Data Driven Analysis of the Potentials of Dynamic Ride Pooling

Min Hao Chen
Carnegie Mellon University
Moffett Field, CA, USA
danielmchen@cmu.edu

Abhinav Jauhri
Carnegie Mellon University
Moffett Field, CA, USA
ajauhri@cmu.edu

John Paul Shen
Carnegie Mellon University
Moffett Field, CA, USA
jpshen@cmu.edu

ABSTRACT

This paper focuses on the challenge of dynamically pooling multiple ride requests in real time in order to achieve greater overall efficiency for ride sharing/hailing services. A rigorous formulation of this problem and an efficient pooling method is introduced. This paper adopts a data-driven approach and uses an extensive ride-request data set from the real world to evaluate this method. The experimental results based on the data set for three US cities show that close to 50% of all ride requests can be pooled, and there are significant benefits for both riders and services. Furthermore dynamic ride pooling can potentially yield significant societal benefits in reducing total fuel consumption (by 15%) and alleviating traffic congestion by reducing the total vehicle count (by 30%).

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**; • **Computer systems organization** → **Real-time systems**; • **Applied computing** → *Transportation*;

KEYWORDS

Ride-sharing, pooling

1 INTRODUCTION

In recent years, ride-sharing platforms such as Uber and Lyft have become a significant means of transportation, accounting for nearly two billion rides in 2016 [17]. On a daily basis, Didi with significant penetration in China reported booking four times the number of ride requests than that of the entire US [2]. With such massive scale of operation, ride-sharing service companies have started to focus on ways to better utilize their resources, i.e. drivers and vehicles. In particular, they have begun to offer services that pool multiple ride requests into a single vehicle in real time. Their goals include: lowering the cost for riders; increasing earnings for drivers and the companies; and keeping added travel time for riders within an acceptable range. Such *dynamic ride pooling* is growing and showing significant potential for benefiting ride-sharing services, drivers, and riders.

Dynamic ride pooling can potentially benefit society as well. Human mobility and transportation is a major issue for many large urban areas in the world [5]. As ride-sharing services become ubiquitous, other than providing convenient and affordable transportation on demand, there is the potential of contributing societal benefits. This work takes a data-driven approach to investigate the subject of dynamic ride pooling. This work leverages an extensive ride-request data set (from a ride-sharing service) for three major US cities: San Francisco, New York, and Los Angeles.

Dynamic on-demand ride pooling without advanced knowledge of ride requests is a challenging problem. The pooling decisions must be made in real time for a large number of moving vehicles. Ride requests occur dynamically and can be widely distributed spatially in terms of pickup and drop off locations. The pooling decisions must take into account both temporal and spatial proximity of the ride requests. Increasing the proximity scope for harvesting poolable ride requests may improve the amount of pooling and overall pooling benefits. However this can also lead to increase in travel time penalty for riders. Thus, effective dynamic ride pooling must take into account the benefit vs cost trade offs.

This work makes the following key contributions: 1) We introduce a rigorous formulation of the dynamic ride pooling problem and propose a method for dynamic ride pooling; 2) We identify the *best-performing method/configuration* and apply it to our ride-request data set for the three cities; 3) We present insights distilled from the experimental results on the benefits and costs of the best-performing pooling method/configuration; 4) We show dynamic ride pooling can produce significant societal benefits, in reducing the *total travel distance* and the *total vehicle count* for a city.

2 RELATED WORK

There have been extensive prior works on matching riders. This section does not exhaustively cover all related works; it highlights some of the most relevant works on formulating the problem of matching riders, and prior works focusing on the potential, including societal, benefits of ride-sharing services.

Numerous graph-based approaches have been studied for matching riders. [14, 15] formulate the problem of finding matching riders as a maximal graph matching problem. [8] applies approximation set cover algorithms to find a set of riders to carpool. The problem of finding matching riders is modeled as a network flow problem by [19]. [6] applies evolutionary algorithms to match one driver with multiple riders. Recently, [1, 12] propose an integer linear program formulation to match riders, and show that less than 25% of active cabs can satisfy 99% of the ride requests with an average delay of 2.5 minutes in New York city.

A peer-to-peer multi-hop ride-matching problem is modeled using a binary program by [11]. An on-line ride-sharing system is proposed by [4] using Maximum Cardinality Matching. An interesting approach by [3] considers the LDA algorithm for recommending sharing opportunities. [18] introduce meeting points, within some

IWCTS'17, November 7–10, 2017, Redondo Beach, CA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5491-2/17/11…$15.00
https://doi.org/10.1145/3151547.3151549

distance of riders' source or destination points, to enable sharing of vehicles, and highlight its potential.

