Online Optimization of dial-a-ride problem with the integral primal simplex

Elahe Amiri, Antoine Legrain, Issmail El Hallaoui

Column Generation 2023

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

GERAD
GROUP FOR RESEARCH IN DECISION ANALYSIS

Agenda

- Introduction
- Overview of Integral Simplex using Decomposition (ISUD)
- Methodology
- Experimental Results
- Conclusion and Future works

Ridesharing

- Mode of transportation in which travel costs are split among travelers by sharing a vehicle for their trip

UberPool
Lyft Shared, …

State-of-the-art optimization techniques, are rarely used in their algorithm

Ride sharing market value

(Wadhwani and Saha, 2019)

measured value 2019 → **more than $34 billion**

expected value 2026 → **more than $50 billion**

any operations management improvements will be very valuable

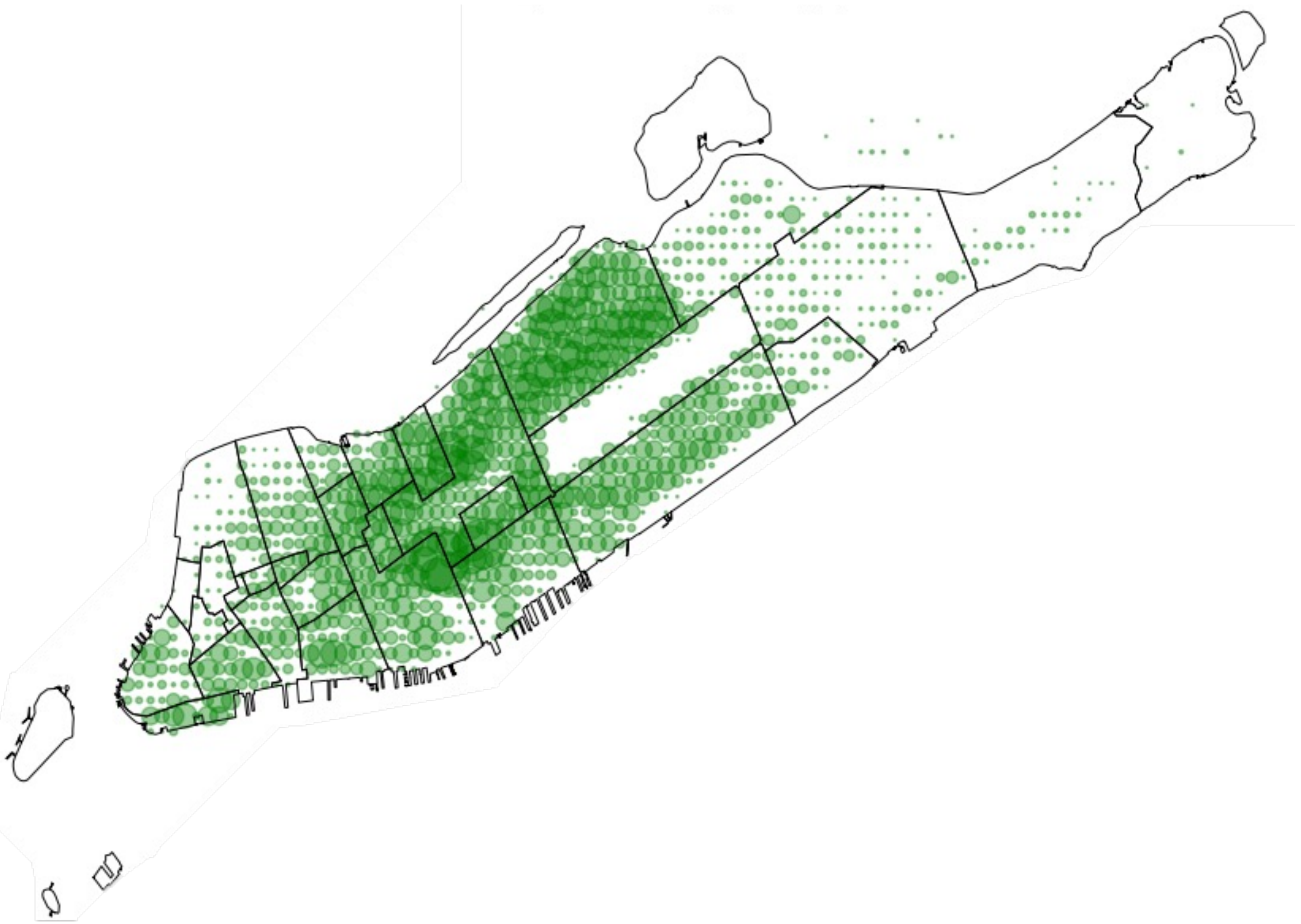
Ridesharing

On-demand transportation system

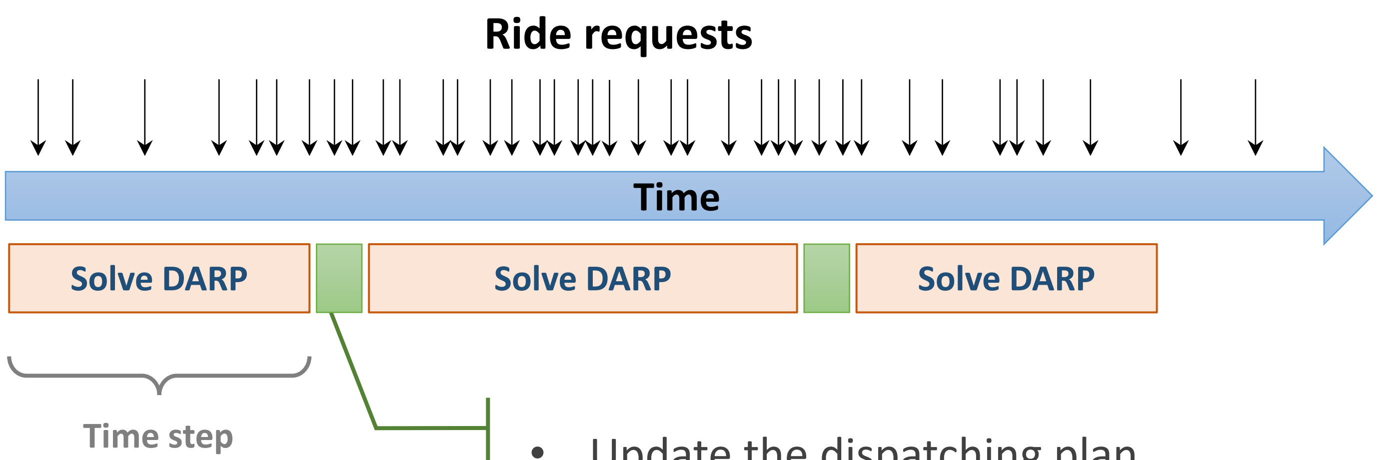
- **Smart phone application** — Receive ride requests from users (origin, destination, ready time)
- **Central server**
  - **non-shared**: match ride requests with drivers to minimize waiting time
  - **shared**: determining the sequence of pickup and drop offs → **Dial-a-Ride Problem (DARP)**

Ridesharing on Manhattan

- 20k requests per hour
- 2k vehicles
- Few seconds to re-optimize the current plan
- Data from the NYC taxi and limousine commission

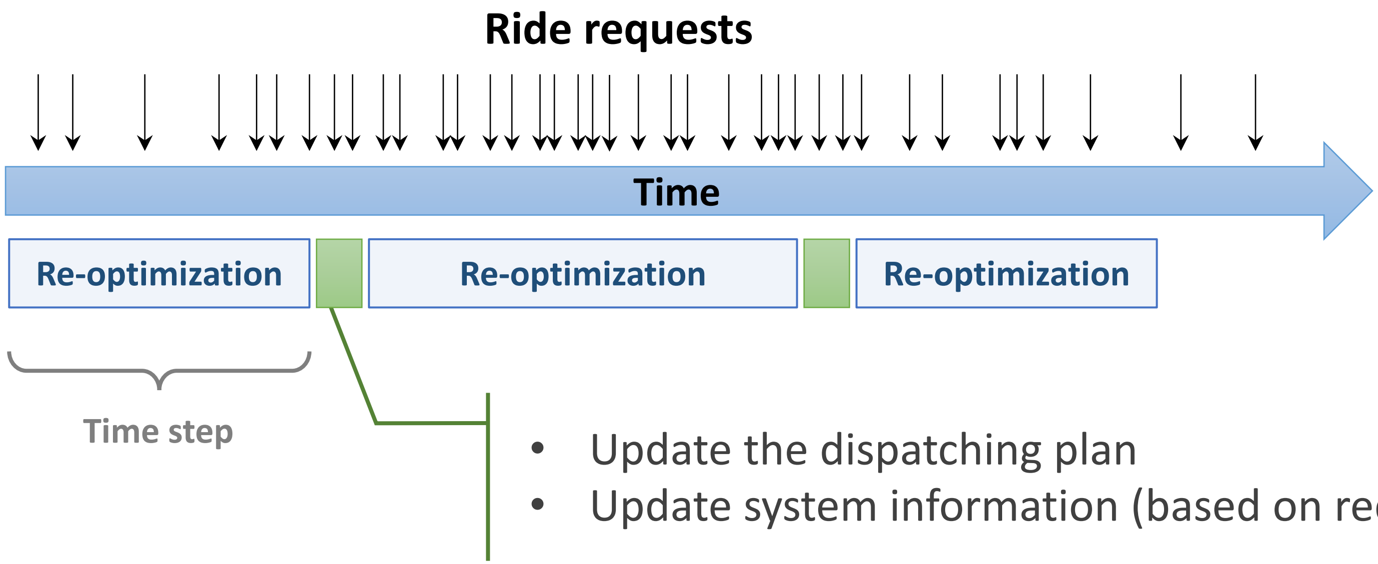


Realtime Ridesharing



- Update the dispatching plan
- Update system information (based on requests received so far)

