The 17th International Symposium on Combinatorial Search
SoCS 2024

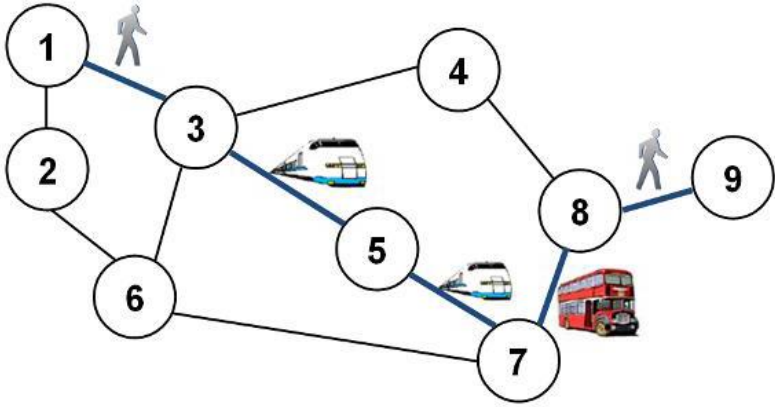
Efficient and Exact Public Transport Routing via a Transfer Connection Database

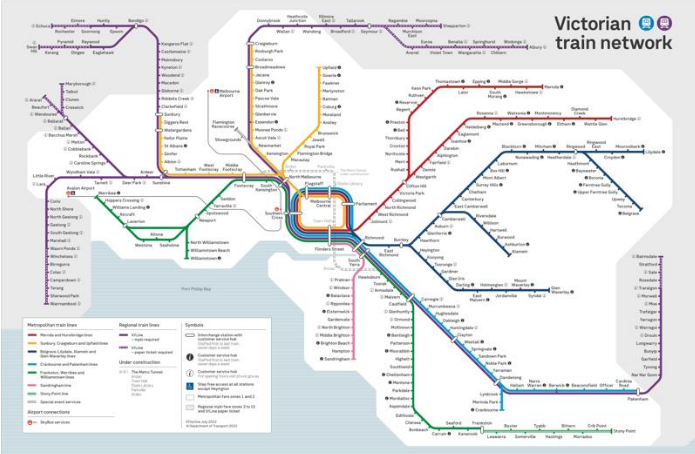
Abdallah Abuaisha, Mark Wallace, Daniel Harabor, Bojie Shen

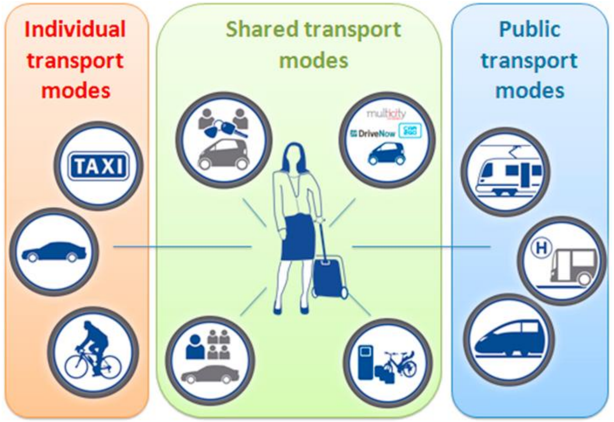
Department of Data Science and AI, Monash University, Australia

June 2024

Introduction

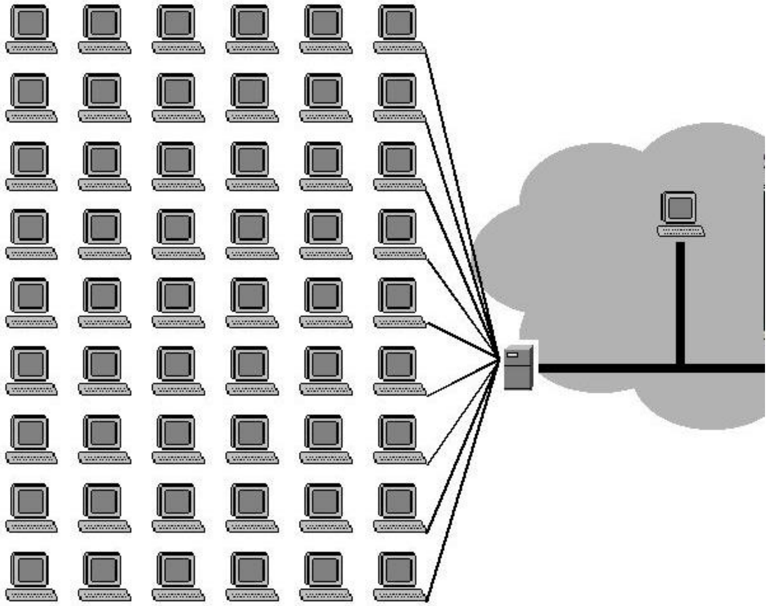






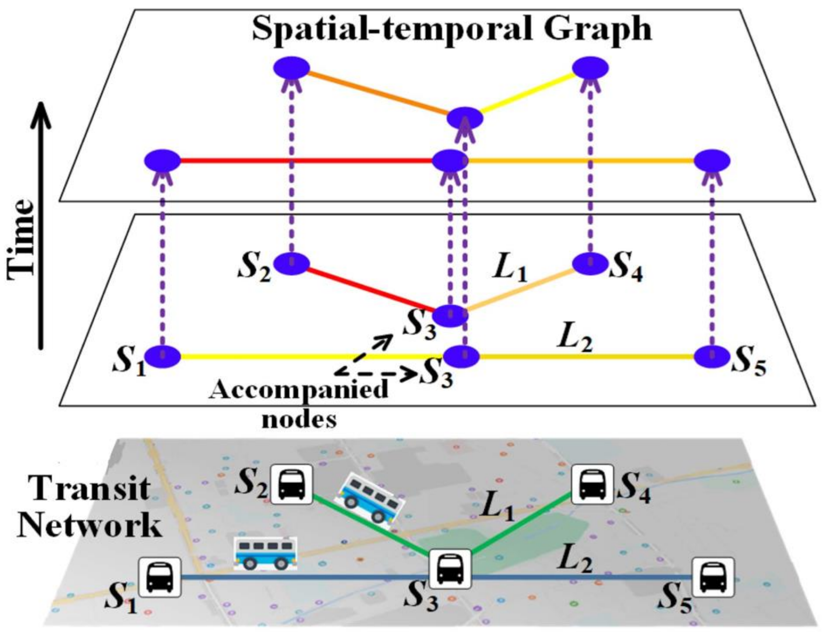
Motivation

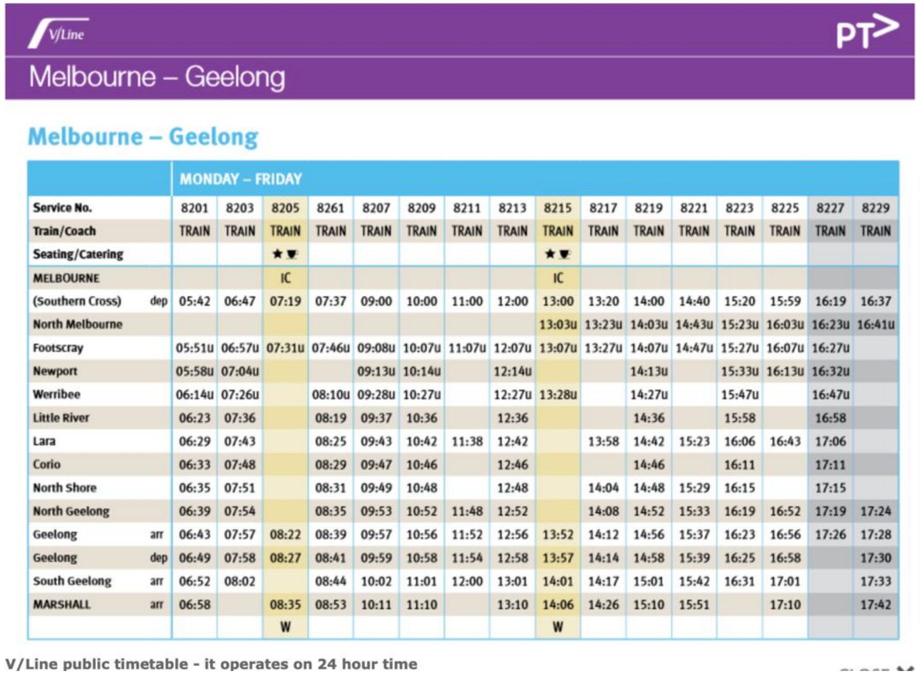
1. Query Efficiency
   - High number of queries to be answered by a central server
   - Queries have to be handled efficiently



Motivation

1. Query Efficiency
   - Unique structure of public transport networks
   - Existing works are not efficient enough





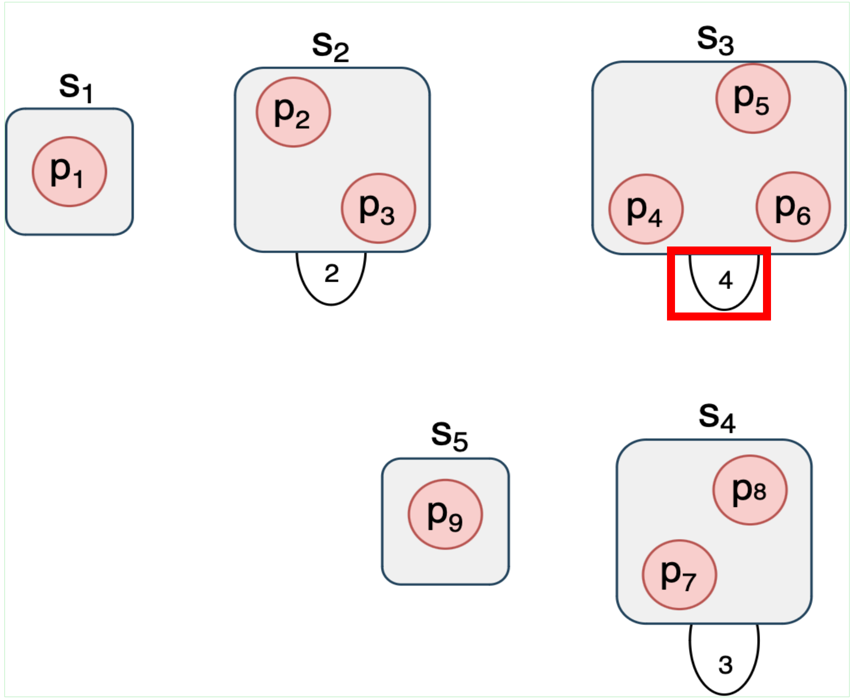
Melbourne – Geelong

		MONDAY – FRIDAY															
Service No.		8201	8203	8205	8261	8207	8209	8211	8213	8215	8217	8219	8221	8223	8225	8227	8229
Train/Coach		TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN
Seating/Catering				★						★							
MELBOURNE				IC						IC							
(Southern Cross)	dep	05:42	06:47	07:19	07:37	09:00	10:00	11:00	12:00	13:00	13:20	14:00	14:40	15:20	15:59	16:19	16:37
North Melbourne										13:03u	13:23u	14:03u	14:43u	15:23u	16:03u	16:23u	16:41u
Footscray		05:51u	06:57u	07:31u	07:46u	09:08u	10:07u	11:07u	12:07u	13:07u	13:27u	14:07u	14:47u	15:27u	16:07u	16:27u	
Newport		05:58u	07:04u			09:13u	10:14u		12:14u			14:13u		15:33u	16:13u	16:32u	
Werribee		06:14u	07:26u		08:10u	09:28u	10:27u		12:27u	13:28u		14:27u		15:47u		16:47u	
Little River		06:23	07:36		08:19	09:37	10:36		12:36			14:36		15:58		16:58	
Lara		06:29	07:43		08:25	09:43	10:42	11:38	12:42		13:58	14:42	15:23	16:06	16:43	17:06	
Corio		06:33	07:48		08:29	09:47	10:46		12:46			14:46		16:11		17:11	
North Shore		06:35	07:51		08:31	09:49	10:48		12:48		14:04	14:48	15:29	16:15		17:15	
North Geelong		06:39	07:54		08:35	09:53	10:52	11:48	12:52		14:08	14:52	15:33	16:19	16:52	17:19	17:24
Geelong	arr	06:43	07:57	08:22	08:39	09:57	10:56	11:52	12:56	13:52	14:12	14:56	15:37	16:23	16:56	17:26	17:28
Geelong	dep	06:49	07:58	08:27	08:41	09:59	10:58	11:54	12:58	13:57	14:14	14:58	15:39	16:25	16:58		17:30
South Geelong	arr	06:52	08:02		08:44	10:02	11:01	12:00	13:01	14:01	14:17	15:01	15:42	16:31	17:01		17:33
MARSHALL	arr	06:58		08:35	08:53	10:11	11:10		13:10	14:06	14:26	15:10	15:51		17:10		17:42
				W						W							

V/Line public timetable - it operates on 24 hour time

Motivation

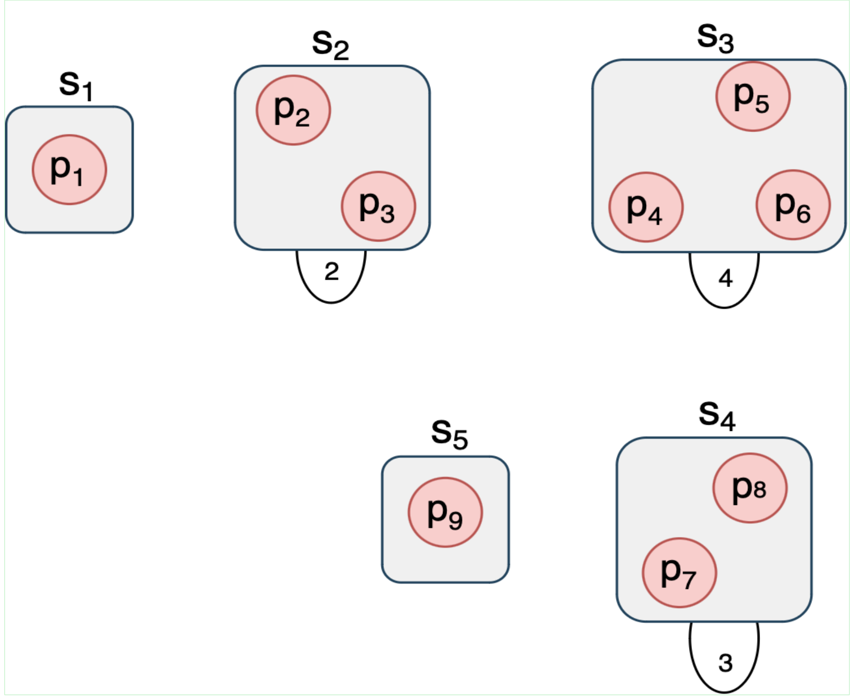
2. Transfer Modelling
   - Uniform transfer costs



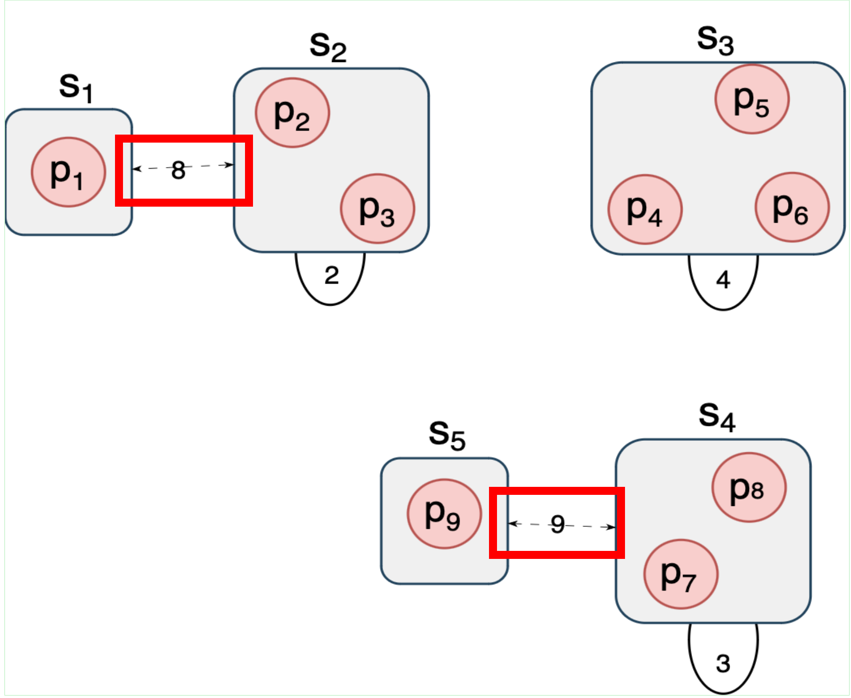
Intra-station transfer model

Motivation

2. Transfer Modelling
   - Uniform transfer costs



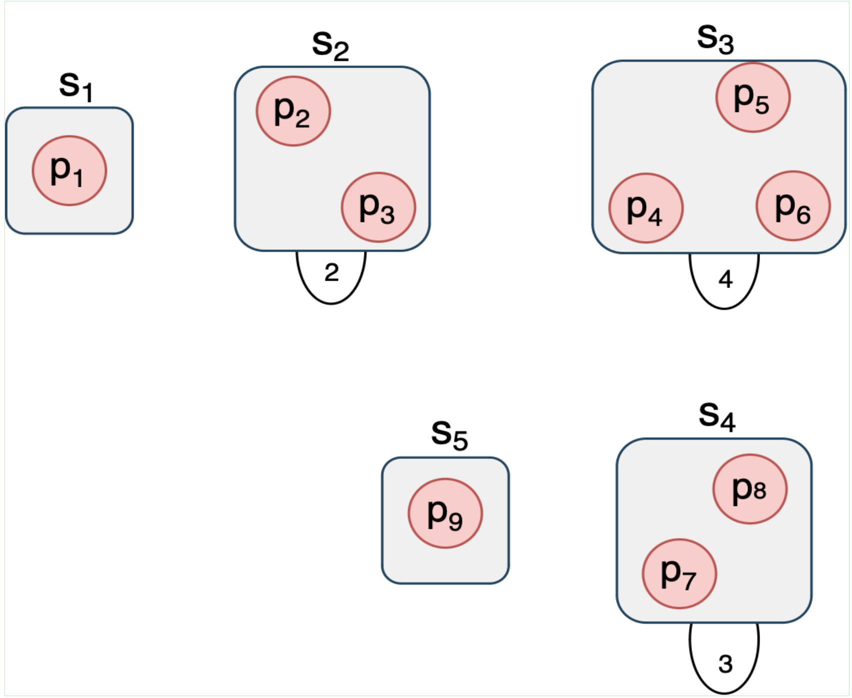
Intra-station transfer model



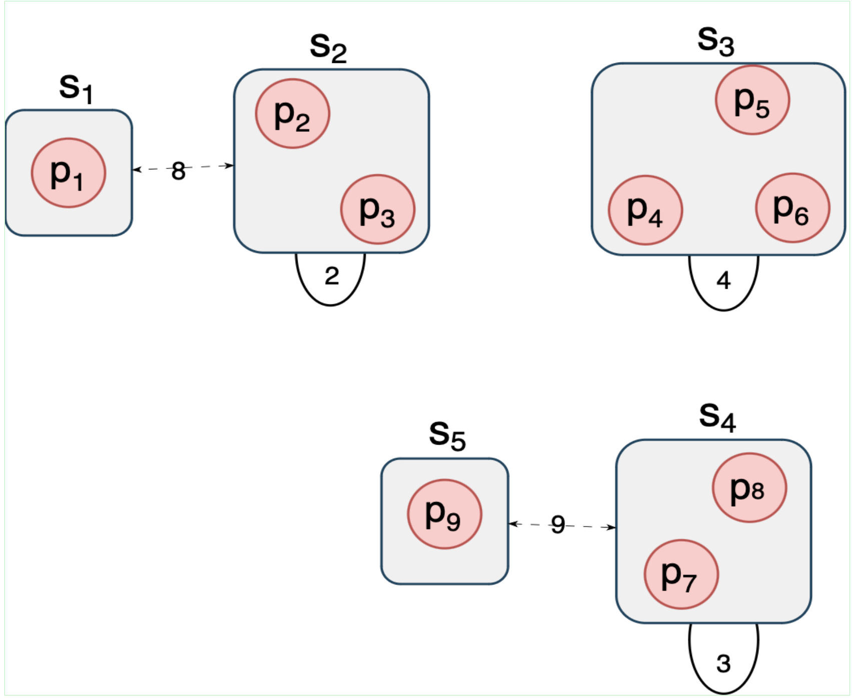
Inter-station transfer model

Motivation

2. Transfer Modelling
    - Uniform transfer costs → Infeasible or suboptimal journeys



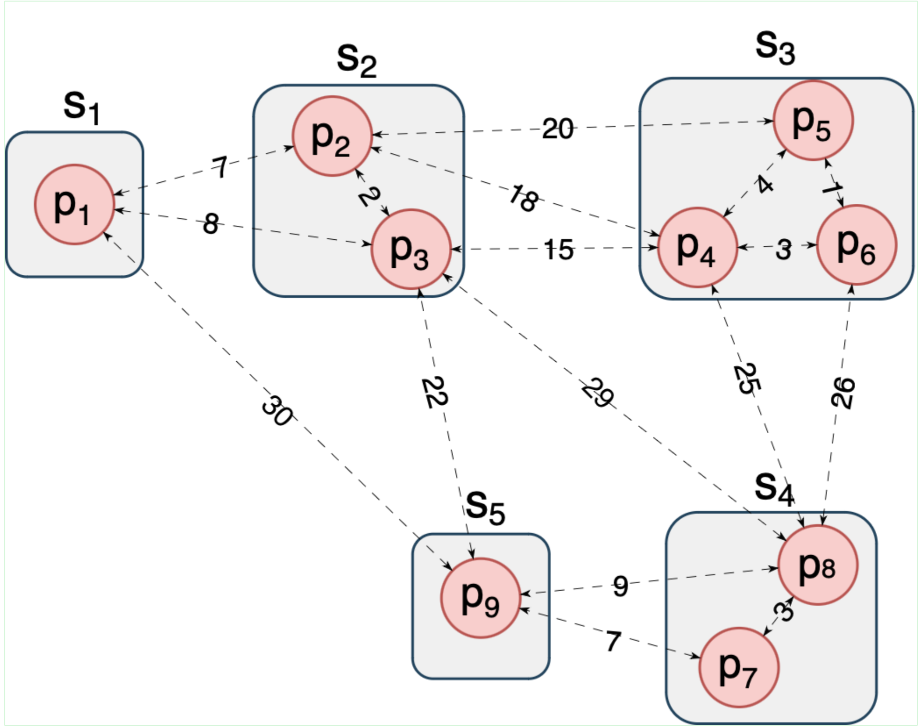
Intra-station transfer model



Inter-station transfer model

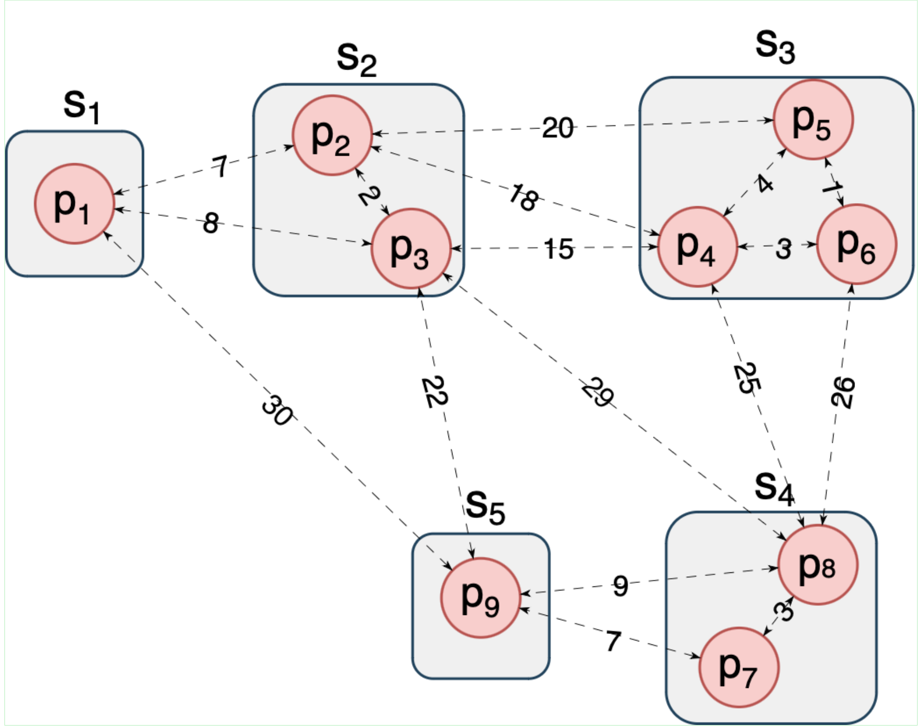
Motivation

2. Transfer Modelling
   - Comprehensive walking graph



Motivation

2. Transfer Modelling
   - Comprehensive walking graph → Costly preprocessing and slow queries



Contributions

1. Introducing a novel algorithm that solves public transport routing problem **efficiently** and **accurately.**
2. Demonstrating the importance of modelling transfers using **exact** transfer costs.
3. Proposing an efficient method for building a **compressed path database** in public transport networks.