[16] highlights that more than 70% of the rides can be pooled when passengers can tolerate a delay of up to five minutes; their analysis is based on 14 million taxi trips also in New York city over a month. [10, 12] highlight that by increasing the number of shared vehicles over a geographical space, the number of people using the shared vehicle increase. [1] quantifies experimentally the trade-off between fleet size, capacity, waiting time, travel delay, and operational costs for low to medium capacity vehicles.

Most of the above studies use data exclusively from taxi rides in dense cities like New York; this constitutes a much smaller sample size than what is used in our work in representing general human mobility patterns. Due to the low pricing and availability, ride-sharing services represent an economically diverse set of riders as compared to traditional cabs. Moreover, most of the prior works require information about pickup and destination locations of ride requests ahead of time, thus they are not practical for real-world implementations where hundreds or even thousands of ride requests need to be handled every second. Our work provides insights about *how* and *when* ride requests occurring in temporal and spatial proximity can be effectively pooled such that high frequency and degree of ride pooling can occur while the increase in overall travel times for riders is kept minimal. In contrast with these previous works, our method can also be implemented without prior knowledge of when and where ride requests occur.

3 RIDE REQUEST PATTERNS

This work is based on real-world ride requests from a ride-sharing service for three major cities in the US, San Francisco, New York, and Los Angeles. The data set contains more than 10 million ride requests for each city. Each ride request is represented by a 5-tuple: 1) request time; 2) pick up time; 3) pick up location (latitude & longitude); 4) drop off time; 5) drop off location (latitude & longitude).

It has been observed that ride requests for a city varies temporally and spatially, and there is strong spatial clustering in densely populated areas even for very short time intervals [7]. We make use of this key observation in formulating our dynamic ride pooling method, and to minimize time overhead of pooling.

4 DYNAMIC RIDE POOLING

The *dynamic ride pooling* problem involves identifying which ride requests and how many ride requests should be bundled into a single vehicle. This involves making decisions on the fly, in real time, in response to spontaneously occurring ride requests. This is done by first identifying a *primary* ride request and use it as the reference for pooling other additional *secondary* ride requests. The pooling of multiple ride requests into a single vehicle is referred to as a plan denoted by $\mathcal{P}$.

Each pooled vehicle is associated with a plan $\mathcal{P}$, and each plan has its associated benefits and costs. With ride pooling, fewer vehicles are needed to service the same amount of total ride requests. Thus, the average cost for servicing a ride request can be reduced. The potential costs for ride pooling include inconvenience to the riders in terms of longer wait time for pick up and/or longer travel time for drop off. Also, to accommodate large number of pooled ride requests in one vehicle, larger sized vehicles may be needed. Hence, dynamic ride pooling requires performing trade offs between the benefits and the costs. The goal is to maximize the benefits while minimizing the costs. Given the set of all ride requests, the task is to find mutually exclusive subsets, or plans of ride requests, such that each plan $\mathcal{P}$ can be pooled into one vehicle.

Effective pooling must take into account both temporal and spatial proximity of ride requests. This means that in searching for requests which can be pooled, one should only consider ride requests that occur very close in time, and with pick up locations that are very near to each other. Also, the drop off locations should not be too far apart. In order to perform real-time and on-demand ride pooling, we need to impose a very small time window as the *temporal proximity constraint*. This means that pooling can only occur for rides that are requested within a small time window of size ϵ_t, typically on the order of few minutes. All rides requested within the same time window t is denoted by $\mathcal{S}_t$. The next time window $t + 1$ immediately follows t with same duration of ϵ_t with its ride requests denoted as $\mathcal{S}_{t+1}$.

4.1 Pooling Method

We propose a *Dynamic Ride Pooling* (DRP) method, that takes advantage of spatially and directionally proximal ride requests. For any ride pooling plan $\mathcal{P}$, all ride requests in the plan must satisfy certain specified proximity constraints. We have already specified the temporal proximity constraint of ϵ_t. There are two aspects to the spatial proximity constraints, one for specifying proximity of all the sources or pick up locations, i.e. between pick up locations of the primary request $loc(\text{'src'}, p)$ and any secondary request j given by $loc(\text{'src'}, j)$. The other aspect specifies proximity of all the destinations or drop off locations, i.e. between $loc(\text{'des'}, p)$ and $loc(\text{'des'}, j)$. The pooling method starts with a reference or primary ride request p; spatial proximity constraints are specified relative to the pick up and drop off locations of p.