Realtime Ridesharing



- Update the dispatching plan
- Update system information (based on requests received so far)

Negligible delay ⟸ **Any Time algorithm**

Return a sub-optimal solution within any small allocated runtime budget

- The system should always be able to provide good solutions in each given short time
- Solutions should be **<u>fast</u>** enough to provide real-time booking for customers

Exact solution methods

DARP is $\mathcal{NP}$-hard

- Obtaining an optimal solution can be computationally expensive
- Make it more challenging in real-time for large-scale problems

Exact methods

Dual fractional methods

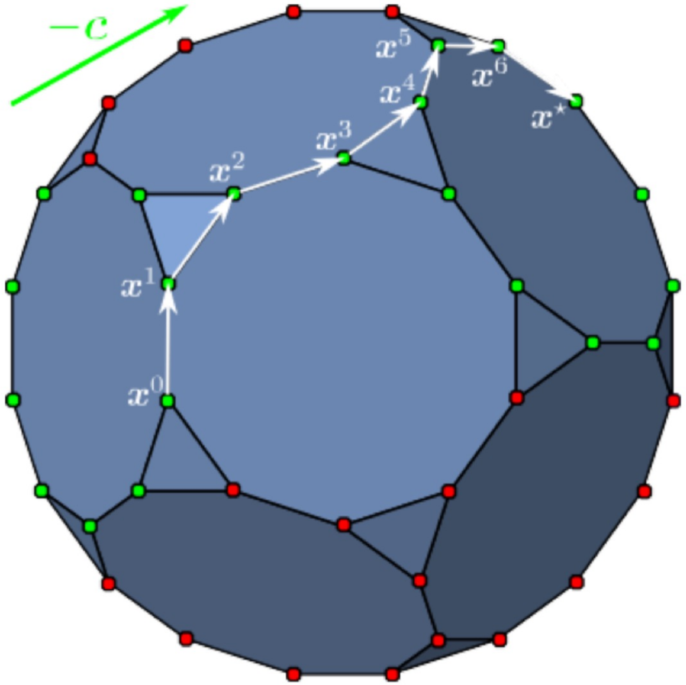
- Maintain optimality as well as the feasibility of the linear model and terminate when reach to **integrality**.
- **Branch and Price**
  - **It should explore a branch-and bound tree to reach an integer solution**
  - **it starts re-optimization from scratch and can not take advantage of a warm start**

Primal approaches

- Maintain feasibility (and integrality) throughout the process and terminate when **optimality** is achieved.

Integral Primal Simplex

- Start from an integer solution
- Find a direction of descent to improve the current solution
- Continue iterations to reach optimality.



Advantages

- **<u>Do not restart</u>** the optimization from **<u>scratch</u>**
  - Take advantage of the previous solution as the **<u>warm start</u>** to produce next solution
- A feasible integer solution is available at **<u>anytime</u>**
  - Possibility of stopping the procedure at any time if the time is limited
  - Avoid the combinatorial exploration of a branching tree to reach integer solution

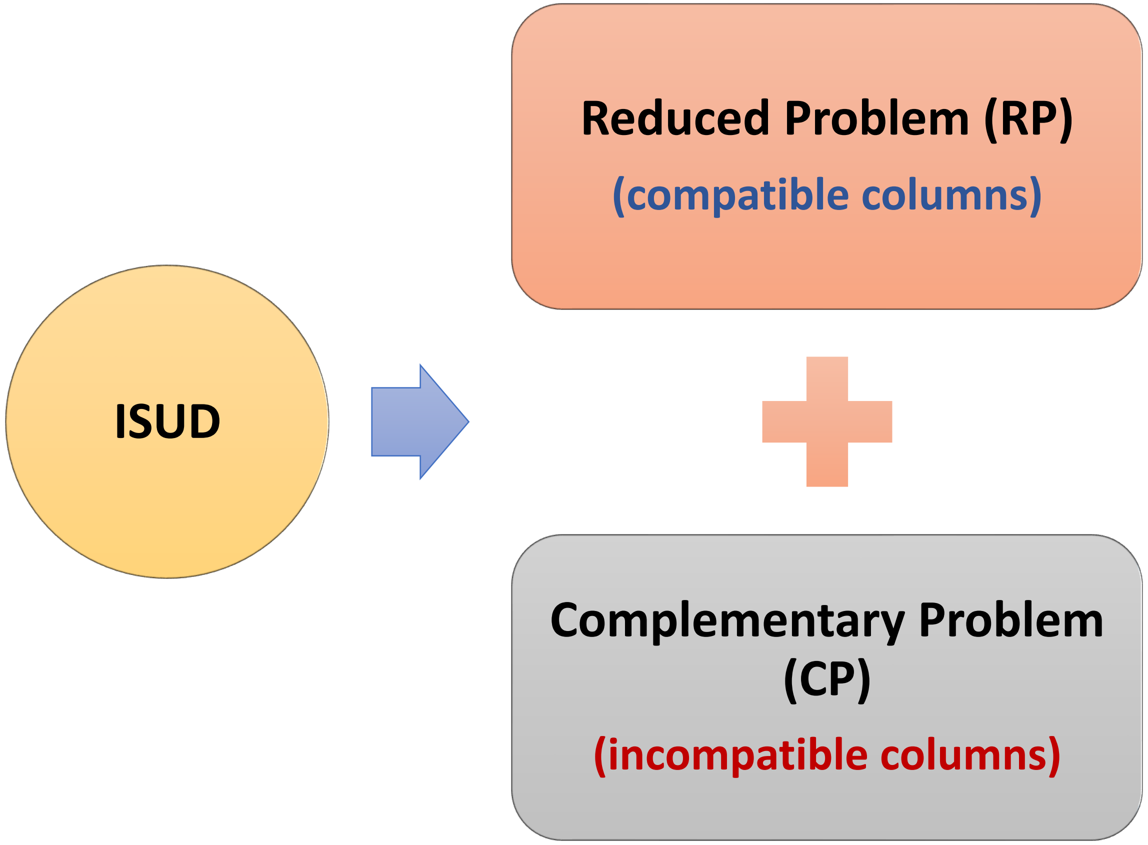
Contributions

- In the literature, a DARP has never been solved using a primal algorithm

Main contribution:

- Performing the first implementation of **Integral Column Generation** in the context of a Dial-a-Ride Problem (Set Packing Problem)
- Using the strength of **integral primal simplex** to propose an **anytime** algorithm for real-time application
- Solve **large scale** instances (2k vehicles and >50k requests over Manhattan)

Integral Simplex using Decomposition (Zaghrouti, et al. 2014)



$\mathcal{P}$: set of positive value variables in current solution:

Definition:

- *A column or a positive combination of columns is said to be **compatible with** $\mathcal{P}$ if it can be written as a **linear combination** of columns of $\mathcal{P}$*

Integral Simplex using Decomposition (Zaghrouti, et al. 2014)

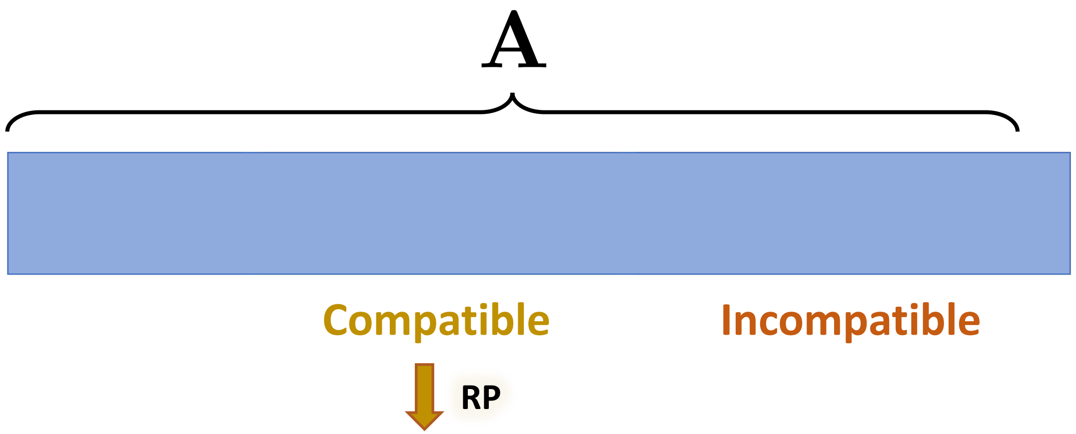
ISUD Algorithm

Step 0: Start from an initial integer solution $\mathcal{P}$

Integral Simplex using Decomposition (Zaghrouti, et al. 2014)

