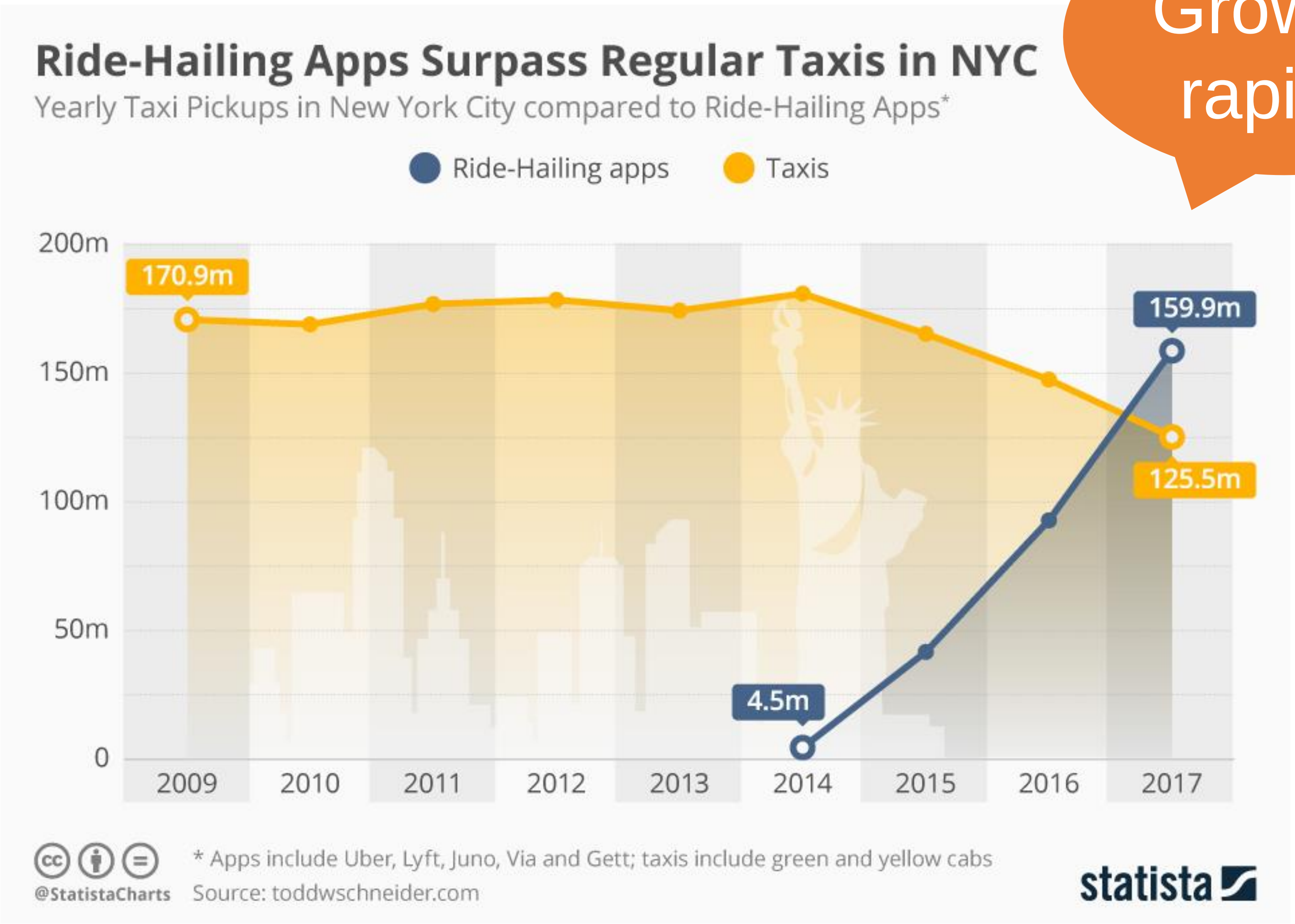
Highly Efficient and Scalable Multi-hop Ride-sharing

Yixin Xu, Lars Kulik, Renata Borovica-Gajic, Abdullah Aldwyish, Jianzhong Qi

The University of Melbourne



Growing rapidly

DiDi

UBER

lyft

Existing research focus

- Speed-up the matching time
  - Large scale real-time ridesharing with service guarantee on road networks, PVLDB, 2014
  - A unified approach to route planning for shared mobility, PVLDB, 2018
  - GeoPrune: Effciently Matching Trips in Ride-sharing Through Geometric Properties, SSDBM, 2020
  - …

- Improve the matching quality
  - Price-aware real-time ride-sharing at scale: an auction-based approach, SIGSPATIAL, 2016
  - Utility-aware ridesharing on road networks, SIGMOD, 2017
  - Mobility-aware dynamic taxi ridesharing, ICDE, 2020
  - …

Existing research focus

- Speed-up the matching time
  - Large scale real-time ridesharing with service guarantee on road networks, PVLDB, 2014
  - A unified approach to route planning for shared mobility, PVLDB, 2018

Existing research focus: single-hop ride-sharing

  - …

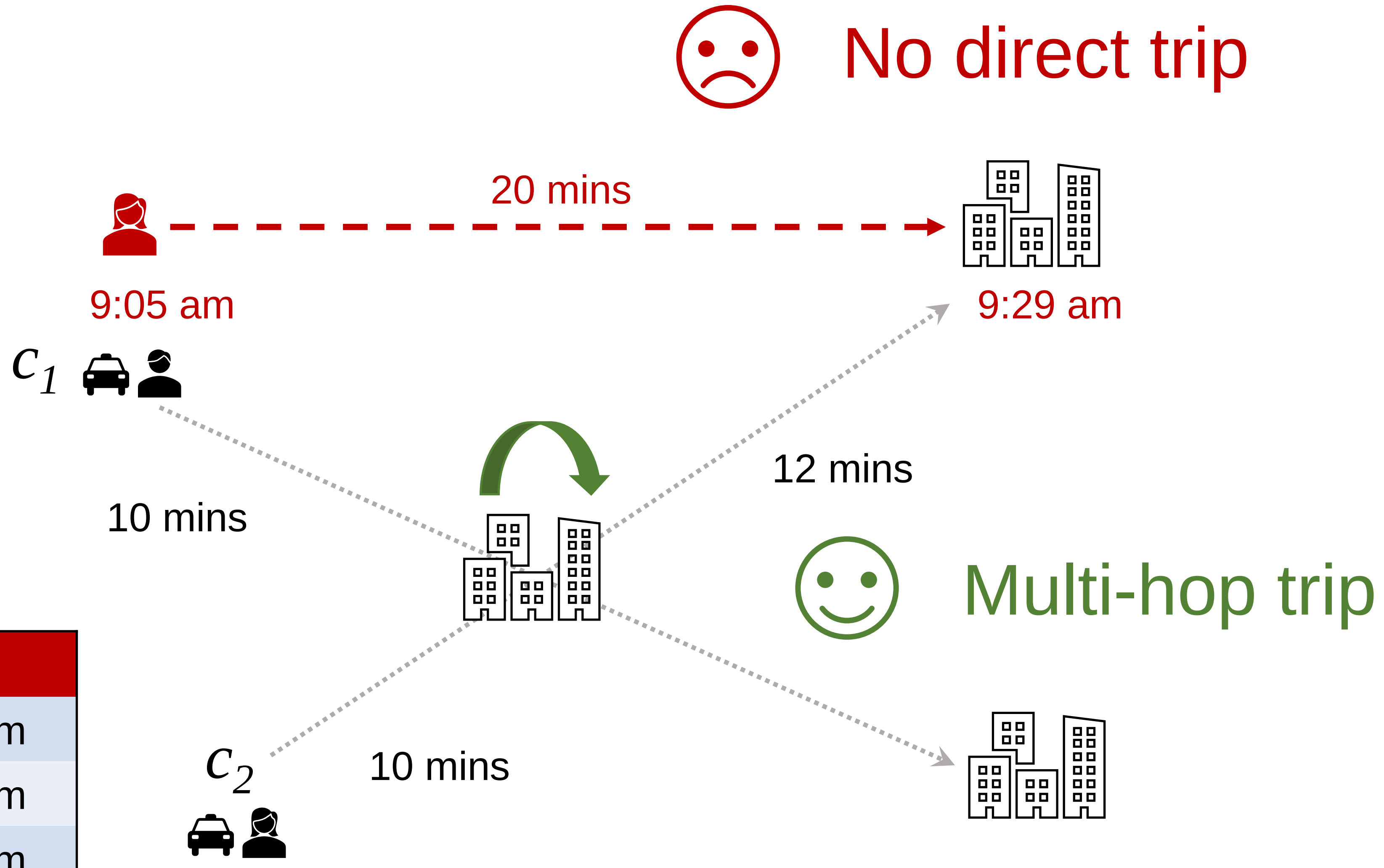
- Improve the matching quality

The possibility of transfers is not considered in previous works

  - Utility-aware ridesharing on road networks, SIGMOD, 2017
  - Mobility-aware dynamic taxi ridesharing, ICDE, 2020
  - …

