRT-C1-S10	1,013	1,953,000	38	35	41	3	12.5
BRT-C1-S11	2,036	2,403,500	48	48	48	6	8.0
BRT-C1-S12	4,083	2,904,000	59	121	60	10	11.9
BRT-C1-S13	8,178	3,455,400	71	503	71	21	24.1
BRT-C1-S14	16,369	4,058,600	87	1,826	76	30	60.5
BRT-C1-S15	32,752	4,714,500	97	3,442	91	39	87.2
BRT-C1-S16	65,519	5,424,000	118	13,691	104	93	147.7
BRT-C1-S17	131,054	6,188,000	131	31,428	111	120	262.6
BRT-C1-S18	262,125	7,007,400	153	67,812	131	164	414.1
BRT-C1-S19	524,268	7,883,100	168	140,361	142	371	378.2
					Arithmetic mean		109.5
					Geometric mean		34.5

Working Paper: Solving the Bus Rapid Transit Route Design Problem with General Topologies via Simultaneous Column and Cut Generation.

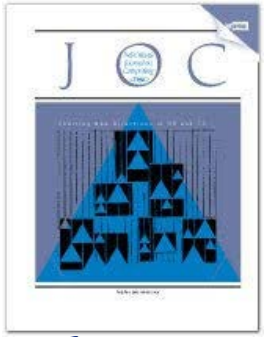
Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
  - Multi-activity shift scheduling problem
  - Bus rapid transit route design problem
  - **Interdiction problem with fortification**
  - Green VRP
  - Other applications
- Part V: perspectives

Interdiction Problems with Fortification

L. Lozano & J. C. Smith

- Lozano, L. & Smith, J. C. (2017). A Backward Sampling Framework for Interdiction Problems with Fortification. *INFORMS Journal on Computing.* 29(1):123-139. url: https://doi.org/10.1287/ijoc.2016.0721



Interdiction Problems with Fortification

Problem statement

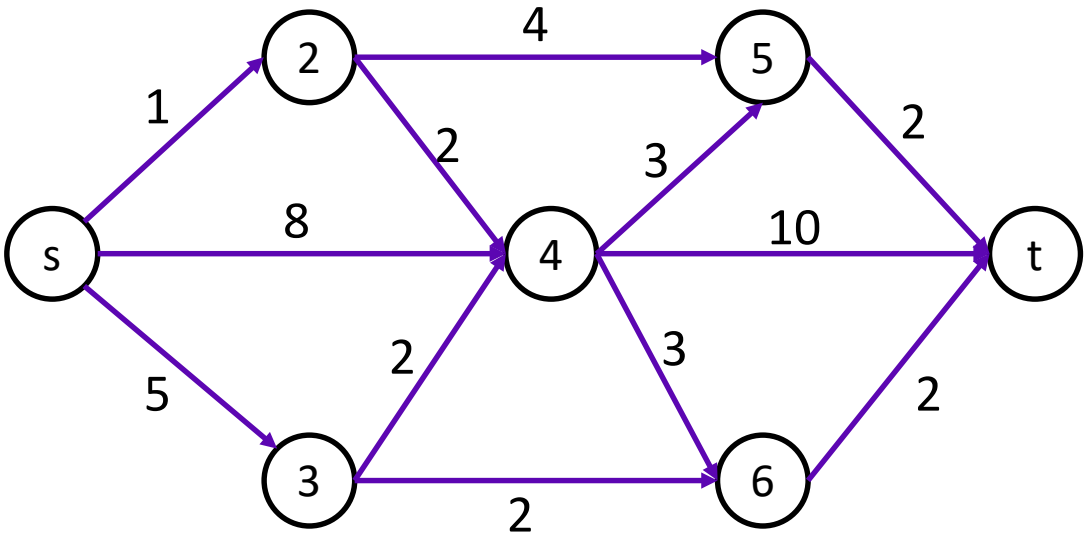
$$\mathcal{P}: \quad z^* = \min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{x}\in\mathcal{X}(\mathbf{w})} \min_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f(\mathbf{y})$$

- Defender-attacker-defender shortest path problem
- Modeled as three-level, two-player Stackelberg game
- First stage: defender fortifies subset of arcs
- Second stage: attacker destroys subset of unprotected arcs
- Third stage: shortest path problem over surviving network

Interdiction Problems with Fortification

Toy Example: consider only one stage

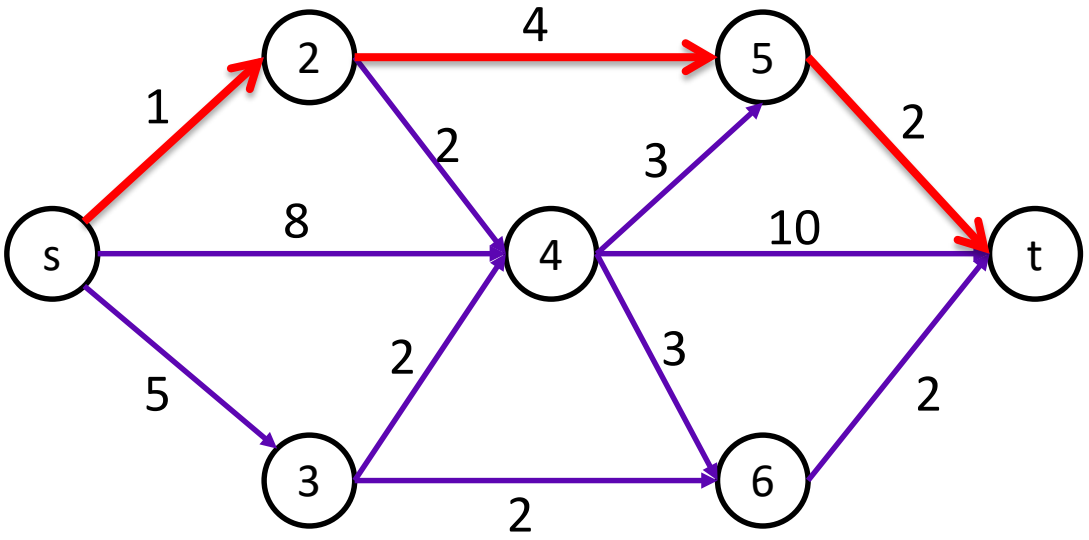
$$z^* = \min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{x}\in\mathcal{X}(\mathbf{w})} \min_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f(\mathbf{y})$$



Interdiction Problems with Fortification

Toy Example: consider only one stage

$$z^* = \min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{x}\in\mathcal{X}(\mathbf{w})} \min_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f(\mathbf{y})$$



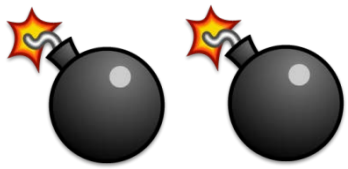
Shortest path: 7

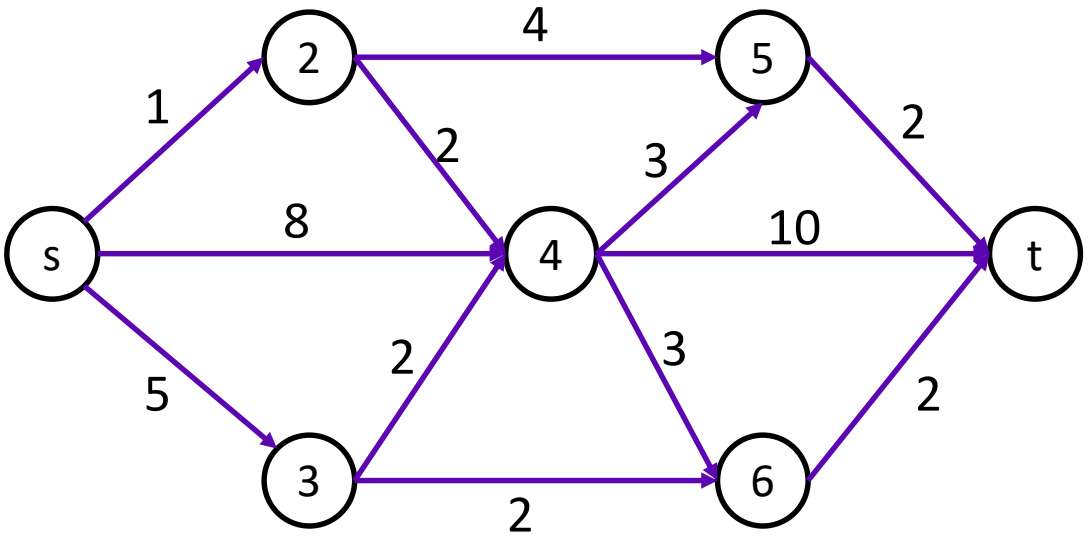
Interdiction Problems with Fortification

Toy Example: consider two stages

$$z^* = \min_{\mathbf{w} \in \mathcal{W}} \max_{\mathbf{x} \in \mathcal{X}(\mathbf{w})} \min_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} f(\mathbf{y})$$

Attacker's budget:

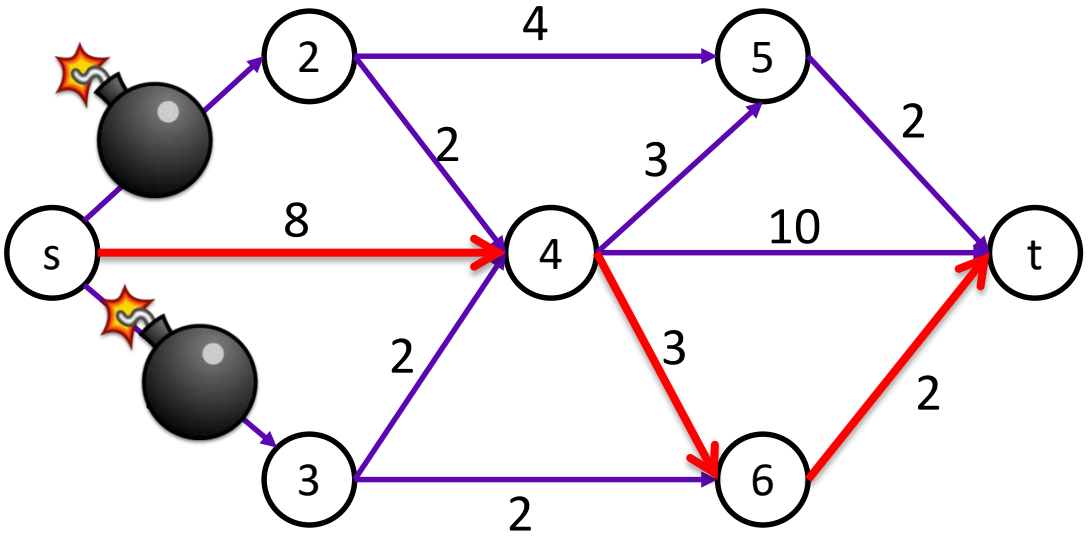




Interdiction Problems with Fortification

Toy Example: consider two stages

$$z^* = \min_{\mathbf{w} \in \mathcal{W}} \max_{\mathbf{x} \in \mathcal{X}(\mathbf{w})} \min_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} f(\mathbf{y})$$



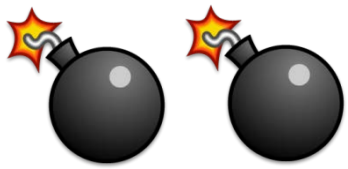
Shortest path: 13

Interdiction Problems with Fortification

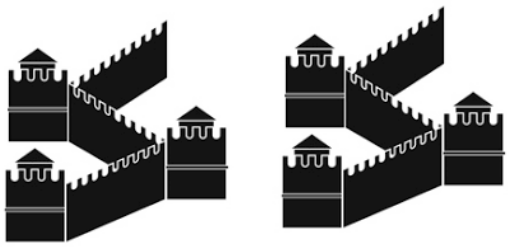
Toy Example: consider three stages

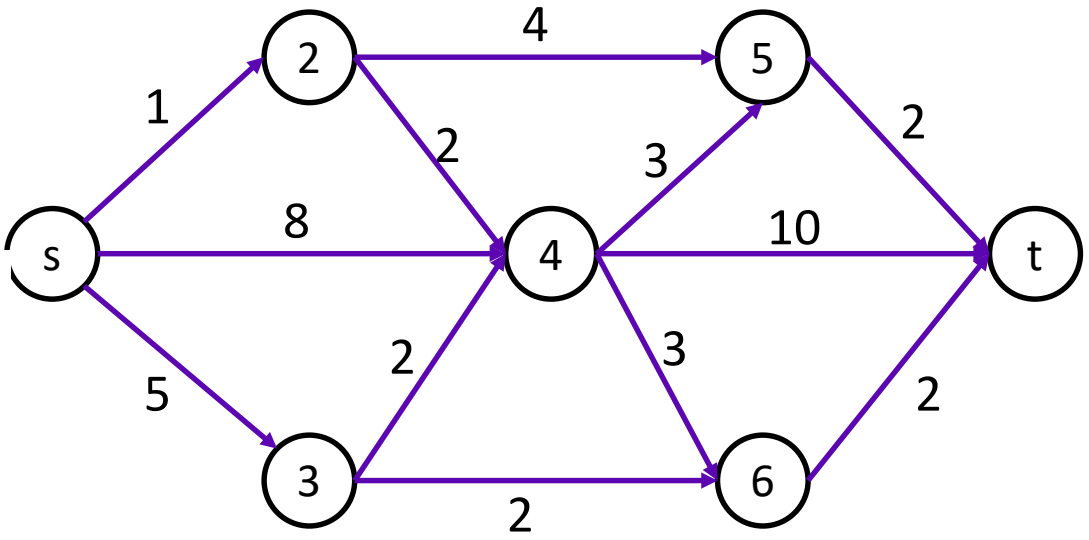
$$z^* = \min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{x}\in\mathcal{X}(\mathbf{w})} \min_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f(\mathbf{y})$$

Attacker's budget:



Defender's budget:



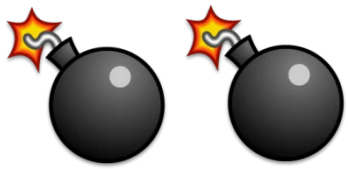


Interdiction Problems with Fortification

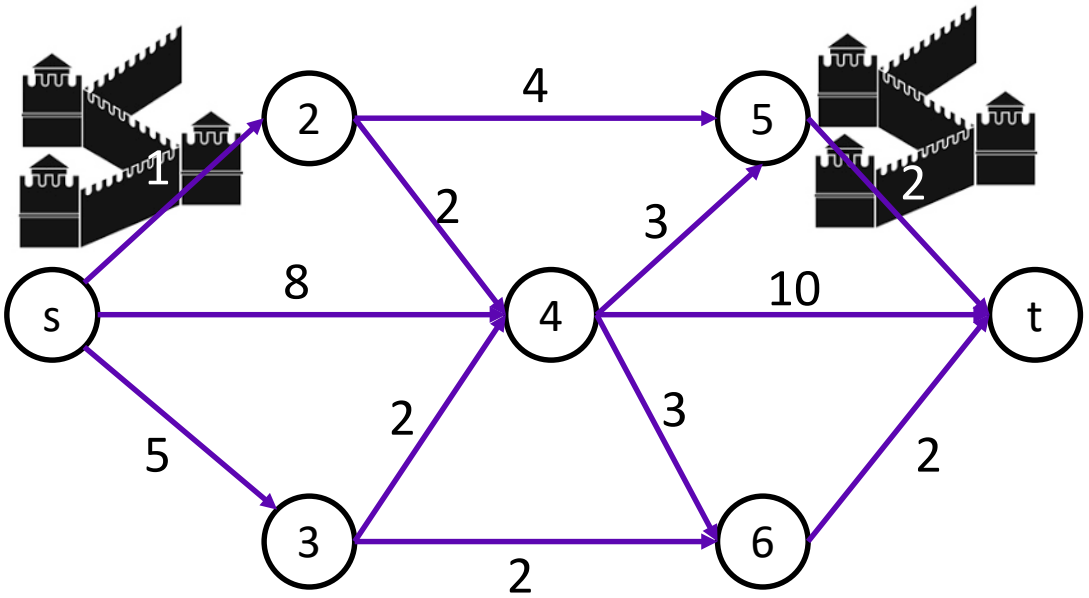
Toy Example: consider three stages

$$z^* = \min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{x}\in\mathcal{X}(\mathbf{w})} \min_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f(\mathbf{y})$$

Attacker's budget:



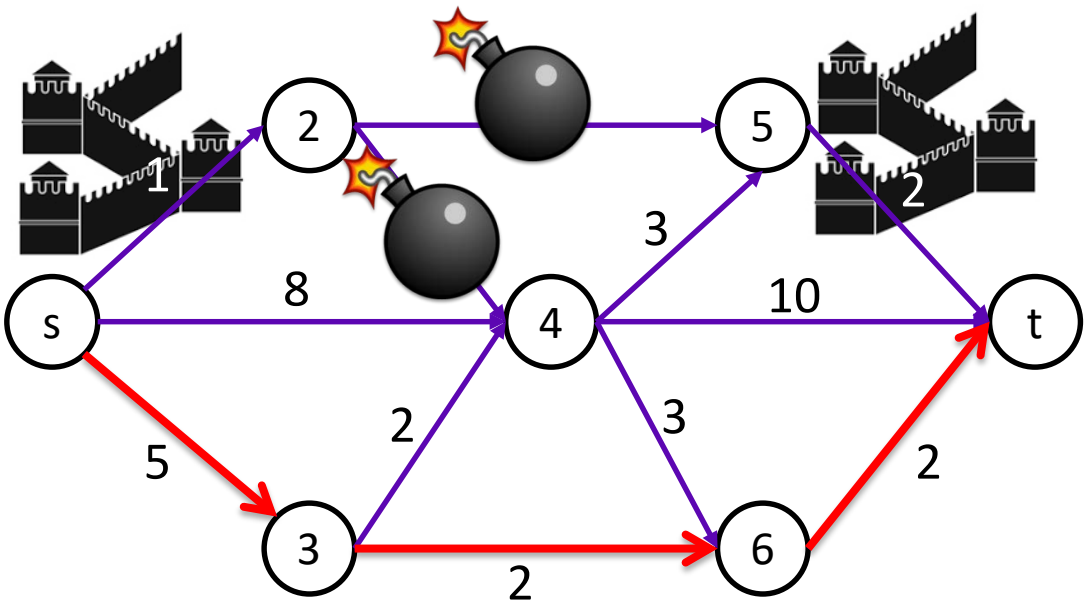
Defender's budget:



Interdiction Problems with Fortification

Toy Example: consider three stages

$$z^* = \min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{x}\in\mathcal{X}(\mathbf{w})} \min_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f(\mathbf{y})$$



Shortest path: 9

Interdiction Problems with Fortification

Backward Sampling Framework

Algorithm 1 Solving bilevel interdiction problem $Q(\mathbf{w})$ via backward sampling

Input: Problem $\mathcal{P}$ and a feasible defense $\mathbf{w} \in \mathcal{W}$

Output: An optimal solution to $Q(\mathbf{w})$

1: Initialize $UB_0 = \infty$ and $LB_0 = -\infty$, select $\hat{\mathcal{Y}}^1 \subseteq \mathcal{Y}$ as a sampling of the third-stage solutio space, and set counter $i = 0$
2: **while** $LB_i < UB_i$ **do**
3: Set $i = i + 1$
4: Solve $UB_i = z^I(\mathbf{w}, \hat{\mathcal{Y}}^i) = \max_{\mathbf{x} \in \mathcal{X}(\mathbf{w})} \min_{\mathbf{y} \in \hat{\mathcal{Y}}^i(\mathbf{x})} f(\mathbf{y})$ and obtain an optimal solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$
5: Solve $LB_i = z^R(\hat{\mathbf{x}}) = \min_{\mathbf{y} \in \mathcal{Y}(\hat{\mathbf{x}})} f(\mathbf{y})$ and obtain an optimal solution $\hat{\mathbf{y}}^*$
6: Set $\hat{\mathcal{Y}}^{i+1} = \hat{\mathcal{Y}}^i \cup \{\hat{\mathbf{y}}^*\}$
7: **if** $UB_i < UB_{i-1}$ **then**
8: Remove from $\hat{\mathcal{Y}}^{i+1}$ all solutions having objective value greater than UB_i
9: Add to $\hat{\mathcal{Y}}^{i+1}$ a subset of new solutions in $\mathcal{Y}_{UB_i}$
10: **end if**
11: **if** $LB_i = UB_i$ **then**
12: Terminate with solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$
13: **end if**
14: **end while**

Interdiction Problems with Fortification

How to sample?

- Pulse algorithm for constrained shortest path problem (Lozano and Medaglia, 2013) to generate a **diverse** sample of **near-optimal** paths.
- Bounds-pruning: only include in the sample “good” paths
- Arc-usage pruning: do not let too many paths in the sample traverse the same arc

Interdiction Problems with Fortification

Computational experiments

Pulse-enabled

C&S (2011)

Instance	Nodes	Arcs	Q	B	Sampling			SPI		
					Avg	Max	# solved	Avg	Max	# solved
10×10	102	416	3	3	0.1	0.3	60	1.9	4.6	60
			5	4	0.3	0.8	60	30.9	162.8	60
			4	5	0.7	2.9	60	67.5	284.2	60
20×20	402	1,826	3	3	0.6	3.3	60	26.7	305.1	60
			5	4	2.2	12.8	60	1128.4	7723.4	60
			4	5	5.8	33.4	60	2495.2	>14,400	56
30×30	902	4,236	3	3	1.7	7.5	60	766.9	12,728.8	60
			5	4	7.2	36.9	60	4857.3	>14,400	45
			4	5	18.6	94.8	60	4256.8	>14,400	26

- Cappanera & Scaparra (2011)

Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
  - Multi-activity shift scheduling problem
  - Bus rapid transit route design problem
  - Interdiction problem with fortification
  - **Green VRP**
  - Other applications
- Part V: perspectives

Green Vehicle Routing Problem (VRP)

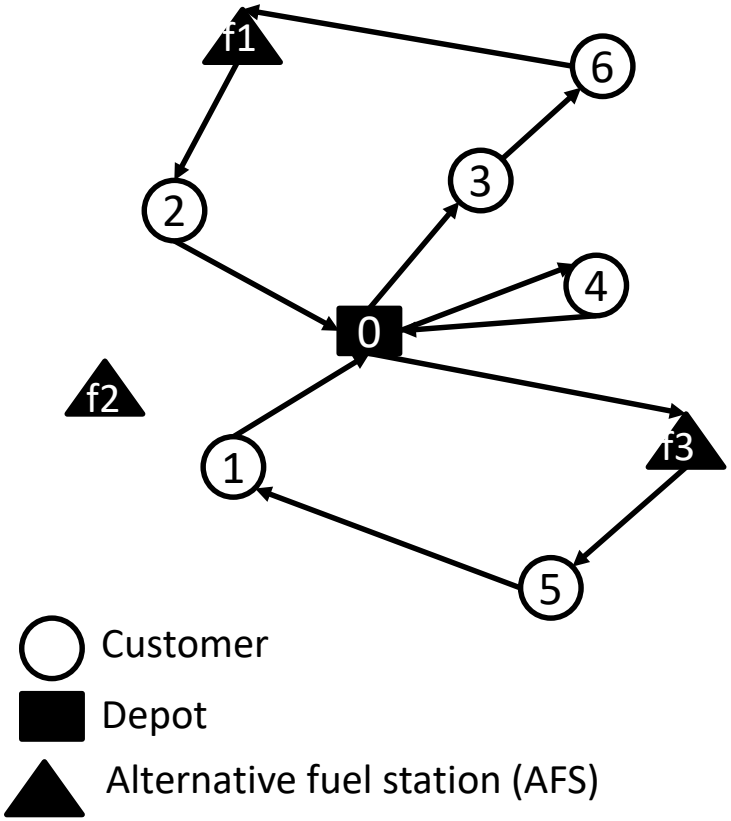
A. Montoya, C. Guéret, J. E. Mendoza, J. G. Villegas

- Montoya, A., Guéret, C., Mendoza, J. E., Villegas, J. G. (2016). A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*. 70: 113-128. DOI: https://doi.org/10.1016/j.trc.2015.09.009

Green Vehicle Routing Problem (VRP)

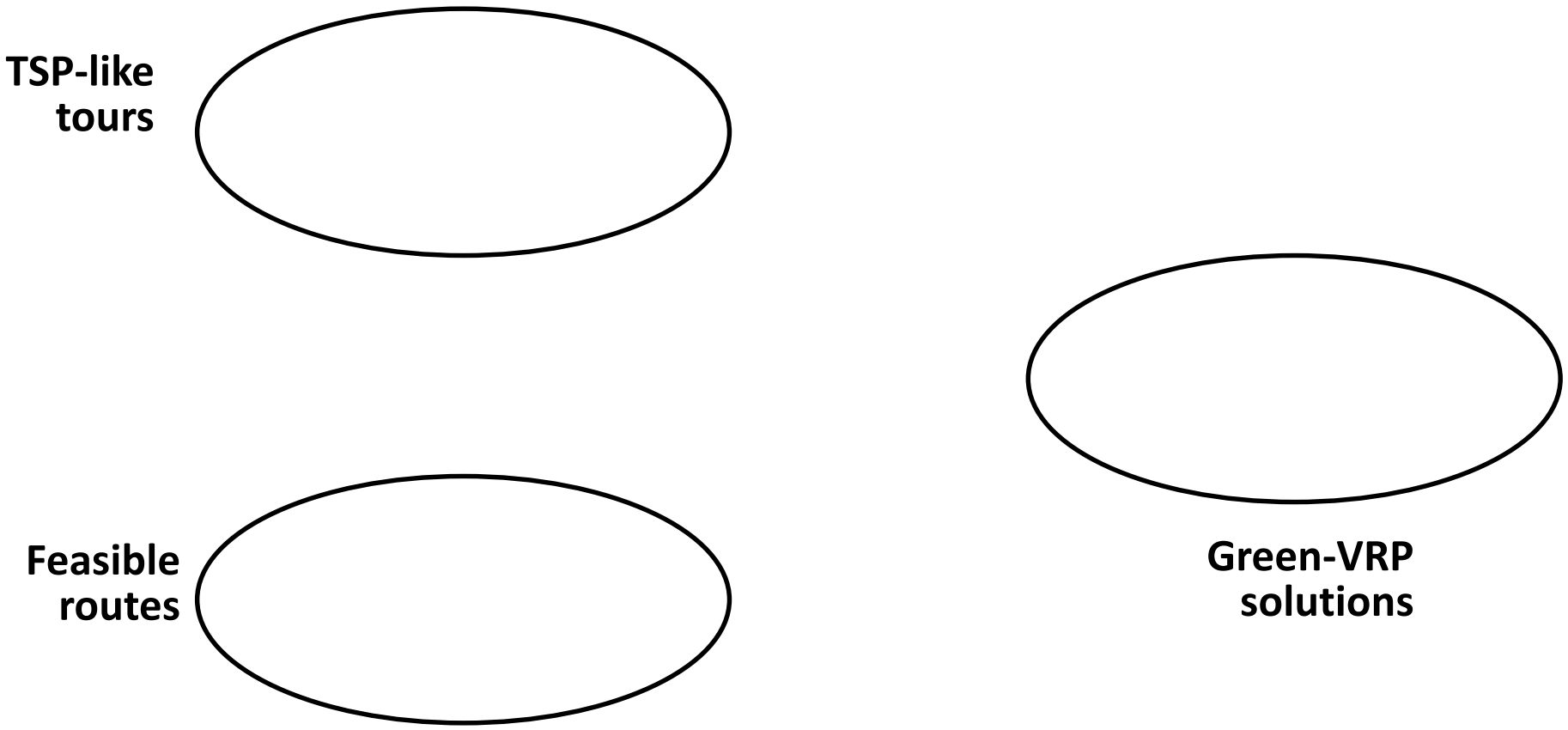
Problem statement

- The fleet are zero emission vehicles (ZEVs) (electric, liquid nitrogen, hydrogen, among others)
- ZEVs have a limited tank capacity
- Alternative fuel stations (AFS) have to be visited en-route in order to refill the tank (or recharge the battery)



Green Vehicle Routing Problem (VRP)

