When Hashing Met Matching: Efficient Spatio-Temporal Search for Ridesharing

Chinmoy Dutta
Turing Research
chinmoy@turingresearch.ai

Abstract

Shared on-demand mobility holds immense potential for urban transportation. However, finding ride matches in real-time at urban scale is a very difficult combinatorial optimization problem and mostly heuristic methods are applied. In this work, we introduce a principled approach to this combinatorial problem. Our approach proceeds by constructing suitable representations for rides and driver routes capturing their essential spatio-temporal aspects in an appropriate vector space, and defining a similarity metric in this space that expresses matching utility. This then lets us mathematically model the problem of finding ride matches as that of Near Neighbor Search (NNS). Exploiting this modeling, we devise a novel randomized spatio-temporal search algorithm for finding ride matches based on the theory of Locality Sensitive Hashing (LSH). Apart from being highly efficient, our algorithm enjoys several practically useful properties and extension possibilities. Experiments with large real-world datasets show that our algorithm consistently outperforms state-of-the-art heuristic methods thereby proving its practical applicability.

Introduction

In this paper, we consider shared on-demand mobility. Its importance for planning urban transportation lies in its promise to provide a solution for the serious urban challenges of reducing excessive traffic congestion, resource consumption and air pollution while providing efficient, flexible and affordable mobility to people (Arnott and Small 1994; Caiazzo et al. 2013; Pant and Harrison 2013; Carrion and Levinson 2012).

Real-time information and monitoring of urban mobility and the ability to do large scale computation on the cloud efficiently allow on-demand shared mobility platforms (e.g. transportation network companies, on-demand microtransit companies, etc., referred to as ridesharing platforms here) to enable sharing of rides to unprecedented levels. A number of works (Agatz et al. 2011; Santi et al. 2014; Friedrich, Hartl, and Magg 2018) have studied urban traffic and confirmed the tremendous potential of real-time urban-scale shared mobility to reduce the burden on urban transportation. In order to exploit this potential, the cost of sharing (detours, increased travel time, loss of privacy etc.) needs to be balanced with its utility (reduced traffic congestion, reduced resource consumption, reduced air pollution, cheaper rides, use of high-occupancy lanes etc.). Thus, ability to match rides minimizing cost while maximizing utility is of paramount importance for the success of shared mobility.

However, the urban scale and real-time nature of this combinatorial problem make it extremely difficult to tackle. Among all the solution approaches, those based on the graph-theoretic framework of *shareability network*, introduced in (Santi et al. 2014), have been most successful. These approaches owe their success to the systematic modeling of the match pool in a graph-theoretic setting which then unlocks a rich set of techniques from graph theory to tackle the challenge. The nodes of the network are either rides or driver routes. An edge between two ride nodes represents feasibility of matching them together while an edge between a ride node and and a driver route node represents feasibility of adding the ride to the driver route. Here, feasibility means that the platform-defined user experience constraints (e.g. maximum pickup wait, maximum detour etc.) are satisfied for all the rides concerned.

The basic building block of shareability network is the foundation for the highly influential work of (Alonso-Mora et al. 2017) which provided a framework for high-capacity ridesharing by computing cliques in the network and solving an Integer Linear Program (ILP) over them. Their computationally prohibitive framework was subsequently simplified in (Simonetto, Monteil, and Gambella 2019) relying on bipartite matching instead of solving an ILP, while still basing their methodology on the shareability network. Subsequent works of (Alonso-Mora, Wallar, and Rus 2017), (Yu and Shen 2020) and (Shah, Lowalekar, and Varakantham 2020) have all relied on the shareability network as their basis.

Computing the shareability network in real-time, however, is a computationally intensive task, and none of these prior works effectively tackled this question in a principled way. Comparing nodes pairwise for feasibility has a time complexity of $O(n^2)$ and requires $O(n^2)$ calls to a routing service for a match pool with n nodes. For a ridesharing platform, it is common to have tens of thousands of nodes in the match pool for a large urban region at peak commute times. These platforms also employ several features to further thicken the match pool, such as batching of arriving ride requests, swapping already made matches, etc. To generate the shareability network by a quadratic complexity search is therefore ruled out.

Heuristic-based methods have been proposed and, unfortunately, comprise the state-of-the-art to deal with this combinatorial challenge (Balardino and Santos 2015; Alonso-Mora et al. 2017). For example, (Alonso-Mora et al. 2017) suggests the heuristic of connecting a node with only a small number of close-by nodes. However, it is not uncommon to find large number of co-located ride requests at roughly the same time (e.g outside train stops when trains arrive, outside event venues when events end). Moreover, a service that lets riders wait longer for cheaper rides or lets riders schedule rides in advance needs to make matches even between far-away nodes. Another class of heuristics often used in practice involves computing Haversine overlap between nodes to approximate the true utility of matching them. However, the feasibility and utility of matching is highly dependent on the physical road network, and ignoring this knowledge yields poor matching decisions.

Apart from ignoring the physical road network, another common drawback of almost all the heuristic methods is that they approximate the costs of routes (and hence match utility) in terms of distances. However, the cost of a route depends on the combination of a variety of factors such as distance, duration, tolls and taxes incurred along the route etc. Inability to account for these various cost factors not only renders the heuristic methods sub-optimal but also makes them unable to cope with changing real-time traffic conditions.

Our Contributions

In this work, we devise an efficient randomized spatio-temporal search algorithm to construct a sparse, utility-aware shareability network. In doing so, we directly tackle the combinatorial difficulty at the heart of most of the recent approaches to ride matching which prevents them from being effective in practice. The proposed algorithm can be used with any of these approaches to enable efficient urban-scale on-demand shared mobility.