Multi-hop ride-sharing: example

9:00 am

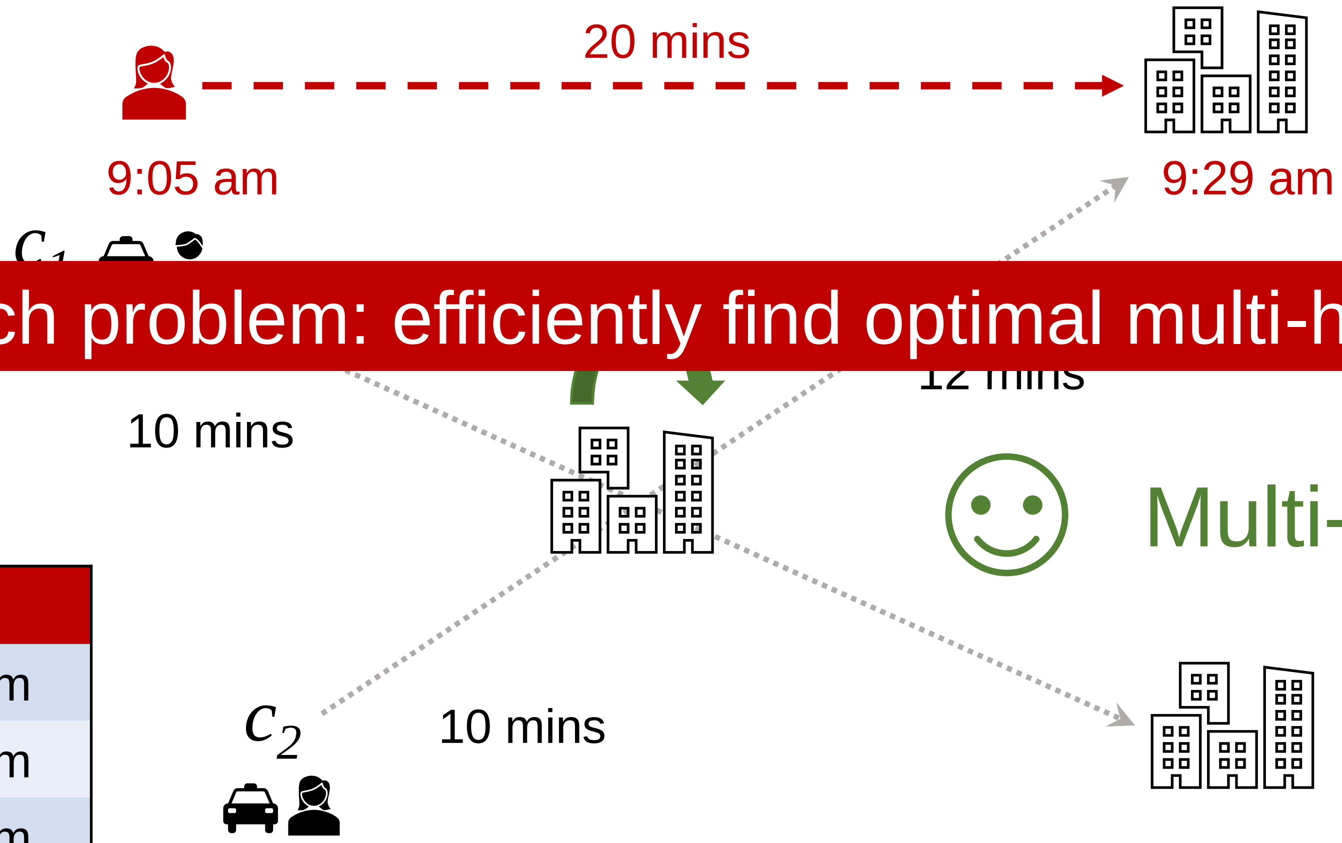


Alice	
issue time	9:00 am
latest pick-up	9:05 am
latest drop-off	9:29 am

Multi-hop ride-sharing: example

9:00 am

No direct trip



Research problem: efficiently find optimal multi-hop trips

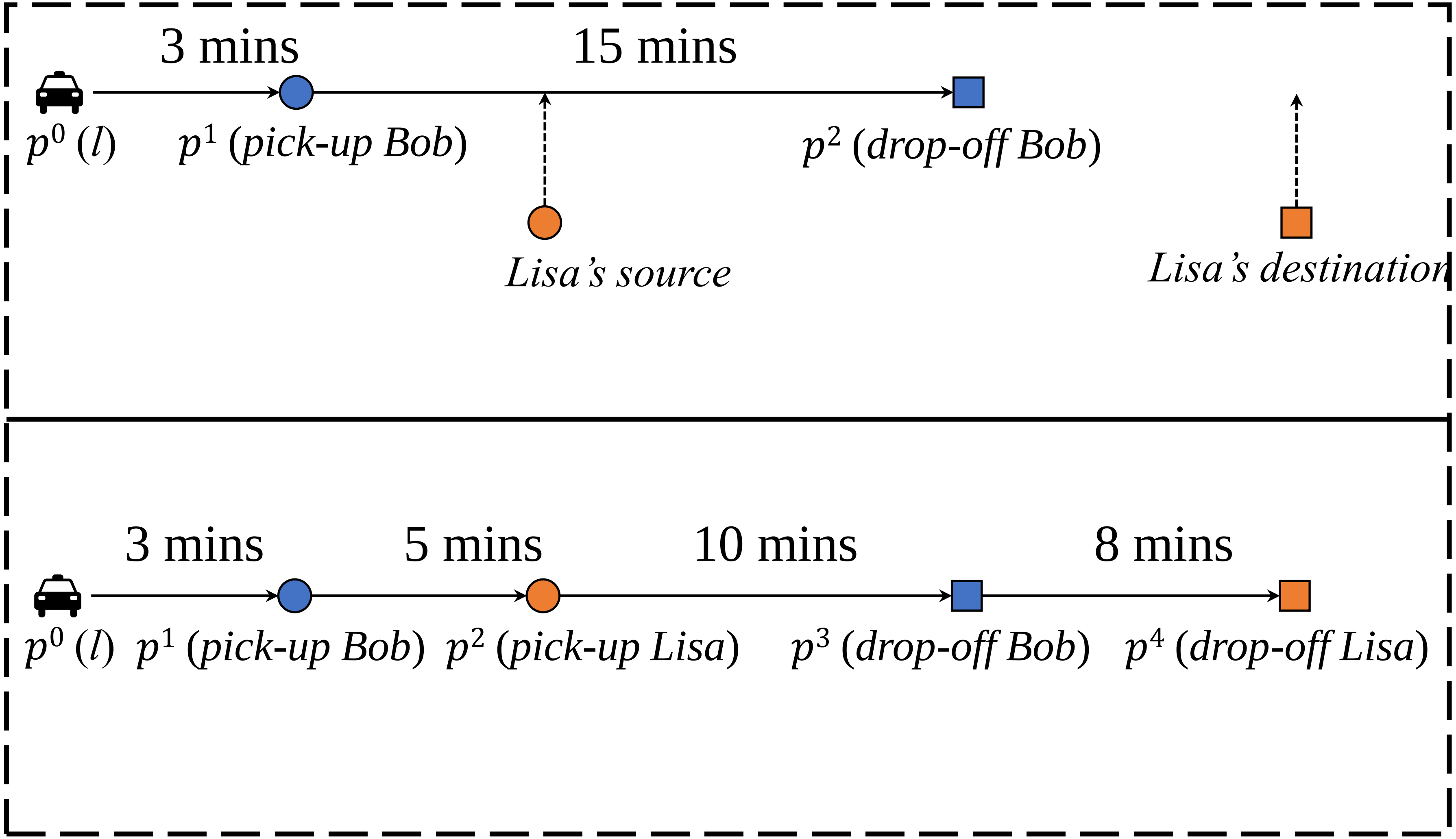
Multi-hop trip

Alice	
issue time	9:00 am
latest pick-up	9:05 am
latest drop-off	9:29 am

Assumption: insertion

Bob	
issue time	9:00 am
latest pick-up	9:05 am
latest drop-off	9:23 am

Lisa	
issue time	9:07 am
latest pick-up	9:12 am
latest drop-off	9:30 am



Multi-hop match:

- $r_n.m = < c_1, c_2, \phi, \Gamma >$
  - c_1: the first vehicle
  - c_2: the second vehicle
  - ϕ: the transfer point
  - Γ: the insert positions
    - $\Gamma(s)$: source to the first vehicle’s schedule
    - $\Gamma(\phi_1)$ transfer point to the first vehicle’s schedule
    - $\Gamma(\phi_2)$: transfer point to the second vehicle’s schedule
    - $\Gamma(e)$: destination to the second vehicle’s schedule

