Discovering Diverse Popular Paths using Transactional Modeling and Pattern Mining

by

P.Revanth Rathan, P Krishna Reddy, Anirban Mondal

in

International Conference on Database and Expert Systems Applications(DEXA)

Report No: IIIT/TR/2019/-1

Centre for Data Engineering
International Institute of Information Technology
Hyderabad - 500 032, INDIA
August 2019

Discovering Diverse Popular Paths using Transactional Modeling and Pattern Mining

P. Revanth Rathan[1], P. Krishna Reddy[1], and Anirban Mondal[2]

[1] IIIT Hyderabad, India,
revanth.parvathaneni@research.iiit.ac.in; pkreddy@iiit.ac.in
[2] Ashoka University,
anirban.mondal@ashoka.edu.in

Abstract. While the problems of finding the shortest path and k-shortest paths have been extensively researched, the research community has been shifting its focus towards discovering and identifying paths based on user preferences. Since users naturally follow some of the paths more than other paths, the popularity of a given path often reflects such user preferences. Moreover, users typically prefer diverse paths over similar paths for gaining flexibility in path selection. Given a set of user traversals in a road network and a set SP of paths between a given source and destination pair, we address the problem of performing top-k ranking of the paths in SP based on both path popularity and path diversity. Our main contributions are three-fold. First, we introduce the notion of *popular paths* and propose a model for computing the popularity scores of paths. Second, we propose a framework for modeling user traversals in a road network as transactions and computing the popularity score of any path based on the itemsets extracted from the transactions using pattern mining techniques. Third, we conducted a performance evaluation with a *real dataset* to demonstrate the effectiveness of the proposed scheme.

Keywords: Popular Paths · Transactional Modeling · Pattern Mining.

1 Introduction

The problems of finding the shortest path [6, 7, 13] and k-shortest paths [10, 12, 16] have been extensively researched. Interestingly, path finding and discovery has significant applications in several important and diverse domains such as city planning, transportation and vehicular navigation, disaster management, tourism and so on. However, the recent focus of the research community as well as that of most commercial route planning and navigation systems has been shifting towards discovering and identifying paths based on user preferences.

Such user preferences may include roads with relatively high thoroughfare (i.e., better for safety), smoother roads with better infrastructure (e.g., less potholes), roads with more facilities or points of interest nearby, lighted roads as opposed to dark and unsafe streets, roads with relatively lower crime rates, roads with better scenic beauty and so on. In practice, given that users naturally follow

some of these paths significantly more as compared to other paths, the **popularity of a given path** often reflects such user preferences. *In this work, we use the notion of popularity as the overarching theme for reflecting the path preferences of users.* Besides traversal time/cost of a route, users also consider the *popularity* of a route as an important factor. Furthermore, users often prefer to obtain **diverse paths** [4, 5] for gaining flexibility in path selection e.g., when their preferred routes become blocked (unusable) possibly due to a wide gamut of reasons such as flooding, traffic congestion and road maintenance works.

Now we explain the related terminology about road network, paths and user traversals. We model a given road network as a weighted graph $G(V, E)$, where V is the set of nodes of the graph. Here, each node represents an intersection of the road segments. E is the set of edges of the graph, where each edge represents a given road segment. Thus, $V = \{v_1,\ v_2,...,\ v_n\}$, where v_i represents the i^{th} node. Additionally, the edges are of the form $< v_i,\ v_j >$ such that v_i, $v_j \in V$. A user traversal is a spatial trajectory, which is represented by a sequence of edges in the graph G. We consider that a given *user traversal* does not involve any break exceeding a pre-defined threshold time limit t_w, which is essentially application-dependent. A given user traversal q_i is of the form $\{(v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n)\}$, where v_i $(i \neq 1$ and $i \neq n)$ is the i^{th} node, while v_1 and v_n denote the source node and the destination node respectively.

Given a set of user traversals in a road network and a set SP of paths between a given source and destination pair, we address the problem of performing ranking of the paths in SP based on both path popularity and path diversity. We refer to this problem as the **Diverse Popular Paths (DPP) problem**.

Research efforts are being made to address the problem of extracting popular routes in road networks [2,8,15]. Our approach differs from existing popular path extraction approaches as it defines the popularity of the path based on the effective number of user traversals, which cover the path. Furthermore, in contrast with existing works, we also propose an efficient methodology for extraction of top-k popular paths by deploying pattern mining techniques.

For a road network G and a set of user traversals, we consider a path p_i between a given source s and a destination t to be more popular than other paths between s and t, if more users follow p_i as compared to the other paths. We develop a model for computing the popularity score of a given path based on the number of user traversals by introducing the notion of the *popularity contribution of a user traversal to a given path.* We also develop the methodology for computing the popularity scores of all the input paths between s and t by processing the set of user traversals. Determining the popularity of the path p_i from the road network G using the proposed model poses scalability issues as the number of edges in p_i increases. By modeling the user traversals as a set of transactions, we compute the knowledge of all frequent patterns of all sizes. Here, each frequent pattern consists of the edges of the path. We propose an efficient algorithm for computing the popularity score of a given path based on the knowledge of frequent patterns. Finally, based on the popularity score and

a pre-defined diversity threshold, we output a ranked list of popular and diverse paths from the set of input paths. Our main contributions are as follows:

1. We introduce the notion of *popular paths* and propose a model for computing the popularity scores of paths.
2. We propose a framework for modeling user traversals in a road network as transactions and computing the popularity score of any path based on the itemsets extracted from the transactions using pattern mining techniques.
3. We conducted a performance evaluation with a *real dataset* to demonstrate the effectiveness of the proposed scheme.

