When Hashing Met Matching: 
Efficient Search for Potential Matches in Ride Sharing

Chinmoy Dutta  
Lyft, Inc.  
San Francisco

Abstract

We study the problem of matching rides in a ride sharing platform. Such platforms face the daunting combinatorial task of finding potential matches for rides from a matching pool of tens of thousands of rides very efficiently while retaining near-optimality compared to an exhaustive search. We formalize this problem and present a novel algorithm for it based on the beautiful theory of locality sensitive hashing for Maximum Inner Product Search (MIPS). The proposed algorithm can find $k$ (can be practically a constant for ride sharing platforms) potential matches for a given ride from a pool of $n$ rides in sub-linear time $O(n^{\rho} (k + \log n) \log k)$ for $\rho < 1$, which is significant saving compared to an exhaustive search in the pool requiring $O(n^2)$ time. The space requirement for our algorithm is $O(n^{1+\rho} \log k)$.

We show that the set of $k$ potential matches include the near-optimal ones with high probability. Implementation of our algorithm could efficiently find near-optimal set of potential matches with high probability from a pool of thousands of real rides.

1 Introduction

We consider the problem of matching riders for sharing rides in a ride sharing platform. In recent times, the rapid advancement and widespread availability of mobile computing have significantly uplifted the face of urban mobility and transportation. A large part of this change is due to the impact created by ride-sharing platforms. These platforms enable anyone to get a ride anytime and anywhere, while also being able to do marketplace optimization to ensure lower estimated time to pickup for riders as well as better time utilization and earnings for drivers.

The major advantage of these platforms, however, is the ride-sharing possibility that they enable. With this feature, riders can choose to share their vehicle with other riders heading in the same direction. There are tremendous socio-economical and environmental advantages to this. It lets the riders share the cost of their rides, thereby providing them an economical and affordable means of transportation. Riders can also take advantage of incentives like use of high occupancy lanes. It also eases the burden on the transportation infrastructure of cities and helps reduce traffic congestion, specially during heavy commute hours. Reducing traffic congestion in turn helps in cutting down commute times, thus providing great economic benefit to individuals, businesses and institutions by reducing lost time and increasing productivity. Perhaps even more importantly, ride-sharing provides great environmental benefits by reducing air pollution as a result of reduced number of vehicles on roads, thereby greatly reducing our carbon footprint.

Sharing rides with others, however, does incur some costs to the riders like increased detours and travel times as well as some loss of privacy and convenience. The task of the ride-sharing platform is to intelligently trade off these costs against the aforementioned benefits. Therefore, the problem of matching rides that incur minimal detour and cause minimal inconvenience to the riders while maximizing marketplace efficiency is of paramount importance for a ride sharing platform.

A standard and well-studied approach to the ride-matching problem is to first generate all “feasible” potential matchings of rides in the match pool. A potential matching is deemed feasible if all the rides involved can be fulfilled by a shared vehicle without violating the (platform defined) user experience constraints for any. Each generated potential matching is then scored for its matching utility which is measured by the amount of cost savings the matching can generate. Lastly, a constraint optimization problem, typically a Mixed Integer Program, is solved to choose an optimal
subset of matchings to commit from the set of all potential matchings. The constraints of the program guarantee that each ride is included in exactly one committed matching. (An unmatched ride can be considered an 1-matching by itself.) The objective of the program is to maximize the sum of the utilities derived from all the committed matchings. An approach very similar to this was also suggested in [AMSW+17].

There are two major bottlenecks with the above approach in the step of generating all feasible potential matchings: the inefficiency in its runtime and the inefficiency in the number of potential matchings generated. A gigantic number of potential matchings puts a lot of stress on later stages of scoring (which might involve requiring information from upstream services like routing) and solving a mixed integer program. These bottlenecks make the whole system neither scalable nor practical.

A typical approach employed in literature [AMSW+17] to make the generation step efficient in its runtime is to iteratively generate the feasible potential matchings of increasing sizes, while exploiting branch-and-bound methods to limit search for higher cardinality potential matchings by requiring that all subsets of a feasible matching must also be feasible. We still need to compute the feasible potential 2-matchings however, which may require comparing each ride to every other which is $O(n^2)$ for a match pool of size $n$. In a large ride-sharing platform, it is common to have thousands of ride requests in the match pool for a dense region at peak times. Such platforms also employ features like batching of arriving ride requests, swapping already formed matches to form new ones etc, which further thicken the match pool. To generate all feasible potential 2-matchings by a quadratic complexity exhaustive search is therefore ruled out.