The imposition of the proximity constraints involve two phases. The first phase (Restricted) focuses on two spatial proximity regions. The source region is a circle centered at the source of p with radius ϵ_{sr}, and the destination region is another circle centered at the destination of p with radius ϵ_{dr}. Any ride request that starts in the source region of p and ends in the destination region of p satisfies the spatial proximity constraints and becomes a candidate for pooling with p. Hence, given a primary ride request p, and other ride requests within the same time window given by the set $\mathcal{S}_t - \{p\}$, any request $j \in \mathcal{S}_t - \{p\}$ is added to the plan $\mathcal{P}$ if:

(1) $dist(loc(\text{'src'}, p) - loc(\text{'src'}, j)) < \epsilon_{sr}$
(2) $dist(loc(\text{'des'}, p) - loc(\text{'des'}, j)) < \epsilon_{dr}$

where ϵ_{sr} and ϵ_{dr} define the circular source and destination regions centered around pick up and drop off locations of the primary ride request p, respectively; $dist()$ returns the straight line distance between the two points.

The first phase of DRP is rather restrictive spatially and misses additional pooling opportunities. Simply enlarging the circular regions is not the most effective way to increase ride pooling. Instead, we introduce a more efficient way to enlarge the search regions by incorporating directionality in the second phase (Directional). For this phase the proximity region is defined by a trapezoid, containing

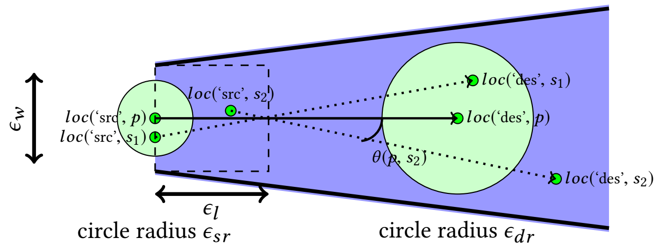


Figure 1: *Dyanmic Ride Pooling (DRP)*: The two circular regions (in light green) are used in the Restricted phase. The rectangle (dashed lines) and open trapezoidal (in blue) are used in the Directional phase. Primary ride request p, and two pooled rides are shown; s_1 and s_2 satisfy conditions for Restricted and Directional phases respectively.

the source and destination locations of p, defined by width ϵ_w (short side of the trapezoid) and an angular parameter ϵ_θ. By imposing this angular parameter as a constraint, more potential ride-pooling candidates can be identified along the direction that matches the trajectory of the primary ride request p. The motivation of using a trapezoid is to bundle more ride requests along the way that do not incur too much of overhead of having to go out of the way to pool more rides for pickup and drop off. The trapezoid has an open side (long side); this means a secondary ride request does not have to end near the destination of the primary ride request, as long as the angle of the secondary request relative to the primary is less than ϵ_θ, it can be included in the plan. This facilitates a much larger proximity region without incurring significant travel time overhead to pick up and drop off additional riders. From a given primary ride request p, and any other request $j \in \mathcal{S}_t - \{p\}$ in the same time window t, j is added to the plan $\mathcal{P}$ if:

(1) $loc(\text{'src'}, j) \in rect(\epsilon_w,\ \epsilon_l,\ loc(\text{'src'}, p))$
(2) $|\theta(p, j)| < \epsilon_\theta$

where $rect(\cdot,\cdot,\cdot)$ defines a rectangular region with width ϵ_w, and length ϵ_l (ϵ_l is the fraction of straight line distance between the pick up and drop off locations of p which can be covered in ϵ_t minutes, thus varies for every p); $|\theta(p, j)|$ defines the angular difference between trajectories of ride requests p & j; ϵ_θ defines the maximum angular difference allowed between any j & p.

Dynamic Ride Pooling method is illustrated in Figure 1 with its proximity regions specified by both phases. The goal for having these two phases is that the Restricted phase would first identify and pool all requests that are highly similar spatiotemporally. The Directional phase would then take advantage of the direction that a primary ride is heading and seek for more pooling candidates.

4.2 Pooling Metrics

To quantify the benefits and costs of a plan, we define the following four metrics. The first two metrics measure the benefits achievable by a pooling method. The third metric concisely captures the percentage of total ride requests that can be pooled by a pooling method. The fourth metric is a per-rider cost metric measuring the inconvenience experienced by each rider participating in pooling.