The remainder of the paper is organized as follows. In Section 2, we discuss related work and background. In Section 3, we present the modeling of popularity score of a path. In Section 4, we discuss the proposed scheme. In Section 5, we report the performance evaluation. Finally, we conclude in Section 6.

2 Related Work and Background

This section discusses related works and background about path diversity.

The problem of finding the shortest path has been researched in [6, 7, 13]. The problem of finding top-k shortest paths have been addressed in [10, 12, 16]. In particular, Yen's algorithm [16] finds loop-less top-k shortest paths. Yen's algorithm exploits the idea that the k^{th} shortest path may share edges and sub-paths (path from source to any other intermediate nodes within the path) from the $(k-1)^{th}$ shortest path. Yen's algorithm first finds the shortest paths using Dijkstra algorithm [7]. It then takes every node in the current shortest path, except for the destination and computes the shortest paths from each selected node to the destination. The work in [5] proposes exact and approximate algorithms to discover the top-k diverse shortest paths based on the concept of *limited overlap* by exploring Yen's algorithm.

Research efforts are being made to find popular paths based on the spatial trajectories of users [2, 3, 15]. The work in [3] uses adaptive Markov model for computing the popularity of a given sequence of edges in order to compute the most popular route. This model is used to deduce the transfer probability for each edge in the path. Here, the transfer probability refers to the probability of traversing from that edge to the destination. The work in [2] improves the above approach for finding the top-k personalized routes from spatial trajectories. The proposal in [15] constructs a route inference model and a routable graph construction from uncertain trajectories for determining the top-k popular paths, which comprise only trajectory points. A popularity-related weight is assigned to each and every edge of the path based on the inference model for computing the popularity of the path.

The work in [8] reviews an extensive collection of existing trajectory data mining techniques and discusses them in a framework of trajectory data mining, which can be used for future data mining solutions in trajectory. The work in [11] introduced two novel concepts to trajectory indexing and introduced algorithms

for various types of spatio-temporal queries that involve routing in road networks such as finding all paths and vehicles in a road network, which are traversed after a given spatio-temporal context. The work in [14] user a route score function that strikes a balance between user preference degrees and the length of the route as a measure of the popularity.

Frequent pattern (FP) mining is one of the important concepts in data mining. The process of mining FPs extracts the interesting information about the associations among the items in patterns in a transactional database based on a user-specified *support (frequency)* threshold. Mining FPs has been extensively studied in the literature [1].

The proposed approach is different from the existing shortest path, personalized, and diverse route computation approaches as the proposed approach addresses the problem of extraction of top-k popular paths for a given source and destination. The proposed approach is different from existing popular path extraction approaches as it defines the popularity of the path based on the effective number of user traversals, which cover the path. Notably, the approach proposed in [15] extracts only one popular path between a given source and destination pair by extending the notion of *transfer probability* and the approach in [2] addresses the different problem of extracting popular routes from uncertain trajectories.

About Diversity among Paths: Now we shall explain the notion of *diversity among paths*. In this work, we model the notion of diversity based on the overlap among paths. In this section, we use path as set of edges. Each edge (v_i, v_j) in path p_i has a positive weight, which represents the cost of traversing from v_i to v_j, e.g., travel time or distance. The weight of a given path p_i, $l(p_i)$, is computed as the sum of the weights of all of the edges present in p_i. The notion of overlap defined in [4, 5] is as follows. Given two paths p_i and p_j in a network G, **overlap** OV of p_i w.r.t p_j is the ratio of weight of common edges in p_j to the weight of all the edges in p_j .

The notion of diversity used in this work is inversely proportional to the notion of overlap between two paths. The equation for **diversity value** between two paths p_i and p_j based on overlap is as follows:

$$DV(p_i,\ p_j) = 1 - \frac{\sum_{e \in p_i \cap p_j} l(e)}{\sum_{e \in p_j} l(e)} \tag{1}$$

Here, $l(e)$ represents the weight of the edge e and $0 \leq DV(p_i,\ p_j) \leq 1$. The maximum value of DV is achieved when intersection of p_i and p_j is the null set, while the minimum value of 0 is achieved when p_i equals p_j.

3 Model of Computing Popularity Score

In this section, we develop a new model for computing the popularity score of a given path. We defer the discussion on the proposed approach for the efficient computation of top-k diverse popular paths from a set of paths to Section 4.

Now we shall introduce the notion of *popularity score* of a given path. We shall henceforth designate the popularity score of a given path p_i with n edges as

$\omega(p_i)$. Intuitively, the value of $\omega(p_i)$ depends upon the number of user traversals that traverse p_i. In practice, all user traversals need not necessarily traverse all of the edges in the path. We can divide the whole traversals of all users into a set of traversals, which cover only 1 edge, only 2 edges and so on upto the set of traversals that cover all of the edges of p_i. These partially covered traversals also contribute to the popularity of the path. To incorporate the effect of such partially covered traversals, we shall now introduce the notion of the *popularity contribution* $PC(q_i, p_i)$ of a user traversal q_i w.r.t. a given path p_i. Intuitively, the popularity contribution PC indicates the extent to which a user traversal contributes to the popularity of a given path.