For a given ride, we will call another ride with which it can form a potential 2-matching as a potential match. Earlier works proposed mostly heuristic ways to limit the search for potential matches of a ride to deal with the combinatorial challenge mentioned above. For example, the authors in [AMSW+17] observed that a feasible potential matching can only be formed between rides that have at least one common available driver who can pick all the rides respecting their pickup time constraints. This allows one to restrict the search for the potential matches of a ride to the ones originating in the vicinity of its own pickup.

While these heuristic methods are helpful in making the search for potential matches more efficient, they do not fully address the problem. For example, It is not uncommon to find large number of co-located riders (e.g. outside train and other public transport stops) requesting shared rides at roughly the same time (e.g at times trains arrive). The above heuristic to take advantage of spatial proximity of ride pickup locations breaks down in these cases. Moreover, it may be advantageous to be able to find potential matches with far-away pickup locations (e.g. if the potential match is a ride scheduled for a later pickup). Excluding such potential matches may hurt the optimality of the matchings committed.

Our contribution

In this work, we propose a novel approach to tackle this combinatorial challenge. The idea is to restrict the search for potential matches not only by feasibility but also the matching utility. Requiring potential matches to have high matching utility with the given ride helps us make the search for potential matches efficient. Moreover, this also solves the second problem of generating huge number of potential matchings. However, this requirement does not hurt optimality of the solution much as the low utility matchings excluded are not likely to be picked by the constraint optimizer.

In order to efficiently find the high utility potential matches, we use the beautiful theory of locality sensitive hashing (LSH) for Maximum Inner Product Search (MIPS). At a high level, our algorithm works by finding suitable representations for rides accompanied by a similarity measure that respect the utility of matching two rides together. We then use the theory of LSH to find $k$ (can be practically a constant for ride sharing platforms) near neighbors (according to the above similarity measure) of a given ride very efficiently in time $O(n^\rho (k + \log n) \log k)$ for a small $\rho < 1$. The space requirement for our algorithm is $O(n^{1+\rho} \log k)$. This set of $k$ neighbors includes the near-optimal potential matches of the given ride with high probability. This is a significant saving compared to the $O(n)$ time required for an exhaustive search for potential matches of a given ride.

In order to make this method work, we first represent rides as vectors in a suitable high dimensional ambient space. We provide a similarity metric for this space between (the vector representations of) two rides based on their physical route with the following property: a small similarity signify matching the two together has low utility; while a large
similarity signify their matching has high utility. Next, we use asymmetric LSH construction of [SL15] for MIPS to obtain a locality sensitive data structure for storing the ride vectors which allows efficient search for near neighbors. Exploiting the property of our similarity measure mentioned above, this lets us efficiently find high utility potential matches.

We also experimentally validated our algorithm on real ride data from the ride sharing platform Lyft. Our experiment results show that our algorithm can very efficiently find near-optimal set of potential matches.

Related literature

Recently, there has been a lot of research interest in ride sharing and associated challenges. Much of the work has been done in transportation-related literature. Many of the studies related to fleet management considers ride-sharing without pooling requests, e.g [PSFR12, ZP15, SRS+14, SSGF16]. A recent study by Santi et al [SRS+14] showed about 80% of rides in Manhattan can be shared by two riders. Heuristic-based solutions to matching problems were studied in [AESW11, MZW13]. There has also been a lot of research interest in studying autonomous ride-sharing systems [PSFR12, SSGF16, dACA16]. Alonso-Mora et al. [AMSW+17] studied real-time high-capacity ride-sharing and route generation.

This nearest neighbor search (NNS) is a problem of major importance in several areas such as data compression, databases and data mining, information retrieval, image and video databases, machine learning, pattern recognition, statistics and data analysis. Approximate nearest neighbors search based on locality sensitive hashing (LSH) was introduced in [IM98, GIM99] to deal with the "curse of dimensionality", which had so far prevented efficient nearest neighbor search for high-dimensional data with low preprocessing time and sublinear query time. The LSH based near neighbors search techniques are widely used and have been very successful in industrial practice as well as applied research [CDF+01, HGI09, GSM03, Yan01, Buh01, BT01, Buh02, OMS+02, JDS11, SS14, STS+13], partly owing to the fact that they work efficiently with high dimensional data and are also highly parallelizable. However, to the best of our knowledge, use of locality sensitive hashing has not yet been introduced in applications related to ride sharing.