In order to efficiently construct a high utility sparse shareability network, we must efficiently find a small set of high utility potential matches for each node; and we do so using the theory of locality sensitive hashing (LSH).

At a high level, our approach works by finding suitable vector representations for the ride and driver route nodes in a high-dimensional ambient vector space, that capture both their spatial aspects (e.g. route in the physical road network) and their temporal aspects (e.g. arrival time, service time constraints, etc.). Next, we define a similarity metric for this vector space in terms of inner product that respects matching utility: larger similarity between the vector representations of nodes imply higher matching utility between the nodes. With this similarity metric in hand, the search for high-utility potential matches reduces to the problem of similarity search, which we solve using LSH techniques. For this, We combine the asymmetric LSH construction of (Shrivastava and Li 2015) for Maximum Inner Product Search (MIPS) and the cross-polytope LSH construction of (Andoni et al. 2015) for Maximum Cosine Similarity Search (MCSS) to construct a locality sensitive hashing data structure for storing these vector representations that allows efficient search for similar vectors: we can find k (say 10) approximately most-similar nodes (according to our similarity measure) for each of the n nodes in the match pool efficiently in time $O(n^{1+\rho}(k + \log n)\log k)$ and space $O(n^{1+\rho}\log k)$ for a small $\rho < 1$. Combining these ingredients lets us efficiently construct a sparse utility-aware shareability network by connecting nodes to their most-similar nodes. In the process, we only make n calls to a routing service, one for each node while constructing its vector representations, which is all we need to enable our algorithm to exploit the knowledge of the physical road network.

In order to demonstrate the practicality and effectiveness of our algorithm, we conduct large-scale experiments using publicly available NY yellow taxi trip datasets under varying traffic patterns and varying load. The results show that our algorithm consistently outperforms even the best state-of-the-art heuristic method by as much as 6 percentage points in terms of matching utility, which translates to significant savings for the riders, drivers and the platform and significant socio-economical and environmental value for cities.

Like most techniques in Artificial Intelligence, effective use of LSH techniques is as much an art as it is a science requiring right tuning of parameters and effective use of domain knowledge. We describe several techniques and insights that we employed and valuable learning that we obtained while carrying out our experiments in a later section. We believe they will aid reproducability of our results. Moreover, the reader might find them useful in applying LSH techniques to other problem domains.

We note that while this work, unlike (Dutta and Sholley 2017; Huang et al. 2018b; Ashlagi et al. 2019), does not attempt to provide an understanding of the complexity-theoretic aspects of the ride matching problem, it does make significant practical progress on the problem by making some novel mathematical contributions for tackling it:

1. A mathematical representation for rides and driver routes capable of summarizing their essential spatial and temporal aspects relevant for matching, exploiting the knowledge of the physical road network. (This accounts for the effectiveness of our algorithm as shown by our experiments.)
2. A mathematical metric on the space of these representations that can express the utility of matching and brings the rich theory of efficient neighbor search at our disposal. (This accounts for the runtime efficiency of our algorithm.)

Our algorithm enjoys several useful properties and can be extended in several interesting ways. First, it works for matching utility defined in terms of any cost function that is linear in the route. Here, linearity means total cost of the route is the sum of costs incurred along the segments of the route. Examples include travel duration, travel distance, tolls and taxes along the route etc., or any linear combination thereof. Second, it can handle time-varying cost functions (e.g. varying with traffic), which static methods (e.g. those based on Haversine distance) can not. Third, it can work for a hybrid ridesharing platform that provides on-demand rides as well as pre-scheduled ones. Lastly, it can even be adapted to allow incremental computation of the shareability network. All these extensions are sketched in a later section.

To summarize, our technical contributions include:

1. A novel principled approach and an efficient randomized algorithm for constructing a sparse utility-aware shareability network.
2. Theoretical proof of correctness and runtime efficiency of the proposed algorithm.
3. Demonstration of the practical utility of the algorithm via large-scale experiments and quantitatively establishing the effectiveness of our approach against various heuristics and a theoretical optimal.
4. Several techniques and insights for applying LSH techniques to the ride matching and potentially other problem domains.
5. Several interesting and useful extensions to the proposed algorithm that make it even more practically appealing.

Related Literature

Traditionally, most of the studies related to mobility-on-demand consider services that do not share rides, e.g (Spieser et al. 2016; Schreieck et al. 2016; Masoud and Jayakrishnan 2017; Dickerson et al. 2018; Zheng, Chen, and Ye 2018; Zhao et al. 2019; Lesmana, Zhang, and Bei 2019; Amirmahdi and Masoud 2020). However recently, there has been a lot of research interest in carpooling and associated challenges after several studies analyzed urban traffic and estimated huge potential for sharing rides (Agatz et al. 2011; Santi et al. 2014; Friedrich, Hartl, and Magg 2018).

Heuristic-based solutions to matching rides were proposed in (Ma, Zheng, and Wolfson 2013; Balardino and Santos 2015). (Alonso-Mora et al. 2017) proposed a framework for high-capacity ridesharing by extending the framework of shareability networks first proposed in (Santi et al. 2014). Simonetto et al. (Simonetto, Monteil, and Gambella 2019) subsequently simplified the framework. Other approaches for ride matching have been proposed in (Koide, Tadokoro, and Yoshimura 2015; Bei and Zhang 2018; Al-Abbasi, Ghosh, and Aggarwal 2019).

Nearest Neighbor Search (NNS) is a problem of major importance in several areas of science and engineering. Approximate nearest neighbor search techniques based on Locality Sensitive Hashing (LSH) was introduced in (Indyk and Motwani 1998; Gionis, Indyk, and Motwani 1999). Owing to being parallelizable and suitable for high dimensional data, these techniques have found widespread use in research and industrial practice (Georgescu, Shimshoni, and Meer 2003; Jgou, Douze, and Schmid 2011; Sundaram et al. 2013). However, these techniques had not been introduced to ride matching prior to this work.