Consider a path p_i between a source s and a destination t with n edges, and Q_{p_i} user traversals, which cover *at least* one edge of p_i. The **popularity score** $\omega(p_i)$ of p_i equals the sum of the individual popularity contributions of Q_{p_i} user traversals. We define $\omega(p_i)$ as follows:

$$\omega(p_i) = \sum_{i=1}^{Q_{p_i}} PC(q_i, p_i) \tag{2}$$

Here, $0 \leq PC(q_i, p_i) \leq 1$. When all edges of p_i are covered by q_i, $PC(q_i, p_i) = 1$. Conversely, when none of the edges of p_i is traversed by q_i, $PC(q_i, p_i) = 0$.

Equation 2 captures PC of all kinds of user traversals. The issue is to compute the value of $\omega(p_i)$ by considering all kinds of user traversals. First, for simplicity, we present a method for estimating $\omega(p_i)$ by considering only those user traversals, which cover all of the edges of the path. (This hypothetical case may not hold good in practice.) Second, we discuss a method for estimating $\omega(p_i)$ by considering the user traversals, which allude to the traversal of a fixed number of edges in the path. Third, we present a method for the generalized case arising in real-world scenarios by considering all kinds of user traversals. Now let us discuss these three cases in detail.

Case 1 - User traversals which cover all edges of p_i: Given a path p_i and a user traversal q_i, if q_i traverses all edges of p_i, $PC(q_i, p_i) = 1$; otherwise, it is 0. Using Equation 2, for Q_{p_i} user traversals, which cover all edges of p_i, $\omega(p_i) = Q_{p_i}$. Given two paths p_1 and p_2 between source s and destination t with the number of user traversals, which cover all edges of p_1 and p_2, being Q_{p_1} and Q_{p_2} respectively, $\omega(p_1) > \omega(p_2)$ iff $Q_{p_1} > Q_{p_2}$. This matches our intuitive understanding of the notion of popularity score, which depends on the number of user traversals of the path.

Case 2 - User traversals which cover exactly k edges of p_i: In practice, any given user may use only one or more edges of the path and then take a detour into other edges/paths. Case 1 does not capture this real-world scenario. In Case 1, if q_i covers all n edges of p_i, $PC(q_i, p_i)$ is 1. Now, if q_i covers only k edges of p_i, $PC(q_i, p_i) = W(k)$, where W is a function with k as the parameter. W can be any monotonically increasing function w.r.t. k. Thus, $W(k) \propto k$ *and* $0 \leq W(k) \leq 1$. If q_i does not cover any edge of p_i, $PC(q_i, p_i) = W(0) = 0$. (The k edges in p_i of a user traversal need not be continuous edges.)

Different kinds of functions can be selected for W. Intuitively, given two traversals, a traversal, which covers more edges of p_i contributes more to $\omega(p_i)$. Hence, W should be selected such that given user traversals q_1 and q_2, which cover k_1 and k_2 edges of p_i such that $k_1 < k_2$, $W(k_1)$ should be less than $W(k_2)$. Using Equation 2, given Q_{p_i} user traversals, which cover only k edges in p_i, $\omega(p_i)$ equals $(\omega(p_i) = \sum_{i=1}^{Q_{p_i}} PC(q_i, p_i) = Q_{p_i} * W(k))$. Given two paths p_1 and p_2 between source s and destination t with user traversals, which cover k edges of p_1 and p_2, being Q_{p_1} and Q_{p_2}, $\omega(p_1)>\omega(p_2)$ iff $\Big(Q_{p_1} * W(k)\Big) > \Big(Q_{p_2} * W(k)\Big)$.

Case 3 - All user traversals which cover p_i: We can divide Q_{p_i} user traversals into user traversals, which cover only 1 edge, only 2 edges and up to all n edges of the path. Let C_k represent the set of all combinations of edges in p_i of size k, and $Q_{p_i}(C_k)$ represents the number of user traversals, which traverse *only* C_k. Thus, the value of Q_{p_i} is $\sum_{k=1}^{n} Q_{p_i}(C_k)$. Each of these user traversals contribute to the popularity score of the path. Thus, using Equation 2, we can compute $\omega(p_i)$ for the generalized real-world case as follows:

$$\omega(p_i) = \sum_{i=1}^{Q_{p_i}} PC(q_i, p_i) = \sum_{k=1}^{n} Q_{p_i}(C_k) * W(k) \tag{3}$$

Intuitively, given two paths, we consider that a path, which is covered by more user traversals with a larger combination of edges, is more popular than the path, which contains a lower number of user traversals. Observe how this intuition of popularity is reflected in Equation 3.

