On the Real-time Vehicle Placement Problem

Abhinav Jauhri, Carlee Joe-Wong, John Paul Shen

ECE Department
Carnegie Mellon University

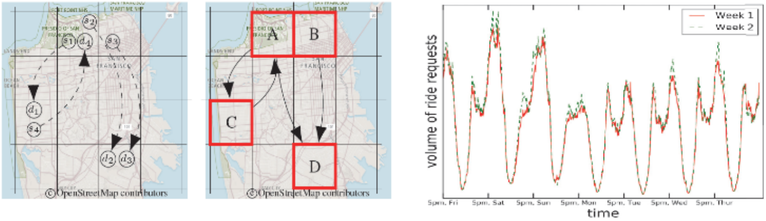
December 9, 2017

Background

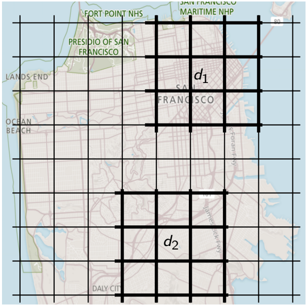
The AAAI-17 Workshop on AI and Operations Research for Social Good - WS-17-01

Space-Time Graph Modeling of Ride Requests Based on Real-World Data

Abhinav Jauhri,[1] Brian Foo,[2] Jérôme Berclaz,[2] Chih Chi Hu,[1]
Radek Grzeszczuk,[2] Vasu Parameswaran,[2] John Paul Shen[1]
[1]Carnegie Mellon University, USA; [2]Uber Technologies, Inc., USA
{ajauhri, chihhu, jpshen}@cmu.edu; {bfoo, jrb, radek, vasu}@uber.com

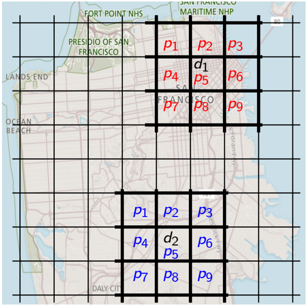


Problem Definition

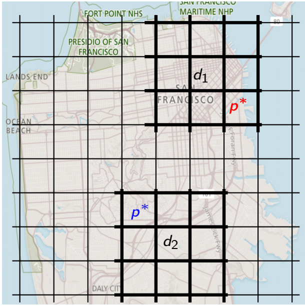


- d_i - dropoffs at time snapshot t

Problem Definition



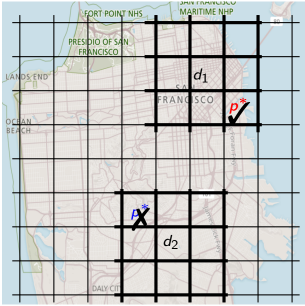
- d_i - dropoffs at time snapshot t
- p_i - possible placements for d_1 by time snapshot $t+1$
- p_i - possible placements for d_2 by time snapshot $t+1$



- d_i - dropoffs at time snapshot t
- p^* - placement for d_1 by time snapshot $t + 1$
- p^* - placement for d_2 by time snapshot $t + 1$

Two placements are made using some algorithm.

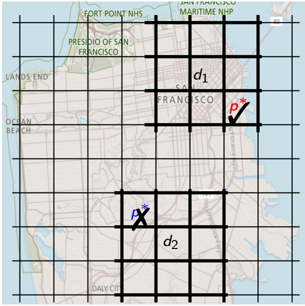
Problem Definition



- d_i - dropoffs at time snapshot t
- p^* - placement for d_1 by time snapshot $t+1$
- p^* - placement for d_2 by time snapshot $t+1$

Two placements are made using some algorithm.

Problem Definition



- d_i - dropoffs at time snapshot t
- p^* - placement for d_1 by time snapshot $t+1$
- p^* - placement for d_2 by time snapshot $t+1$

Two placements are made using some algorithm.

