Identifying Top-K Optimal Locations for Placement of Large-Scale Trajectory-Aware Services

Shubhadip Mitra
Supervised by Arnab Bhattacharya
Dept. of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India
smitr@cse.iitk.ac.in

ABSTRACT

Optimal location problems identify the best sites to set up new facilities for providing service to its users. Majority of the existing work in this space assumes that the users are static and the datasets are small. Such assumptions are too restrictive and unrealistic for real-life services such as setting up of fuel stations, upgradation of cell-phone base-stations, etc. The placement of such services should, however, factor in the mobility patterns of its consumers, i.e., the user trajectories. For example, given a budget of k locations to set up fuel stations, the objective should be to cover the maximum number of user trajectories. In this doctoral work, we introduce top-k optimal location problems for large-scale *trajectory-aware* services. Since these problems are NP-hard, we design scalable techniques with bounded quality guarantees that work directly with user trajectories over city-scale road networks. Empirical evaluations show that the proposed heuristics are highly *efficient*, both in terms of space overhead and running time, as well as quite *effective*, with quality close to that of the optimal.

1. INTRODUCTION AND MOTIVATION

One of the most important problems in planning of services is to identify the best locations to set up new facilities (or improve existing facilities) with respect to a given service [4, 20]. Examples include setting up new services such as fuel stations, banking, retail stores, etc. Such *optimal location (OL)* problems have also been extensively studied as *Facility Location* problems [6, 9].

Majority of the existing works on OL problems assume that the users of the service are *static*. Such an assumption is often too prohibitive. For example, services such as mobile services, fuel stations, bill boards, traffic monitoring systems, etc. are widely accessed by users while commuting. Consequently, the placement of such facilities require taking into consideration the mobility patterns (or *trajectories*) of the users rather than their static locations [1–3, 19]. We refer to such services as *trajectory-aware* services. A *trajectory* is simply a sequence of location-time coordinates that lie on the path of a moving user. Such trajectory data are commonly available from GPS traces [10], CDR (Call Detail Records) data [11] recorded through cellphones, social network check-ins [5], etc. Due to more availability of real trajectory data, trajectory data analytics has attracted considerable research attention in recent years [12, 13, 18].

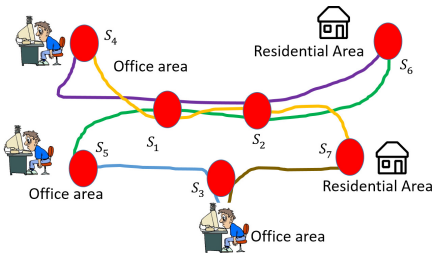

Figure 1: Illustration of the need of trajectory-aware querying for optimal locations. The lines represent the trajectories of users. We assume each trajectory belongs to a separate user.

To illustrate the need for trajectory-aware planning of services, consider Fig. 1. There are seven candidate locations, $S_1, \ldots, S_7$, to set up a service, out of which five are either home or office locations. There are five users commuting between home and office locations whose trajectories are shown. Using this figure, we next motivate three scenarios of trajectory-aware services.

Scenario 1: A company wants to open two new fuel stations. A trajectory of a user is *satisfied* only if it passes through at least one of the fuel stations. If only the static locations are considered, i.e., any two out of the five office and residential areas are to be selected, no combination would satisfy all the trajectories. In contrast, if we factor in the mobility of the users, choosing S_1 and S_3 as the installation locations satisfies all trajectories. Note that it is not enough to simply look at trajectory counts in each possible installation location and then choosing the two most frequent ones. By that strategy, a location such as S_2 would be chosen along with S_1. This combination is not effective since the same trajectories pass through both, thereby reducing each other's utilities (a process sometimes referred to as *cannibalization* in economics).