Authors in [IM98, HPIM12, GIM99] provided LSH functions for the case when the points live in binary Hamming space with provably sublinear query time. They also showed that it is possible to extend the algorithm to the $l_2$ norm, by embedding $l_2$ space into $l_1$ space, and then $l_1$ space into Hamming space. In a followup work [DIIM04], the authors introduced LSH functions that work directly in Euclidean space resulting in somewhat faster running time. This algorithm is the basis of the $E^2 LSH$ package [AI04] for high-dimensional similarity search, which has been used in several applied scenarios. A different family of LSH functions for $l_2$ was obtained in [AI06]. The influential work of Charikar [Cha02] introduced hyperplane LSH that works well in practice. Multiprobe LSH was introduced in [LJW+07] which leads to a significantly reduced memory footprint for the hyperplane LSH.

The recent works [AR15, AINR14] on data-dependent hashing algorithms achieve performance better than the classic LSH algorithms for both Hamming and Euclidean data. Authors in [AIL+15] showed that cross-polytope LSH can match the theoretical guarantee of Spherical LSH [AR15] while, when combined with additional techniques, can also give better experimental results than the hyperplane LSH [Cha02]. In order to make the cross-polytope LSH competitive in practice with the multiprobe hyperplane LSH, a novel multiprobe scheme for the cross-polytope LSH was also proposed in [AIL+15].

The Maximum Inner Product search (MIPS) is a fundamental problem with variety of applications such as matrix factorization based Recommendation systems [KRS12, KBV09, LCYM17, SRJ05], multi-class label prediction [DRS+13, JK09], structural SVM [JFY09], and Deep Learning [SS17]. Due to the difficulty of this problem in high dimensional space, the approximate MIPS has attracted extensive studies. However, since inner product is not a metric, LSH schemes cannot be directly adapted.

The concept of Asymmetric LSH (ALSH) was proposed and the first ALSH scheme named L2-ALSH for MIPS with provable sub-linear query time was presented in [SL14]. L2-ALSH converts MIPS into NNS and then solves NNS by E2LSH [DIIM04]. In a followup work, [SL15] developed another asymmetric transformation named Sign-ALSH which reduces MIPS to Maximum Cosine Similarity Search (MCSS), and then solves the MCSS by SimHash [Cha02]. Authors of [BFGB+14] proposed an exact asymmetric transformation named XBOX which converts MIPS into NNS and solves NNS by PCA-Tree. [NS15] proposed a symmetric transformation named Simple-LSH which reduces MIPS to MCS search and uses SimHash [Cha02] to solve MCS. A very recent work [] proposed a novel Asymmetric LSH
scheme based on Homocentric Hypersphere partition (H2-ALSH) for high-dimensional MIPS and claimed it significantly outperforms the state-of-the-art schemes, such as L2-ALSH, XBOX, Sign-ALSH, and Simple-LSH.

Organization

We provide some preliminaries as well as formalize our problem in Section 2. We then proceed to present our algorithm in Section 3. Experimental results are presented in Section 4. We conclude in Section 5 with some discussion for future improvements.

2 Preliminaries

Locality sensitive hashing and approximate NNS

Define a ball for a similarity measure D on a domain S as \( B(q, r) := \{ p : D(q, p) \geq r \} \). An LSH family is defined as:

Definition 1. A family \( \mathcal{H} = \{ h : S \rightarrow U \} \) is called \((r, cr, p_1, p_2)\)-sensitive for space S with similarity measure D if for any \( p, q \in S \):

- if \( p \in B(q, r) \) then \( \Pr_h[h(q) = h(p)] \geq p_1 \).
- if \( p \notin B(q, cr) \) then \( \Pr_h[h(q) = h(p)] \leq p_2 \).

In order for a locality-sensitive hash (LSH) family to be useful, it has to satisfy inequalities \( p_1 > p_2 \) and \( c < 1 \). An LSH family can be used to solve the approximate near neighbor problem.

Definition 2. The \((c, r)\)-approximate nearest neighbor problem \((c, r)\)-NN for \( c < 1 \) with failure probability \( f \) is to construct a data structure over a set of points \( P \) in space \( S \) with similarity measure \( D \) supporting the following query: given any query point \( q \in S \), if \( p \in B(q, r) \) for some \( p \in P \), then report some \( p' \in P \cap B(q, cr) \), with probability \( 1 - f \).

The following is a restatement of Theorem 3.4 in [HPIM12].