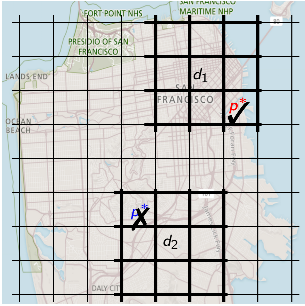
Reward R is computed for every time snapshot:

$$R(t+1) = \frac{\#\text{good placements}}{\#\text{total placements}}$$

For the example above:

$$R(t+1) = \frac{1}{2}$$

Problem Definition



- d_i - dropoffs at time snapshot t
- p^* - placement for d_1 by time snapshot $t+1$
- p^* - placement for d_2 by time snapshot $t+1$

Two placements are made using some algorithm.

Reward R is computed for every time snapshot:

$$R(t+1) = \frac{\#\text{good placements}}{\#\text{total placements}}$$

Objective: Maximize the reward R over multiple time snapshots.

Assumptions & Properties

1. Future pickup cells are not known.

Assumptions & Properties

1. Future pickup cells are not known.

2. Possible placements cells are close to drop off and finite.

Assumptions & Properties

1. Future pickup cells are not known.
2. Possible placements cells are close to drop off and finite.
3. Each cell covers a small geographical area (like $100x100m^2$).

Assumptions & Properties

1. Future pickup cells are not known.
2. Possible placements cells are close to drop off and finite.
3. Each cell covers a small geographical area (like $100x100m^2$).
4. $|(t+1)-t| < \tau_{epsilon}$ (usually a few minutes).

Potential Algorithms

1. Pick a cell uniformly at random, and no history (URand-NH).

Potential Algorithms

1. Pick a cell uniformly at random, and no history (URand-NH).

2. Follow the Leader with Complete History (FTL-CH).

Potential Algorithms

1. Pick a cell uniformly at random, and no history (URand-NH).
2. Follow the Leader with Complete History (FTL-CH).
3. Assume each cell follows a Poisson Process for ride requests (PP-LH).

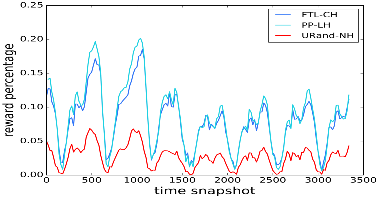
Experimental Setup

1. Looked at ≈ 10 million real ride requests for over a week in four US cities. Each ride request is defined by:
   - Pickup
   - Dropoff
   - Time of pickup
   - Time of dropoff

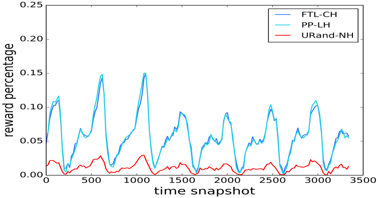
2. Each time snapshot is 3 minutes long.

3. Grid length $100m$.

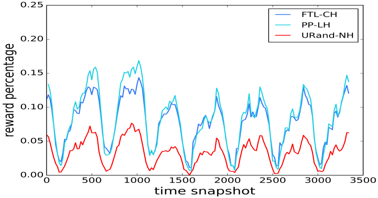
Results



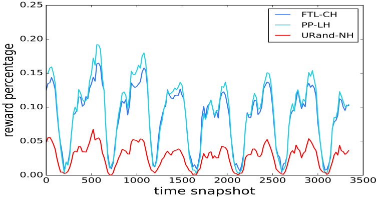
(a) Chicago



(b) Los Angeles



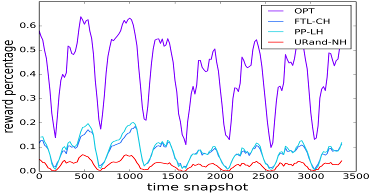
(c) New York



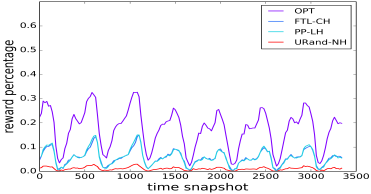
(d) San Francisco

Figure: The PP-LH algorithm out-performs FTL-CH slightly and URand-NH significantly across all four cities in terms of the reward.

