Mining the Most Influential k-Location Set From Massive Trajectories

Yuhong Li[1] Jie Bao[2] Yanhua Li[3] Yingcai Wu[4] Zhiguo Gong[1] Yu Zheng[2,5,6]

[1]*University of Macau, Macau, China, {yb27407,fstzgg}@umac.mo*
[2]*Microsoft Research, Beijing, China, {jiebao,yuzheng}@microsoft.com*
[3]*Worcester Polytechnic Institute, MA, USA, yli15@wpi.edu*
[4]*State Key Lab of CAD&CG, Zhejiang University, Zhejiang, China, ycwu@zju.edu.cn*
[5]*Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China*
[6]*School of Computer Science and Technology, Xidian University, China*

ABSTRACT

Mining the most influential k-location set finds k locations, traversed by the maximum number of unique trajectories, in a given spatial region. These influential locations are valuable for resource allocation applications, such as selecting charging stations for electric automobiles and suggesting locations for placing billboards. This problem is NP-hard and usually calls for an interactive mining processes, e.g., changing the spatial region and k, or removing some locations (from the results in the previous round) that are not eligible for an application according to the domain knowledge. Thus, efficiency is the major concern in addressing this problem. In this paper, we propose a system by using greedy heuristics to expedite the mining process. The greedy heuristic is efficient with performance guarantee. We evaluate the performance of our proposed system based on a taxi dataset of Tianjin, and provide a case study on selecting the locations for charging stations in Beijing.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining, Spatial database and GIS*

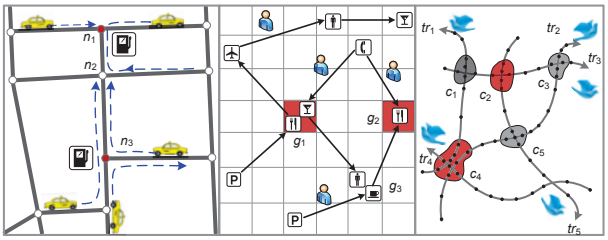
Keywords

Location Selection, Trajectory Data Mining, Maximum Coverage

1. INTRODUCTION

Advances in location acquisition technology have resulted in massive trajectories, representing the mobility of a diversity of moving objects, e.g., human, vehicles, and animals. Finding k locations traversed by the maximum number of unique trajectories in a given spatial region is vital to many resource allocation problems:

The first application is selecting charging stations for electric vehicles according to their GPS trajectories. As shown in Fig. 1(a), the location is defined as road intersection. Intersections n_1 and n_3 form the most influential 2-location set by covering 5 unique trajectories. n_2 and n_3 are not the most influential set, as they only cover 4 unique trajectories. Though they individually cover the most number of trajectories (i.e., 3 for each).



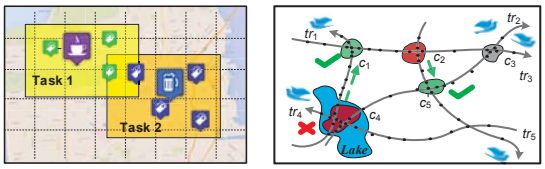
(a) EV Charging Station (b) Billboard (c) Observation Station

Figure 1: Application Scenarios.

The second application is to select locations for placing billboards based on users' check-in histories. As shown in Fig. 1(b), a location can be defined as a uniform grid covering a few points of interests (POIs). g_1 and g_2 form a most influential 2-location set, traversed by 4 unique trajectories, i.e., visited by 4 users.

The third application is to place observation stations for migratory birds, where a location can be a cluster of birds' stay points. As shown in Fig. 1(c), c_2 and c_4 form the most influential 2-location set that covers all birds' trajectories.

There are three major challenges in mining the most influential location set from massive trajectories: i) this problem can be mapped to the MAX-k-COVER problem, which is NP-hard and computational intensive; ii) different users may be interested in mining k locations in different spatial areas. For instance, as shown in Fig. 2(a), two local business owners may want to place different numbers of billboards in different areas. As a result, it is not possible to pre-compute one location set to serve all requests with different mining parameters; and iii) users, e.g., domain experts, may need to interact with our system several times. For example, as depicted in Fig. 2(b), c_4 is located in a lake where we cannot find land to place an observation station, we need to remove it from the returned set. Then, c_1 and c_5 become the most influential 2-location set. Although the MAX-k-COVER problem has been studied [1, 2, 3], existing methods are off-line approaches that find a one-time



(a) Location-awareness (b) Domain Knowledge & Interaction

Figure 2: Summary of Challenges.

