2016 17th IEEE International Conference on Mobile Data Management

A Model of Multimodal Ridesharing and its Analysis

Jane Lin, Sandeep Sasidharan
Department of Civil and Materials Engineering
University of Illinois at Chicago
Chicago, IL, USA
janelin@uic.edu, ssasid3@uic.edu

Shuo Ma
Google, Inc.
Mountain View, CA, USA
shuoma86@gmail.com

Ouri Wolfson
Department of Computer Science
University of Illinois at Chicago
Chicago, IL, USA
wolfson@uic.edu

***Abstract*—Getting a taxi in highly congested areas (e.g. airports, conferences) is both time consuming and expensive. Chicago Tribune reports that wait at Chicago O'Hare International airport for taxi cabs can be as long as 45 minutes [1]. In this paper we propose RSVP, a ridesharing system that uses walking and virtual pools. RSVP is aimed mainly for transportation hubs, such as airports, railway stations, etc. In these places, a steady stream of passengers arrive via some public transport mode, say train, and then depart to different destinations. We introduce a model for ride-sharing that involves walking, devise ridesharing algorithms, and evaluate them using a database that recorded real taxi trips in NYC.**

***Keywords*-virtual pool; ride-sharing; Slugging-Multiple-drop-off; match making; shareability**

I. INTRODUCTION

Getting a taxi in highly congested areas (e.g. airports, conferences) is both time consuming and expensive. Chicago Tribune reports that wait at in Chicago O'Hare International airport for taxi cabs can be as long as 45 minutes [1]. While the emergence of novel Transportation Network Companies (e.g. Uber) has helped increase the supply of drivers during peak times, they have done little to reduce congestion in hubs such as airports, major stations, and stadiums. Creative on-demand transit service is needed more than ever to capture the benefits of the smartphone and health-consciousness revolutions.

In this paper we propose RSVP (Ride Sharing by Virtual Pools), a ridesharing system based on walking and virtual-pools, aimed mainly at transportation hubs. The RSVP scheme combines in a unique way three existing mechanisms: virtual queues, slugging (i.e. walking for the purpose of ride-sharing), and multiple-drop-off ride-sharing. Assume a designated taxi ride-sharing pickup location at a hub H (e.g., airport, train station), with virtual ride-sharing demand pools associated with time-intervals. For example, a virtual pool of ride-sharing passengers will be picked up between 11:00am and 11:05am, followed by another pool to be picked up between 11:05am and 11:10am, etc. Initially, each pool consists of a number n of trips, which after merging will be reduced to m merged (or ride-sharing) trips. m should not be allowed to grow beyond the number of taxi-pickups that can occur at a specific curb location during the time-interval associated with the pool (e.g. 5 minutes).

Upon arriving at the hub H using another mode of transportation (e.g. a plane), a passenger expresses interest in taxi-ridesharing by specifying her trip and electronically enrolling it into a pool, e.g. the 11:00-11:05 pool. The trip specification indicates the destination, and the bounds on walking and delay times that the passenger is willing to tolerate in order to enable ride-sharing. It is envisioned that the passenger will register the trip in the earliest pool that allows enough time to walk from the arrival location to the ride-sharing pickup location. For example, if the passenger deplanes at 8:00am, and it takes 10 minutes to walk from her arrival gate to the ride-sharing pickup location, then the passenger will enroll her trip in the 8:10–8:15am pool. A pool closes, say, one minute before its start-time, or when it is full, whichever occurs first.

After a pool P closes, a MatchMaking (MM) system is run on the set of n trips in the pool, creating a smaller set of m merged trips, each of which will be served by a single taxi. Each merged trip may consist of multiple drop-off points of the ride-sharing passengers. The selection of drop-off points must satisfy the walking and delay time constraints specified by the passengers. Because the pick-up and drop-off points in RSVP may differ from the passenger's actual origin and destination respectively within a tolerable walking distance, RSVP incorporates slugging [2].

The proposed scheme benefits travelers, businesses, and municipalities. Travelers can check in remotely, thus are freed from standing in a physical line, and can save money by ride-sharing. Businesses benefit from travelers free to spend money instead of standing in line. Municipalities benefit from reduced vehicle-miles-traveled, congestion, and emissions.

Our proposed scheme differs from existing taxi ridesharing studies in two aspects. First it applies the virtual queue concept to create ride-sharing pools and efficiently manage the ride-sharing demand. Second it considers *SLuggIng-Multiple-drop-off (SLIM)*, a hybrid form of ride-sharing that combines slugging[2] and multiple-drop-off ride-sharing [3],[4],[5],[6]. In this paper we formally prove that this increases the ride-sharing opportunities.

In summary, the contributions of this paper are:

- We formalize the SLuggIng-Multiple-dropoffs (SLIM) ridesharing problem and mathematically prove that some trips are shareable only if they allow slugging.

- We propose efficient algorithms, including a performance-improvement technique based on Euclidean filtering, for producing a SLIM ride-sharing plan on a virtual pool.
- We evaluate the scheme with a database of real taxi trips in NYC, and demonstrate that it produces savings of 25-40% in terms of the total number of trips.
- We quantify the benefits of adding walking to multiple-drop-off ride-sharing.

The reminder of this paper is organized as follows. In Section II, we review relevant work. We introduce the model of multimodal ridesharing Section III, and present the MM system and algorithms the virtual queueing system in Sec. VI. We evaluate it in Section V. In Sec. VI we conclude and discuss future work.

II. Related Works

There have been several studies on wait line management for taxi cabs [7],[8] and attempts to enhance the taxi cab operations [9],[10] at various service stations. However, these techniques fall short at eliminating physical queues at the taxi cab service stations. Taxi cab demand prediction engines [8] and decision making systems [10] may help in better taxi cab operations, but do not guarantee service on demand without entering a physical queue.

There has been extensive research on traditional ridesharing, where driving is the only mode of transportation. Detailed overviews of this research can be found in surveys on vehicle routing problem [11], and ridesharing [12]. Few works have studied multimodal ridesharing where other mode of transport (especially walking or biking) are allowed. In [2] the authors studied the slugging form of ridesharing, where passengers walk to the origin of the driver to get on, then get off at the destination of the driver, and finally walk back to their original destination. The challenge there is to assign the role of driver and passenger to ridesharing participants, and group passengers to ridesharing plans. Therefore, the problem of choosing pickup/drop-off points for passengers is not tackled in [2]. Sester et.al. studied ridesharing with walking in a setting where the role of drivers and passengers are known as input, and each driver is assigned with passengers with the same destination [13][14]. The main problems in both [13], [14] is to determine a rendezvous point for each passenger to be picked up by the assigned driver. In [13] each driver is matched with only one driver, whereas a driver is matched with multiple passengers using Integer Linear Programming (ILP) in [14]; and subsequently the passenger pickup order is determined for each driver. ILP is NP-hard and thus not applicable to large problem instances. Neither [13] nor [14] provides a formal model in which to determine whether trips are shareable.