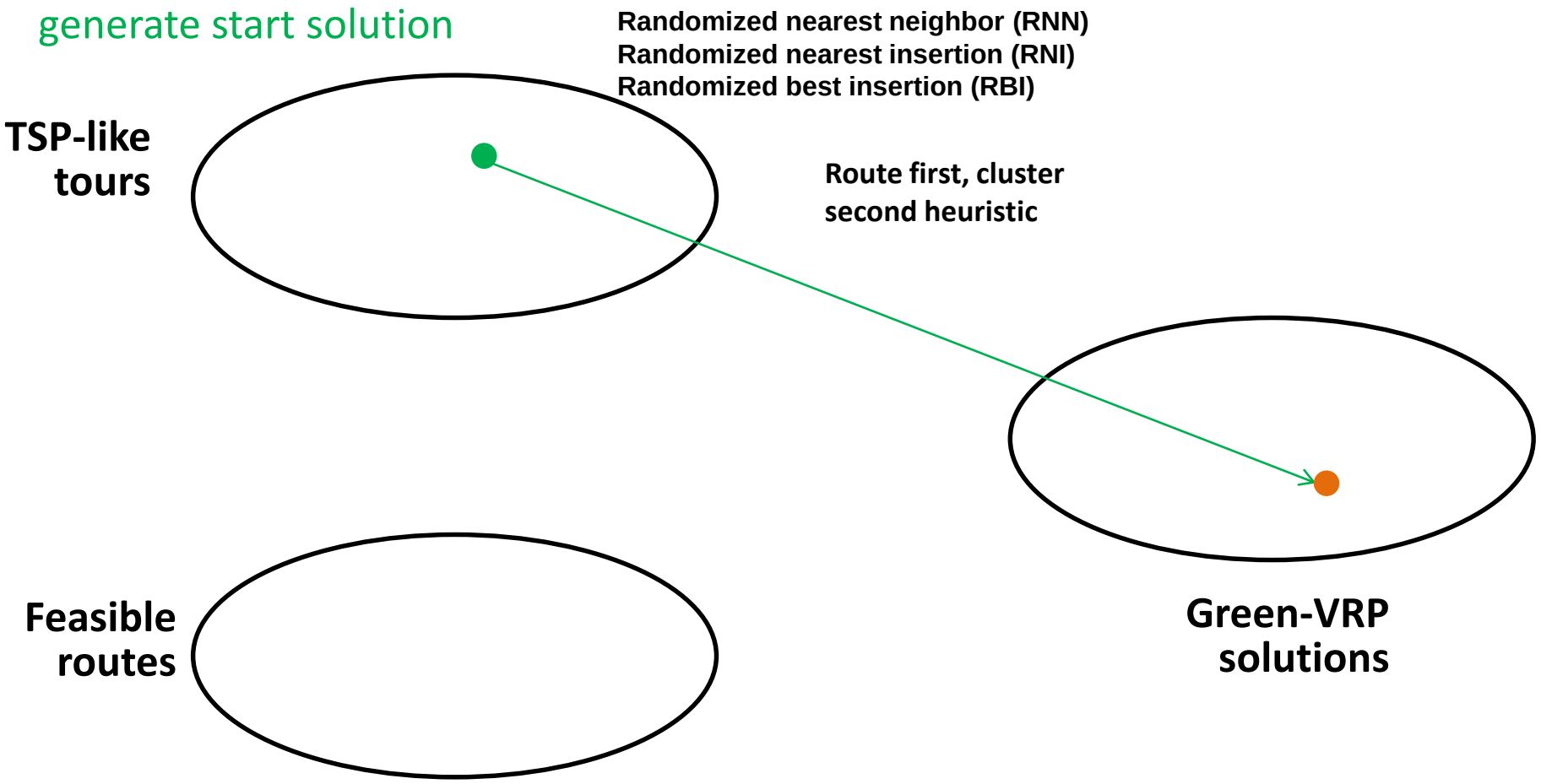
Multi-space sampling heuristic (MSH)



- Mendoza & Villegas (2013)

Green Vehicle Routing Problem (VRP)

Multi-space sampling heuristic (MSH)

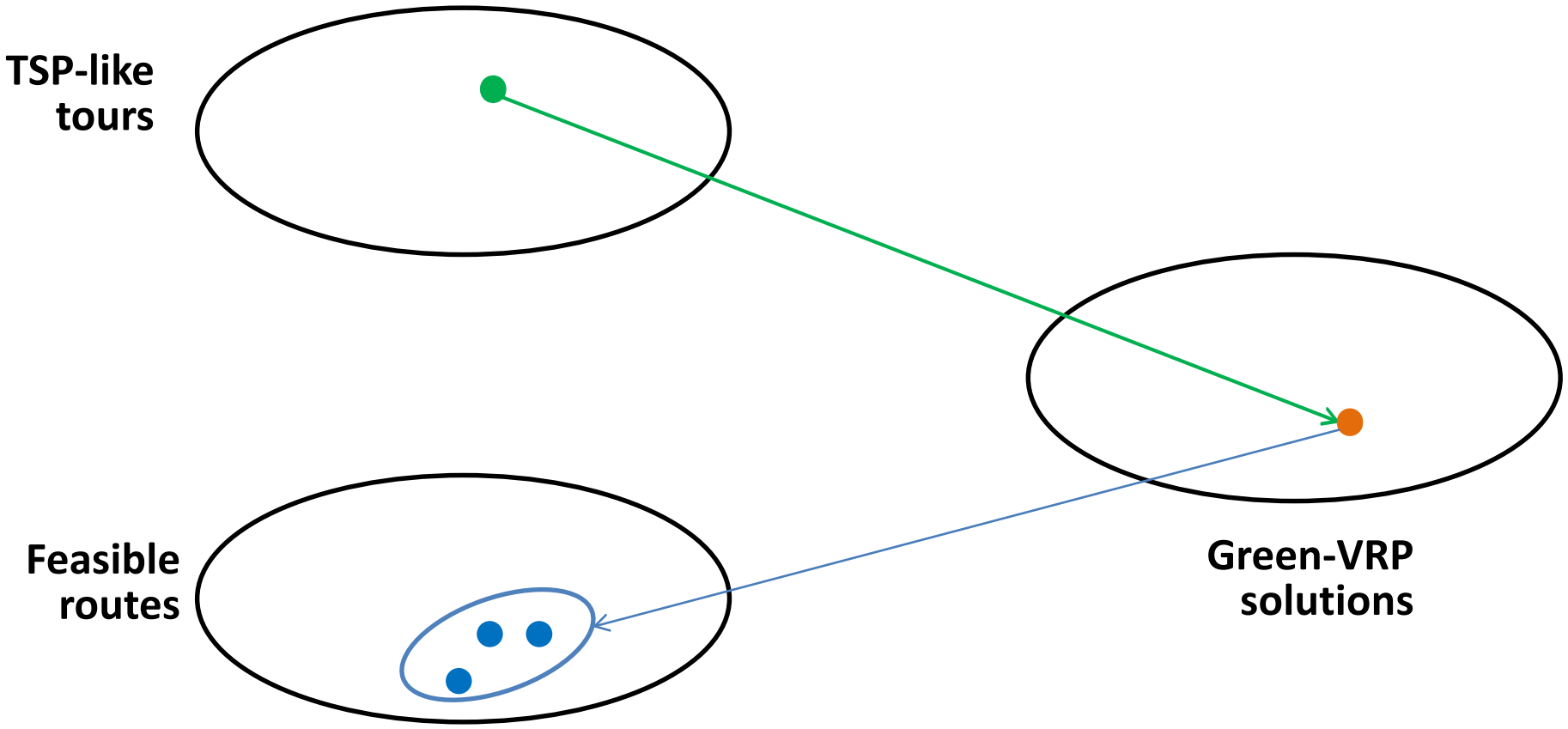


- Mendoza & Villegas (2013)

Green Vehicle Routing Problem (VRP)

Multi-space sampling heuristic (MSH)

generate start solution + route storing

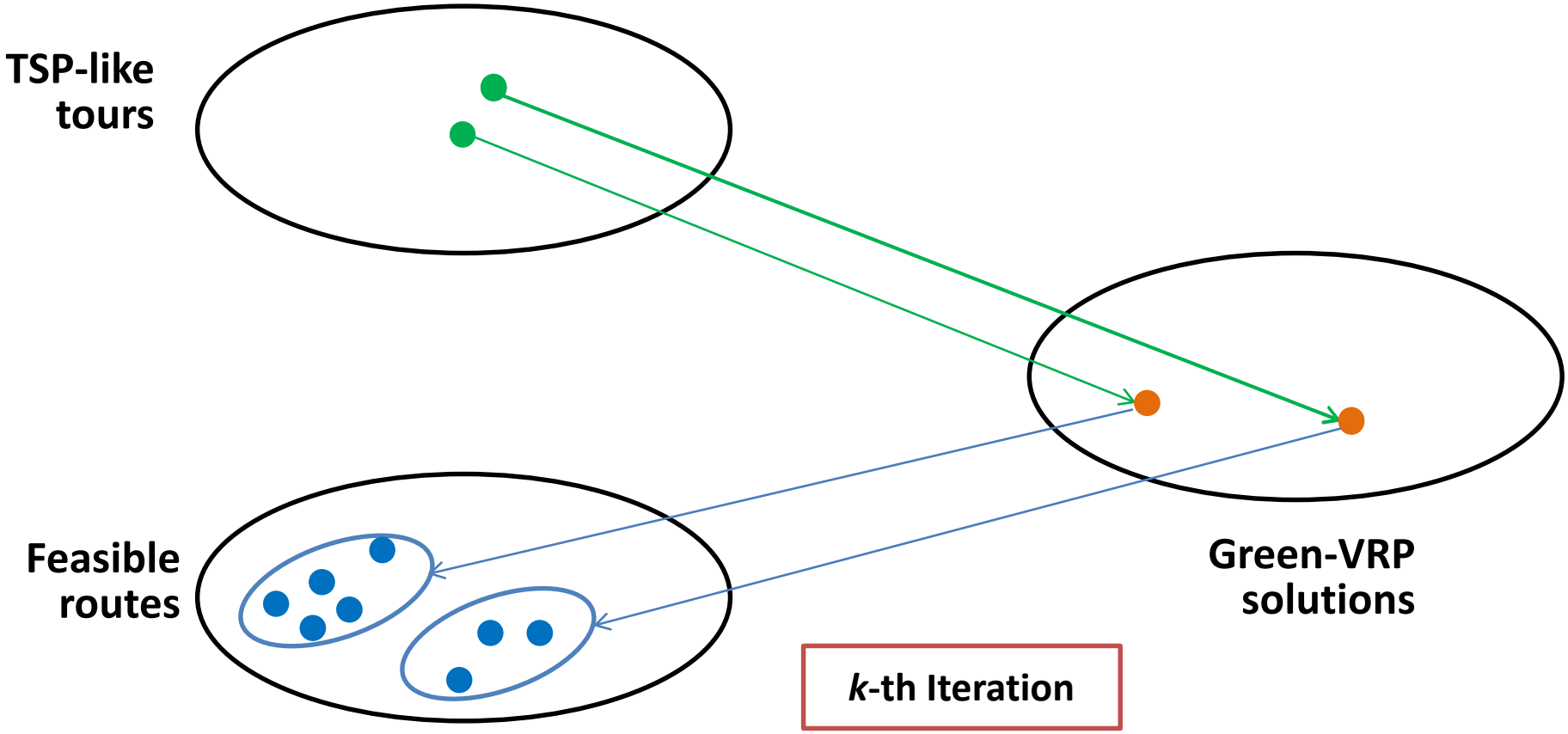


- Mendoza & Villegas (2013)

Green Vehicle Routing Problem (VRP)

Multi-space sampling heuristic (MSH)

generate start solution + route storing

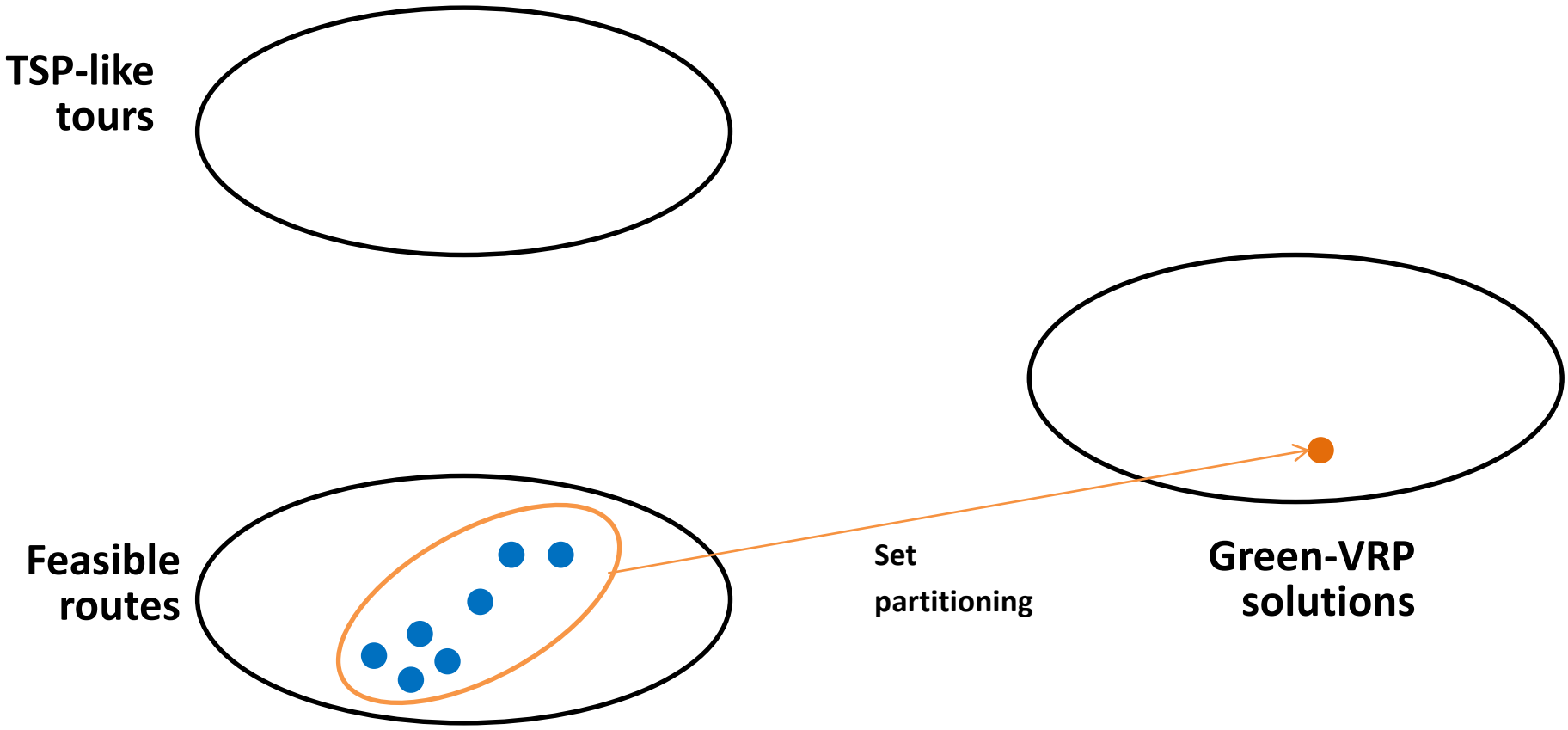


- Mendoza & Villegas (2013)

Green Vehicle Routing Problem (VRP)

Multi-space sampling heuristic (MSH)

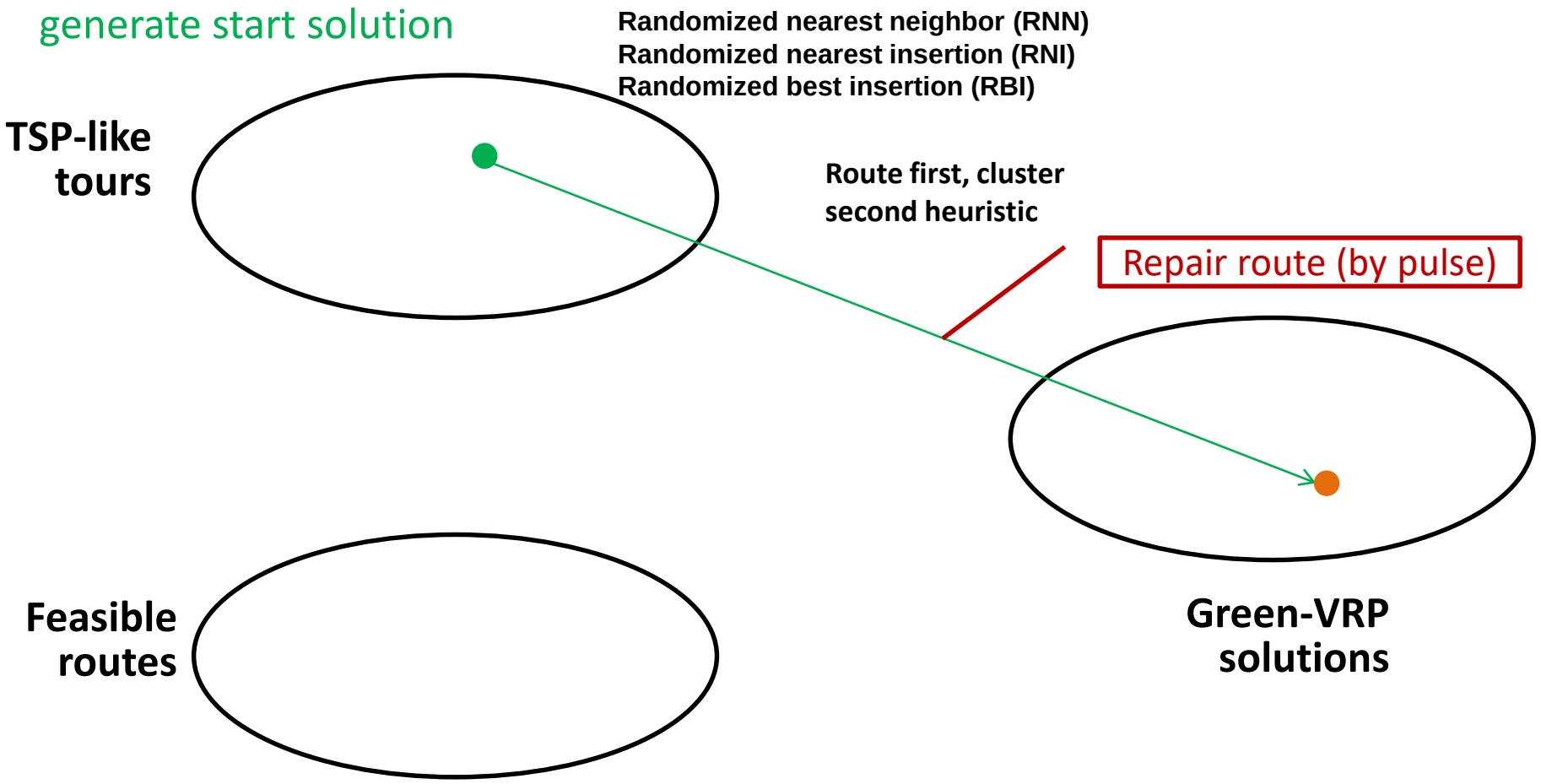
Assembly route



- Mendoza & Villegas (2013)

Green Vehicle Routing Problem (VRP)

Multi-space sampling heuristic (MSH): repairing routes

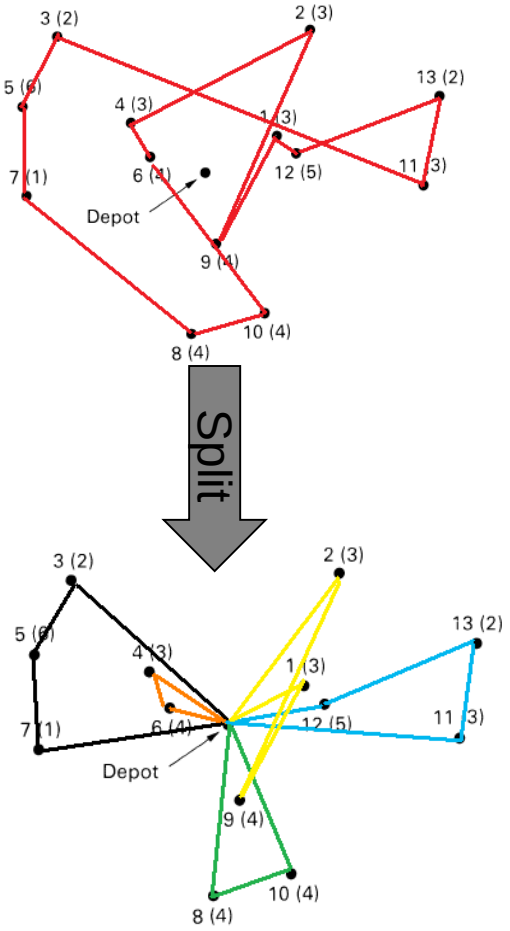


- Mendoza & Villegas (2013)

Green Vehicle Routing Problem (VRP)

Split method: a primer

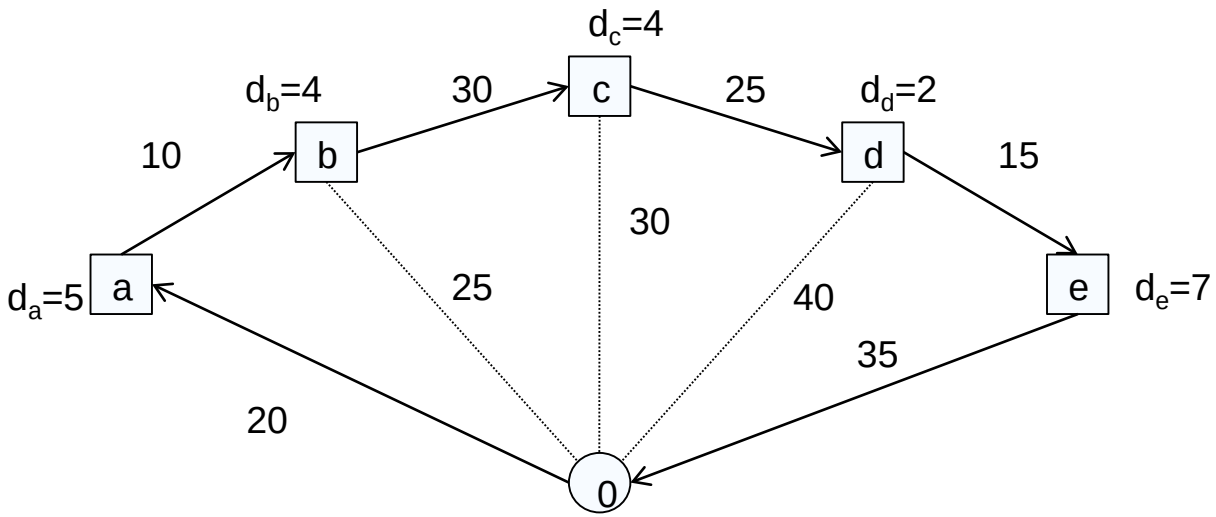
- Route-first, cluster-second (Beasley, 1983)
- Given a solution of a TSP, split it into feasible routes of the VRP at minimum cost
  - Creation of an auxiliary graph
  - Solution of a shortest path problem in the auxiliary graph
  - Extraction of the VRP solution



Green Vehicle Routing Problem (VRP)

Split method: a primer

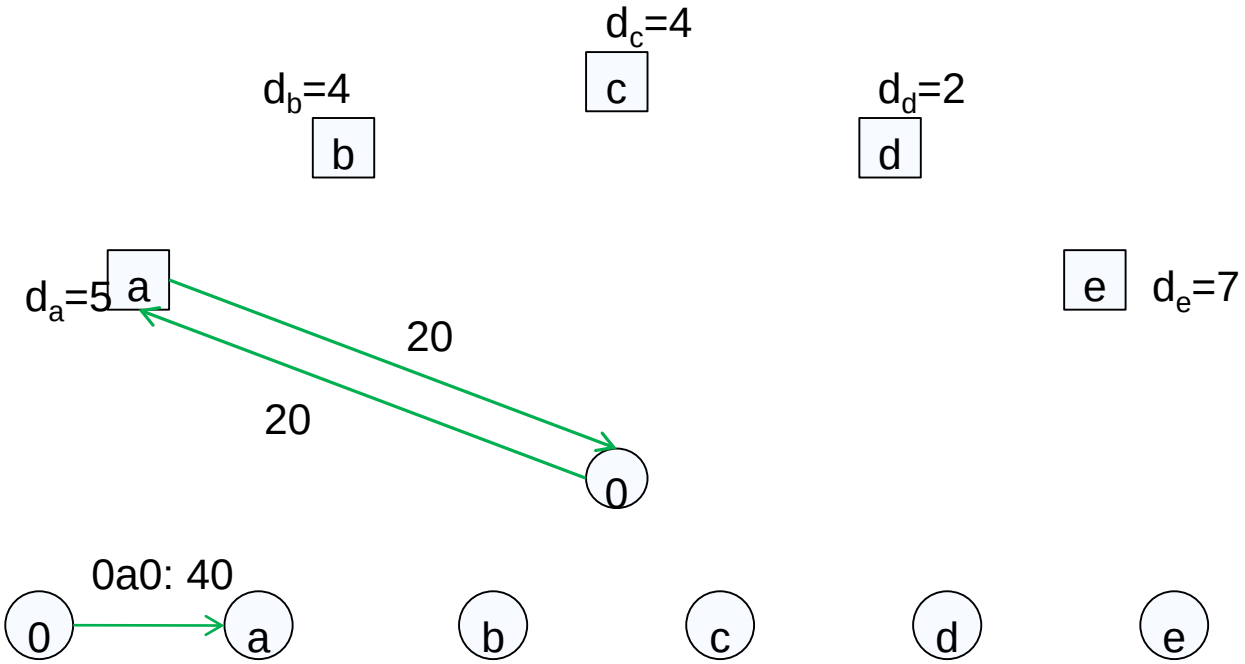
- Example (Prins, 2004):
  - Permutation: | a | b | c | d | e |
  - Vehicle capacity: 10



Green Vehicle Routing Problem (VRP)

Split method: a primer

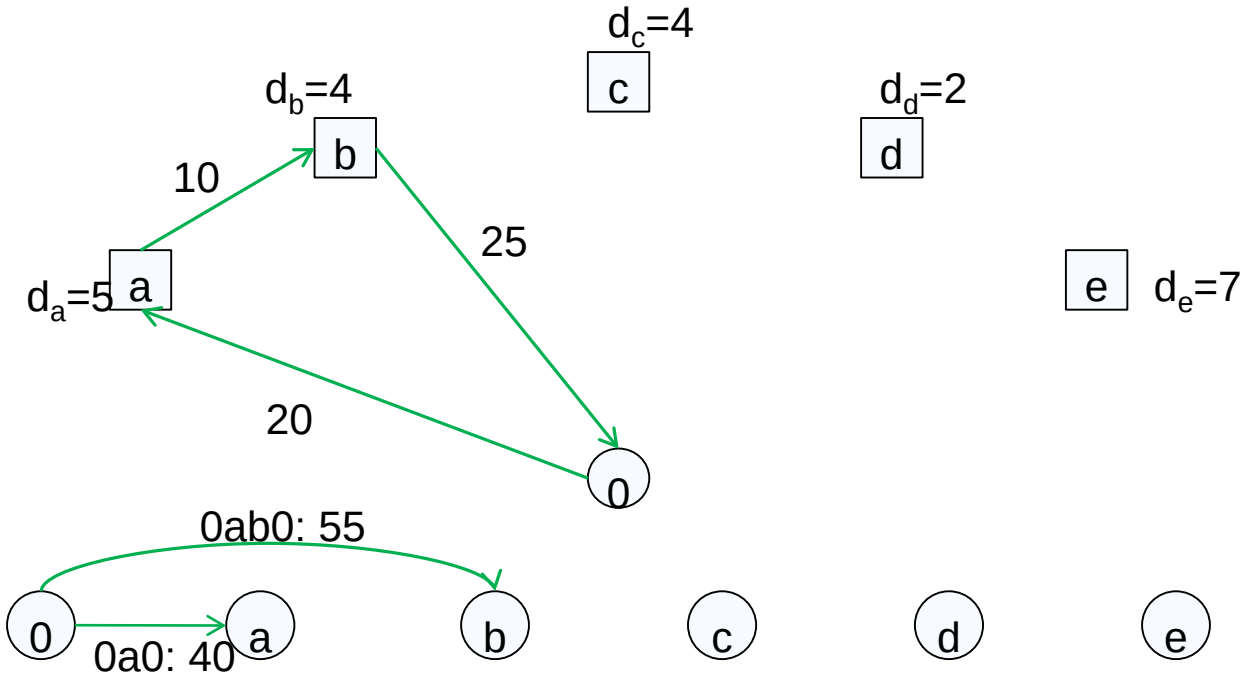
- Example (Prins, 2004):
  - Permutation: | a | b | c | d | e |
  - Vehicle capacity: 10