The approximate Maximum Inner Product search (MIPS) is a fundamental problem with variety of applications in areas such as recommendation systems (Li et al. 2017), deep learning (Spring and Shrivastava 2017), etc. The concept of Asymmetric LSH (ALSH) was proposed in (Shrivastava and Li 2014) and several ALSH schemes for MIPS have been proposed since (Shrivastava and Li 2014, 2015; Huang et al. 2018a).

Preliminaries

For simplicity of exposition, we only consider ride nodes in the rest of the paper. Handling driver route nodes is exactly the same conceptually.

A gentle discussion on LSH and its use in solving problems such as NNS and MIPS can be found in (Shrivastava and Li 2014).

Potential Match Search

Let $p_1, p_2, \ldots p_l$ be points on the physical road network. Let C($< p_1, p_2, \ldots, p_l >$) denote the minimum cost of a route that starts at point p_1, traverses $p_2, \ldots p_{l-1}$ in order and ends at p_l. Given a ride r, let r_s and r_t denote its pick-up and drop-off locations respectively. The cost of the ride r, which we denote by $C(r)$ abusing notation, is defined as $C(< r_s, r_t >)$. The cost can be measured in terms of travel distance, travel duration or any other metric linear in the route.

Fix a linear route cost function. The utility of matching two rides comes from the cost savings achieved by serving them together compared to serving them individually. More formally, abusing notation, let $C(\{r, r'\})$ denote the cost of serving rides r and r' together:

$$\begin{aligned} C(\{r, r'\}) = \\ &\min\{C(< r_s, r'_s, r_t, r'_t >), C(< r_s, r'_s, r'_t, r_t >), \\ &\quad C(< r'_s, r_s, r'_t, r_t >), C(< r'_s, r_s, r_t, r'_t >)\}. \end{aligned}$$

The feasibility of matching two rides together is expressed by the feasibility function F: $F(\{r, r'\})$ is 1 if it is feasible to match r and r', 0 otherwise. The matching utility U of matching rides r and r' together is then defined as:

$$U(\{r, r'\}) = (C(r) + C(r') - C(\{r, r'\})) * F(\{r, r'\}).$$

Note that in order to maximize the total matching utility from all the matches made, it may not be desirable to match a given ride with the one with which it has the highest matching utility. Therefore, the approximate potential match search problem is to find k potential matches for every ride with approximately highest matching utilities, for a small enough k. From this, a sparse, utility-aware shareability network can be constructed. An optimal matching in this sparse shareability network instance constructed then yields near-optimal total matching utility.

Definition 1 (Approximate Potential Match Search (PMS)). *Let P be a pool of rides. Let S be the similarity measure between two rides defined as the matching utility between them. The (c, s, k)-approximate Potential Match Search (PMS) for $c < 1$ with failure probability f is to construct a data structure over P supporting the following query: given any query ride q, if there exists k rides $r_1, r_2, \ldots, r_k \in P$ such that $S(q, r) \geq s \; \forall r \in \{r_1, r_2, \ldots, r_k\}$, then report some k rides $r'_1, r'_2, \ldots, r'_k \in P$ such that $S(q, r') \geq cs \; \forall r' \in \{r'_1, r'_2, \ldots, r'_k\}$, with probability $1 - f$.*

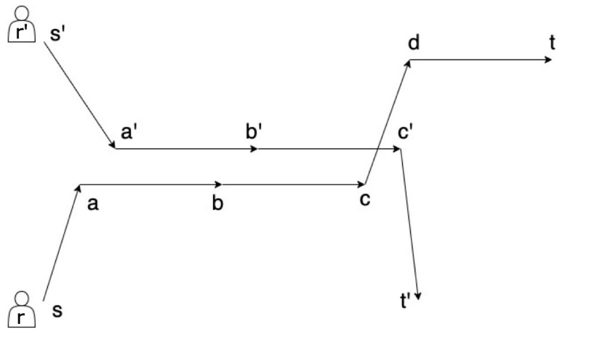

Figure 1: Two rides r and r'. The route for r is $< s, a, b, c, d, t >$ and for r' is $< s', a', b', c', t' >$.

.

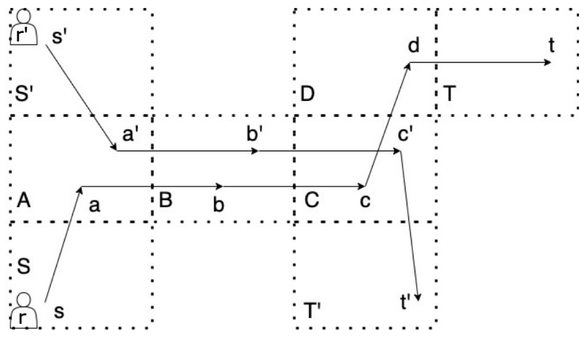

Figure 2: Rides r and r' from Figure 1. The space-discretized route for r is $< S, A, B, C, D, T >$ and for r' is $< S', A, B, C, T' >$. The space-discretization is shown with dotted lines and the discretized spatial blocks are shown in capital letters.

.