References [15],[16] also studied the benefit of meeting points (i.e. middle points for pickup/dropoff) for ridesharing systems. In [15] the match is between a single driver and a single rider. And the [15] model provides constraints in terms of time windows instead of maximum walking time. Reference [16] provides a solution in which sources, destinations, and intermediate points are in the Euclidean plane rather than networks. From a computational perspective, similar to our approach here, both [15], [16] devise and apply heuristics to reduce the search space for meeting points thus speed-up computation. However, note that the heuristics used in this paper prune the search space without compromising optimality of the solution. Furthermore, in [13], [14], [15], [16] each driver has its own destination. In contrast, in our model a driver does not have an individual destination.

This work is also relevant to existing work on taxi ridesharing [4],[6]. Those works differ from our paper in: 1) they do not consider walking as a second mode; 2) they do not build a ridesharing plan from a time windowed pool of requests. That is, in those papers, whenever a ride request arrives, all taxis are considered for matching the new query. Thus, they focus on quickly finding candidate taxis for ridesharing based on spatial indexing. In terms of savings, [4] reports about 25%~35% more taxi requests can be served if ridesharing with at most two passengers is allowed (depending on taxi shortage, modeled by parameter Δ). This is similar to our results here.

Similar to the work here, both [2],[5] consider matching trips in a small time window. Reference [2] does not consider multiple drop-offs, and as a result, it requires a higher similarity between trips that can be merged. Unlike this paper, where trips are bounded within New York City, all trips in [5] are bounded within Manhattan. Thus all the destinations are in a denser area than NYC as a whole, thus ridesharing is more probable. In terms of savings, [5] reports a 50% reduction in # of trips with ridesharing allowing at most one more passenger and maximum delay of 5 minutes, using a 3 minute pool size. In contrast, our study finds a 28% ~ 35% (depending on the pool size) reduction in the number of trips, with ridesharing allowed between at most two trips, and maximum 10% travel time delay. The difference between the savings in the two papers are attribute to multiple factors: 1) aforementioned denser destinations in [5]; 2) requests with different origin locations are merged as well in [5]; 3) the impact of walking on the total travel time delay. Furthermore, [5] performs an offline analysis and does not address issues of real-time algorithm efficiency.

While the hybrid walking-and-driving mode in SLIM ridesharing provide flexibility for ridesharing opportunities, it also greatly complicates the ridesharing algorithm, especially for pairwise shareability determination (see Sec. III for more details), the process of determining whether or not two trips are sharable (i.e. mergeable to form one ride-sharing trip). Since in SLIM passengers can be dropped off at intersections away from their respective final destinations, the search space for ridesharing paths is dramatically expanded. For example, given a walking time of five minutes, we find that a destination in New York can have 20~30 candidate drop-off points on average. There have been some works on calculating shortest path for multimodal networks, especially for transit networks [17][18], however, these do not consider ride-sharing. In this paper we describe the shareability determination procedure only for trip pairs. It can be extended to combining more than two trips. In this case, after the ride-sharing plan is obtained, a variant of the Traveling Salesman Problem (TSP) with 3 or more stops needs to be solved to determine the rote of each vehicle. Existing TSP solvers [19][20] obtain the optimal solution for large instances of TSP problems, e.g. the Concorde TSP solver can solve optimally an instance consisting of more than eighty-five thousands stops.

III. The model of trips and their shareability

In this section we first define the multimodal road network (A), then trips and their constraints, and then shareability of trips that satisfy those constraints (C).

A. The Road Network and Multimodal Paths

A *road network* is a directed graph; the vertices are the intersections of the roads, and the edges are the road segments connecting the intersections. Assume that there are n vertices in the road network, denoted v_i, where $i = 1,2,\dots,n$, and let edge e_{ij} be the edge from vertex v_i to vertex v_j. Each edge e in the network has a length $L(e)$. We assume that there is a walking speed which is the same for all edges (e.g. 3 mi/hr) and is denoted WS. The *walking time* of e, denoted $WT(e)$, is $L(e)/WS$. Additionally, each edge e has a maximum driving speed $mDS(e)$, e.g. 60mi/hr on a highway edge. To compute the drive time on an edge e we use a *congestion fraction* denoted cf, where $0<cf<1$. This is a fraction used to compute the driving-time on each edge e by assuming that the driving speed on e is $mDS(e)*cf$. In other words, we assume that cf is the same for all edges. In practice, cf is determined by the time of day, e.g., at rush hour all maximum speeds are cut in half. Of course, this fraction can be adapted to the type of road, but we ignore this refinement here.

Thus the *driving time* on an edge e, denoted $DT(e)$, is $L(e)/(mDS(e)*cf)$. Intuitively, $DT(e)$ is the time it takes to traverse the edge at a speed reflected by the congestion cf; if driving (in the direction of the edge) is not allowed, then the speed is 0 and the driving time is infinity.

Consequently, for every path p in the road network, the *walking time* on p, denoted $WT(p)$, is the sum of the walking times of edges of p, i.e., $WT(p) = \sum_{\forall e_{ij} \in p} WT(e_{ij})$; the *driving time* on p denoted $DT(p)$ is the sum of the driving times of edges of p, i.e., $DT(p) = \sum_{\forall e_{ij} \in p} DT(e_{ij})$. In this paper we consider paths that are unimodal, i.e. consist of a single mode, either walking or driving. Consequently, the time of a path p is either its walking time or its driving time. If p is the shortest (in terms of walking- or driving-time) path between two vertices v and w, then $WT(p)$ and $DT(p)$ are also denoted $WT(v,w)$ and $DT(v,w)$ respectively. The pickup intersection is called the *hub*, denoted by H. For a vertex v, for conciseness we denote by $SP(v)$ the time $DT(H,v)$, i.e. the drive-time on the fastest-drive path from H to v.

B. Trips and Their Constraints

A trip A is a triplet: <*destination-address* $dest(A)$, *number-of-travelers-in-party, constraints*>. We assume that $dest(A)$ is a vertex, i.e. intersection, and a trip starts at time 0. Denote $SP(dest(A))$ by $SP(A)$. Namely, $SP(A)$ is the drive-time on the fastest-drive path from H to $dest(A)$. The constraints are:

(1) Maximum walking time, denoted $W(A)$, from the drop-off point, denoted $d(A)$, to the final destination $dest(A)$, and
(2) Maximum delay (including the walking time from $d(A)$ to $dest(A)$) denoted $D(A)$. In other words, $D(A)$ is the maximum difference between the total travel time to $dest(A)$ in a ride-share (including driving and walking), denoted $TT(A)$, and $SP(dest(A))$; i.e., $TT(A)-SP(A) \le D(A)$.