Green Vehicle Routing Problem (VRP)

Split method: a primer

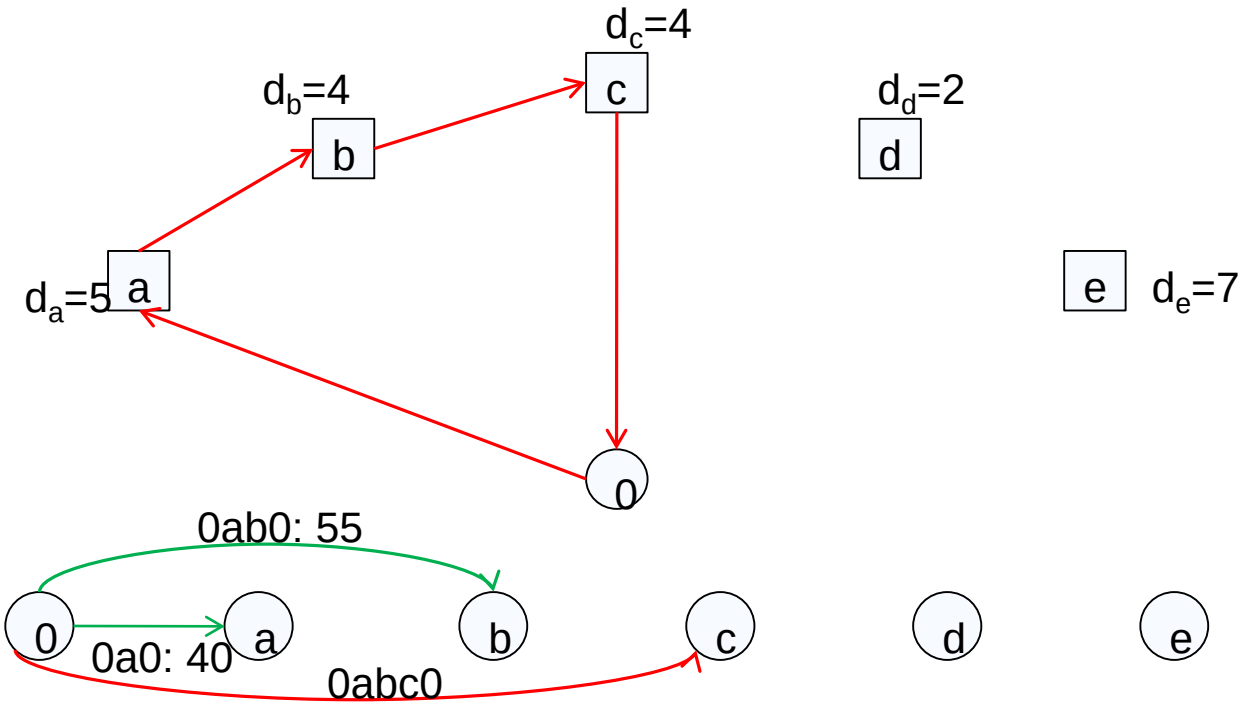
- Example (Prins, 2004):
  - Permutation: a b c d e
  - Vehicle capacity: 10



Green Vehicle Routing Problem (VRP)

Split method: a primer

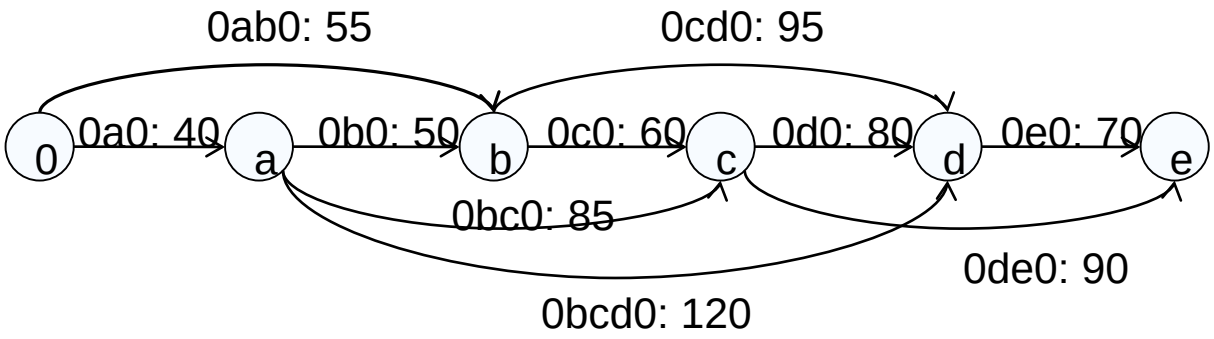
- Example (Prins, 2004):
  - Permutation: | a | b | c | d | e |
  - Vehicle capacity: 10



Green Vehicle Routing Problem (VRP)

Split method: a primer

- Example (Prins, 2004):
  - Permutation: | a | b | c | d | e |
  - Vehicle capacity: 10

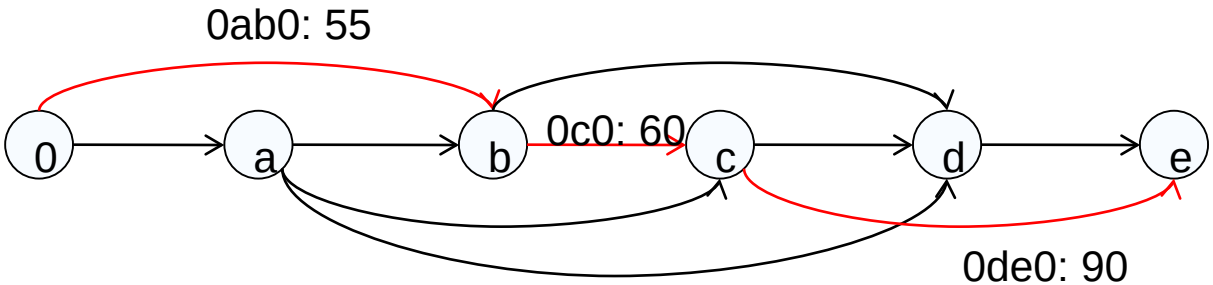


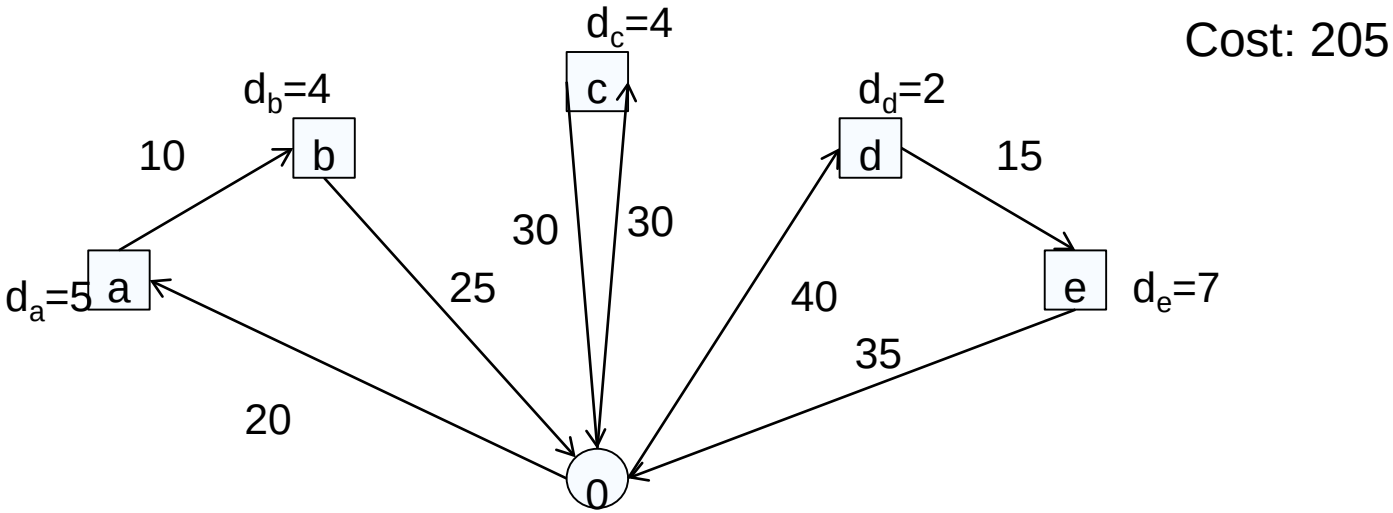
Green Vehicle Routing Problem (VRP)

Split method: a primer

Permutation:

a	b	c	d	e



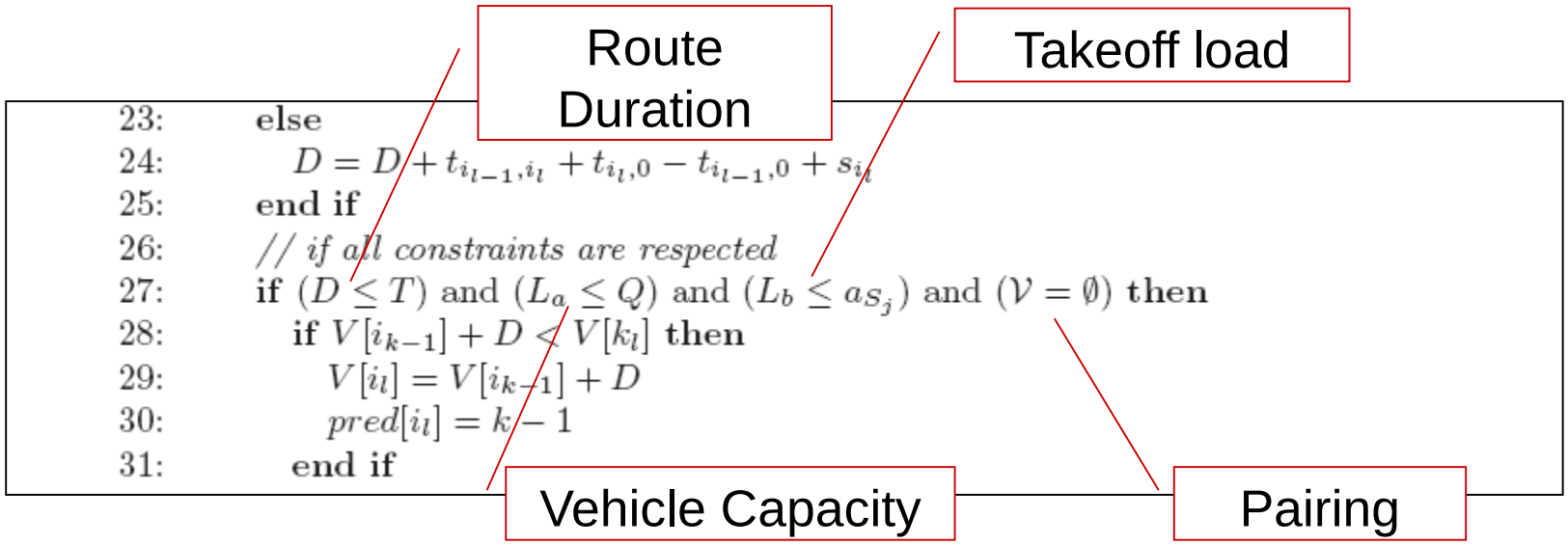


Cost: 205

Green Vehicle Routing Problem (VRP)

Split method: a primer

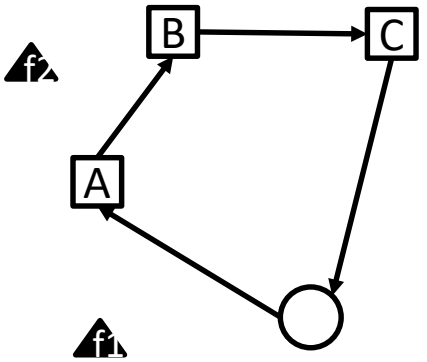
- Many elements of rich VRPs can be handled while constructing the auxiliary graph.
- Example: VRPPD



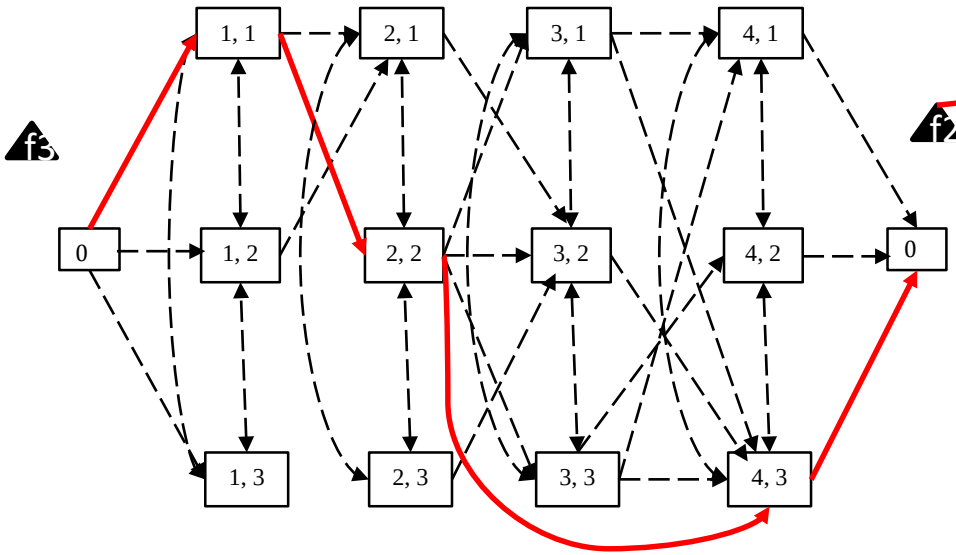
Green Vehicle Routing Problem (VRP)

How to repair routes in the Green VRP?

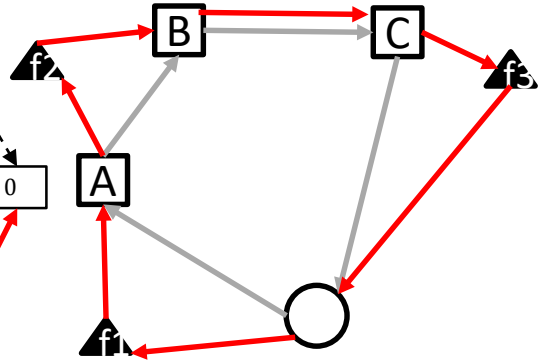
Fuel	✗
Time	✓



R: (0, A, B, C)



Fuel	✗ (✓)
Time	✓



- Pulse algorithm for the constrained shortest path problem with a maximum travel time.

Green Vehicle Routing Problem (VRP)

Computational experiments

Table 1
Summary results and comparison of our MSH with other methods on the small instances of Erdoğan and Miller-Hooks (2012).

Metric	MCWS/DBCA	VNS/TS	AVNS	48A	MSH(1k)	MSH(5k)	MSH(10k)
Number of BKS	2/40	38/40	40/40	29/40	35/40	40/40	40/40
Avg. gap (%)	NR	NR	0.15	NR	0.38	0.04	0.01
Max. gap (%)	NR	NR	1.73	NR	3.61	0.47	0.13
Avg. best gap (%)	8.72	0.63	0.00	0.46	0.09	0.00	0.00
Cum. number of veh.	245	223	NR	225	222	222	222
Avg. time (min)	NR	0.65	0.17	0.02	0.01	0.04	0.07
Max. avg. time (min)	NR	0.88	0.38	0.04	0.02	0.06	0.12
Computer	Pentium 4 3.2 GHz	Core I5 2.67 GHz	Core I5 2.67 GHz	Core I5 2.8 GHz	XEON E5410 2.33 GHz		
Runs	NR	10	10	1	10	10	10

NR: not reported.

Table 2
Summary results and comparison of our MSH with other methods on the large instances of Erdoğan and Miller-Hooks (2012).

Metric	MCWS/DBCA	VNS/TS	AVNS	48A	SA	MSH (1k)	MSH (5k)	MSH (10k)
Number of BKS	0/12	0/12	4/12	0/12	0/12	0/12	0/12	8/12
Avg. gap (%)	NR	NR	0.92	NR	NR	2.64	1.48	1.02
Max. gap (%)	NR	NR	1.84	NR	NR	5.77	4.21	3.62
Avg. best gap (%)	15.97	1.38	0.17	4.50	4.97	1.42	0.40	0.05
Cum. number of veh.	508	461	NR	466	459	454	445	444
Avg. time (min)	NR	159.58	6.20	157.03	156.05	27.92	31.47	35.04
Max. avg. time (min)	NR	525.52	19.51	514.68	456.26	80.11	84.95	89.95
Computer	Pentium 4 3.2 GHz	Core I5 2.67 GHz	Core I5 2.67 GHz	Core I5 2.8 GHz	Core I5 2.8 GHz	XEON E5410 2.33 GHz		
Runs	NR	10	10	1	1	10	10	10

NR: not reported.

Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
  - Multi-activity shift scheduling problem
  - Bus rapid transit route design problem
  - Interdiction problem with fortification
  - Green VRP
  - **Other applications**
- Part V: perspectives

Evasive Flow Capturing Problem

O. Arslan, O. Jabali, G. Laporte

- Arslan, A., Jabali, O., Laporte, G. (2018). Exact Solution of the Evasive Flow Capturing Problem. *Operations Research*. Forthcoming.

- The Evasive Flow Capturing Problem is the problem of locating a set of law enforcement facilities on the arcs of a road network to intercept unlawful vehicle flows traveling between origin-destination pairs, who in turn deviate from their route to avoid any encounter with such facilities.

- Under a branch-and-cut algorithm, the pulse algorithm is used in the separation algorithm for the fractional solutions modeled as a resource constrained shortest path problem in a transformed graph, where the resource on each arc is the actual distance in the road network and the length of the arc is a value between 0 and 1.

Agenda

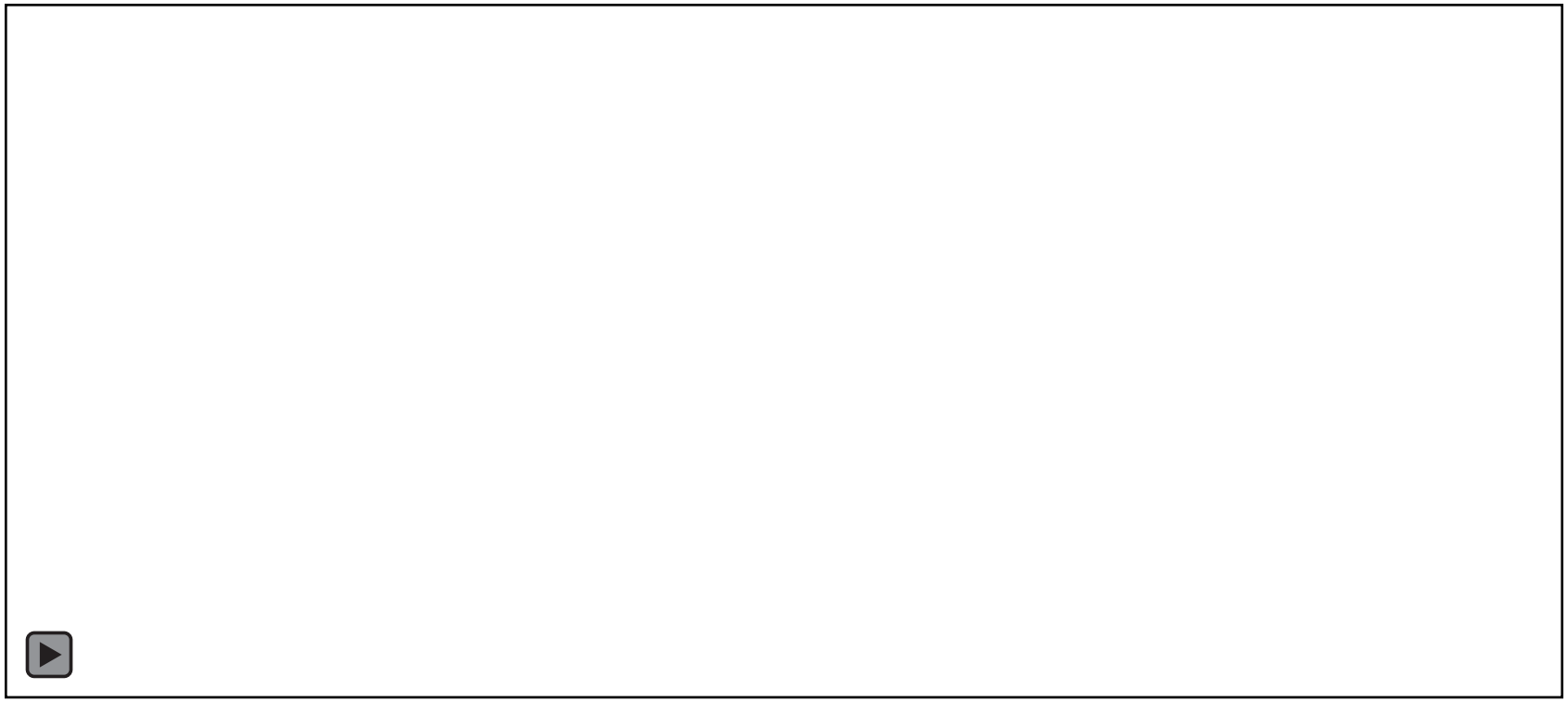
- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
- Part V: perspectives
  - Bidirectional pulse algorithm
  - α-Reliable shortest path
  - Least expected travel time path problem

Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
- Part V: perspectives
  - **Bidirectional pulse algorithm**
  - α-Reliable shortest path
  - Least expected travel time path problem

Bidirectional pulse algorithm

CSP motivation



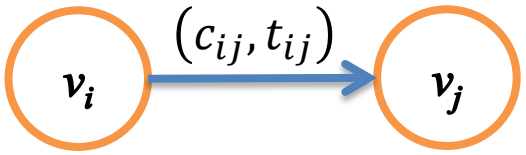
Bidirectional pulse algorithm

CSP problem statement

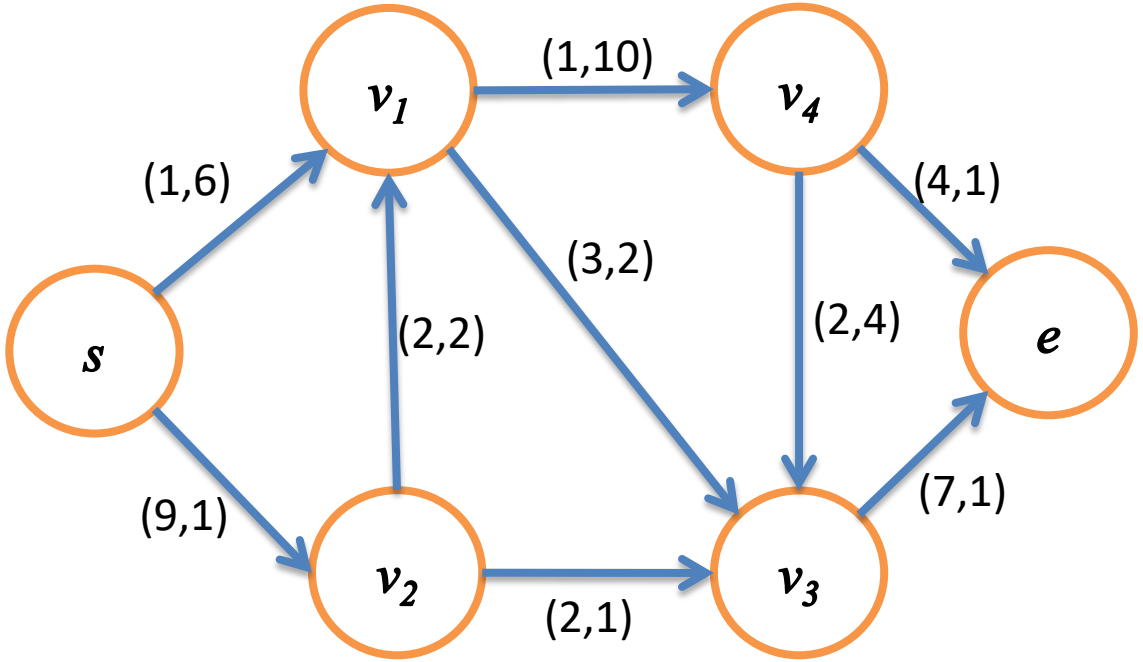
- The CSP is defined by:
  - Directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$
    - $\mathcal{N} = \{v_1, \dots, v_i, \dots, v_n\}$
    - $\mathcal{A} = \{(i,j) | v_i \in \mathcal{N}, v_j \in \mathcal{N},\ i \neq j\}$
  - Find a minimum cost path starting at node v_s and ending at node v_e
  - Nonnegative weights c_{ij} and t_{ij} are the cost and travel time of traversing arc $(i,j) \in \mathcal{A}$
  - Time constraint T

Bidirectional pulse algorithm

CSP example

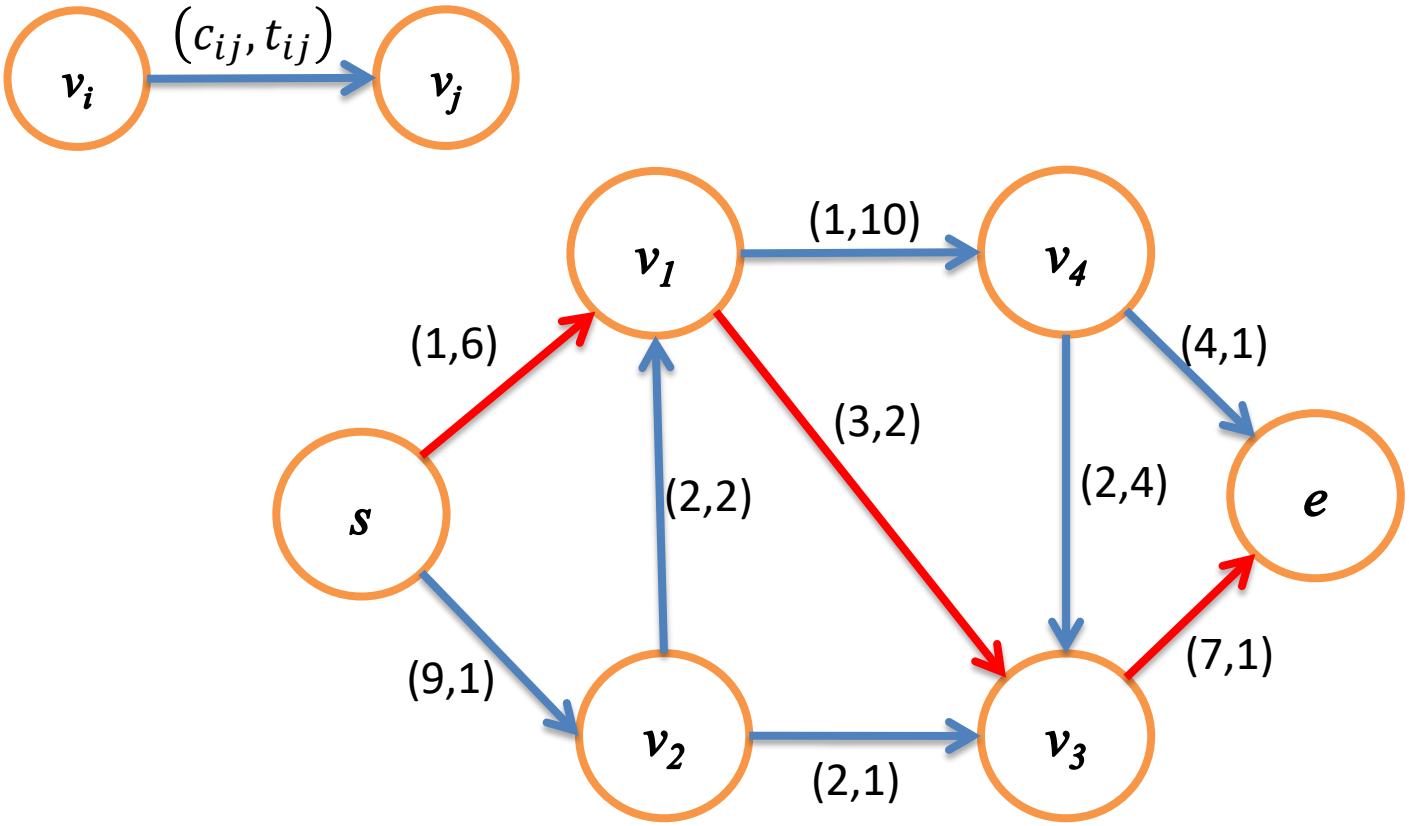


$T = 14$



Bidirectional pulse algorithm

CSP example



$T = 14$

$\mathcal{P} \leftarrow \{s, v_1, v_3, e\}$
$c(\mathcal{P}) = 11$
$t(\mathcal{P}) = 9$