Scenario 2: Suppose a mobile service provider wants to set up two base-stations to provide good quality of experience to the commuting users. In this case, a user is *satisfied* if it receives good service for a large fraction of its trajectory. The selection $\{S_1, S_2\}$ satisfying 3 trajectories is the optimal choice. An alternative selection such as $\{S_1, S_3\}$ can offer good quality of service only in a limited segment of the incident trajectories, thereby not satisfying even a single trajectory.

Scenario 3: Suppose a company plans to set up two fuel stations that minimize the maximum inconvenience, i.e. the extra distance travelled by a user to avail a service. The optimal solution is $\{S_1, S_3\}$

Proceedings of the VLDB 2016 PhD Workshop, September 9, 2016. New Delhi, India.

(or $\{S_2, S_3\}$) with maximum inconvenience being 0. Any other solution would have non-zero maximum inconvenience since at least one user needs to travel non-zero extra distance to reach to the nearest fuel station.

The objective of this doctoral thesis is to develop an efficient framework for planning of large-scale trajectory-aware services such as the scenarios described above. More specifically, given n candidate service locations, the proposed models enable the service provider to identify the best k locations to set up new service or improve existing service by factoring in the trajectories of its users. The models allow the service providers to specify a wide range of objectives and constraints depending on the choice of service. To model various objectives, we associate a utility function U_j with each trajectory T_j in the problem. This utility function captures how well a trajectory is served by a given set of service locations. The goal is to determine k out of n candidate locations that *maximize* the sum of trajectory utilities U_j over all the trajectories.

Motivated by the above three scenarios, we study three classes of problems namely, TOPS, TUMP and TIPS, which are described in Sec. 2, 3 and 4, respectively.

The contributions and significance of this doctoral thesis are summarized as follows:

⋆ **Novelty:** To the best of our knowledge, this is the first extensive work to study efficient computation of placement of *large-scale trajectory-aware services*. The proposed models are highly generic and can be easily adapted to meet various service requirements.

⋆ **Hardness:** We show that the TOPS, TUMP and TIPS problems are NP-hard. We develop optimal algorithms for each of them and show why they are impractical.

⋆ **Efficiency:** We develop several interesting heuristics for each of the three problems. The proposed solutions are highly efficient both in terms of space overhead and running time.

⋆ **Effectiveness:** While the proposed heuristics have theoretically bounded quality guarantees, the empirical evaluations show that they are quite close to the optimal.

⋆ **Extensive Benchmarking:** We benchmark the proposed solutions through extensive experimental evaluation over real and synthetic large-scale datasets. The impact of various parameters such as budget, coverage threshold, trajectory-length, city geometries, etc. are thoroughly evaluated across multiple large-scale datasets.

We next describe in detail the TOPS, TUMP and TIPS problems.

2. THE TOPS PROBLEM

Referring to Scenario 1, here we consider the problem of OL queries for services such as fuel-stations, ATMs, convenience stores, bill boards, etc., that are typically demanded only intermittently, and not continuously throughout the trip.

Consider a road network $G = \{V, E\}$ over a geographical area where $V = \{v_1, \ldots, v_N\}$ denotes the set of road intersections (usually referred to as vertices or nodes), and E denotes the road segments between two adjacent road intersections. To model the direction of the underlying traffic that passes over a road segment, we assume that the edges are directed. Assume a set of candidate sites $\mathcal{S} = \{s_1, \cdots, s_n\} \subseteq V$ where a certain service or facility can be set up. The set $\mathcal{S}$ can be in addition to existing service locations. Without loss of generality, we can augment the vertices V to include all the sites. Thus, $\mathcal{S} \subseteq V$.

The set of trajectories over G is denoted by $\mathcal{T} = \{T_1, \cdots, T_m\}$ where each trajectory T_j is a sequence of nodes $T_j = \{v_{j_1}, \cdots, v_{j_l}\}$, $v_{j_i} \in V$. The trajectories are usually recorded as GPS traces and may contain arbitrary spatial points on the road network. For our purpose, each trajectory is map-matched [14] to form a sequence of road intersections through which it passes. We assume that each trajectory belongs to a separate user. However, the framework can easily generalize to multiple trajectories belonging to a single user.