The number of travelers is used in match-making. For example, two trips, each of which has two travelers, cannot be combined in a taxi with 3 passenger seats.

C. Shareability of Trips

A trip pair (A,B) is *shareable with A first* if there exist:

(1) a driving path $dp(A,B)$ starting at H, and having two dropoff vertices, $d(A)$ and $d(B)$, where $d(B)$ is the last vertex of $dp(A,B)$, (see Fig.1), and
(2) at most two walking paths $wp(A)$ and $wp(B)$, from $d(A)$ to $dest(A)$ and from $d(B)$ to $dest(B)$, respectively,

that satisfy the following 2 conditions:

(1) If $d(A)$ is the same vertex as $dest(A)$, then $wp(A)$ is empty (this means that the dropoff point of A is its destination); otherwise there is a walking path $wp(A)$ from $d(A)$ to $dest(A)$ that satisfies the following conditions:

$$WT(wp(A)) \le W(A) \qquad (1)$$

$$DT(q) + WT(wp(A)) \le SP(A) + D(A) \qquad (2)$$

Equation (1) says the walking time on the path $wp(A)$ is no greater than the maximum walking time limit on A, i.e., $W(A)$. In (2), q is the prefix of the path $dp(A,B)$ from H to $d(A)$. Then (2) indicates that the total travel time from H to $dest(A)$ is no greater than the sum of the shortest path from H to $dest(A)$ and the maximum tolerable delay. This, to comply with constraint (2) of the trip definition because

$$DT(q) + WT(wp(A)) = TT(A) \qquad (3)$$

(2) Similarly, if $d(B)$ is the same vertex as $dest(B)$, then $wp(B)$ is empty (this means that the dropoff point of B is its destination) and $DT(dp(A,B)) - SP(B) \le D(B)$; otherwise there is a walking path $wp(B)$ from $d(B)$ to $dest(B)$ that satisfies the following conditions:

$$WT(wp(B)) \le W(B) \qquad (4)$$

$$DT(dp(A,B)) + WT(wp(B)) \le SP(B) + D(B) \qquad (5)$$

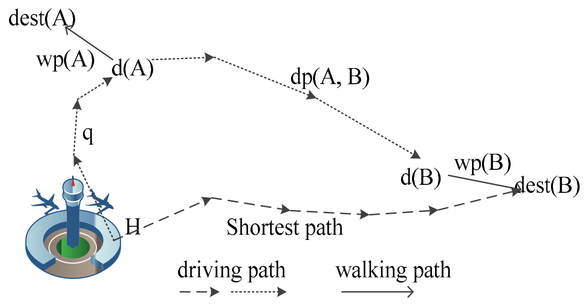

Fig 1: Illustration of shareability of trip pair (A,B)

Trip pair (A,B) is *shareable* if it is shareable with A first or with B first.

The following proposition indicates that adding walking times enriches ride-sharing possibilities.

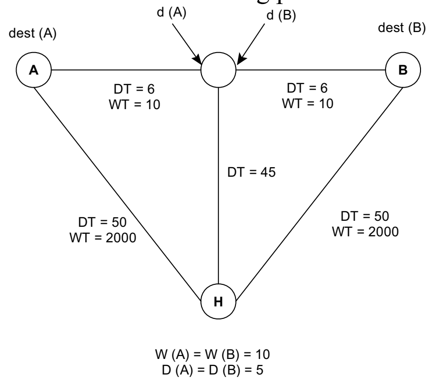

Fig 2: Trips A and B that are shareable if walking is allowed, but not otherwise.

Proposition 1: There exist trips A and B that are shareable if their maximum walking times, $W(A)$ and $W(B)$, are greater than 0, but not otherwise. This is true even if the walking time is slower than the driving time for each edge.

Proof: Consider the road network of Fig. 2, giving the driving time and walking time on each edge. And consider trips A and B starting at H with maximum delays $D(A)=D(B)=5$ and destinations $dest(A)$ and $dest(B)$ respectively. If the maximum walking times are $W(A)=W(B)=10$, then the two trips are shareable with either A first or B first. In either case, both travelers are driven to vertex $d(A)=d(B)$ and are dropped off there, from which they walk, each to their respective destination. The total shared trip time for A is the driving time, 45, plus the walking time, 10, i.e. 55 in total. Since the driving time directly from H to $dest(A)$ along the shortest path is 50, the maximum walking and delay constraints are satisfied for A. And similarly for B.

Now, it is easy to see that if the maximum delays are kept at 5, but the maximum walking times are reduced to 0 for both trips, then A and B are not shareable with A first, nor with B first. The reason is that the drive from H to $dest(A)$ is 50, and from $dest(A)$ to $dest(B)$ is 12, exceeding the maximum delay for B. Thus the trips are not shareable with A first. Similarly for B first.

IV. The MatchMaking (MM) system

In this section we describe the MM system. We first give an overview of the approach (A), then devise the PST algorithm that produces the shareability graph and analyze its complexity (B); finally we discuss Euclidean filtering, a step executed before a pool of trips is fed into the PST algorithm to eliminate in constant time pairs of trips that are not shareable (C).

A. The Approach

After a pool P closes, a MatchMaking (MM) system is run on the set of n trips in the pool, creating a smaller set of m merged trips, each of which will be served by a taxi. Obviously, among the m trips there will be some that have not been merged. This will be the case for a trip A in which the constraints do not allow its merging with any other trip. For example, if all the trips in A's pool allow a delay of at most 5 minutes, and any other destination in the same pool is at least 10 miles away from the shortest path to A's destination, then A cannot be merged with any other trip.

The output of the MM system is a set of merged trips. For each merged trip T, MM produces the route to be taken by the taxi servicing T and the drop-off points, such that the constraints of all the individual trips merged into T are satisfied. If some drop-off point is not a destination, then MM will also produce the walking path that the passenger has to follow to reach her destination.

Now we discuss the approach used by the MM system. MM consists of two stages: (1) construction of a shareability graph (SG), and (2) finding the maximum matching of SG. The first stage finds all the possible pairs that can be merged in a way that satisfies the constraints of the two trips. In other words, it constructs a graph in which the nodes are the trips, and each edge indicates that the two connected trips can be merged.

To see the need for the second stage, suppose a pool initially consists of 4 trips, and that at the end of the first stage we have a graph of 4 nodes A, B, C, D and 3 edges A-B, B-C, and C-D. If B and C are merged, then no more trips can be merged, and the total number of resulting trips in the pool is 3. If, on the other hand, A and B are merged, and C and D are merged, the resulting number of trips is two, which is superior to the first option.

Thus, the second stage finds, for an arbitrary graph, the merging of pairs which results in the minimum number of merged trips in the pool. For finding the maximum matching we use a standard existing algorithm [21].