4.2.1 Total Travel Distance Reduction. The total travel distance saved by a plan is calculated by taking the difference between the sum of individual travel distances if no pooling occurred and the total travel distance of the pooled plan (accounting for picking up and dropping off all the riders):

$$dist\_red(\mathcal{P}) = \sum_{i \in \mathcal{P}} travel\_dist(i) - travel\_dist(\mathcal{P}) \tag{1}$$

The total travel distance reduced by pooling as a percentage over all the ride requests for an entire week is defined as:

$$total\_dist\_red\_\% = \frac{\sum_{\mathcal{P}} dist\_red(\mathcal{P})}{\sum_{j=1}^{m} travel\_dist(j)} \tag{2}$$

where m is count of all ride requests in a week.

4.2.2 Total Vehicle Count Reduction. Number of vehicles reduced by a plan is:

$$veh\_red(\mathcal{P}) = |\mathcal{P}| - 1 \tag{3}$$

The total number of vehicle count reduced as a percentage over all the ride requests for an entire week is defined as:

$$total\_veh\_red\_\% = \frac{\sum_{\mathcal{P}} veh\_red(\mathcal{P})}{m} \tag{4}$$

4.2.3 Average Poolability. Poolability is defined as the percentage of ride requests that are *poolable* over all ride requests in a time window t of ϵ_t minutes [7]. If a ride is *poolable*, it will belong to a plan $\mathcal{P}$ along with the primary ride request. Poolability for a given time window can be calculated as:

$$poolability(\mathcal{P}_t, \mathcal{S}_t) = \frac{\sum_{\mathcal{P} \in \mathcal{P}_t} |\mathcal{P}|}{|\mathcal{S}_t|} \tag{5}$$

where $\mathcal{P}_t$ is the set of all the plans for the time window t.
Average poolability over all time windows is given by:

$$avg\_poolability = mean(\sum_t poolability(\mathcal{P}_t, \mathcal{S}_t)) \tag{6}$$

4.2.4 Average Trip Time Penalty. We model the average trip time penalty to be the delayed *arrival time at destination* for a rider when participating in a pooling plan. This penalty for any request i can be calculated as:

$$arrival\_penalty(i) = timestamp(\text{'des\_p'}, i) - timestamp(\text{'des'}, i) \tag{7}$$

where 'des' denotes the event of arriving at destination if i were *not* pooled and 'des_p' denotes the same but pooled.

It is important to note that $arrival\_penalty(i)$ may have a negative value. This usually happens when a plan is able to pick up a request earlier than its original pick up time i.e. $timestamp(\text{'src\_p'}, i)$ $< timestamp(\text{'src'}, i)$. Also, Equation 7 only makes sense when i is poolable; non-poolable rides are not considered when calculating average trip time penalty. The overall average trip time penalty is computed by taking the mean across all *pooled* riders:

$$avg\_penalty = \frac{\sum_{\mathcal{P}} \sum_{i \in \mathcal{P}} arrival\_penalty(i)}{\sum_{\mathcal{P}} |\mathcal{P}|} \tag{8}$$