4 Diverse Popular Paths Query and Proposed Scheme

The *diverse popular path (DPP)* query is as follows. Given a set of user traversals in a road network G, source s, destination t and the user-specified diversity threshold value θ and a set SP of paths between s and t, the DPP query determines the ranked list L of diverse popular paths in SP (in descending order of popularity score) such that diversity threshold criterion is satisfied. Here, L is of the form $<p_1, p_2, p_3, ..., p_k>$, where $\forall i, j \in [1, k]$, if $i<j$, then $\omega(p_i)>\omega(p_j)$. (Recall the computation of popularity score in Equation 3.) Furthermore, recall the computation of the diversity (DV) between any two given paths in Equation 1 of Section 2. We deem the **diversity threshold criterion** to be satisfied for any two paths p_i and p_j if $DV(p_i, p_j) \geq \theta$.

4.1 Basic Idea

To process a given DPP query, we need to compute the popularity scores of paths in set SP of paths (using Equation 3). However, this requires the frequencies of potential combinations of edges of user traversals, which traversed at least one edge of p_i. Hence, given a path p_i comprising n edges, we need to examine the frequencies of $2^n - 1$ combinations of these edges. Notably, for every

path p_i, when a DPP query comes in, it would be prohibitively expensive to generate all combinations of edges and compute their frequencies in an *online* manner because each path may have a different set of edges. However, there is an opportunity here for *offline* pre-processing and extraction of the knowledge of frequency of edge combinations; we designate such knowledge as *patterns*. Thus, when a DPP query comes in, we can use the extracted patterns towards *efficiently* processing the DPP query *online*.

Our DPP query processing scheme is as follows. First, we convert the user traversals into a transactional dataset D. Next, we extract the knowledge of patterns from D by using the existing FP-Growth [9] algorithm. Then we compute the popularity score of each path of SP using Equation 3, extract the set PP of popular paths and order the paths based on popularity score. Finally, we extract top-k DPP from PP such that the diversity threshold criterion is satisfied.

4.2 Proposed Scheme

The proposed scheme has three phases: (1) modeling of user traversal transactions ($UTTs$) (2) extraction of combinations of edges and their frequencies from $UTTs$ (3) extraction of diverse popular paths. Now we shall discuss each phase.

1. Modeling of user traversal transactions ($UTTs$): Here, the input is a set of user traversals (UTs). We convert each UT into one or more $UTTs$ between two nodes in a road network. We assume that a UT contains the details of user arrival time and departure time at each node. For the starting and ending nodes, we only have departure time and arrival time respectively. We discuss two options to form $UTTs$ from UTs. First, we specify a threshold time window t_w. Each UT is traversed from the starting node. All parts of a given UT occurring within t_w are considered to be one UTT. When a UT exceeds t_w, the parts of the UT exceeding t_w are considered to be a new UTT. Second, whenever a user backtracks or visits one of the already visited nodes, the edges visited so far are considered as one UTT; the traversal process is then continued from the preceding node. For both cases, the same process is repeated till the last node to identify further $UTTs$.

2. Extraction of combinations of edges and their frequencies from $UTTs$: By considering $UTTs$ as transactions and edges as items, all combinations of edges with support greater than or equal to the user-specified value of *minsup* is extracted by the existing FP-growth algorithm [9]. We can create the database of edge combinations ($DBEC$) with the corresponding count of transactions. Here, the entries in $DBEC$ are of the form <edge combination, frequency>. Algorithm 1 depicts the creation of $DBEC$.

3. Extraction of diverse popular paths: Algorithm 2 depicts the computation of top-k diverse popular paths (DPP). We compute the popularity score $\omega(p_i)$ of a given path p_i in SP using Equation 3. We compute $\omega(p_i)$ by extracting (from $DBEC$) the knowledge of the respective frequencies of UTTs of all edge combinations, which are formed with the edges of p_i. We use the principle of inclusion and exclusion to obtain the frequencies of $UTTs$, which traverse *only* certain edge combinations from $DBEC$. In Algorithm 2, the computation

Algorithm 1 Compute_DBEC(G, UT)

Input: G: Graph; UT: User Traversals; $minSUP$: minimum support
Output: $DBEC$: Database of edge combinations

1: Form $UTTs$ from UT
2: Extract edge combinations (and corresponding frequencies) which satisfy minimum support $minSUP$ from $UTTs$ using FP-growth and store in $DBEC$

Algorithm 2 $DPP(SP, DBEC, \theta, W, k)$

Input: SP: set of paths; θ: diversity threshold; $DBEC$: list of $<e_i, f_i>$. Here, f_i represents the frequency of i^{th} edge combination e_i; W: weight function; k: required number of paths;
Output: DPP: set of diverse popular paths;
Variables: PP: list of $<p_i, \omega(p_i)>$; F: list of $<e_i, f_i>$;

1: Initialize PP to null
2: **for** each $p_i \in SP$ **do** {
3: Generate all edge combinations of p_i
4: Initialize F to null
5: **for** each e_i of p_i **do** {
6: Extract $<e_i, f_i>$ from DBEC and store in F //We use hash map. }
7: Initialize ps to zero
8: **for** each $<e_i, f_i> \in F$ **do** {
9: $count$=number of edges in e_i
10: ps = ps + (W(count) * f_i) // W is the given weight function }
11: Insert $<p_i, ps>$ to PP }
12: Sort PP w.r.t. ps in the descending order
13: Intialize DPP to null
14: Remove p_i having the highest ps value from PP and insert in DPP
15: **while** PP $\neq$ null **do** {
16: Remove p_i having the highest ps value from PP and insert in DPP
17: for all $p_k \in DPP$ { if ($DV(p_i, p_k) \geq \theta$) **then** add p_i to DPP }
18: **if** (size of (DPP) == k) **then break** }

of $\omega(p_i)$ is shown in Lines 3-10. Here, computing all combinations of edges in a given path p_i would be prohibitively expensive. Hence, we use a hashmap on $DBEC$ to obtain the edge combinations, which are a subset of p_i, and their corresponding frequencies (see Line 6). After computing the popularity scores for all paths in SP, we sort the paths in descending order based on their respective popularity scores into a list PP (see Line 12).