A. Building the Shareability Graph

A shareability graph is a graph in which the vertices are trips and the edges indicate that the trip pair connected by the edge is shareable. The shareability graph is constructed as follows.

First, to speed up the graph-building process, we perform following precomputations: (assuming that drop-off points and trip destinations are always intersections). For each intersection P, we precompute only once the following:

a) $\mathbf{I}(P) = \{i \mid WT(P,i) \le C\}$, i.e., the set of the neighboring intersections, from which the walking time to intersection *P* is no greater than *C*. Intuitively, these are candidate drop-off points for trips that have *P* as a destination assuming that the maximum walking time of any trip is C (say 10 minutes).

b) $DT(H,P)$, i.e. the driving time from the hub *H* to intersection *P*, using the speed limits of road segments.

Second, the following pairwise shareability test (PST), which uses the above precomputations, is applied to check whether or not trip a pair (*A,B*) is *shareable with A first.*

Discussion of the PST Algorithm: Line 1 checks whether the route driving from H to dest(A) (along the shortest path), and from there to dest(B), satisfies the maximum delay of B. If so, then this route satisfies the delay constraints of both trips, thus they are shareable.

Otherwise, the rest of the PST algorithm checks for every pair of feasible drop-off points, one of A and the other of B, whether they satisfy the delay constraints of A and B.

Line 4 checks whether the drop-off point satisfies the delay constraint of A, and if not the drop-off point is abandoned. Line 11 checks the same condition for the delay constraint of B, with a lower bound given by the Euclidean distance. Lines 4, 9, and 11 use calculations that involve only constants and precomputed values[1]. They serve as defenses, to avoid the expensive shortest path calculations executed by Line 13.

Line 13 calls *PathSearch* function, which tries to find a path from a drop-off point *i* of dest(A) to some drop-off point *j* of dest(B) within a given travel time budget that satisfies the delay constraint of B. This is expensive because the drive-time between i and j is not precomputed (the table giving the shortest drive-path between every pair of intersections would be too large to search efficiently). And executing the shortest path computation between

[1] the precomputed driving times are multiplied by the congestion fraction (*cf)* parameter.

Algorithm 1: Pairwise Sharability Test (PST), i.e. determine whether or not trip pair (A, B) is sharable with A first

```
Data:
$A$, $B$, given trip pair;
$I(dest(A))$, precomputed drop-off points within the maximum walking
time $W(A)$ from $dest(A)$;
$I(dest(B))$, precomputed drop-off points within the maximum walking
time $W(A)$ from $dest(B)$;
$WT(i,P), i \in I(P)$, precomputed walking time from a drop-off point $i$ of
intersection $P$ to $P$;
$DT(H,P)$, precomputed driving time from hub $H$ to any intersection $P$
using speed limits;
$S_{max}$, maximum speed limit for driving;
$G = (V,E)$, graph representing the road network
$cf$, congestion fraction (of maximum speed);
Result: True or False
/* if shareable using destinations as drop-off points */
1  if $DT(H,dest(A)) + SP(dest(A),dest(B)) - DT(H,dest(B)) \le D(B)$
   then
2      return True
3  for vertex $v \in V$ do /* budget array is to be used in Line 13 */
4      $budget[v] \leftarrow 0$
5  for drop-off point $i \in I(dest(A))$ do
       /* delay for $A$ exceeds the delay threshold */
6      if $DT(H,i) + WT(i,dest(A)) - DT(H,dest(A)) > D(A)$ then
7          continue to Line 3
8      for drop-off point $j \in I(dest(B))$ do
9          $budget_{left} \leftarrow DT(H,dest(B)) + D(B) - DT(H,i) - WT(j,dest(B))$

10         $E(i,j) \leftarrow$ Euclidean distance from $i$ to $j$
           /* delay for $B$ exceeds the delay threshold with lower
              bound approximation for travel time from $i$ to $j$ */
11         if $budget_{left} < E(i,j)/(S_{max} * cf)$ then
12             continue to Line 8
           /* $PathSearch$ excutes at most once for each $i$ */
13         $feasibility \leftarrow$
           $PathSearch(i,j,G,budget_{left} + WT(j,dest(B)),budget)$
14         if $feasibility == True$ then
15             return True
16 return False
```

every pair of feasible drop-off points, one of A and the other of B, involves hundreds of shortest-path computations (we find out that each destination usually have 20~30 drop-off points for a 5 minutes walk). The Path Search Algorithm (PSA, Algorithm 2) improves the efficiency by using the following idea. First, do a single-source shortest path computation from a drop-off point of *A* to all the feasible drop-off points of *B*. The resulting shortest-path tree *T* may contain multiple drop-off points of *A*; and if for any pair of drop-off points in *T*, one for *A* and the other for *B*, the *budget$_{left}$* is not exceeded, then *A* and *B* are shareable with *A* first. Otherwise, the single-source-shortest-path computation is repeated for other feasible drop-off points of *A*, with the following cutoff improvement. If a vertex v that was "seen" in previous single-source-shortest-path computations is reached, and if the shortest path to *v* is not improved, then *v* is "cutoff", i.e. not expanded. In other words, PSA combines multiple single-source-shortest-path computations. Specifically, PSA is executed at most once for each drop-off point of *A*.

Complexity of Constructing the Shareability Graph: In the worst case, the PST Algorithm constructs a shortest-path tree for each drop-off point of A. This takes

Algorithm 2: Path Search Algorithm, based on Dijkstra's Algorithm

```
   Data:
   $i \in I(dest(A))$;
   $I(dest(B))$, precomputed drop-off points of trip $B$;
   $WT(u, dest(B))$, precomputed walking time from a drop-off point $u$ of
   $dest(B)$ to $dest(B)$;
   $G = (V, E)$, graph representing the road network (edge values represent
   driving time under speed limits);
   $budget_{left}$, total travel time budget left for vertex $i$;
   $budget[]$, an array which keeps track of the maximum budget for each
   vertex that has been explored (global varible, modifications to it are
   reflected in the caller function);
   Result: True or False
   /* initial budget is not greater than the maximum budget that
      has been seen for vertex $i$                                    */
 1 if $budget_{left} \leq budget[i]$ then
 2     return False
 3 for vertex $v \in V$ do                      /* distance initialization */
 4     $dist[v] \leftarrow \infty$
 5 $dist[i] \leftarrow 0$
 6 $budget[i] \leftarrow budget_{left}$
 7 $Q \leftarrow$ empty priority queue
 8 add $i$ to $Q$
 9 while $Q$ is not empty do
10     $v \leftarrow$ pop vertex with min $dist[v]$ from $Q$
11     for each neighbor $u$ of $v$ do
12         $tmp \leftarrow budget[v] - e_{vu}$
13         if $tmp \leq budget[u]$ && $budget[u] > 0$ then  /* cut the branch
           if budget left for $u$ is not greater than the max budget
           has been seen for $u$ */
14             continue to Line 11
15         $budget[u] \leftarrow tmp$
16         if $u \in I(dest(B))$ && $WT(u, dest(B)) \leq budget[u]$) then
17             return True
18         $dist[u] \leftarrow min(dist[u], dist[v] + e_{vu})$
19         add $u$ to queue $Q$
20 return False
```

$O(|E| + |V|log|V|)$. Since the number of drop-off points that are at most C time-units (e.g. C = 10 minutes) away from any destination is a constant, this is also the complexity of the PST algorithm. Since A and B are shareable if and only if they are shareable with A first or with B first, and since the number of trips in a pool is bounded by a constant (in our experiments the average number of trips ranges between 25 and 40 depending on the length of the time interval), the above is the asymptotic complexity of constructing the shareability graph.