Transfer Connection Database (TCD)

Network Modelling

Offline Preprocessing Phase

Online Query Phase

Transfer Connection Database (TCD)

Network Modelling

Offline Preprocessing Phase

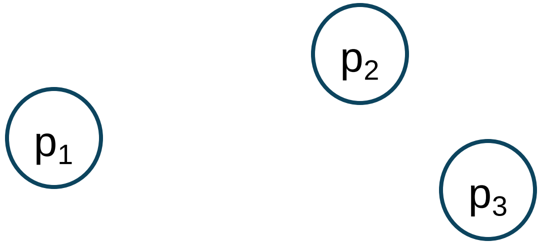
Online Query Phase

Network Modelling

1. Timetable Modelling
   - Timetable-based approach.

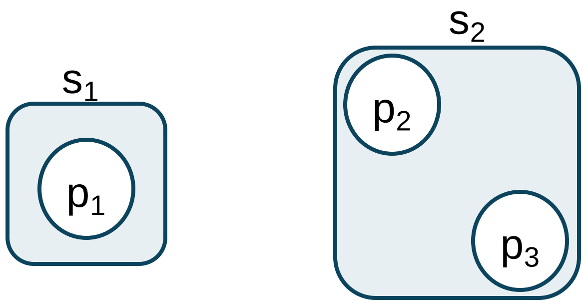
Network Modelling

1. Timetable Modelling
   - Timetable-based approach.
   - Timetable: Stops P,



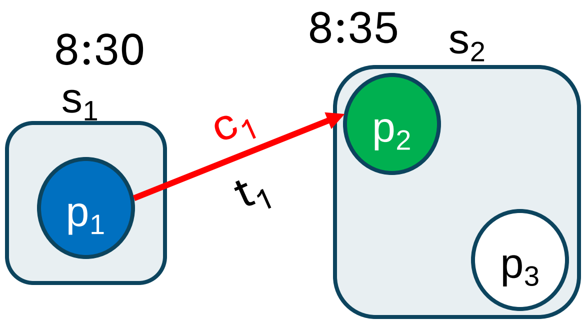
Network Modelling

1. Timetable Modelling
   - Timetable-based approach.
   - Timetable: Stops P, Stations S,



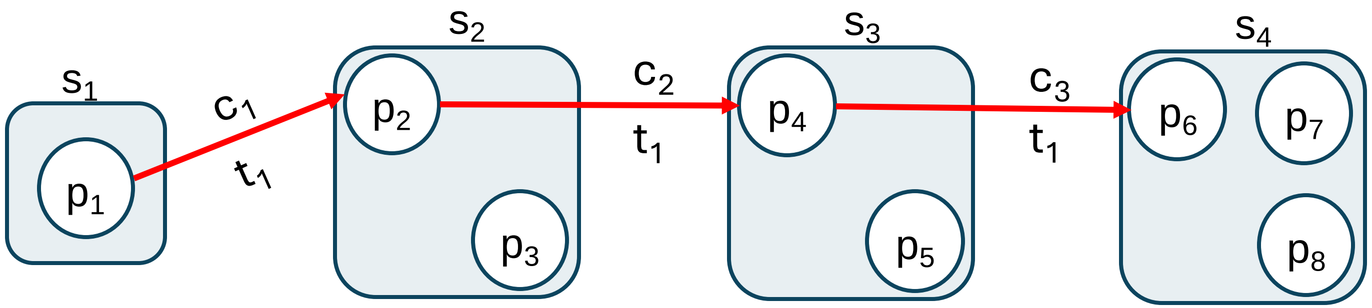
Network Modelling

1. Timetable Modelling
   - Timetable-based approach.
   - Timetable: Stops P, Stations S, Connections C,
   - $c = (p_{dep}, \tau_{dep}, p_{arr}, \tau_{arr}, t)$
   - $c_1 = (p_1, 8{:}30, p_2, 8{:}35, t_1)$



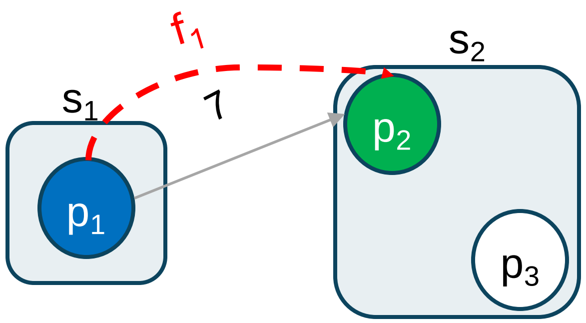
Network Modelling

1. Timetable Modelling
    - Timetable-based approach.
    - Timetable: Stops P, Stations S, Connections C, Trips T,
    - t_1 = <c_1, c_2, c_3>



Network Modelling

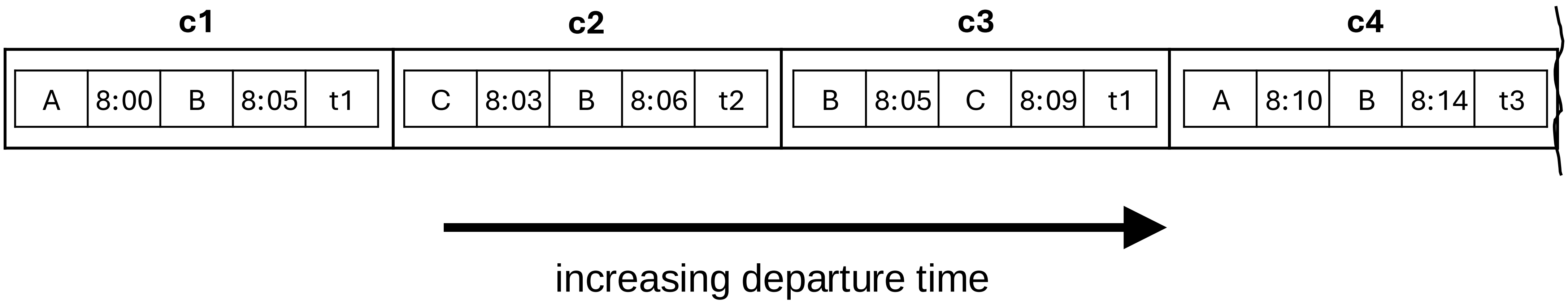
1. Timetable Modelling
   - Timetable-based approach.
   - Timetable: Stops P, Stations S, Connections C, Trips T, Footpaths F
   - $f = (p_{dep}, p_{arr}, \Delta\tau)$
   - $f_1 = (p_1, p_2, 7)$



Network Modelling

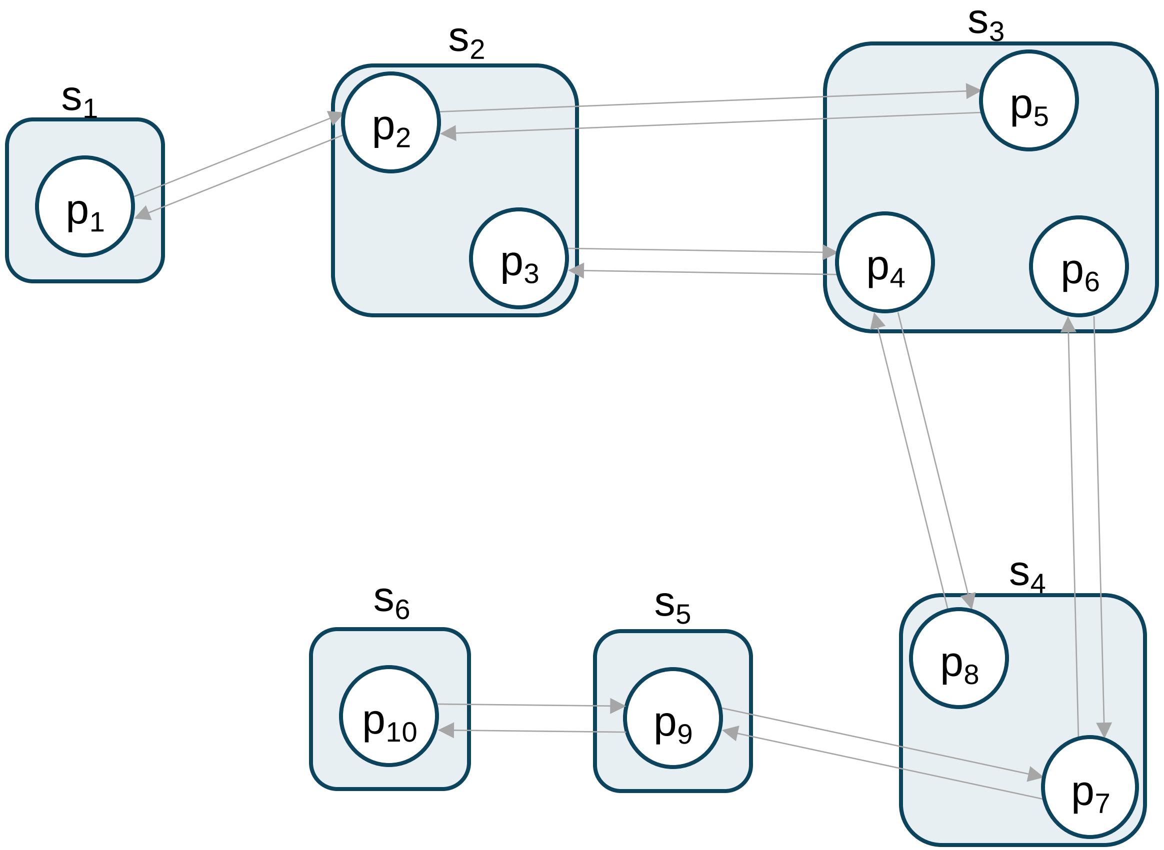
1. Timetable Modelling
   - Timetable-based approach.

Connections Array



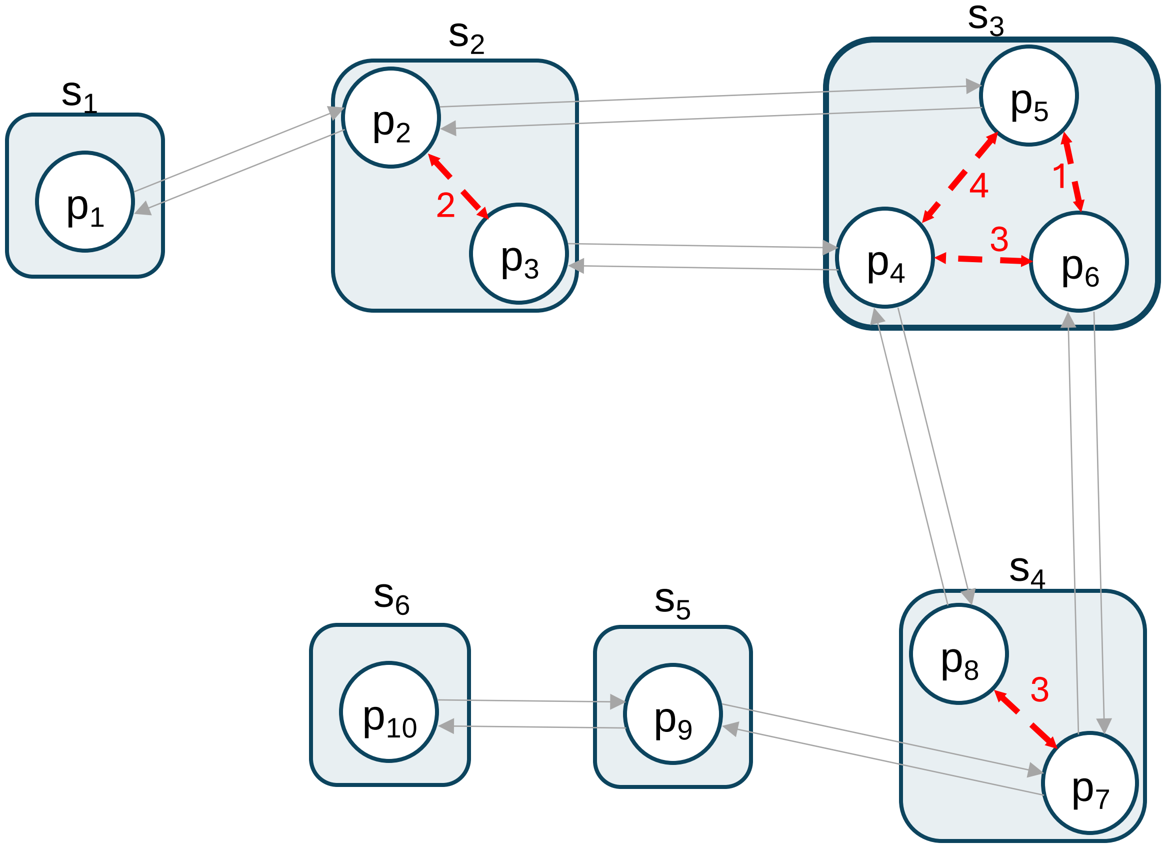
Network Modelling

2. Transfer Modelling
   - Consider exact transfer costs.



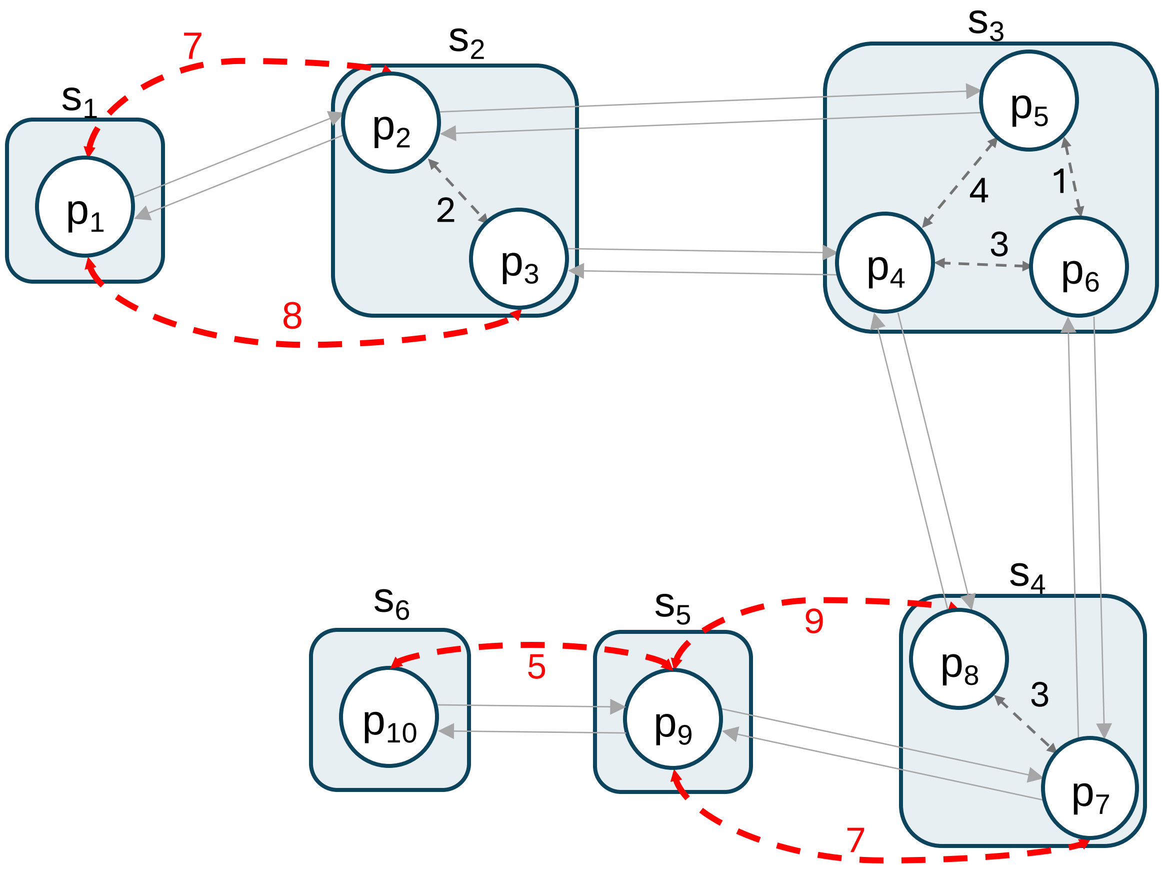
Network Modelling

2. Transfer Modelling
   - Consider exact transfer costs.
   - Add footpath between every pair of stops within stations.



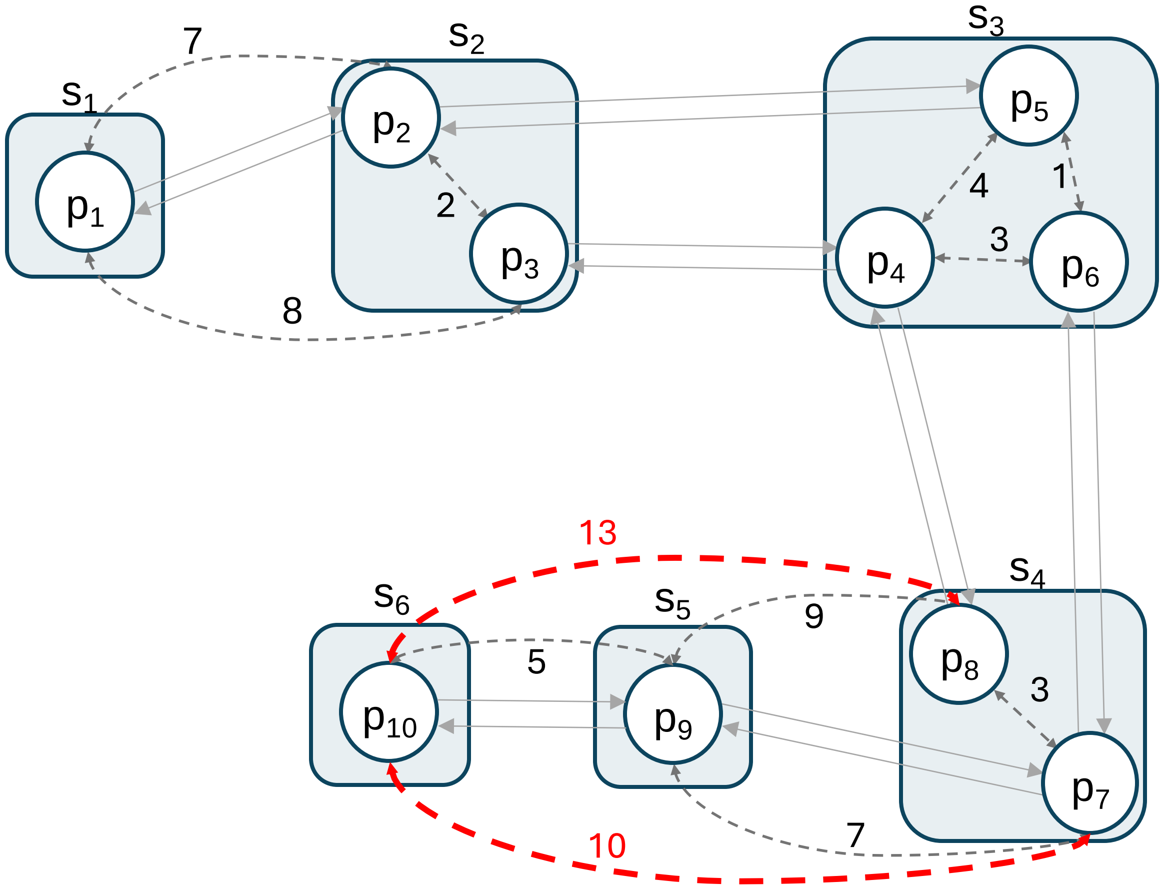
Network Modelling

2. Transfer Modelling
   - Consider exact transfer costs.
   - Add footpath between every pair of stops within stations.
   - Add footpath between every pair of stops between nearby stations.



Network Modelling

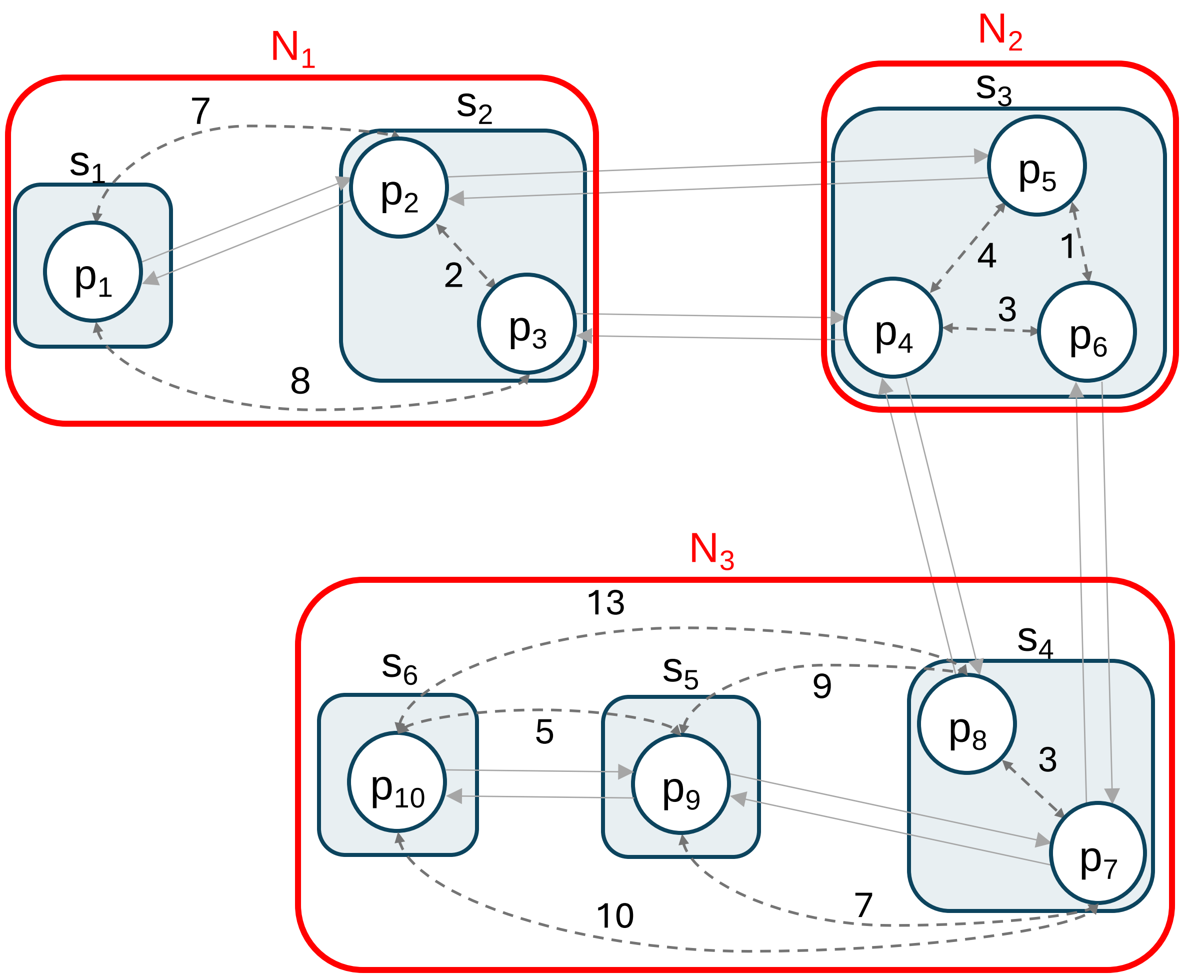
2. Transfer Modelling
   - Consider exact transfer costs.
   - Add footpath between every pair of stops within stations.
   - Add footpath between every pair of stops between nearby stations.
   - Add more footpaths to create a transitively-closed graph.



Network Modelling

2. Transfer Modelling

- Consider exact transfer costs.
- Add footpath between every pair of stops within stations.
- Add footpath between every pair of stops between nearby stations.
- Add more footpaths to create a transitively-closed graph.
- Define neighbours and neighbourhoods.



Network Modelling

3. Query Modelling
    - Focus on the earliest arrival time problem.

Network Modelling

3. Query Modelling
    - Focus on the earliest arrival time problem.
    - Observation: commencing/concluding stop at the origin/destination station hold minimal significance to users.

Network Modelling

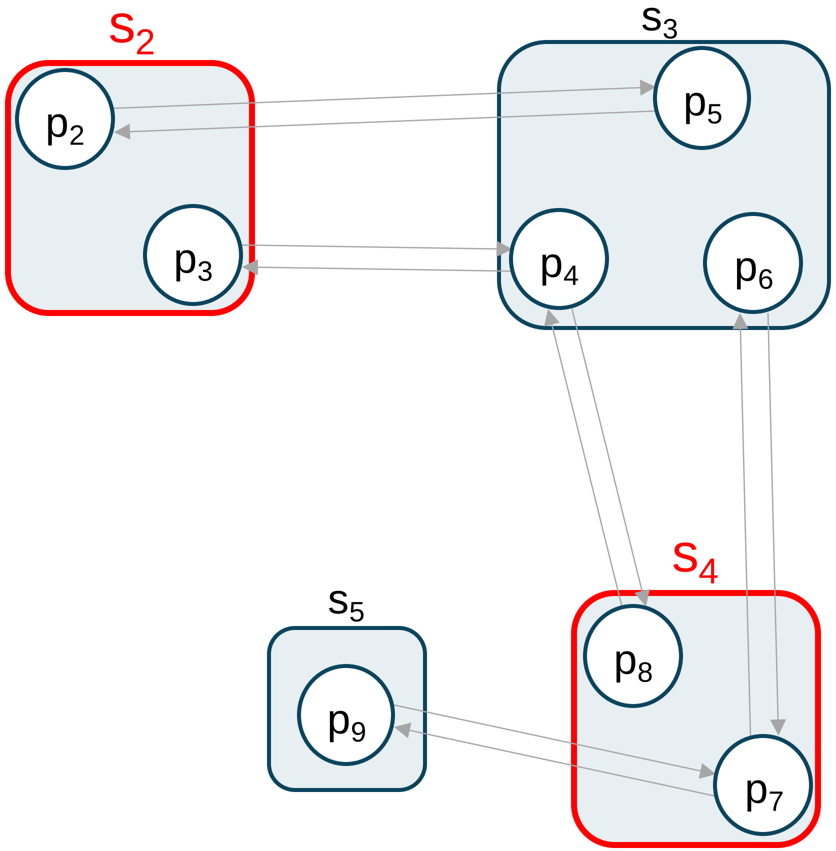
3. Query Modelling
    - Focus on the earliest arrival time problem.
    - Observation: commencing/concluding stop at the origin/destination station hold minimal significance to users.
    - Station-based query: $q = (s_o, s_d, \tau_q)$.
    - Objective: find journey j_q departing from s_o no earlier than τ_q and arriving at s_d as early as possible.

Network Modelling

3. Query Modelling
   - Focus on the earliest arrival time problem.
   - Observation: commencing/concluding stop at the origin/destination station hold minimal significance to users.
   - Station-based query: $q = (s_o, s_d, \tau_q)$.
   - Objective: find journey j_q departing from s_o no earlier than τ_q and arriving at s_d as early as possible.
   - Journey is a sequence of connections from s_o to s_d.