Now, from PP, we need to extract the top-k paths that satisfy the minimum diversity threshold criterion. Hence, we need to identify the list DPP, which contains the paths that have a minimum diversity with other paths in the list. A path p_i added to list DPP if the diversity between p_i and every path p_j in DPP is greater than the minimum diversity threshold θ. Observe how in Line 14, the path with the highest popularity score in the set PP is added to DPP. Based on the order of popularity, we add a path to DPP if it satisfies the condition of

minimum diversity, as explained above (see Line 17). In this manner, we continue to iterate until DPP contains the required number of k paths or until we have exhausted iterating over all of the paths in PP.

4.3 Illustrative Example

We explain the working of our approach through an example. Consider 1000 users traverse the road network given in Figure 1, which consists of four paths, namely p_1 ($<e_{12}, e_{23}, e_{34}>$), p_2 ($<e_{15}, e_{56}, e_{64}>$), p_3 ($<e_{17}, e_{78}, e_{84}>$) and p_4 ($<e_{12}, e_{26}, e_{64}>$) from the source node 1 to the destination node 4. Table 1 contains the details of edge combinations with frequencies ($DBEC$). Given source node 1 and destination node 4, for computing the top-*3* DPPs, we need to compute the popularity score (ω) of four paths using Equation 3.

We consider the weight function $W(i) as \frac{i*(i+1)}{n*(n+1)}$, where i is the number of edges in the combination and n represents the number of edges in the path. Popularity scores (ω) of the paths in Figure 1 are computed as follows:

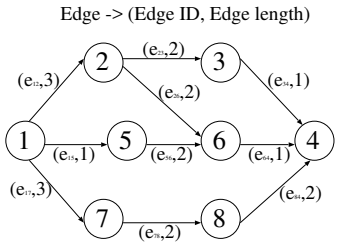


Fig. 1: Sample Graph

Table 1: DBEC of user traversals

Edge combination	Freq-uency	Edge combination	Freq-uency
$\{e_{12}\}$	40	$\{e_{12}, e_{26}\}$	25
$\{e_{23}\}$	20	$\{e_{23}, e_{34}\}$	40
$\{e_{34}\}$	30	$\{e_{15}, e_{56}\}$	15
$\{e_{15}\}$	6	$\{e_{15}, e_{64}\}$	10
$\{e_{56}\}$	5	$\{e_{56}, e_{64}\}$	20
$\{e_{64}\}$	4	$\{e_{17}, e_{78}\}$	10
$\{e_{17}\}$	100	$\{e_{17}, e_{84}\}$	15
$\{e_{78}\}$	50	$\{e_{78}, e_{84}\}$	15
$\{e_{84}\}$	60	$\{e_{26}, e_{64}\}$	20
$\{e_{26}\}$	80	$\{e_{12}, e_{23}, e_{34}\}$	40
$\{e_{12}, e_{23}\}$	30	$\{e_{12}, e_{26}, e_{64}\}$	30
$\{e_{12}, e_{34}\}$	20	$\{e_{15}, e_{56}, e_{64}\}$	15
$\{e_{12}, e_{64}\}$	15	$\{e_{17}, e_{78}, e_{84}\}$	0

$$\omega(p_1) = \sum_{k=1}^{3} W(k) * Q(C_k) = W(1) * Q(C_1) + W(2) * Q(C_2) + W(3) * Q(C_3)$$
$$= \frac{2}{12} * (40 + 20 + 30) + \frac{6}{12} * (30 + 20 + 40) + \frac{12}{12} * 40 = 100$$

Similarly, the values of $\omega(p_2)$, $\omega(p_3)$ and $\omega(p_4)$ are 40, 55 and 80.67 respectively. Paths are arranged in the descending order $p_1 > p_4 > p_3 > p_2$ based on the popularity score. Here, notice that p_1 is more popular than p_3, even though the number of users traversing through p_3 is more than that of p_1. This is because more users are traversing through the combinations of edges, which results

in the increase in $\omega(p_1)$ w.r.t. $\omega(p_3)$. Even though p_2 has more users traversing through the combinations of edges than p_3, the *effective number* of users traversing through p_2 are significantly low when compared to that of p_3. As a result, p_3 is receiving more popularity score w.r.t. p_2.

After extracting PP, we extract DPP by comparing other paths with the top popular path by applying the minimum diversity threshold criterion. We set the diversity threshold θ as 0.6. The popular path to p_1 is p_4, but p_4 cannot be added to DPP because $DV(p_1, p_4) = 0.5$, which is less than θ. All the remaining paths are added to the set DPP in the order of their popularity scores, subject to the satisfaction of diversity threshold as per Algorithm 2. Hence, the final DPP is $<p_1, p_3, p_2>$.