Set partitioning Problem

$$
\begin{aligned}
Z^*_{\text{SPP}} = \quad & \min_{\boldsymbol{\theta}} \ \boldsymbol{c}^\top \boldsymbol{\theta} \\
\text{s.t.} \ & \mathbf{A}\boldsymbol{\theta} = \boldsymbol{e} \\
& \boldsymbol{\theta} \in \{0,1\}^n
\end{aligned}
$$



$\mathcal{C}_\mathcal{P}$: Index set of compatible columns

$\mathbf{A}_{\mathcal{C}_\mathcal{P}}$: Set of columns in $\mathbf{A}$ indexed by $\mathcal{C}_\mathcal{P}$

$$
\begin{aligned}
Z^*_{RP1} = \quad & \min_{\boldsymbol{\theta}_{\mathcal{C}_\mathcal{P}}} \ \boldsymbol{c}^\top_{\mathcal{C}_\mathcal{P}} \boldsymbol{\theta}_{\mathcal{C}_\mathcal{P}} \\
\text{s.t.} \ & \\
& \mathbf{A}_{\mathcal{C}_\mathcal{P}} \boldsymbol{\theta}_{\mathcal{C}_\mathcal{P}} = \boldsymbol{e} \\
& \boldsymbol{\theta}_{\mathcal{C}_\mathcal{P}} \in \{0,1\}^{|\mathcal{C}_\mathcal{P}|}
\end{aligned}
$$

Integral Simplex using Decomposition (Zaghrouti, et al. 2014)

ISUD Algorithm

Step 0: Start from an initial integer solution $\mathcal{P}$

Step 1: ***Improve*** the current integer solution $\mathcal{P}$ by solving the RP

Integral Simplex using Decomposition (Zaghrouti, et al. 2014)

Set partitioning Problem

$$Z^*_{\text{SPP}} = \min_{\boldsymbol{\theta}} \boldsymbol{c}^\top \boldsymbol{\theta}$$
$$\text{s.t. } \mathbf{A}\boldsymbol{\theta} = \boldsymbol{e}$$
$$\boldsymbol{\theta} \in \{0,1\}^n$$

λ_j, ν_j : Weight variables defining the linear combination of compatible and incompatible columns

$$\begin{cases} \nu_j > 0 & \text{entering variables } (j \in \mathcal{I}_\mathcal{P}) \\ \lambda_j > 0 & \text{leaving variables } (j \in \mathcal{P}) \end{cases}$$

$Z^*_{CP1} < 0$ ➡ $\boldsymbol{d} = (\nu_j, -\lambda_j, 0)$

descent direction

$$\overbrace{\mathbf{A}_\mathcal{P} \quad \mathbf{A}_{\mathcal{C}_\mathcal{P}} \quad \mathbf{A}_{\mathcal{I}_\mathcal{P}}}^{\mathbf{A}}$$

Compatible

Incompatible

CP

$$Z^*_{CP1} = \min_{\nu,\lambda} \sum_{j \in \mathcal{I}_\mathcal{P}} c_j \nu_j - \sum_{l \in \mathcal{P}} c_l \lambda_l$$

Decrease cost

s.t.

$$\sum_{j \in \mathcal{I}_\mathcal{P}} \nu_j \mathbf{A}_j - \sum_{l \in \mathcal{P}} \lambda_l \mathbf{A}_l = \mathbf{0}$$

Compatibility constraints

$$\sum_{j \in \mathcal{I}_\mathcal{P}} w_j \nu_j + \sum_{l \in \mathcal{P}} w_l \lambda_l = 1$$

Normalization constraint

$$\nu \geq 0$$

Integral Simplex using Decomposition (Zaghrouti, et al. 2014)

ISUD Algorithm

Step 0: Start from an initial integer solution $\mathcal{P}$

Step 1: ***Improve*** the current integer solution $\mathcal{P}$ by solving the RP

Step 2: Solve the CP and ***Improve*** the current integer solution with a compatible combination of columns

Control: **If** *Step 2* improves the solution, go to *Step 1*. Otherwise, **return** the current solution.

Problem Description (Riley et al. 2019)

INPUTS

Vehicle Data

u_v^0 : departure time

T_v^B : vehicle start time

T_v^E : vehicle end time

Q_v : capacity of the vehicle

Ride requests data

q_i : number of people to pickup ($q_i > 0$) or drop off ($q_i < 0$)

e_i : earliest possible pickup

o_i : pickup location

d_i : drop-off location

t_i : shortest travel time between its pickup and drop-off locations

Master Problem

set packing Problem modelled as a ***set partitioning Problem***

c_r : sum of the waiting times of customers

p_i : penalty of unserved requests

$$Z^*_{MP} = \quad \min \sum_{r \in R} c_r y_r + \sum_{i \in P} p_i z_i$$

Minimize the total waiting time of served requests + penalties of unserved requests

$$\text{s.t.} \quad \left(\sum_{r \in R} y_r a_i^r \right) + z_i = 1 \qquad (\pi_i) \qquad \forall i \in P$$

Scheduling of requests

$$\sum_{r \in R^v} y_r = 1 \qquad (\sigma_v) \qquad \forall v \in V$$

Assign routes to vehicles

$$z_i \in \mathbb{N} \qquad \forall i \in P$$

Determine unscheduled requests

$$y_r \in \{0,1\} \qquad \forall r \in R$$

Determine selected/assigned routes

Pricing Subproblems

$$Z^*_{\mathrm{SP}} = \min \sum_{i \in P_v} (u_i - e_i) - \sum_{i \in P_v} \sum_{j \in \mathcal{N}_v} x_{ij}\pi_i - \sigma_v \quad (1)$$

s.t.

flow constraints

$$\sum_{j \in \mathcal{N}_v} x_{ij} = \sum_{j \in \mathcal{N}_v} x_{ij} \qquad \forall i \in \mathcal{N}_v \setminus \{0, s\} \quad (2)$$

$$\sum_{j \in \mathcal{N}_v} x_{0j} = 1 \quad (3)$$

$$\sum_{j \in \mathcal{N}_v} x_{js} = 1 \quad (4)$$

ensure to drop off onboard passengers and those that are picked up

$$\sum_{j \in \mathcal{N}_v} x_{ij} - \sum_{j \in \mathcal{N}_v} x_{n+i,j} = 0 \qquad \forall i \in P_v \quad (5)$$

$$\sum_{j \in \mathcal{N}_v} x_{ij} = 1 \qquad \forall j \in I_v \quad (6)$$

control arrival time to nodes

$$u_j \geq (u_i + \varepsilon_i + t_{ij})\, x_{ij} \qquad \forall i, j \in \mathcal{N}_v \quad (7)$$

$$u_i \geq e_i \qquad \forall i \in P_v \quad (8)$$

$$u_0 \geq T^B_v \quad (9)$$

$$u_s \leq T^E_v \quad (10)$$

control travel time duration

$$t_i \leq u_{n+i} - (u_i + \varepsilon_i) \leq \max\{\alpha t_i, \beta + t_i\} \qquad \forall i \in P_v \quad (11)$$

$$t_i \leq u_i - \left(u^P_i + \varepsilon_i\right) \leq \max\{\alpha t_i, \beta + t_i\} \qquad \forall i \in I_v \quad (12)$$