Suppose $d(v_i, v_j)$ denotes the shortest network distance along a directed path from node v_i to v_j, and $d_r(v_i, v_j)$ denotes the shortest distance of a round-trip starting at node v_i, visiting v_j, and returning to v_i, i.e., $d_r(v_i, v_j) = d(v_i, v_j) + d(v_j, v_i)$. In general, $d(v_i, v_j) \neq d(v_j, v_i)$, but $d_r(v_i, v_j) = d_r(v_j, v_i)$. With a slight abuse of notation, assume that $d_r(T_j, s_i)$ denotes the *extra* distance traveled by the user on trajectory T_j to avail a service at site s_i. Formally, $d_r(T_j, s_i) = \min_{\forall v_k, v_l \in T_j}\{d(v_k, s_i) + d(s_i, v_l) - d(v_k, v_l)\}$.

It is convenient for a user to avail a service only if its location is not too far off from her trajectory. Thus, beyond a distance τ, we assume that the utility offered by a site s_i to a trajectory T_j is 0. We call this user-specified distance τ as the *coverage threshold*.

DEFINITION 1 (COVERAGE). *A site s_i covers a trajectory T_j if the distance $d_r(T_j, s_i)$ is at most τ, where $\tau \geq 0$ is the coverage threshold.*

For all sites within the coverage threshold τ, the user also specifies a preference function ψ. The preference function $\psi(T_j, s_i)$ assigns a score (normalized to $[0, 1]$) for a trajectory T_j and a site s_i that indicates how much s_i is preferred by the user on trajectory T_j. Higher values indicate higher preferences with 0 indicating no preference. In general, sites that are *closer* to the trajectory have *higher* preferences than those farther away.

DEFINITION 2 (PREFERENCE FUNCTION ψ). *$\psi : (\mathcal{T}, \mathcal{S}) \rightarrow [0, 1]$ is a real-valued preference function defined as follows:*

$$\psi(T_j, s_i) = \begin{cases} f(d_r(T_j, s_i)) & \text{if } d_r(T_j, s_i) \leq \tau \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where f is a non-increasing function of $d_r(T_j, s_i)$.

The goal of the TOPS query is to report a set of k sites $\mathcal{Q} \subseteq \mathcal{S}$, $|\mathcal{Q}| = k$, that maximizes the preference score over the set of trajectories. The preference score of a trajectory T_j over a set of sites $\mathcal{Q}$ is defined as the *utility function* U_j for T_j, which is simply the *maximum* score corresponding to the sites in $\mathcal{Q}$, i.e., $U_j = \max_{s_i \in \mathcal{Q}}\{\psi(T_j, s_i)\}$.

The generic TOPS query formulation is stated next.

PROBLEM 1 (TOPS). *Given a set of trajectories $\mathcal{T}$, a set of candidate sites $\mathcal{S}$ that can host the services, the TOPS problem with query parameters (k, τ, ψ) seeks to report the best k sites, $\mathcal{Q} \subseteq \mathcal{S}$, $|\mathcal{Q}| = k$, that maximize the sum of trajectory utilities, i.e., $\mathcal{Q} = \arg\max \sum_{j=1}^{m} U_j$ where $U_j = \max_{s_i \in Q}\{\psi(T_j, s_i)\}$.*

We show that the TOPS problem is NP-hard [16]. Further, we also prove that the sum of utilities $U = \sum_{j=1}^{m} U_j$ is a *non-decreasing sub-modular* function [16].