High computationl complexity

- $r_n.m = < c_1, c_2, \phi, \Gamma >$

vehicles (|C|)   # transfer points (|P|)   (Possible insertion positions)^4 ($|S|$)^4

- Possible solutions: $|C| * |C| * |P| * |S|^4$

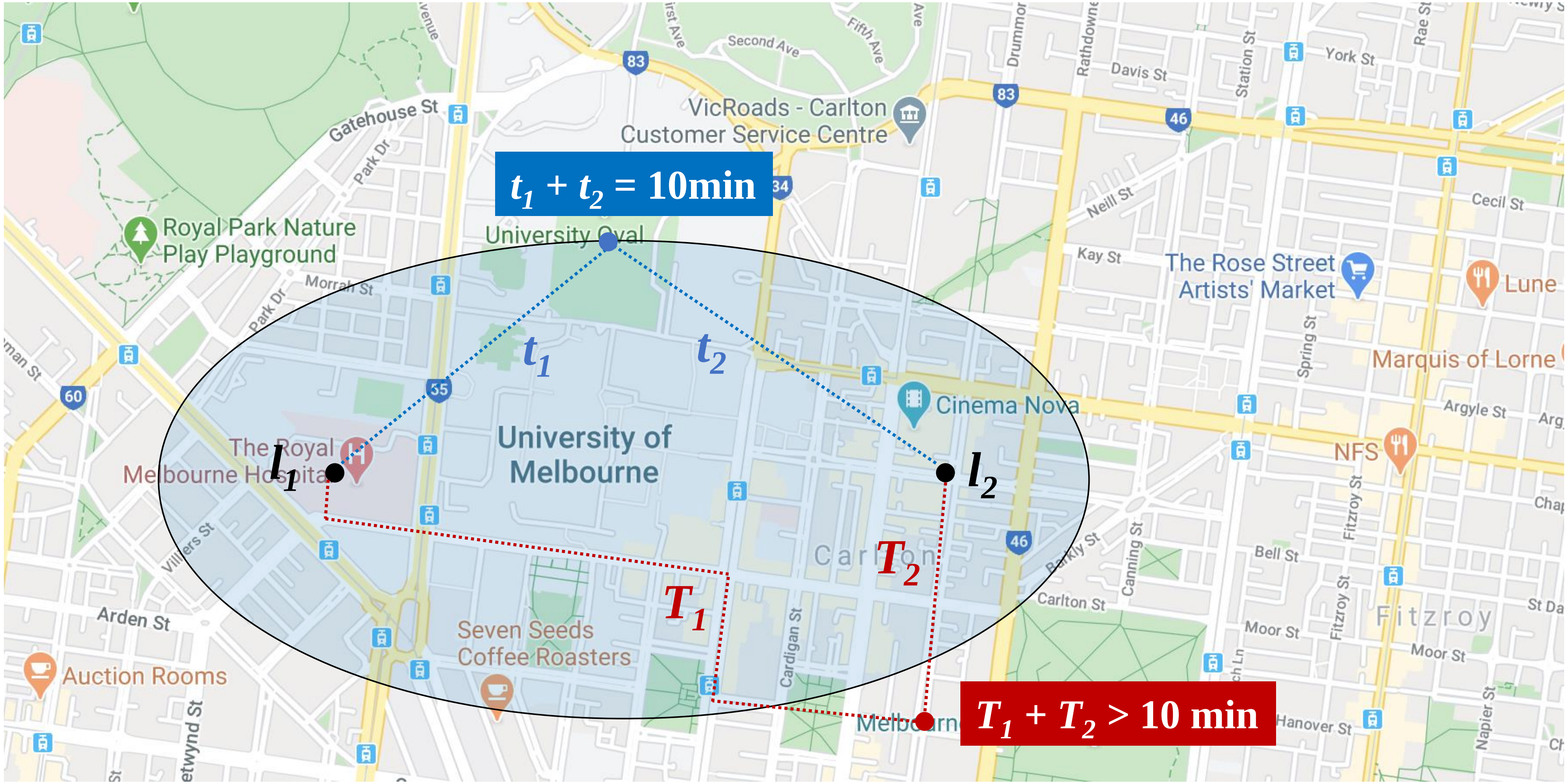
Two algorithms are proposed

- Station-first algorithm
- Vehicle-first algorithm

Station-first algorithm

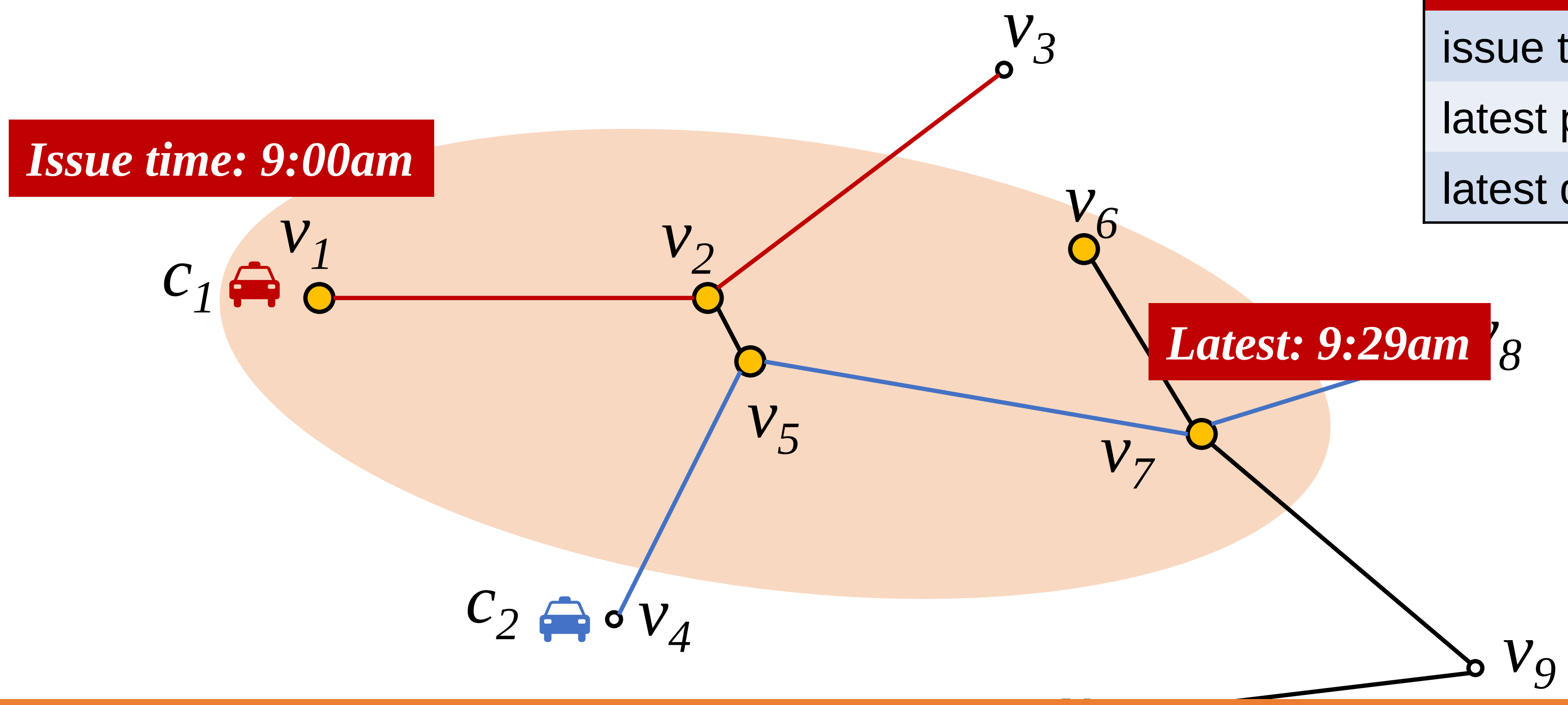
- $r_n.m = < c_1, c_2, \phi, \Gamma >$
  - Filter out possible transfer point ϕ
  - For each possible ϕ
    - Search for c_1, c_2, and Γ

Bounding ellipse



Station-first algorithm : prune ϕ

Alice	
issue time	9:00 am
latest pick-up	9:05 am
latest drop-off	9:29 am



Pruning strategy: the transfer point must be within a detour ellipse

Station-first algorithm: determine c_1, c_2, and Γ

- A transfer point ϕ splits the trip into two itineraries
  - Itinerary 1: $s - \phi$
  - Itinerary 2: $\phi - d$

- State-of-the-art single-hop algorithm – GeoPrune[1]
  - GeoPrune (s, ϕ)
  - GeoPrune (ϕ, d)

Preferable when possible transfer points are sparse

1. Yixin Xu, Jianzhong Qi, Renata Borovica-Gajic, Lars Kulik. GeoPrune: Effciently Matching Trips in Ride-sharing Through Geometric Properties, International Conference on Scientic and Statistical Database Management (SSDBM), 2020.