Complexity of finding the Maximum Matching of the Shareability Graph: The maximum matching can be found in O(|E||V|$^{1/2}$), where |V| is the number of trips, and |E| is the number of edges in the shareability graph (see [21]). Since the number of trips is a constant, finding the maximum matching can be done in a negligibly small constant time.

Incremental updating the Shareability Graph: After the shareablity graph is built, a maximal matching M is computed on the graph. Then all nodes and links that are included in M are removed from the graph, and each pair boards a vehicle. At this point there are 2 possibility for the unmatched trips. They can remain in the pool and board at the same time-interval as single-trips, or drop back to the next pool, attempting to be matched there. If the second option is selected, the remaining part of the graph can be incrementally reused for constructing the new shareability graph. That is, given a pool of trips consisting of c old trips and d new trips, to build the new sharability graph, we only need to run the PST algorithm *d(d+c)* times instead of *(d+c)*^{2} times.

B. Euclidean Filtering

In order to eliminate the infeasible pairs of trips quickly, rather than feeding them through the PST algorithm directly, the MM method uses the principles of Euclidian geometry. We call this Euclidean filtering. More specifically, in this subsection we present an inequality, (9), which, if not satisfied, for a pair of trips, then the trip-pair cannot be shareable; thus the pair does not need to be fed to the PST algorithm. Furthermore, (9) can be computed using only the precomputed tables, but independently of the road network, i.e. in constant time.

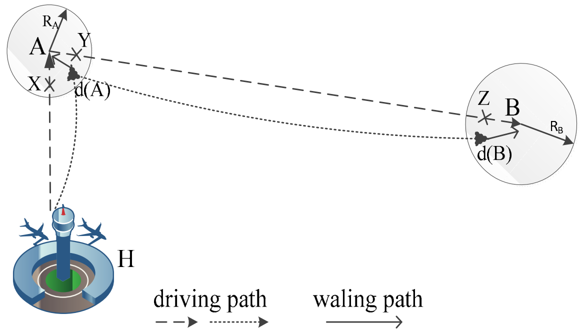

Fig. 3: Demonstration of the proof of Theorem 2

Denote by S_{max} the maximum driving speed among all edges (e.g. 60 mi/hr) without congestion, i.e. when *cf=1*. We trivially assume that (S_{max} * *cf*)>*WS* (Otherwise, walking from H to the destination would be faster than driving).

Given two points X and Y in the Euclidean space, denote by by $D_{X,Y}$ the Euclidean distance between two points *X* and *Y*, and by $T_{X,Y}$ the time to cover $D_{X,Y}$ at speed (S_{max} * *cf)*.

Theorem 2: If *A* and *B* are two trips that are shareable with *A* first, then:

$$T_{H,dest(A)} + T_{dest(A),dest(B)} - 2 * W(A) < SP(B) + D(B) \quad (9)$$

Proof: Consider the two shareable trips *A* and *B* with drop-off points *d(A)* and *d(B),* as represented in Fig 3. *A*, *B* represent the destination locations of the two trips respectively, and R_A, R_B represent the maximum distances that can be covered in times *W*(*A*) and *W*(*B*), respectively, at a walking speed of *WS*.

We will prove that:

$$T_{H,dest(A)} - W(A) < SP(d(A)), \quad (10)$$

and,

$$T_{dest(A),dest(B)} - W(A) < DT(d(A), d(B)) + T(wp(B)) \quad (11)$$

Recall from sec. III.A that *SP*(*d*(A)) is the shortest drive-time from *H* to the *d(A)*, and *DT*(*d*(A), *d*(B)) is the shortest driving-time from the drop-off point of *A* to that of *B*.

Thus, combining (10) and (11), we have

$$T_{H,dest(A)} + T_{dest(A),dest(B)} - 2 * W(A) < SP(d(A)) + DT(d(A), d(B)) + WT(wp(B)) \quad (12)$$

where the right side of (12) is the total travel time for *B* in any ridesharing plan with *A* first, thus not larger than $SP(B) + D(B)$; the theorem follows.

Now we prove (10). Denote by *X* the point on the straight line from H to *dest(A)* such that $D_{X,dest(A)} = D_{d(A),dest(A)}$. In other words, *X* is the point on the straight line whose Euclidean distance from *dest(A)* is the same as the Euclidean distance between *d(A)* and *dest(A).*

Since a straight line is the shortest distance between two points in Euclidean space, then $D_{HX} \leq D_{H,d(A)} \leq D(d(A))$, where $D(d(A))$is the distance of the shortest-time driving path from *H* to *d*(*A*). Thus we have $T_{H,X} \leq T_{H,d(A)} \leq \frac{D(d(A))}{S_{max}*cf} \leq SP(d(A))$. Also, since ($S_{max}$ * *cf*)> *WS*, $T_{X,dest(A)} < WT(D_{X,dest(A)}) \leq W(A)$. Thus: $T_{H,dest(A)} = T_{H,X} + T_{X,dest(A)} < SP(d(A)) + W(A)$, giving (10).

Now we prove (11). Denote by *Y* the point on the straight line between *dest*(A) and *dest*(B) such that $D_{dest(A),Y} = D_{dest(A),d(A)}$, and by *Z* the point on the same line such that $D_{Z,dest(B)} = D_{d(B),dest(B)}$. Since ($S_{max}$ * *cf)*> *WS*, $T_{dest(A),Y} < W(A)$ and $T_{Z,dest(B)} < WT(D_{d(B),dest(B)})$, then $T_{dest(A),dest(B)} \leq T_{dest(A),Y} + T_{YZ} + T_{Z,dest(B)} < T_{YZ} + W(A) + WT(D_{d(B),dest(B)})$.