2.1 Algorithms for TOPS

We propose a greedy approximation algorithm INC-GREEDY to solve TOPS. This is based on the principle of *maximizing marginal gain*. In successive k iterations, it picks a site that offers maximal gain in the utility U. The approximation bound of this algorithm is $\max\{1-1/e, k/n\}$. To improve the performance of this algorithm, we use FM Sketches [7] for efficiently identifying the site offering maximal marginal gain in the utility U. The major drawback of this scheme is its high memory overhead of $O(mn)$ which is infeasible for large datasets. To address this limitation, we develop a scalable framework NETCLUS, discussed next.

We first state one basic observation. If two sites are close, the sets of trajectories they cover are likely to have a high overlap. Hence,

when $k \ll n$, which is typically the case, the sites chosen in the answer set are likely to be distant from each other. The index structure, NETCLUS, is designed based on the above observation.

Clustering of sites based on similarities between set memberships (such as Jaccard similarity of trajectory sets covered by two sites) will not be useful since they depend on the coverage threshold τ which is available only at run time. Hence, we adopt distance-based clustering. If two sites are close to each other, their utilities as well as the sets of trajectories they cover are likely to be similar. Hence, it is unlikely for two sites belonging to the same cluster to be part of the answer set. Therefore, if INC-GREEDY is performed only on the clusters instead of the sites, the answer sets returned and their corresponding utilities are likely to be similar.

Our method follows two main phases: offline and online. In the offline phase, clusters are built at multiple resolutions. This forms the different index instances. A particular index instance is useful for a particular range of query coverage thresholds. In the online phase, when the query parameters are known, first the appropriate index instance is chosen. The INC-GREEDY algorithm is then run with the cluster representatives on that instance. We omit the details in the interest of space.

Next we outline the solution at a high level. Given the raw GPS traces of user movements, they are map-matched [14] to the corresponding road network. From the map-matched trajectories, in conjunction with the road network, the NETCLUS index structure is built. NETCLUS performs a multi-resolution clustering of the road network following which indexed views of both the network and the trajectories are constructed in a compressed format at various granularities. This completes the offline phase. In the online phase, given the query parameters, the optimum clustering resolution to answer the query is identified, and the corresponding views of the trajectories and road network are analyzed to retrieve the best k sites for facility locations.

2.2 Summary of Experimental Results

We evaluate our heuristics on a publicly available and widely used real dataset consisting of GPS traces of taxis from Beijing [21, 22]. This dataset has $123,179$ trajectories, and $269,686$ sites.

To study the impact of city geographies, we also evaluate our solutions on three synthetic datasets that emulate trajectories in the patterns followed in New York, Atlanta and Bangalore. We use an online traffic generator tool, MNTG (http://mntg.cs.umn.edu/tg/index.php) to generate the traffic traces.

We observe that NETCLUS is up to 2 orders of magnitude faster than INC-GREEDY while yielding solutions that are within 90% of that of INC-GREEDY. The use of FM sketches for efficiently computing the site offering maximal marginal utility, yields speed up of 3-6 times. Importantly, while INC-GREEDY goes out of memory for $\tau \geq 1.6$ km on a 32 GB machine, NETCLUS requires less than 3 GB memory for different values of the query parameters. We also find that the longer trajectories are easier to serve than the shorter trajectories, as they can be served through a larger pool of candidate sites. In addition, NETCLUS can efficiently absorb dynamic updates such as change in trajectories or candidate locations.

3. THE TUMP PROBLEM

Referring to Scenario 2, here we focus on cellular services that aim to provide good quality of experience (QoE) not just at few discrete points, but throughout the trip.

Consider a cellular network $\mathcal{B} = \{B_1, \ldots, B_n\}$ of n base-stations spread across a region. A *trajectory* T_j is represented as a sequence of tuples of the form $\Phi_i = \langle B_i, \Delta_i, \eta_i \rangle$ that captures the user experience. The user on this trajectory was connected to the base-station $B_i \in \mathcal{B}$ for a time interval of Δ_i units and received a throughput of η_i bytes per unit of time. Note that η_i can be any metric as long as a greater η_i denotes better experience (e.g., throughput or packet success rate) when associated to a respective base-station. Henceforth, for brevity, we refer to this metric as throughput. These trajectories and Φ_i's can be constructed by scanning the active transaction records maintained by the network operator.