Theorem 1 (Restatement of Theorem 3.4 in [HPIM12]). Given a \((r, cr, p_1, p_2)\)-sensitive family \( \mathcal{H} \) for a space \( S \), where \( p_1, p_2 \in (0, 1) \) and \( c < 1 \) and let \( \rho = \log(1/p_1)/\log(1/p_2) < 1 \). Then there exists a data structure for \((c, r)\)-NN over a set \( P \subset S \) of at most \( n \) points with constant failure probability requiring \( O(n^\rho \log n) \) query time and \( O(n^{1+\rho}) \) space.

The approximate near neighbor data structure can be used to solve the approximate nearest neighbor problem.

Definition 3. The \( c \)-approximate nearest neighbor problem \((c)\)-NN for \( c < 1 \) with failure probability \( f \) is to construct a data structure over a set of points \( P \) in space \( S \) with similarity measure \( D \) supporting the following query: given any query point \( q \in S \), report a point \( p' \in P \) such that \( D(q, p') \geq c \max_{p \in P} D(p, q) \) with probability \( 1 - f \).

Theorem 2 (Restatement of Theorem 2.9 in [HPIM12]). Let \( c < 1, f \in (0, 1), \) and \( \gamma \in (1/n, 1) \) be prescribed parameters. Assume that we are given a data structure for the \((c, r)\)-approximate near neighbor with failure probability \( f \) over a set \( P \) of at most \( n \) points that uses space \( S(n, c, f) \), and has query time \( Q(n, c, f) \). Then there exists a data structure for answering \( c(1 + O(\gamma)) \)-approximate nearest neighbor queries that uses \( O(S(n, c, f)/\gamma \log^2 n) \) space and has query time \( O(\log n)Q(n, c, f) \) with failure probability \( O(f \log n) \).

Asymmetric LSH

The notion of asymmetric LSH was defined in [SL14].

Definition 4. A family \( \mathcal{H} \) along with the two vector functions \( Q: R^d \rightarrow R^d \) (Query Transformation) and \( P: R^d \rightarrow R^d \) (Preprocessing Transformation), is called \((r, cr, p_1, p_2)\)-sensitive if for a given query \( p, q \in R^d \):

4
• if \( D(q,p) \geq r \) then \( \Pr_{h \in H}[h(Q(q))] = h(P(x))] \geq p_1 \),
• if \( D(q,p) \leq cr \) then \( \Pr + h \in H[h(Q(q))] = h(P(x))] \leq p_2 \).

Just like with regular LSH, it is also possible to solve approximate near neighbor problem using asymmetric LSH [SL14].

**Theorem 3** (Theorem 2 of [SL14]). Given a family \( \mathcal{H} \) of \((r, cr, p_1, p_2)\)-sensitive asymmetric LSH functions along with the associated query and preprocessing transformations \( Q \) and \( P \), where \( p_1, p_2 \in (0, 1) \) and \( c < 1 \) and let \( \rho = \log(1/p_1)/\log(1/p_2) < 1 \). Then one can construct a data structure for \((c, r)\)-NN over a set \( P \subset \mathbb{R}^d \) of at most \( n \) points with constant failure probability requiring \( O(n^{\rho} \log n) \) query time and \( O(n^{1+\rho}) \) space.

**Approximate MIPS**

**Definition 5.** Given a collection \( P \subset \mathbb{R}^d \) of size \( n \) and an input query point \( q \in \mathbb{R}^d \), the \( c \)-approximate Maximum Inner Product Search (MIPS) problem is to find \( p' \in P \) such that

\[
q^T p' \geq \max_{p \in P} q^T p
\]

[SL15] established a beautiful connection between MIPS and Maximum Cosine Similarity Search (MCSS) (which they called correlation-NNS in their paper) by defining the following query and preprocessing transformations from \( \mathbb{R}^d \) to \( \mathbb{R}^{d+m} \).

\[
P(x) := [x; 1/2 - ||x||^2_2; 1/2 - ||x||^4_2; \ldots; 1/2 - ||x||^{2m}_2]
\]

\[
Q(x) := [x; 0; 0; \ldots; 0]
\]

Using the SimHash LSH functions for MCSS of [Cha02] (which are basically signed random projections), the above transformations enabled the authors of [SL15] to construct asymmetric LSH functions for MIPS, which they called Sign-ALSH.

**Potential match finding problem**

Let \( p_1, p_2, \ldots, p_l \) be points on earth and \( C(<p_1, p_2, \ldots, p_l>) \) denote the 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 pickup and dropoff points respectively. The cost of the ride \( r \), denoted by \( C(r) \) abusing notation, is defined as \( C(<r_s, r_t>) \).