Since a straight line is the shortest distance between two points in Euclidean space, $D_{YZ} \leq fp(d(A),d(B))$ where *fp*(*d*(*A*),*d*(*B*)) is the length of the path that gives the shortest driving-time between *d(A)* and *d(B)*. Since S_{max} is maximum speed of all road segments along the shortest path between *d(A)* and *d(B)*, we have $T_{YZ} \leq DT(d(A), d(B))$. Thus: $T_{dest(A),dest(B)} - W(A) < DT(d(A), d(B)) + WT(wp(B))$, which is (11).

If between every pair of intersections driving is faster than walking, then the lower bound of Th. 2 can be improved by replacing $T_{H,dest(A)}$ by the higher *SP(A)*. Precisely:

Theorem 3: If A and B are two trips that are shareable with A first, and between every pair of intersections the driving time is shorter than the walking time, then:

$$SP(A) + T_{dest(A),dest(B)} - 2 * W(A) < SP(B) + D(B) \quad (13)$$

Proof: In this case the proof of Th. 2 can be repeated verbatim, except that (10) is: $SP(A) - W(A) \leq SP(d(A))$; and it holds for the following reason. If $SP(A) - W(A) > SP(d(A))$, then *SP*(*A*) can be shortened as follows: drive from *H* to *d*(*A*) along the shortest path (which will take less than $SP(A) - W(A)$) and then drive from *d*(*A*) to *dest*(*A*) (which will take less than *W*(*A*)).

V. Evaluation

A. Databases

The first database used in the evaluation of the MM system is the NYC taxi trip database (see[22],[23]). This database records over four years of taxi operations in New York City (NYC) and includes nearly 700 million trips. The database is stored in CSV format, organized by year and month. In each file, each row represents a single taxi trip described by fields such as taxi ID, timestamped origin and destination, travel time and distance, and passengers count. The database does not provide GPS sequences for a trip. Reference [24] provides a detailed description of the NYC taxi data.

The experiment was conducted on randomly selected pools formed from 1.8 million trips. These trips originated from LaGuardia airport with destinations in NYC, during 261 weekdays in 2013 between 10am and 10pm. The pools were created based on the departure time. So, if the pool size is 5 minutes, the input (before merging) trips in the Jan. 15th, 10:00-10:05a pool are all the trips that departed on that date between 10:00 and 10:05, as reflected in the dataset. As Fig. 8 indicates, the average number of input trips ranges from 20 (for a 5 minute pool) to 40 (for a 10 minute pool), assuming that 90% of the trips can be shared. Observe that this method of generating the pools is conservative in the sense that it does not model unsatisfied demand. More precisely, it is possible that many passengers faced with a taxi line have decided to, for example, take the bus. A less conservative approach would have inflated the actual demand reflected in the database to model this unsatisfied demand, leading to a higher number of trips per pool, and thus to higher savings resulting from ride-sharing.

In this paper, the results presented in section D are based on a hundred pools randomly chosen from the above trip database. We observe that a hundred random pools yield an acceptable confidence level of the statistics to be presented in Section D. For example, consider 100 random 5-minute pools extracted for the following experimental setup: percentage of willingness to ride share = 90%, Max Delay = 10% of the individual shortest path trip time, and Maximum Walk Time = 5 min (see Fig. 4). The result indicates that the average number of trips saved per pool is 6.25 with a standard deviation of 5.29. If a normal distribution is assumed, then there is at least 88% confidence level that the average number of trips saved per pool is not lower than 90% of the mean value.

The second database used in the experiments is the road network. For creating the street network of New York City, the data from openstreetmap.org was used, consisting 486,746 road links and 261,187 intersections (i.e. vertices).

A. Metrics

RSVP is evaluated according to the following performance metrics.

- *Computation time*. Since ride-sharing plans must be computed continuously as customers arrive and depart the virtual queue, efficient computation is important.
- *Reduction in total number of trips in RSVP*. It is expected that trip reduction is related to a number of factors including the willingness to ride share by passengers, the pool size, maximum walking time tolerated, maximum total delay tolerated, and traffic condition (reflected by traffic speed). We examine the trip savings with respect to each of those key input parameters in the experiment.

B. *Setup of Experiment*

The congestion fraction: For determining the shareability of trips A and B, travel time was assigned to each road segment based on the road type, its maximum travel-speed (e.g. 40mi/hr on an arterial road), and a congestion fraction *cf* computed as follows. Denote by cf_1 the fraction = (travel-time from the hub to *dest*(*A*) at maximum speed allowed by each edge)/(actual travel time of trip A from the hub to *dest*(*A*)). And denote by cf_2 the fraction = travel-time from the hub to *dest*(*B*) at maximum speed allowed by each edge)/(actual travel time of trip B from the hub to *dest*(*B*)). Then the congestion fraction $cf=(cf_1+cf_2)/2$. In other words, *cf* is the average of the congestions reflected by trips A and B.

Destinations: The trip destinations were matched to the nearest intersections on the road map, using a kd-tree based KNN search. Intersections from the openstreetmap data were computed using QGIS. For each intersection a collection of neighboring intersections within *10* minutes walking distance (at 3mi/hr) were precomputed using Breadth-First-Search.

Taxi capacity: We assume that each taxi cab has 4 passenger seats. Therefore trip combinations with a total passenger count of at most four can be merged. For example, a trip with 2 passengers can be merged with another trip with 2 passengers but not with a 3-passenger trip.

C. *Results*

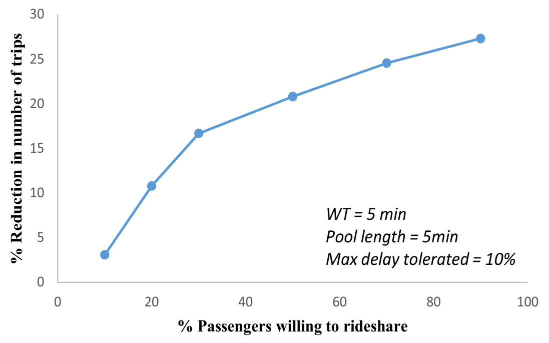

Fig. 4 % trips reduction by willingness to ride share

Fig. 4 shows the percentage of trip reduction by percentage of passengers' willingness to ride share starting at 10%. It is assumed the maximum walking time is 5 minutes, pool size is in a 5-minute interval, and the maximum delay tolerated is 10% of the individual shortest path trip time. As expected, as more passengers are willing to ride share, the percent trip reduction increases.

Fig. 5 gives the percent trip reduction as a function of the driving speed. This speed is given as a percentage of the corresponding maximum speed, depending on the road type. So for example, 50% indicates that on a highway the average speed is 30mi/hr, and on an arterial road (where the maximum speed without traffic is 40mi/hr) is 20mi/hr. It is assumed here that the percent passenger willingness to ride share (*rf*) is 90%, the maximum walking time is 5 minutes, pool size is in a 5-minute interval, and the max delay tolerated is 10% of the individual shortest path trip time. It is interesting to see that the % trip reduction remains at about 28% regardless of the network driving speed. In other words, RSVP will consistently deliver a significant trip reduction regardless of the network traffic condition. That is an encouraging finding.