SIGSPATIAL'16, October 31-November 03, 2016, Burlingame, CA, USA

DOI: http://dx.doi.org/10.1145/2996913.2997009

result for a given dataset. Different from existing works, our problem setting allows users: i) to specify a spatial region R and value k, and ii) to refine the returned result interactively and iteratively.

To this end, we propose a system to find the most influential k-location set efficiently in this paper. Our system contains two main modules: i) *pre-processing module*, which creates the spatial networks from different types of trajectory data and builds a set of index structures to speed up the mining process; and ii) *location set mining module*, which finds k locations by taking spatial region R, value k, and choices made during the user's interaction as the input. Our main contributions are summarized as follows:

- We introduce a novel problem, i.e., mining the most influential k-location set, with many potential applications.
- We propose an efficient algorithm to find the k-location set. Even the solution is based on greedy heuristics, it can provide the performance guarantee.
- Evaluation results on real datasets demonstrate the efficiency of our proposed solution. We also provide a case study to show the effectiveness of our proposed system.

2. OVERVIEW

Preliminary. We first introduce some definitions used in our paper.

DEFINITION 1 (TRAJECTORY). *A trajectory tr is a sequence of spatial points that a moving object follows through space as a function of time. Each point consists of an object ID, latitude, longitude, and a time stamp.*

DEFINITION 2 (LOCATION). *A location is a spatial point or region, which can be defined in three forms: 1) an intersection in a road network, e.g., n_1 shown in Fig. 1(a); 2) a grid cell, e.g., g_1 as depicted in Fig. 1(b); or 3) a stay point or a cluster of points from trajectories, e.g. c_2 as illustrated in Fig. 1(c).*

DEFINITION 3 (SPATIAL NETWORK). *A spatial network can be denoted as a directed graph $G = (V, E)$, where the vertex set V represents the locations and the directed edge set E represents the set of edges where each has two terminal locations.*

DEFINITION 4 (COVERAGE). *A location v_i covers a trajectory tr_j, if and only if the trajectory tr_j passes the location v_i. Thus, given a location on a spatial network (e.g., an intersection v_i on a road network), its coverage set $Tr(v_i)$ represents the set of trajectories passing through the location v_i.*

Problem Definition. *Given a user-specified spatial region R, a k value and a set of trajectories Tr, we denote the spatial network in R as $G_s = (V_s, E_s)$. The most influential k-location set in R finds k[1] locations in V_s, such that the total number of unique trajectories being covered by the k locations is maximized.*

To be precise, we use the following integer linear programming (ILP) formulation to captures the problem exactly. We denote $v_i.s$ and $tr_j.s$ indicate the solution of the problem, namely, for each location $v_i \in V_s$, $v_i.s = 1$ if v_i is selected in the result set, and $v_i.s = 0$ otherwise; for each trajectory $tr_j \in Tr$, $tr_j.s = 1$ if tr_j is covered by the selected locations, and $tr_j.s = 0$, otherwise.

$$\max : \sum_{tr_j \in Tr} tr_j.s, s.t. : \sum_{v_i \in V_s} v_i.s \leq k, \sum_{tr_j \in Tr(v_i)} v_i.s \geq tr_j.s \quad (1)$$

[1]Here $k \leq |V_s|$.

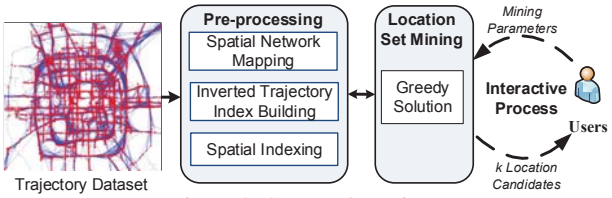


Figure 3: System Overview.

The objective of Eq. 1 is to maximize the total number of trajectories being covered by the selected locations. The first constraint guarantees that the total number of selected locations is no more than k; the second constraint ensures that if a location v_i is selected, all trajectories that traverse v_i are covered. This problem is equal to the MAX-k-COVER problem and is NP-hard as in [1, 4].