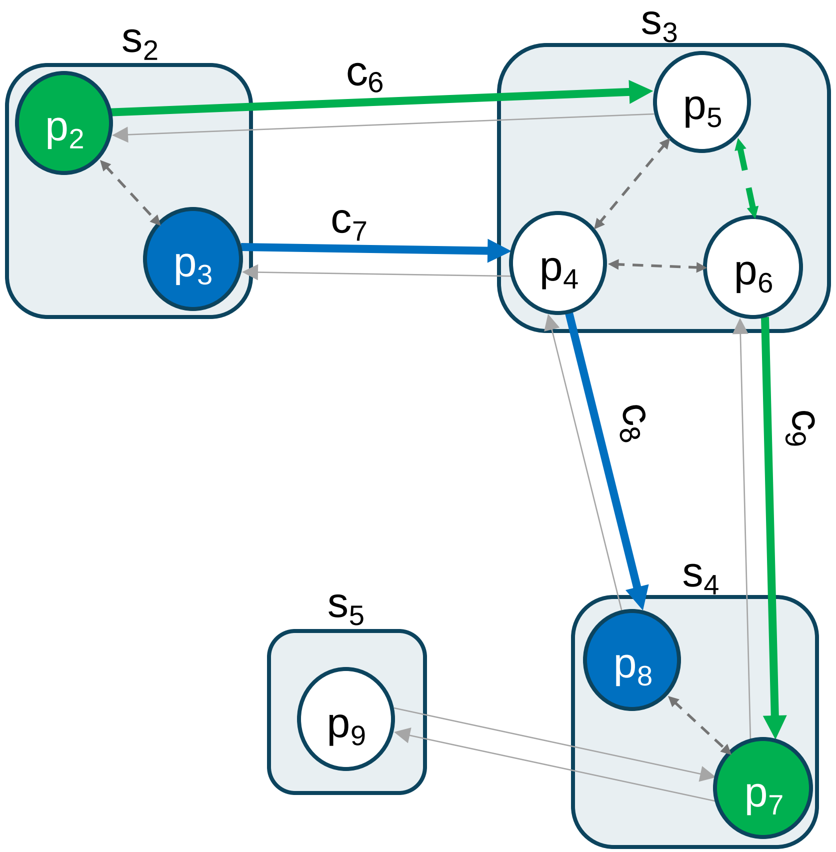
Network Modelling

3. Query Modelling
   - Focus on the earliest arrival time problem.
   - Observation: commencing/concluding stop at the origin/destination station hold minimal significance to users.
   - Station-based query: $q = (s_o, s_d, \tau_q)$.
   - Objective: find journey j_q departing from s_o no earlier than τ_q and arriving at s_d as early as possible.
   - Journey is a sequence of connections from s_o to s_d.
   - $q = (s_2, s_4, \tau_q)$



Network Modelling

3. Query Modelling
   - Focus on the earliest arrival time problem.
   - Observation: commencing/concluding stop at the origin/destination station hold minimal significance to users.
   - Station-based query: $q = (s_o, s_d, \tau_q)$.
   - Objective: find journey j_q departing from s_o no earlier than τ_q and arriving at s_d as early as possible.
   - Journey is a sequence of connections from s_o to s_d.
   - $q = (s_2, s_4, \tau_q) \rightarrow$ $j_1 = \langle c_6, c_9 \rangle$ $j_2 = \langle c_7, c_8 \rangle$



Transfer Connection Database (TCD)

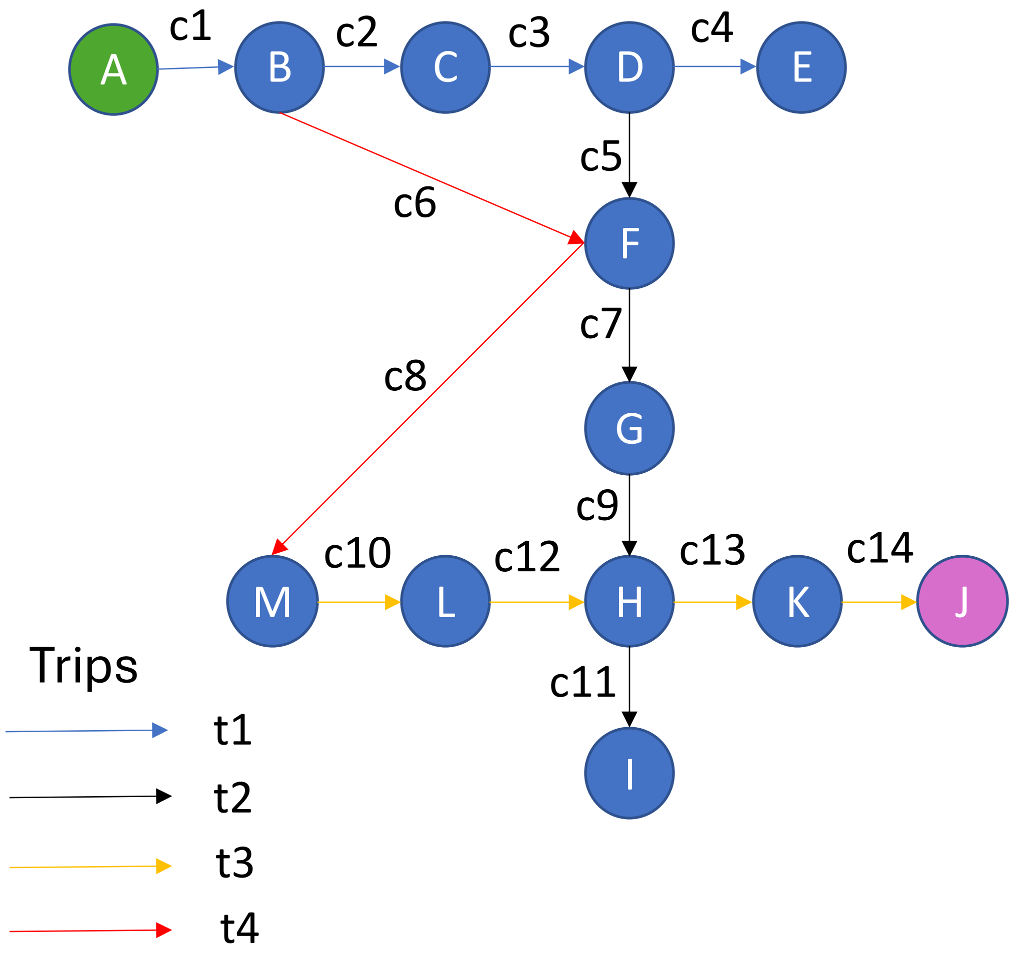
Network Modelling

Offline Preprocessing Phase

Online Query Phase

Compressed Path Database (CPD)

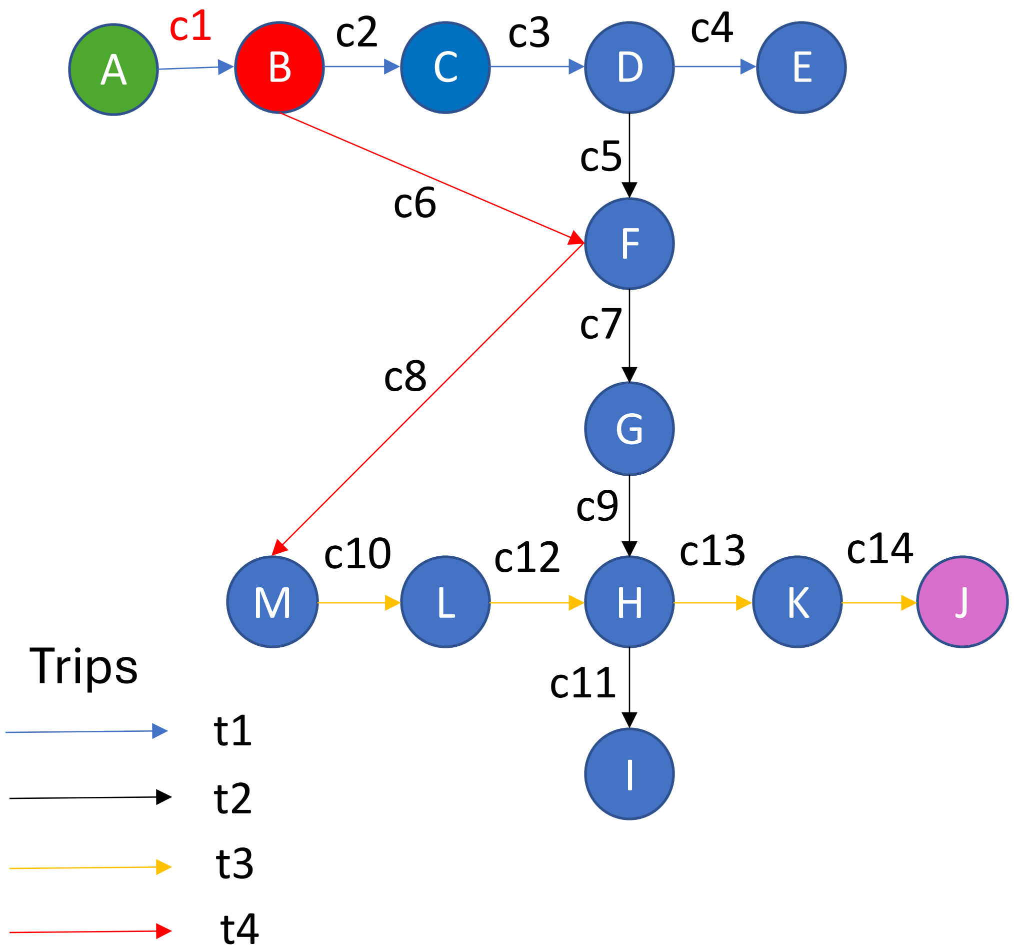
- Identify the first move on the optimal journey for any OD pair at a given departure time.
- Query q = (A, J, τ_q).
- First optimal move from A to J?



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

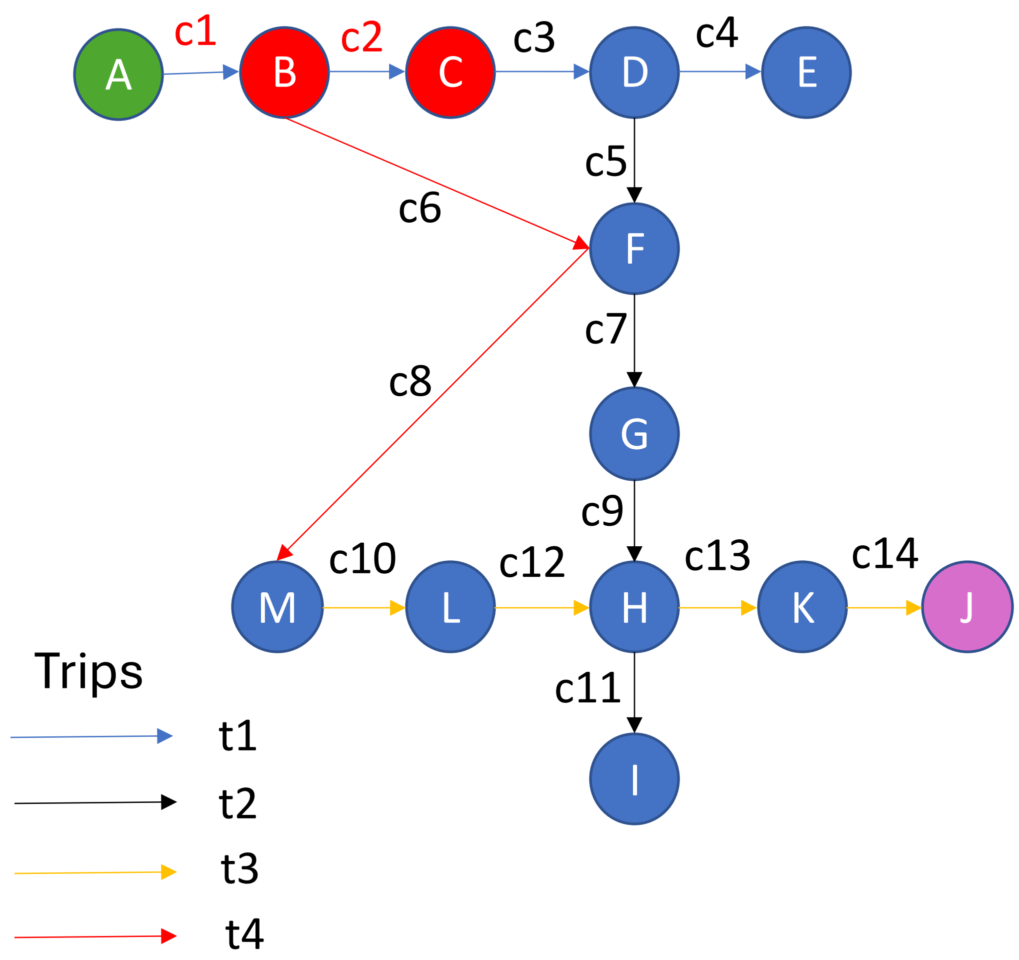
- Identify the first move on the optimal journey for any OD pair at a given departure time.
- Query q = (A, J, τ_q).
- First optimal move from A to J?
- j_q = <c1, ...>



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

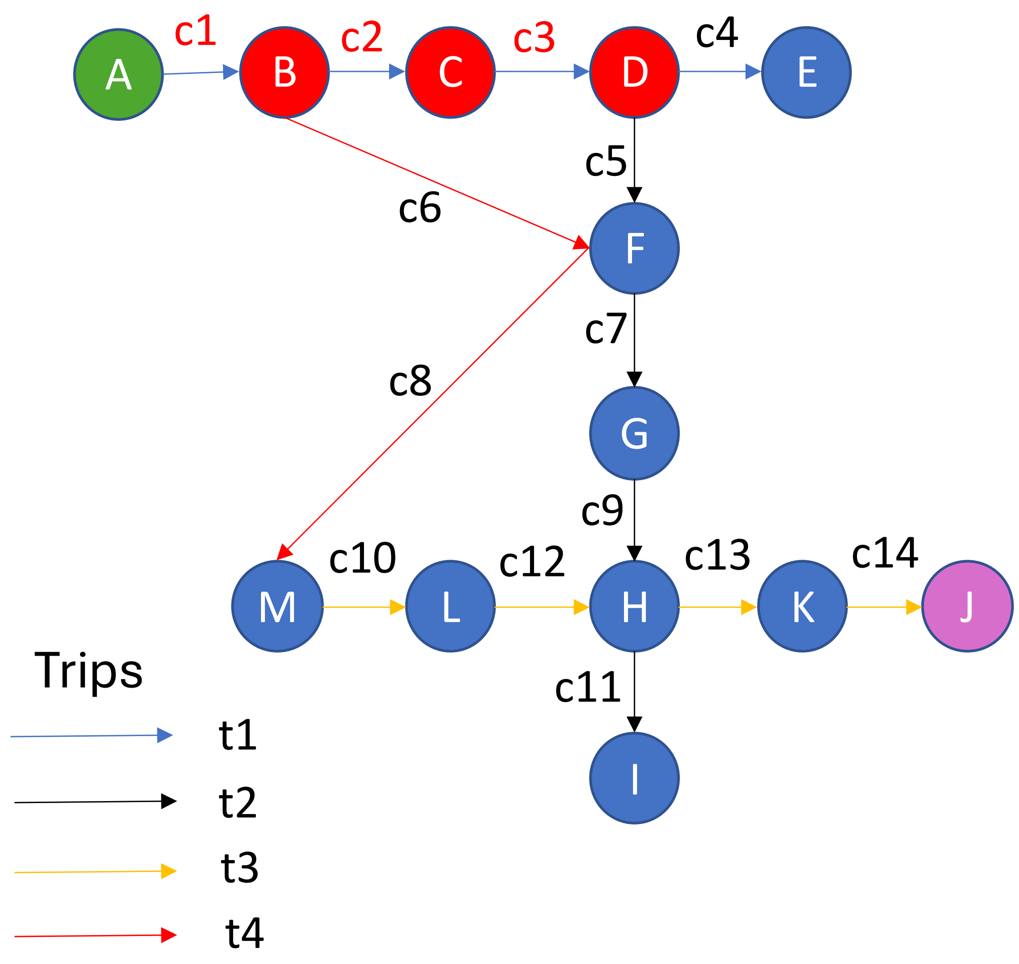
- Identify the first move on the optimal journey for any OD pair at a given departure time.
- Query q = (A, J, τ_q).
- First optimal move from B to J?
- j_q = <c1, c2, ...>



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

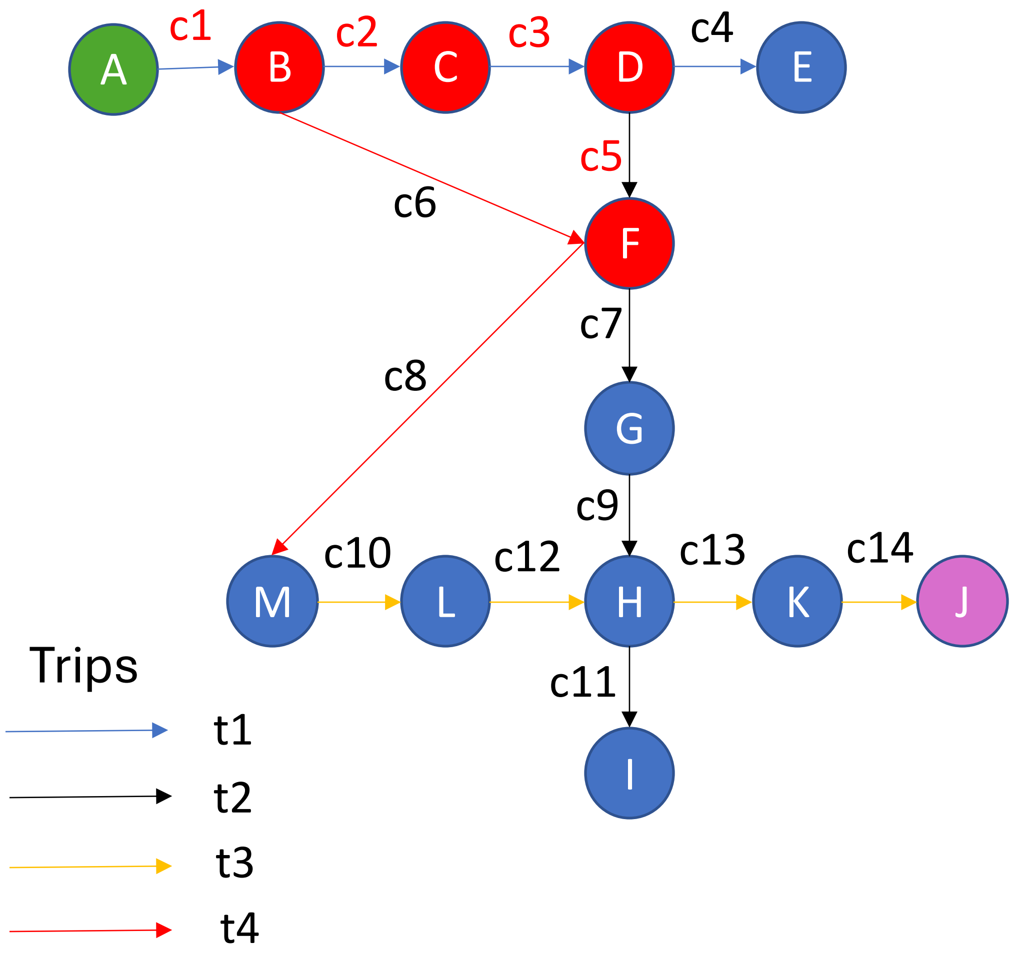
- Identify the first move on the optimal journey for any OD pair at a given departure time.
- Query q = (A, J, τ_q).
- First optimal move from C to J?
- j_q = <c1, c2, c3, ...>



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

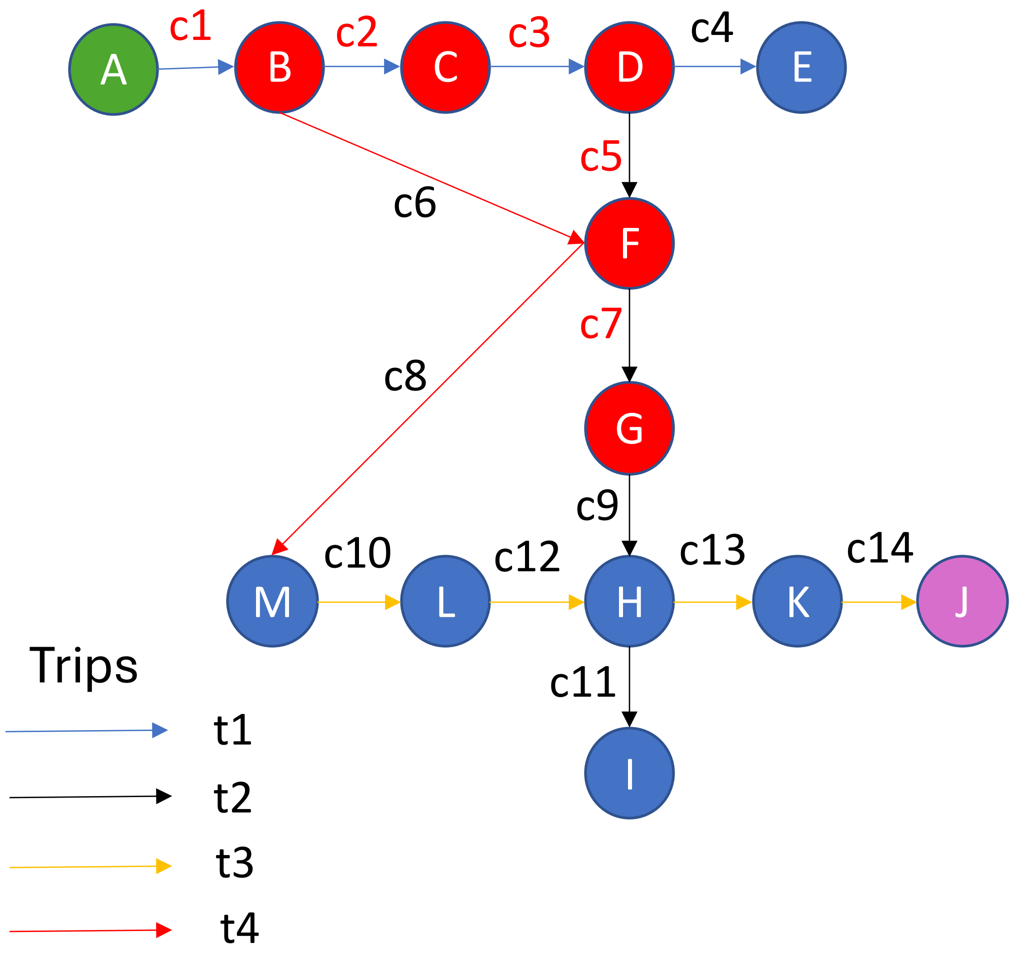
- Identify the first move on the optimal journey for any OD pair at a given departure time.
- Query q = (A, J, τ_q).
- First optimal move from D to J?
- j_q = <c1, c2, c3, c5, …>



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

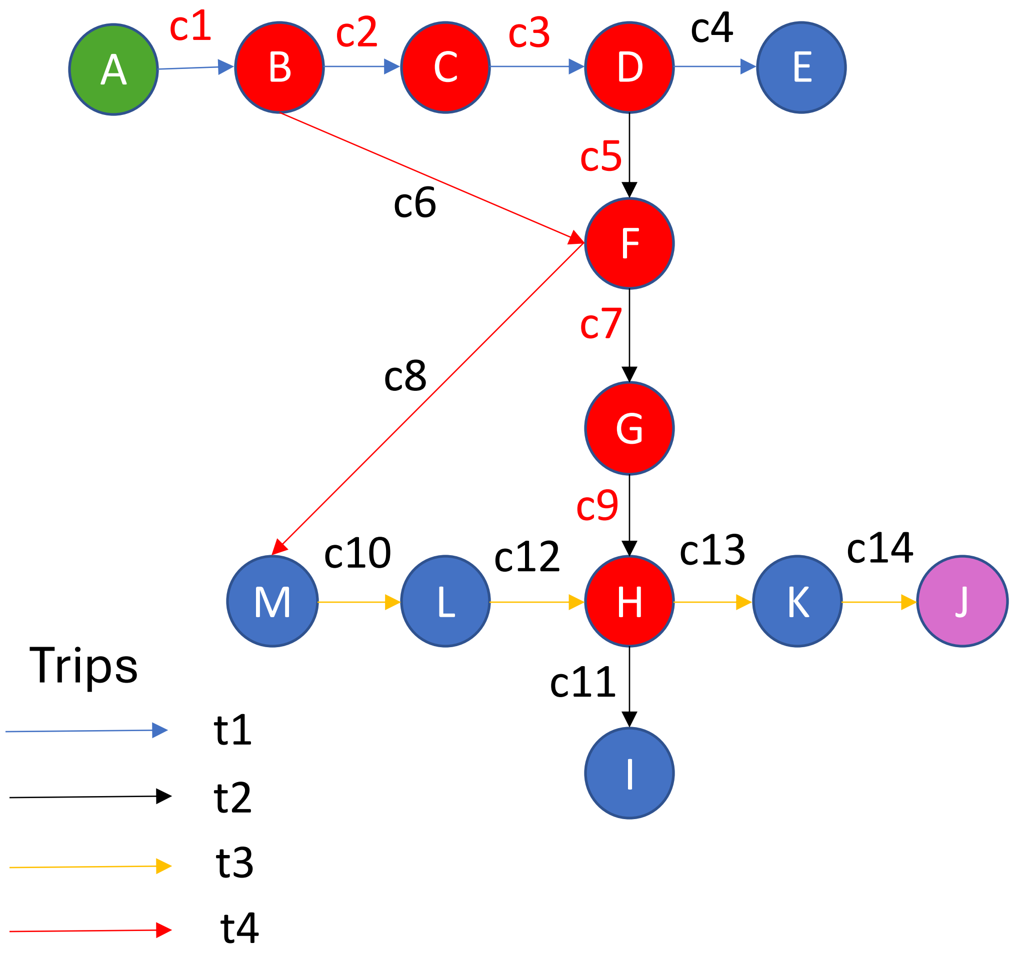
- Identify the first move on the optimal journey for any OD pair at a given departure time.
- Query q = (A, J, τ_q).
- First optimal move from F to J?
- j_q = <c1, c2, c3, c5, c7, …>