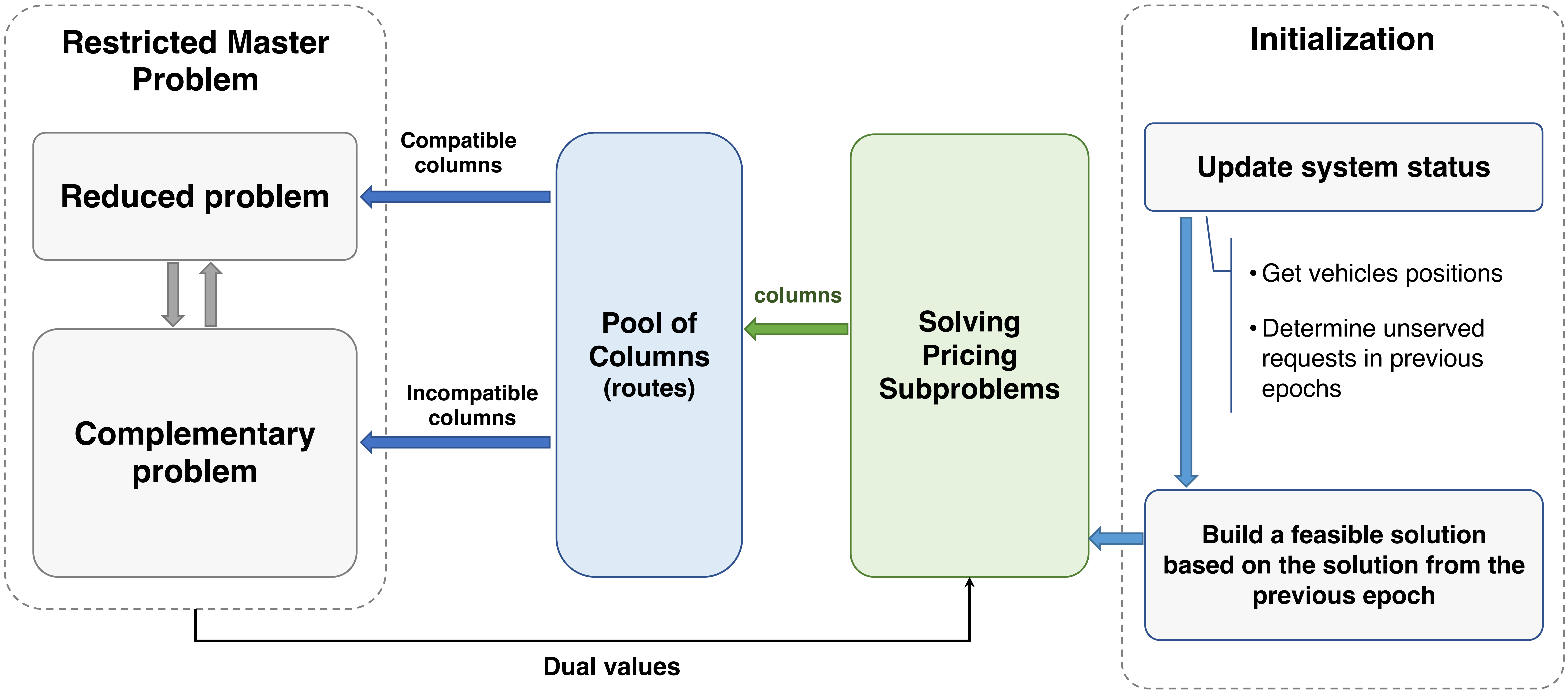
ensure vehicle capacity

$$\omega_j \geq (\omega_i + q_j)\, x_{ij} \qquad \forall i, j \in \mathcal{N}_v \quad (13)$$

$$0 \leq \omega_i \leq Q_v \qquad \forall i \in \mathcal{N}_v \quad (14)$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{N}_v \quad (15)$$

General ICG framework (re-optimization in real-time)



Pricing Subproblems

- Use Dynamic Programming approach **(Ghilas et al. 2018)**
- **<u>Forward labeling algorithm</u>**

Label Data

- last node of the partial path
- accumulated reduced cost
- reach time to the last node
- set of onboard requests
- set of completed/onboard requests
- number of passengers in the vehicle at last node
- available travel times for onboard requests based on Max travel time

pairwise comparison within the dominance rules

ensure vehicle capacity constraint

ensure trip duration deviation

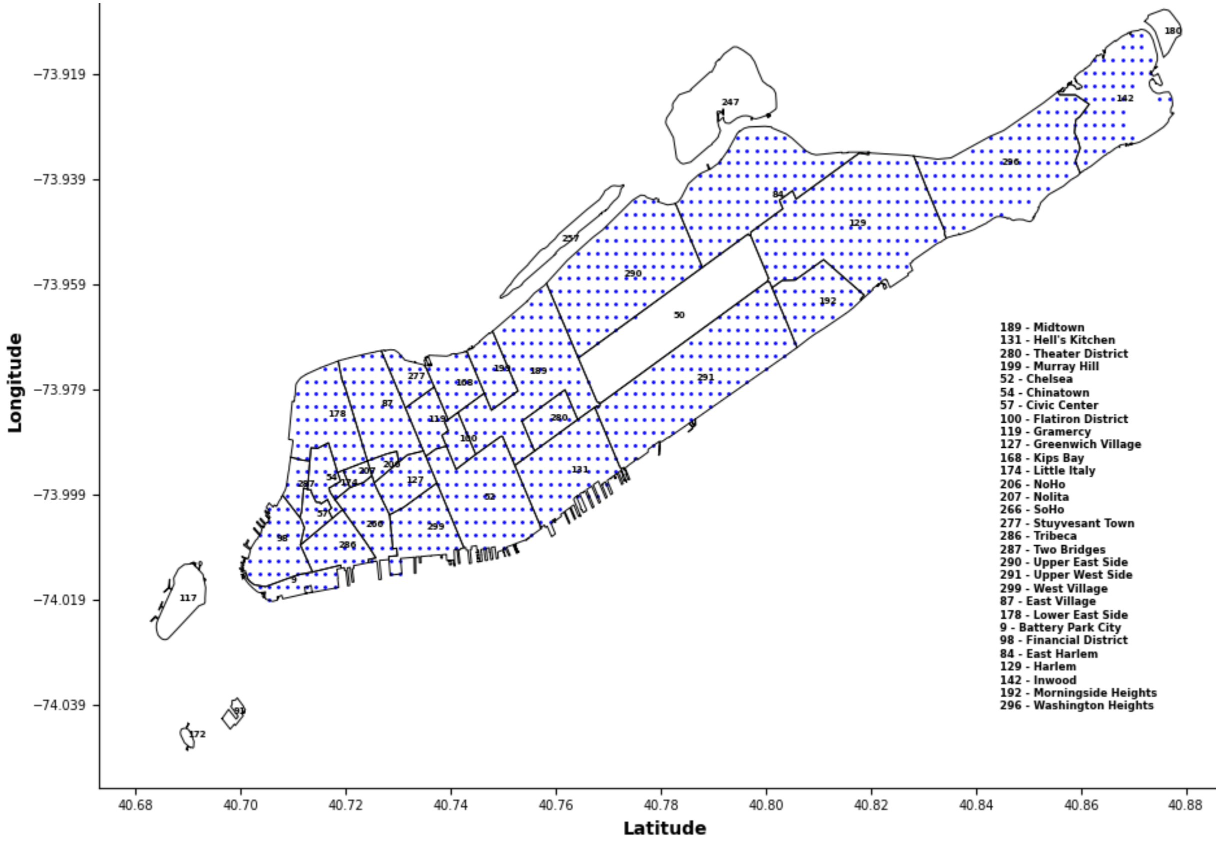
Acceleration Techniques

- Truncated labeling **(Dabia et al. 2017)**
- Avoid visiting pickup nodes after drops

Instance Description (Riley et al. 2020)

- **New York City Taxi and Limousine Commission**
- Manhattan is divided into a grid