Moreover, to take the domain knowledge into the mining task, the system needs to support the interactive process with multiple iterations to find the qualified k locations. Specificity, at the initial step, the system returns k locations that maximize the number of covered trajectories based on the parameters. Then, the expert involves and marks $0 \leq \ell \leq k$ disqualified locations from these k (selected) locations, based on his domain knowledge. In the following steps, the system needs to remove these ℓ marked locations and re-selects k locations, covering the maximum number of unique trajectories. This process iterates, until the expert accepts all the returned k locations [2]. This kind of mining process motivates our proposed system to achieve two objectives: 1) maximizing the coverage; and 2) minimizing response time.

System Overview. Fig. 3 gives the overview of our proposed system. It contains two main modules:
Pre-processing Module: This takes the trajectory dataset as the input and performs three major procedures: i) *Spatial Network Mapping*; ii) *Inverted Trajectory Indexing*; and iii) *Spatial Indexing*. Their details are presented in Sect. 3.
Location Set Mining Module: This takes a user's query parameters, i.e., a spatial range R, a value k and a set of marked locations (e.g., by the expert) as the input, and returns k locations as the result. The process goes multiple iterations until the user satisfies the final result. In this paper, we propose utilizing the greedy heuristics to choose the candidate locations efficiently (detailed in Sect. 4).

3. PRE-PROCESSING

Spatial Network Mapping. This step contains two tasks: 1) spatial network construction, the system first identifies the locations based on different scenarios, e.g., the intersections, spatial cells, or the stay points, and then constructs the spatial network; 2) trajectory mapping, the system then maps the raw trajectories onto the corresponding spatial network, e.g., using a map matching algorithm [5]. The output of the procedure is a *trajectory-vertex index*, where each trajectory is represented by a set of vertexes (locations).
Inverted Trajectory Index Building. This step builds the *vertex-trajectory index*, which stores trajectory IDs for each vertex.
Spatial Index Building. The spatial index is used to speed up the spatial selection. In this step, we take the vertexes V as the input and use R$^+$-tree [6] to index these spatial locations.

4. LOCATION SET MINING

The optimal solution to find the most influential k-location set is computing infeasible for large k due to its NP-hardness. An efficient approximate solution becomes more promising. In the lit-

[2]Pre-estimating the quality of all locations for different applications maybe infeasible.

Algorithm 1 Framework of Greedy Heuristics

Input: Vertex-trajectory index $\mathcal{I}_{vt}$, spatial index $\mathcal{I}_{spatial}$, spatial range R, and k value.
Output: k vertices V_{gdy}
1: $V_s := SpatialRangeSearch(\mathcal{I}_{spatial}, R)$
2: **for** i = 1 to k **do**
3: $V_{gdy} \leftarrow V_{gdy} \bigcup v_{cur}, v_{cur} \in V_s \setminus V_{gdy}$ with max coverage.
4: Update the coverage values in the *vertex coverage table*.
5: return V_{gdy}

Algorithm 2 Updating Algorithm

Input: vertex-trajectory index $\mathcal{I}_{vt}$, trajectory-vertex index $\mathcal{I}_{tv}$, candidate vertices V_s, selected vertex v_{cur}, vertex coverage table vct.
Output: Updated *vertex coverage table*
1: $Tr_{new} \leftarrow$ newly covered trajectories from v_{cur}
2: **for** each $tr \in Tr_{new}$ **do**
3: **for** each $v \in \mathcal{I}_{tv}[tr]$ **do**
4: **if** $v \in V_s \setminus V_{gdy}$ **then**
5: Update coverage value of v in vct.

erature, greedy heuristics have been proven [7] as the best polynomial time solution with $1 - \frac{1}{e}$ approximation guarantee to the optimal solution. In this section, we first present the framework of the greedy heuristic algorithm. Then, we introduce an *updating algorithm* to reduce the response time by taking the advantage of *trajectory-vertex index*.

4.1 Framework of Greedy Heuristic

In the greedy heuristic algorithm, we maintain a *vertex coverage table*, where each entry is identified by the vertex id v_i, and is associated with a coverage value. The coverage value is the number of unselected trajectories of this vertex. The framework of the greedy heuristic algorithm is very simple, as shown in Alg. 1. It first selects a set of candidate vertices in the spatial region R (i.e., Line 1). Then a k-iterative process is executed with two phases:

• *Selection Phase.* In this phase, the algorithm selects the vertex with the maximum trajectory coverage at the current iteration and put it in the result set (i.e., Line 3).