Heuristic Methods

To the best of our knowledge, search for potential ride matches has not been formalized before this work and there are no principled methods known for the same. The state-of-the-art heuristic methods include:

- **CLOSEBY:** k closest rides to the given ride with respect to pickup location are chosen as potential matches.
- **HAVERSINE:** k rides with highest Haversine matching utilities with the given ride are chosen as the potential matches. Haversine matching utility is defined with respect to Haversine cost function which is the Haversine distance between two points.
- **CLOSEBY-HAVERSINE:** A hybrid approach of the above two, often used in practice. A sufficiently large number of rides closest to the given ride with respect to pickup location are first shortlisted, and the k among them with highest Haversine matching utilities with the given ride are then chosen as potential matches.

Algorithm for Approximate PMS

Ride Representation and Similarity

Consider two rides r and r' as shown in Figure 1. For brevity, we denote the pickup r_s and dropoff r_t for ride r simply as s and t respectively. Similarly, the pickup and dropoff of ride r' are denoted by s' and t' respectively. Let $< s, a, b, c, d, t >$ and $< s', a', b', c', t' >$ denote shortest routes of rides r and r' respectively.

For simplicity, in this paper, we assume it is feasible to match r and r' together. This assumption is relaxed in the full version. Given the feasibility assumption, the matching utility between r and r', which is the cost savings by matching them together, is approximately the sum of the costs of roughly overlapping edges. Intuitively, since points a, b and c are close to points a', b' and c' respectively, this is approximately $C(< a, b >) + C(< b, c >)$. In order to capture the notion of spatial proximity, we discretize space and represent each point of the route by the discretized spatial block it falls in. We can use discretization using geohashes or S2 cells for this purpose. As shown in Figure 2, let $< S, A, B, C, D, T >$ and $< S', A, B, C, T' >$ represent space-discretized shortest routes of rides r and r' respectively.

For relaxing the assumption on matchability, we annotate route points with the time of reaching there if served without any delay. Similar to discretizing space, we discretize time using an appropriate time interval to capture temporal proximity. This gives us a sequence of space-time-discretized points representing the route of each ride. Since we are interested in the cost of the overlapping edges, we represent a ride by the set of space-time-discretized edges in its route which we call its *spatio-temporal set representation*. The cost of a space-time-discretized edge can be approximated as the cost between the mid-points of the corresponding discretized spatial blocks when traveled during the discretized time interval. This approximation works well in practice if we have sufficiently fine space discretization. Representing rides by their set of edges has the added benefit that we get directionality for free. For example, two rides with exactly reversed routes of each other will have no edges in common.

Let *overlap* between two rides be defined as the sum of the costs of the edges in the intersection of their spatio-temporal set representations. Our solution strategy for approximate PMS is via solving approximate Overlapping Match Search:

Definition 2 (Approximate Overlapping Match Search (OMS))**.** *Let P be a pool of rides. Let S be the similarity measure between two rides defined as the overlap between them. The (c, s, k)-approximate Overlapping Match Search (OMS) for $c < 1$ with failure probability f is to construct a data structure over P supporting the following query: given any query ride q, if there exists k rides $r_1, r_2, \ldots, r_k \in P$ such that $S(q, r) \geq s \ \forall r \in \{r_1, r_2, \ldots, r_k\}$, then report some k rides $r'_1, r'_2, \ldots, r'_k \in P$ such that $S(q, r') \geq cs \ \forall r' \in \{r'_1, r'_2, \ldots, r'_k\}$, with probability $1 - f$.*

For simplicity, let us further assume that the cost of each edge is unit. This assumption is somewhat justified if it can be ensured that the routing engine returns routes with approximately equidistant adjecent points. However, this assumption is relaxed in the full version. With this unit cost assumption, the overlap between two rides is simply the car-

Algorithm 1: Algorithm for approximate OMS.

```
Input: A pool $P$ of $n$ rides.
Output: For each ride $r \in P$, a set $S_r \subseteq P$ with
        $|S_r| = k$.
Let $U = 0.75$ (consult (Shrivastava and Li 2015));
for each ride r in P do
    Construct the preprocessing vector representation
      $p_r$ of $r$;
    Normalize the vector $p_r$ to have $||p_r||_2 \leq U$;
    Apply the preprocessing transformation of
      (Shrivastava and Li 2015) on $p_r$ to get $\mathbb{P}(p_r)$ with
      $m = 2$ (consult (Shrivastava and Li 2015));
end
Construct the LSH dataset using cross-polytope LSH of
  (Andoni et al. 2015) with vectors $\mathbb{P}(p_r), \forall r \in P$;
for each ride r in P do
    Construct the query vector representation $q_r$ of $r$;
    Normalize the vector $q_r$ to have $||q_r||_2 = 1$;
    Apply the query transformation of (Shrivastava and
      Li 2015) on $q_r$ to get $\mathbb{Q}(q_r)$ with $m = 2$;
    Construct $S_r$ by retrieving $k$ nearest neighbors of
      $\mathbb{Q}(q_r)$ from the LSH dataset using cross-polytope
      LSH of (Andoni et al. 2015);
end
```

dinality of their intersection. Since cardinality of set intersection can be computed as the inner product between the characteristic vectors of the sets, we represent each ride r by two *spatio-temporal vector representations*. The ambient vector space is a high-dimensional space where each dimension is indexed by a space-time-discretized edge. The reason for using two vector representations is to facilitate relaxing the unit cost assumption.

- *Preprocessing vector representation* (p_r): A vector with all 0's except for along dimensions for which the indexing edge is in the spatio-temporal set representation of the ride, in which case it is the cost of that edge.
- *Query vector representation* (q_r): A vector with all 0's except for along dimensions for which the indexing edge is in the spatio-temporal set representation of the ride, in which case it is 1.

Solving approximate OMS, which is our plan to solve approximate search for potential matches, can now be achieved by solving approximate Maximum Inner Product Search (MIPS) as shown in the next subsection.

The Algorithm

With all the necessary ingredients in place, our algorithm is formally presented in Algorithm 1.