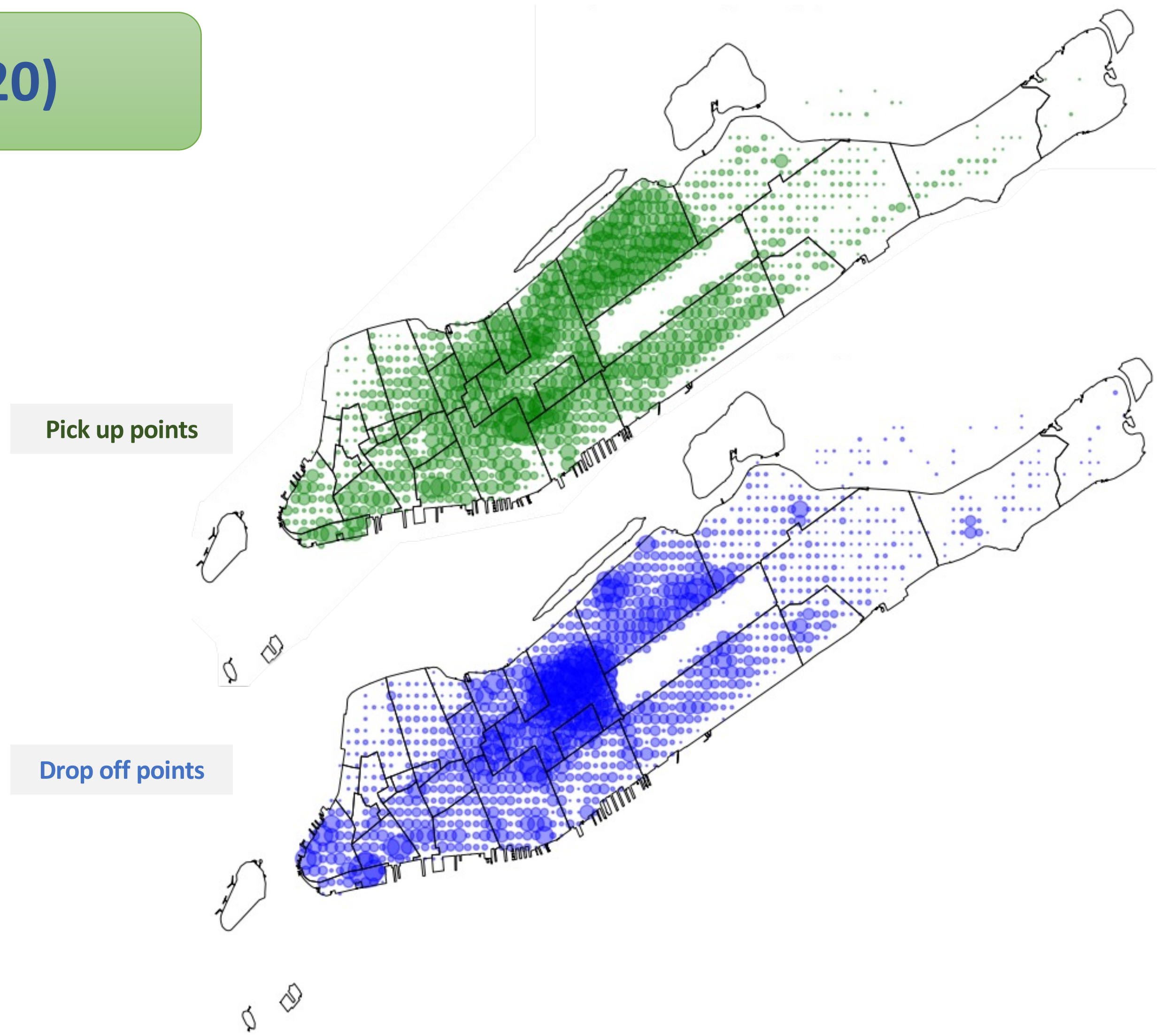
  of cells of 200 square meters



Experimental Results

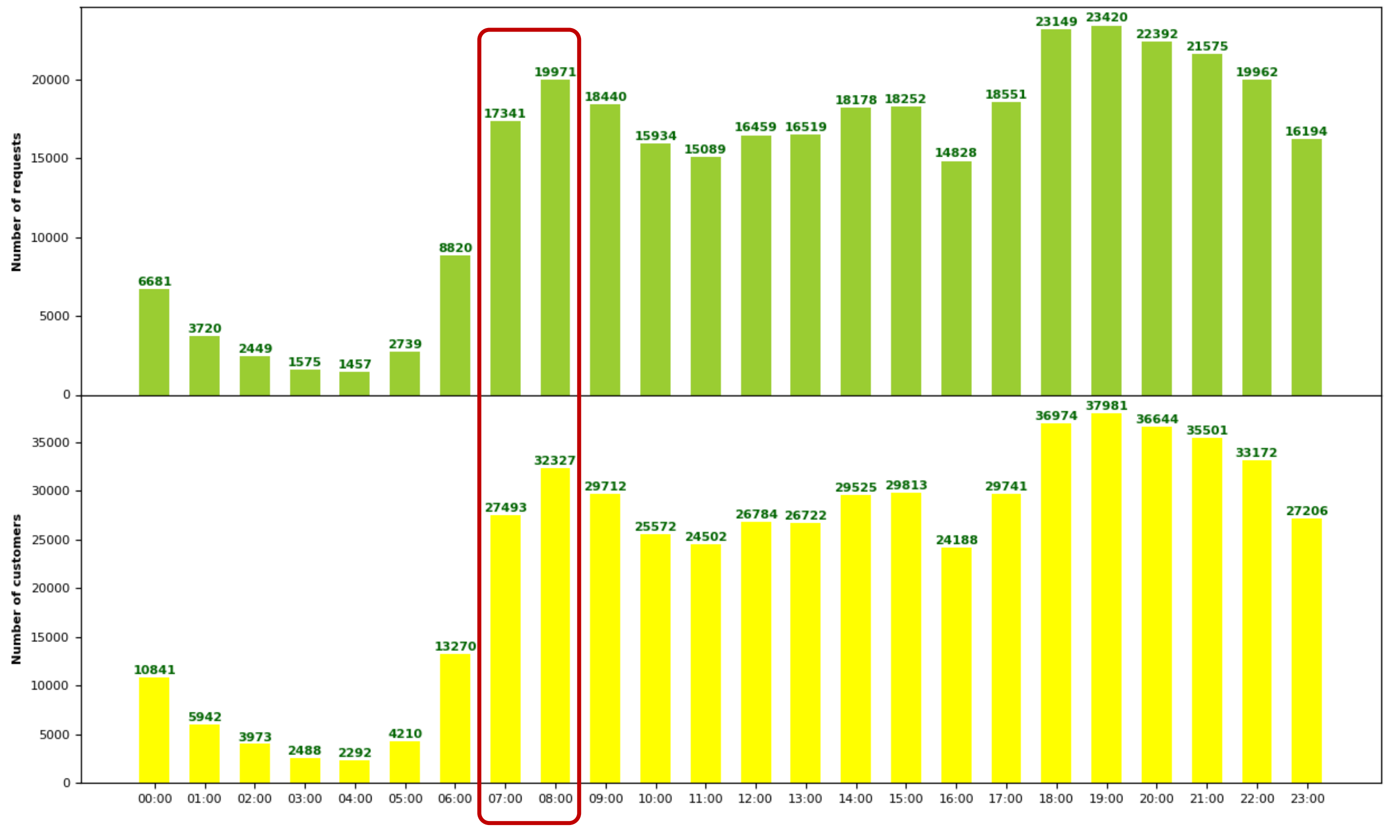
Instance Description (Riley et al. 2020)

- New York City Taxi and Limousine Commission
- Manhattan is divided into a grid of cells of 200 square meters
- 24 Instances
- July 2015 to June 2016



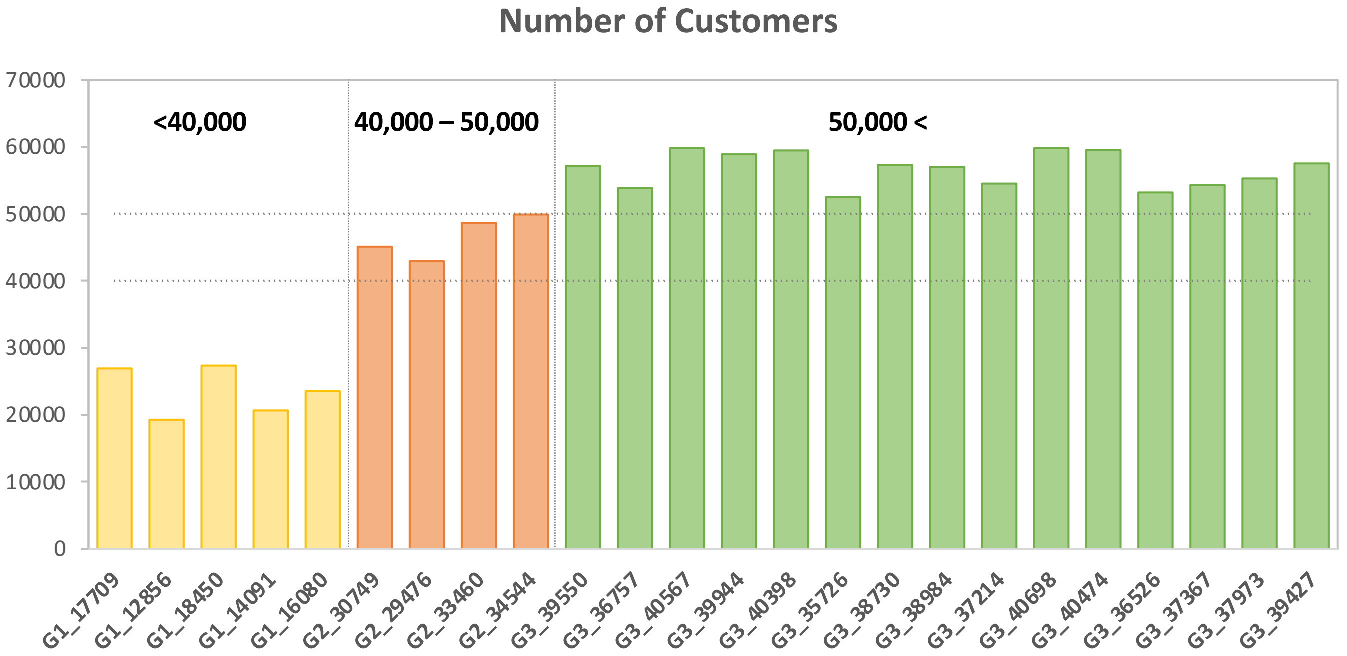
Instance Description (Riley et al. 2020)

- New York City Taxi and Limousine Commission
- Manhattan is divided into a grid of cells of 200 square meters
- 24 Instances
- July 2015 to June 2016
- 2 days a month (7 AM to 9 AM)



Instance Description (Riley et al. 2020)