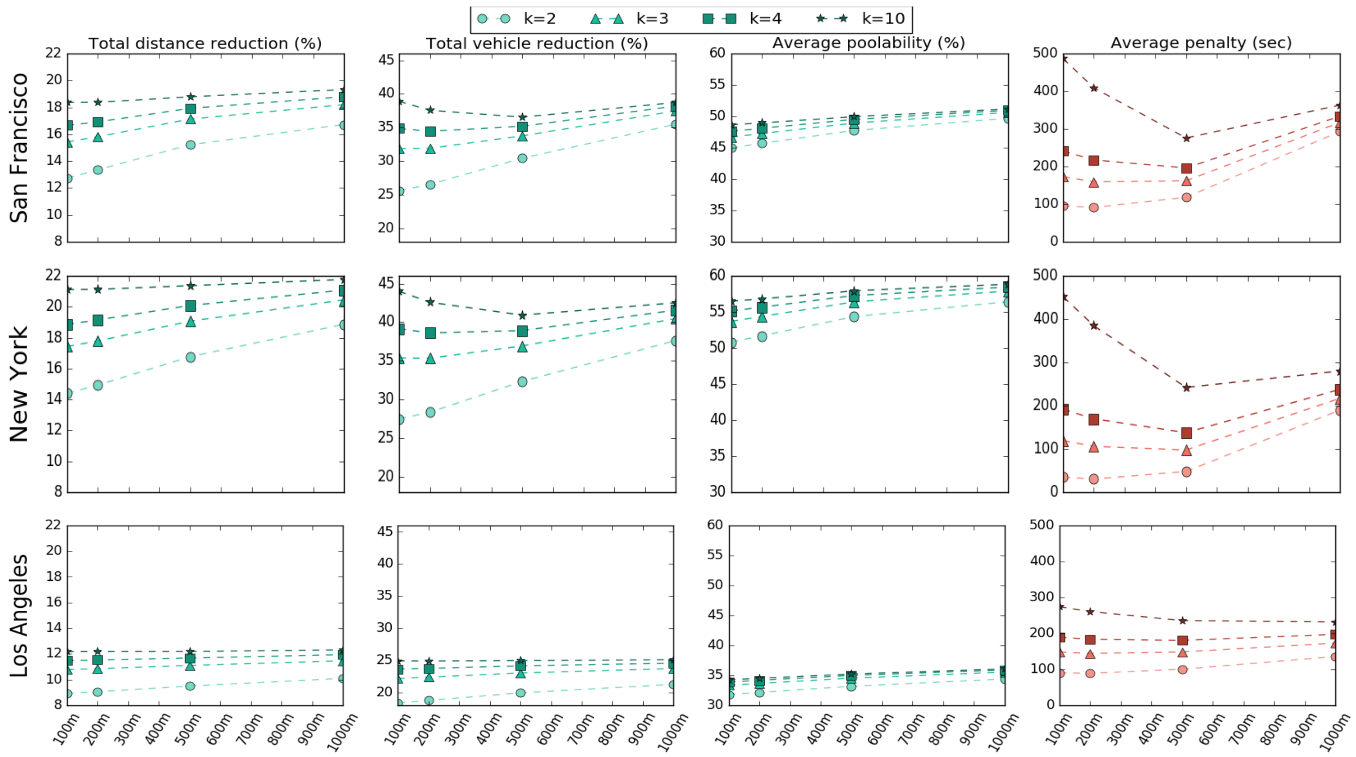


Figure 2: DRP with varying e_{sr}, k for three benefits and one cost metrics for San Francisco, New York, and Los Angeles . $\epsilon_{dr} = 1000m, \epsilon_w = 2000m, \epsilon_\theta = 20°$ are kept constant.

5 EXPERIMENTAL RESULTS & ANALYSIS

5.1 Experimental Setup

In this paper we set the *temporal proximity constraint* $\epsilon_t = 5$ minutes. Ride pooling can only occur among ride requests occurring in the same 5-minute time window. Enlarging this window will certainly increase the opportunity for pooling, but can potentially incur unacceptable added wait time for pick up.

We partition our week-long ride request data set into 5-minute intervals or time windows. All the ride requests in the same time window are processed one at a time in chronological order. Each request is considered as the primary request seeking for other pooling candidates within the same time window. Once a plan, $\mathcal{P}$, is established, the decision of the order of pick up and drop off for each passenger and the exact routes to take in between these locations are as follows:

(1) The order of pick up follows the order of request times.
(2) The order of drop off is based on the distances away from the last pick up location, from nearest to farthest.
(3) Routes to take in between any two geographical points are queried from GraphHopper, an open-source route navigation and planning tool [9] that make use of the maps and traffic speeds using PBF [1] files by OpenStreetMap [13].

GraphHopper is also useful in calculating the travel distance and arrival time of each request if no pooling occurred. This is necessary for computing both the distance saved and extra distance incurred when a request participates in a pooling plan.

[1]PBF files are available at http://download.geofabrik.de/

5.2 Experimental Goals

The goal of our experiments based on the real world data set is to help answer the following questions: 1) What are the best values to use for the proximity constraint parameters, ϵ's? Ideally ϵ's should be kept small to better exploit spatio-temporal proximity, but expanding these values can capture more pooling candidates; 2) What is the best value of the pooling degree (i.e. maximum number of ride requests that can be pooled in one vehicle), k? 3) How do all these factors correlate and vary in different cities, on different days of the week, at different times of the day? 4) What is the best combination of these parameters that gives the best trade off between benefits and costs for dynamic ride pooling?