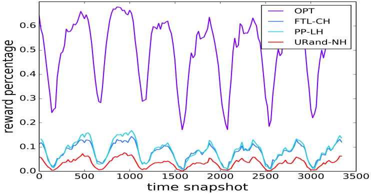
Results with OPT



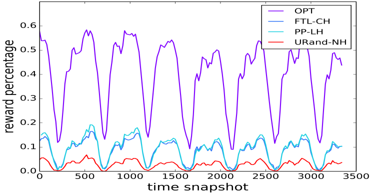
(a) Chicago



(b) Los Angeles

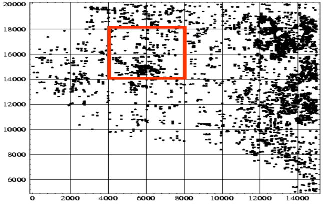


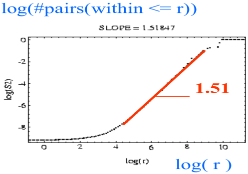
(c) New York



(d) San Francisco

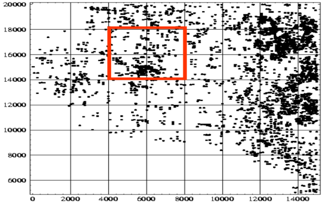
Figure: Comparison of reward percentage plots for 3 algorithms along with optimal (OPT) reward.

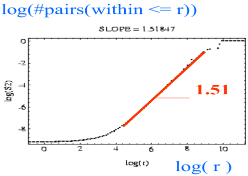




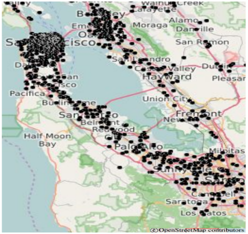
(a) **Known work:** Self-similarity for cross roads of Montgomery county.

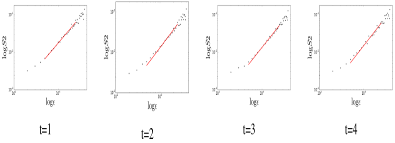
Fractals





(a) **Known work:** Self-similarity for cross roads of Montgomery county.





(b) **Our contribution:** Self-similarity for ride requests in Bay Area.

Fractal Dimensionality & Human Mobility Pattern

[Belussi 1998] Given a set of points $\mathbb{P}$ with finite cardinality and D_2, the average number of points within a square of radius ϵ' follow a power law:

$$\overline{nb}(\epsilon') \propto \epsilon'^{D_2} \tag{1}$$

Same can be said for ride requests.

Fractal Dimensionality & Human Mobility Pattern

[Belussi 1998] Given a set of points $\mathbb{P}$ with finite cardinality and D_2, the average number of points within a square of radius ϵ' follow a power law:

$$\overline{nb}(\epsilon') \propto \epsilon'^{D_2} \tag{1}$$

Same can be said for ride requests.

Expected Performance of FTL-CH is strictly better than URand-NH:

$$\mathbb{E}_{\text{FTL-CH}}[R_t] > \mathbb{E}_{\text{URand-NH}}[R_t] \tag{2}$$

Conclusion

1. We provide a formalization of the real-time vehicle placement problem, and draw similarities to known problems like $k-$ server problem.

Conclusion

1. We provide a formalization of the real-time vehicle placement problem, and draw similarities to known problems like $k-$ server problem.

2. Highlight using real data connection between human mobility and chaos theory (fractals).

Conclusion

1. We provide a formalization of the real-time vehicle placement problem, and draw similarities to known problems like $k-$ server problem.

2. Highlight using real data connection between human mobility and chaos theory (fractals).

3. Propose potential online algorithms with guarantees which could reduce rider wait time, and driver idle time.