Bidirectional pulse algorithm

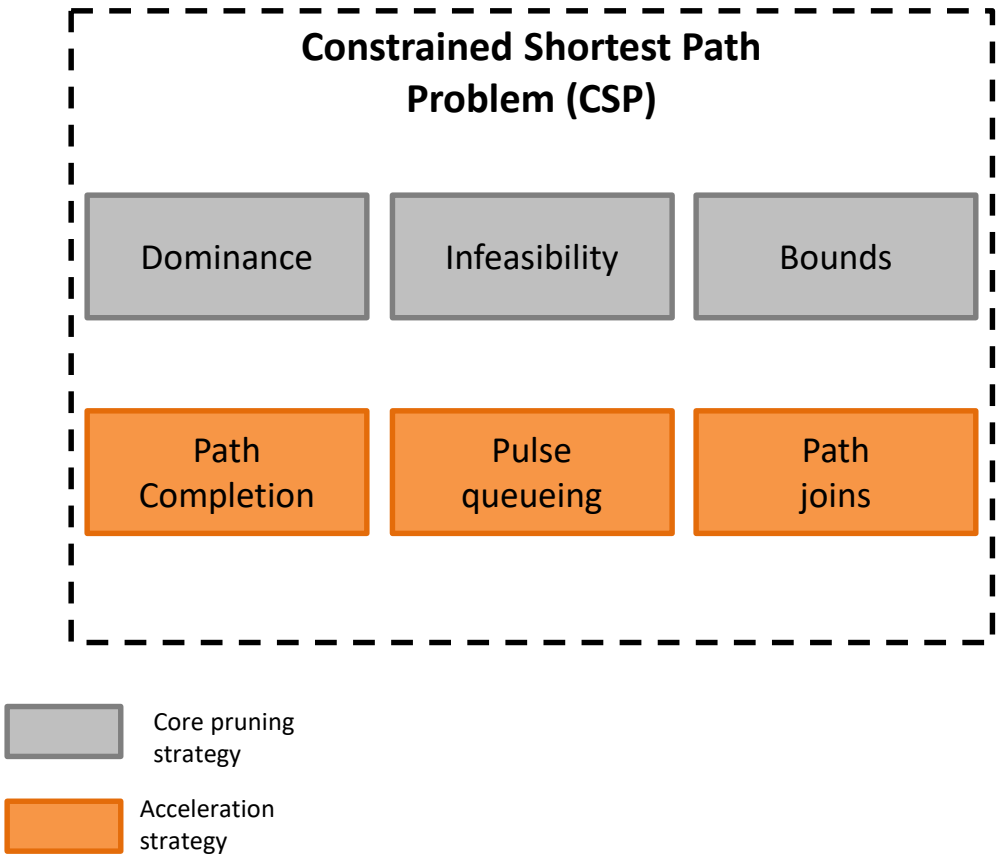
(Original) Pulse Algorithm for the CSP

- Since 2013, some new competitive algorithms appeared:
  - Sedeño-Noda & Alonso-Rodriguez (2015) - k-SP
  - Horváth & Kis (2016) – LP-based
  - **Thomas, Calogiuri & Hewitt (2019) – RC-BDA***

- Lessons learned:
  - The pulse algorithm has a strong **DFS behavior**.
  - Might explore **unpromising regions** of the search space before **backtracking**.
  - **Primal bound** updates lead to significant pruning improvements.
  - The pulse favors a **parallel** search.

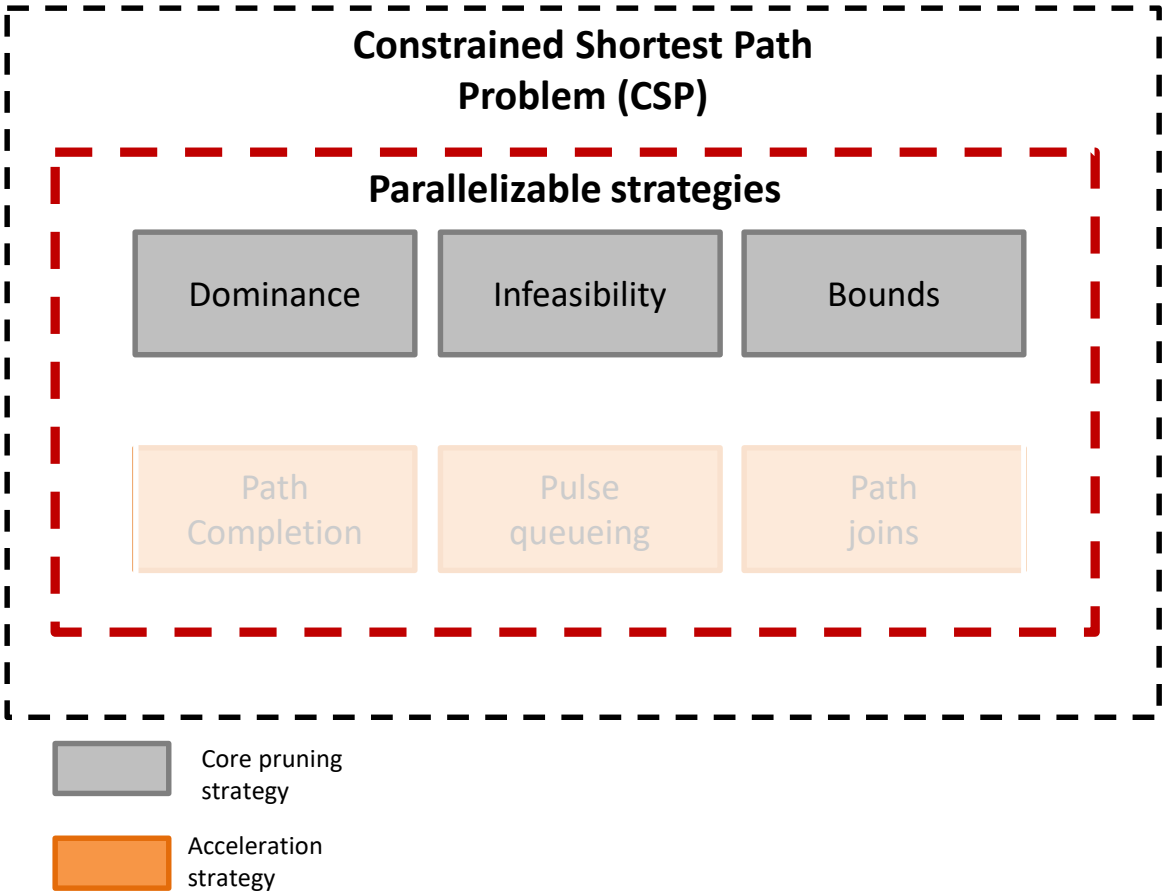
Bidirectional pulse algorithm

Acceleration strategies for the CSP



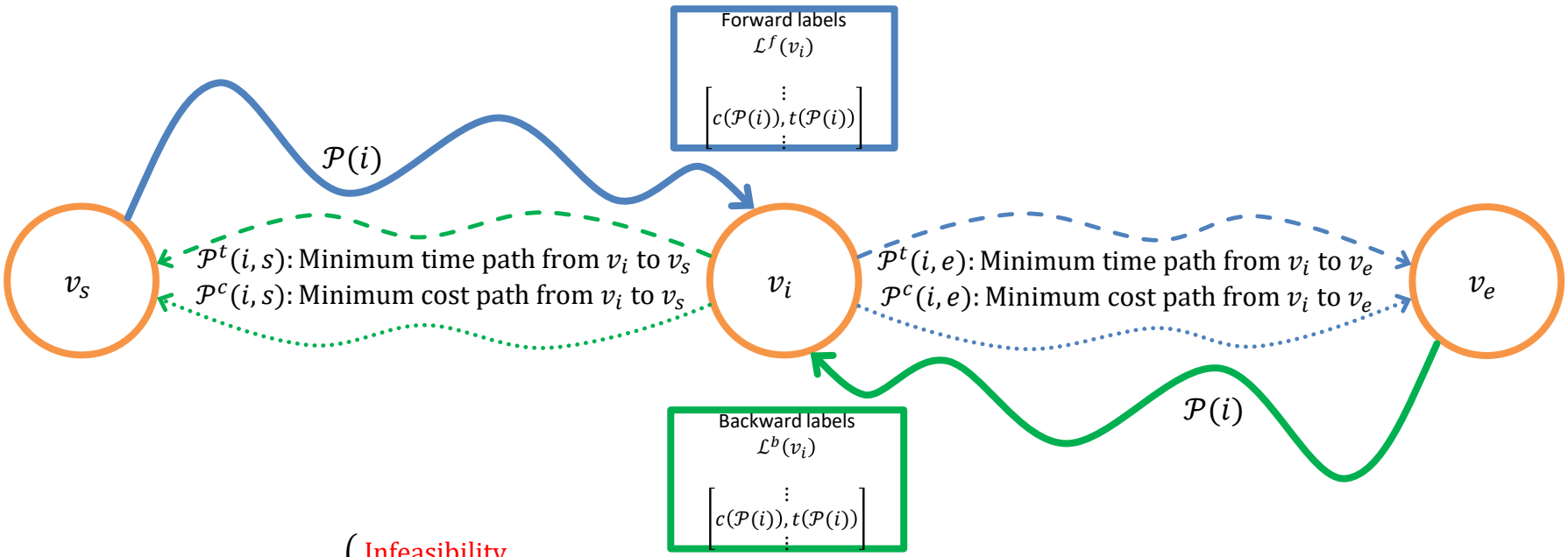
Bidirectional pulse algorithm

Parallelization



Bidirectional pulse algorithm

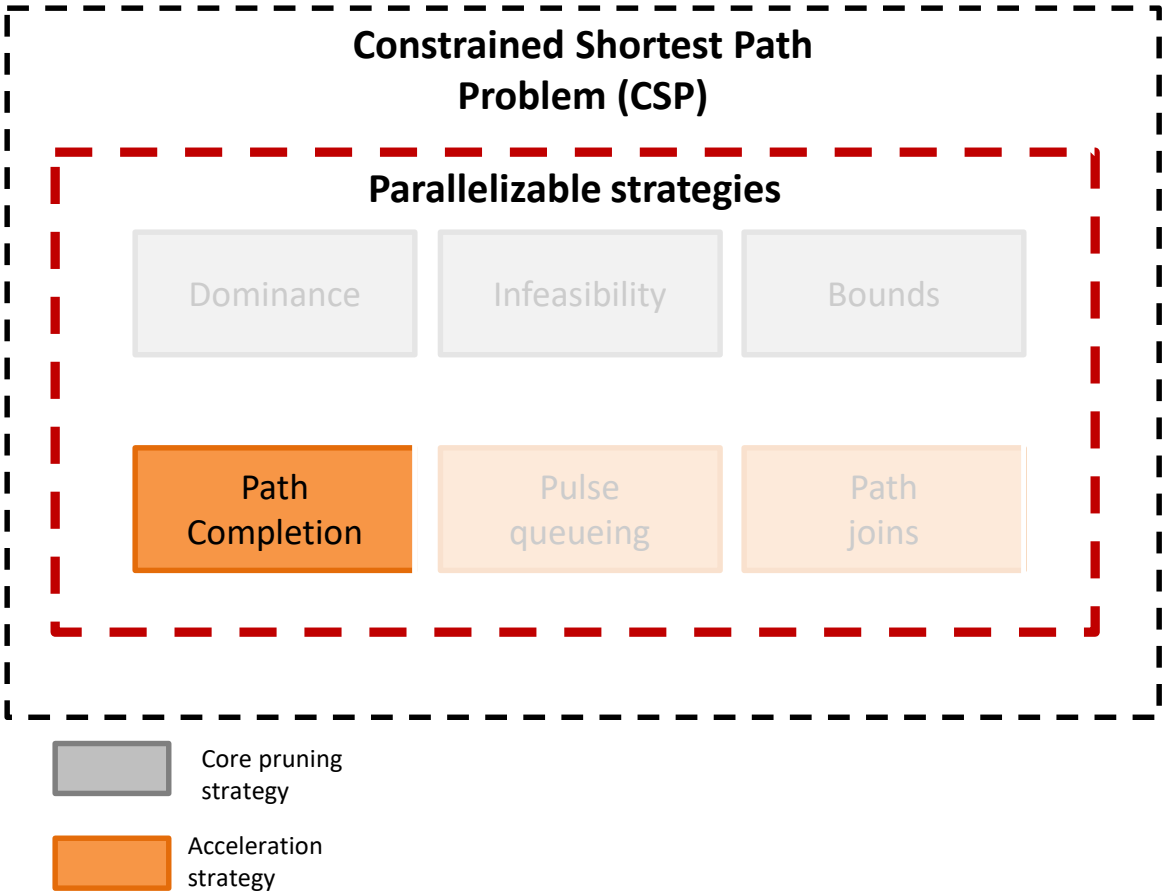
Parallelization



$\mathcal{P}(i)$ is pruned by { Infeasibility, Bounds, Dominance }

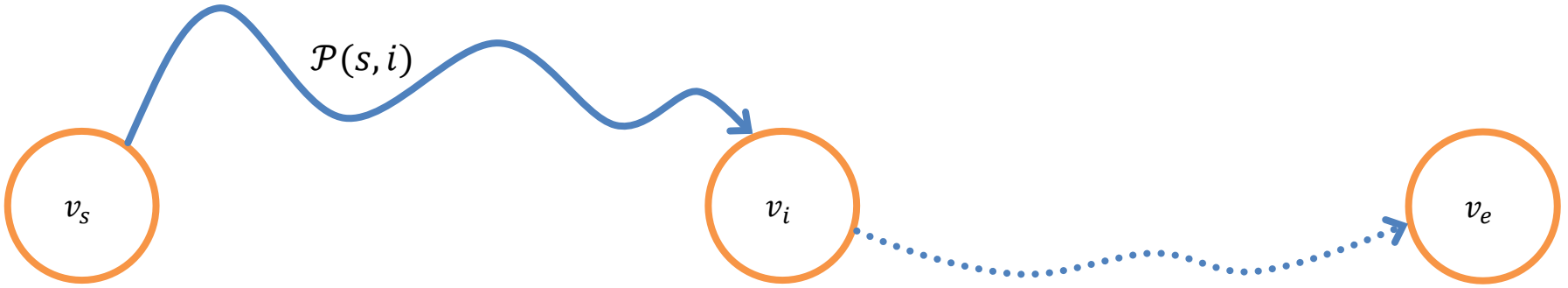
Bidirectional pulse algorithm

Path completion



Bidirectional pulse algorithm

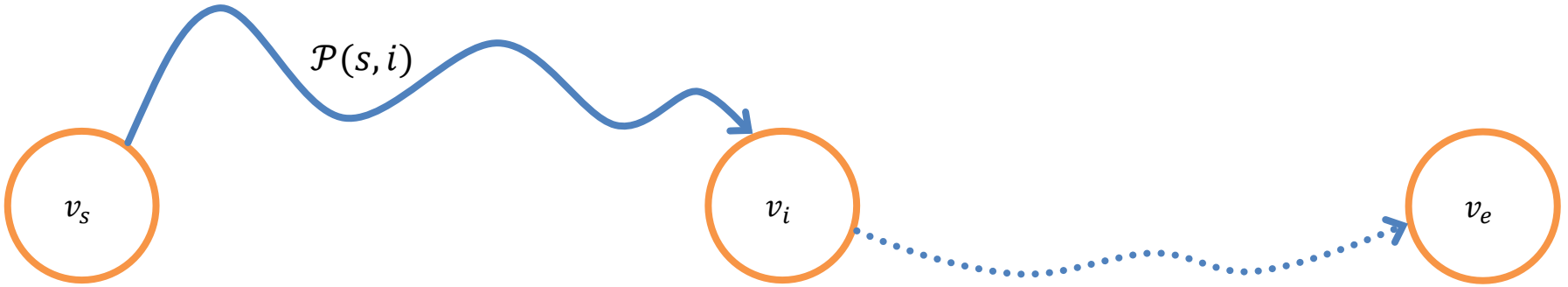
(Cost) Path completion



$\mathcal{P}^c(i,e)$: Minimum cost path from v_i to v_e

Bidirectional pulse algorithm

(Cost) Path completion

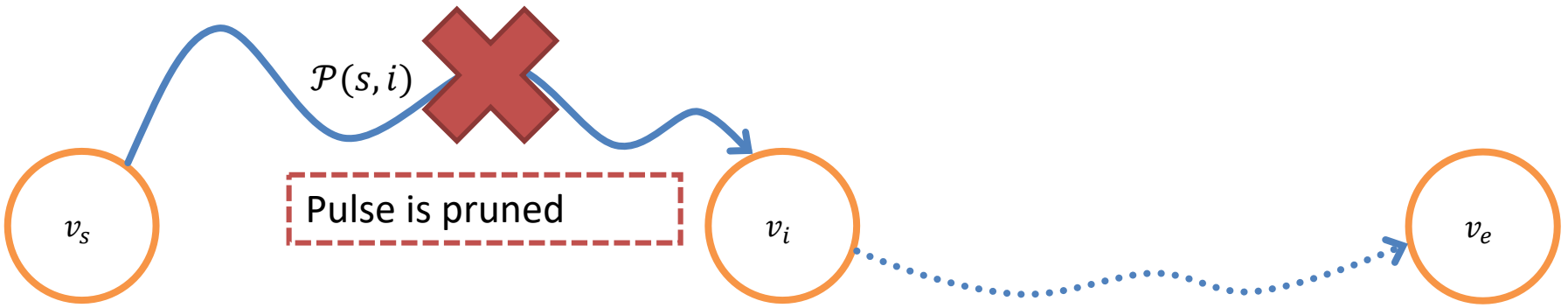


$\mathcal{P}^c(i,e)$: Minimum cost path from v_i to v_e

$$\text{if } c(\mathcal{P}(s,i)) + \boxed{c(\mathcal{P}^c(i,e))} < \overline{c} \ \wedge \ t\big(\mathcal{P}(s,i)\big) + \boxed{t(\mathcal{P}^c(i,e))} \leq T$$

Bidirectional pulse algorithm

(Cost) Path completion



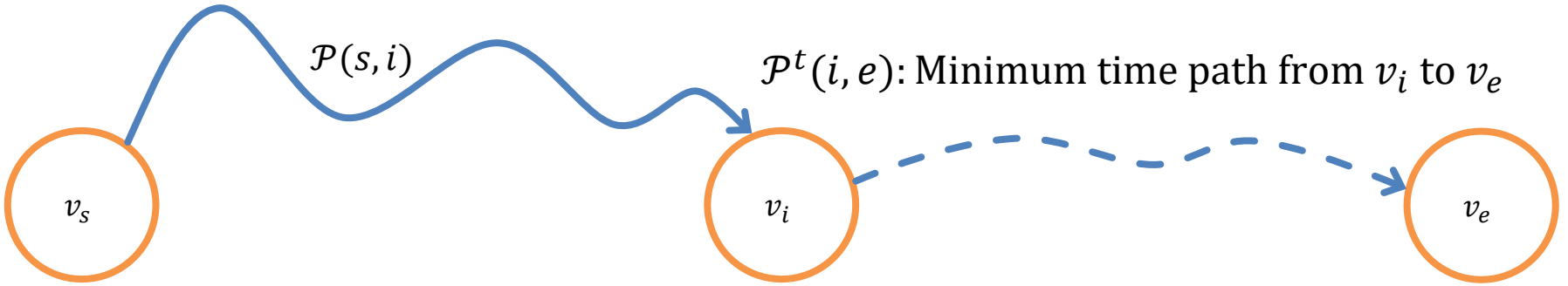
$\mathcal{P}^c(i,e)$: Minimum cost path from v_i to v_e

$$\text{if } c(\mathcal{P}(s,i)) + c(\mathcal{P}^c(i,e)) < \overline{c} \ \wedge \ t\big(\mathcal{P}(s,i)\big) + t(\mathcal{P}^c(i,e)) \leq T$$

- The primal bound $\overline{c}$ is updated
- The incoming path $\mathcal{P}(s,i)$ is pruned

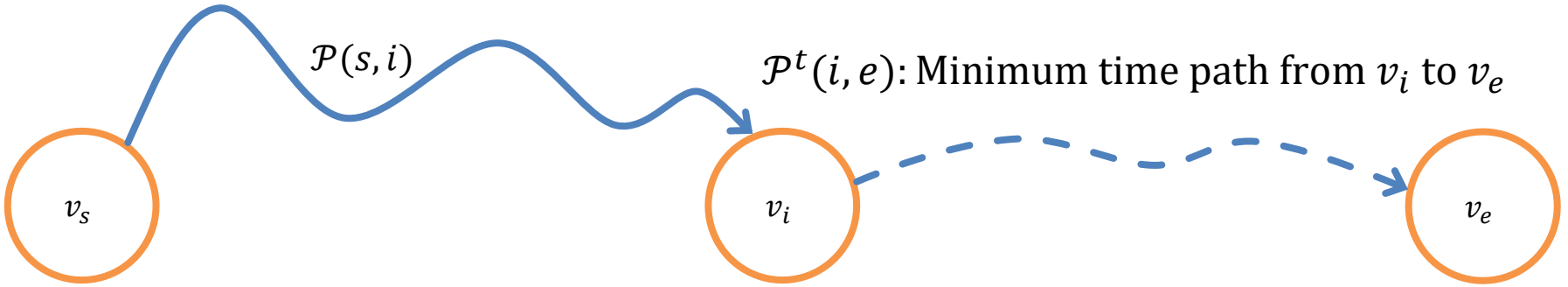
Bidirectional pulse algorithm

(Time) Path completion



Bidirectional pulse algorithm

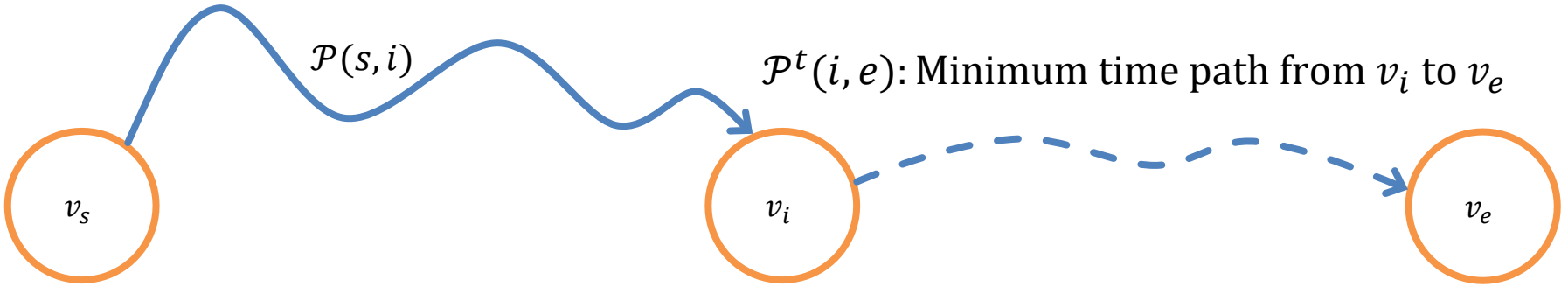
(Time) Path completion



$$\text{if } c(\mathcal{P}(s,i)) + \boxed{c(\mathcal{P}^t(i,e))} < \overline{c} \ \wedge \ t\big(\mathcal{P}(s,i)\big) + \boxed{t(\mathcal{P}^t(i,e))} \leq T$$

Bidirectional pulse algorithm

(Time) Path completion

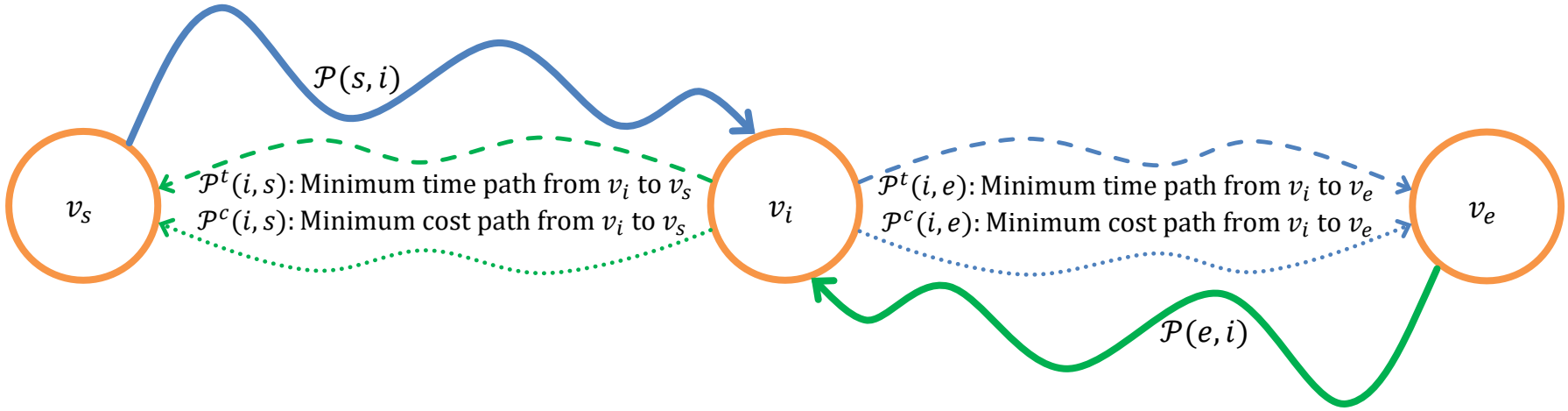


$$\text{if } c(\mathcal{P}(s,i)) + c(\mathcal{P}^t(i,e)) < \overline{c} \ \wedge \ t\big(\mathcal{P}(s,i)\big) + t(\mathcal{P}^t(i,e)) \leq T$$

- The primal bound $\overline{c}$ is updated
- The path $\mathcal{P}(s,i)$ cannot be pruned

Bidirectional pulse algorithm

Path completion



Cost-path completion

$$\text{if } c(\mathcal{P}(\alpha,i)) + c(\mathcal{P}^c(i,\omega)) < \overline{c} \ \wedge \ t\big(\mathcal{P}(\alpha,i)\big) + t(\mathcal{P}^c(i,\omega)) \leq T$$

- The primal bound $\overline{c}$ is updated
- The incoming path $\mathcal{P}(\alpha,i)$ is pruned