5.3 Experimental Results

In search for the best configuration for our pooling method, we need to explore numerous combinations of parameter values. We first narrow down the search space by appropriately fixing some parameters as constant values without compromising our search for the best pooling method configuration. We fixed $\epsilon_w = 2000m$ which maximizes the size of the pooling regions for the Directional phase without incurring unacceptable travel time penalty. Smaller values unnecessarily limit opportunities for pooling. The two parameters $\epsilon_{dr}, \epsilon_\theta$ are somewhat correlated; we fixed them at $1000m$, and $20°$

Metric	*San Francisco*	*New York*	*Los Angeles*	*Mean across cities*
Total Travel Distance Reduction (%)	17.13	19.06	11.01	15.76
Total Vehicle Count Reduction (%)	33.76	36.93	23.03	31.23
Mean Poolability (%)	48.94	56.39	34.52	46.61
Mean Travel Time Penalty (sec)	162.12	97.55	148.17	135.94

Table 1: Summary of benefits and costs. Parameters used $\epsilon_t = 5$ mins., $\epsilon_{sr} = 500m$, $\epsilon_{dr} = 1000m$, $\epsilon_w = 2000m$, $\epsilon_\theta = 20°$, $k = 3$

respectively, which provide significant poolability while keeping travel time penalty reasonable. In our experiments we vary the values for the remaining parameters in our search for the best configuration; we vary ϵ_{sr} from 100m to 1000m, and k from 2 to 10, for all three cities of San Francisco, New York, and Los Angeles.

Looking at the results in Figure 2, we see both similarities and differences across the three cities. All the benefit metrics increase with increasing ϵ_{sr} (size of source region) and k (maximum pooling degree). For k there is a strong diminishing return beyond $k = 3$. There is also a slight diminishing return beyond $\epsilon_{sr} = 500m$. The travel time penalty curves for all three cities exhibit similar patterns, with an interesting inflection point around $\epsilon_{sr} = 500m$. The "dip" is more apparent as k increases. This has to do with the balancing of pooling attributed to Restricted versus Directional phases. When ϵ_{sr} is small (100m) most of the pooling comes from the Directional phase which yields great benefits but also incurs heavier penalty. As ϵ_{sr} increases, Restricted can start to capture all the spatially proximal requests while keeping the overall average penalty down. This indicates that $\epsilon_{sr} = 500m$ yields the best balance for DPR where strengths of both phases are leveraged.

Another important observation derived from Figure 2 is about the conflicting nature of the two phases of DRP. The vehicle reduction plot for NYC has a negative slope (at $k = 10$) for values of ϵ_{sr} in the range $[100, 500]m$ although it has an inflection point at $\epsilon_{sr} \approx 500m$ characterizing the greedy nature of the Restricted phase. Given a high value of k, the Restricted phase consumes ride requests which could otherwise be used as primary requests in the Directional phase to pool many other riders and thereby lowering the overall vehicle reduction percentage. Eventually, vehicle reduction percentage for higher values of ϵ_{sr} increases but never more than its initial value at $\epsilon_{sr} = 100m$, when Restricted is least greedy and Directional maximizes its potential.

There are also significant differences across the three cities. In New York, 50-60% of all ride requests are poolable, while in Los Angeles only 30-35%. San Francisco is between these two extremes. We see very strong correlation between poolability and the two benefits of total travel distance reduction and total vehicle count reduction. Again, New York has much higher levels for both benefits as compared to Los Angeles. What makes New York so ride pooling friendly and why is it so much more difficult to do ride pooling in Los Angeles? We conjecture this is due to the sprawling nature of Los Angeles. Also, why is increasing ϵ_{sr} not a very effective way of increasing the benefits in Los Angeles comparing to San Francisco and New York? Again, this can be due to the sprawling nature of Los Angeles, unlike both San Francisco and New York which have dominant commuting corridors connecting densely populated regions due to the topology of the greater metro areas of these two cities.

Given the observed results, we conclude that the best set of pooling parameters are $\epsilon_{sr} = 500m$, $\epsilon_w = 2000m$, $k = 3$. Across the three cities, this configuration yields on average 15.76% of total travel distance saved and 31.23% of total number of vehicles reduced, while incurring an average trip time penalty of only 135.94 seconds (see Table 1). This translates to roughly one out of every three vehicles operated by these ride-sharing services can be removed from the roads if riders are willing to pool and accept a travel time penalty of slightly over two minutes! These results are quite encouraging and imply that dynamic ride pooling is a must for densely populated urban areas.