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

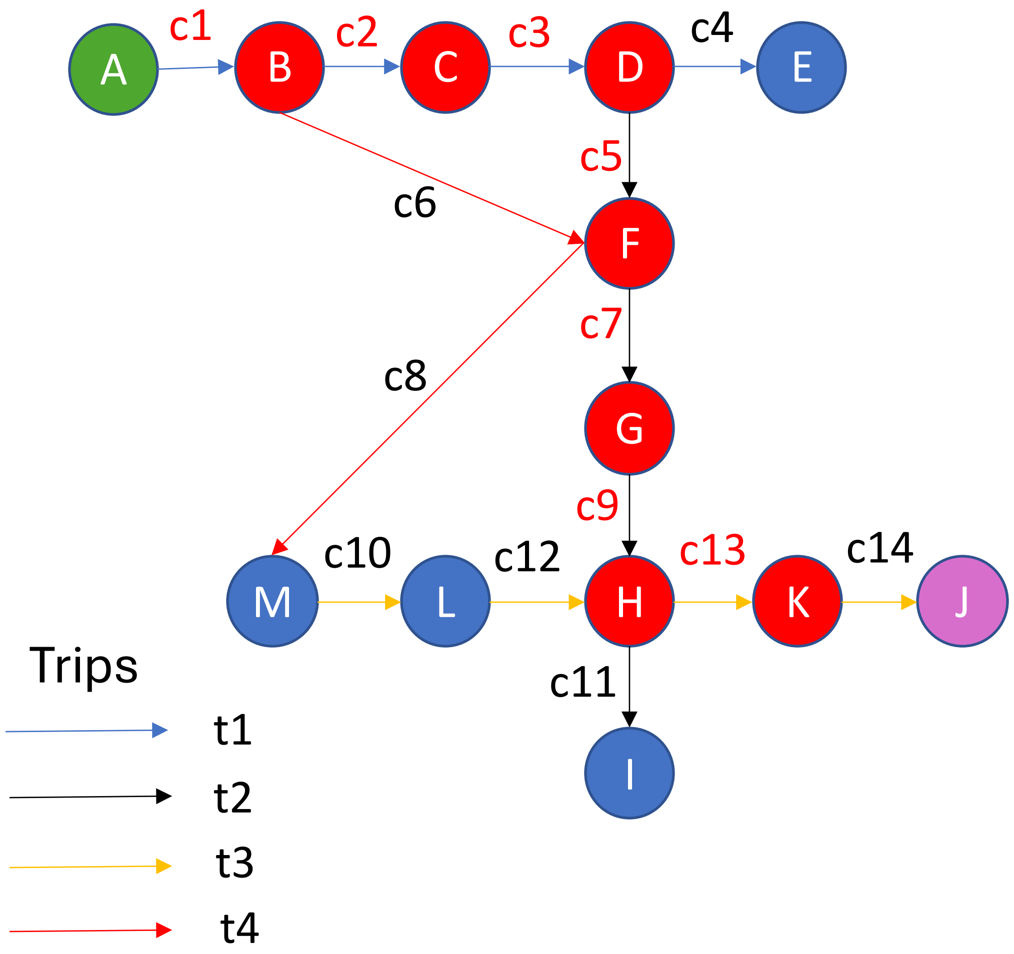
- Identify the first move on the optimal journey for any OD pair at a given departure time.
- Query q = (A, J, τ_q).
- First optimal move from G to J?
- j_q = <c1, c2, c3, c5, c7, c9, ...>



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

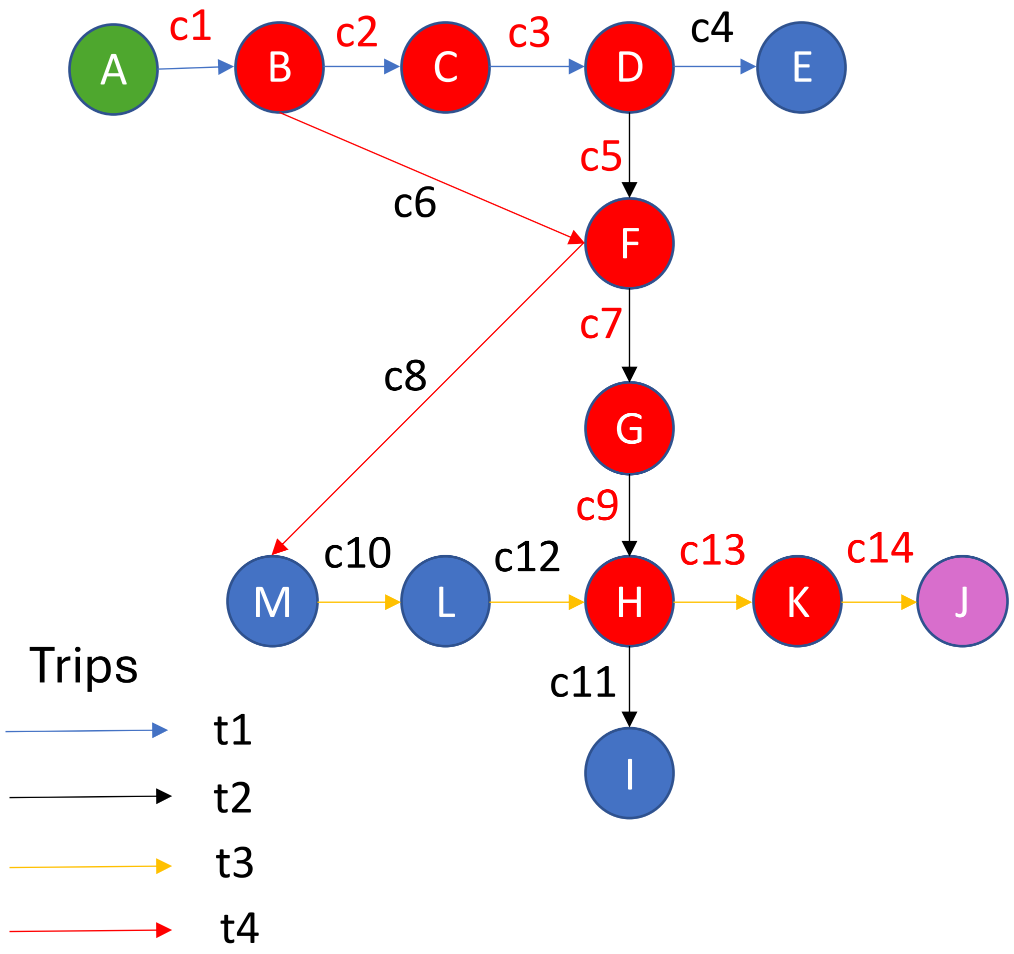
- Identify the first move on the optimal journey for any OD pair at a given departure time.
- Query $q = (A, J, \tau_q)$.
- First optimal move from H to J?
- j_q = <c1, c2, c3, c5, c7, c9, c13, ...>



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

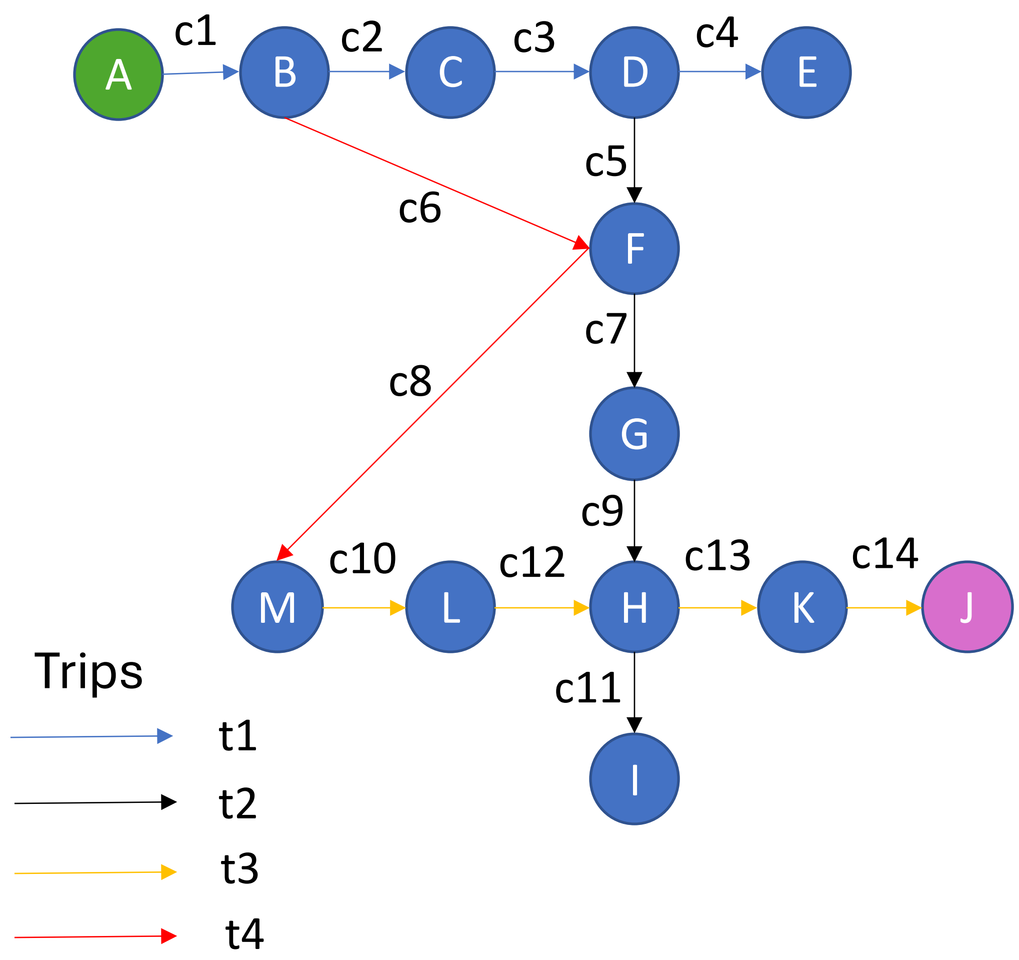
- Identify the first move on the optimal journey for any OD pair at a given departure time.
- Query q = (A, J, τ_q).
- First optimal move from K to J?
- j_q = <c1, c2, c3, c5, c7, c9, c13, c14>
- 8 lookups!



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

- Observation: information about journey transfers is sufficient to answer queries.
- First move ← next transfer connection instead of next connection.



* Assuming one stop in each station for simplicity

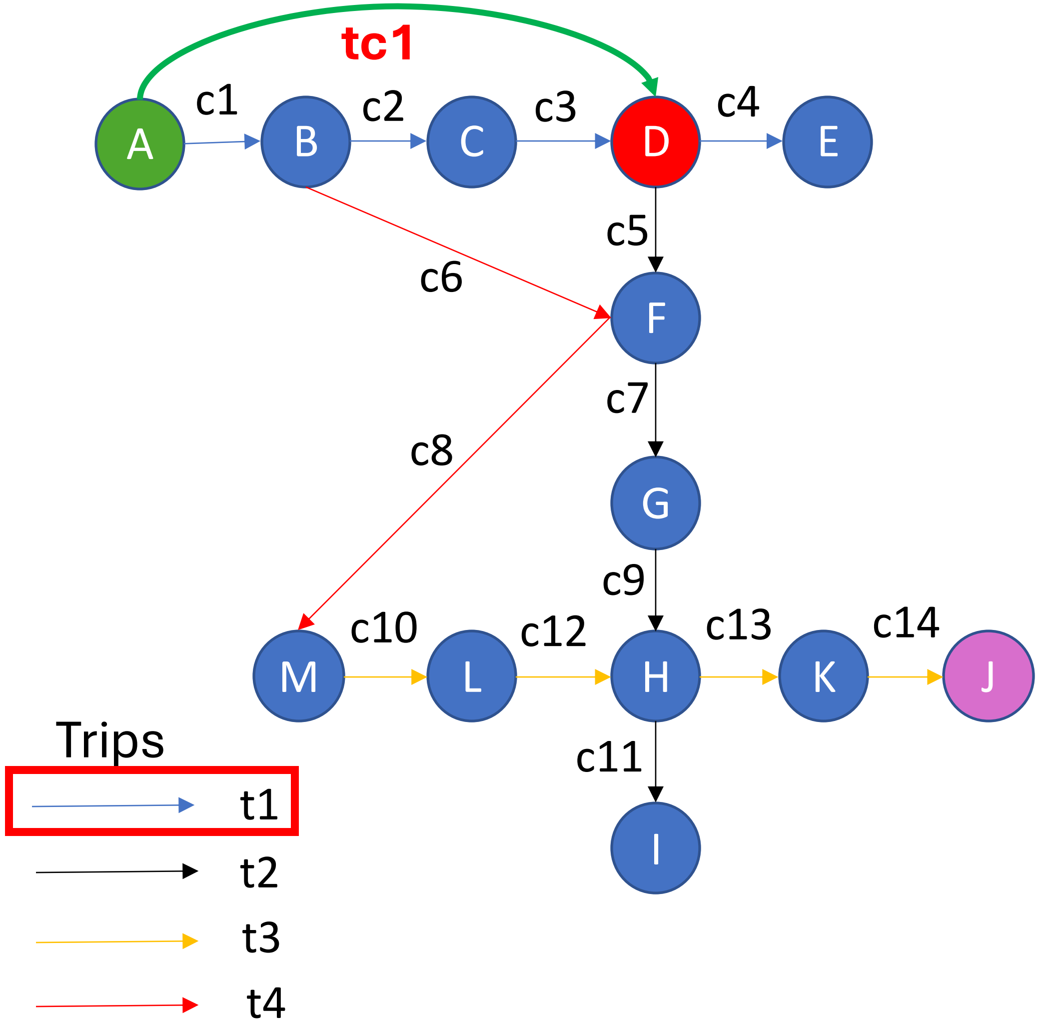
Compressed Path Database (CPD)

- Observation: information about journey transfers is sufficient to answer queries.
- First move ← next transfer connection instead of next connection.
- Transfer connection is a sequence of connections sharing the same trip.

  j_q = <c1, c2, c3, c5, c7, c9, c13, c14>

  tc1 = <c1, c2, c3>
- Can be represented by the first and last connections

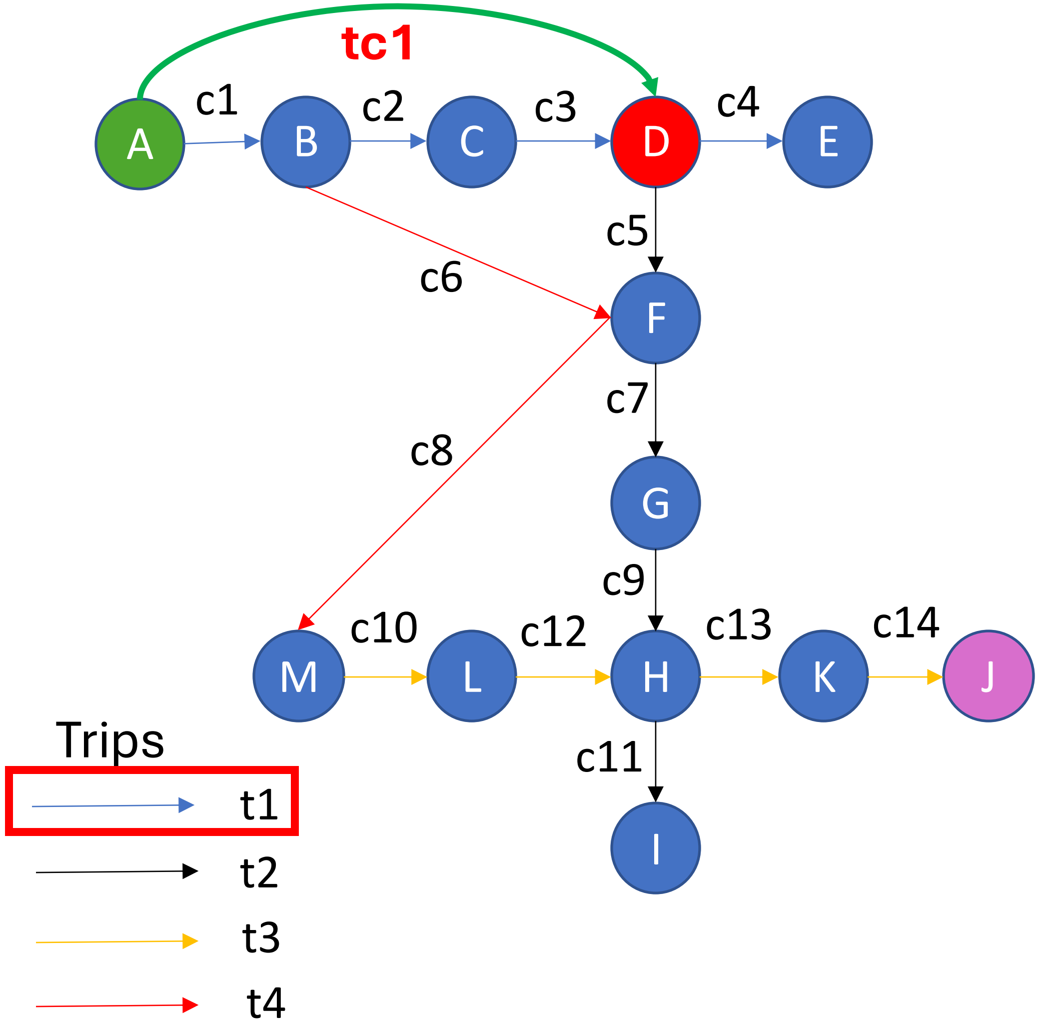
  tc1 = (c1, c3)



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

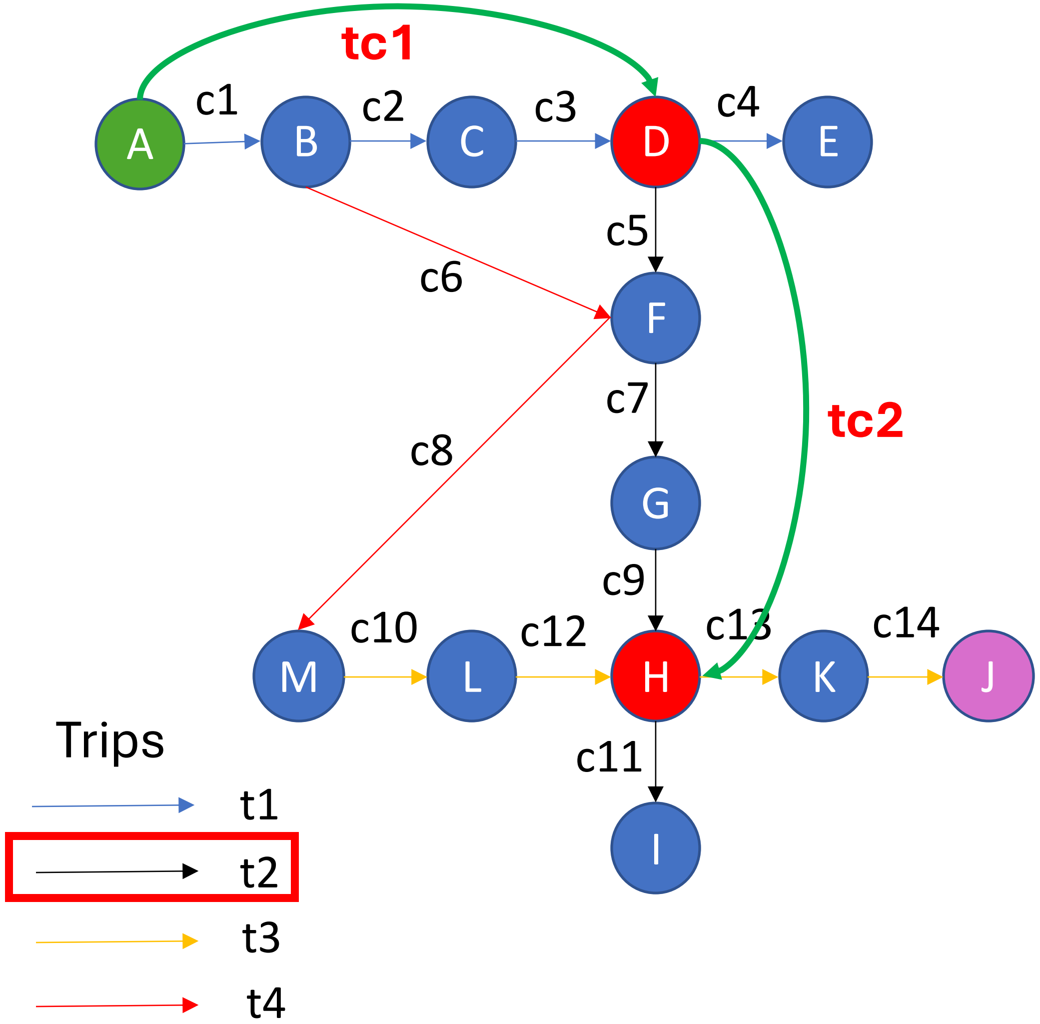
- Observation: information about journey transfers is sufficient to answer queries.
- First move ← next transfer connection instead of next connection.
- First optimal move from A to J?
- jq = <tc1, ...>



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

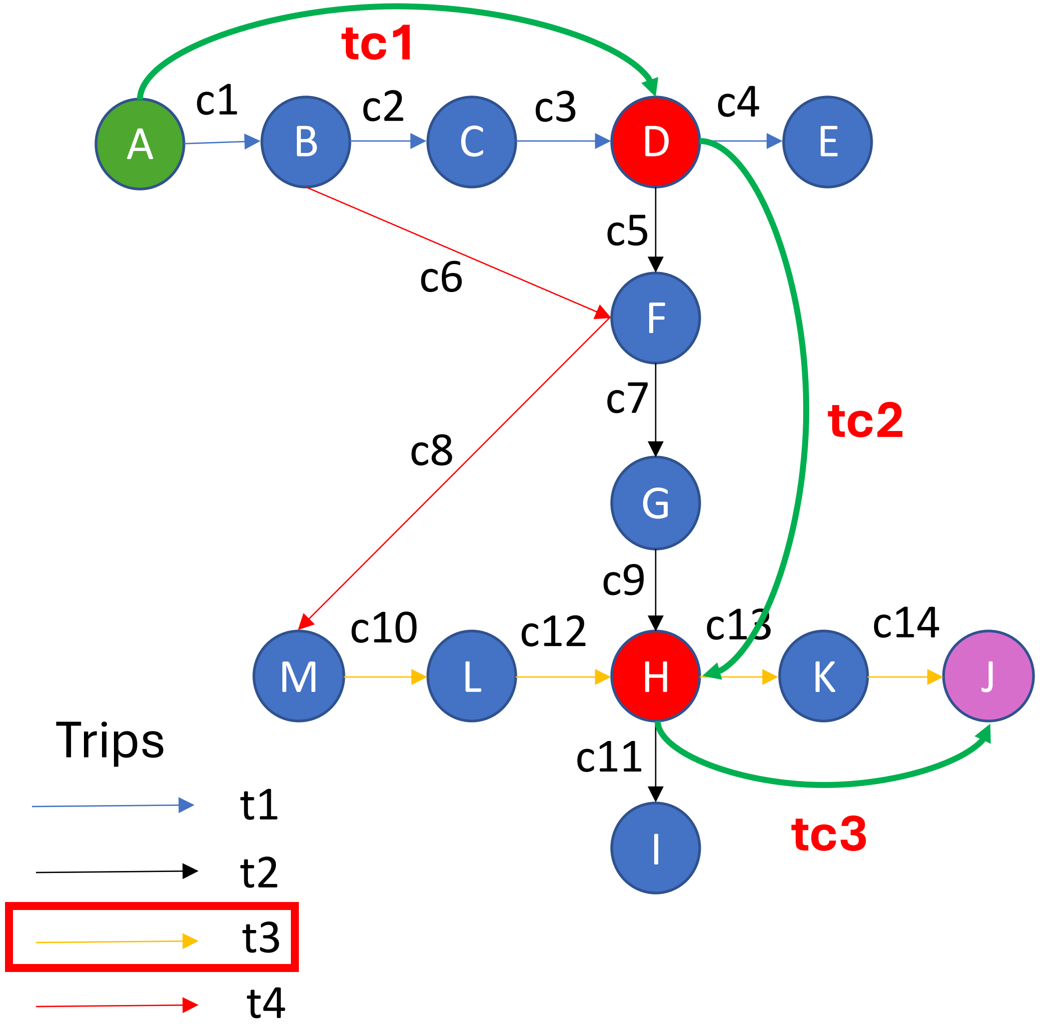
- Observation: information about journey transfers is sufficient to answer queries.
- First move ← next transfer connection instead of next connection.
- First optimal move from D to J?
- jq = <tc1, tc2, ...>



* Assuming one stop in each station for simplicity

Compressed Path Database (CPD)

- Observation: information about journey transfers is sufficient to answer queries.
- First move ← next transfer connection instead of next connection.
- First optimal move from H to J?
- jq = <tc1, tc2, tc3>
- 3 lookups only (instead of 8)!
- Number of transfers is small in practice.



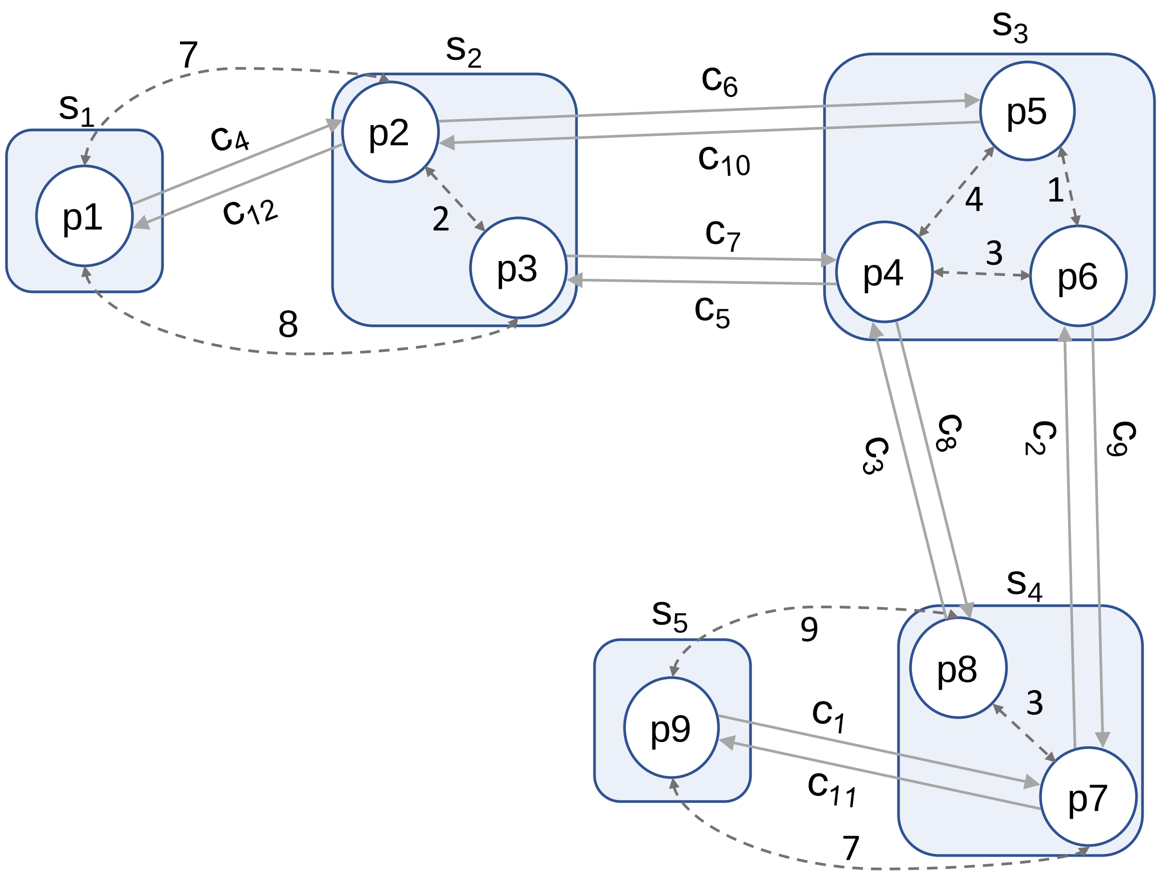
* Assuming one stop in each station for simplicity

Offline Preprocessing Phase

- Objective: Build a table that stores all first moves (transfer connections) for all OD pairs considering all potential departure times, called FT table.
- Naïve approach: station-station FT table.