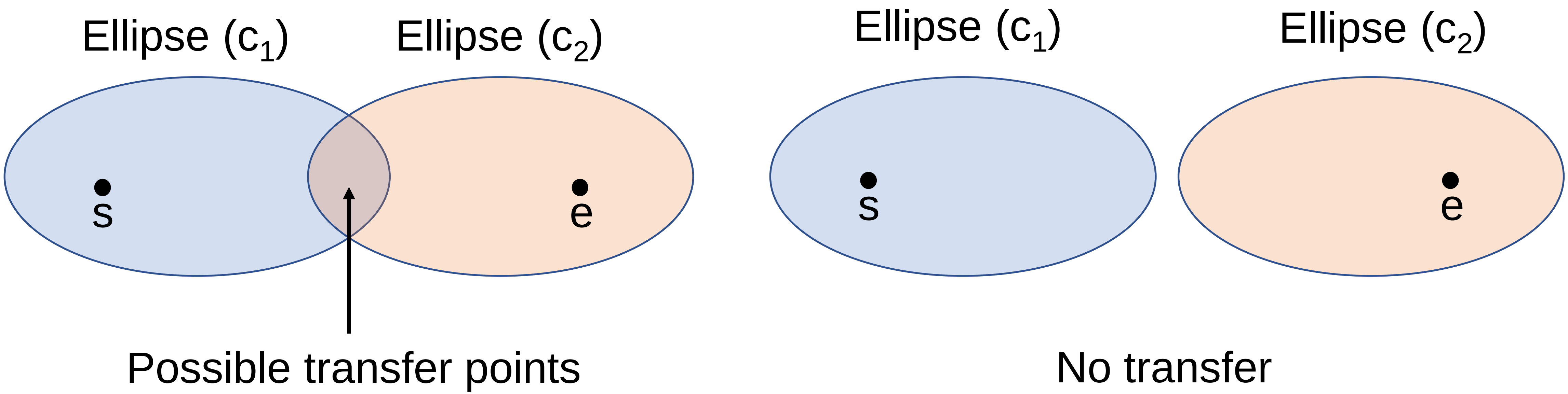
Vehicle-first algorithm

- $r_n.m = < c_1, c_2, \phi, \Gamma >$
  - first determine c_1, c_2, and Γ
  - For each $< c_1, c_2, \phi, \Gamma >$
    - Search for the optimal transfer point ϕ

Preferable when possible transfer points are ~~sparse~~ **dense**

Vehicle-first algorithm: determine c_1, c_2, and Γ

- Possible transfer
  - The detour ellipses of two vehicles must overlap



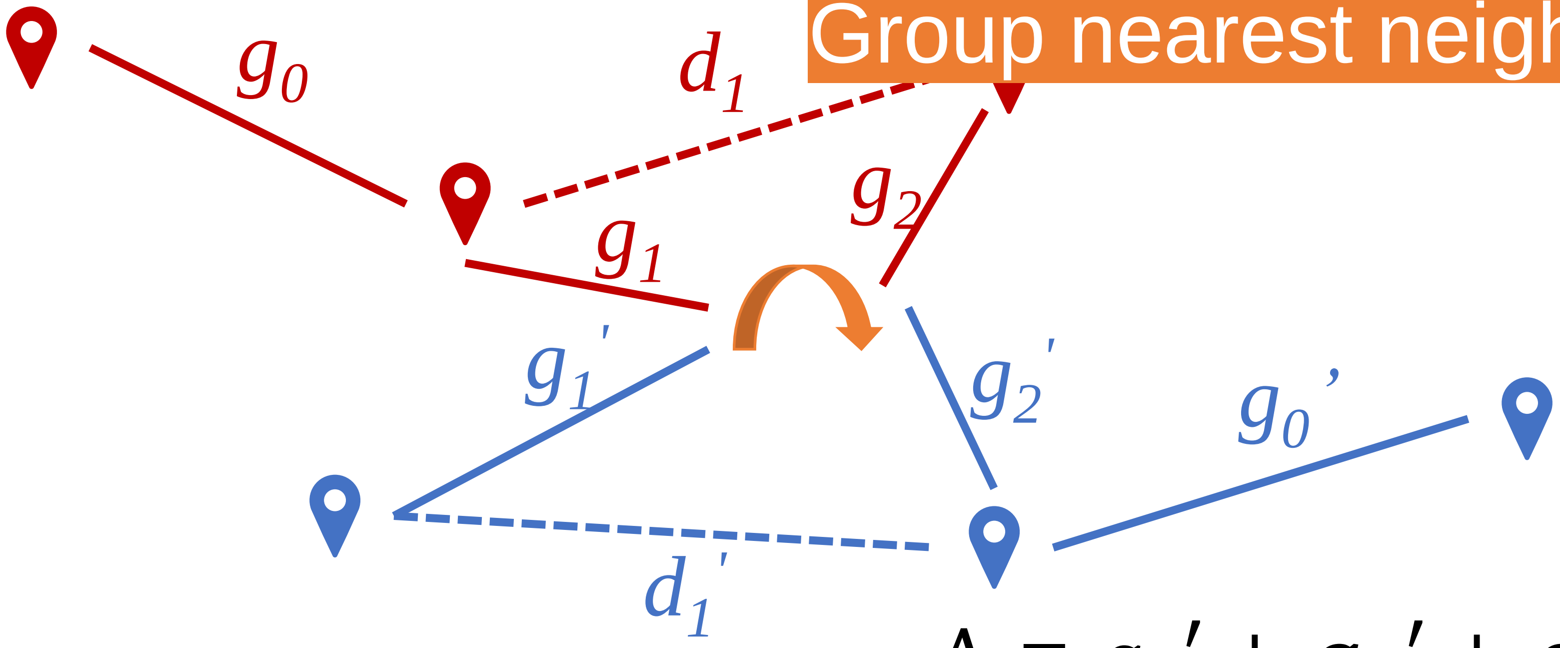
Vehicle-first algorithm: determine ϕ

- Key observation: the optimal transfer point depends on only several fixed locations

Minimize $g_1 + g_2 + g_1' + g_2'$

Group nearest neighbor query

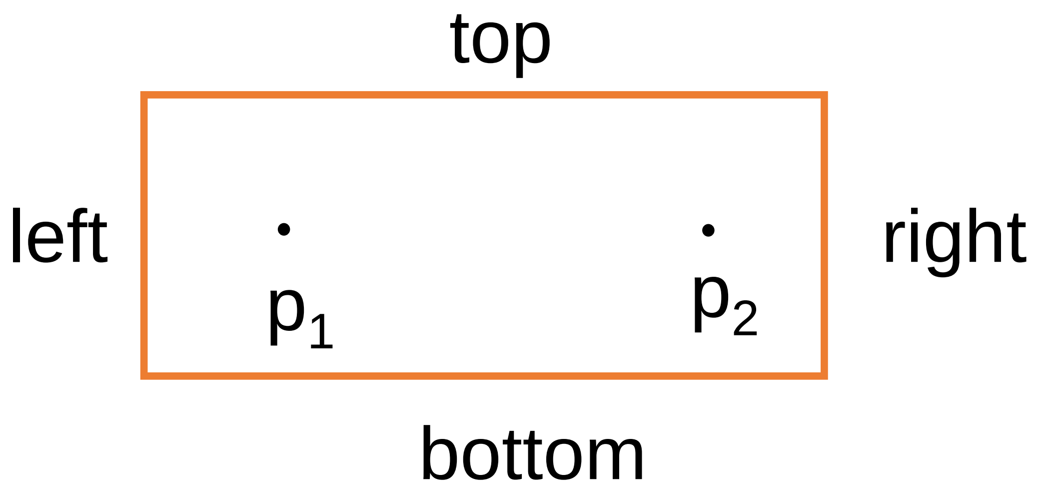
$$\Delta_1 = g_0 + \boldsymbol{g_1} + \boldsymbol{g_2} - d_1$$