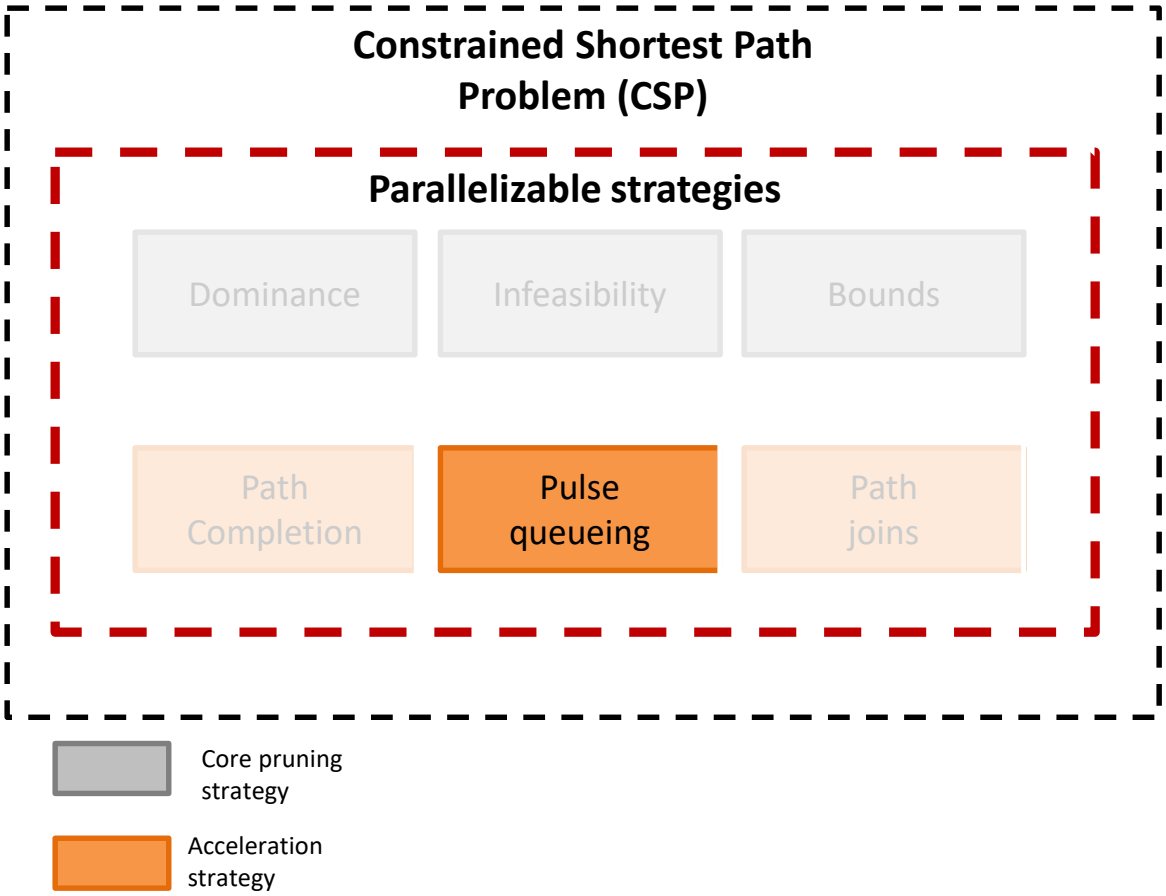
Time-path completion

$$\text{if } c(\mathcal{P}(\alpha,i)) + c(\mathcal{P}^t(i,\omega)) < \overline{c} \ \wedge \ t\big(\mathcal{P}(\alpha,i)\big) + t(\mathcal{P}^t(i,\omega)) \leq T$$

- The primal bound $\overline{c}$ is updated

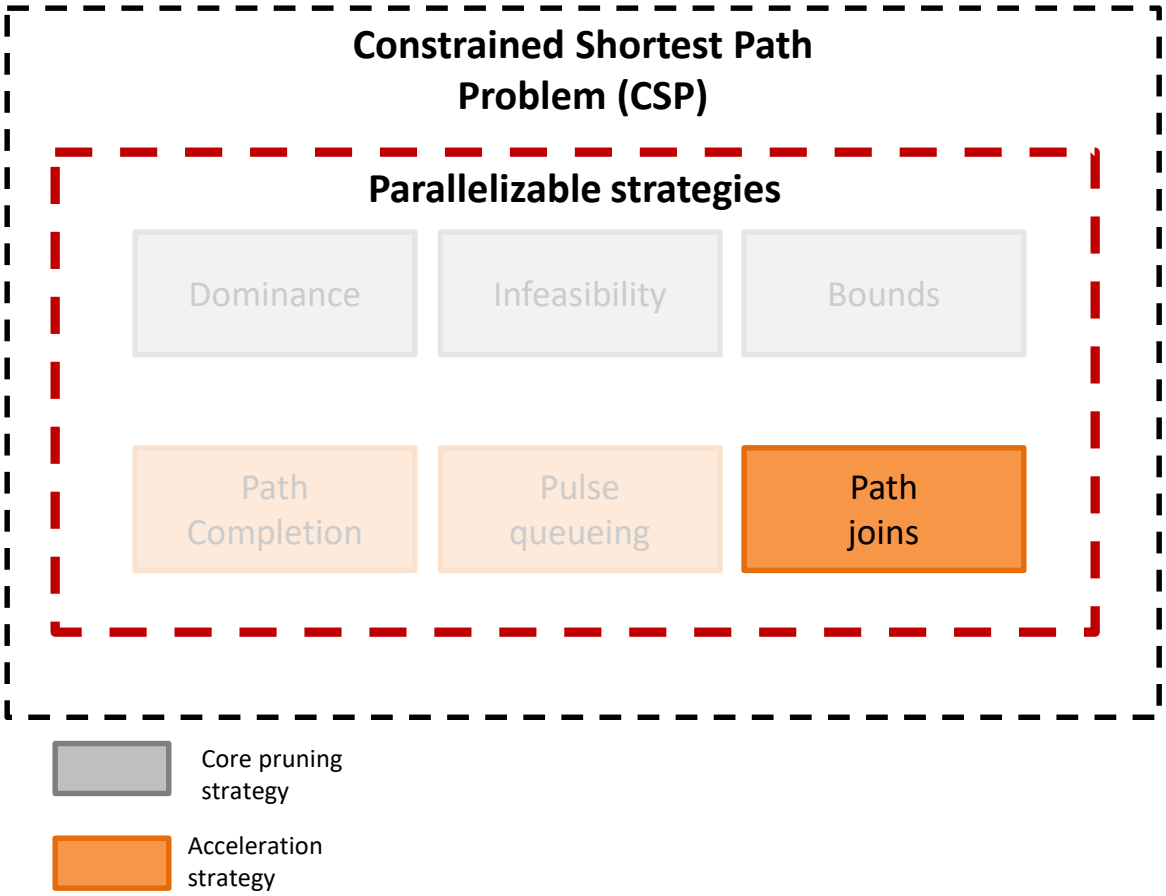
Bidirectional pulse algorithm

Pulse queueing



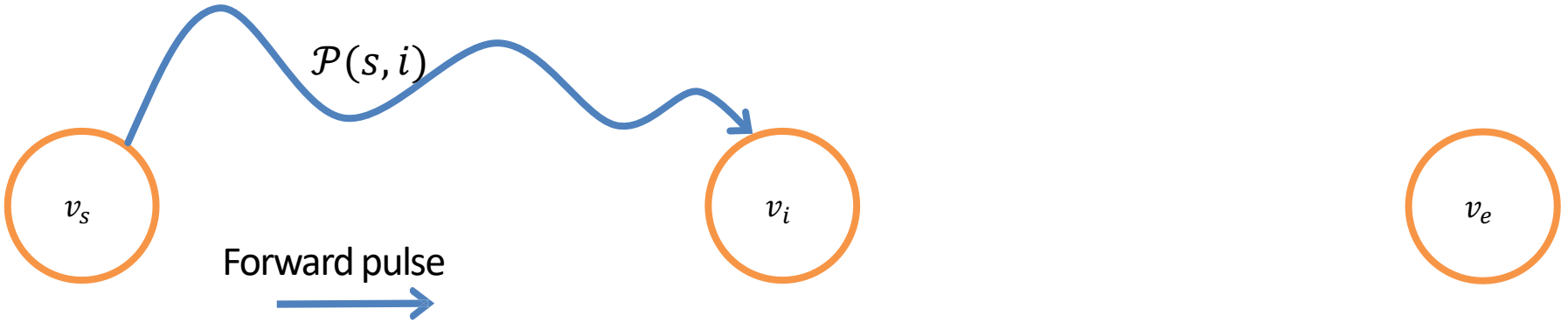
Acceleration strategies for the CSP

Path joins



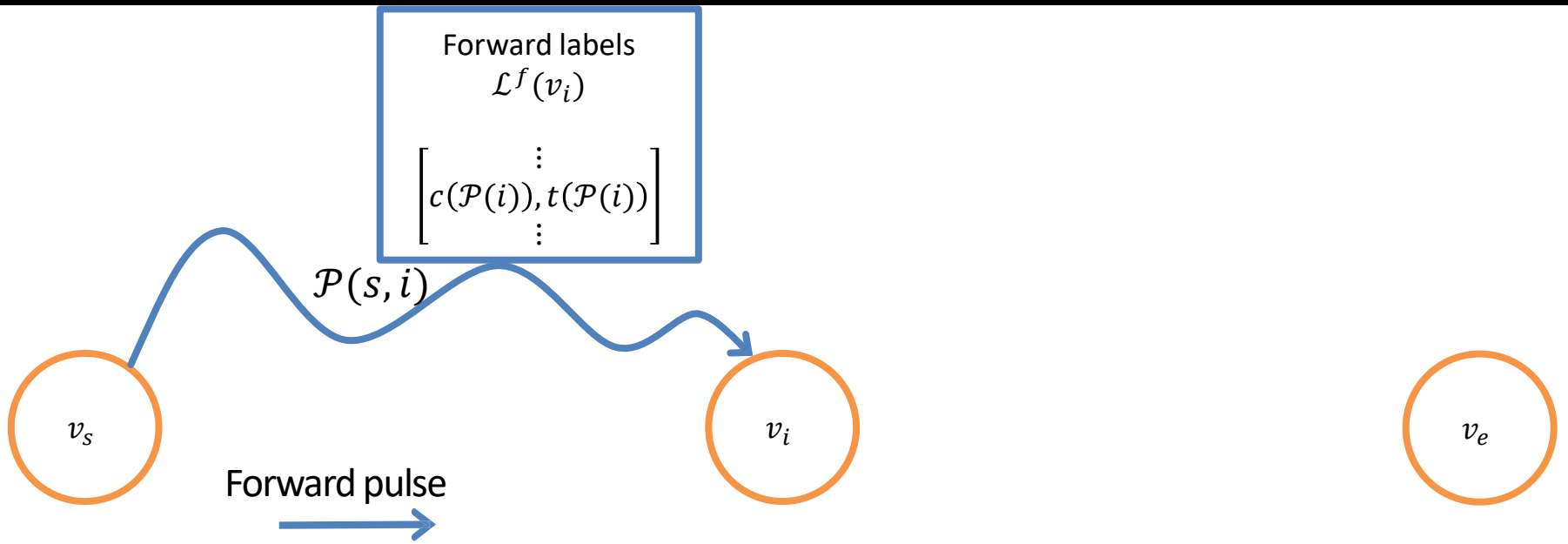
Bidirectional pulse algorithm

Path joins



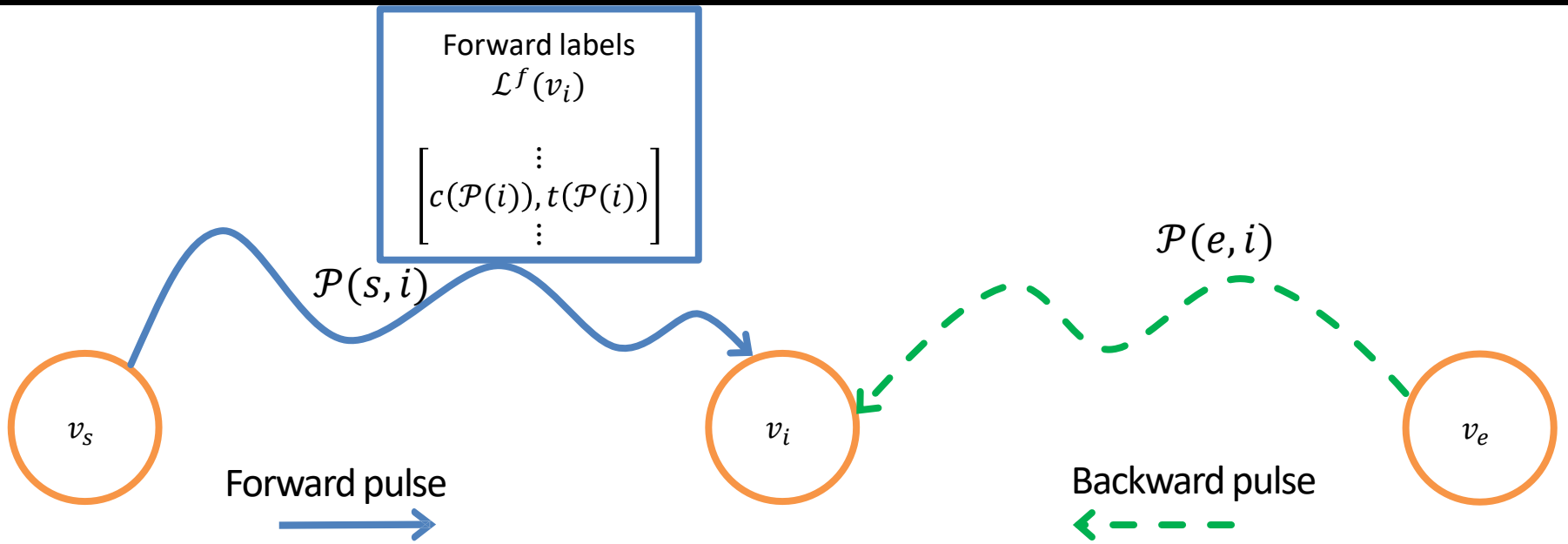
Bidirectional pulse algorithm

Path joins



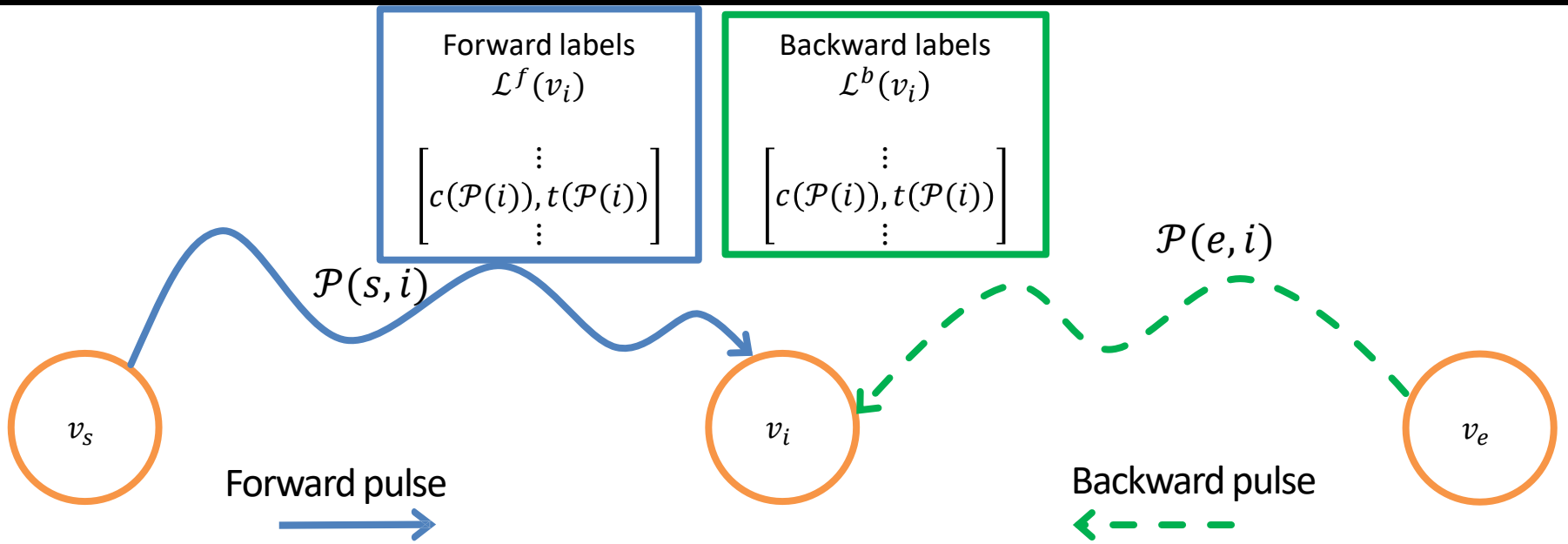
Bidirectional pulse algorithm

Path joins



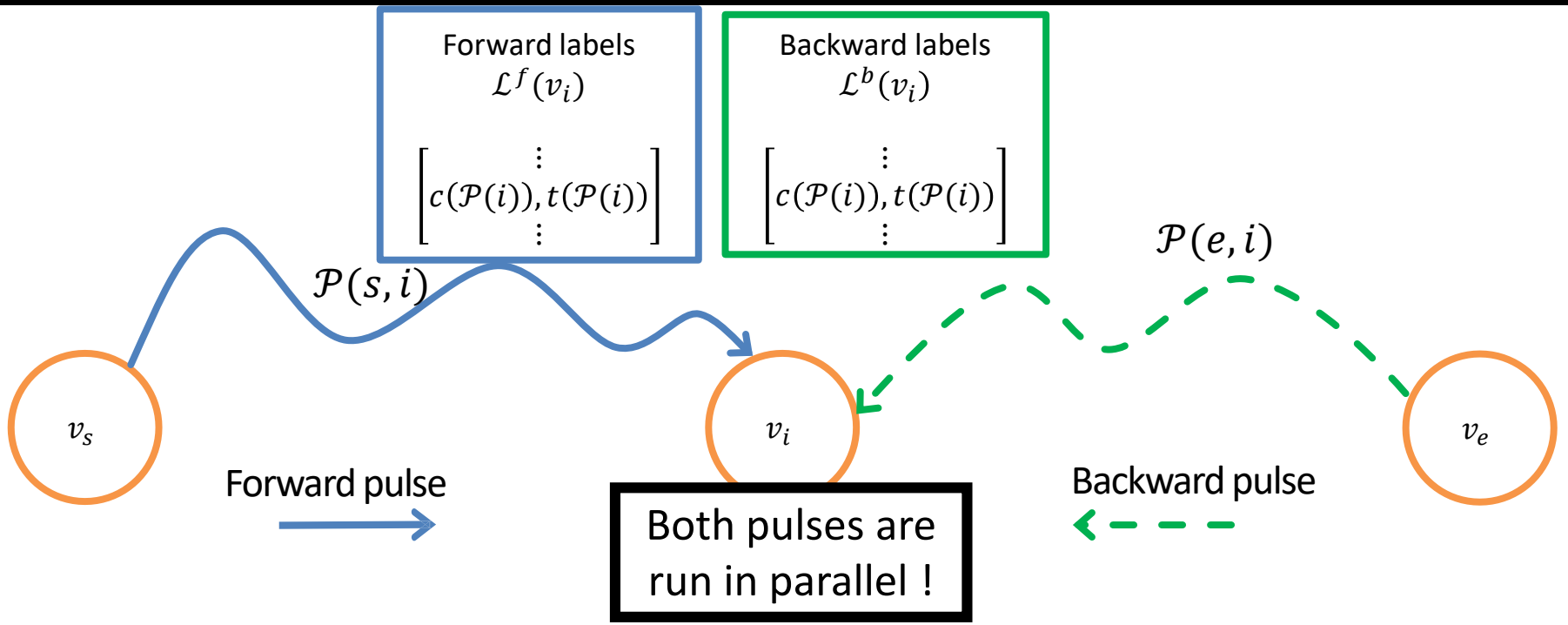
Bidirectional pulse algorithm

Path joins



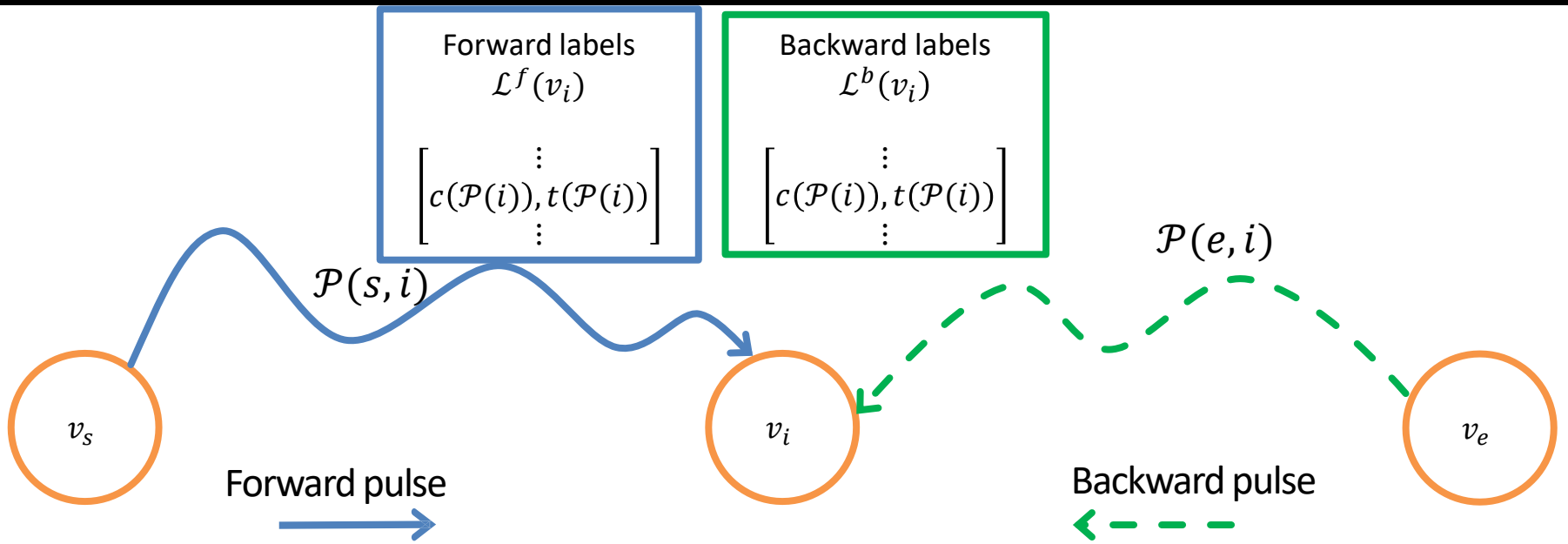
Bidirectional pulse algorithm

Path joins



Bidirectional pulse algorithm

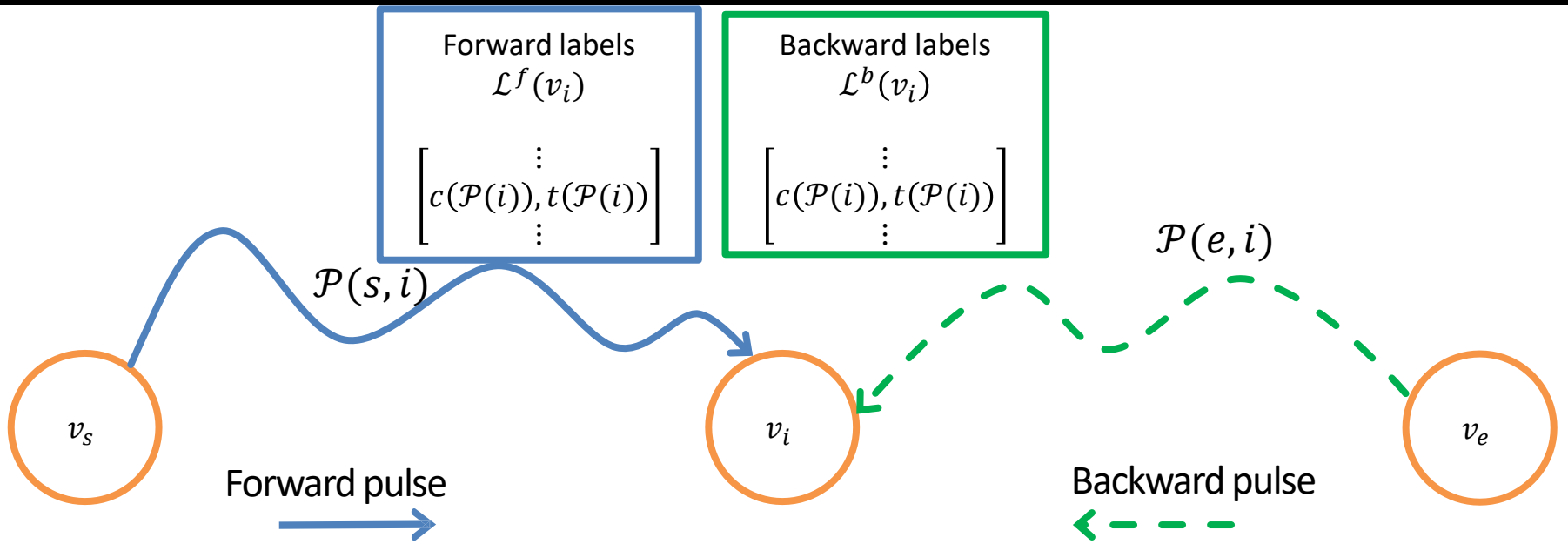
Path joins



$$\mathcal{P}(s,e) = \mathcal{P}(s,i) \cup \mathbf{reverse}(\mathcal{P}(e,i))$$

Bidirectional pulse algorithm

Path joins



$$\mathcal{P}(s,e) = \mathcal{P}(s,i) \cup \mathbf{reverse}(\mathcal{P}(e,i))$$

$$\text{if } c(\mathcal{P}(s,e)) < \overline{c} \ \wedge \ t\big(\mathcal{P}(s,e)\big) \leq T$$

- The primal bound $\overline{c}$ is updated

Bidirectional pulse algorithm

Computational experiments: instances

- 360 instances from the 9th DIMACS challenge
  - http://users.diag.uniroma1.it/challenge9/download.shtml

- Each instance is a combination of:
  - US-road network (nine)
  - Destination (five)
  - Tightness factor (p) between 0.1 (tight) and 0.8 (loose) (eight)

Bidirectional pulse algorithm

Instances: network size

	Road network	Nodes	Arcs
BAY	San Francisco Bay Area	321,270	800,172
NY	New York City	264,346	733,846
COL	Colorado	435,666	1,057,066
FLA	Florida	1,070,376	2,712,798
NE	Northeast USA	1,524,453	3,897,636
CAL	California and Nevada	1,890,815	4,657,742
LKS	Great Lakes	2,758,119	6,885,658
E	Eastern USA	3,598,623	8,778,114
W	Western USA	6,262,104	15,248,146

Bidirectional pulse algorithm

Computational experiments: setup

- Benchmark algorithms proposed by:
  - Lozano & Medaglia (2013) (labeled by “PA”)
  - Thomas et al. (2019) (labeled by “RC-BDA”)
- RC-BDA times are scaled based on the LINPACK benchmark
- BP and PA coded in Java and compiled in Eclipse SDK 4.8.0
- CPU: Intel core i7-4610M @3.00 GHz 8GB RAM for the JVM
- The computational time limit is 14,400 seconds
- The amount of labels is 3
- The depth limit is 2

Bidirectional pulse algorithm

Average computing time and instances solved

Road network	PA		RC-BDA		BP	
	Average time (s)	Solved	Average time (s)	Solved	Average time(s)	Solved
BAY	25.57	40/40	1.27	40/40	0.30	40/40
NY	100.82	40/40	0.59	40/40	0.23	40/40
COL	3.99	36/40	4.60	40/40	1.67	40/40
FLA	120.72	24/40	62.32	40/40	2.64	40/40
NE	16.21	35/40	4.06	40/40	1.24	40/40
CAL	21.90	28/40	643.87	40/40	109.93	35/40
LKS	5.08	24/40	816.74	37/40	319.26	36/40
E	9.30	29/40	65.08	40/40	13.63	40/40
W	1.01	16/40	1,063.44	40/40	139.59	38/40
		272/360		357/360		349/360