4.4 Time and Space Complexity

DBEC is extracted from UTTs in an offline manner using Algorithm 1, which employs an existing pattern extraction algorithm such as FP-growth [9]. Given SP, source and destination, DPPs are extracted using Algorithm 2 online. Now, we present the complexity of Algorithm 2. Let TC denote the number of edge combinations (in DBEC) for all paths in SP. The average number C of edge combinations per path is $TC/|SP|$. Time complexity for different steps is as follows: (i) computing the popularity score for all paths in SP is $O(|SP| * C)$ (ii) sorting SP is $O(|SP| * log(|SP|))$ and (iii) computing DPP is $O(|SP|^2)$. Hence, the total time complexity is $O(|SP| * C)+O(|SP| * log(|SP|))+O(|SP|^2)$. In general, $C \gg |SP|$. So, **time** complexity of Algorithm 2 comes to $O(|SP| * C)$. Also, the **space** required for Algorithm 2 is equal to the space for storing all the paths in SP and $DBEC$.

5 Performance Evaluation

We conducted the performance evaluation on an Intel i7 processor with 8GB RAM running Ubuntu Linux. We used OpenStreetMap[3] for Beijing and done the implementation using Osmnx[4] framework. We used Microsoft's *Geolife* real dataset [17]. This GPS trajectory dataset was collected in the Geolife project by 182 users during a period of over 5 years. The dataset contains 17,621 trajectories with a total distance of 1,292,951 kilometres and a total duration of 50,176 hours. These trajectories were recorded by different GPS loggers/phones with varied sampling rates. Each trajectory is a sequence of time-stamped points, each of which contains information about latitude, longitude and altitude. In this work, we considered 14,175 traversals in Beijing. Figure 2 depicts the heat map of the user traversals. First, we mapped each trajectory of the user to the road network using a map-matching tool Graphhopper[5]. Then we used nominatim[6] to obtain the corresponding sequence of edge IDs; we consider this sequence as a path.

[3] https://www.openstreetmap.org
[4] https://osmnx.readthedocs.io/en/stable/
[5] https://graphhopper.com/api/1/docs/map-matching/
[6] https://nominatim.openstreetmap.org/

We used the backtracking method (see Section 4.2) for dividing user traversals into multiple user traversal transactions ($UTTs$). We deployed the FP-Growth algorithm [9] to obtain all frequent patterns from $UTTs$ with support greater than 300. To select the candidate set of paths between s and t, we introduce the notion of distance threshold δ. Let sl be the shortest path length between s and t. We extract all paths, whose length is less than $(sl + \delta * sl)$.

Table 2 summarizes the parameters of our performance study. In Table 2, we set the default value of the distance threshold δ to 0.2. Furthermore, we set the default value of the diversity threshold θ (discussed in Section 4) to 0.6. In our experiments, we studied the effect of variations in δ as well as θ.

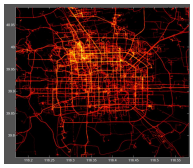


Fig. 2: Heat map of user traversals in Beijing's fifth Ring Road

Table 2: Performance study parameters

Parameter	Default	Variations
The distance threshold (δ)	0.2	0.1, 0.3, 0.4, 0.5
The diversity threshold (θ)	0.6	0.2, 0.4, 0.8, 1

We conducted our experiments with three representative queries, where each query is a source and destination pair. To generate these queries, we randomly selected 10 queries each from areas of low, medium and high spatial densities such that the distance between each of the source and destination pairs is less than 1 km. Then, for each type of region (i.e., regions having low, medium and high spatial densities), we randomly selected one query from among these 10 queries. We shall henceforth refer to the queries from regions of high, medium and low spatial densities as $Q1$, $Q2$ and $Q3$ respectively.

Our performance metrics include: average length (AL) of a set of paths, average popularity score (APS) of a set of paths, number of popular paths (NPP), number of diverse popular paths (NDP) and execution time (ET).

As reference, we have used the implementation of our model for computing the popularity scores of paths (see Section 3) without using frequent pattern information. Given a transactional database with user traversal transactions, we traverse all the transactions online to obtain the frequencies of each and every edge combination of each path. In this approach, we have constructed DBEC for a specific path through online traversal. We shall henceforth refer to this scheme as *online approach* and our proposed scheme as *offline approach*.

Given source and destination, for Q1, Q2, and Q3, we extract PP and DPP with the proposed approach and compare with SP and DSP. We also compare the performance of the proposed approach with the offline approach.

Average length (AL) and Average Popularity Score (APS) of extracted set of paths: For $Q1$, $Q2$, and $Q3$, Figure 3a shows the results of AL of top-k shortest paths (SP), popular paths (PP), diverse shortest paths (DSP) and diverse popular paths (DPP). For three queries, it can be observed that AL of SP is less than AL of PP. Also, AL of DSP is less than AL of DPP. For $Q1$,

$Q2$, and $Q3$, Figure 3b shows the results of APS of top-k SP, PP, DSP and DPP. For three queries, it can be observed that APS of SP is less than APS of PP. Also, APS of DSP is less than APS of DPP.