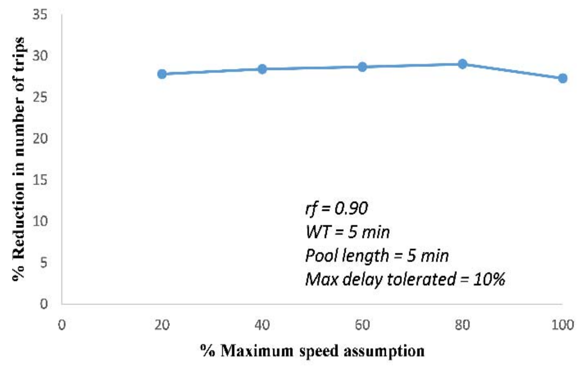

Fig. 5 % trips reduction by congestion fraction

When the average maximum delay tolerated varies from 5% to 20% of the travel time, percent trip reduction goes up from 18% to 36% accordingly (Fig. 6). That should come as no surprise - as passengers are more

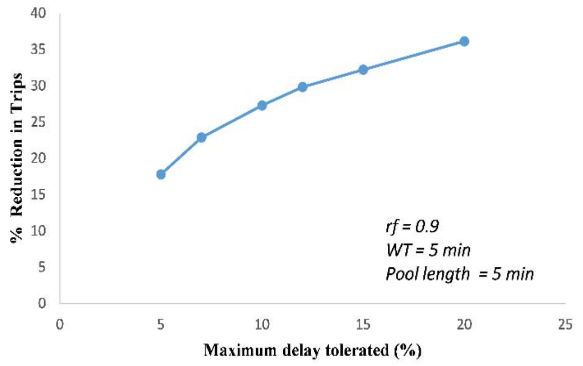

Fig. 6 % trips reduction by average maximum delay flexible with their travel time budget more trips can be shared and thus the total number of trips is further reduced.

One of the important features of RSVP is the incorporation of walking option from the drop-off point to the final destination by allowing passengers to specify the maximum tolerable walk time to their destinations after drop-off. It was hypothesized that RSVP would increase ride-sharing by incorporating this feature. Fig. 7 confirms the hypothesis. Moreover, when the maximum walk time goes from zero to a mere 3-minute bound, it results in 15% additional trip reduction, which is a significant reduction. Notice that the % trip reduction levels off after 5 minutes, which suggests a 5-minute walk time tolerance would be a good cut-off point in practice.

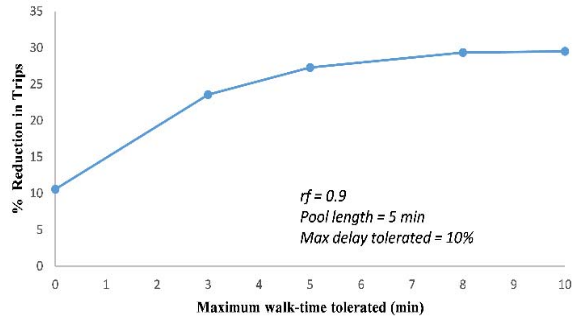

Fig. 7 % trip reduction by maximum walk time

Another interesting feature to observe is that percent trip reduction seemingly has little to do with the pool size (Fig. 8). This is a desirable feature because it implies that the similar ride sharing result will be obtained regardless of how the trips are pooled. Therefore, in practice the pool size should be 6 minutes, the point at which the savings levels off. The reason for this is as follows. In a pool of size n minutes, the average wait to board is n/2 minutes. Thus, to minimize this wait, the pool size should be the smallest such that beyond it the savings is marginal.

Fig. 8 also gives the actual numbers of trips before and after MatchMaking, for each pool-size.

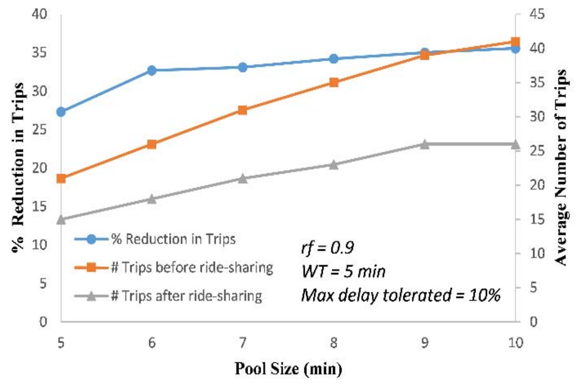

Fig. 8 % trip reduction by pool size

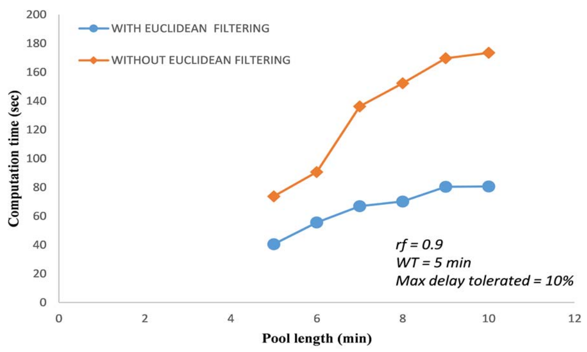

Fig. 9 Computation time by pool size

Computation time of the MM system is investigated as a function of the pool size. The MM system was implemented in Java, and run on a single system equipped with 2.5 GHz CPU and 16 GB RAM using a single thread. Two cases were analyzed – with and without Euclidean filtering. Fig. 9 shows the result, assuming a 90% willingness to ride share (*rf*), a 5-min maximum walking time, and a 10% max delay tolerated. The Euclidean filtering algorithm is proved to be effective in reducing the computation time by a factor ranging from 33% to over 50%. Without the Euclidean elimination algorithm, the computation time escalates 2.5 times when the pool size increases from 5 minutes to 10 minutes.

VI. CONCLUSION

In this paper we proposed the RSVP scheme to facilitate ride-sharing at transportation hubs. The scheme combines in a unique way three existing mechanisms: virtual queues, slugging, and multiple-drop-off ridesharing. The scheme produces pools of trips, which are then consolidated into ride-sharing plans by the MatchMaking (MM) system. Technically, the heart and novelty of the MM system is a combination of two components: (1) Euclidean filtering that uses Euclidean geometry to reduce the complexity of finding an optimal ride-sharing plan, and (2) the PST algorithm which uses a middle ground between single-source-shortest-path and all-pairs-shortest-path.