The following theorem, the proof of which is deferred to the full version, establishes the correctness and runtime efficiency of our algorithm.

Theorem 1. *Given a set R of n rides, Algorithm 1 solves (c, s, k)-approximate OMS requiring $O(n^{\rho}(k+\log n)\log k)$ time per query and $O(n^{1+\rho}\log k)$ total space for some $\rho < 1$ depending on c and s. Therefore, the total running time for the algorithm is $O(n^{1+\rho}(k + \log n)\log k)$.*

Experiments

In order to validate our approach, we conducted extensive experimentation comparing our algorithm with heuristic methods and a theoretical optimal. After presenting a brief outline of our experiment setup and the results, we devote this section to a discussion of the results.

Setup

Our experiments were carried out on a machine with a 2.4 GHz Intel i9 processor and 64 GB RAM. Routes were computed using the Open Source Routing Machine (OSRM).

We used the publicly available New York city yellow taxi trip datasets for our experiments (available from https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page). To evaluate our algorithm under varying traffic patterns, we constructed two experiment scenarios: (i) *Morning commute*, consisting of 21310 rides requested on 2016-06-08 (Wed) between 8:00 AM and 9:00 AM local time, and (ii) *Evening commute*, consisting of 19610 rides requested on the same day between 6:00 and 7:00 PM local time. For varying load, we sub-sampled the rides in each scenario at sub-sampling rates of 20%, 40%, 60%, 80% and 100% (full dataset).

For our experiments, we chose *travel duration* to be the cost function. Match feasibility was obtained by enforcing a matching constraint for maximum allowable pickup delay of 10 minutes. We allowed $k = 10$ potential matches per ride to construct the sparse shareability network.

We implemented[1] our algorithm (which we call LSH) along with the three benchmark state-of-the-art heuristic methods described earlier: CLOSEBY, HAVERSINE, and CLOSEBY-HAVERSINE. For the LSH algorithm, a space discretization of geohash-7 and time discretization of 20 minutes were chosen. The CLOSEBY method was implemented using the Ball-Tree algorithm for near-neighbor search using Haversine distance between pickup locations as the distance measure. For the CLOSEBY-HAVERSINE method, 1000 candidate matches were first shortlisted using the CLOSEBY heuristic, out of which k potential matches were chosen using the HAVERSINE heuristic. An utility-optimal approach was also implemented by an exhaustive search for k feasible potential matches with highest utility with a given ride (which was computationally extremely expensive). The total matching utilities achieved by various methods were evaluated against this optimal.

Given a shareability network, the total matching utility was computed by first computing the true utlility for each edge of the shareability network using OSRM, and then computing an optimal non-bipartite matching on this network by solving an Integer Linear Program.

[1]Code available at https://github.com/chinmoy-dutta/ridematch_lsh

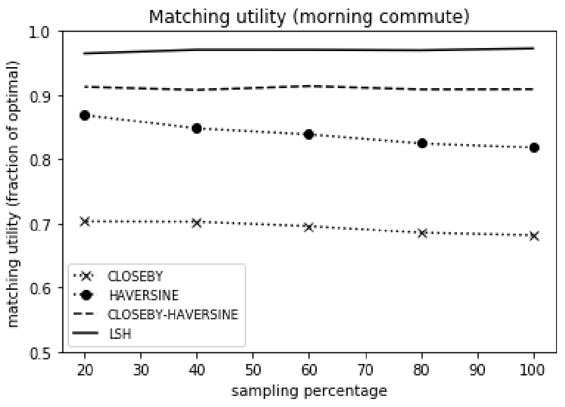

Figure 3: Matching utility as a fraction of the optimal for the morning commute scenario under varying load.

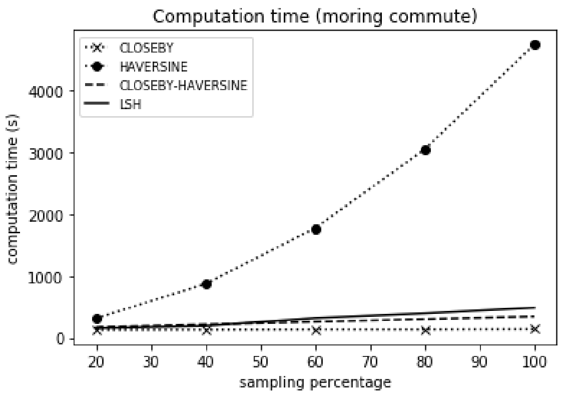

Figure 4: Shareability network computation time for the morning commute scenario under varying load.

Results

Matching utilities obtained and the time to compute the shareability network using different methods under varying load for the morning commute scenario are presented in Figure 3 and Figure 4 respectively. Results for the evening commute scenario are very similar and deferred to the full version.

Discussion

For each scenario and each load factor, the optimal matching utility was found to be around 40% of the total ride cost, suggesting huge potential for sharing rides and reducing cost.

Out of the four methods, CLOSEBY is the fastest as it simply needs to find near-neighbors based on Haversine distance between pickup locations. Not surprisingly, it performs the worst in terms of matching utility. HAVERSINE is impractically slow, being required to perform an exhaustive search on a quadratic search space, even if without routing calls. Perhaps surprisingly, even with this exhaustive search, its performance in terms of matching utility is worse than that of CLOSEBY-HAVERSINE. This is because an exhaustive search finds matches that may have better Haversine overlap with a given ride but not feasible to match with it due to the matching constraint. CLOSEBY-HAVERSINE is better at filtering those out.