*Average time on solved instances

Bidirectional pulse algorithm

Speedups

Road network	PA			RC-BDA		
	Arithmetic mean of speedups	Geometric mean of speedups	BP Wins	Arithmetic mean of speedups	Geometric mean of speedups	BP Wins
BAY	27.88	1.03	9/40	4.42	2.89	36/40
NY	415.34	3.93	18/40	2.62	2.33	38/40
COL	126.58	1.37	7/40	3.61	2.79	40/40
FLA	3117.64	69.33	24/40	25.82	8.35	40/40
NE	1106.97	3.56	16/40	3.40	3.07	40/40
CAL	371.62	3.81	19/40	12.59	4.76	35/40
LKS	321.56	5.53	20/40	12.03	5.16	35/40
E	553.00	4.82	13/40	5.13	2.20	34/40
W	719.34	23.40	22/40	37.59	7.19	37/40
Overall avg.	751.10	12.97	148/360	11.91	4.30	335/360

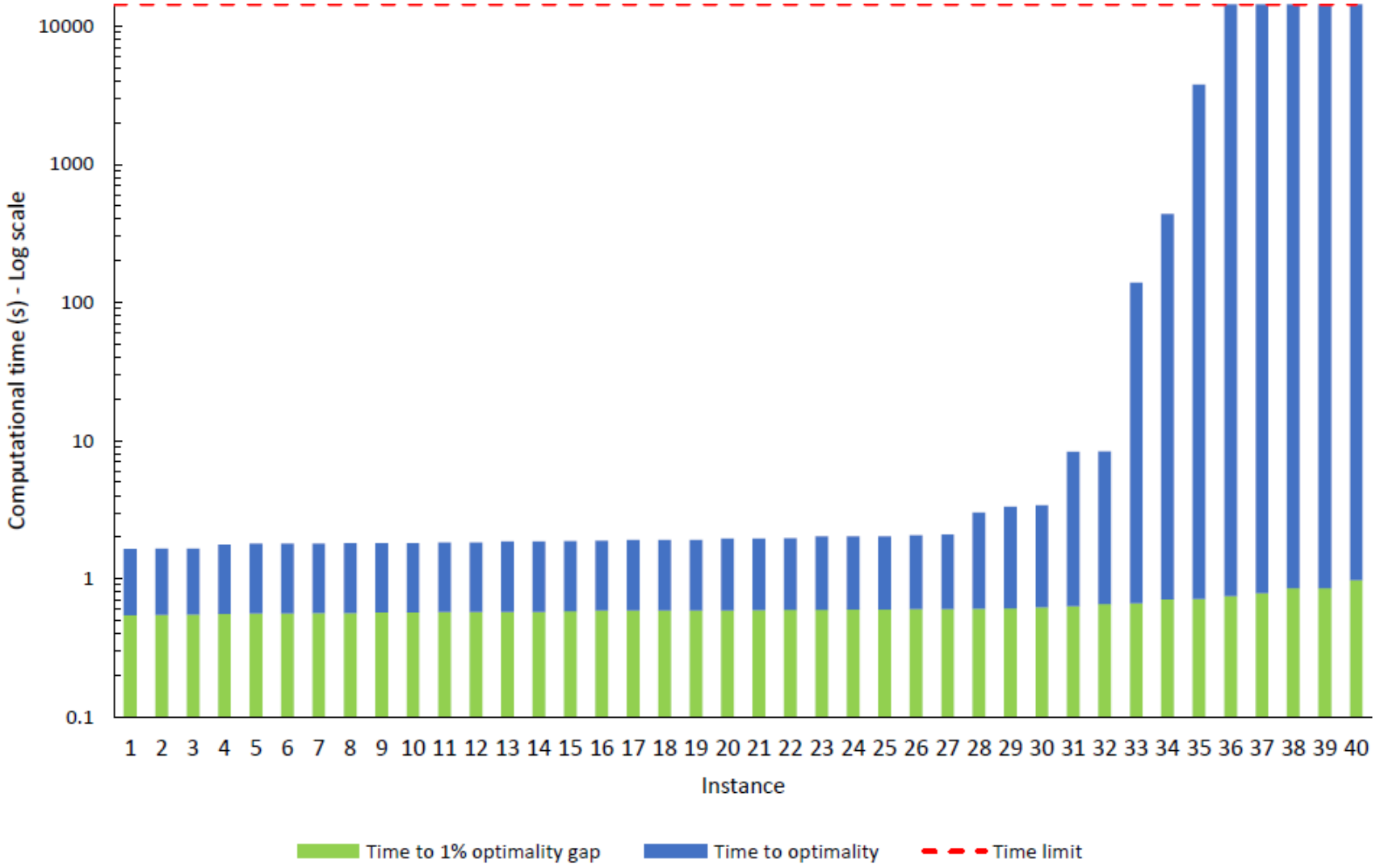
Bidirectional pulse algorithm

Speedups

Road network	PA			RC-BDA		
	Arithmetic mean of speedups	Geometric mean of speedups	BP Wins	Arithmetic mean of speedups	Geometric mean of speedups	BP Wins
BAY	27.88	1.03	9/40	4.42	2.89	36/40
NY	415.34	3.93	18/40	2.62	2.33	38/40
COL	126.58	1.37	7/40	3.61	2.79	40/40
FLA	3117.64	69.33	24/40	25.82	8.35	40/40
NE	1106.97	3.56	16/40	3.40	3.07	40/40
CAL	371.62	3.81	19/40	12.59	4.76	35/40
LKS	321.56	5.53	20/40	12.03	5.16	35/40
E	553.00	4.82	13/40	5.13	2.20	34/40
W	719.34	23.40	22/40	37.59	7.19	37/40
Overall avg.	751.10	12.97	148/360	11.91	4.30	335/360

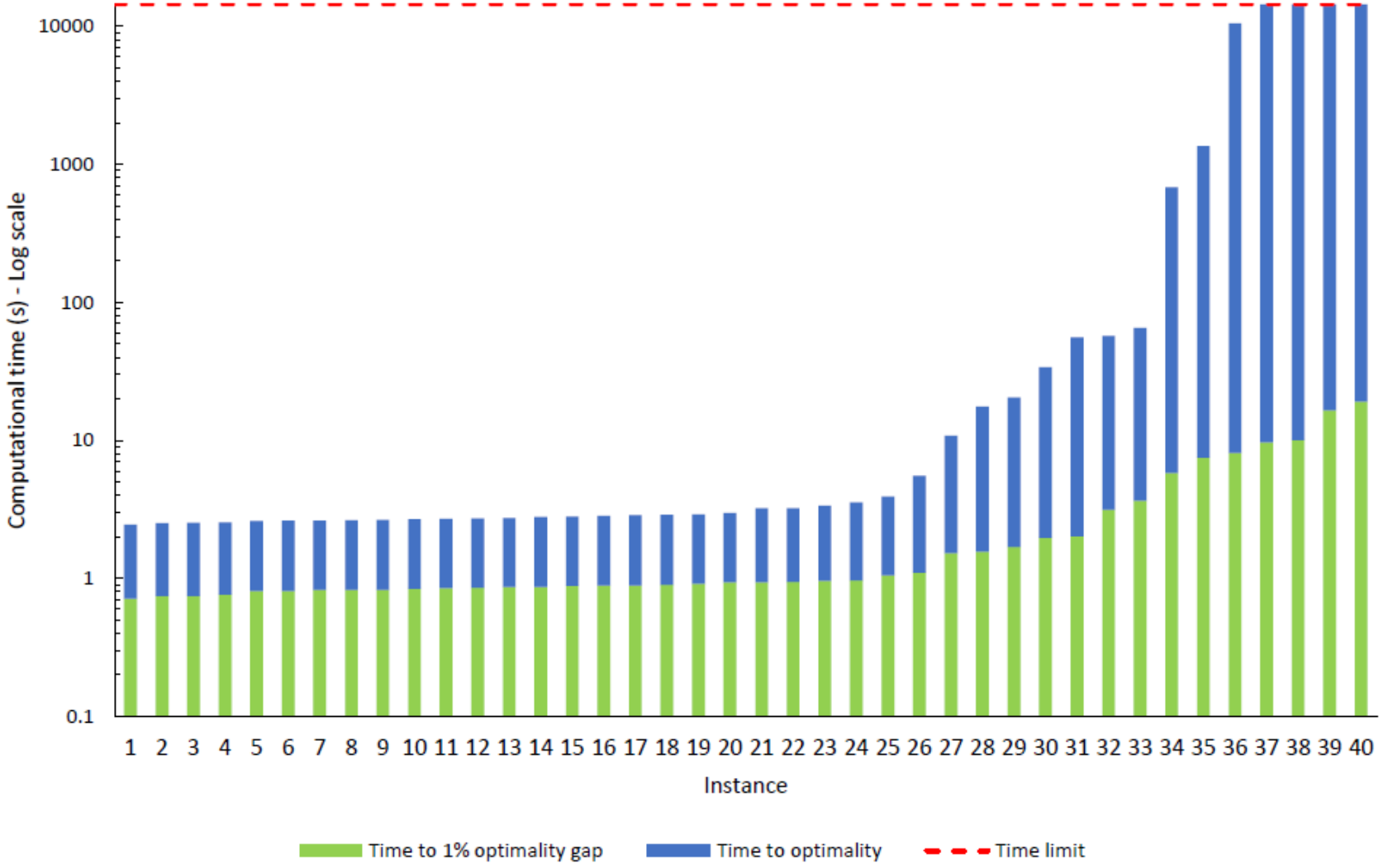
Bidirectional pulse algorithm

Pulse-based heuristic – CAL instances



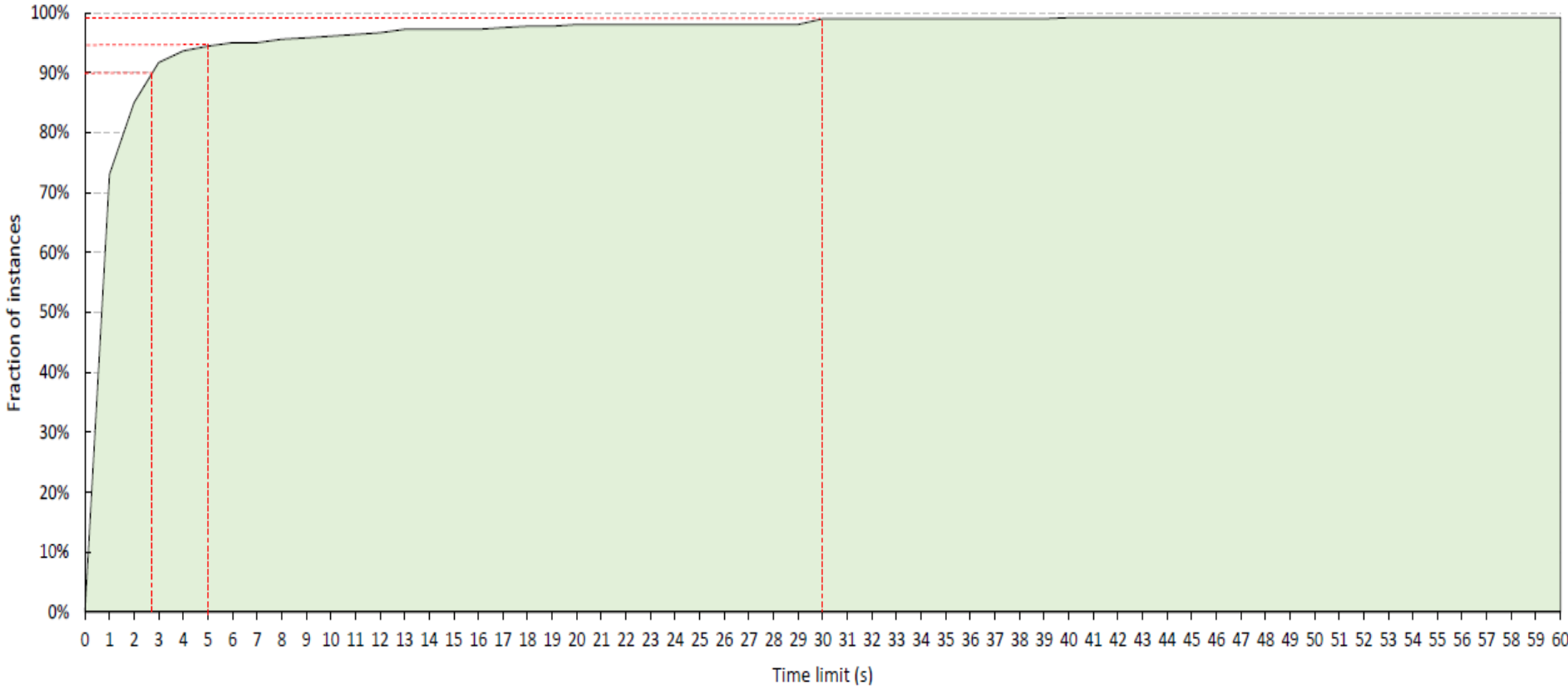
Bidirectional pulse algorithm

Pulse-based heuristic – LKS instances



Bidirectional pulse algorithm

Pulse-based heuristic – Time budget for 1%-optimality



90% instances in less than 3s
95% instances in less than 5s
99% instances in less than 30s

Bidirectional pulse algorithm

Concluding remarks

- We presented an exact algorithm for the CSP based on a bidirectional adjustable (depth-first or breadth-first) search strategy leveraged on parallelism and a fixed set of labels.
- Improves upon the original pulse algorithm, showing a robust performance and better scalability.
- The bidirectional pulse shows speedups against benchmark algorithms from the literature on large-sized networks.
- We presented an alternative pulse-based heuristic that quickly finds near optimal solutions and shows great potential for column generation.

Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
- Part V: perspectives
  - Bidirectional pulse algorithm
  - **α-Reliable shortest path**
  - Least expected travel time path problem

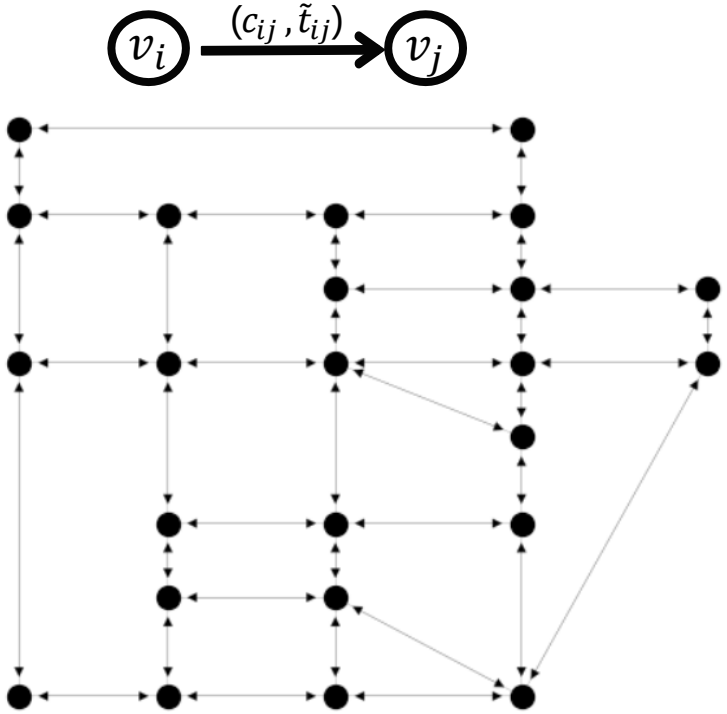
α-Reliable shortest path problem

Problem statement

- The Chance Constraint Shortest Path problem (CCSP) is defined by:
  - Directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$
  - $\mathcal{N} = \{v_1, \dots v_i, \dots, v_n\}$
  - $\mathcal{A} \subseteq \{(i,j) | v_i \in \mathcal{N}, v_j \in \mathcal{N}, i \neq j\}$

- Each arc $(i,j) \in \mathcal{A}$ has a cost c_{ij} and a time represented by the random variable $\tilde{t}_{ij}$.

- The CCSP consists of finding the minimum cost path $\mathcal{P}$ between a start node $v_s \in \mathcal{N}$ and an end node $v_e \in \mathcal{N}$



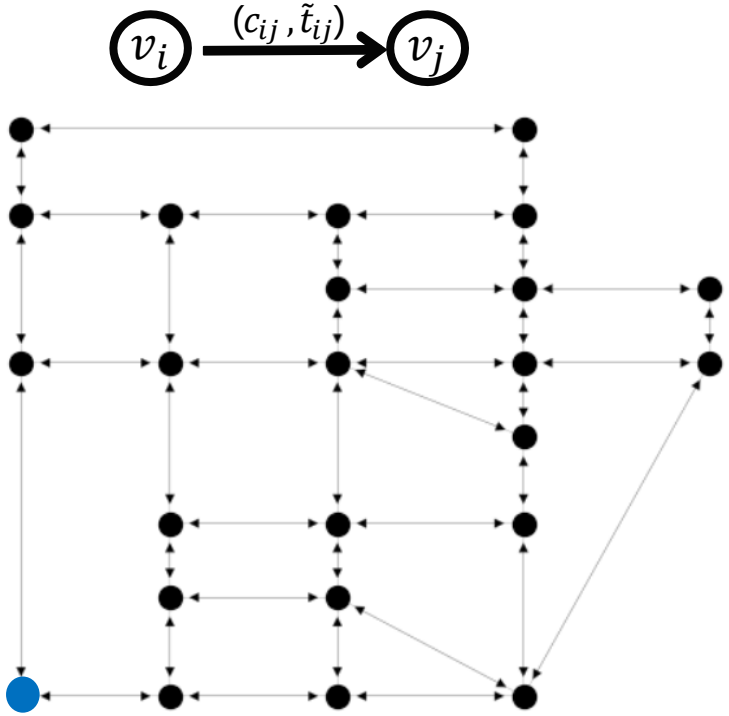
α-Reliable shortest path problem

Problem statement

- The Chance Constraint Shortest Path problem (CCSP) is defined by:
  - Directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$
  - $\mathcal{N} = \{v_1, \dots v_i, \dots, v_n\}$
  - $\mathcal{A} \subseteq \{(i,j) | v_i \in \mathcal{N}, v_j \in \mathcal{N}, i \neq j\}$

- Each arc $(i,j) \in \mathcal{A}$ has a cost c_{ij} and a time represented by the random variable $\tilde{t}_{ij}$.

- The CCSP consists of finding the minimum cost path $\mathcal{P}$ between a **start node $v_s \in \mathcal{N}$** and an end node $v_e \in \mathcal{N}$



α-Reliable shortest path problem

Problem statement

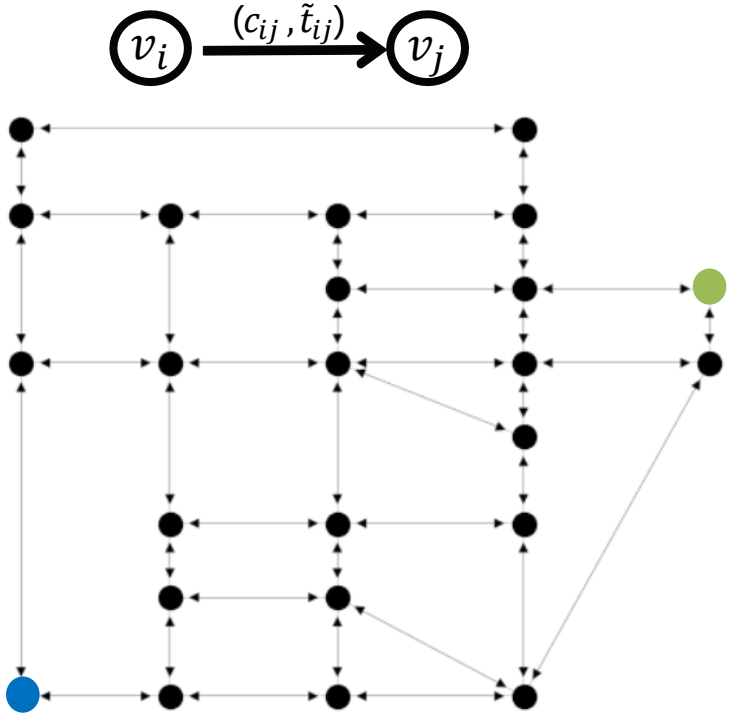
- The Chance Constraint Shortest Path problem (CCSP) is defined by:
  - Directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$
  - $\mathcal{N} = \{v_1, \dots v_i, \dots, v_n\}$
  - $\mathcal{A} \subseteq \{(i,j) | v_i \in \mathcal{N}, v_j \in \mathcal{N}, i \neq j\}$

- Each arc $(i,j) \in \mathcal{A}$ has a cost c_{ij} and a time represented by the random variable $\tilde{t}_{ij}$.

- The CCSP consists of finding the minimum cost path $\mathcal{P}$ between a **start node $v_s \in \mathcal{N}$** and an **end node $v_e \in \mathcal{N}$**

- While that the probability that the time of the path is less or equal than the maximum time T is greater than a reliability threshold α

$$\min_{\mathcal{P} \in \Omega_s^e} c(\mathcal{P}) \qquad \mathbb{P}(\tilde{t}(\mathcal{P}) \leq T) \geq \alpha$$



α-Reliable shortest path problem

Problem statement

- The Chance Constraint Shortest Path problem (CCSP) is defined by:
  - Directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$
  - $\mathcal{N} = \{v_1, \dots v_i, \dots, v_n\}$
  - $\mathcal{A} \subseteq \{(i,j) | v_i \in \mathcal{N}, v_j \in \mathcal{N}, i \neq j\}$

- Each arc $(i,j) \in \mathcal{A}$ has a cost c_{ij} and a time represented by the random variable $\tilde{t}_{ij}$.

- The CCSP consists of finding the minimum cost path $\mathcal{P}$ between a start node $v_s \in \mathcal{N}$ and an end node $v_e \in \mathcal{N}$

- While that the probability that the time of the path is less or equal than the maximum time T is greater than a reliability threshold α

$$\min_{\mathcal{P} \in \Omega_s^e} c(\mathcal{P})$$

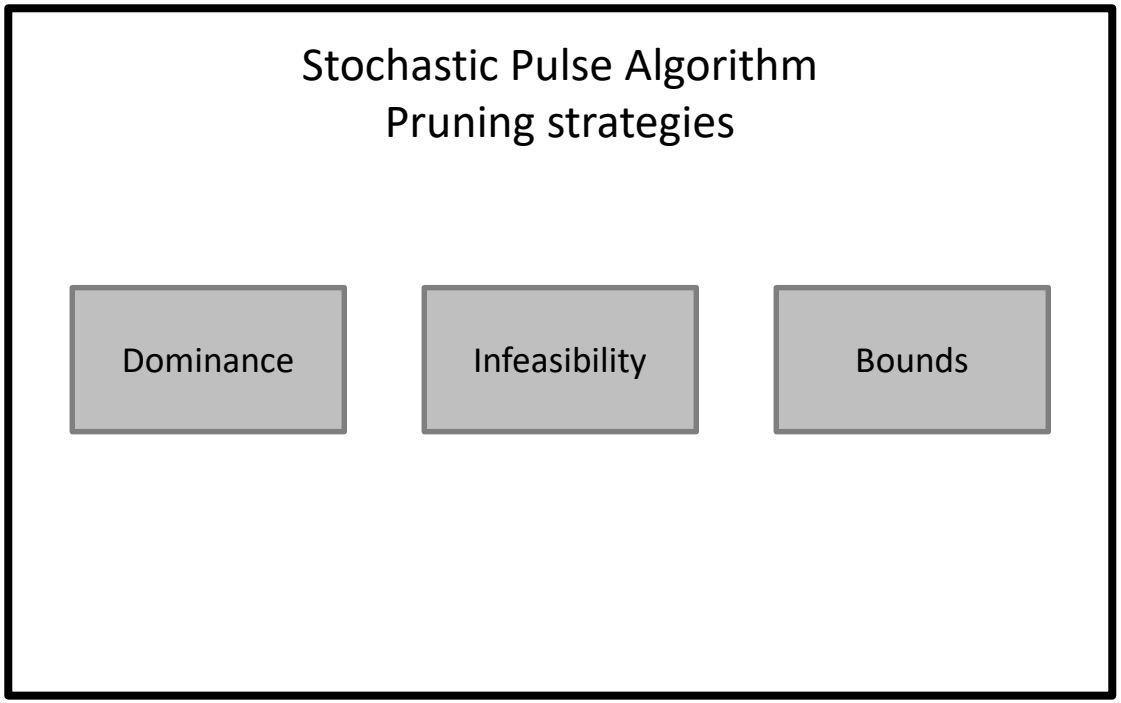
s.t.,

$$\mathbb{P}(\tilde{t}(\mathcal{P}) \leq T) \geq \alpha$$

$c(\mathcal{P})$: Cost of the path $\mathcal{P}$
$\tilde{t}(\mathcal{P})$: Travel time R.V of the path $\mathcal{P}$

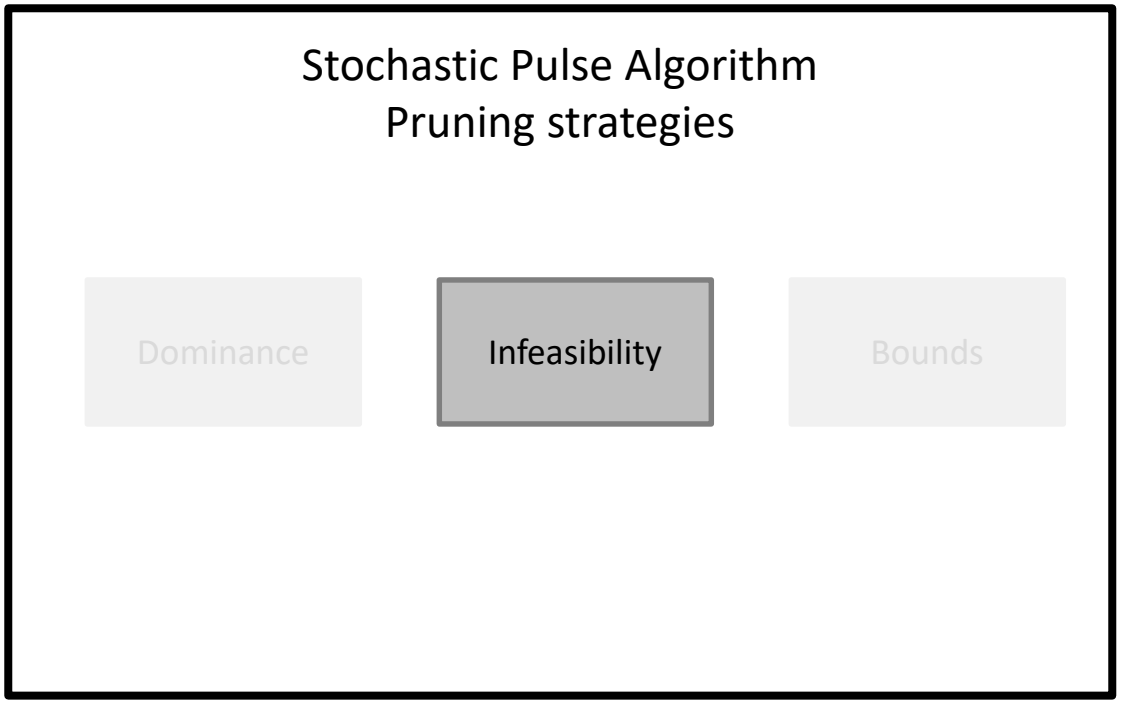
α-Reliable shortest path problem

Pruning strategies



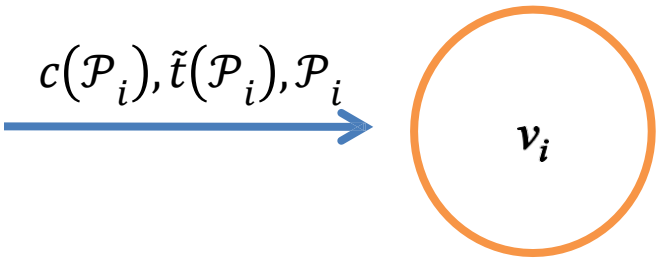
α-Reliable shortest path problem

Pruning strategies



α-Reliable shortest path problem

Infeasibility pruning



An incoming pulse to v_i is pruned if:

$$\pi(\mathcal{P}_i) < \alpha$$

$$\mathbb{P}\left(\tilde{t}(\mathcal{P}_i) < T\right) < \alpha$$

Black box

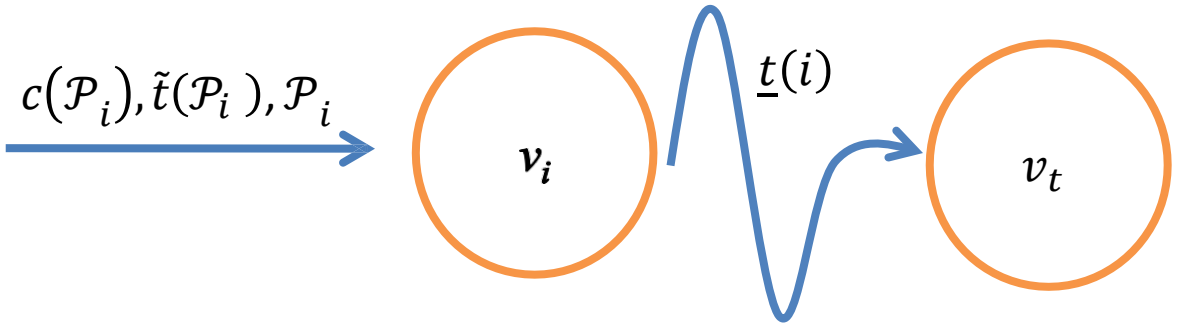
- Model properly $\tilde{t}_{ij}$
- Compute probabilities of convolution of $\tilde{t}_{ij}$

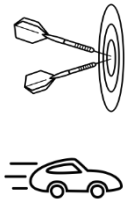
$$\tilde{t}(\mathcal{P}) = \sum_{(i,j)\in\mathcal{P}} \tilde{t}_{ij}$$

- Phase-type distributions.
- Monte Carlo simulation.