For ease of notation, we write $B_i \in T_j$ if the base-station $B_i \in \mathcal{B}$ lies on the trajectory T_j. The length of a trajectory T_j, denoted by $|T_j|$, is simply the count of base-stations that lie on it. Suppose d denotes the maximum length of a trajectory.

For a trajectory T_j, a base-station $B_i \in T_j$ is a *bottleneck base-station* if it offers a degraded quality of service, e.g., an extremely low upload/download speed, a call-drop, etc. In our model, we assume that a base-station B_i acts as a bottleneck w.r.t. a trajectory T_j if the corresponding throughput is less than a threshold, i.e., $\eta_i < \psi$. The value of ψ is computed from a combination of network parameters. A base-station that is a bottleneck for one trajectory may *not* be a bottleneck for other trajectories since different users may experience different throughputs based on various factors such as data plan, time of the day, etc.

Our goal is to maximally improve the mobile user experience by selectively upgrading k out of n base-stations that act as bottlenecks on some trajectories. Suppose $\mathbf{X} = \{x_1, \ldots, x_n\}$ denotes the boolean solution vector such that $x_i = 1$ if and only if base-station B_i is chosen for upgradation and 0 otherwise.

Given a trajectory T_j, we define the weight w_{ji} for each base-station $B_i \in T_j$, that accounts for the fraction of the total time that the user (on this trajectory) was connected to the base-station B_i. More precisely, $w_{ji} = \frac{\Delta_i}{\sum_{B_i \in T_j} \Delta_i}$. Suppose b_{ji} denotes a bottleneck indicator variable that takes value 1 if the base-station $B_i \in T_j$ is a bottleneck base-station w.r.t. T_j, and 0 otherwise.

Given a trajectory T_j and solution $\mathbf{X}$, we define a *bottleneck utility* function W_j as follows:

$$W_j = \sum_{B_i \in T_j, b_{ji}=0} w_{ji} + \sum_{B_i \in T_j, b_{ji}=1} w_{ji}.x_i \qquad (2)$$

W_j essentially captures the fraction of the total time when the user enjoys acceptable QoE on the trajectory T_j. If all the base-stations on T_j are non-bottleneck, then $W_j = 1$; otherwise, $W_j < 1$. Henceforth, we consider the bottleneck utility function as the default trajectory utility function.

Based on this, we next define a class of trajectories that enjoy satisfactory QoE after upgradation of the base-stations.

DEFINITION 3 (γ-BOTTLENECK-FREE TRAJECTORY). *A trajectory $T_j \in \mathcal{T}$ is* γ-bottleneck-free *if its utility $W_j \geq \gamma$ where W_j is given by Eq.* (2)*, and $\gamma \in [0, 1]$ is the bottleneck parameter.*

Our aim is to maximize the number of trajectories with high utility. To do so, given the bottleneck utility function W_j, we map it to a step utility function U_j using a threshold γ ($0 \leq \gamma \leq 1$), henceforth referred to as the *bottleneck parameter*:

$$U_j = \begin{cases} 1 & \text{if } W_j \geq \gamma \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

We now formally state the Trajectory-Aware Macro-Cell Planning Problem, $\mathrm{TUMP}(\gamma)$.

PROBLEM 2 ($\mathrm{TUMP}(\gamma)$). *Given a base-station network $\mathcal{B}$ of size n, a budget parameter k, a bottleneck parameter γ, and a set of m trajectories $\mathcal{T} = \{T_1, \ldots, T_m\}$, each of which has an associated utility function W_j, determine the set of k base-stations*

to upgrade such that the sum of utilities $\sum_{T_j \in \mathcal{T}} U_j$ is maximized, where U_j is given by Eq. (3).