• *Updating Phase.* In this phase, the algorithm updates the coverage values for all the unselected vertices by removing the newly covered trajectories from their coverage (i.e., Line 4).

In Alg. 1, the spatial range query and selection operations can be processed very efficiently. However, the updating step is hideous and time consuming, especially when the trajectory dataset is huge, e.g., millions of entries, as the updating process needs to remove all the newly covered trajectories in the *vertex coverage table*. Thus, we focus on improving the efficiency of the updating phase.

4.2 Updating Algorithm

After each selection phase, a set of new trajectories are covered by the newly selected vertex. Thus, the coverage values of the remaining vertices in the *vertex coverage table* need to be updated, i.e., removing trajectories covered by the newly selected vertex. The updating algorithm scans the newly covered trajectories and find the vertices to be updated using the *trajectory-vertex index* $\mathcal{I}_{tv}$.

Alg. 2 shows the *updating algorithm* of the greedy heuristic algorithm. After the current vertex with maximum coverage v_{cur} is selected, the algorithm gets the *newly* covered trajectories Tr_{new} by adding the v_{cur} to V_{gdy} (i.e., Line 1). Specifically, Tr_{new} is calculated as $Tr_{new} = \mathcal{I}_{vt}[v_{cur}] \setminus Tr_{gdy}$, where $\mathcal{I}_{vt}[v_{cur}]$ and Tr_{gdy} are the covered trajectories of v_{cur} and V_{gdy} respectivially. Finally, the algorithm goes through the *trajectory-vertex index* for each newly added trajectory to update the values (i.e., minus one)

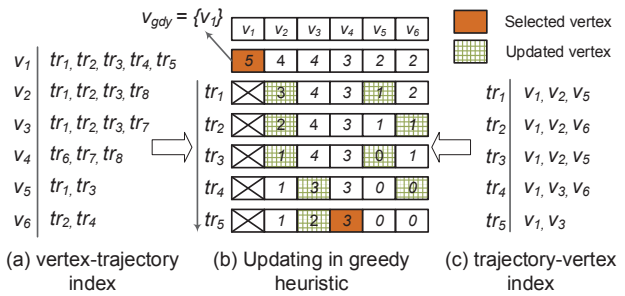


Figure 4: Running Example of Updating Algorithm.

in vertex coverage table vct (Line 2-5). This process stops after getting k vertices.

Example. Fig. 4 gives a running example of the *updating algorithm* to extract the most influential 2-location set using greedy heuristics. In this example, there are 6 vertices within the query spatial region, i.e., $\{v_1, v_2, v_3, v_4, v_5, v_6\}$. The corresponding *vertex-trajectory index* is shown in Fig. 4(a), and the *trajectory-vertex index* is shown in Fig. 4(c). The updating details of the *vertex coverage table* are demonstrated in Fig. 4(b), where each row indicates the updated coverage values after removing each trajectory in Tr_{new}. Initially, the coverage value of each vertex is their original covered trajectories, i.e., $\{5, 4, 4, 3, 2, 2\}$ for $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ respectively.

At the first iteration, v_1 is selected and added to the result set V_{gdy}, and all trajectories covered by v_i, i.e., $Tr_{new} = \{tr_1, tr_2, tr_3, tr_4, tr_5\}$, should be removed from the trajectory sets of other locations. The algorithm then utilizes the *trajectory-vertex index* of each trajectory in Tr_{new} to update the coverage values of the remaining vertices, i.e., v_2 to v_6. As shown in the first row of Fig. 4(b), the algorithm notices that the trajectory tr_1 passes the vertices v_1, v_2 and v_5 from the *trajectory-vertex index*. Thus, the coverage values of v_2 and v_5 should be updated, i.e., decreasing by 1, as tr_1 has been covered. The updating process continues until it checks all the newly added trajectories, i.e., tr_1 to tr_5. After the updating phase, the coverage values of the remaining vertices are $\{1, 2, 3, 0, 0\}$.

Based on the updated vertex coverage table, the greedy heuristic algorithm continues to select the second vertex. In this case, it will select v_3, as v_3 covers the most trajectories, i.e., 3. Finally, the location mining module stops as it has enough candidates.