Then we evaluated the RSVP scheme on 100 random pools formed from 1.8 Million trips that originated from LaGuardia Airport in NYC. The results indicate that:

(1) The trip-savings enabled by RSVP-ride-sharing are significant, e.g., about 25% of the trips are saved if about 75% of the passengers are willing to ride-share. This is true for a modest delay of 10% of the trip time, and a maximum walk of 5 minutes.
(2) Walking is valuable in combination with multiple-drop-off ride-sharing. For example, if passengers allow a 10-minutes walk, then the trips-reduction by ride-sharing increases from about 10% to about 30%. Considering that at airports passengers often walk for 10 minutes from the gate to the curb, this assumption seems reasonable.

(3) The computation time for a 6 minutes pool of trips is less than 1 minute.
(4) Euclidean filtering is effective, reducing the computation time by 30%-50%.

Much remains to be done in terms of future work. First, the optimization criteria needs to be refined to travel-time per vehicle rather than number of trips. Second, the sharing of more than two trips needs to be investigated, although [5] determined that the additional savings from allowing the sharing of 3 trips is marginal. Third, the results need to be compared with other hubs. Finally, the RSVP schemes can be reversed to traveling to the hub, rather than from it. In other words, instead of having a single source, the passengers would have a single destination, the hub. And they would walk to pick-up locations, rather than from drop-off locations.

Acknowledgment

This work was supported in part by NSF grants IIS-1213013, IIP-1534138 and by the NURail Center, under grant DTRT12-G-UTC18 of the U.S. DOT.

References

[1] J. Hilkevitch, “O’Hare taxi passengers, drivers often in holding pattern,” *Chicago tribune*, 12-Jul-2015. [Online]. Available: http://www.chicagotribune.com/business/breaking/chi-taxicabs-ohare-getting-around-met-20150316-column.html#page=1.

[2] S. Ma and O. Wolfson, “Analysis and evaluation of the slugging form of ridesharing,” *Proc. 21st ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst. - SIGSPATIAL’13*, pp. 64–73, 2013.

[3] G. Gidofalvi, T. B. Pedersen, T. Risch, and E. Zeitler, “Highly scalable trip grouping for large-scale collective transportation systems,” in *Proc. of the 11th Int. Conf. on Extending database technology Advances in database technology*, 2008, p. 678.

[4] S. Ma, Y. Zheng, and O. Wolfson, “Real-Time City-Scale Taxi Ridesharing,” *IEEE Trans. Knowl. Data Eng.*, vol. 27, pp. 1782–1795, 2015.

[5] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, “Quantifying the benefits of vehicle pooling with shareability networks,” *Proc. Natl. Acad. Sci.*, vol. 111, no. 37, pp. 13290–13294, 2014.

[6] C. Tian, Y. Huang, Z. Liu, F. Bastani, and R. Jin, “Noah: a dynamic ridesharing system,” *Proc. 2013 ACM SIGMOD Int. Conf. Manag. Data. ACM*, pp. 985–988, 2013.

[7] G. L. Curry, A. De Vany, and R. M. Feldman, “A queueing model of airport passenger departures by taxi: Competition with a public transportation mode,” *Transp. Res.*, vol. 12, no. 2, pp. 115–120, 1978.

[8] A. Anwar, M. Volkov, and D. Rus, “ChangiNOW : A Mobile Application for Efficient Taxi Allocation at Airports,” *Proc. 16th Int. IEEE Annu. Conf. Intell. Transp. Syst. (ITSC 2013)*, no. Itsc, pp. 694–701, 2013.

[9] D. C. T. da Costa and R. de Neufville, “Designing efficient taxi pickup operations at airports,” *Transp. Res. Rec. J. Transp. Res. Board*, vol. 2300, pp. 91–99, 2012.

[10] M. A. Yazici, C. Kamga, and A. Singhal, “A Big Data Driven Model for Taxi Drivers ’ Airport Pick-up Decisions in New York City,” *IEEE Int. Conf. Big Data*, pp. 37–44, 2013.

[11] G. Laporte, “Fifty Years of Vehicle Routing,” *Transp. Sci.*, vol. 43, no. 4, pp. 408–416, Nov. 2009.

[12] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, “Optimization for dynamic ride-sharing: A review,” *Eur. J. Oper. Res.*, vol. 223, no. 2, pp. 295–303, Dec. 2012.

[13] R. Rudnicki, K. Anders, and M. Sester, “Rendezvous-Problem in Local Shared-Ride Trip Planning,” in *International Society for Photogrammetry and Remote Sensing Congress*, 2008.

[14] P. Czioska, D. Mattfeld, and M. Sester, “GIS-based Identification and Assessment of Suitable Meeting Point Locations for Ride-Sharing,” *19th Euro Work. Gr. Transp. Meet. EWGT 2016*, 2016.

[15] K. Aissat and A. Oulamara, “Meeting Locations in Real-Time Ridesharing Problem: A Buckets Approach,” *Oper. Res. Enterp. Syst.*, vol. 577, pp. 71–92, 2015.

[16] M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar, “The Benefits of Meeting Points in Ride-sharing Systems,” *Transp. Res. Part B Methodol.*, vol. 82, pp. 36–53, 2015.

[17] R. A. Abbaspour and F. Samadzadegan, “An Evolutionary Solution for Multimodal Shortest Path Problem in Metropolises,” *Comput. Sci. Inf. Syst.*, vol. 7, no. 4, pp. 1820–0214, 2010.

[18] A. Lozano and G. Storchi, “Shortest viable path algorithm in multimodal networks,” *Transp. Res. Part A Policy Pract.*, vol. 35, no. 3, pp. 225–241, 2001.

[19] W. C. David Applegate, Ribert Bixby, Vasek Chvatal, “Concorde TSP solver,” 2006. .

[20] J. Dubois-lacoste, H. H. Hoos, and T. Stützle, “On the Empirical Scaling Behaviour of State-of-the-art Local Search Algorithms for the Euclidean TSP,” *Proc. 2015 Genet. Evol. Comput. Conf. ACM*, pp. 377–384, 2015.

[21] Z. Galil, “Efficient algorithms for finding maximum matching in graphs,” *ACM Comput. Surv.*, vol. 18, no. 1, pp. 23–38, 1986.

[22] N. Y. C. T. & L. C. (TLC) in partnership with the N. Y. C. D. of I. T. and T. (DOITT), “Big Step for Big Data: Yellow/Green Taxi Trip Records Now Available Online,” 2015.

[23] B. Donovan and D. B. Work, “New York City Taxi Trip Data (2010-2013),” 2014.

[24] A. J. T. Swoboda, “New York City Taxicab Transportation Demand Modeling for the Analysis of Ridesharing and Autonomous Taxi Systems,” *B.S. Thesis, Dep. Oper. Res. Financ. Eng. Princet. Univ.*, no. June, 2015.