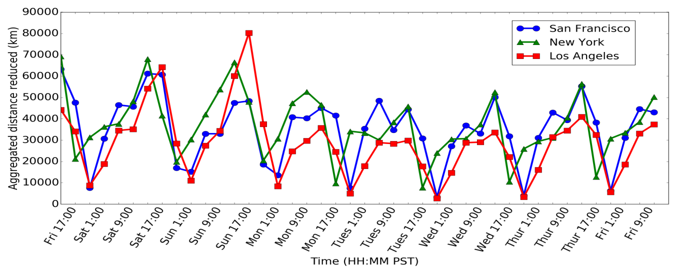
5.4 Experimental Analysis

Temporal Patterns: There are interesting weekly temporal patterns for all four pooling metrics. Figures 3a, 3b, 3c, and 3d, plot the weekly temporal patterns of the four metrics for the three cities, using the best pooling configuration. The plots start at 5:00 P.M. PST (12 A.M. UTC) on a Friday and each data point is an aggregated value over a 4-hour period (hence a total of 42 data points for a week). These plots share similar weekly patterns and correlate well with the weekly pattern shown for ride requests in [7]. During times of higher ride request volumes there is greater poolability and potential benefits. For instance, poolability can reach up to 60% on weekend afternoons. Figure 3d shows the average penalty is higher during low ride request volumes. Average penalty is highest for San Francisco/Bay Area, most likely due to its low population density among the three cities. The temporal pattern can be useful in guiding the implementation of ride pooling algorithms that can be temporally adaptive.

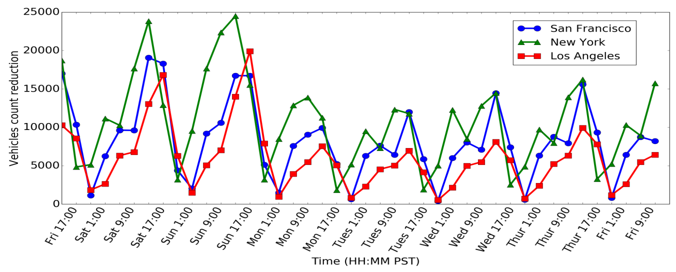
Societal Benefits: Dynamic ride pooling has potential for significant societal benefits. Reducing total travel distance by 16% leads to proportional reduction in fuel consumption and CO_2 emission. Taking 31% of vehicles off the roads can substantially reduce traffic congestion, especially during rush hours. Across these three cities, on average almost 50% of the ride requests are dynamically poolable. Our results indicate that as the ride request volume increases there is also greater potential benefit for ride pooling. Ride-sharing services can reduce overall operation cost and improve overall efficiency. This leads to lower cost for riders. There is the potential for a simultaneous three-way win that benefits ride-sharing companies, mobile population, and densely populated cities.

6 CONCLUSION

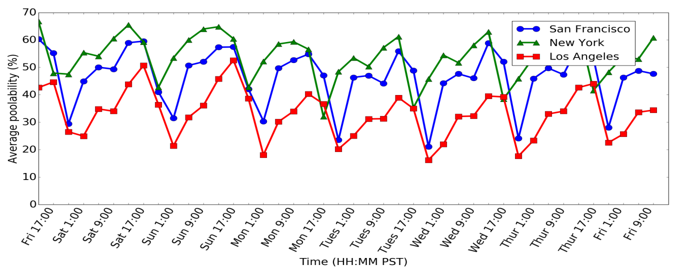
In this work, we take the data-driven approach to systematically study the problem of dynamic ride pooling. We derive a rigorous formulation of the problem, propose a pooling method and its key parameters, explore the parameter space for the best pooling



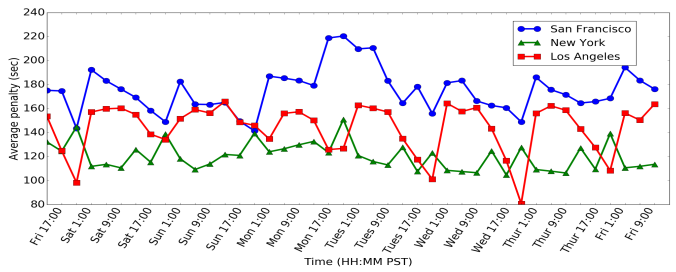
(a) Travel distance reduction.



(b) Vehicle count reduction.



(c) Average poolability (%).



(d) Average penalty (sec).