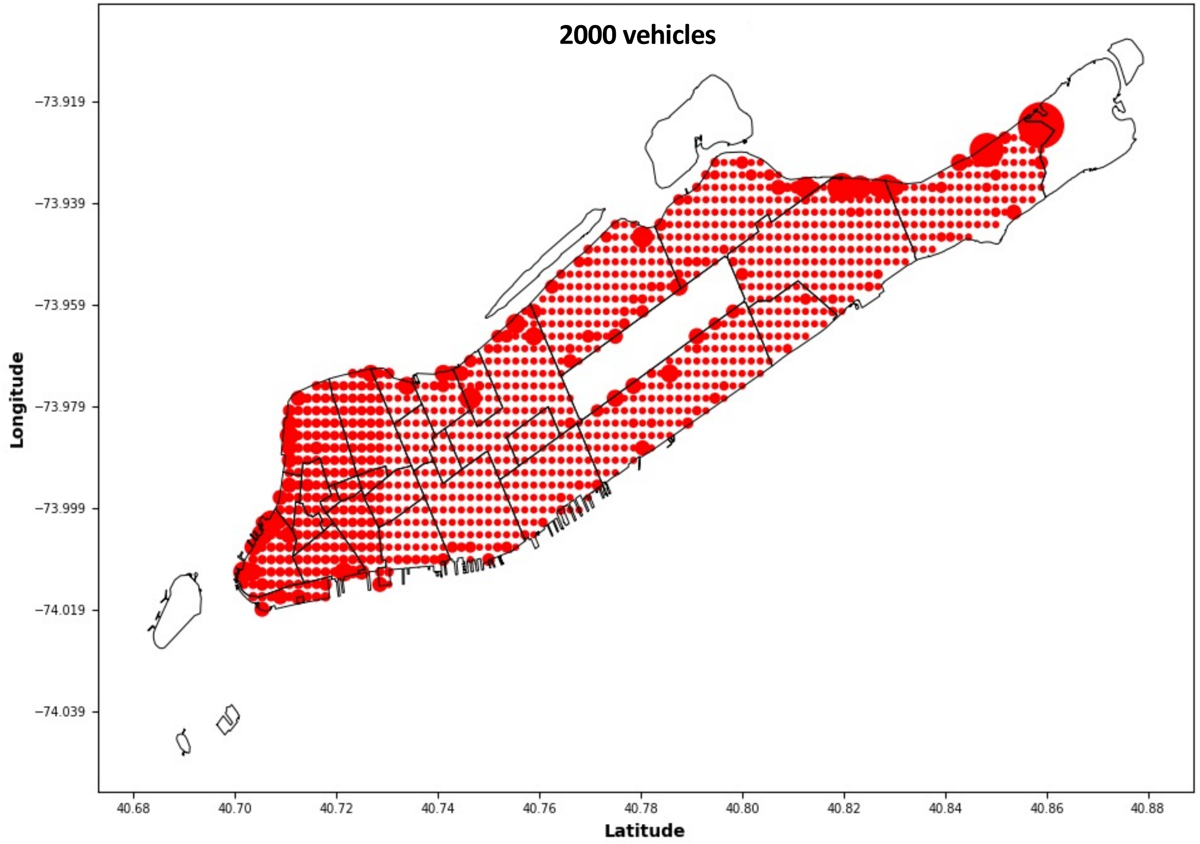
- New York City Taxi and Limousine Commission
- Manhattan is divided into a grid of cells of 200 square meters
- 24 Instances
- July 2015 to June 2016
- 2 days a month (7 AM to 9 AM)
- Customers ranges from 19,276 to 59,820