$$\Delta_2 = g_0' + \boldsymbol{g_1}' + \boldsymbol{g_2}' - d_1'$$

Speed-up

- Learn the reachable area (ellipse)
- Only check the first few transfer points

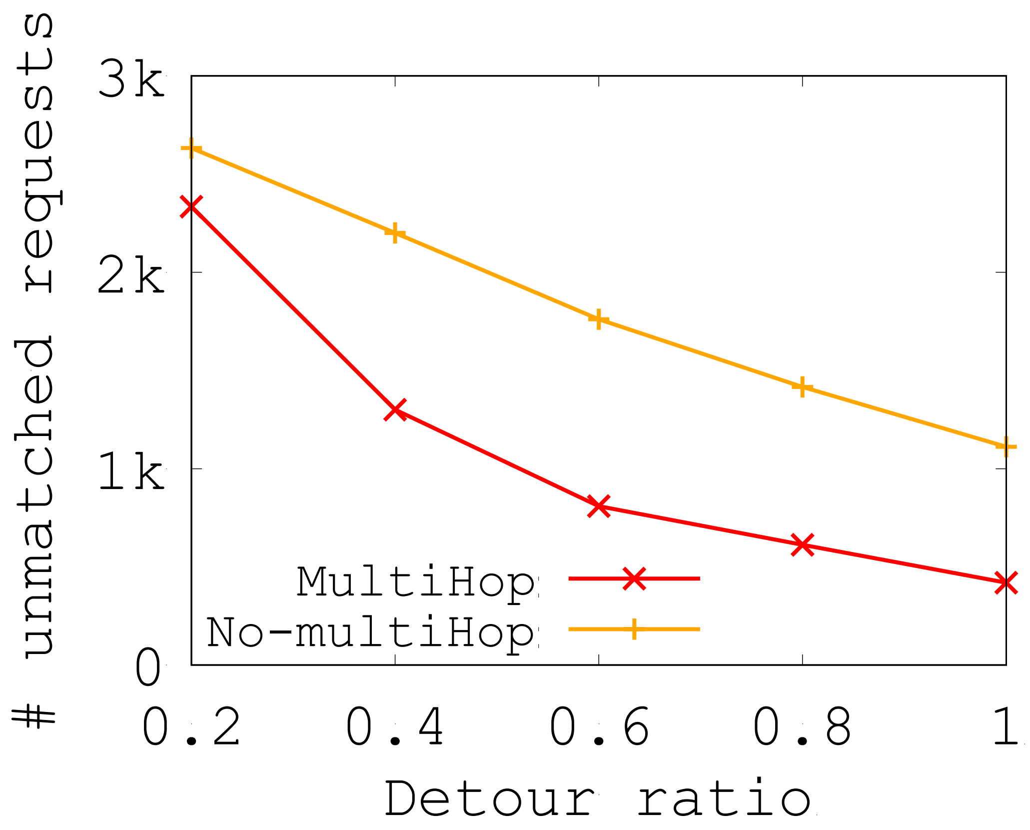


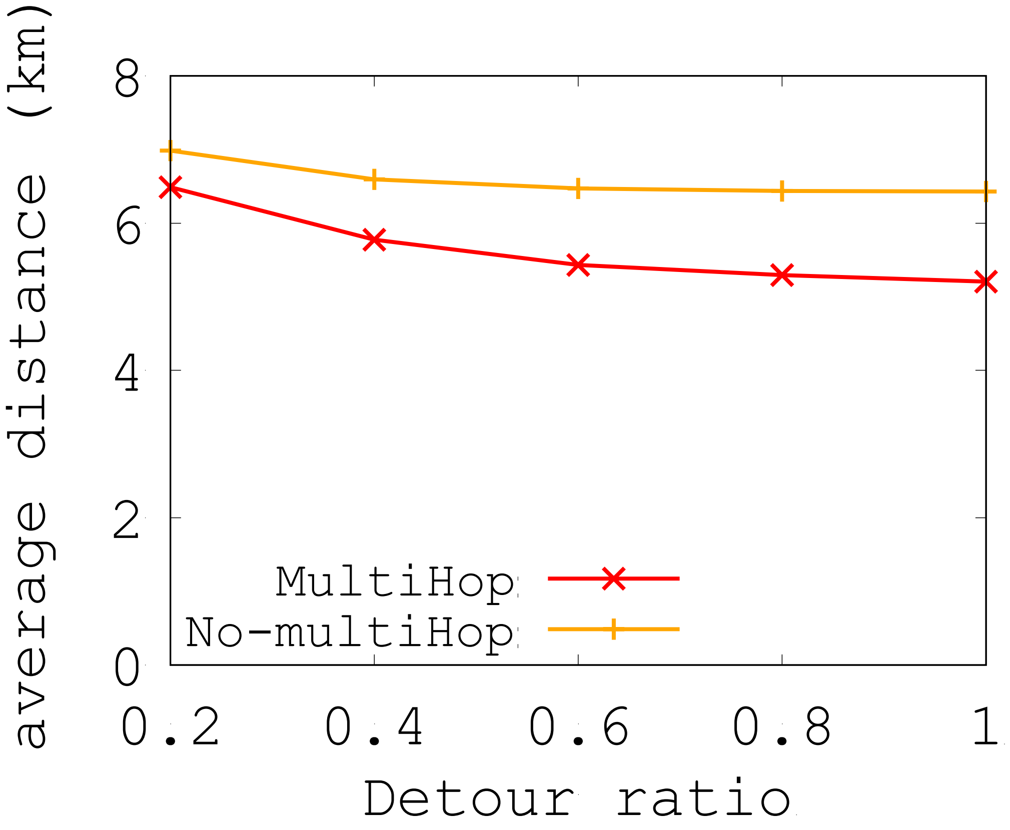
Experimental results

- **Real-word datasets**
  - Chengdu (CD): 166,296 nodes, 405,460 edges
- **Default settings**
  - # requests: 10000
  - # vehicles: 4096
  - Waiting time: 4min
  - Detour raio: 0.2
  - Minimize the total travel distance