In each scenario and for each load factor, LSH consistently outperforms all the heuristic methods. It achieves about 6 parentage points higher matching utility compared to the next best method, CLOSEBY-HAVERSINE. This can generate significant cost savings that can be shared by the riders, the drivers and the platform, and can contribute to significant reduction in traffic congestion, resource consumption and air pollution. It is important to note that CLOSEBY-HAVERSINE misses this opportunity not because of restricted computation time or limited shortlisted candidate matches. This can be confirmed by noting that the matching utility achieved by this heuristic stays almost constant at slightly over 90% of optimal even when load decreases to 20% but the number of shortlisted candidate matches stays constant at 1000. This means the heuristic cannot achieve additional matching utility even when it searches over a higher fraction of total rides. The reason CLOSEBY-HAVERSINE consistently misses out on achievable matching utility is that it relies on Haversine overlap between rides and thus suffers from its inherent lack of knowledge of the underlying physical road network.

We further note that the computation time for LSH, although slightly larger than CLOSEBY-HAVERSINE for higher loads, is still very practical. Moreover, the algorithm is highly parallelizable, and the computation time of about 300 secs can be brought down to under 30 secs with 10 parallel workers even for a gigantic match pool of about 20000 rides. Such parallelization is a very common practice for production grade on-demand mobility platforms. One can also employ dimensionality reduction and incremental computation techniques mentioned in the next section for further improving efficiency in real-time operation.

Alternate routes. Having access to alternate routes for a ride helps the LSH algorithm to find better matches. A ride often has multiple routes with almost similar route costs. A feasible match may have a very good overlap with one of the routes that is not the minimum cost one. The computation time does increase with number of alternate routes, however. In our experiments, we used up to one additional alternate route whenever available (available for about 40% of rides). Figure 5 compares LSH performance with exactly one route and up to two routes.

Discretization. It is important to choose the space discretization for the LSH algorithm carefully. Choosing a discretization too coarse improves computation time as the dimension of the vector space decreases. But it hurts matching utility by finding spurious route overlaps (edges too dissimilar might look the same with coarser space discretization) as well as missing some genuine ones (endpoints of an overlapping edge might fuse with coarser space discretization). Choosing a discretization too fine increases the computation time significantly. In some cases, it might also hurt matching utility by causing the algorithm to miss some useful over-

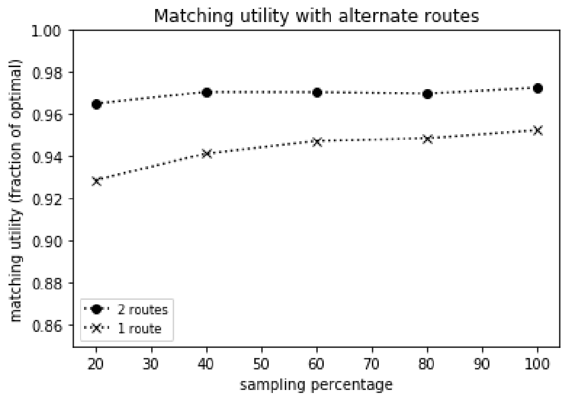

Figure 5: Matching utility as a fraction of the optimal with alternate routes.

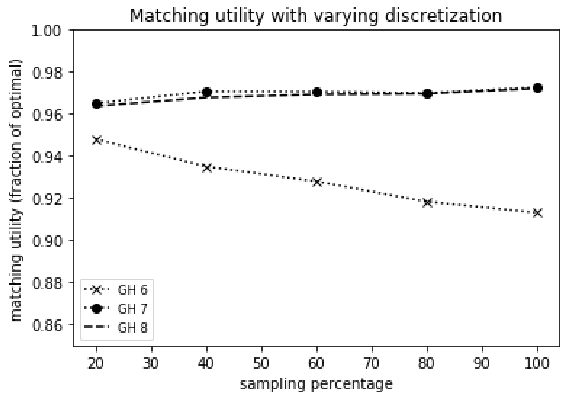

Figure 6: Matching utility as a fraction of the optimal with varying geohash discretization.

laps. Figure 6 compares performance with different space discretizations. In general, choosing a space discretization proportional to the typical length of a route edge works well (e.g. geohash-7 in our case). For time discretization, it is best to choose a reasonably large interval proportional to the maximum allowable delay. This avoids missing feasible matches, but can lead to some infeasible matches which can later be filtered out.

Optimizations and Extensions

In this section, we present several useful optimizations and extensions for a real-world implementation of our algorithm.

Optimizations

Number of tables. Increasing the number of tables in the LSH data structure improves success probability for finding nearest neighbors but also increases the space requirement and the query time. It is advisable to first choose a suitable value based on space availability and the dataset size.

Number of hash functions. Increasing the number of hash functions per table causes fewer hash collisions improving query time but also decreases the success probability for finding nearest neighbors and necessitates larger number of tables. This parameter should be chosen roughly equal to the logarithm of the dataset size and further tuning should be done via parameter search jointly with the number of query probes, as explained next.

Multi-probe query. It is possible to query a table multiple times using multi-probe query strategies to improve success probability without increasing the number of tables. The total number of probes across all table per query should be chosen jointly with number of hash functions per table. This can be achieved by iterating over the latter, and for each value doing a binary search over the former to achieve desired success probability and query time efficiency.

Dimension reduction. The ride representations are extremely sparse. This is because most rides happen around busy areas while the remote areas only see few rides. Thus, we can boost runtime efficiency significantly without hurting success probability for finding good potential matches by employing dimensionality reduction techniques to transform the ride vectors to a lower dimensional space before storing them in the LSH data structure.

Normalization. Centering the ride vectors along each dimension boosts success probability significantly.

Useful Properties and Extensions

General linear cost function. Our algorithm works for any linear route cost function. This is useful as the cost of serving a ride is often estimated by a linear combination of travel duration, travel distance and other factors.