Performance Analysis. The cost of the vertex selection process is relatively small, i.e., a linear scan of the coverage values of the remaining vertices, with the time complexity of $O(|V_s|)$. The dominant cost of the algorithm lays in the updating phase as it needs to scan the *trajectory-vertex index* one by one. The time complexity of updating phase by using Alg. 2 is $O(Tr_{gdy} \times \gamma)$, where Tr_{gdy} is the total number of trajectories covered by the final selected results, i.e., V_{gdy}, and γ is the average length of these trajectories.

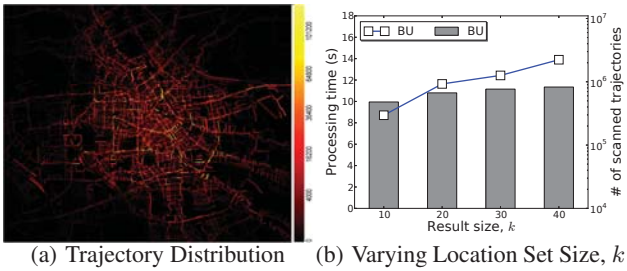
5. EXPERIMENTS

In this section, we first show the efficiency of our proposed system (Sect. 5.1). After that, we provide a case studies to demonstrate the utilization of our system (Sect. 5.2).

5.1 Efficiency Study

The performance are evaluated on a machine running Ubuntu 12.04 with Intel Core 6-Cores (12-Threads) i7-3930K 3.2GHz and 16GBytes of main memory. The processing time of our proposed system are reported by averaging 100 randomly selected queries.

Dataset. We use the GPS trajectories of 3,501 taxicabs from Tian-



(a) Trajectory Distribution (b) Varying Location Set Size, k

Figure 5: Efficiency study.

jin in 61 days. We perform a map-matching algorithm to map the trajectories on the road network of Tianjin, which contains 99,007 vertices and 133,726 segments. The road network covers an area of 123×187 km^2 with a total length of 32,487 km. Fig. 5(a) shows the spatial distribution of trajectories using a heat map[3]. Most of the trajectories are crowded within the downtown area.

Result. Fig 5(b) shows the processing time (by lines) and the total number of scanned trajectories (by bars) for our proposed system (denoted as *BU*), with different k values. The processing time and number of scanned trajectories increase linearly with k. Our proposed system can find the k-location set in 13.9 seconds even when $k = 40$. The efficiency of the proposed system make it possible to attract users to pursue interactions in the mining process.

5.2 Case Study

The government wants to deploy three charging stations for electric vehicles in the Wangjing Area (a district in Beijing) to promote green-energy. As the charging station is a public service, we need to cover as many users' travels as possible. Moreover, the placement of EV charging stations also need to consider the following three domain constraints: 1) the selected locations should have space for parking; 2) the nearby area needs a diverse array of POI categories; and 3) the selected locations should not be very close to each other.

Dataset. We use the GPS trajectories of 33,619 taxicabs in Beijing as a sample of vehicles' movements. We perform a map-matching algorithm to map the trajectories on the road network of Beijing, which contains 186,266 vertices and 249,080 segments. The target area is demonstrated as the shaded polygons in Fig 6(a) and 6(c) .

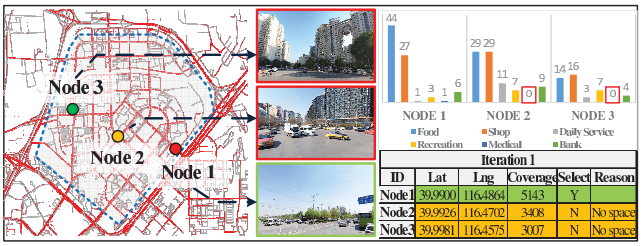
Results. Fig. 6 demonstrates the results using our technique with multiple iterations from the field experts.

• Fig. 6(a) gives the selection results in the first iteration, where three intersections are selected on the map (marked as red, orange and green). The three selected locations cover a total of 11,558 trajectories in the area. However, when we closely examine each location, we find that: 1) Node 2 and Node 3 do not have enough places for parking, as demonstrated in the street map view; and 2) the nearby POI distributions of Node 2 and Node 3 do not satisfy the diversity requirement (i.e., without any medical services), illustrated in the POI distribution view [4]. As a result, we only keep Node 1 in the result and perform a new selection iteration.