Benefits of multi-hop ride-sharing

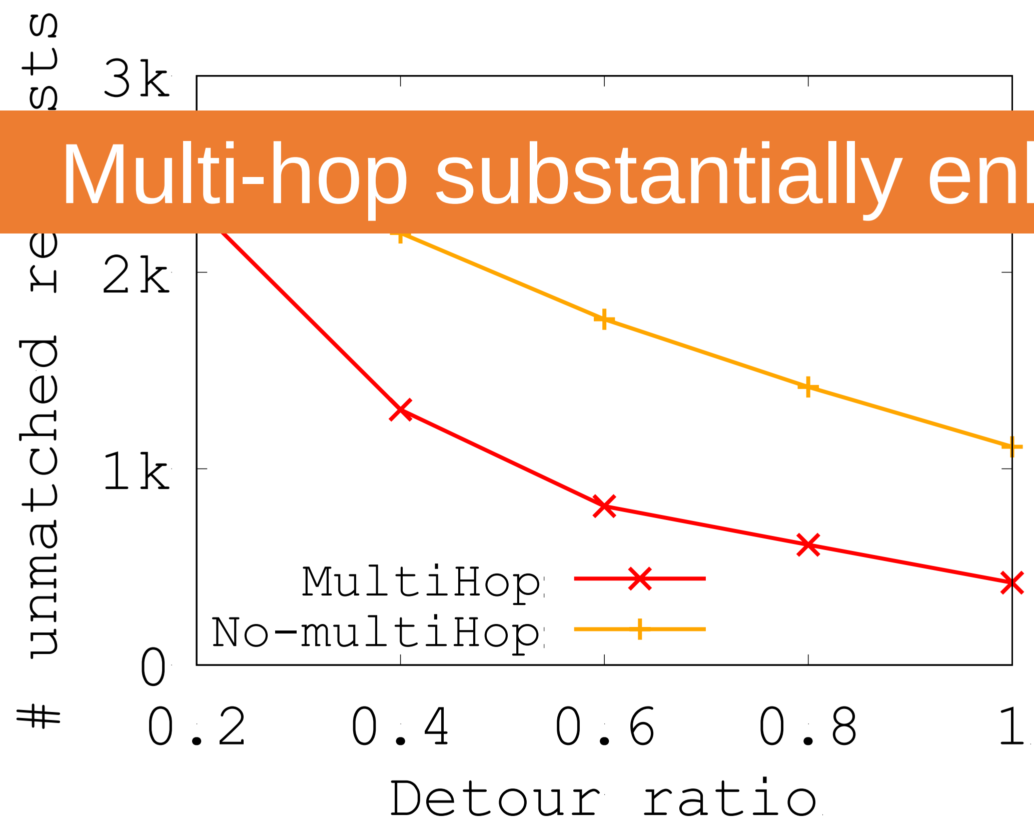
- Effect of the detour ratio

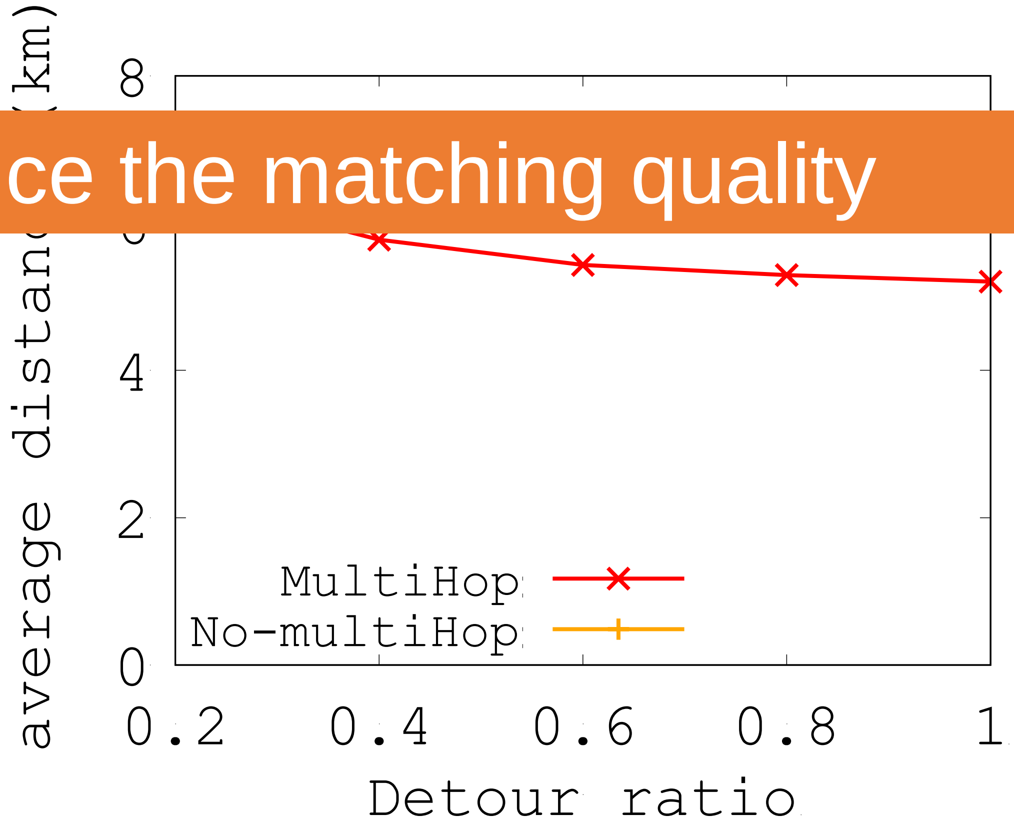




Benefits of multi-hop ride-sharing

- Effect of the detour ratio

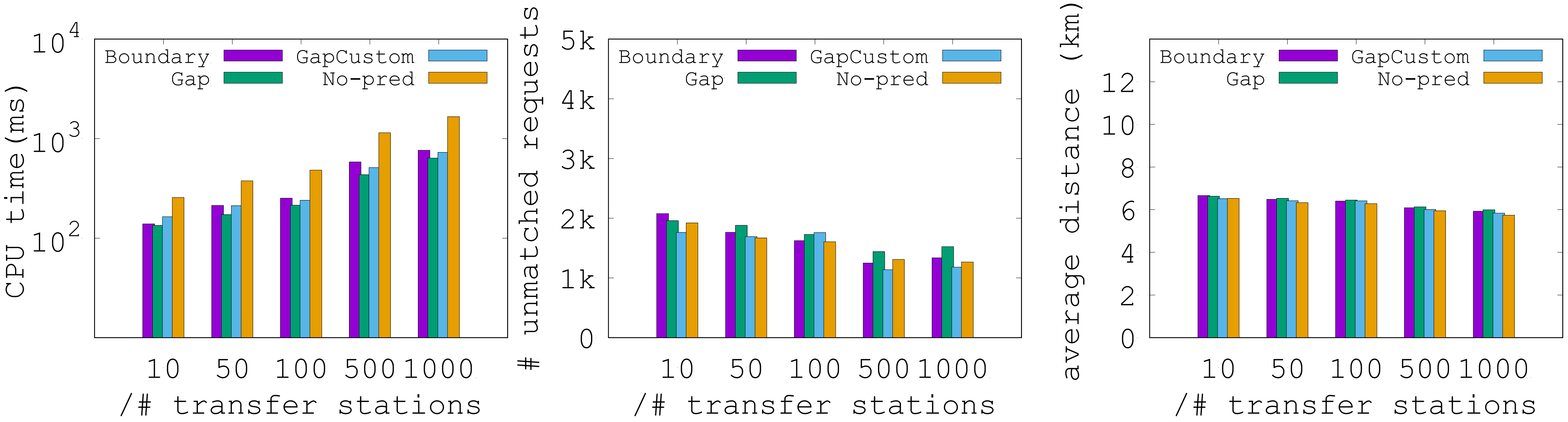




Multi-hop substantially enhance the matching quality

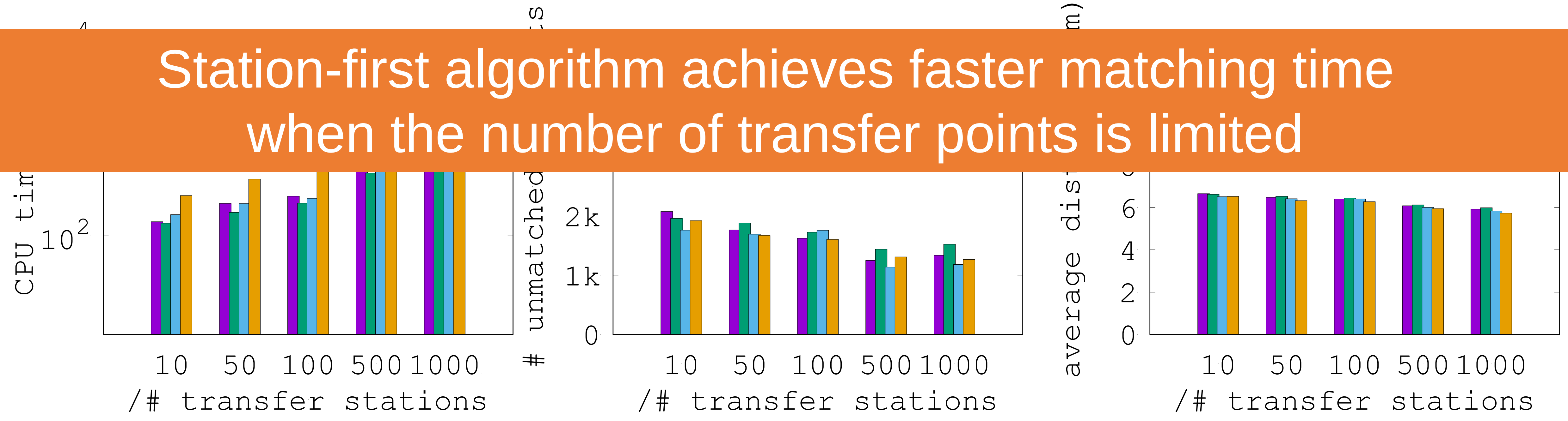
Algorithm performance

➢ Effect of the # transfer points



Algorithm performance

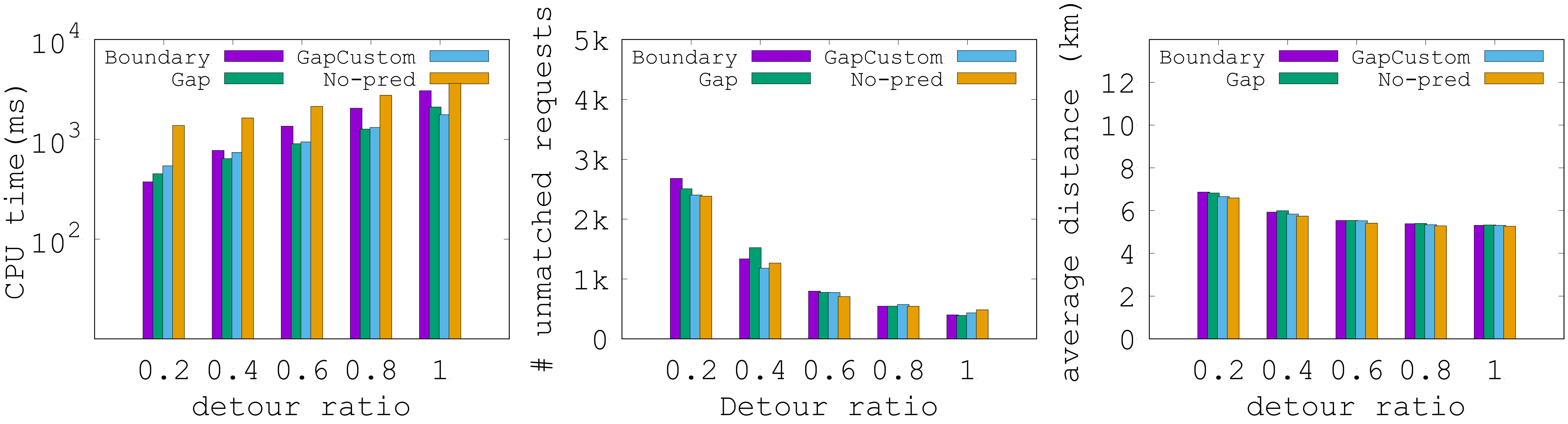
- Effect of the # transfer points



Station-first algorithm achieves faster matching time when the number of transfer points is limited