Offline Preprocessing Phase

- **Naïve approach: Station-station**

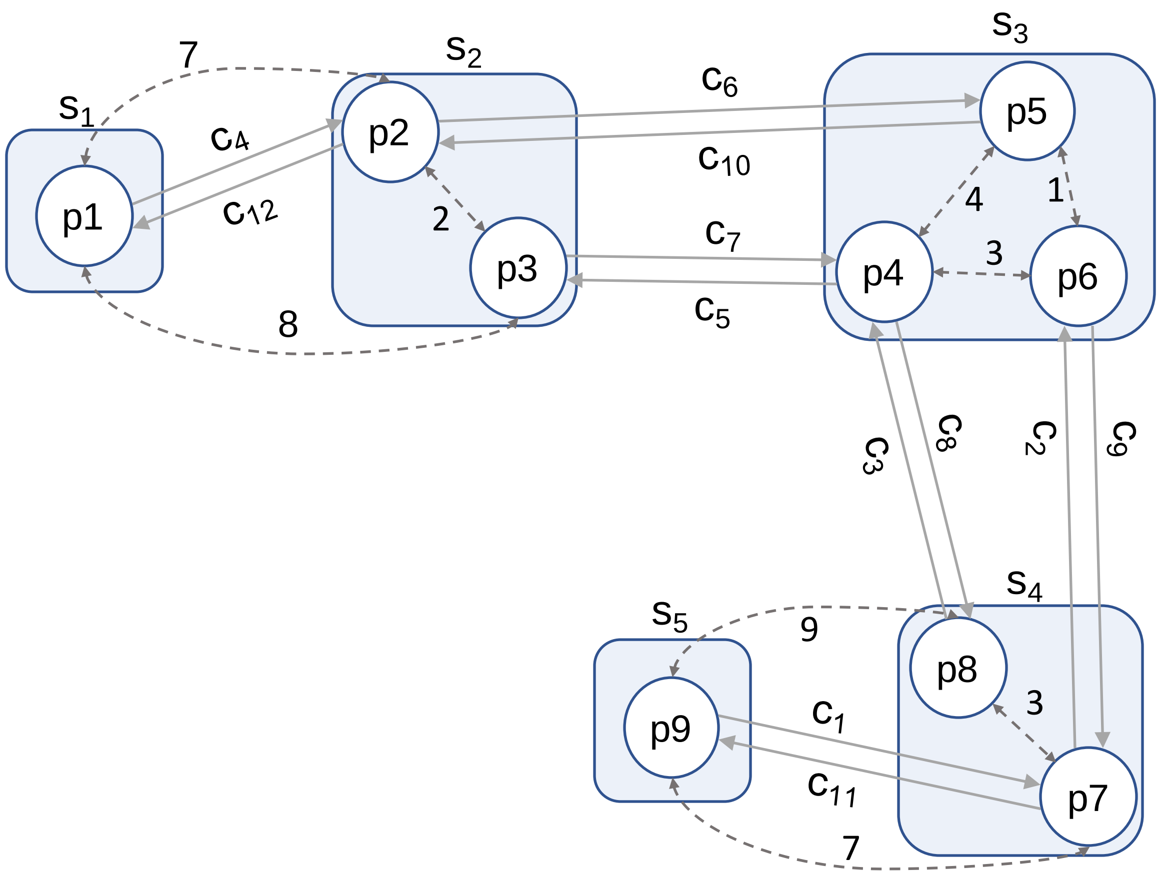


Station-station FT table

O\D	s_1	s_2	s_3	s_4	s_5
s_1	--	(c_4,c_4)	(c_6,c_6) (c_4,c_6) (c_7,c_7)	(c_7,c_8) (c_4,c_4) (c_6,c_6)	(c_7,c_8) (c_4,c_4) (c_6,c_6)
s_2	(c_{12},c_{12}) (c_7,c_7) (c_6,c_6)	--	(c_6,c_6) (c_4,c_6) (c_7,c_7)	(c_7,c_8) (c_4,c_4) (c_6,c_6)	(c_7,c_8) (c_4,c_4) (c_6,c_6)
s_3	(c_5,c_5) (c_{10},c_{12})	(c_5,c_5) (c_{10},c_{10})	--	(c_8,c_8) (c_9,c_9)	(c_8,c_8) (c_9,c_{11})
s_4	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_2,c_2) (c_1,c_2) (c_3,c_3)	--	(c_3,c_3) (c_2,c_2) (c_{11},c_{11})
s_5	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_2,c_2) (c_1,c_2) (c_3,c_3)	(c_1,c_1)	--

Offline Preprocessing Phase

- Issue: redundant labels!
- Cause: journey can start with walking.

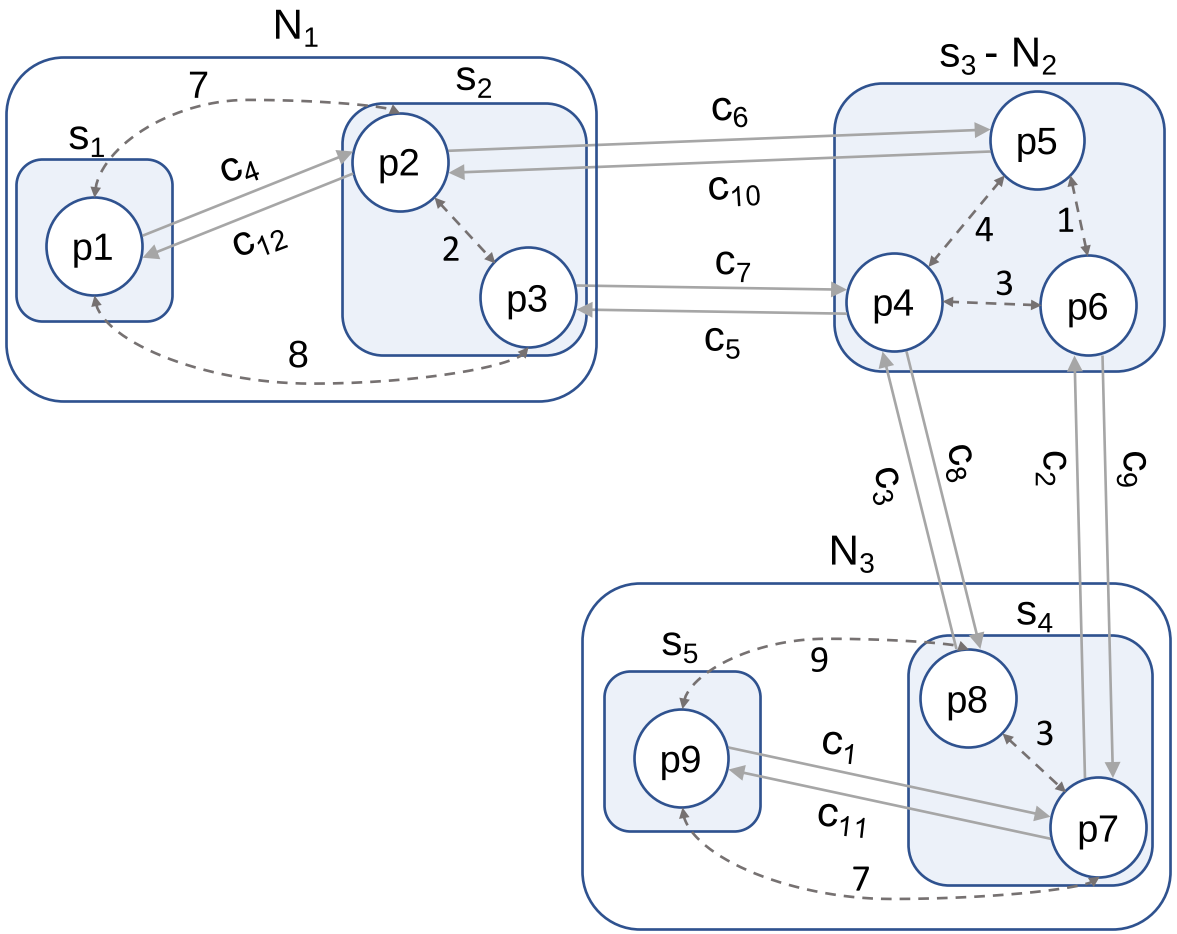


Station-station FT table

O\D	s_1	s_2	s_3	s_4	s_5
s_1	--	(c_4,c_4)	(c_6,c_6) (c_4,c_6) (c_7,c_7)	(c_7,c_8) (c_4,c_4) (c_6,c_6)	(c_7,c_8) (c_4,c_4) (c_6,c_6)
s_2	(c_{12},c_{12}) (c_7,c_7) (c_6,c_6)	--	(c_6,c_6) (c_4,c_6) (c_7,c_7)	(c_7,c_8) (c_4,c_4) (c_6,c_6)	(c_7,c_8) (c_4,c_4) (c_6,c_6)
s_3	(c_5,c_5) (c_{10},c_{12})	(c_5,c_5) (c_{10},c_{10})	--	(c_8,c_8) (c_9,c_9)	(c_8,c_8) (c_9,c_{11})
s_4	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_2,c_2) (c_1,c_2) (c_3,c_3)	--	(c_3,c_3) (c_2,c_2) (c_{11},c_{11})
s_5	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_2,c_2) (c_1,c_2) (c_3,c_3)	(c_1,c_1)	--

Offline Preprocessing Phase

- **Refined approach: Neighbourhood-station**

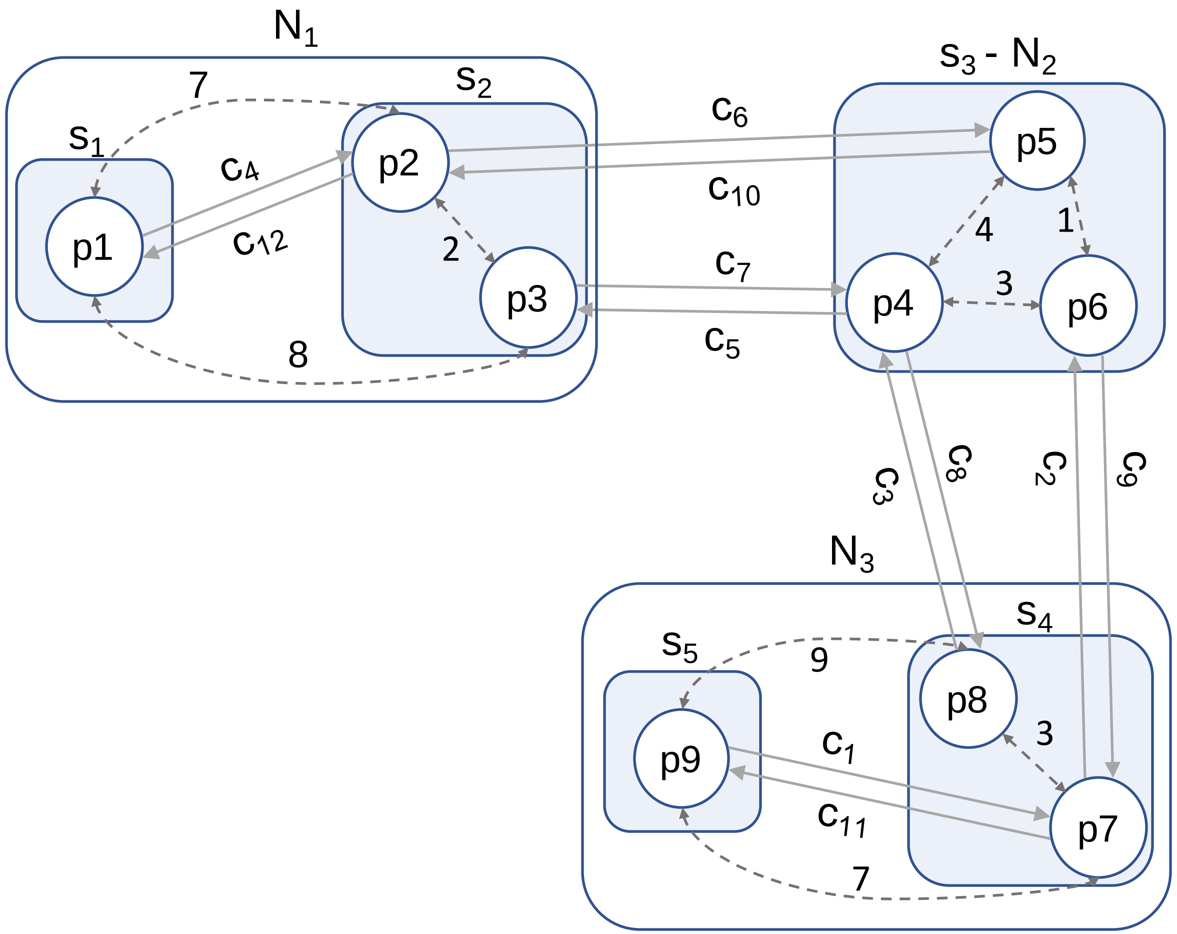


Neighbourhood-station FT table

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},c_{12}) (c_7,c_7) (c_6,c_6)	(c_4,c_4)	(c_6,c_6) (c_4,c_6) (c_7,c_7)	(c_7,c_8) (c_4,c_4) (c_6,c_6)	(c_7,c_8) (c_4,c_4) (c_6,c_6)
N_2	(c_5,c_5) (c_{10},c_{12})	(c_5,c_5) (c_{10},c_{10})	--	(c_8,c_8) (c_9,c_9)	(c_8,c_8) (c_9,c_{11})
N_3	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_2,c_2) (c_1,c_2) (c_3,c_3)	(c_1,c_1)	(c_3,c_3) (c_2,c_2) (c_{11},c_{11})

Offline Preprocessing Phase

- **Refined approach: Neighbourhood-station**



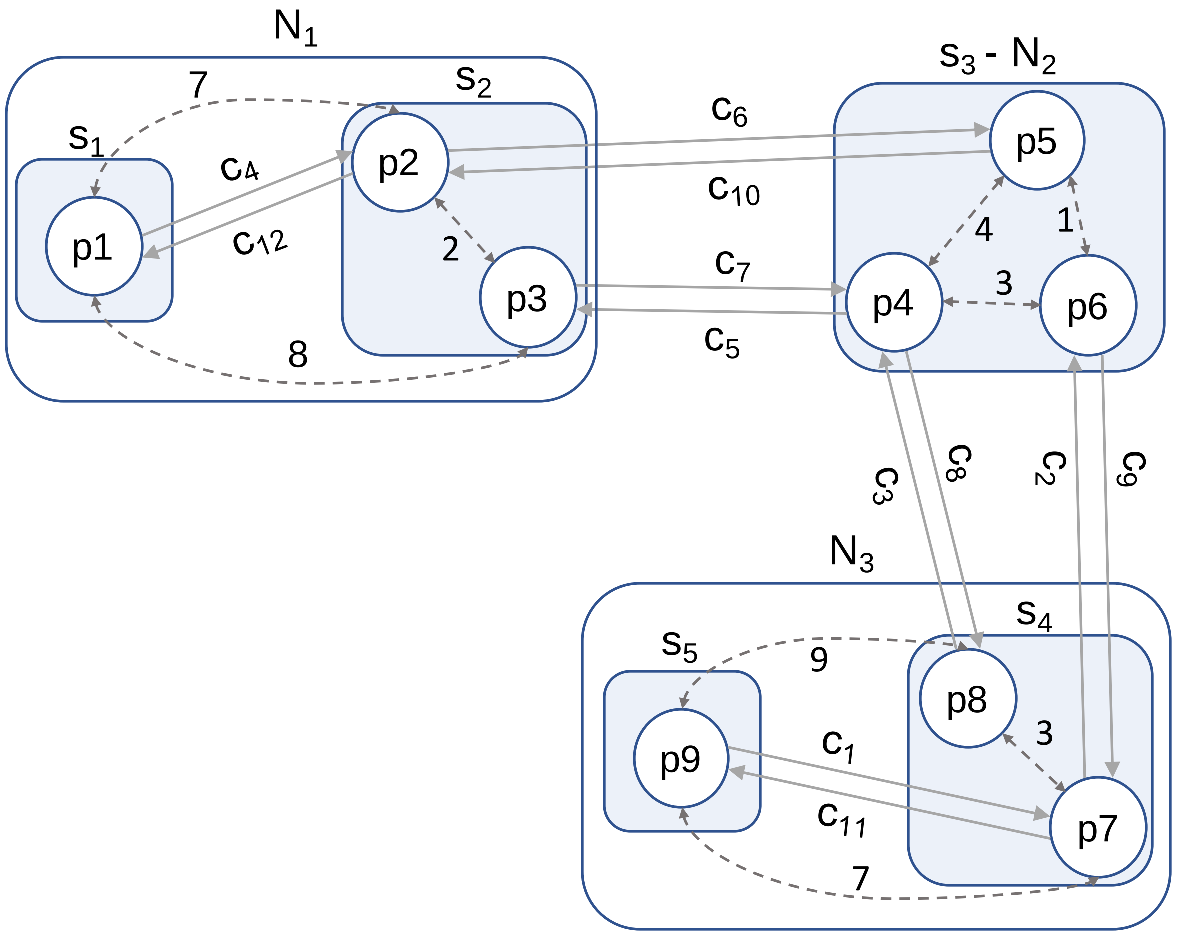
Neighbourhood-station FT table

Origin neighbourhoods

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},c_{12}) (c_7,c_7) (c_6,c_6)	(c_4,c_4)	(c_6,c_6) (c_4,c_6) (c_7,c_7)	(c_7,c_8) (c_4,c_4) (c_6,c_6)	(c_7,c_8) (c_4,c_4) (c_6,c_6)
N_2	(c_5,c_5) (c_{10},c_{12})	(c_5,c_5) (c_{10},c_{10})	--	(c_8,c_8) (c_9,c_9)	(c_8,c_8) (c_9,c_{11})
N_3	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_2,c_2) (c_1,c_2) (c_3,c_3)	(c_1,c_1)	(c_3,c_3) (c_2,c_2) (c_{11},c_{11})

Offline Preprocessing Phase

- **Refined approach: Neighbourhood-station**



Neighbourhood-station FT table

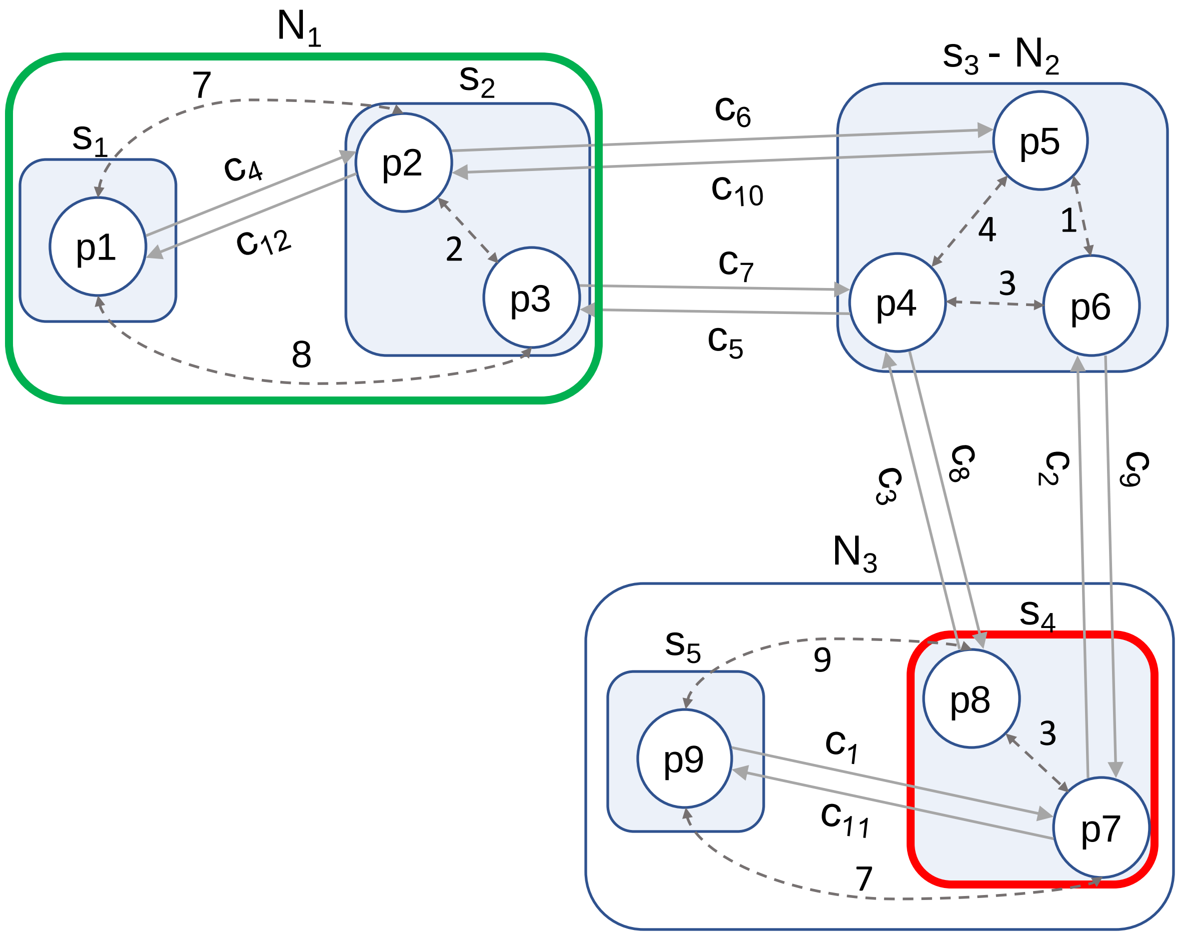
Destination stations

Origin neighbourhoods

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},c_{12}) (c_7,c_7) (c_6,c_6)	(c_4,c_4)	(c_6,c_6) (c_4,c_6) (c_7,c_7)	(c_7,c_8) (c_4,c_4) (c_6,c_6)	(c_7,c_8) (c_4,c_4) (c_6,c_6)
N_2	(c_5,c_5) (c_{10},c_{12})	(c_5,c_5) (c_{10},c_{10})	--	(c_8,c_8) (c_9,c_9)	(c_8,c_8) (c_9,c_{11})
N_3	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_2,c_2) (c_1,c_2) (c_3,c_3)	(c_1,c_1)	(c_3,c_3) (c_2,c_2) (c_{11},c_{11})

Offline Preprocessing Phase

- **Refined approach: Neighbourhood-station**



FT[N_1][s_4]

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},c_{12}) (c_7,c_7) (c_6,c_6)	(c_4,c_4)	(c_6,c_6) (c_4,c_6) (c_7,c_7)	(c_7,c_8) (c_4,c_4) (c_6,c_6)	(c_7,c_8) (c_4,c_4) (c_6,c_6)
N_2	(c_5,c_5) (c_{10},c_{12})	(c_5,c_5) (c_{10},c_{10})	--	(c_8,c_8) (c_9,c_9)	(c_8,c_8) (c_9,c_{11})
N_3	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_2,c_2) (c_1,c_2) (c_3,c_3)	(c_1,c_1)	(c_3,c_3) (c_2,c_2) (c_{11},c_{11})

+ arrival @ s_4

Offline Preprocessing Phase

- All optimal journeys have to be precomputed for all OD pairs, and then the first moves on these journeys are stored in the FT table.
- To compute the row FT[N_i], run one Dijkstra search for every departure event from the origin neighbourhood N_i.

Offline Preprocessing Phase

Challenges in building the FT table

1. Computation time

Offline Preprocessing Phase

Challenges in building the FT table

1. Computation time
   - Solution: use Connection Scan Algorithm (CSA) instead of Dijkstra.
   - CSA is faster, cache-efficient, and does not require priority queue.

Offline Preprocessing Phase

Challenges in building the FT table

2. Storage (FT table size)

Offline Preprocessing Phase

Challenges in building the FT table

2. Storage (FT table size)
   - Solution: propose optimisations.

   a) Dominance Check (DC): Remove all-time dominated transfer connections in each cell of FT table.