The following observations can be drawn from Figure 3a and Figure 3b. First, as expected, the popular path is longer than the shortest path and diverse popular path is longer than diverse shortest path. Notably, the additional distance is not significant. So, the results show that it is possible to obtain popular paths (which have the benefits of security, good roads and so on) for a given source and destination by traveling small additional distance. Second, the results also show that, independent of spatial density, it is possible to obtain popular paths for $Q1$, $Q2$, and $Q3$ as there are reasonable number of user traversals in the order of thousand for each query. The popularity score is independent of spatial density w.r.t. roads. This is because the proposed notion of popularity considers user traversals.

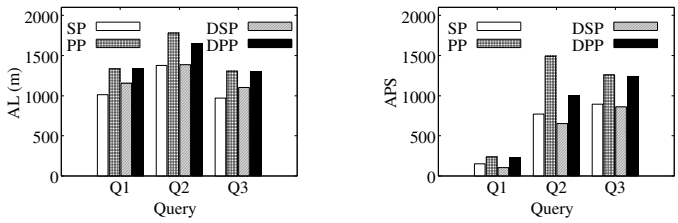
Effect of varying the distance threshold (δ): Figure 4 depicts the results of varying the value of δ. Here, δ denotes additional percentage distance w.r.t. the shortest path distance. The experiment was conducted to determine the number of popular paths as we vary δ. The results in Figure 4a indicate that as the value of δ increases, the number of paths increases; hence, the number NPP of popular paths also increases. Observe that for $Q1$ (which is a query in a dense region), 34 popular paths could be obtained even by adding a small δ (say 5%). Similarly, for $Q2$ and $Q3$, it is possible to obtain 22 and 6 popular paths respectively with δ of 5%. The results are very encouraging because they indicate that users need not travel any significant additional distance to obtain popular paths. As popular paths have other potential benefits, we believe that the proposed approach would be instrumental in facilitating the design of popular path discovery services as a complement to services for finding only the shortest paths.

Figure 4b depicts the comparison of the execution of $Q1$, $Q2$, and $Q3$ of the proposed approach w.r.t. the online approach. In the graph, the performance of the proposed offline approach is indicated by $Q1$, $Q2$ and $Q3$, whereas the performance of the online approach is indicated by $*Q1$, $*Q2$, and $*Q3$. It can be observed that the proposed approach improves the performance significantly over the online approach as we extract frequencies of each combination of edges in an offline manner using pattern mining techniques and employed hashmap to *efficiently* search the patterns.

On the other hand, in the online approach, the database of UTTs need to be scanned to extract frequency of combinations of each path in an online manner, which is computationally prohibitively intensive. As a result, the proposed approach exhibits significant performance improvement and returning the popular paths within a few seconds. The implication is that our proposed approach can be used as a foundation for building near real-time path discovery services. Notably, the time complexity of our proposed online approach is in the order of the transaction size and the number of edges in the path.

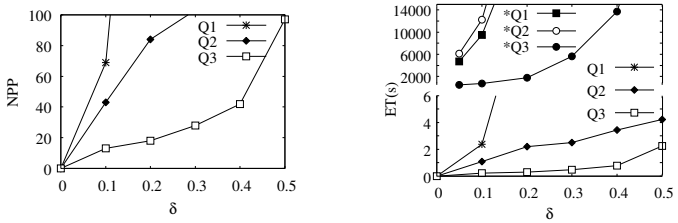
It can also be observed that as δ increases, the execution time to compute NPP also vary among $Q1$, $Q2$ and $Q3$. It can be observed that the execution time

ET to extract NPP for $Q1$ increases exponentially with δ due to a significant increase in the number of paths. For $Q2$ and $Q3$, ET increases gradually as compared to $Q1$ due to a lower number of NPP.



(a) Average Length (b) Average Popularity Score

Fig. 3: Results on queries in regions of varying Spatial Densities

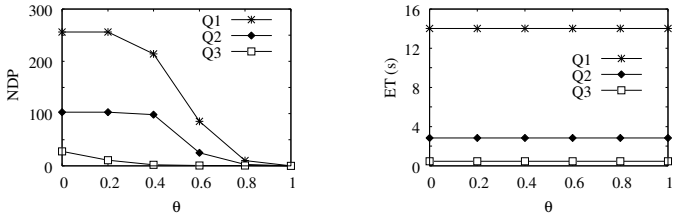


(a) NPP (b) ET

Fig. 4: Effect of variations in the distance threshold

Effect of varying the diversity threshold (θ): Figure 5 depicts the results of varying the value of the diversity threshold θ. $\theta = 0$ implies that all the paths, irrespective of their diversity constraint, can be a part of the DPP result set. $\theta = 1$ implies only those paths, which are completely different (i.e., diverse) can contribute to the DPP result set.

The results in Figure 5a indicate that as the value of θ decreases, more number of paths are included in the diverse popular set and it reaches to a saturation point (i.e., the point at which all of the paths in the candidate set are in the diverse set). For some of the paths, saturation point reaches earlier because all of the extracted paths have minimum diversity with each other. More number of paths are generated in dense region, because of this the set DPP also increases. Figure 5b shows that the execution time for computing the top-k DPP for $Q1$ is high as compared to $Q2$ and $Q3$. The execution time decreases slowly as we vary θ due to less number of paths.