α-Reliable shortest path problem

Infeasibility pruning





Best scenario

Fastest route from v_i to v_e

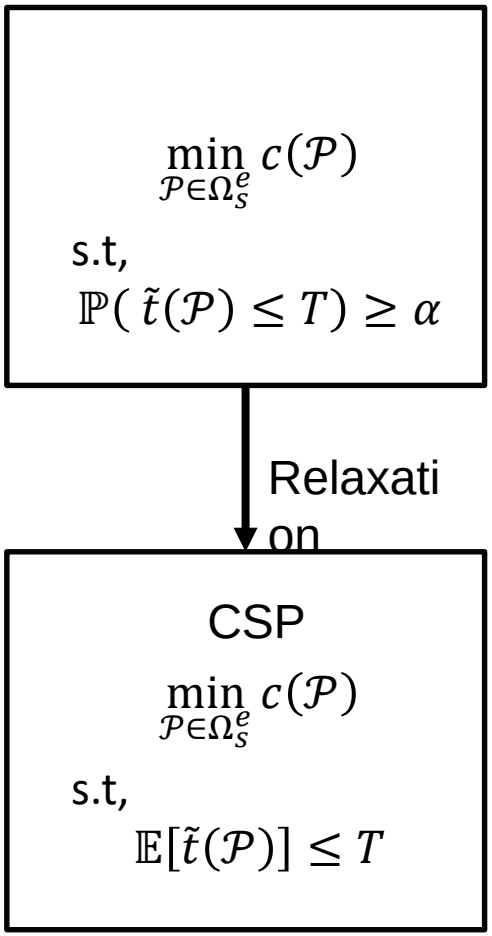
Now, an incoming pulse to v_i is pruned if:

$$\pi(\mathcal{P}) < \alpha$$

$$\mathbb{P}\left(\tilde{t}(\mathcal{P}_i) < T - \underline{t}(i)\right) < \alpha$$

α-Reliable shortest path problem

Comparison against expected value



What we are aiming to solve with the stochastic pulse algorithm (SPA) in its two variants:

- Modeling travel times with Phase-Type distributions. (SPA-PH)
- Modeling travel times with the best fit distribution and estimating the probabilities with Monte Carlo simulation. (SPA-MC)

We can solve efficiently with the Pulse Algorithm (PA)

α-Reliable shortest path problem

Preliminary results

Universidad de los Andes Colombia

Reliable
Not reliable
Infeasible

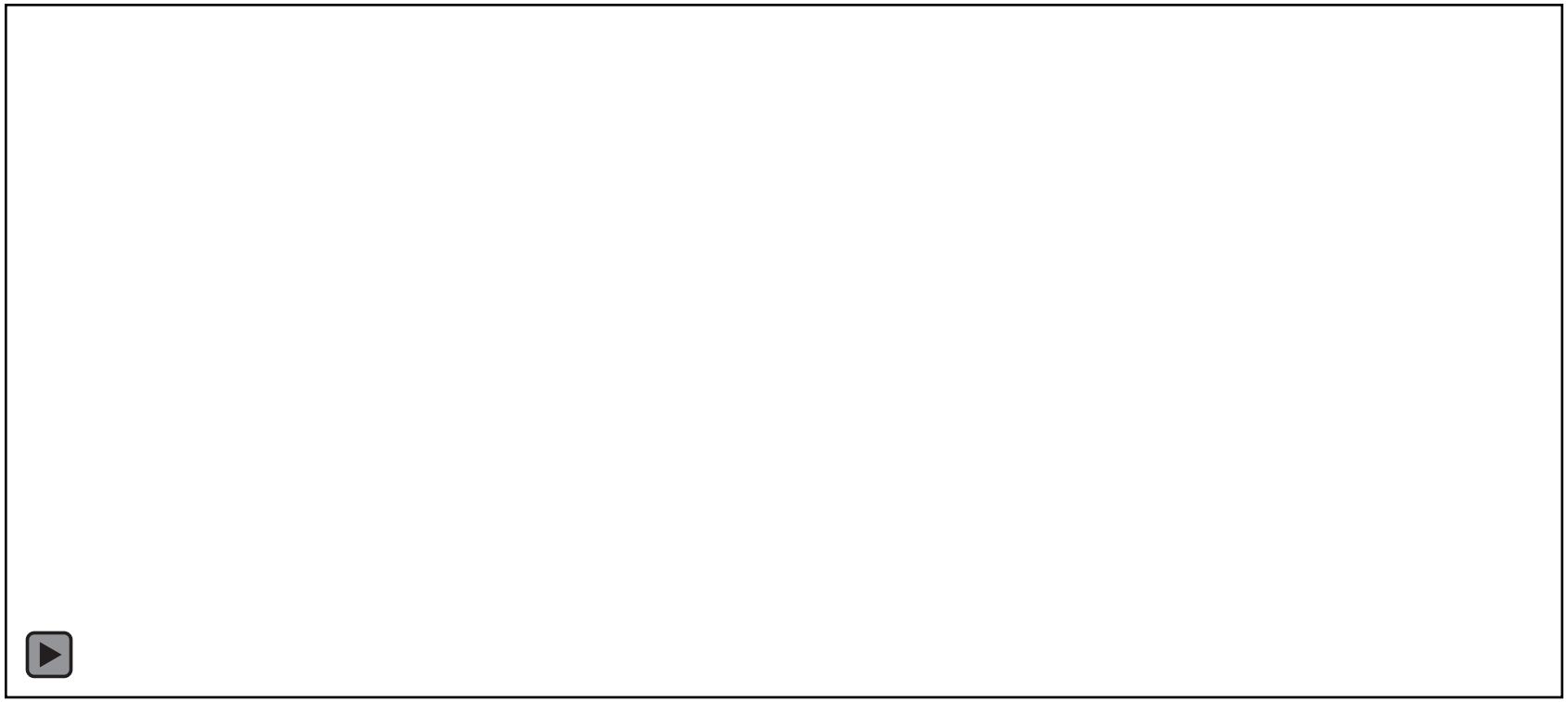
Instance	Probability of arriving on time						
	PA	SPA-MC		SPA-PH-3		SPA-PH-5	
	Real	Real	Fit	Real	Fit	Real	Fit
1	0.7785	0.9989	0.9995	0.9994	0.9977	0.9991	0.9996
2	0.6829	0.9999	1.0000	0.9997	0.9863	0.9999	0.9974
3	0.6892	1.0000	1.0000	1.0000	0.9963	1.0000	0.9996
4	0.6224	0.9984	0.9986	0.9989	0.9284	0.9837	0.9046
5	0.7307	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6	0.6517	0.9233	0.9103	1.0000	0.9744	1.0000	0.9933
7	0.7387	0.9956	0.9966	0.9976	0.9543	0.9971	0.9832
8	0.6935	1.0000	1.0000	1.0000	0.9761	1.0000	0.9936
9	0.7803	0.7659	0.9150	1.0000	0.9862	1.0000	0.9969
10	0.6297	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
11	0.7410	1.0000	0.9999	0.9999	0.9998	0.9999	1.0000
12	0.6731	0.9825	0.9893	0.9828	0.9806	0.9842	0.9924
13	0.7632	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
14	0.7325	0.9697	0.9775	-	-	0.9698	0.9008
15	0.7688	1.0000	1.0000	1.0000	0.9925	1.0000	0.9990
16	0.7754	0.9169	0.9167	0.9983	0.9907	0.9975	0.9977
17	0.5827	0.9956	0.9956	0.9958	0.9113	0.9952	0.9402
18	0.8017	0.9998	0.9999	0.9998	0.9971	1.0000	0.9995
19	0.7400	0.9974	0.9986	0.9976	0.9987	0.9983	0.9989
20	0.7157	0.9955	0.9959	0.9961	0.9826	0.9955	0.9936
Total	0/20	19/20		19/20		20/20	

Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
- Part V: perspectives
  - Bidirectional pulse algorithm
  - α-Reliable shortest path
  - **Least expected travel time path problem**

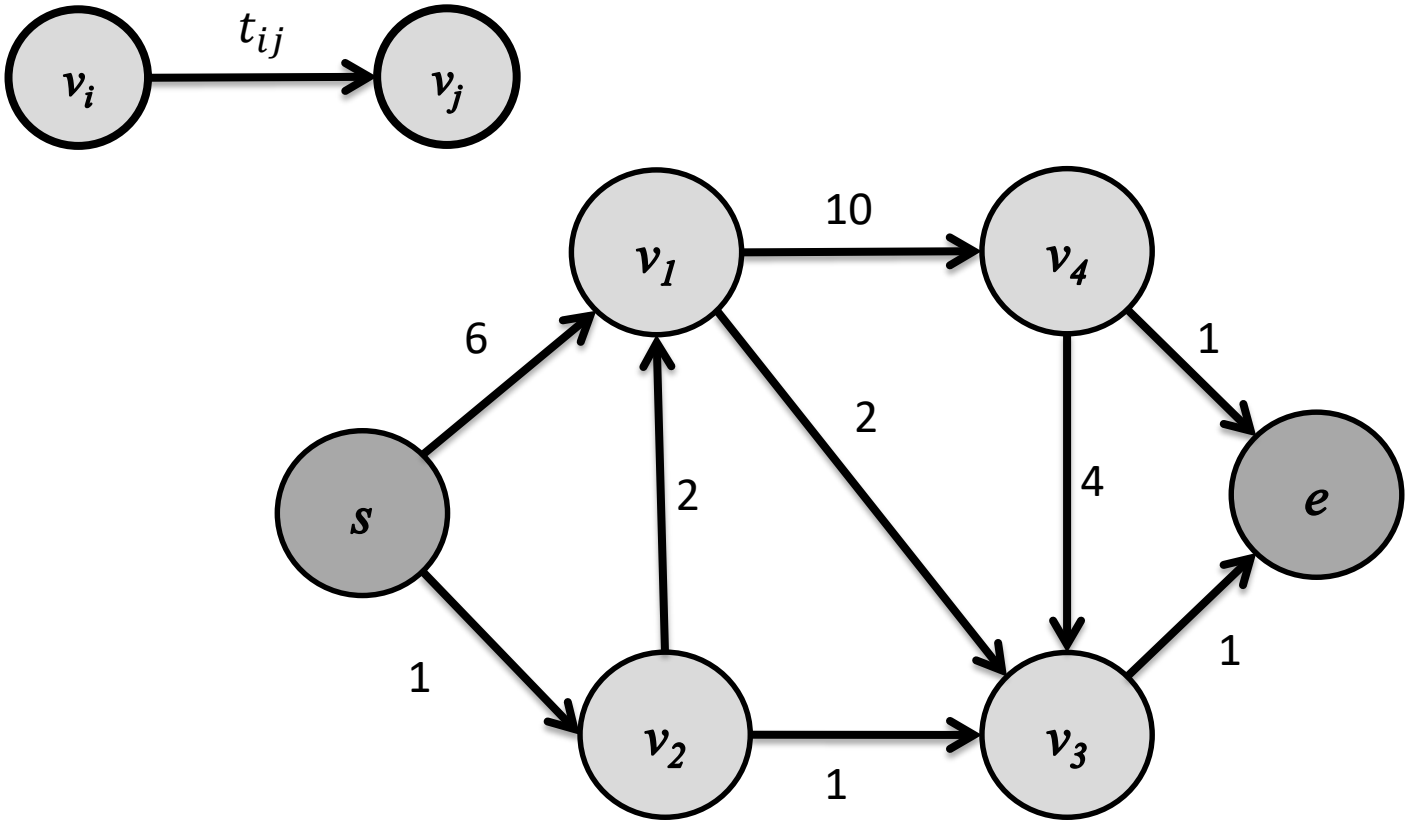
Least Expected Travel Time Path Problem (LETP)

Motivation



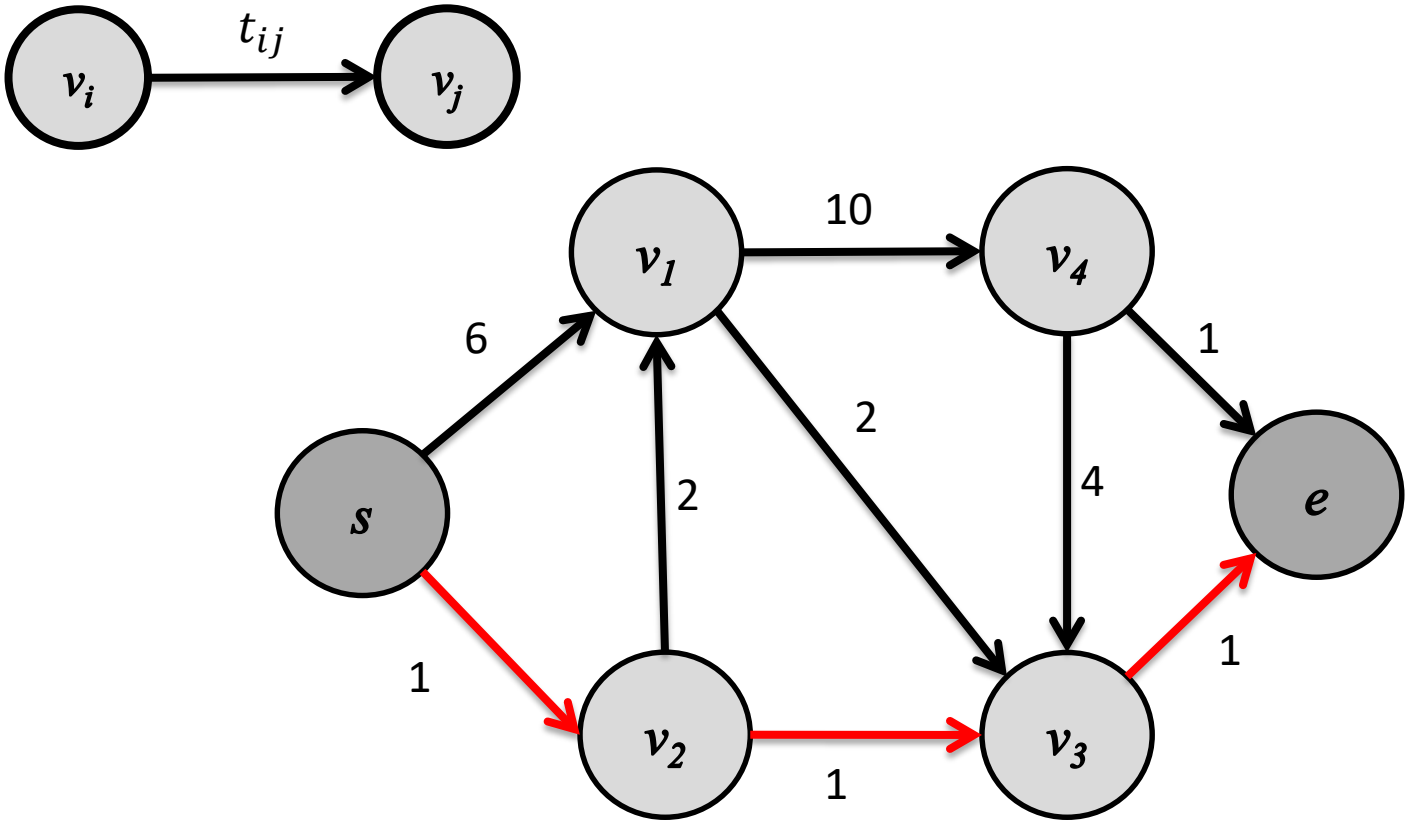
Least Expected Travel Time Path Problem (LETP)

Motivation



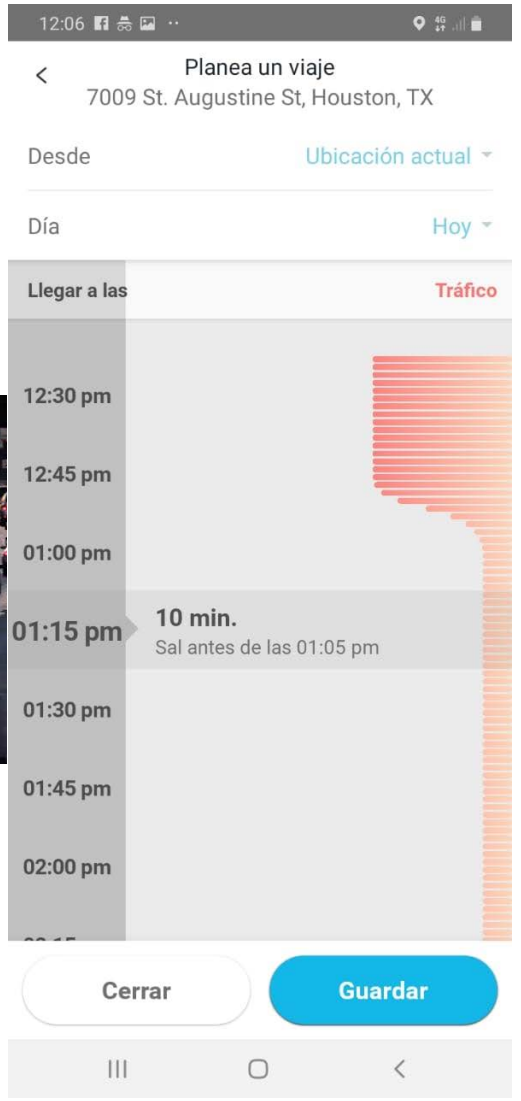
Least Expected Travel Time Path Problem (LETP)

Motivation



Least Expected Travel Time Path Problem (LETP)

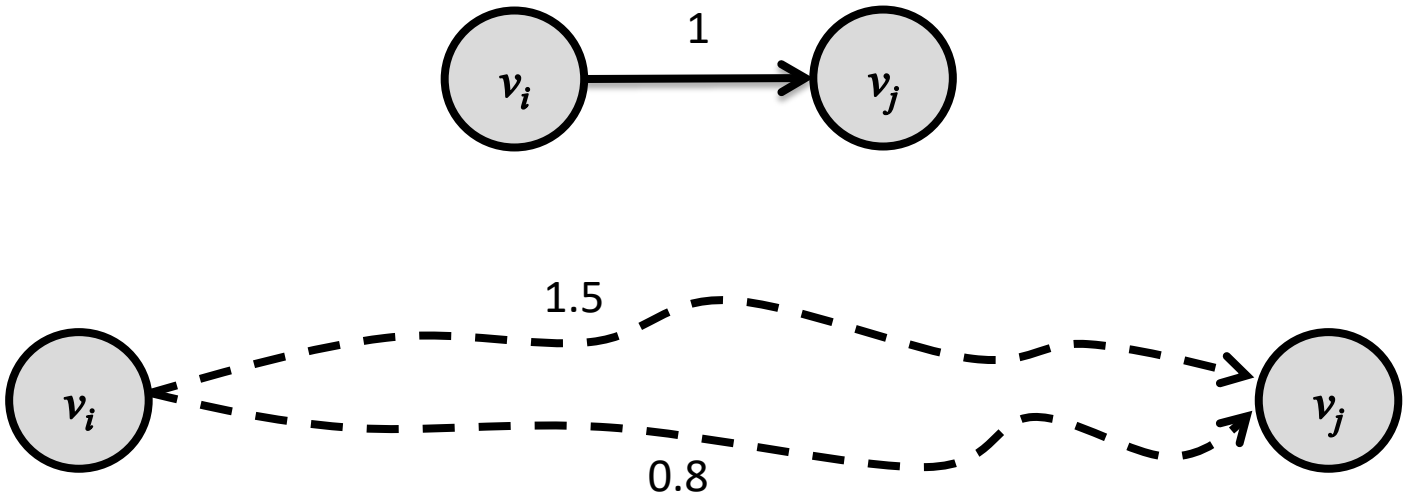
Motivation



Least Expected Travel Time Path Problem (LETP)

Motivation

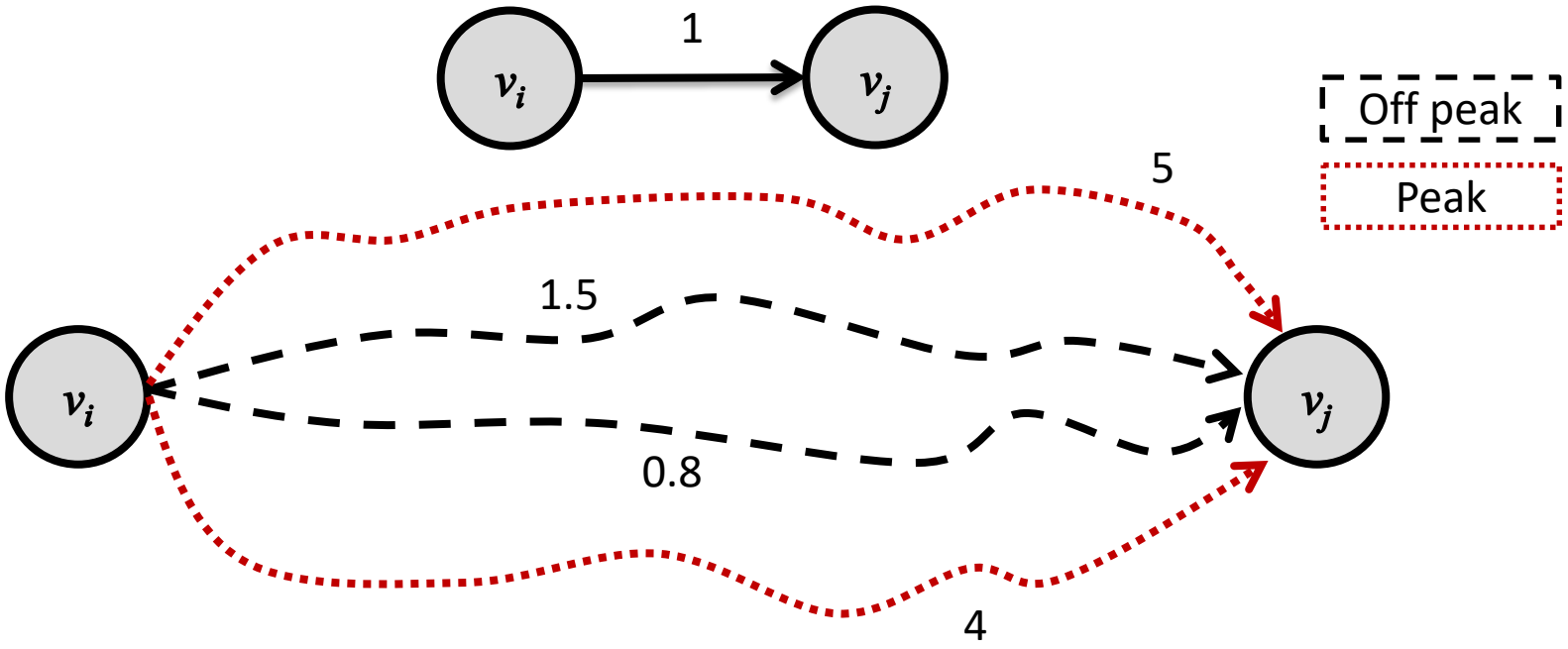
- ***Uncertainty*** captures random fluctuations of the links travel time whereas ***dynamics*** considers its variation in relation to the time of day.



Least Expected Travel Time Path Problem (LETP)

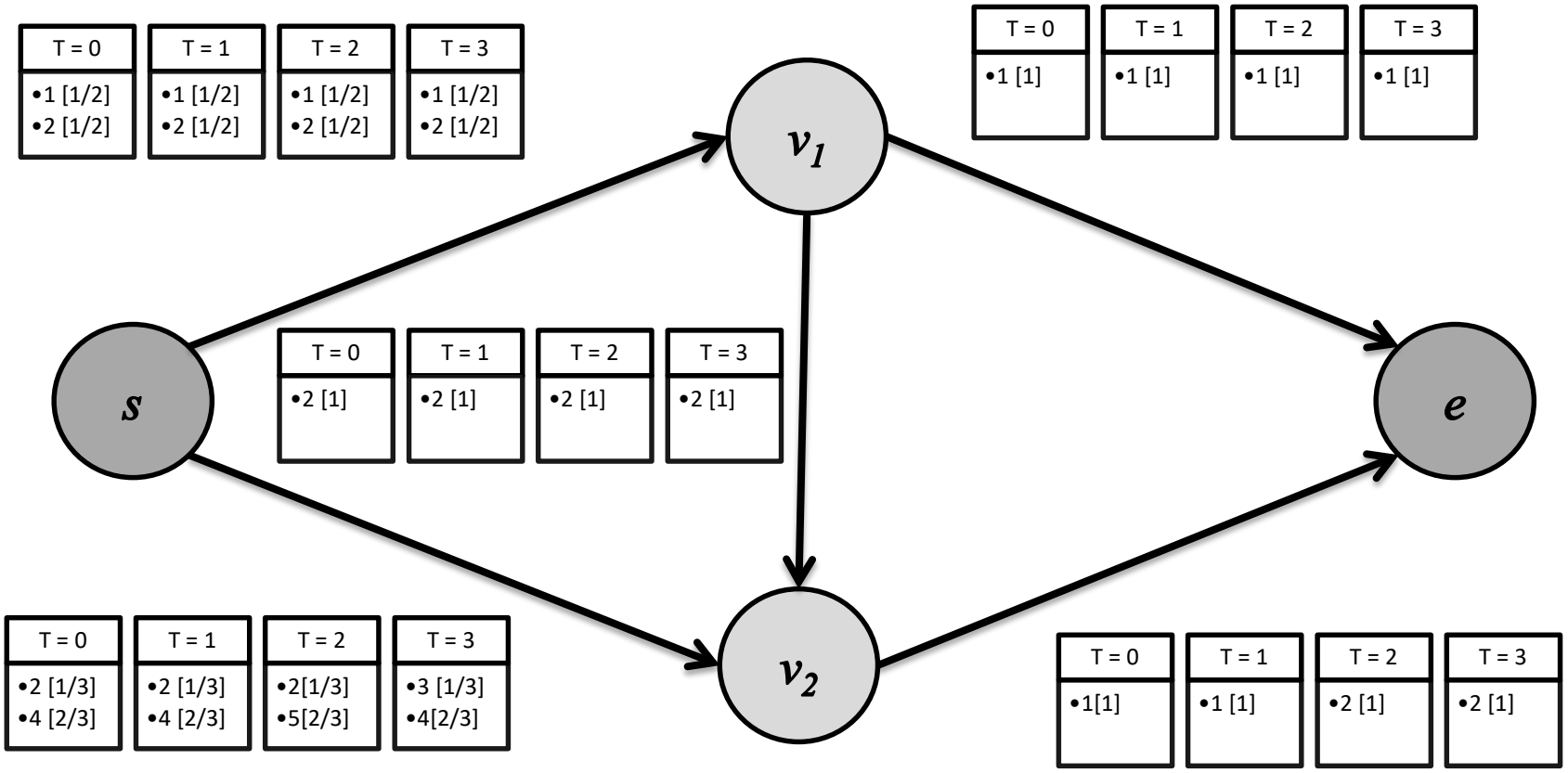
Motivation

- ***Uncertainty*** captures random fluctuations of the links travel time whereas ***dynamics*** considers its variation in relation to the time of day.