Figure 3: Temporal pattern of the four pooling metrics over a week for all three cities.

configuration based on real world ride request data from three cities, show the potential benefits of ride pooling, and highlight insights gained from analyzing the results.

Future Work: The *DRP* method presented is restrictive in its temporal constraint; pooling candidates are considered only if they fall within the same time window, ϵ_t. Requests outside this time window are currently not considered for pooling. Thus, on-demand en-route pooling may be explored with emphasis to further improve vehicle capacity with minimal overhead to riders. Note this differs from works discussed in Section 2 as pooling decisions need to be made on-the-fly with no prior knowledge of when and where ride requests are coming from. Another follow-on step is to focus on the implementation of effective and scalable dynamic ride pooling algorithms, and the trial deployment of such algorithms in a few cities for in-situ experimentation.

Acknowledgements: This project is funded in part by Carnegie Mellon University's National USDOT University Transportation Center for Mobility, Mobility21, which is sponsored by the US Department of Transportation.

REFERENCES

[1] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences* (2017), 201611675.

[2] Mark Bergen. 2016. Didi Booking China. Recode. (2016). https://www.recode.net/2016/6/1/11835620/didi-booking-china-apple

[3] Nicola Bicocchi and Marco Mamei. 2014. Investigating ride sharing opportunities through mobility data analysis. *Pervasive and Mobile Computing* 14 (2014), 83–94.

[4] Blerim Cici, Athina Markopoulou, and Nikolaos Laoutaris. 2015. Designing an on-line ride-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 60.

[5] Regina R. Clewlow and Gouri Shankar Mishra. 2017. Disruptive Transportation: The Adoption, Utilization, and Impacts of Ride-Hailing in the United States. (2017). https://itspubs.ucdavis.edu/wp-content/themes/ucdavis/pubs/download_pdf.php?id=2752

[6] Wesam Mohamed Herbawi and Michael Weber. 2012. A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation.* ACM, 385–392.

[7] Abhinav Jauhri, Brian Foo, Jerome Berclaz, Chih Chi Hu, Radek Grzeszczuk, Vasu Parameswaran, and John Paul Shen. 2017. Space-Time Graph Modeling of Ride Requests Based on Real-World Data. *arXiv preprint arXiv:1701.06635* (2017).

[8] Ece Kamar and Eric Horvitz. 2009. Collaboration and Shared Plans in the Open World: Studies of Ridesharing.

[9] Peter Karich and S Schröder. 2014. Graphhopper. *http://www. graphhopper. com, last accessed* 4, 2 (2014), 15.

[10] Matteo Mallus, Giuseppe Colistra, Luigi Atzori, Maurizio Murroni, and Virginia Pilloni. 2017. A persuasive real-time carpooling service in a smart city: A case-study to measure the advantages in urban area. In *Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on.* IEEE, 300–307.

[11] Neda Masoud and R Jayakrishnan. 2015. A decomposition Algorithm to solve the multi-hop peer-to-peer ride-matching problem. In *Transportation Research Board 94th Annual Meeting.*

[12] Joe Naoum-Sawaya, Randy Cogill, Bissan Ghaddar, Shravan Sajja, Robert Shorten, Nicole Taheri, Pierpaolo Tommasi, Rudi Verago, and Fabian Wirth. 2015. Stochastic optimization approach for the car placement problem in ridesharing systems. *Transportation Research Part B: Methodological* 80 (2015), 173–184.

[13] OpenStreetMap contributors. 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org. (2017).

[14] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven Strogatz, and Carlo Ratti. 2013. Taxi pooling in New York City: a network-based approach to social sharing problems. (2013).

[15] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, and Carlo Ratti. 2014. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences* 111, 37 (2014), 13290–13294.

[16] Erez Shmueli, Itzik Mazeh, Laura Radaelli, Alex Sandy Pentland, and Yaniv Altshuler. 2015. Ride sharing: a network perspective. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction.* Springer, 434–439.

[17] Brian Solomon. 2016. Uber Just Completed Its Two Billionth Ride. Forbes. (2016). https://www.forbes.com/sites/briansolomon/2016/07/18/uber-just-completed-its-two-billionth-ride/#9e4db5c52242

[18] Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. 2015. The benefits of meeting points in ride-sharing systems. *Transportation Research Part B: Methodological* 82 (2015), 36–53.

[19] Shangyao Yan, Chun-Ying Chen, and Chuan-Che Wu. 2012. Solution methods for the taxi pooling problem. *Transportation* 39, 3 (2012), 723–748.