The utility of matching two rides comes from the cost saving achieved by matching 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: \( C(\{r, r'\}) = \min\{C(<r_s, r'_s, r_t, r'_t>), C(<r_s, r'_s, r'_s, r_t>), C(<r'_s, r_s, r_t, r'_t>), C(<r'_s, r_s, r'_s, r'_t>)\} \). The matching utility function \( U \) for matching ride \( r \) with \( r' \) is then defined as:

\[
U(\{r, r'\}) = C(r) + C(r') - C(\{r, r'\})
\]

**Definition 6.** Let \( n > k > i_1 > i_2 > 0 \). The \((n, k, i_1, i_2)\)-approximate Potential Match Search (PMS) problem for ride sharing is the following. Let \( R \) be a set of \( n \) rides. For each ride \( r \in R \), output a set of rides \( S_r \subset R \) with \( |S_r| = k \) such that \( S_r \) contains at least \( i_2 \) of the top \( i_1 \) rides in \( R \) with highest matching utility for matching with \( r \).

### 3 Algorithm for approximate PMS

We first find a suitable representation for rides in an ambient high-dimensional space and define a similarity measure between two rides that approximately captures the matching utility between them. We then use the asymmetric LSH construction of [SL15] (for MIPS) to obtain our algorithm for approximate PMF.
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 $r'$ is denoted by $s'$ and $t'$ respectively. Let $<s,a,b,c,t>$ and $<s',a',b',c',d',t'>$ denote the routes of ride $r$ and $r'$ respectively.

Intuitively, since points $a,b,c$ are close to points $a',b',c'$, the matching utility which is the cost saving by matching $r$ and $r'$ is approximately the cost $C(<a,b>)$ of the route segment $(a,b)$ plus the cost $C(<b,c>)$ of the route segment $(b,c)$. In order to capture the notion of spatial proximity of points, we discretize space and represent each point of the route by its discretization. We can use discretization using S2 cells or geohashes for this purpose. Let the discretized value for points $a$ (and also $a'$) be $A$, and so on, as shown in Figure 2. The discretized route for rides $r$ and $r'$ are $<S,A,B,C,T>$ and $<S',A,B,C,D,T'>$ respectively.

Since we are interested in the cost of the overlapping segments, we represent rides by the set of edges in their routes instead of the sequence of nodes. This is equivalent of forming 2-shingles; shingling is a popular technique in document similarity search. Note that representing rides by set of edges has the additional benefit that we get directionality for free. For example, two rides with exactly reversed routes of each other will have no edge in common.

Representing ride $r$ and $r'$ with the edge sets $\{SA,AB,BC,CT\}$ and $\{S'A,AB,BC,CD,DT'\}$ respectively, we observe that utility of matching them is roughly the sum of the costs of the edges in their intersection. Here, the cost of an edge between two adjacent discretized nodes can be approximated as the cost between their mid-points. Note that, for this to work correctly, we must keep adjacent discretized nodes along the route as edges.

In view of the above discussion, we propose to solve approximate PMS by solving the approximate Overlapping Match Search (OMS) problem:

**Definition 7.** Let $R$ be a set of rides and $D$ be the similarity measure between two rides defined as the sum of the
Figure 2: Two rides $r$ and $r'$. The discretized route for $r$ is $< S, A, B, C, T >$ and for $r'$ is $< S', A, B, C, D, T' >$.
costs of the edges in the intersection of the set representations of the two rides. The \((c, s)\)-approximate Overlapping Match Search (OMS) problem for \(c < 1\) with failure probability \(f\) is to construct a data structure over the set \(R\) with similarity measure \(D\) supporting the following query: given any query ride \(q\), if \(D(q, r) \geq s\) for some \(r \in R\), then report some \(r' \in R\) such that \(D(q, r') \geq cs\), with probability \(1 - f\).

**Definition 8.** The \((c, s)\)-approximate \(k\) Overlapping Match Search (k-OMS) problem for \(c < 1\) with failure probability \(f\) is to construct a data structure over the set \(R\) with similarity measure \(D\) as above supporting the following query: given any query ride \(q\), if there exists \(k\) rides \(r_1, r_2, \ldots, r_k \in R\) such that \(D(q, r) \geq s\) \(\forall r \in \{r_1, r_2, \ldots, r_k\}\), then report some \(k\) rides \(r_1', r_2', \ldots, r_k' \in R\) such that \(D(q, r') \geq cs\) \(\forall r' \in \{r_1', r_2', \ldots, r_k'\}\), with probability \(1 - f\).