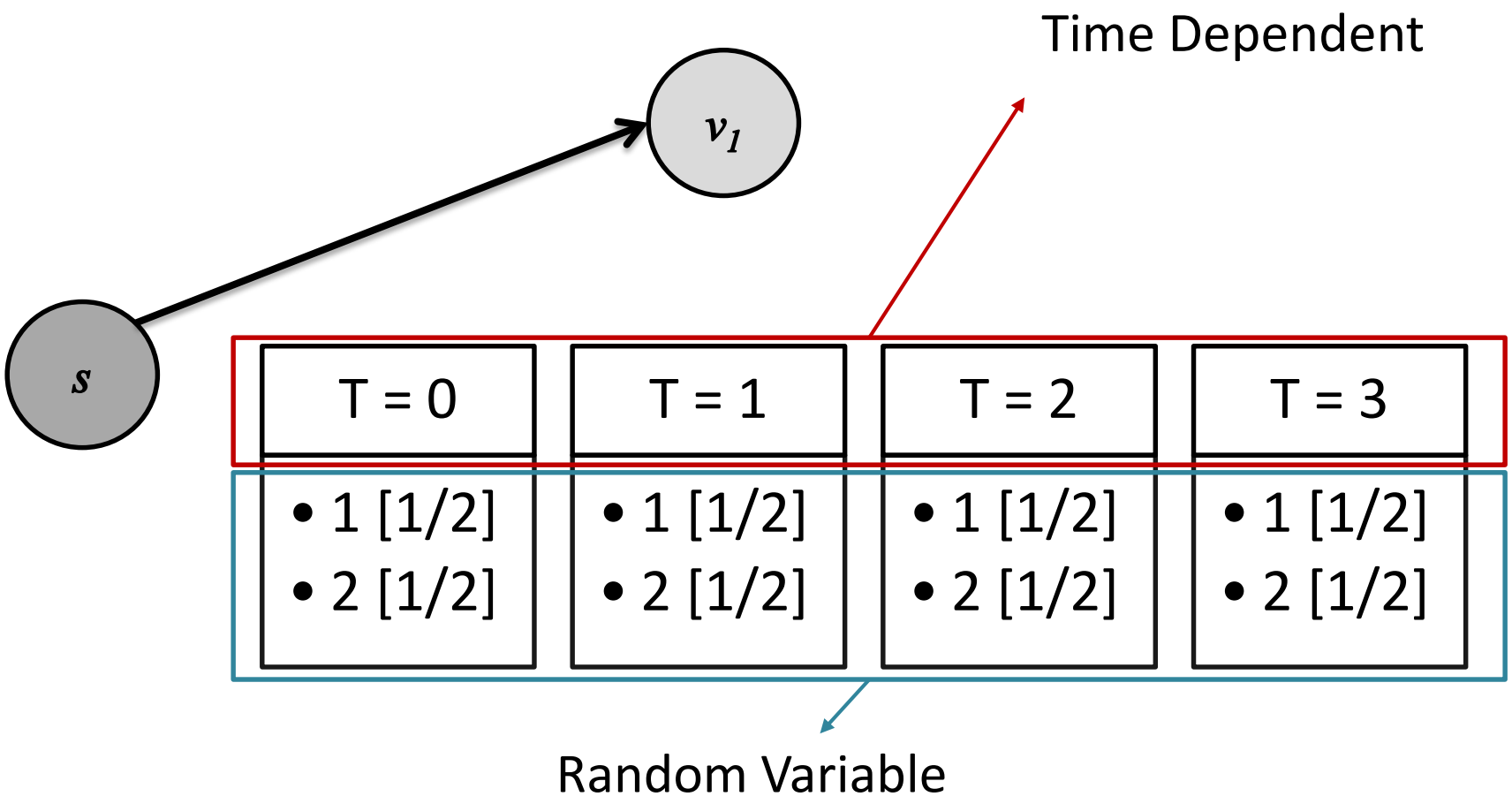
Least Expected Travel Time Path Problem (LETP)

Motivation



Least Expected Travel Time Path Problem (LETP)

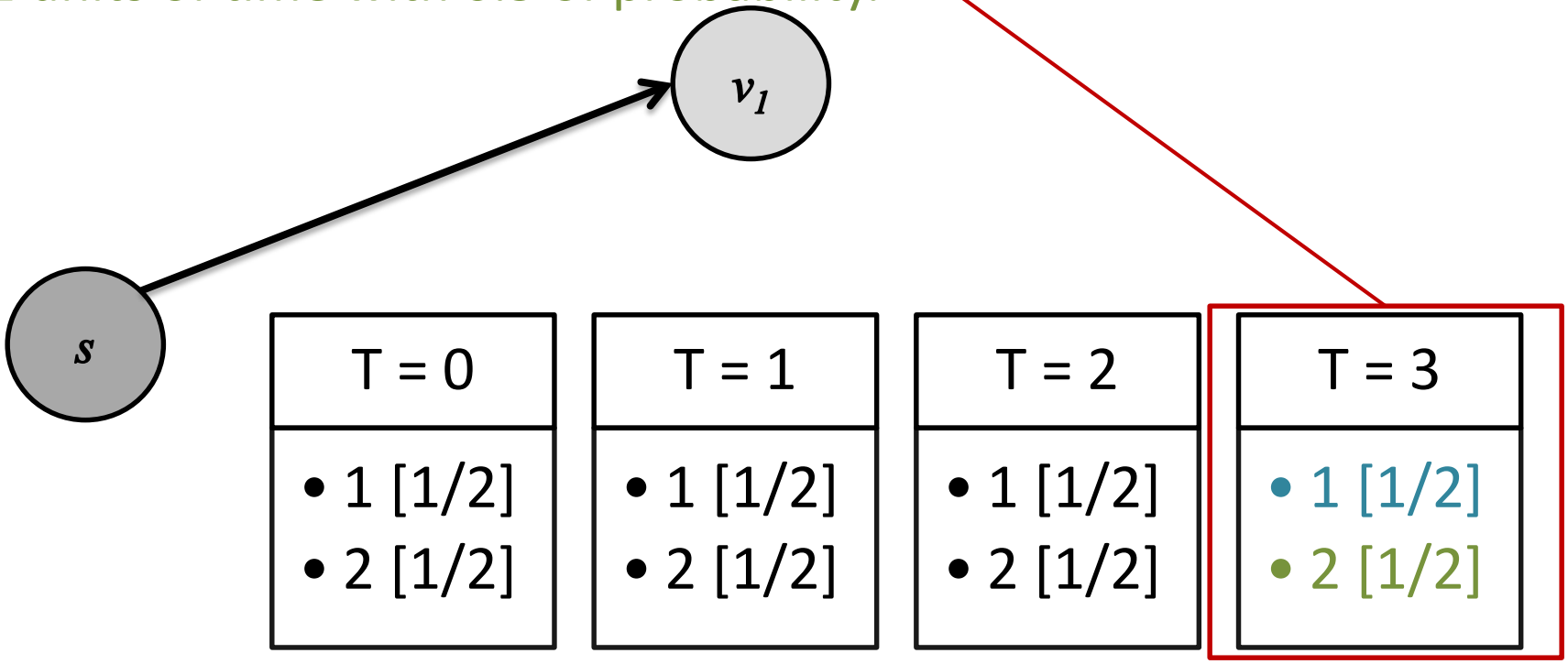
Motivation



Least Expected Travel Time Path Problem (LETP)

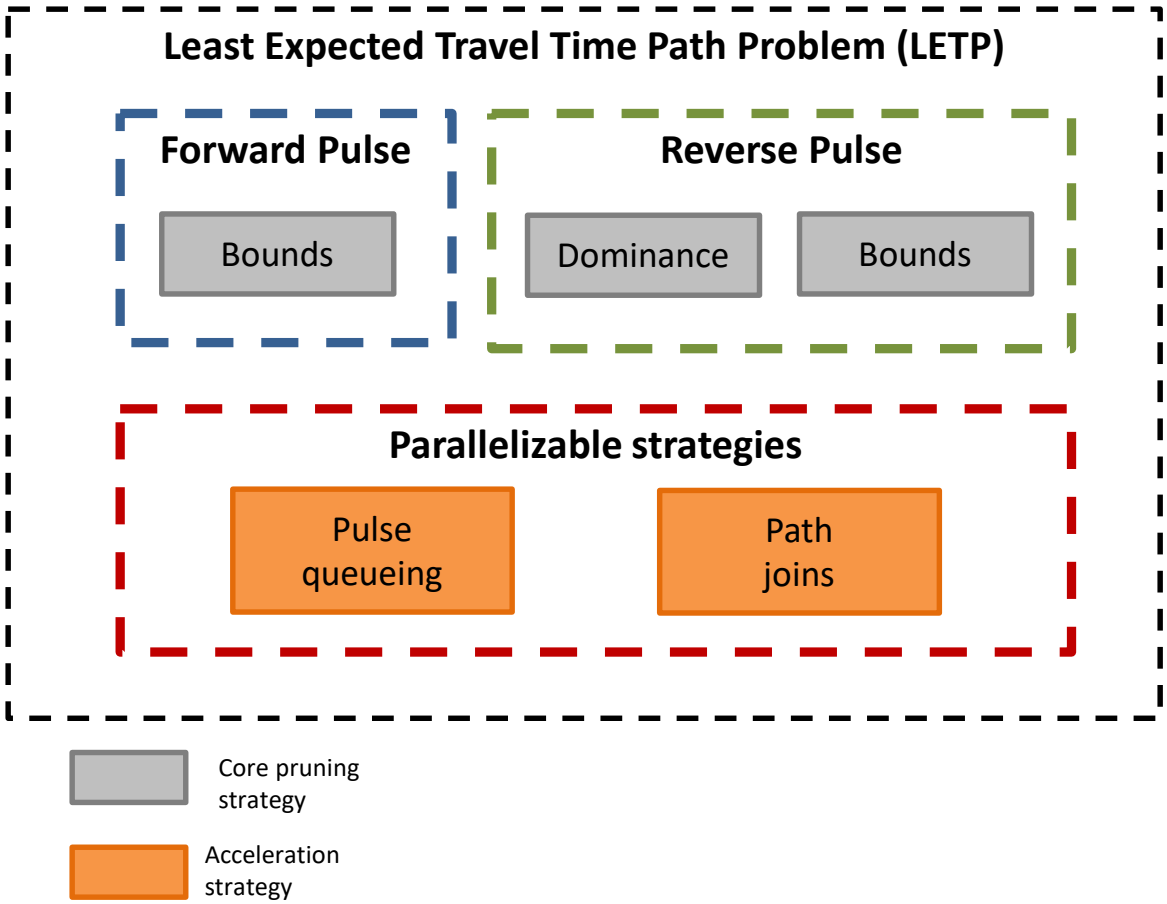
Motivation

When departing from node s to node v_1 at time $\tau = 3$, the link travel time is 1 unit of time with 0.5 of probability, and 2 units of time with 0.5 of probability.



Pulse Algorithm for the LETP on STD networks

Overview

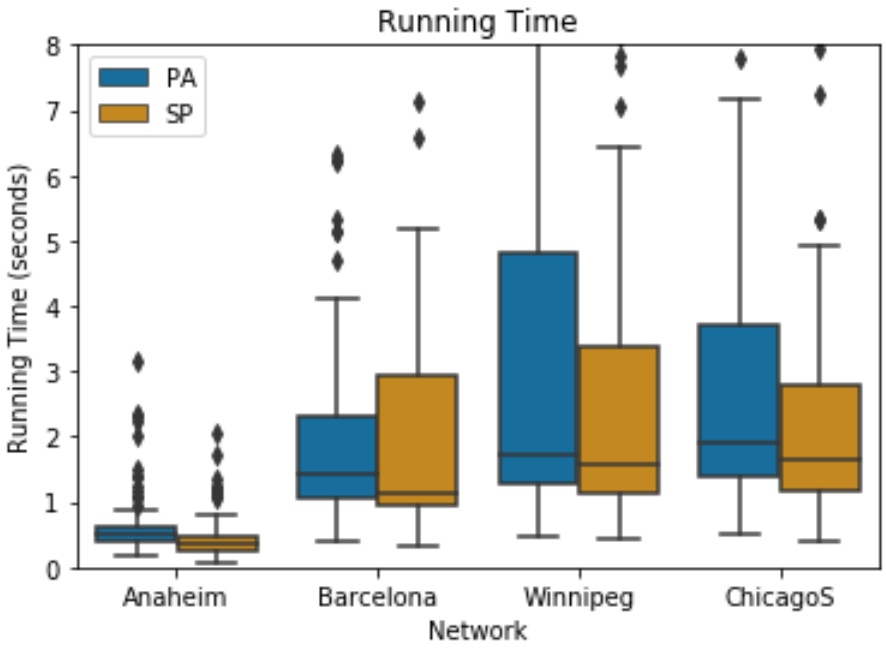


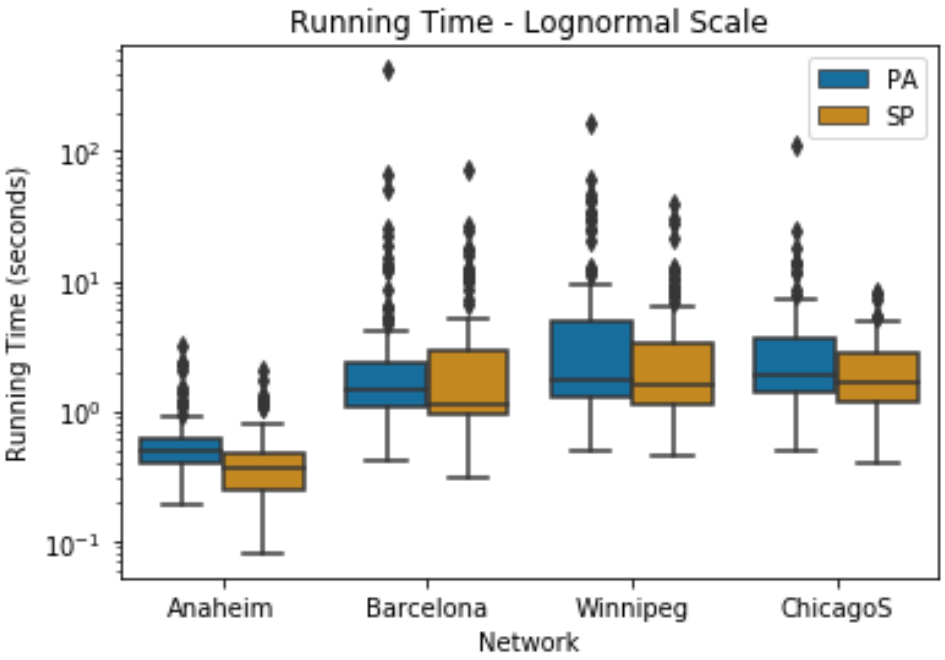
Computational Experiments

Preliminary Results

- 4 medium-sized networks.
- Anaheim, Barcelona, Winnipeg and ChicagoS.
- 100 replications → 100 randomly generated OD.
- $Q = 4, \quad \delta = 4$
- Stochastic Pulse (SP)
- Pruning Algorithm (PA) (Prakash, 2018)

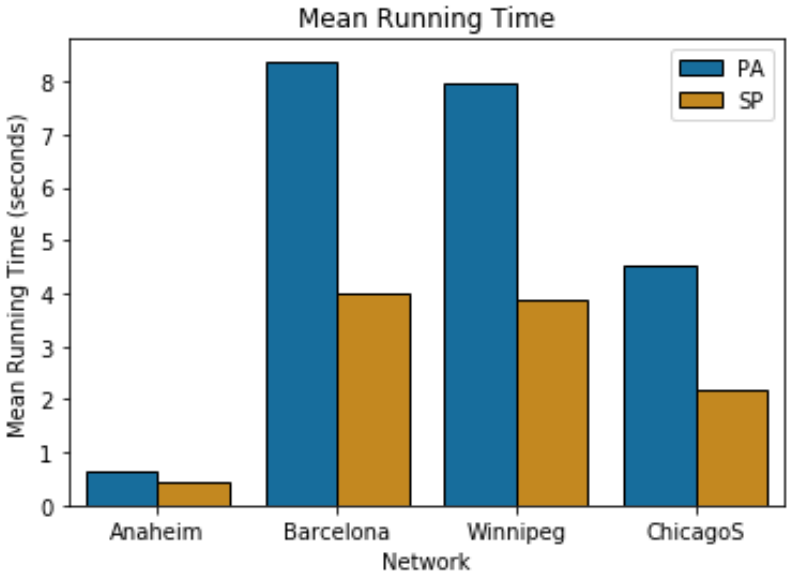
Computational Experiments
Benchmark

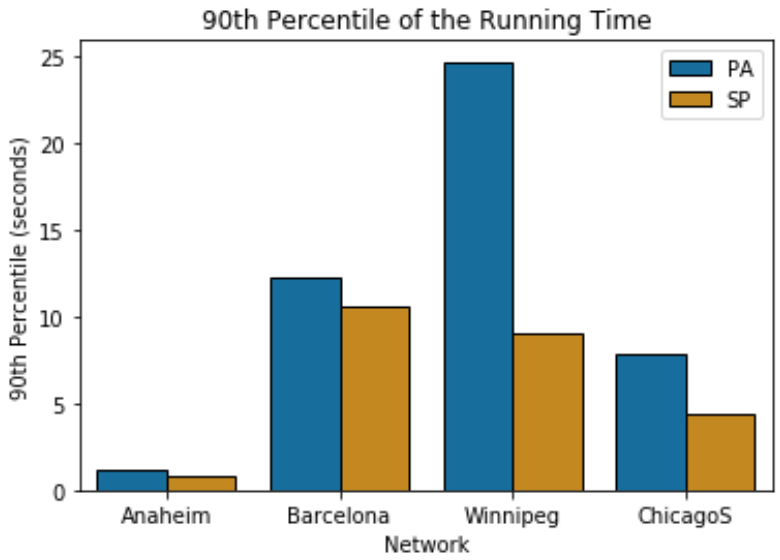


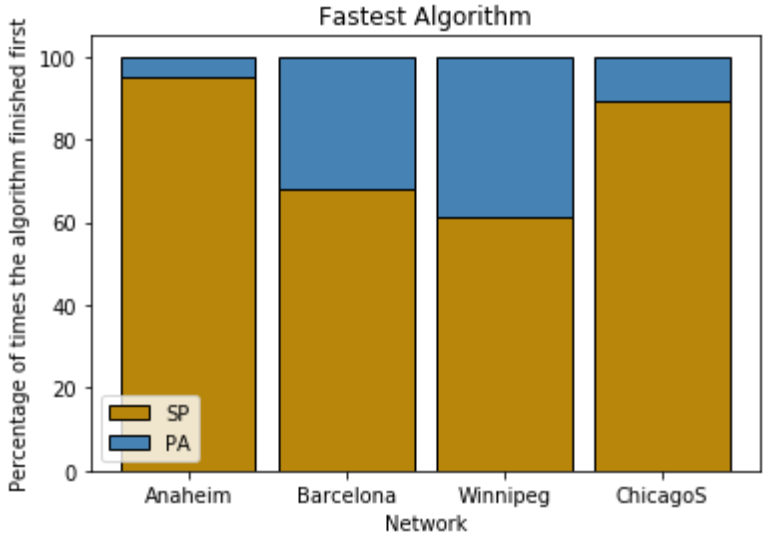


Computational Experiments

Benchmark

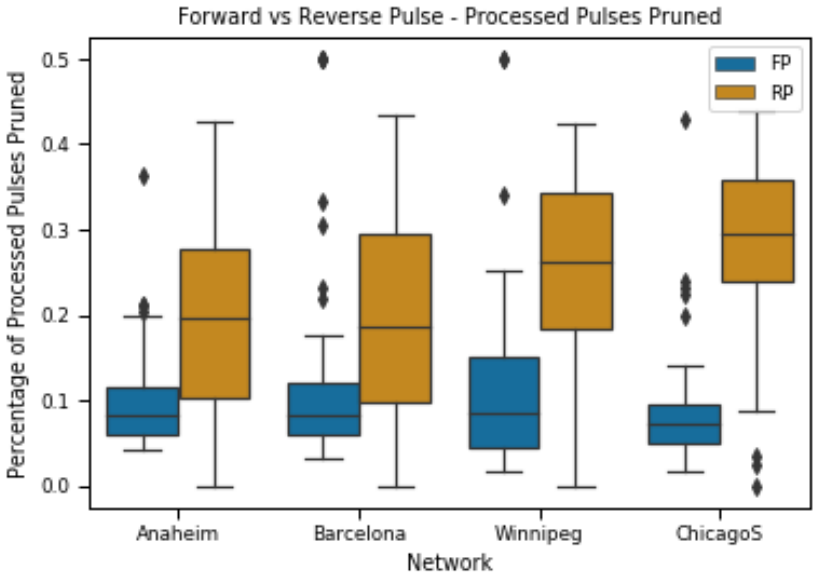


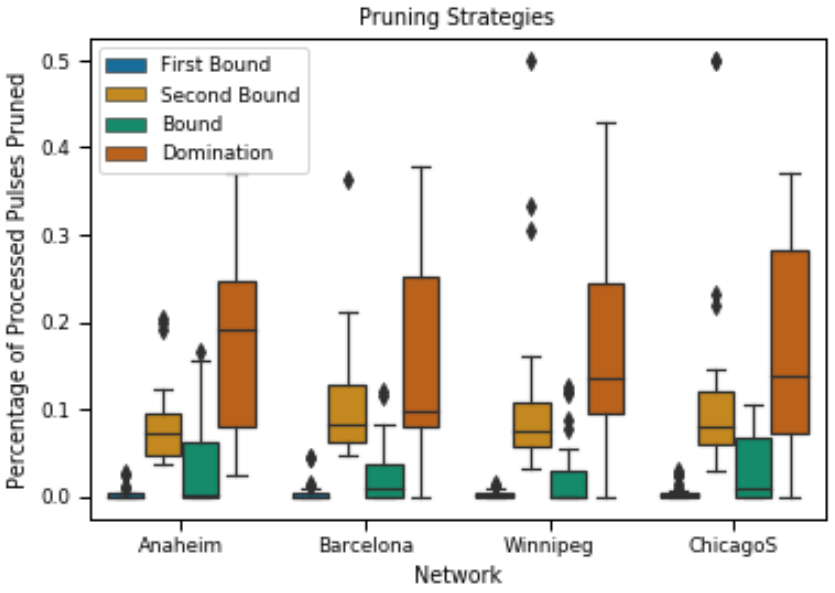


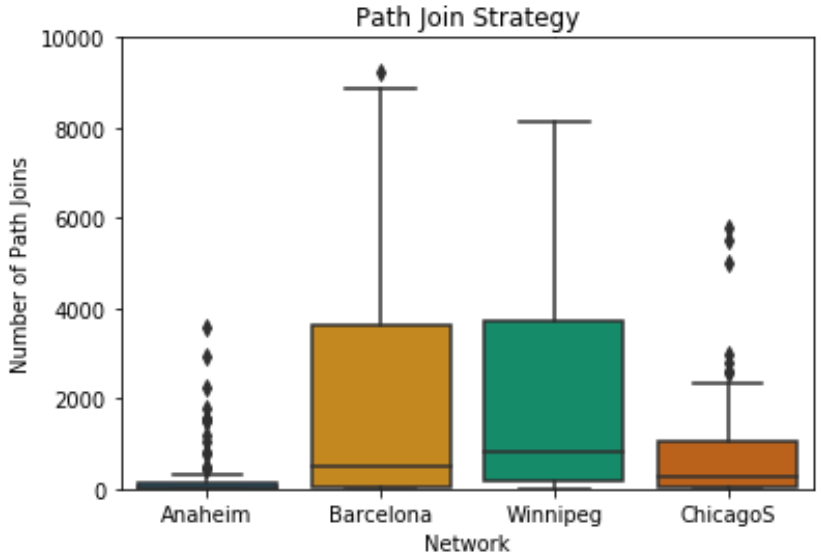


Computational Experiments

Stochastic Pulse







Agenda

- Part I: fundamentals
- Part II: intuition
- Part III: extensions
- Part IV: applications
- Part V: perspectives

Conclusions

- The pulse algorithm is a modular framework for hard shortest path variants.
- The algorithm has a very simple intuition inspired on a pulse propagation.
- Pruning strategies narrow the solution space to avoid complete enumeration.
- The pruning strategies use a collection of results from the network optimization literature.
- Pruning strategies can be designed and used as modules to tackle different problems (problem-specific strategies).
- The pulse framework provides competitive solution methods against state-of-the-art algorithms for hard shortest path variants.
- The pulse algorithm is easy to parallelize and shows promising results in multi-threaded versions

Referencias (pulso)

- Bolívar, M. A., Lozano, L. & Medaglia, A. L. (2014). Acceleration Strategies for the Weight Constrained Shortest Path Problem with Replenishment. Optimization Letters. 8(8): 2155-2172. DOI:10.1007/s11590-014-0742-x **[Pulse extension]**
- Cabrera, N., Medaglia, A. L., Lozano, L., Duque, D. (2019). An exact bidirectional pulse algorithm for the constrained shortest path. COPA Technical Report. **[Pulse core paper]**
- Duque, D., Lozano, L. & Medaglia, A. L. (2015a). An exact method for the biobjective shortest path problem for large-scale road networks. European Journal of Operational Research. 242:788-797. DOI:10.1016/j.ejor.2014.11.003 **[Pulse extension]**
- Duque, D., Lozano, L. & Medaglia, A. L. (2015b). Solving the Orienteering Problem with Time Windows via the Pulse Framework. Computers & Operations Research. 54:168-176. DOI: 10.1016/j.cor.2014.08.019 **[Pulse extension]**
- Lozano, L. and Medaglia, A. L. (2013). On an exact method for the constrained shortest path problem. Computers & Operations Research. 40 (1):378-384. DOI:10.1016/j.cor.2012.07.008 **[Pulse core paper]**
- Duque, D. & Medaglia, A. L. (2019). An exact method for a class of robust shortest path problems with scenarios. Networks. DOI: 10.1002/net.21909. Available at: https://doi.org/10.1002/net.21909 **[Pulse extension]**

Referencias (pulso)

- Lozano, L. and Smith, J.C. (2016). A backward sampling framework for interdiction problems with fortification. INFORMS Journal on Computing, 29(1):123-139. DOI:10.1287/ijoc.2016.0721 **[Pulse application]**
- Lozano, L., Duque, D. & Medaglia, A. L. (2016). An exact algorithm for the elementary shortest path problem with resource constraints. Transportation Science. 50(1):348–357. DOI:10.1287/trsc.2014.0582 **[Pulse extension]**
- Medaglia, A. L., Lozano, L., & Duque, D. (2018). Solving hard shortest path problems with the pulse framework. In IFORS News. Tutorial Section. 12(2). Available at: http://ifors.org/newsletter/ifors-news-june-2018.pdf , ISSN: 2223-4373 **[Pulse core paper]**
- Montoya, A., Guéret, C., Mendoza, J. E., & Villegas, J. G. (2016). A multi-space sampling heuristic for the green vehicle routing problem. Transportation Research Part C: Emerging Technologies, 70, 113-128. DOI:10.1016/j.trc.2015.09.009 **[Pulse application]**
- Restrepo, M. I., Lozano, L., and Medaglia, A. L. (2012). Constrained network-based column generation for the multi-activity shift scheduling problem. International Journal of Production Economics. 140(1):466-472. DOI: 10.1016/j.ijpe.2012.06.030 **[Pulse application]**

Acknowledgments

Q&A

¡Ta!
….gracias.

Andrés L. Medaglia
amedagli@uniandes.edu.co
http://copa.uniandes.edu.co