• Fig. 6(b) gives the selection results for the next two iterations. In the second iteration, we find a new set of three locations, where we keep Node 1 and Node 2 in the result and remove Node 3, as it does not have enough space for parking. On the third iteration, we find a new Node 3. However, it still does not satisfy our requirement, as it is very close to Node 1. Thus, we need to remove Node 3 and perform our algorithm continuously.

[3]A full version can be found at http://goo.gl/QUiIpZ.

[4]The POI distribution is calculated by aggregating the POIs within a 1 km range of the target location.



Iteration 1

ID	Lat	Lng	Coverage	Select	Reason
Node1	39.9900	116.4864	5143	Y	
Node2	39.9926	116.4702	3408	N	No space
Node3	39.9981	116.4575	3007	N	No space

(a) Charging Station Suggestions in the 1st Iteration

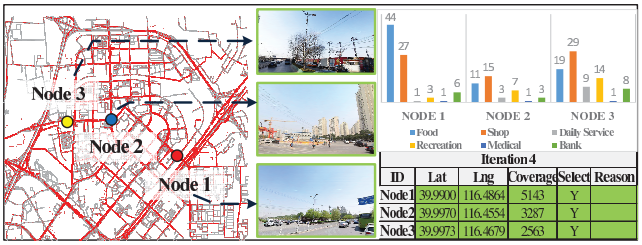
Iteration 2

ID	Lat	Lng	Coverage	Select	Reason
Node1	39.9900	116.4864	5143	Y	
Node2	39.9970	116.4554	3287	Y	
Node3	39.9927	116.4704	2768	N	No space

Iteration 3

ID	Lat	Lng	Coverage	Select	Reason
Node1	39.9900	116.4864	5143	Y	
Node2	39.9970	116.4554	3287	Y	
Node3	39.9903	116.4862	2689	N	Close to 1

(b) Interactive Process in the 2nd and 3rd Iterations



Iteration 4

ID	Lat	Lng	Coverage	Select	Reason
Node1	39.9900	116.4864	5143	Y	
Node2	39.9970	116.4554	3287	Y	
Node3	39.9973	116.4679	2563	Y	

(c) Charging Station Suggestions in the 4th Iteration

Figure 6: Placing Charging Stations in Wangjing Area, Beijing.

• Fig. 6(c) gives the final result in the Wangjing Area. All selected locations fulfill our requirement, where each of them has enough space for parking (demonstrated in the street view) and satisfies the POI diversity (shown in the bar chart). Most importantly, the 3-location set covers a total of 10,993 trajectories, which is very close to the total number of the first iteration, i.e., 11,558.

6. CONCLUSION

This work presents a system on mining most influential k-location set over massive trajectory data. It has many potential applications in resource allocation applications. We propose utilizing a greedy heuristic algorithm to support interactive queries. Extensive experiments on real datasets demonstrate that our proposed solution is efficient. We also demonstrate how to best place EV charging stations in Beijing using our proposed system.

7. ACKNOWLEDGMENTS

This work was supported by the NSF of China (grant No. 61672399 and No. U1401258), the National 973 Program of China (grant No. 2015CB352400), MYRG105-FST13-GZG, MYRG2015-00070-FST from UMAC Research Committee, FDCT/106/2012/A3, FDCT/116/2013/A3 from Macau FDCT, the National 973 Program of China (grant No. 2015CB352503), the NSF of China (grant No. 61502416), a grant from Microsoft Research Asia, a gift funding from Pitney Bowes, Inc.

8. REFERENCES

[1] F. Chierichetti, R. Kumar, and A. Tomkins, "Max-cover in map-reduce," in *WWW*. ACM, 2010, pp. 231–240.

[2] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor, "The online set cover problem," in *STOC*, 2003, pp. 100–105.

[3] B. Saha and L. Getoor, "On maximum coverage in the streaming model & application to multi-topic blog-watch," in *SDM*, 2009, pp. 697–708.

[4] D. S. Hochbaum, "Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems," in *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996, pp. 94–143.

[5] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate gps trajectories," in *SIGSPATIAL*. ACM, 2009, pp. 352–361.

[6] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The r+-tree: A dynamic index for multi-dimensional objects," 1987.

[7] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.