Offline Preprocessing Phase

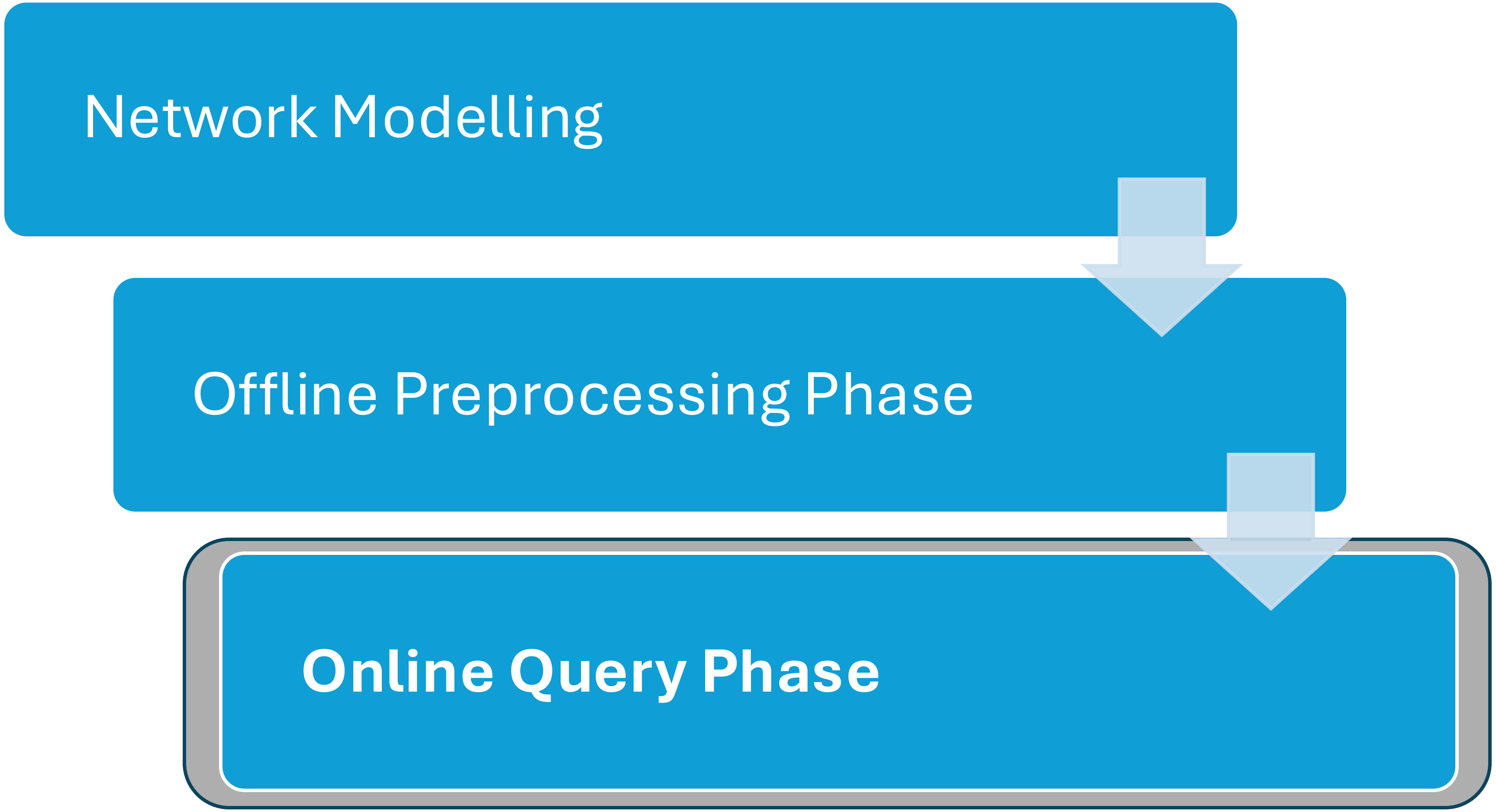
Challenges in building the FT table

2. Storage (FT table size)
   - Solution: propose optimisations.

   a) Dominance Check (DC): Remove all-time dominated transfer connections in each cell of FT table.

   b) Transfer Connection Compression (TCC): Change the representation of transfer connections to enable label merging in each cell of FT table.

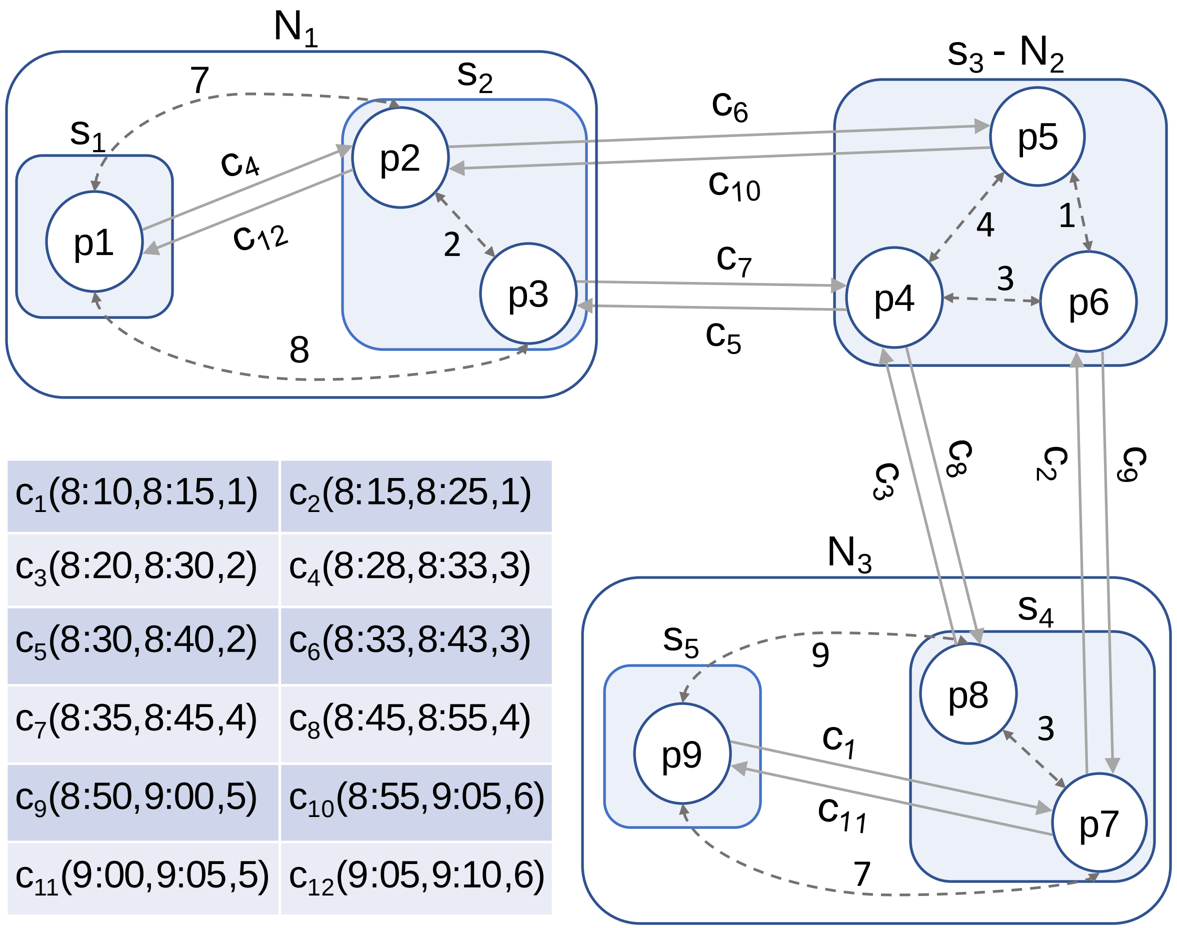
Transfer Connection Database (TCD)



Online Query Phase

Example: $q = (s_2, s_5, 8{:}30)$

j_q = <???>



c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

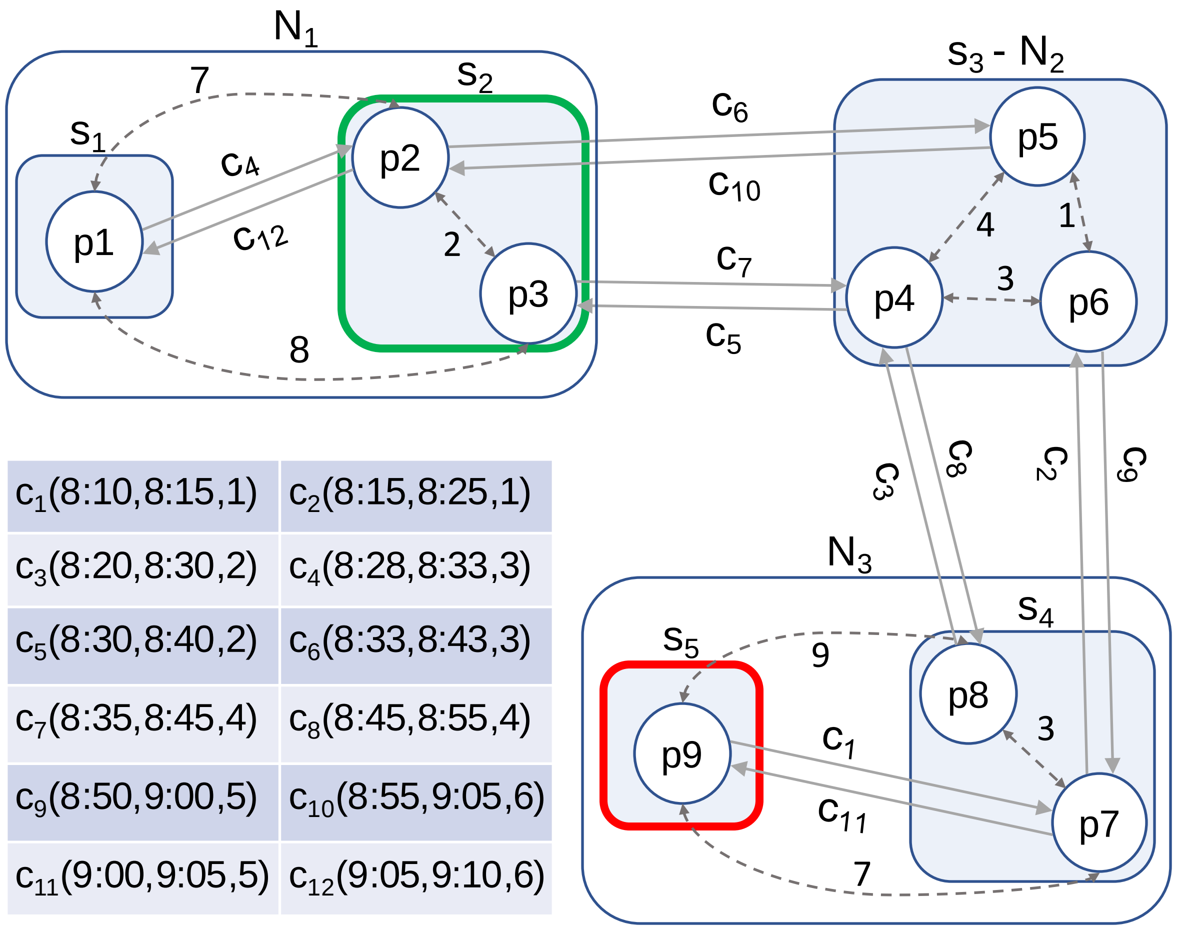
Online Query Phase

Example: q = (s_2, s_5, 8:30)

j_q = <???>

FT table

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},p_1)	(c_4,p_2)	(c_6,p_5) (c_4,p_5) (c_7,p_4)	(c_7,p_8) (c_4,p_2)	(c_7,p_8) (c_4,p_2)
N_2	(c_5,p_3) (c_{10},p_1)	(c_5,p_3) (c_{10},p_2)	--	(c_8,p_8) (c_9,p_7)	(c_8,p_8) (c_9,p_9)
N_3	(c_3,p_3)	(c_3,p_3)	(c_2,p_6) (c_1,p_6) (c_3,p_4)	(c_1,p_7)	(c_3,p_4) (c_{11},p_9)



c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

Online Query Phase

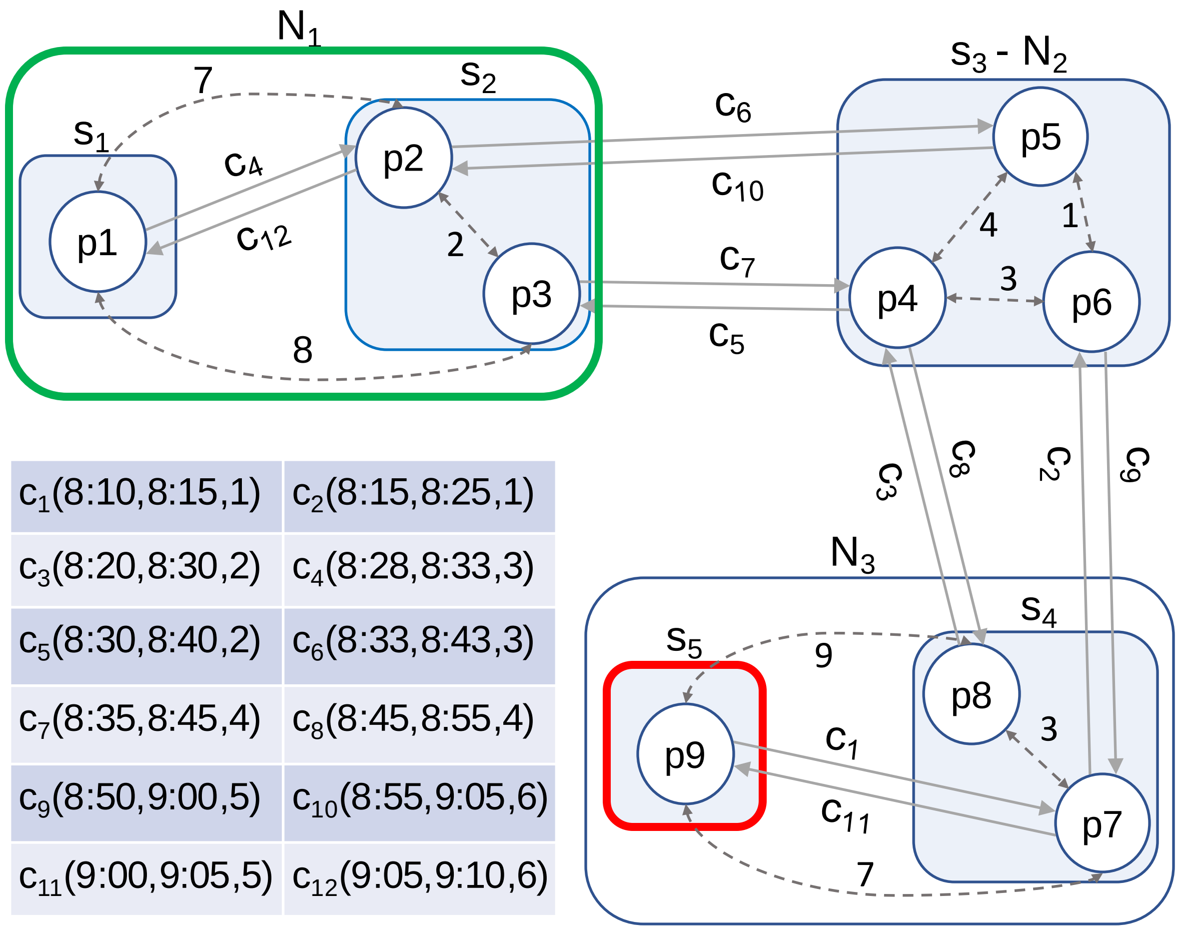
Example: q = (s_2, s_5, 8:30)

- s_2 belongs to N_1

FT[N_1][s_5]

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},p_1)	(c_4,p_2)	(c_6,p_5) (c_4,p_5) (c_7,p_4)	(c_7,p_8) (c_4,p_2)	(c_7,p_8) (c_4,p_2)
N_2	(c_5,p_3) (c_{10},p_1)	(c_5,p_3) (c_{10},p_2)	--	(c_8,p_8) (c_9,p_7)	(c_8,p_8) (c_9,p_9)
N_3	(c_3,p_3)	(c_3,p_3)	(c_2,p_6) (c_1,p_6) (c_3,p_4)	(c_1,p_7)	(c_3,p_4) (c_{11},p_9)

j_q = <???>



c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

Online Query Phase

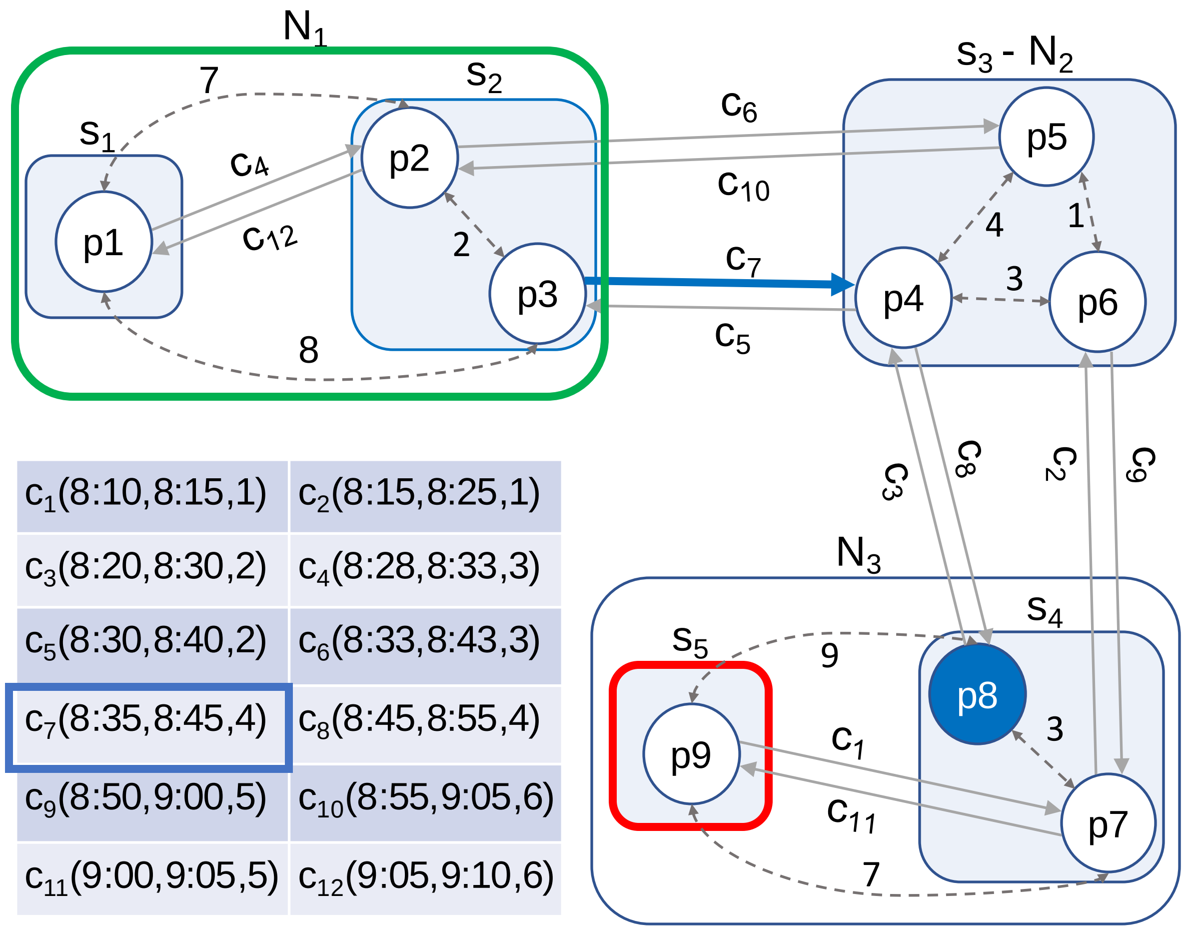
j_q = <???>

Example: q = (s_2, s_5, 8:30)

- (c_7, p_8) is the first reachable transfer connection in FT[N_1][s_5]

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},p_1)	(c_4,p_2)	(c_6,p_5) (c_4,p_5) (c_7,p_4)	(c_7,p_8) (c_4,p_2)	(c_7,p_8) (c_4,p_2)
N_2	(c_5,p_3) (c_{10},p_1)	(c_5,p_3) (c_{10},p_2)	--	(c_8,p_8) (c_9,p_7)	(c_8,p_8) (c_9,p_9)
N_3	(c_3,p_3)	(c_3,p_3)	(c_2,p_6) (c_1,p_6) (c_3,p_4)	(c_1,p_7)	(c_3,p_4) (c_{11},p_9)

c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)



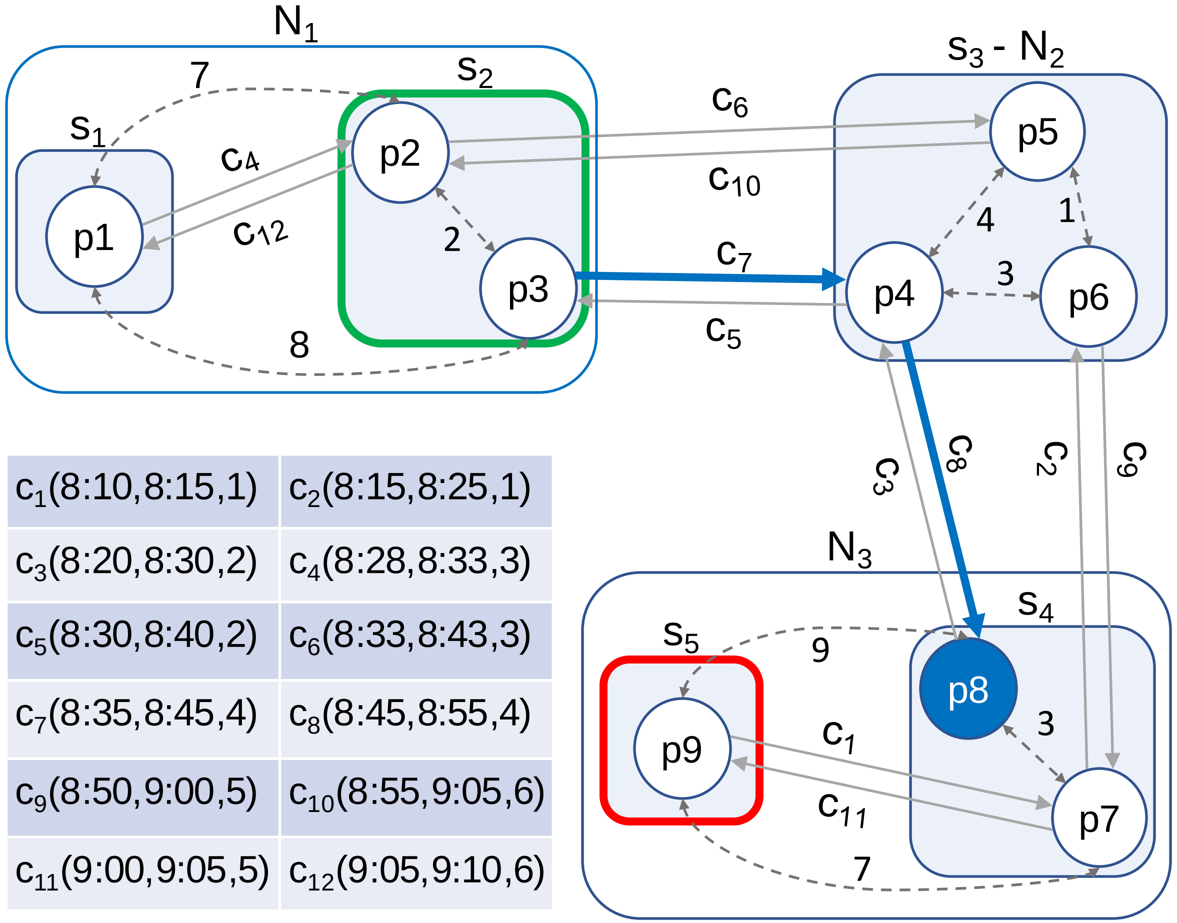
Online Query Phase

Example: q = (s_2, s_5, 8:30)

- c_7 departs from s_2

j_q = <(c_7,p_8), ...>

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},p_1)	(c_4,p_2)	(c_6,p_5) (c_4,p_5) (c_7,p_4)	(c_7,p_8) (c_4,p_2)	(c_7,p_8) (c_4,p_2)
N_2	(c_5,p_3) (c_{10},p_1)	(c_5,p_3) (c_{10},p_2)	--	(c_8,p_8) (c_9,p_7)	(c_8,p_8) (c_9,p_9)
N_3	(c_3,p_3)	(c_3,p_3)	(c_2,p_6) (c_1,p_6) (c_3,p_4)	(c_1,p_7)	(c_3,p_4) (c_{11},p_9)



c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

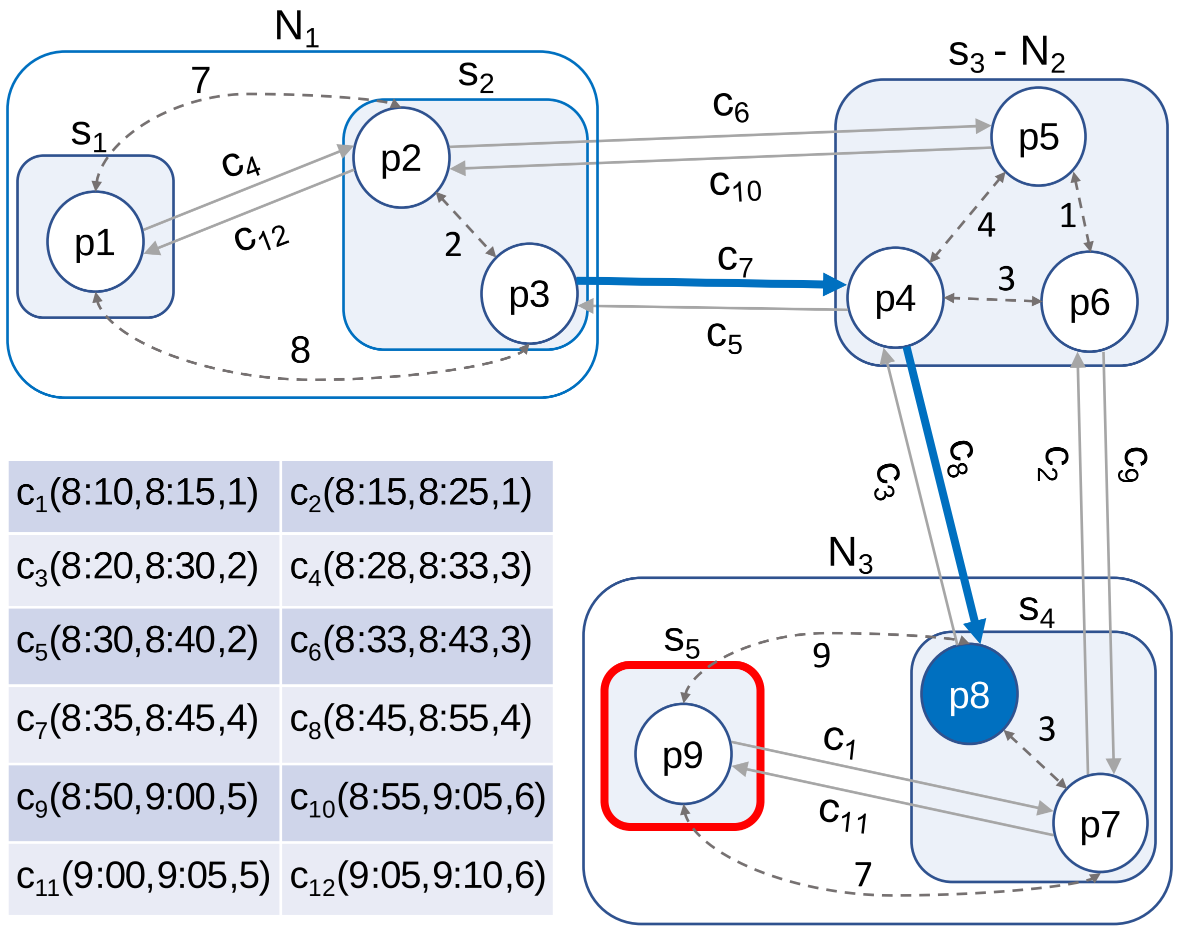
Online Query Phase

Example: $q = (s_2, s_5, 8{:}30)$

- Get off the train at p_8

$j_q = \langle(c_7,p_8), \ldots\rangle$

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},p_1)	(c_4,p_2)	(c_6,p_5) (c_4,p_5) (c_7,p_4)	(c_7,p_8) (c_4,p_2)	(c_7,p_8) (c_4,p_2)
N_2	(c_5,p_3) (c_{10},p_1)	(c_5,p_3) (c_{10},p_2)	--	(c_8,p_8) (c_9,p_7)	(c_8,p_8) (c_9,p_9)
N_3	(c_3,p_3)	(c_3,p_3)	(c_2,p_6) (c_1,p_6) (c_3,p_4)	(c_1,p_7)	(c_3,p_4) (c_{11},p_9)