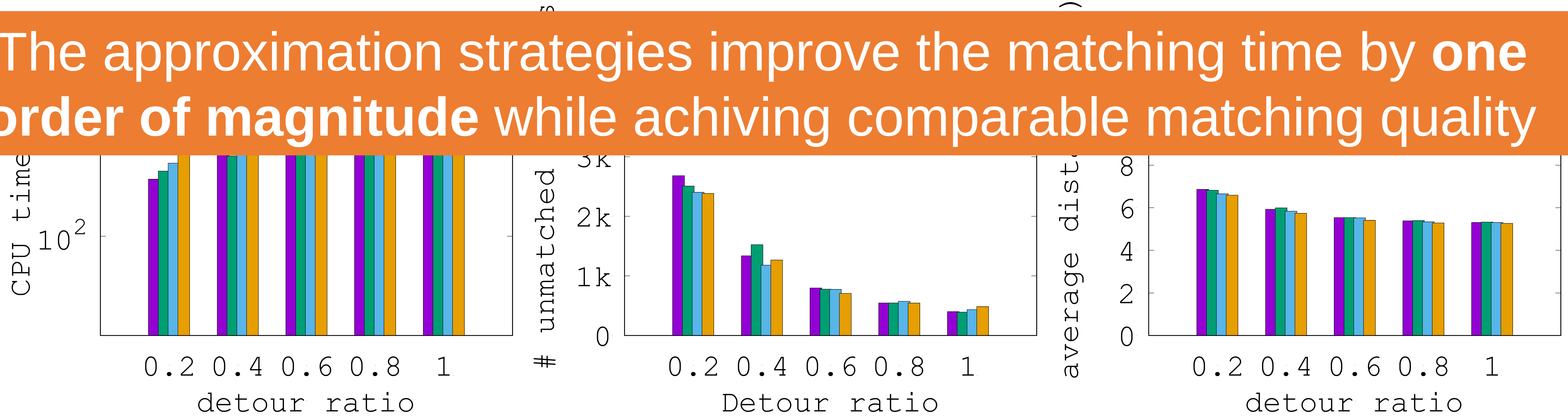
Algorithm performance

➢ Effect of the detour raio



Algorithm performance

- Effect of the detour raio



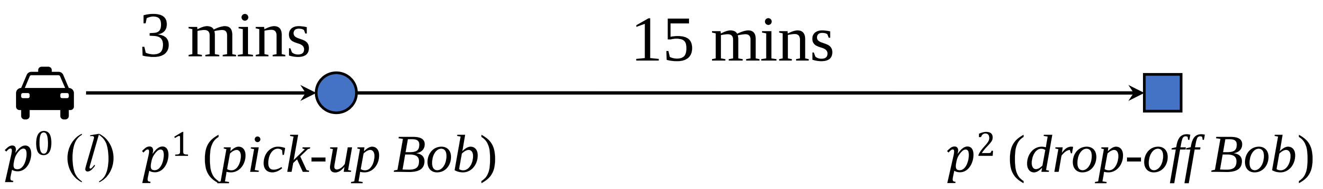
The approximation strategies improve the matching time by **one order of magnitude** while achiving comparable matching quality

Conclusion

- Benefits of Multi-hops
  - Substantially enhance flexibility of ride-sharing
  - More requests served
  - Less travel time required
- Our proposed exact algorithm outperforms the state-of-the-art by 2~3 orders of magnitude
- Our speed-up techniques accelerate the matching time by another order of magnitude
- Our efficient and scalable algorithms enable multi-hop ride-sharing in real-world

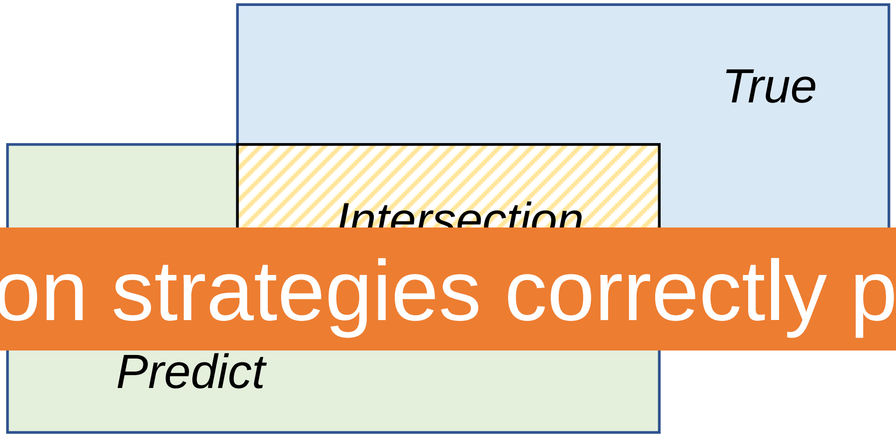
Insertion -- vehicle schedule

Bob	
issue time	9:00 am
latest pick-up	9:05 am
latest drop-off	9:23 am



	p_0	p_1	p_2
Est arrival (Arr)	9:00 am	9:03 am	9:18 am
Lat arrival (Lat)	9:00 am	9:05 am	9:23 am

Prediction quality



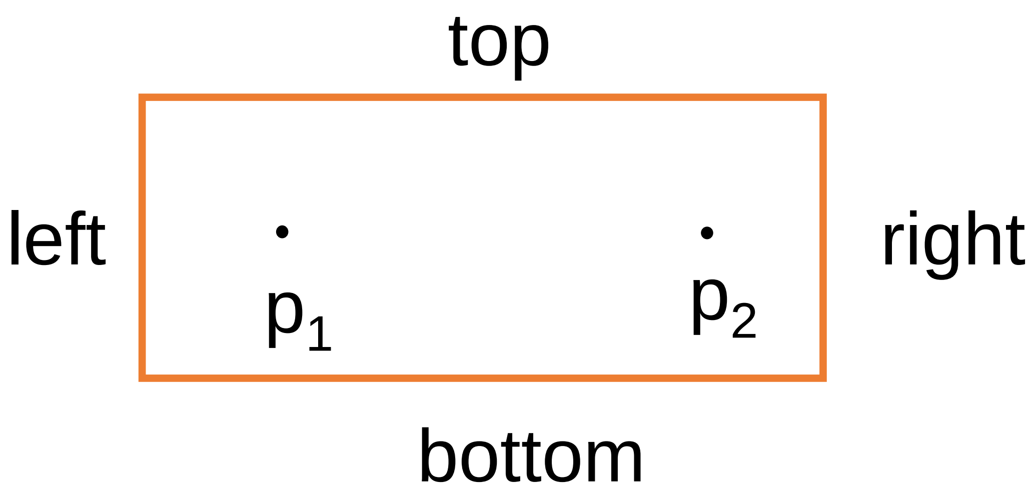
Our prediction strategies correctly predict **98%** reachable areas