(a) NDP (b) ET

Fig. 5: Effect of variations in the diversity threshold

Table 3: Illustrative comparison of shortest path rank and popularity rank

T1O-SP: Rank of 10 paths as per the distance (shortest path receives the first rank); **PR**: Popularity Rank; **T1O-PP**: Rank of 10 paths as per the popularity score (the path with highest popularity receives the first rank); **SPR**: Shortest Path's Rank											
Q1				Q2				Q3			
T10 - SP	PR	T10 - PP	SPR	T10 - SP	PR	T10 - PP	SPR	T10 - SP	PR	T10 - PP	SPR
1	2	1	7	1	6	1	9	1	8	1	5
2	3	2	3	2	8	2	10	2	9	2	2
3	4	3	5	3	1	3	4	3	5	3	8
4	5	4	1	4	4	4	2	4	1	4	10
5	6	5	6	5	9	5	1	5	7	5	7
6	7	6	4	6	10	6	8	6	2	6	1
7	1	7	9	7	5	7	5	7	4	7	4
8	8	8	10	8	2	8	6	8	10	8	6
9	9	9	8	9	7	9	3	9	3	9	3
10	10	10	2	10	3	10	7	10	6	10	9

Comparison of shortest path rank and popularity rank: Table 3 depicts the illustrative comparison of shortest path rank and popularity rank and vice versa of all three queries for top-10 results. We present the list of top-10 shortest paths and their corresponding popularity rank and the list of top-10 popular paths and their ordering based on path length for all three queries. For each query, we can observe that the path having the highest shortest path rank (SPR) i.e., shortest path between source and destination may not be the path with highest popularity. Similarly, the paths with higher popularity score may not have the highest value of SPR. Hence, we can say that top most shortest path might not be most popular path and vice versa. So, the ordering of paths based on popularity score indicates the different kind of knowledge than the ordering based on distance.

6 Conclusion

In this paper, we have addressed the problem of ranking the input paths between a given source and destination pair based on popularity. We have proposed a

model to compute the popularity score of a path by combining the effect of the number of user traversals and the number of edges of the path covered by each user traversal. We have also presented an efficient approach for computing the popularity score of a given path by modeling user traversals as transactions and deploying pattern mining techniques. Our performance study with a real dataset shows the effectiveness of the proposed scheme. We believe that the proposed approach would be instrumental in designing popular path discovery services as a complement to services for finding only the shortest paths.

References

1. Aggarwal, C.C., Han, J. (eds.): Frequent Pattern Mining. Springer (2014)
2. Chang, K.P., Wei, L.Y., Yeh, M.Y., Peng, W.C.: Discovering personalized routes from trajectories. In: Proc. ACM SIGSPATIAL. pp. 33–40 (2011)
3. Chen, Z., Shen, H.T., Zhou, X.: Discovering popular routes from trajectories. In: Proc. ICDE. pp. 900–911 (2011)
4. Chondrogiannis, T., Bouros, P., Gamper, J., Leser, U.: Alternative routing: k-shortest paths with limited overlap. In: Proc. ACM SIGSPATIAL. p. 68 (2015)
5. Chondrogiannis, T., Bouros, P., Gamper, J., Leser, U.: Exact and approximate algorithms for finding k-shortest paths with limited overlap. In: Proc. EDBT 2017. pp. 414–425 (2017)
6. Chondrogiannis, T., Gamper, J.: ParDiSP: A partition-based framework for distance and shortest path queries on road networks. In: Proc. MDM. vol. 1, pp. 242–251 (2016)
7. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**(1), 269–271 (1959)
8. Feng, Z., Zhu, Y.: A survey on trajectory data mining: Techniques and applications. IEEE Access **4**, 2056–2067 (2016)
9. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD record. vol. 29, pp. 1–12 (2000)
10. Hershberger, J., Maxel, M., Suri, S.: Finding the k shortest simple paths: A new algorithm and its implementation. ACM Transactions on Algorithms **3**(4), 45 (2007)
11. Koide, S., Tadokoro, Y., Yoshimura, T., Xiao, C., Ishikawa, Y.: Enhanced indexing and querying of trajectories in road networks via string algorithms. ACM Transactions on Spatial Algorithms and Systems **4**(1), 3 (2018)
12. Martins, E.Q., Pascoal, M.M.: A new implementation of Yen's ranking loopless paths algorithm. Quarterly Journal of the Belgian, French and Italian Operations Research Societies **1**(2), 121–133 (2003)
13. Sommer, C.: Shortest-path queries in static networks. ACM Computing Surveys (CSUR) **46**(4), 45 (2014)
14. Wei, L.Y., Chang, K.P., Peng, W.C.: Discovering pattern-aware routes from trajectories. Distributed and Parallel Databases **33**(2), 201–226 (2015)
15. Wei, L.Y., Zheng, Y., Peng, W.C.: Constructing popular routes from uncertain trajectories. In: Proc. ACM SIGKDD. pp. 195–203 (2012)
16. Yen, J.Y.: Finding the k shortest loopless paths in a network. Management Science **17**(11), 712–716 (1971)
17. Zheng, Y., Xie, X., Ma, W.Y.: Geolife: A collaborative social networking service among user, location and trajectory. IEEE Data Eng. Bull. **33**(2), 32–39 (2010)