$c_1(8{:}10,8{:}15,1)$	$c_2(8{:}15,8{:}25,1)$
$c_3(8{:}20,8{:}30,2)$	$c_4(8{:}28,8{:}33,3)$
$c_5(8{:}30,8{:}40,2)$	$c_6(8{:}33,8{:}43,3)$
$c_7(8{:}35,8{:}45,4)$	$c_8(8{:}45,8{:}55,4)$
$c_9(8{:}50,9{:}00,5)$	$c_{10}(8{:}55,9{:}05,6)$
$c_{11}(9{:}00,9{:}05,5)$	$c_{12}(9{:}05,9{:}10,6)$

Online Query Phase

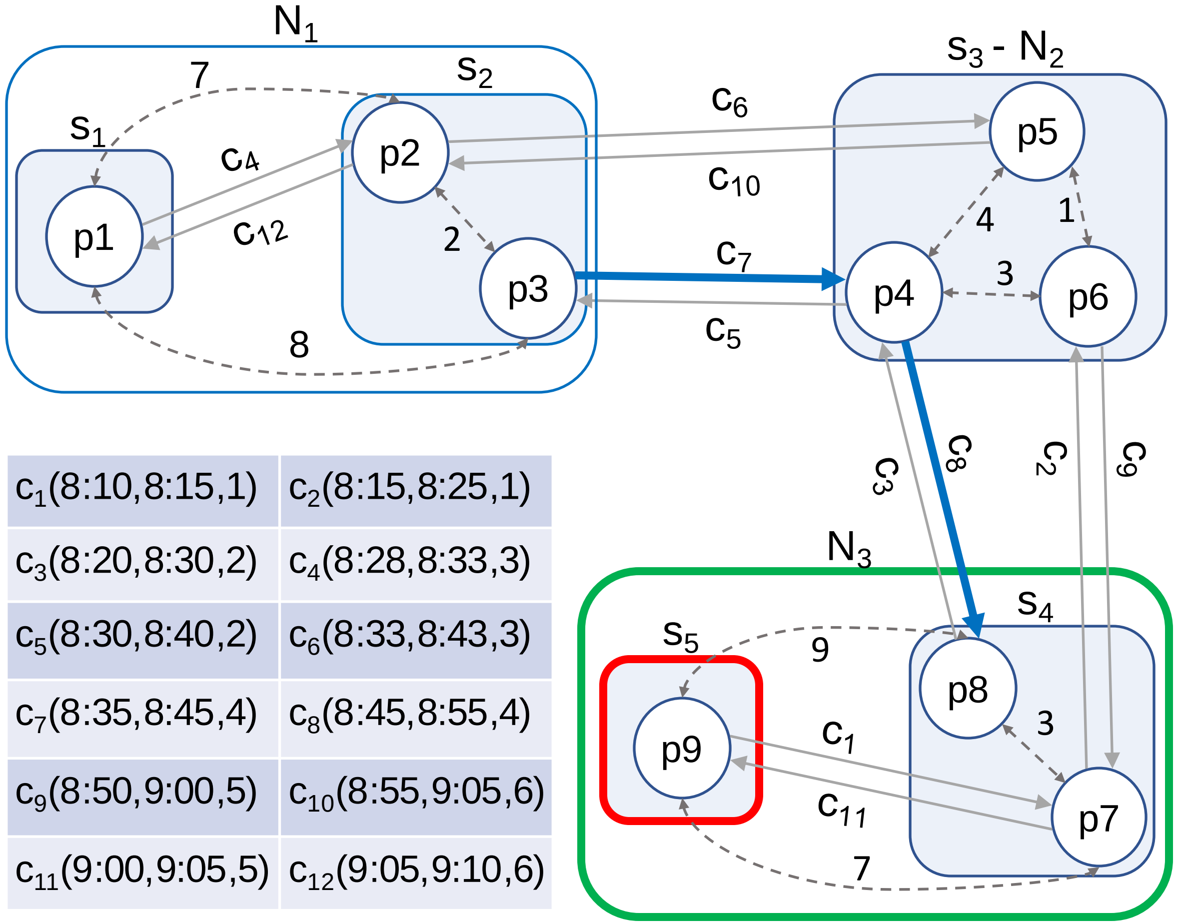
Example: q = (s_2, s_5, 8:30)

- p_8 belongs to N3

j_q = <(c_7,p_8), ...>

FT[N_3][s_5]

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},p_1)	(c_4,p_2)	(c_6,p_5) (c_4,p_5) (c_7,p_4)	(c_7,p_8) (c_4,p_2)	(c_7,p_8) (c_4,p_2)
N_2	(c_5,p_3) (c_{10},p_1)	(c_5,p_3) (c_{10},p_2)	--	(c_8,p_8) (c_9,p_7)	(c_8,p_8) (c_9,p_9)
N_3	(c_3,p_3)	(c_3,p_3)	(c_2,p_6) (c_1,p_6) (c_3,p_4)	(c_1,p_7)	(c_3,p_4) (c_{11},p_9)



c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

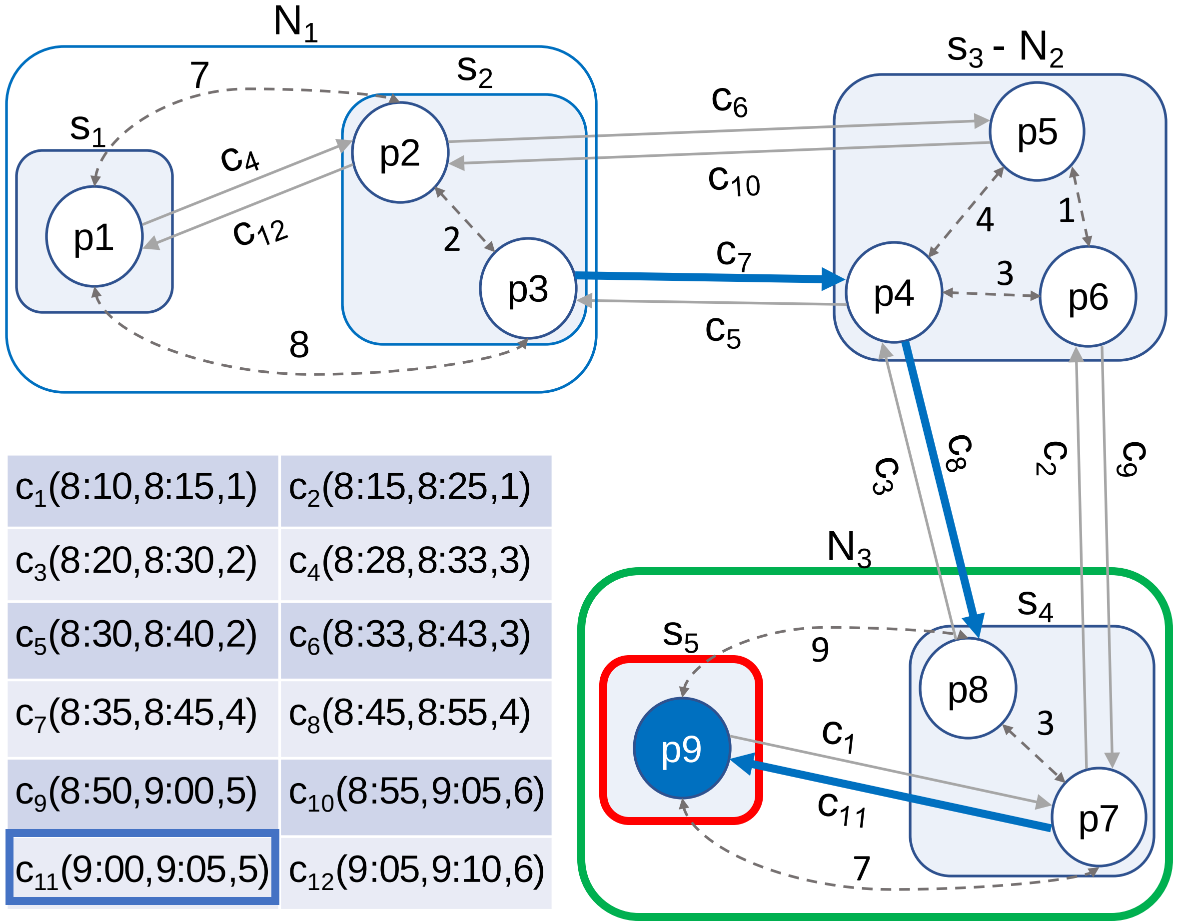
Online Query Phase

Example: q = (s_2, s_5, 8:30)

- (c_{11}, p_9) is the first reachable transfer connection in FT[N_3][s_5]

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},p_1)	(c_4,p_2)	(c_6,p_5) (c_4,p_5) (c_7,p_4)	(c_7,p_8) (c_4,p_2)	(c_7,p_8) (c_4,p_2)
N_2	(c_5,p_3) (c_{10},p_1)	(c_5,p_3) (c_{10},p_2)	--	(c_8,p_8) (c_9,p_7)	(c_8,p_8) (c_9,p_9)
N_3	(c_3,p_3)	(c_3,p_3)	(c_2,p_6) (c_1,p_6) (c_3,p_4)	(c_1,p_7)	(c_3,p_4) (c_{11},p_9)

j_q = <(c_7,p_8), ...>



c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

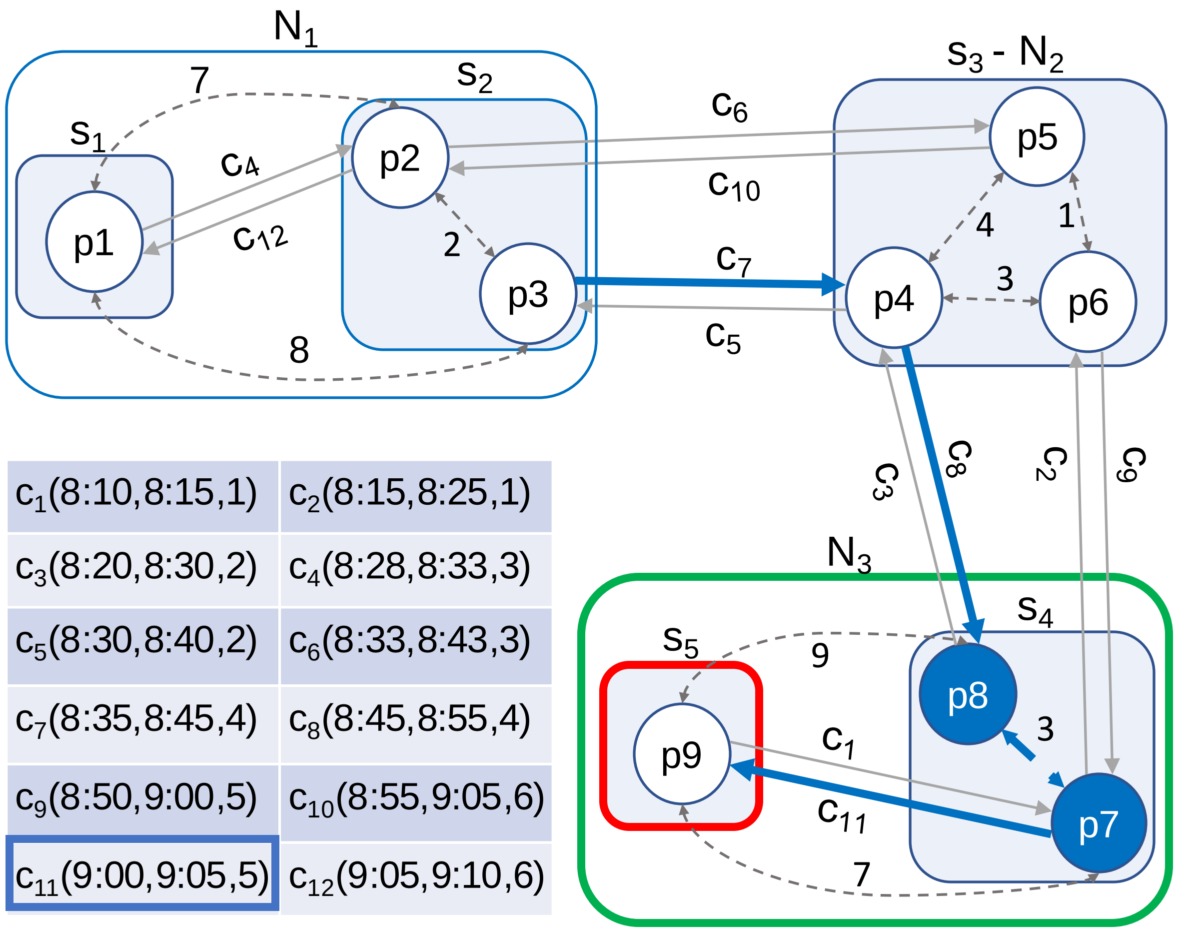
Online Query Phase

j_q = <(c_7,p_8), ...>

Example: q = (s_2, s_5, 8:30)

- c_{11} departs from p_7 and can be reached on time from p_8
- 8:55 + 0:03 ≤ 9:00

O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},p_1)	(c_4,p_2)	(c_6,p_5) (c_4,p_5) (c_7,p_4)	(c_7,p_8) (c_4,p_2)	(c_7,p_8) (c_4,p_2)
N_2	(c_5,p_3) (c_{10},p_1)	(c_5,p_3) (c_{10},p_2)	--	(c_8,p_8) (c_9,p_7)	(c_8,p_8) (c_9,p_9)
N_3	(c_3,p_3)	(c_3,p_3)	(c_2,p_6) (c_1,p_6) (c_3,p_4)	(c_1,p_7)	(c_3,p_4) (c_{11},p_9)



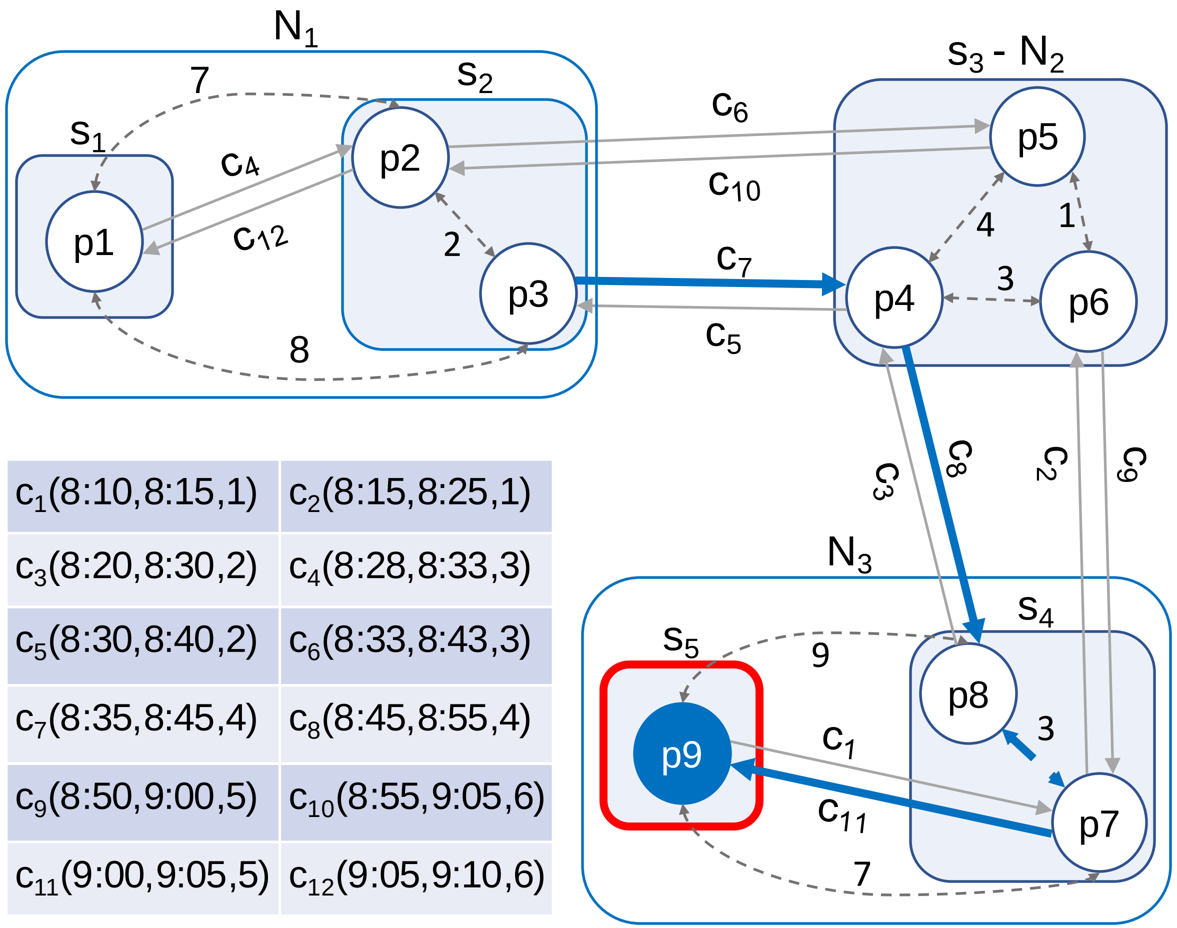
c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

Online Query Phase

Example: q = (s_2, s_5, 8:30)

- s_5 is reached at 9:05
- Algorithm terminates

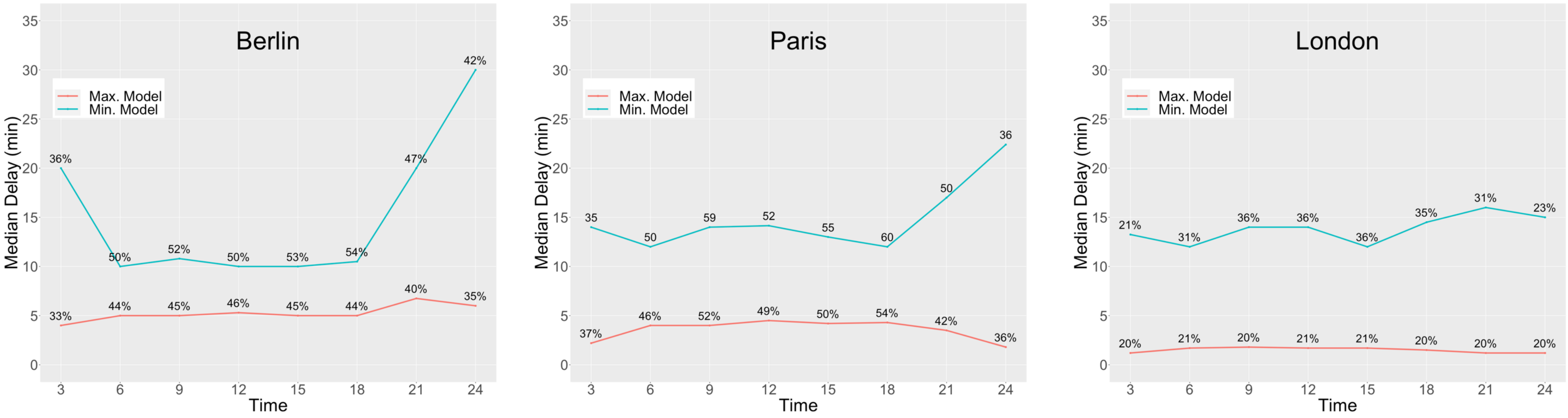
$j_q = <(c_7, p_8), (c_{11}, p_9)>$



c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

Experiments

Experiment 1: Transfer Model Impact



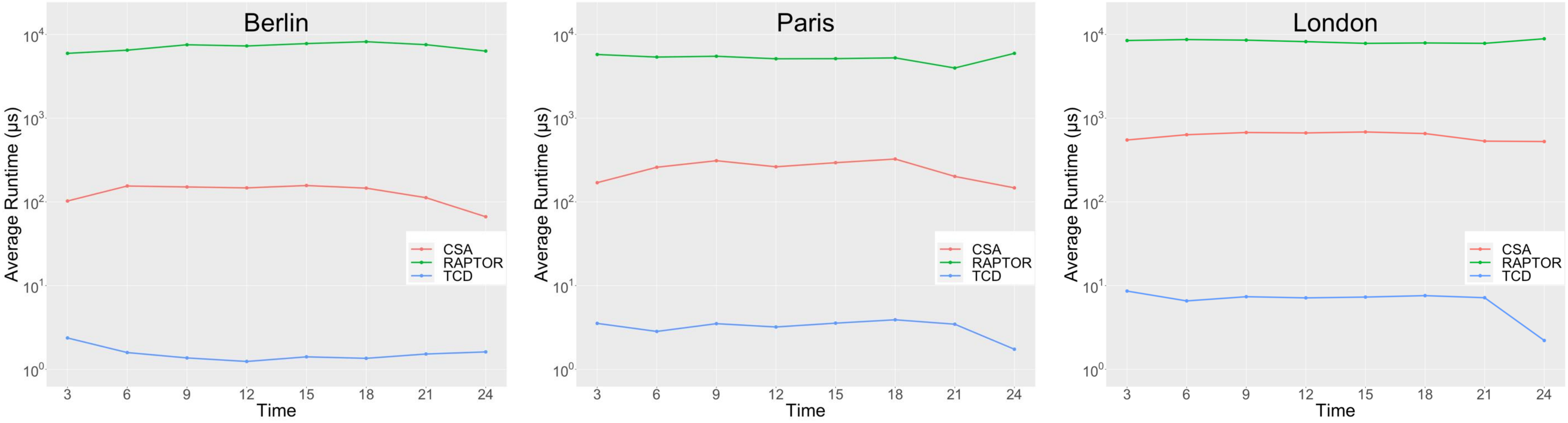
Experiments

Experiment 2: Preprocessing Cost

	Berlin				Paris				London			
TCC	-	✓	-	✓	-	✓	-	✓	-	✓	-	✓
Dominance	-	-	✓	✓	-	-	✓	✓	-	-	✓	✓
Memory (GB)	25.4	18.5	8.1	5.2	88.0	64.4	24.4	16.0	223.8	159.2	53.5	34.2
Time (min)	8	8	8	8	32	32	33	33	111	111	113	113
# labels/OD	362	332	116	68	575	533	160	99	647	547	155	86

Experiments

Experiment 3: Runtime Comparison



Future Work

- Extending TCD to address additional aspects, such as multi-criteria routing problem.
- Enhancing the compression of the database even further.

Thanks for listening!
Questions?

Offline Preprocessing Phase

Challenges in building the FT table

2. Storage (FT table size)
   - Solution: propose optimisations.

   a) Dominance Check (DC).

   b) Transfer Connection Compression (TCC).

Offline Preprocessing Phase

a) Dominance Check

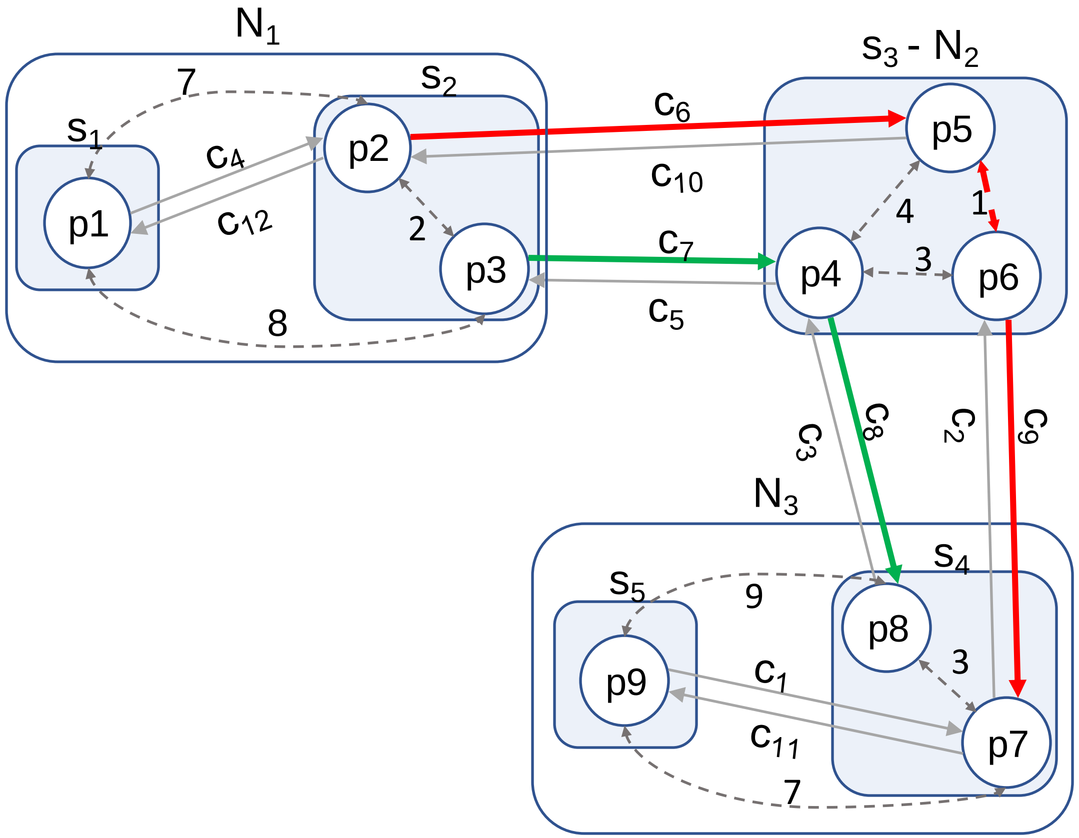
- Remove all-time dominated transfer connections in each cell of FT table.