Time varying cost function. Our approach can naturally adapt to time-varying route costs. For example, travel durations vary significantly at different times, a smart transportation system might impose time-varying tolls to regulate traffic flow. In contrast, none of the heuristic methods can adapt to these changes intelligently.

Incremental computation. Our algorithm can be implemented to do incremental computation of the shareability network where the LSH data structure is persisted, and gets updated as new rides come (by adding them to the data structure) and rides get matched (by replacing them with the combined driver route). Dynamic LSH schemes are excellent candidates for such an implementation.

Hybrid rideshare. A shared mobility platform can have a ride pool with on-demand as well as pre-scheduled rides. Since our algorithm employs a spatio-temporal search, it can intelligently match rides from such hybrid pools respecting matching constraints. Heuristic methods cannot make such intelligent choices.

Ethics Statement

This paper develops and evaluates a novel algorithm to enable sharing of rides efficiently in real-time while dealing with a vast urban-scale ride pool of tens of thousands of rides. We devote this section to discussing the societal implications of our work.

Efficient mobility is not only important for economic development of cities and ensuring quality of life for its population, but equitable access to affordable, convenient and efficient mobility is also key to providing equal access to education, employment, healthcare and other civic facilities. The senior and elderly population, who often live on limited income and have restricted mobility options, especially need convenient point-to-point on-demand transportation. However, mobility has become an immense technological, societal and environmental challenge for cities worldwide, especially the growing and densely-populated ones (Arnott and Small 1994; Caiazzo et al. 2013; Pant and Harrison 2013). The challenge arises from conflicting objectives of providing convenient and flexible on-demand point-to-point transportation; ensuring it is efficient; ensuring it is economical and affordable for all; ensuring it can be supported by the available transportation infrastructure which mostly has limited, if any, potential to grow; and making it environmentally sustainable. The magnitude of the cost of vehicular traffic congestion imposed on cities can be understood by noting that (Arnott and Small 1994) estimated it to be about USD 60 Billion for the 83 largest urban areas of US alone measured in terms of wasted time and fuel, and World Health Organization estimated more than 1 million deaths annually worldwide are caused by air pollution which, in turn, is largely caused by traffic congestion (Caiazzo et al. 2013).

High-capacity shared mobility has tremendous potential to cater to the societal need for mobility while effectively addressing the infrastructural and environmental concerns. By enabling efficient urban-scale ride matching in real-time, this work can foster convenient and flexible point-to-point on-demand mobility while ensuring it is highly resource-efficient resulting from the high level of sharing. This can further ensure affordability creating equity among all sections of society in access to convenient and flexible mobility; reduction in traffic congestion saving wasted time in traffic for individuals and businesses; easing of the burden on transportation infrastructure of cities; and environmentally sustainability.

We also discuss the scope and nature of this work in relation to several recent studies that have concluded that ridesharing platforms, in fact, increase traffic congestion. Furthermore, some studies have also found the cost of using ridesharing services to be even larger than personal car ownership, and its usage to be highly skewed towards more affluent and higher educated segments of the population, raising questions on its ability to provide affordable and convenient mobility for all. We note that an increase in ride demand as a result of convenience of hailing a ride, roaming empty vehicles, and sparingly used vehicle capacity are cited as causes for increased traffic and costs in these studies. Our work precisely tackles some of these very causes by filling up empty vehicle seats and fostering an optimal usage of vehicle capacity. It, therefore, can not only help reduce traffic congestion and air pollution, but also reduce the cost of rides making them more affordable.

Moreover, the methodology and its extensions presented in this work are not restricted to any particular form of on-demand shared mobility service. It is applicable to the operations of any mobility service, platform or transportation agency that desires to provide smart, flexible and convenient on-demand shared mobility while ensuring it is resource efficient and affordable. We believe this work fits within the range of growing urban mobility innovations that promise to tackle the mobility challenge using principled approaches leading to efficient, socially useful technologies.

References

Agatz, N. A. H.; Erera, A. L.; Savelsbergh, M. W. P.; and Wang, X. 2011. Dynamic ride-sharing: a simulation study in metro Atlanta. *Transportation Research Part B: Methodological* .

Al-Abbasi, A. O.; Ghosh, A.; and Aggarwal, V. 2019. Deep-Pool: Distributed Model-Free Algorithm for Ride-Sharing Using Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems* .

Alonso-Mora, J.; Samaranayake, S.; Wallar, A.; Frazzoli, E.; and Rus, D. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences USA* .

Alonso-Mora, J.; Wallar, A.; and Rus, D. 2017. Predictive routing for autonomous mobility-on-demand systems with ride-sharing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Amirmahdi, T.; and Masoud, N. 2020. Trip-based graph partitioning in dynamic ridesharing. *Transportation Research Part C: Emerging Technologies* .

Andoni, A.; Indyk, P.; Laarhoven, T.; Razenshteyn, I.; and Schmidt, L. 2015. Practical and optimal LSH for angular distance. In *Proc. of the 29th Annual Conference on Neural Information Processing Systems*.

Arnott, R.; and Small, K. 1994. The Economics of Traffic Congestion. *American Scientist* .

Ashlagi, I.; Burq, M.; Dutta, C.; Jaillet, P.; Saberi, A.; and Sholley, C. 2019. Edge Weighted Online Windowed Matching. In *Proc. of the 20th ACM Conference on Economics and Computation*.

Balardino, A. F.; and Santos, A. G. 2015. Heuristic and exact approach for the close enough ridematching problem. In *Proc. of the 15th International Conference on Hybrid Intelligent Systems*.