$$IoT = \frac{\quad}{True}$$

Prediction	Boundary	Gap	GapCustom
IoT	94.62%	93.25%	97.68%

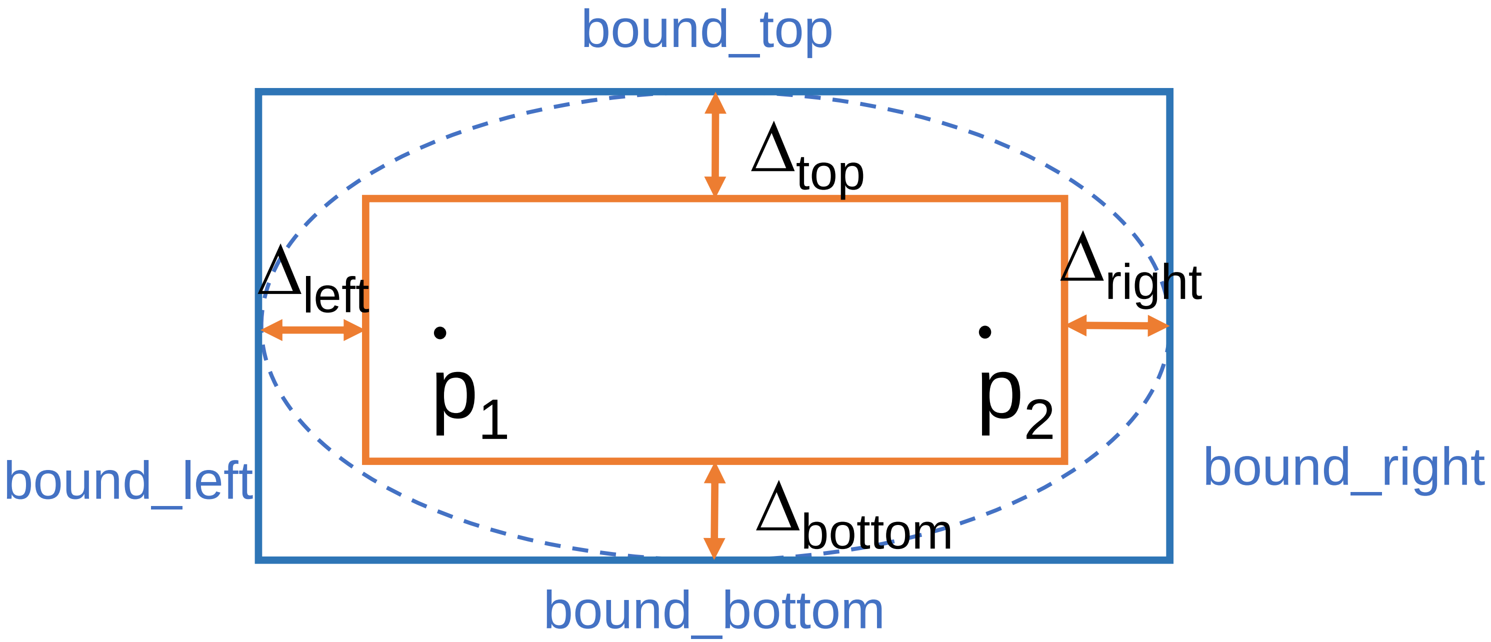
Speed-up: learning the reachable area

- Input: the two locations, the time budget
- Strategy 1:
  - predict the four boundaries (top, left, bottom, right)
  - Loss: mean square error



Speed-up: learning the reachable area

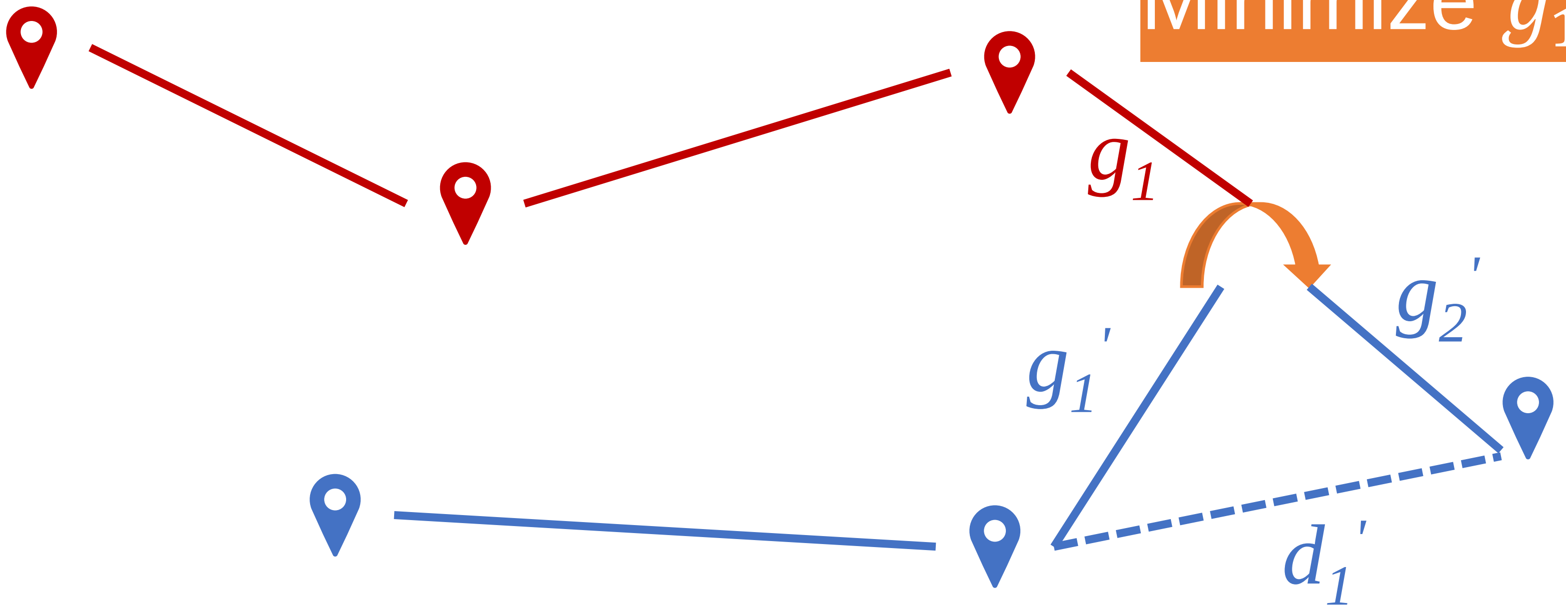
- Observation: ellipses bound the reachable area
- Strategy 2:
  - predict the four gap values (Δ_{top}, Δ_{left}, Δ_{bottom}, Δ_{right})
  - Loss function: mean square error



Vehicle-first algorithm: determine ϕ

- Key observation: the optimal transfer point depends on only 3-4 stops in the schedule.

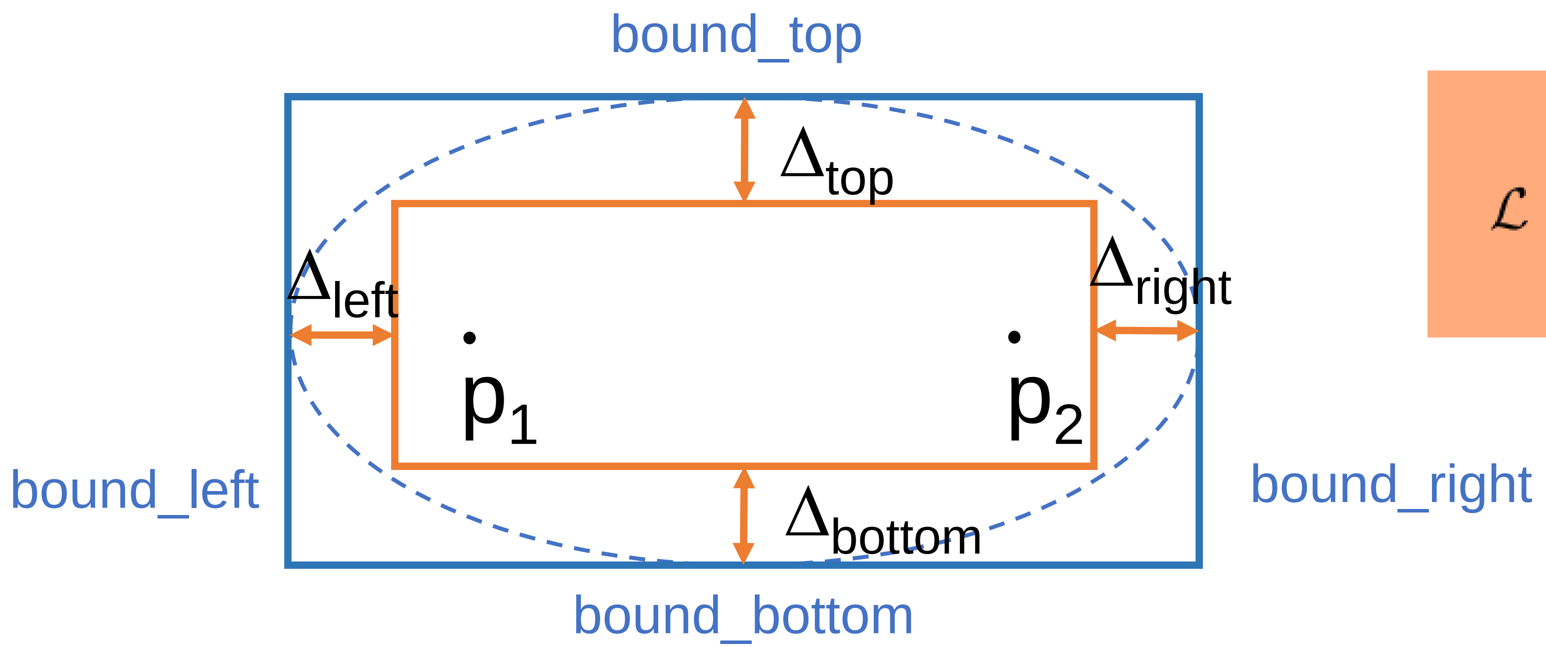
Minimize $g_1 + g_1' + g_2'$



$$\Delta = g_1 + g_1' + g_2' - d_1'$$

Speed-up: learning the reachable area

- Observation: penalize large prediction to avoid false negatives
- Strategy 2:
  - predict the four gap values (Δ_{top}, Δ_{left}, Δ_{bottom}, Δ_{right})
  - Customize loss function



$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\varphi(y\_true - y\_pred))^2$$