Vehicle fleet distribution

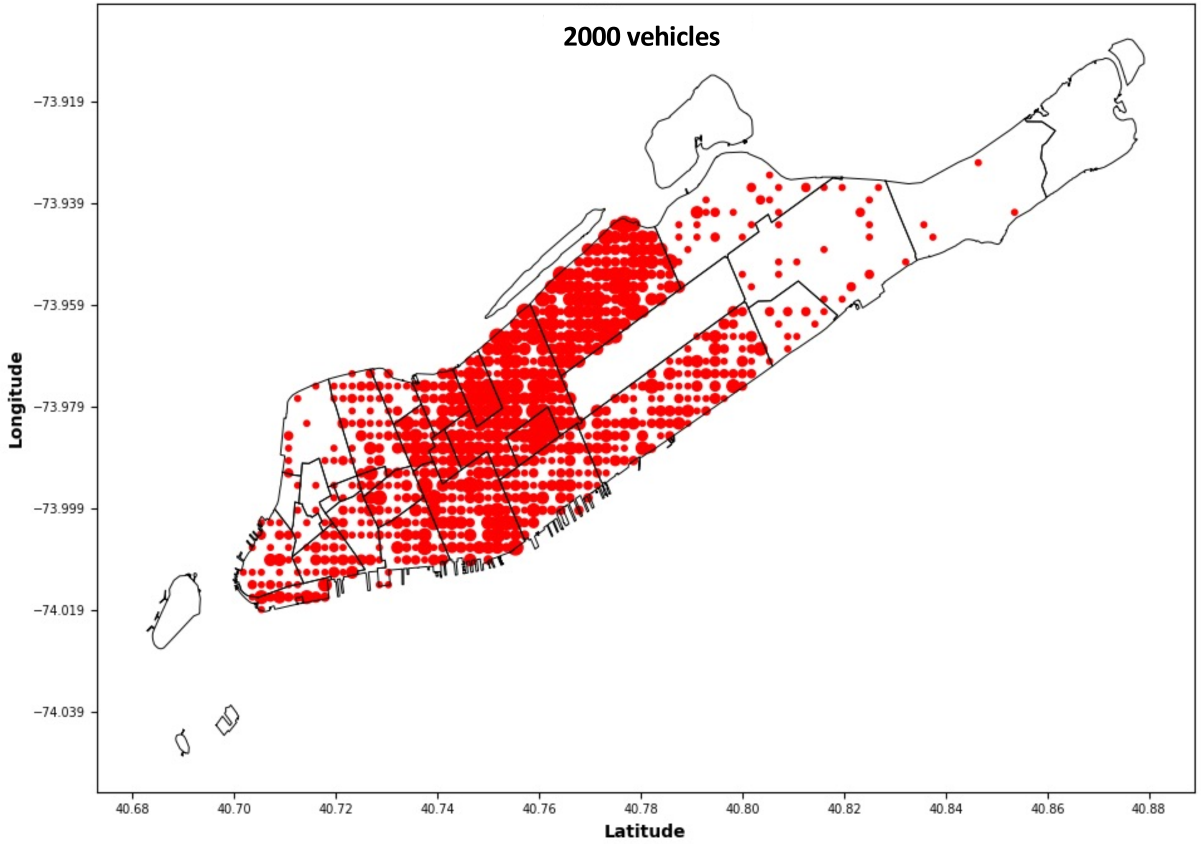
Even Distribution

Set V1



Distribution Based on average demands

Set V2



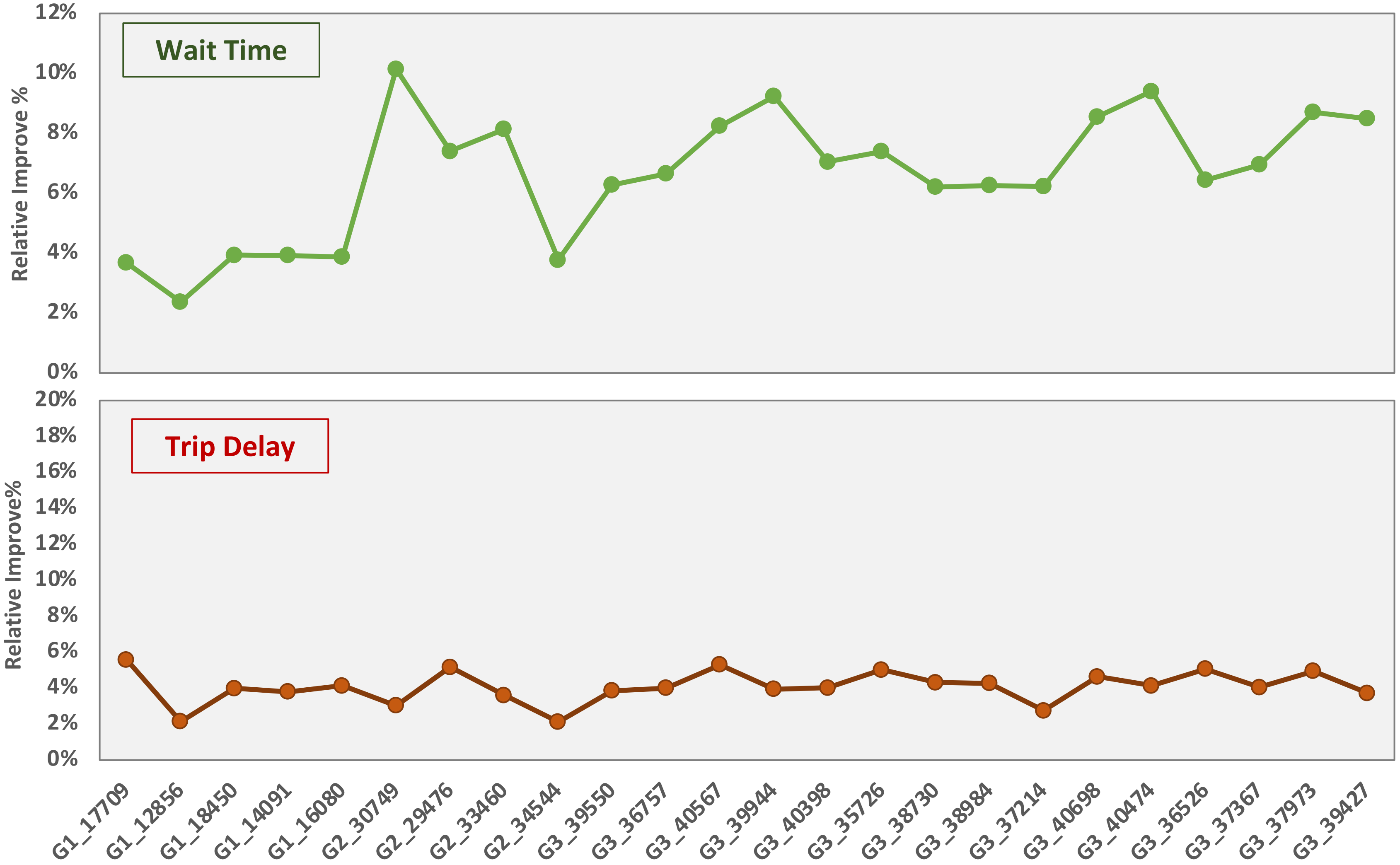
Numerical Results

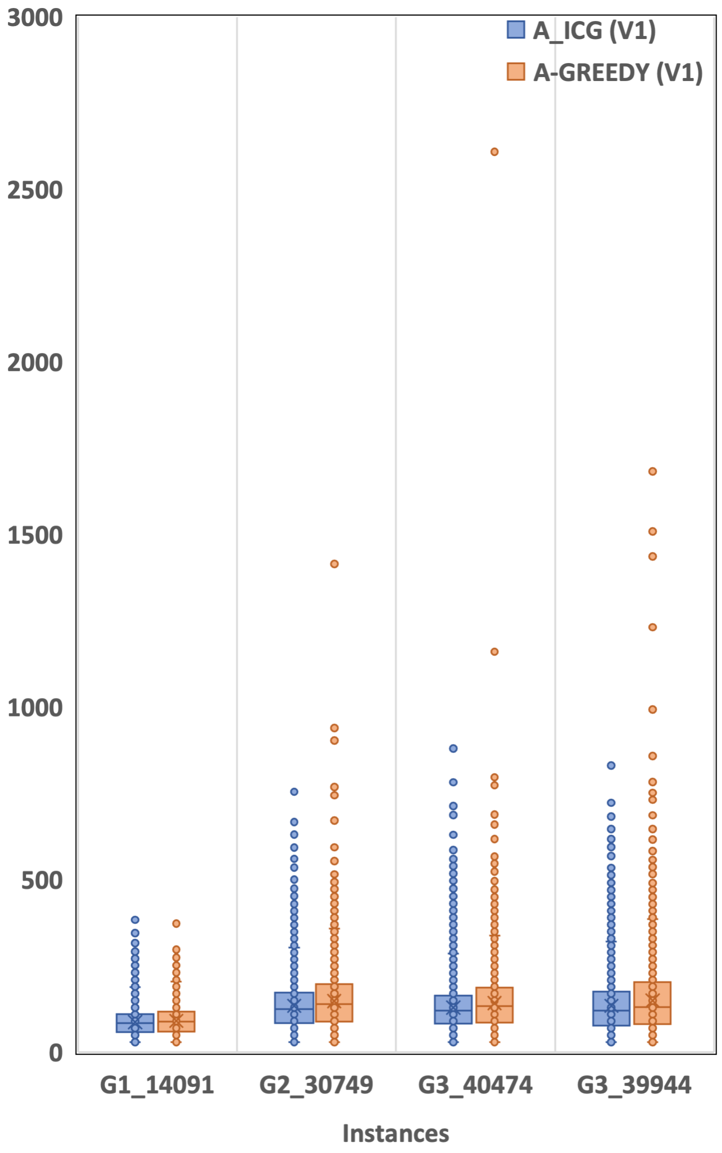
Instance	2000 vehicles (V1)				1600 vehicles (V2)	
	F-Greedy	F-ICG	A-Greedy	A-ICG	A-Greedy	A-ICG
G1_17709	103.2	100.0	85.3	82.1	79.8	79.5
G1_12856	97.9	95.1	81.4	79.5	74.0	72.8
G1_18450	110.9	110.1	97.0	93.2	94.7	93.0
G1_14091	107.0	104.5	90.8	87.2	78.6	76.5
G1_16080	103.0	98.7	84.7	81.4	73.7	72.8
< 40,000	**104.4**	**101.7**	**87.8**	**84.7**	**80.2**	**78.9**

Instance	2000 vehicles (V1)				1600 vehicles (V2)	
	F-Greedy	F-ICG	A-Greedy	A-ICG	A-Greedy	A-ICG
G2_30749	166.0	147.2	149.5	134.3	139.4	128.4
G2_29476	142.5	132.9	127.0	117.6	116.6	110.7
G2_33460	155.7	137.1	136.0	124.9	121.9	111.3
G2_34544	160.4	140.3	137.6	132.4	134.5	125.4
40,000 - 50,000	**156.2**	**139.4**	**137.5**	**127.3**	**128.1**	**119.0**

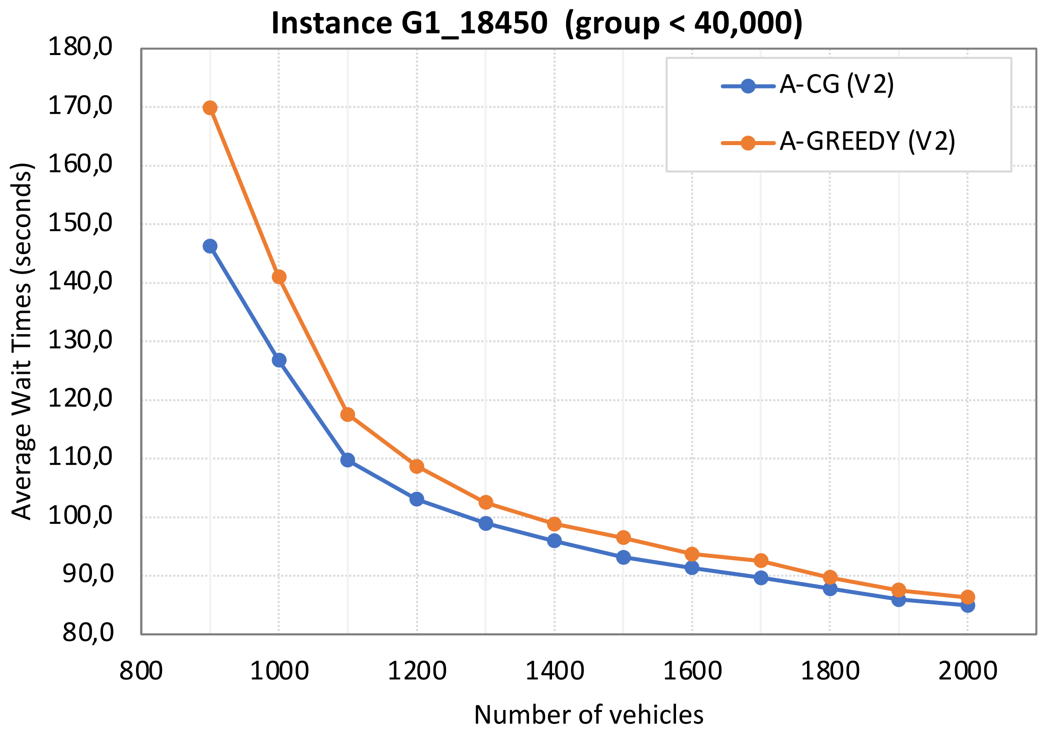
Instance	2000 vehicles (Set V1)				1600 vehicles (V2)	
	F-Greedy	F-ICG	A-Greedy	A-ICG	A-Greedy	A-ICG
G3_39550	176.7	151.5	154.1	144.4	164.0	149.1
G3_36757	166.2	149.1	150.8	140.8	153.2	143.6
G3_40567	159.8	140.8	140.6	129.0	148.3	135.2
G3_39944	164.5	146.9	148.4	134.7	143.1	130.8
G3_40398	158.5	141.1	139.6	129.8	128.5	116.3
G3_35726	150.3	134.5	127.6	118.2	119.3	110.0
G3_38730	144.6	129.3	122.6	115.0	114.3	108.8
G3_38984	145.3	132.5	127.4	119.4	120.1	107.7
G3_37214	155.8	136.9	136.2	127.7	134.4	126.5
G3_40698	166.4	146.3	146.9	134.3	150.4	133.5
G3_40474	164.3	143.7	142.7	129.3	140.7	130.5
G3_36526	170.6	152.3	153.6	143.7	161.3	149.2
G3_37367	164.6	145.6	145.4	135.3	146.4	131.6
G3_37973	159.8	141.2	142.0	129.6	135.9	126.0
G3_39427	163.7	141.2	143.7	131.5	133.5	122.0
50,000 <	**160.7**	**142.2**	**141.4**	**130.9**	**139.6**	**128.1**