Bei, X.; and Zhang, S. 2018. Algorithms for Trip-Vehicle Assignment in Ride-Sharing. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*.

Caiazzo, F.; Ashok, A.; Waitz, I. A.; Yim, S. H. L.; and Barrett, S. R. H. 2013. Air pollution and early deaths in the United States. Part I: Quantifying the impact of major sectors in 2005. *Atmospheric Environment* .

Carrion, C.; and Levinson, D. 2012. Value of travel time reliability: a review of current evidence. *Transportation Res Part A* .

Dickerson, J. P.; Sankararaman, K. A.; Srinivasan, A.; and Xu, P. 2018. Allocation Problems in Ride-Sharing Platforms: Online Matching with Offline Reusable Resources. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*.

Dutta, C.; and Sholley, C. 2017. Online Matching in a Ridesharing Platform. 3rd Marketplace Innovation Workshop, available at https://arxiv.org/pdf/1806.10327.pdf.

Friedrich, M.; Hartl, M.; and Magg, C. 2018. A modeling approach for matching ridesharing trips within macroscopic travel demand models. *Transportation* .

Georgescu, B.; Shimshoni, I.; and Meer, P. 2003. Mean shift based clustering in high dimensions: A texture classification example. In *Proc. of the 9th IEEE International Conference on Computer Vision*.

Gionis, A.; Indyk, P.; and Motwani, R. 1999. Similarity search in high dimensions via hashing. In *Proc. of the 25th International Conference on Very Large Data Bases*.

Huang, Q.; Ma, G.; Feng, J.; Fang, Q.; and Tung, A. K. H. 2018a. Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In *Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Huang, Z.; Kang, N.; Tang, Z. G.; Wu, X.; Zhang, Y.; and Zhu, X. 2018b. How to match when all vertices arrive online. In *Proc. of the 50th Annual ACM Symposium on Theory of Computing*.

Indyk, P.; and Motwani, R. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. of the 30th Annual ACM Symposium on Theory of Computing*.

Jgou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* .

Koide, S.; Tadokoro, Y.; and Yoshimura, T. 2015. SNT-index: Spatio-temporal index for vehicular trajectories on a road network based on substring matching. In *Proc. of the 1st International ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics*.

Lesmana, N. S.; Zhang, X.; and Bei, X. 2019. Balancing Efficiency and Fairness in On-Demand Ridesourcing. In *Proc. of the 33rd Annual Conference on Neural Information Processing Systems*.

Li, H.; Chan, T. N.; Yiu, M. L.; and Mamoulis, N. 2017. FEXIPRO: Fast and exact inner product retrieval in recommender systems. In *Proc. of the ACM International Conference on Management of Data*.

Ma, S.; Zheng, Y.; and Wolfson, O. 2013. T-Share: A large scale dynamic taxi ridesharing service. In *Proc. of the 29th IEEE International Conference on Data Engineering*.

Masoud, N.; and Jayakrishnan, R. 2017. A Real-Time Algorithm to Solve the Peer-to-Peer Ride-Matching Problem in a Flexible Ridesharing System. *Transportation Research Part B* .

Pant, P.; and Harrison, R. M. 2013. Estimation of the contribution of road traffic emissions to particulate matter concentrations from field measurements: a review. *Atmospheric Environment* .

Santi, P.; Resta, G.; Szell, M.; Sobolevsky, S.; Strogatz, S. H.; and Ratti, C. 2014. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences* .

Schreieck, M.; Safetli, H.; Siddiqui, S. A.; Pflgler, C.; Wiesche, M.; and Krcmar, H. 2016. A Matching Algorithm for Dynamic Ridesharing. *Transportation Research Procedia* .

Shah, S.; Lowalekar, M.; and Varakantham, P. 2020. Neural approximate dynamic programming for on-Demand ride-Pooling. In *Proc. of the 34th AAAI Conference on Artificial Intelligence*.

Shrivastava, A.; and Li, P. 2014. Asymmetric LSH (ALSH) for sublinear time Maximum Inner Product Search (MIPS). In *Proc. of the International Conference on Neural Information Processing Systems*.

Shrivastava, A.; and Li, P. 2015. Improved asymmetric locality sensitive hashing (ALSH) for Maximum Inner Product Search (MIPS). In *Proc. of the 31st Conference on Uncertainty in Artificial Intelligence*.

Simonetto, A.; Monteil, J.; and Gambella, C. 2019. Real-time city-scale ridesharing via linear assignment problems. *Transportation Research Part C: Emerging Technologies* .

Spieser, K.; Samaranayake, S.; Gruel, W.; and Frazzoli, E. 2016. Shared-vehicle mobility-on-demand systems: A fleet operators guide to rebalancing empty vehicles. *Transportation Research Board 95th Annual Meeting* .

Spring, R.; and Shrivastava, A. 2017. Scalable and sustainable deep learning via randomized hashing. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Sundaram, N.; Turmukhametova, A.; Satish, N.; Mostak, T.; Indyk, P.; Madden, S.; and Dubey, P. 2013. Streaming similarity search over one billion tweets using parallel locality sensitive hashing. *Proceedings of the VLDB Endowment* .

Yu, X.; and Shen, S. 2020. An Integrated Decomposition and Approximate Dynamic Programming Approach for On-Demand Ride Pooling. *IEEE Transactions on Intelligent Transportation Systems* .

Zhao, B.; Xu, P.; Shi, Y.; Tong, Y.; Zhou, Z.; and Zeng, Y. 2019. Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach. In *Proc. of the 33rd AAAI Conference on Artificial Intelligence*.

Zheng, L.; Chen, L.; and Ye, J. 2018. Order Dispatch in Price-Aware Ridesharing. *Proceedings of the VLDB Endowment* .