For simplicity of exposition, first let us assume the cost of each edge is unit. This assumption is somewhat justified as the edges are between adjacent discretized nodes of the route; however, we will soon remove this assumption. With this assumption, the similarity between two rides is simply the cardinality of the intersection of their set representations. Since set intersection can be computed as the inner product of the corresponding characteristic vectors of the sets, we represent each ride by the characteristic vector of its set representation. In other words, The ambient space is a high-dimensional space where each dimension is an ordered 2-tuple of discretized nodes. The vector representation of a ride is a vector with all 0’s except at the corresponding edge is in the set representation of the ride. The cardinality of the intersection between the set representations of two rides can be seen as the inner product between their vector representations. Approximate OMS can thus be solved by solving approximate MIPS.

Let us now turn our attention to the general case relaxing the assumption of unit cost per edge. We will define two vector representations for each ride: *preprocessing vector representation* used while creating the asymmetric LSH dataset and *query vector representation* used while querying for overlapping matches of a ride. Both the vector representations are defined in the same ambient space as defined above where each dimension corresponds to an ordered 2-tuple of discretized nodes.

- **preprocessing vector representation of ride** \(r\) \((p_r)\): A vector with all 0’s except for along dimensions for which the corresponding edge is in the set representation of the ride, in which case it is the cost of that edge.

- **query vector representation of ride** \(r\) \((q_r)\): A vector with all 0’s except for along dimensions for which the corresponding edge is in the set representation of the ride, in which case it is 1.

It is easy to see that with the above representations, the similarity between two rides \(r\) and \(q\) can be seen now as the inner product between the preprocessing vector representation of \(r\) and the query vector representation of \(q\). Thus, like before, approximate OMS can be solved by solving approximate MIPS.

**The algorithm**

We are now ready to present our algorithm for approximate potential match search. As noted in the previous subsection, our strategy is to solve it via solving approximate k-OMS. The algorithm is formally presented in Algorithm 1.

We now show that our algorithm is efficient and has high probability of success.

**Theorem 4.** Given a set \(R\) of \(n\) rides, Algorithm 1 solves \((c, s)\)-approximate k-OMS requiring \(O(n^\rho (k + \log n) \log k)\) query time and \(O(n^{1+\rho} \log k)\) space for some \(\rho < 1\) depending on \(c\) and \(s\). Therefore, the total running time for Algorithm 1 is \(O(n^{1+\rho} (k + \log n) \log k)\).

**Proof.** The proof follows the proof techniques used to solve near neighbor problem using LSH known in literature. Let \(\mathcal{H}'\) be the family of signed random projection functions of [Cha02]. As shown in [SL15], coupled with the preprocessing and the query transformations, they form a \((c, s, p_1, p_2)\)-sensitive asymmetric LSH family \(\mathcal{H}\) for MIPS where \(p_1, p_2\) depends on \(c\) and \(s\). Authors in [SL15] show that \(U = 0.75\) and \(m = 2\) are reasonable parameter choices which we fixed in our algorithm.

We use the technique of amplification by concatenating \(t = \log n/\log(1/p_2)\) randomly picked hash functions from \(\mathcal{H}\) to get the family of hash functions \(\mathcal{G}\). If the similarity between two rides \(q\) and \(r\), as defined in Definition 7, is at least \(s\), we call them “similar”. On the other hand, if the similarity between them is at most \(cs\), we call they are “dissimilar”. We have:
cretize space at the geohash 6 level. For constructing the LSH dataset, we did not use the signed random projections consisting of rides requested close in time so that they could potentially be matched.

In order to validate our algorithm and the overall approach of solving approximate PMS via solving approximate OMS, we implemented our algorithm and evaluated on a dataset of real ride data obtained from a ride-sharing platform consisting of rides requested close in time so that they could potentially be matched.

For implementing our algorithm, we used the routes for rides obtained from a routing service. We chose to discretize space at the geohash 6 level. For constructing the LSH dataset, we did not use the signed random projections

\section{Experimental setup and results}

In order to validate our algorithm and the overall approach of solving approximate PMS via solving approximate OMS, we implemented our algorithm and evaluated on a dataset of real ride data obtained from a ride-sharing platform consisting of rides requested close in time so that they could potentially be matched.

For implementing our algorithm, we used the routes for rides obtained from a routing service. We chose to discretize space at the geohash 6 level. For constructing the LSH dataset, we did not use the signed random projections
hash functions of [Cha02] for Maximum Cosine Similarity (MCS). Instead, we used the cross-polytope hash functions for MCS from the more recent work [AIL+15]. The reason for this choice is that the construction of [AIL+15] has better theoretical guarantees and have been shown to have superior practical performance. Moreover, [AIL+15] provided multi-probe LSH schemes which make our implementation space efficient and scalable to tens of thousands of rides. The LSH dataset was constructed with $L = 50$ hash tables and 800 probes.