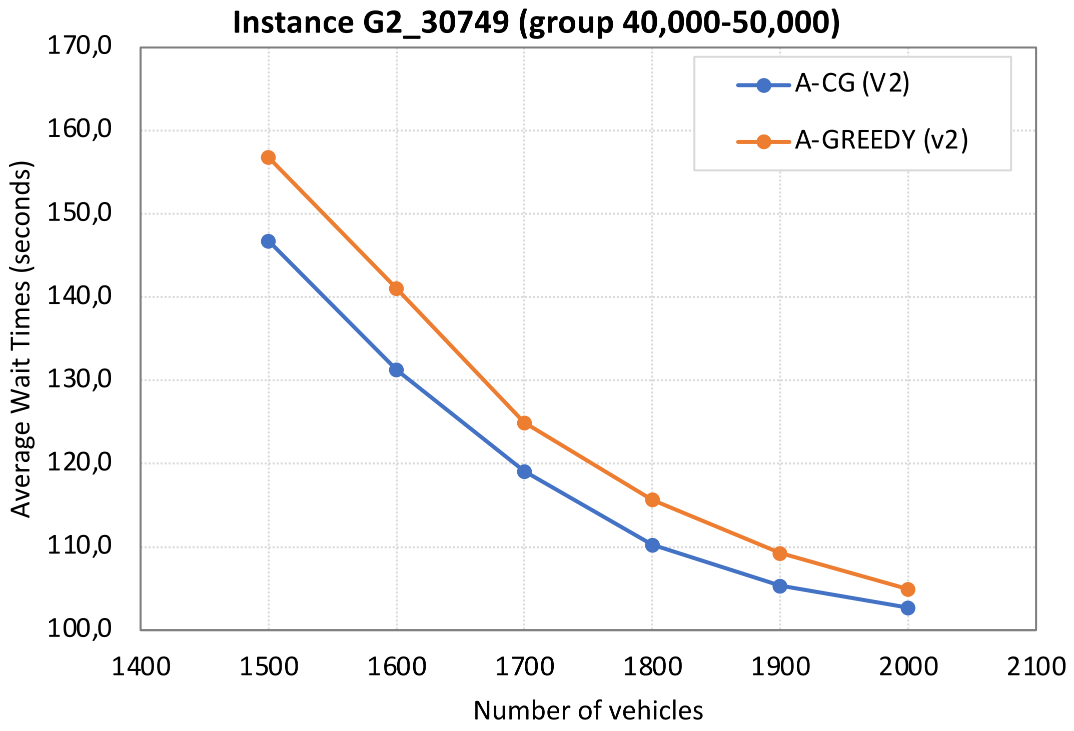
Comparison with Greedy Approach:



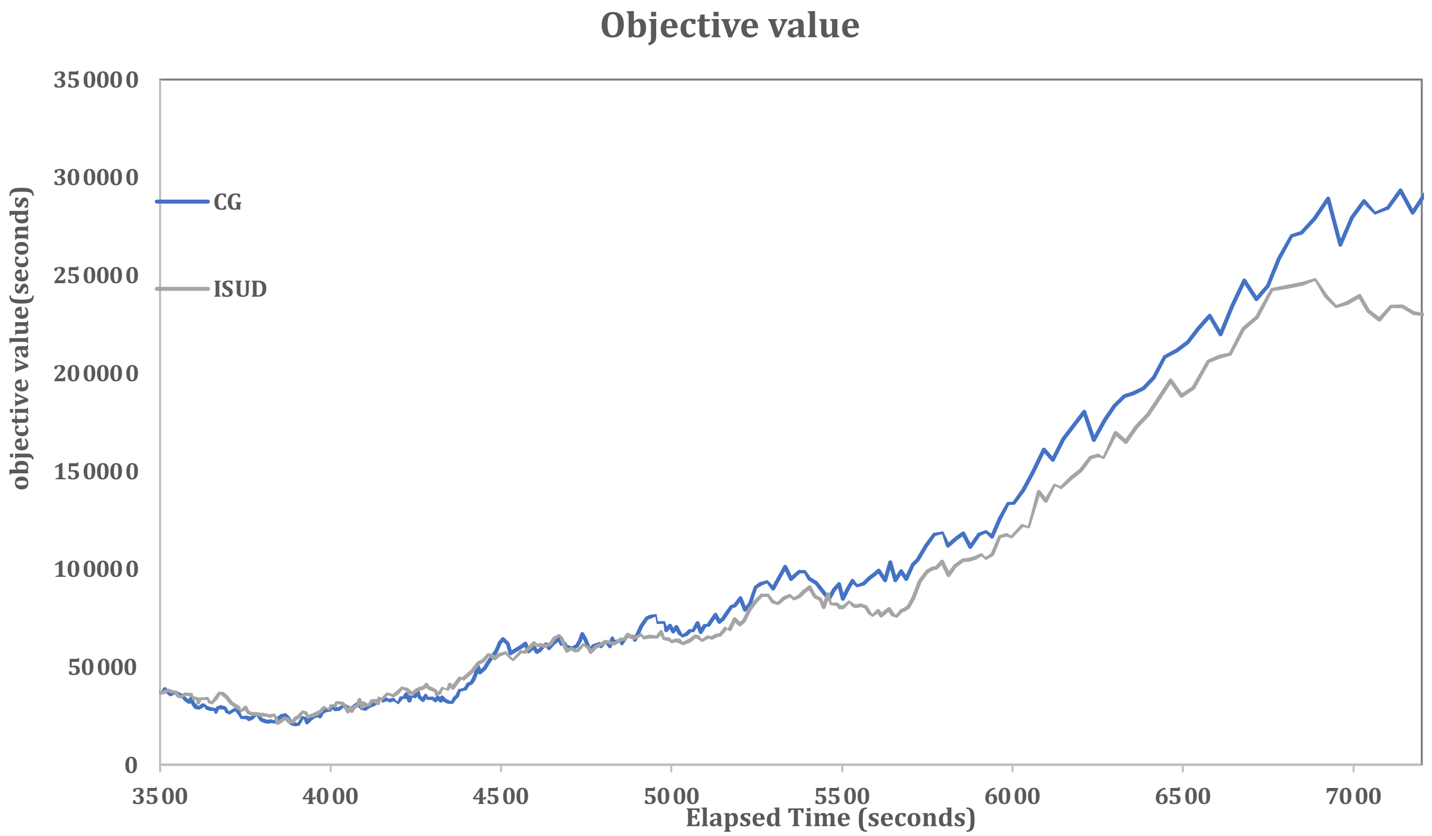


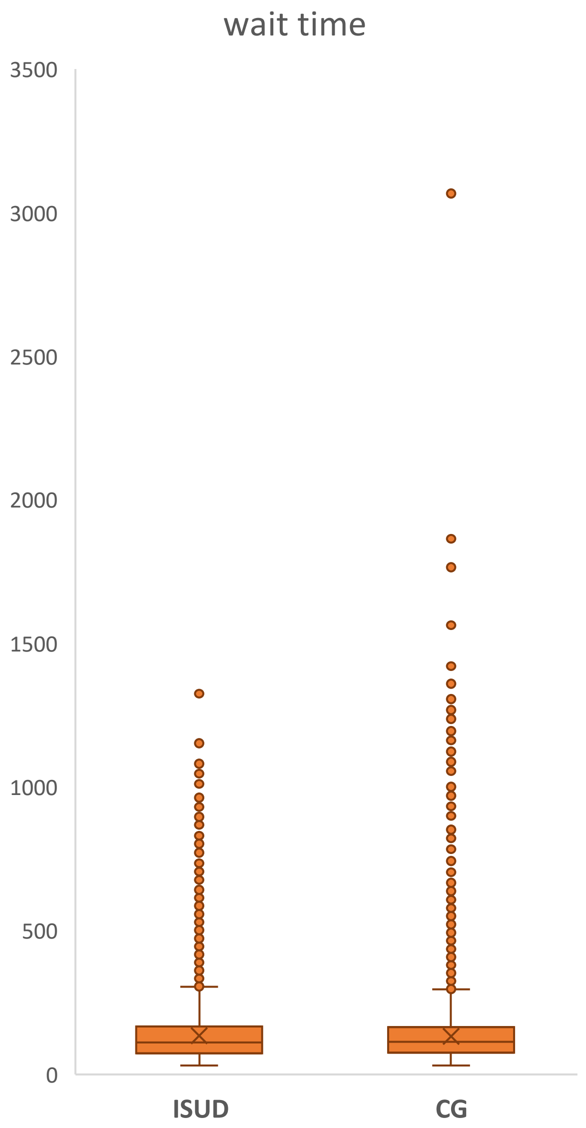
Sensitivity analysis





Comparison with Column generation (preliminary results)





Take-home message

- Develop a nearly **anytime** discrete optimization algorithm for dynamic in a large-scale ride-sharing system
- Propose a flexible rolling horizon for re-optimizing the dispatching plan
- Evaluate the proposed method on large-size instances from New York City Taxi Dataset with up to 59820 customers
- About 45% decrease in average wait time compared to M-RTRS and 20% improve over A-RTRS
- Decreasing the size of vehicle fleet by 20% with out reducing the efficiency by just distributing the vehicles based on average demands

Future Work

- build a policy based on RL techniques to adjust the parameters of CP
- Put the algorithm into practice