Offline Preprocessing Phase

a) Dominance Check

- tc_1 (c_7, c_8) vs. tc_3 (c_6, c_6) in FT[N_1][s_4].

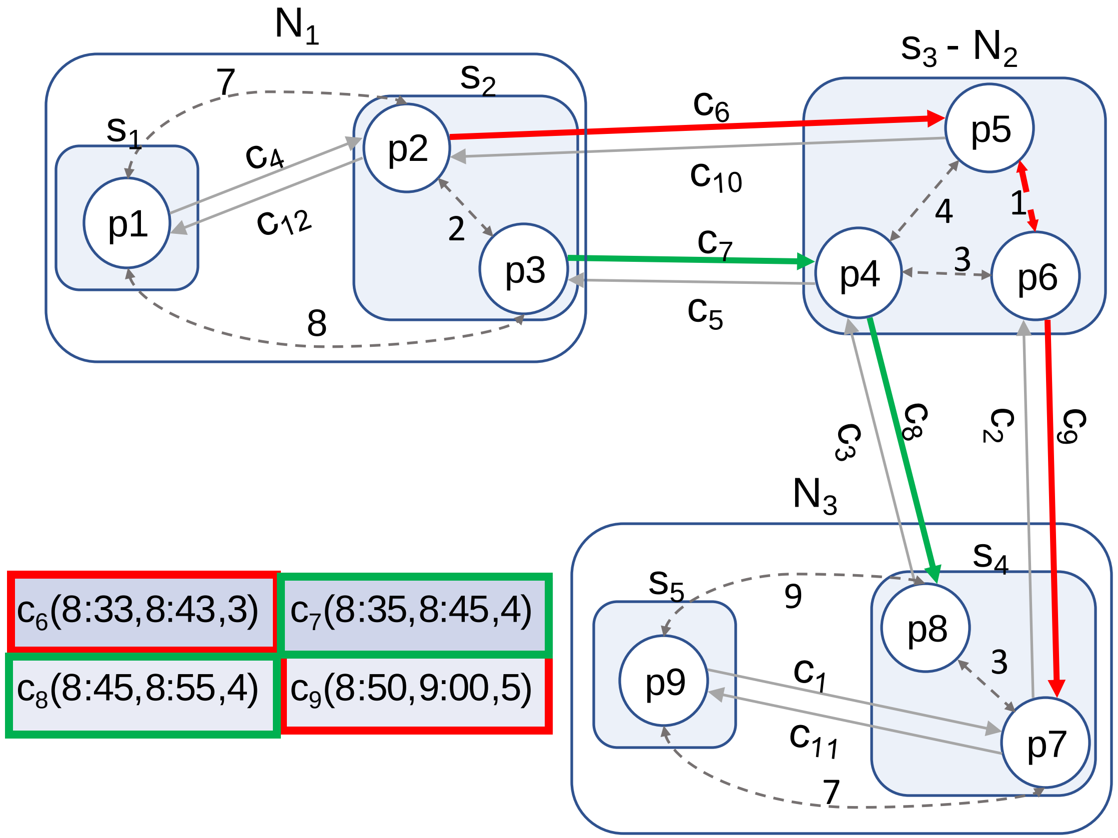
O\D	s_1	s_2	s_3	s_4	s_5
N_1	(c_{12},c_{12}) (c_7,c_7) (c_6,c_6)	(c_4,c_4)	(c_6,c_6) (c_4,c_6) (c_7,c_7)	(c_7,c_8) (c_4,c_4) (c_6,c_6)	(c_7,c_8) (c_4,c_4) (c_6,c_6)
N_2	(c_5,c_5) (c_{10},c_{12})	(c_5,c_5) (c_{10},c_{10})	--	(c_8,c_8) (c_9,c_9)	(c_8,c_8) (c_9,c_{11})
N_3	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_3,c_5) (c_2,c_2) (c_1,c_2)	(c_2,c_2) (c_1,c_2) (c_3,c_3)	(c_1,c_1)	(c_3,c_3) (c_2,c_2) (c_{11},c_{11})



Offline Preprocessing Phase

a) Dominance Check

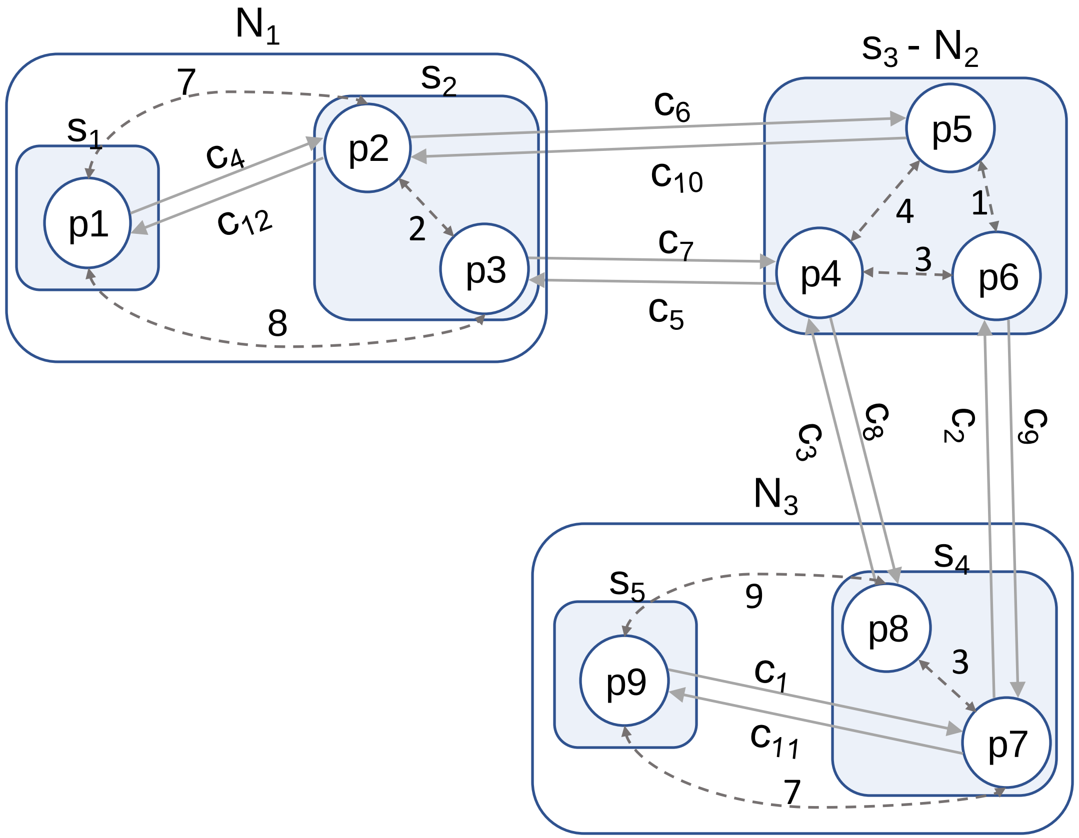
- tc$_1$ (c_7, c_8) vs. tc$_3$ (c_6, c_6) in FT[N$_1$][s$_4$]
- tc$_1$ departs later than tc$_3$ (8:35 vs 8:33)
- tc$_1$ arrives earlier than tc$_3$ (8:55 vs 9:00)
- Difference in departure time is no longer than walking time (2min vs 2min)
- Keep tc$_1$ and remove tc$_3$



Offline Preprocessing Phase

b) Transfer Connection Compression

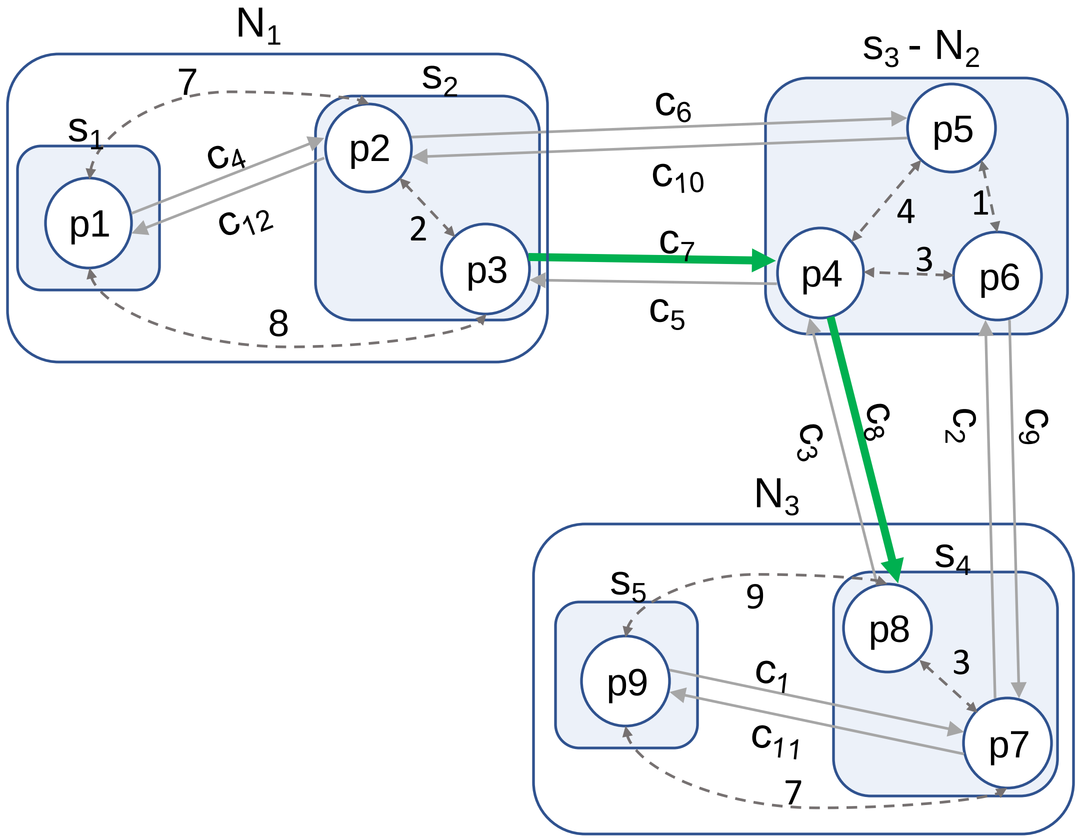
- Observation: optimal paths for a given OD pair often involves a first transfer at the same stop.



Offline Preprocessing Phase

b) Transfer Connection Compression

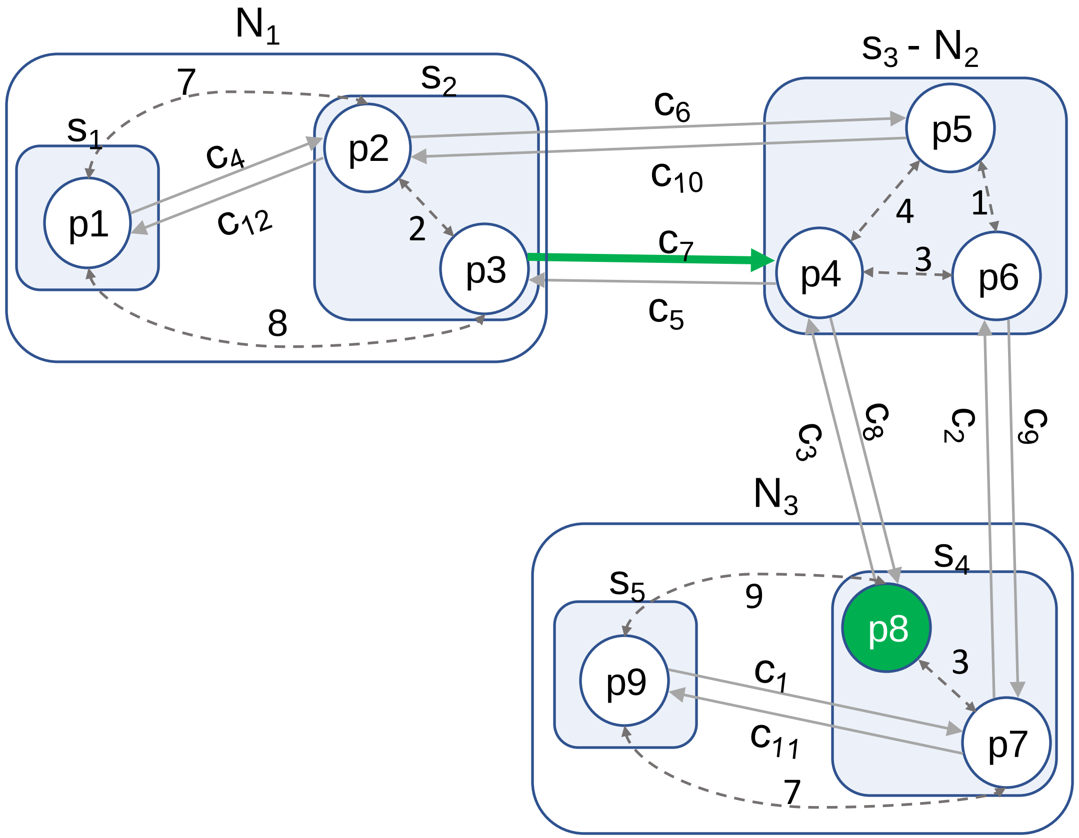
- Observation: optimal paths for a given OD pair often involves a first transfer at the same stop.
- Transfer connection representation from (c_{dep}, c_{arr}) to (c_{dep}, p_{arr}).
- $tc_1 = (c_7, c_8)$



Offline Preprocessing Phase

b) Transfer Connection Compression

- Observation: optimal paths for a given OD pair often involves a first transfer at the same stop.
- Transfer connection representation from (c_{dep}, c_{arr}) to (c_{dep}, p_{arr}).
- $tc_1 = (c_7, c_8) \rightarrow tc1 = (c_7, p_8)$.



Offline Preprocessing Phase

b) Transfer Connection Compression

1. p_{arr} can be stored as a 2-byte integer due to limited number of stops.

Offline Preprocessing Phase

b) Transfer Connection Compression

1. p_{arr} can be stored as a 2-byte integer due to limited number of stops.
2. Consecutive transfer connections sharing same p_{arr} can be merged into a single label.

O\D	s_1
N_1	(c_7, p_8) (c_{14}, p_8) (c_{18}, p_8)

→

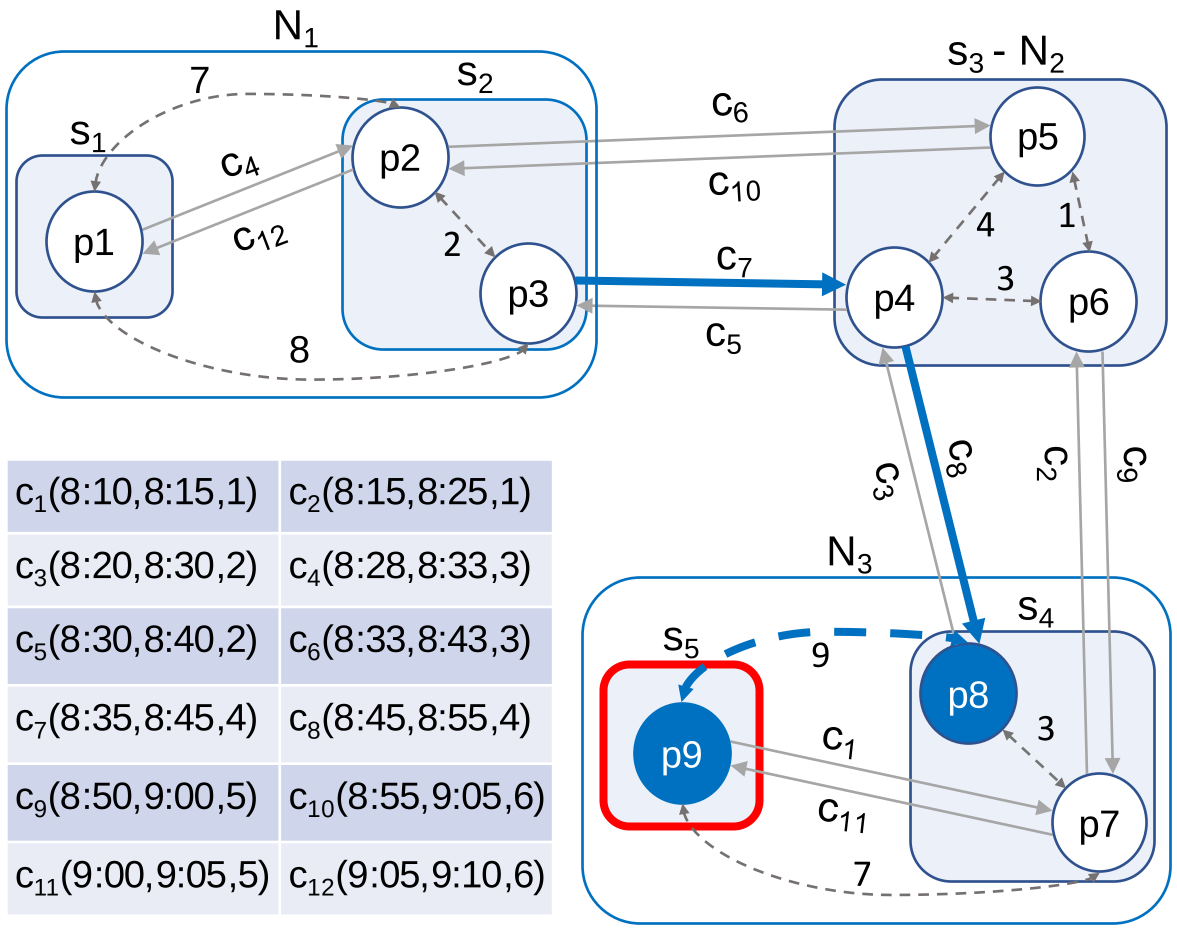
O\D	s_1
N_1	$((c_7, c_{14}, c_{18}), p_8)$

Online Query Phase

Example: q = (s_2, s_5, 8:30)

- s_5 is reachable via footpath from p_8

j_q = <(c_7,p_8), (c_{11}, p_9)>



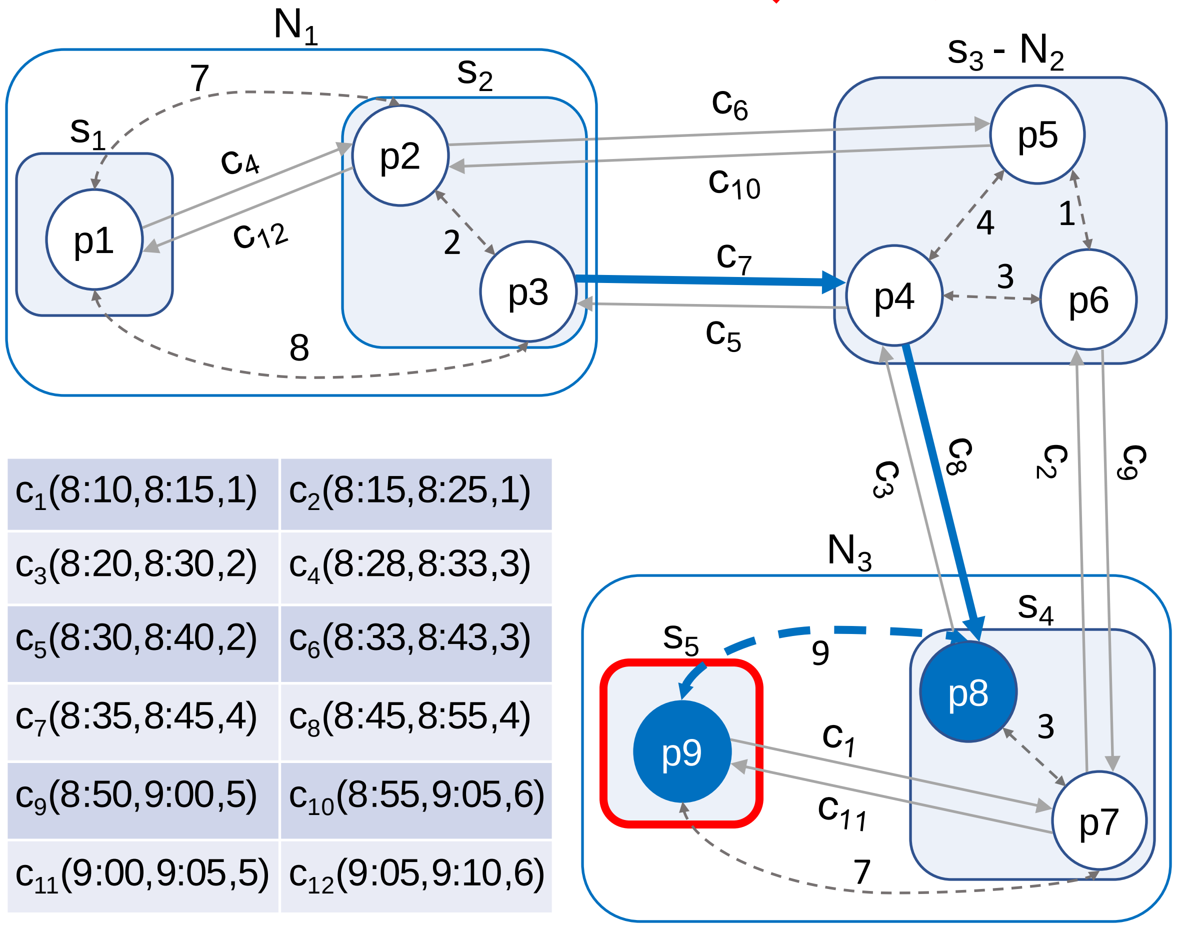
c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

Online Query Phase

$j_q = \langle (c_7,p_8), \cancel{(c_{11}, p_9)} \rangle$

Example: q = (s_2, s_5, 8:30)

- Footpath leads to earlier arrival time at s_5
- 8:55 + 0:09 < 9:05



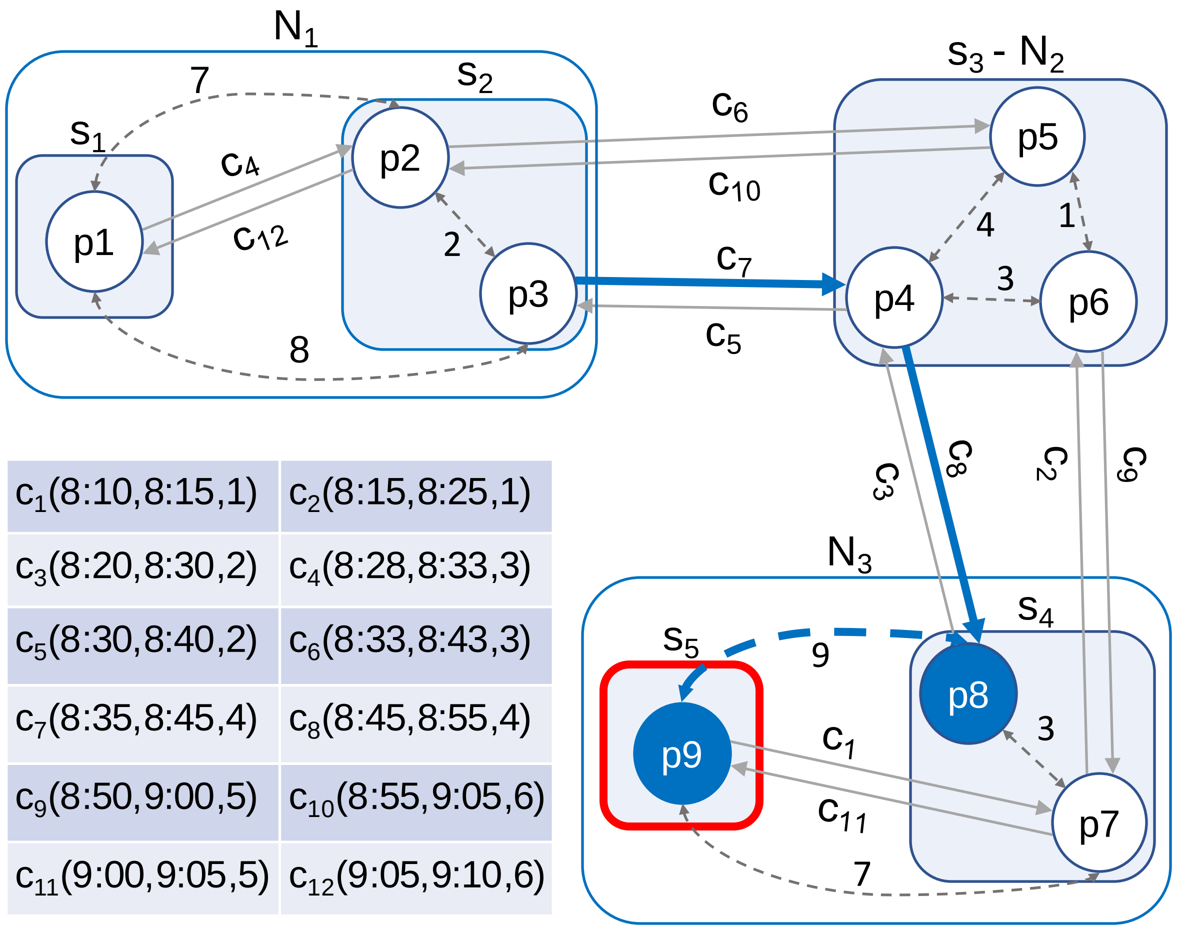
c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

Online Query Phase

Example: q = (s_2, s_5, 8:30)

- Footpath leads to earlier arrival time at s_5
- 8:55 + 0:09 < 9:05

j_q = <(c_7,p_8), f(p_8, p_9)>



c_1(8:10,8:15,1)	c_2(8:15,8:25,1)
c_3(8:20,8:30,2)	c_4(8:28,8:33,3)
c_5(8:30,8:40,2)	c_6(8:33,8:43,3)
c_7(8:35,8:45,4)	c_8(8:45,8:55,4)
c_9(8:50,9:00,5)	c_{10}(8:55,9:05,6)
c_{11}(9:00,9:05,5)	c_{12}(9:05,9:10,6)

Experiments

Experiments Setup

- 3 metropolitan networks
- 5,000 random station pairs
- 8 fixed departure times across the day
- 40,000 queries in total for each network
- Transfer modelled using exact transfer costs

Dataset	Stations	Stops	Connections	Trips	Footpaths
Berlin	3,365	8,359	1,006,375	42,518	45,553
Paris	6,263	12,047	1,836,496	78,757	148,444
London	9,798	14,516	3,088,661	87,898	162,543

Conclusion

- Proposed an efficient solution for earliest arrival problem, integrating exact transfer costs and utilising well-structured transfer database.
- Demonstrated the significance of employing exact transfer costs compared to uniform costs.