We evaluated our algorithm on a set of randomly picked 525 rides, and on a set of randomly picked 2200 rides to test its scalability and performance.

For the set of 525 rides, the dimension of the ambient space was found to be about 3000. Constructing the LSH dataset took about 0.1 sec. Query time for all the 525 rides was about 0.95 secs without any parallelism with a single-threaded execution on a standard laptop. Note that the queries can be made highly parallelized.

The fraction of rides succeeding in the $(n = 525, k = 50, i_1, i_2)$-approximate PMS is shown below.

| $i_1$ = 1 | $i_2 = 2$ | $i_2 = 3$ | $i_2 = 4$ | $i_2 = 1$ |
|-----------|-----------|-----------|-----------|-----------|
| 0.85      |           |           |           |           |
| $i_1$ = 3 |           | 0.99      | 0.92      | 0.66      |
|           | 1.0       | 0.99      | 0.92      | 0.72      | 0.53      |

For the set of 2200 rides, the dimension of the ambient space was found to be about 5800. Constructing the LSH dataset took about 0.89 sec. Query time for all the 2200 rides was about 10.5 secs, again without any parallelization.

The fraction of rides succeeding in the $(n = 2200, k = 100, i_1, i_2)$-approximate PMS is shown below.

| $i_1$ = 1 | $i_2 = 2$ | $i_2 = 3$ | $i_2 = 4$ | $i_2 = 1$ |
|-----------|-----------|-----------|-----------|-----------|
| 0.67      |           |           |           |           |
| $i_1$ = 3 |           | 0.87      | 0.68      | 0.44      |
|           | 0.94      | 0.84      | 0.70      | 0.54      | 0.33      |

5 Conclusions

We studied the problem of efficiently finding $k$ potential matches for a ride from a pool of $n$ rides. We formalized the problem as approximate Potential Match Search (PMS) by defining a generic utility function capturing the notion of cost saving by matching two rides together.

In order to solve this problem, we constructed suitable ride representations in a high-dimensional ambient space for which defined an inner product based similarity measure that roughly captures the utility value of a matching. We then showed how approximate PMS can be solved by solving an alternate problem we defined - approximate Overlapping Match Search (OMS). For the latter, we devised a locality sensitive hashing based algorithm.

We implemented and evaluated our algorithm on real ride sets obtained from a ride-sharing platform. Our results showed that our algorithm is highly efficient compared to exhaustive search while being able to find high utility matches with high probability for most settings.

Several directions for future research work remain. We have used only one route for each ride to come up with its dataset and query representations. It is conceivable that a ride has few possible alternate routes and can be matched with another ride with high utility if it traverses one of these alternate routes. It is straightforward to incorporate these alternate routes in our algorithm. It will be interesting to see how much it can increase the success probability of our algorithm. One can also try alternate spacial discretization techniques. Finally, it will also be interesting to compare the practical performance of our algorithm using SimHash family of LSH functions against using cross-polytope family of LSH functions for MCS.
References

[AESW11] N. A. H. Agatz, A. L. Erera, M. W. P. Savelsbergh, and X. Wang. Dynamic ride-sharing: a simulation study in metro atlanta. *Transportation Research Part B: Methodological*, 2011.

[AI04] A. Andoni and P. Indyk. E2lsh: Exact euclidean locality-sensitive hashing, implementation available at http://web.mit.edu/andoni/www/lsh/index.html. 2004.

[AI06] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proc. of the Annual IEEE Symp. on Foundations of Computer Science*, 2006.

[AIL+15] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal lsh for angular distance. In *Proc. of the Intl. Conf. on Neural Information Processing Systems*, 2015.

[AINR14] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn. Beyond locality-sensitive hashing. In *Proc. of the Annual ACM-SIAM Symp. on Discrete Algorithms*, 2014.

[AMSW+17] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proc. of the National Academy of Sciences USA*, 2017.

[AR15] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proc. of the Annual ACM Symp. on Theory of Computing*, 2015.

[BFGB+14] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and Ulrich Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proc. of the ACM Conf. on Recommender systems*, 2014.

[BT01] J. Buhler and M. Tompa. Finding motifs using random projections. In *Proc. of the Annual Intl. Conf. on Computational Molecular Biology (RECOMB)*, 2001.

[Buh01] J. Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 2001.

[Buh02] J. Buhler. Provably sensitive indexing strategies for biosequence similarity search. In *Proc. of the Annual Intl. Conf. on Computational Molecular Biology (RECOMB)*, 2002.

[CDF+01] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, 2001.

[Cha02] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. of the Annual ACM Symp. on Theory of computing*, 2002.

[dACA16] G. H. de Almeida Correia and B. V. Arem. Solving the user optimum privately owned automated vehicles assignment problem (uo-poavap): a model to explore the impacts of self-driving vehicles on urban mobility. *Transportation Research Part B: Methodological*, 2016.

[DIIM04] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality sensitive hashing scheme based on p-stable distributions. In *Proc. of the ACM Symp. on Computational Geometry*, 2004.

[DRS+13] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *CVPR*, 2013.

[GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc of the Intl. Conf. on Very Large Data Bases*, 1999.

[GSM03] B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: A texture classification example. In *Proc. of the Intl. Conf. on Computer Vision*, 2003.
[HGI09] T. H. Haveliwala, A. Gionis, and P. Indyk. Scalable techniques for clustering the web. In Proc. of the WebDB Workshop, 2009.

[HPIM12] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. 2012.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In Proc. of the ACM Symp. on Theory of Computing, 1998.

[JDS11] H. Jgou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011.

[JFY09] T. Joachims, T. Finley, and C. J. Yu. Cutting-plane training of structural svms. Machine Learning, 2009.

[JK09] P. Jain and A. Kapoor. Active learning for large multi-class problems. In CVPR, 2009.

[KBV09] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. Computer, 2009.

[KRS12] N. Koenigstein, P. Ram, and Y. Shavitt. Efficient retrieval of recommendations in a matrix factorization framework. In Proc. of the ACM Intl. Conf. on Information and Knowledge Management (CIKM), 2012.

[LCYM17] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis. Fexipro: Fast and exact inner product retrieval in recommender systems. In Proc. of the ACM Intl. Conf. on Management of Data (SIGMOD), 2017.

[LJW+07] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In Proc. of the Intl. Conf. on Very Large Data Bases, 2007.

[MZW13] S. Ma, Y. Zheng, and O. Wolfson. T-share: A large scale dynamic taxi ridesharing service. In Proc. of the International Conference on Data Engineering, 2013.

[NS15] B. Neyshabur and N. Srebro. On symmetric and asymmetric lshs for inner product search. In Proc. of the Intl. Conf. on Machine Learning, 2015.

[OMS+02] Z. Ouyang, N. Memon, T. Suel, , and D. Trendafilov. Cluster-based delta compression of collections of files. In Proc. of the Intl. Conf. on Web Information Systems Engineering (WISE), 2002.

[PSFR12] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus. Robotic load balancing for mobility-on-demand systems. Intl. Journal of Robotics Research, 2012.

[SL14] A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In Proc. of the Intl. Conf. on Neural Information Processing Systems, 2014.

[SL15] A. Shrivastava and P. Li. Improved asymmetric locality sensitive hashing (alsh) for maximum inner product search (mips). In Proc. of the Conf. on Uncertainty in Artificial Intelligence, 2015.

[SJR05] N. Srebro, J. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In Proc. of the Intl. Conf. on Neural Information Processing Systems, 2005.

[SRS+14] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti. Quantifying the benefits of vehicle pooling with shareability networks. In Proc. of the National Academy of Sciences USA, 2014.

[SS17] R. Spring and A. Shrivastava. Scalable and sustainable deep learning via randomized hashing. In Proc. of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, 2017.

[SSGF16] K. Spieser, S. Samararayake, W. Gruel, and E. Frazzoli. Shared-vehicle mobility-on-demand systems: A fleet operators guide to rebalancing empty vehicles. In Transportation Research Board 95th Annual Meeting, 2016.
[SSM14] L. Schmidt, M. Sharifi, and I. L. Moreno. Large-scale speaker identification. In Proc. of the IEEE Intl. Conf. on Acoustic, Speech and Signal Processing (ICASSP), 2014.

[STS+13] N. Sundaram, A. Turmukhametova, N. Satish, T. Mostak, P. Indyk, S. Madden, and P. Dubey. Streaming similarity search over one billion tweets using parallel locality sensitive hashing. In Proc. of the VLDB Endowment, 2013.

[Yan01] C. Yang. Macs: Music audio characteristic sequence indexing for similarity retrieval. In Proc. of the Workshop on Applications of Signal Processing to Audio and Acoustics, 2001.

[ZP15] R. Zhang and M. Pavone. Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. Intl. Journal of Robotics Research, 2015.