We show that the $\mathrm{TUMP}(\gamma)$ problem is NP-hard due to a reduction from the k-Vertex Cover (k-VC) problem [15].

Our problem formulation enables the network operator to suitably select the bottleneck parameter γ based on the application requirements. For example, $\gamma = 1$ is suitable for real-time applications such as voice calls or video conferences whereas $\gamma = 0.8$ may suffice for video streaming since video players can mask-off certain durations of low connectivity by buffering. Similarly, even $\gamma = 0.5$ may be enough for elastic applications such as background synchronization of emails.

3.1 Algorithms for TUMP

Since the $\mathrm{TUMP}(\gamma)$ problem is NP hard, we design four approximation algorithms based on linear programming, and greedy paradigm [15]. Among the proposed schemes, the algorithm that stands out in terms of quality and practical running times is DEC-GREEDY. This is a greedy algorithm that works on the principle of *minimizing marginal loss*. Initially, we assume that the solution comprises of the full set of bottleneck base-stations. Given that the budget is k, the algorithm runs for $n - k$ iterations, where in each iteration, it prunes away a base-station that results in minimal loss in the utility U. The approximation bound of this algorithm is $\binom{k}{d}/\binom{n}{d}$.

We show that DEC-GREEDY algorithm can be incrementally applied to an evolving network; i.e., as and when the operator allocates additional budget, this algorithm can be applied to incrementally evolve the network from one generation to another.

3.2 Summary of Experimental Results

Among the different algorithms that we design for the TUMP problem, DEC-GREEDY provides the optimum balance between quality and running time. On thorough empirical evaluation across multiple datasets emulating different city topologies, it is observed that DEC-GREEDY offers significantly higher quality than the other algorithms, especially at low budgets and higher γ, i.e., higher QoE requirements. For example, with an upgrade budget of $k = 20\%$ and $\gamma = 0.8$, DEC-GREEDY returns solutions that serve 3-8 times more number of users than an approach that uses a greedy location-based base-station upgrade.

We also observe that the investment required to provide satisfactory QoE to mobile users is dependent on the population distributions and their road-network. Specifically, cities with a dense central business districts, such as New York, need less budget to satisfy a large segment of mobile users than cities where businesses are spread out (e.g., Atlanta).

4. THE TIPS PROBLEM

Referring to Scenario 3, the next problem is related to the TOPS problem, which we refer to as Trajectory-aware Inconvenience-minimizing Placement of Services (TIPS), stated as follows:

PROBLEM 3 (TIPS). *Given a set of trajectories $\mathcal{T}$, a set of candidate sites $\mathcal{S}$ that can host the services, the TIPS problem with query parameter k seeks to report the best k sites, $\mathcal{Q} \subseteq \mathcal{S}$, $|\mathcal{Q}| = k$, that minimizes the maximum (average) inconvenience, i.e., the extra distance travelled by a commuting user in order to avail a service at her nearest service location.*

The maximum (average) inconvenience minimization problem is a generalization of the k-center problem [8] (k-medoidS problem [17]), and is thus, NP Hard. We design multiple heuristics for each of the two versions of the problem. We are working towards a scalable implementation of these heuristics. For evaluation, we use the same real and synthetic datasets, as in the case of the TOPS problem. The preliminary results show that the proposed solutions are highly effective and efficient.

Conclusions

With expansion of cities, and growing urban population, more people are required to commute longer distances, and thereby generating new demands for various services such as fuel stations, food joints, mobile services, retail stores, etc. In this light, identifying the best service-locations is highly critical to the success of these trajectory-aware services. The large-scale road networks and high volume of trajectories to be served, make these problems extremely challenging. This doctoral thesis introduces three such optimal location problems, namely TOPS, TUMP, and TIPS and develop efficient and effective frameworks to solve them.

5. REFERENCES

[1] O. Berman, D. Bertsimas, and R. C. Larson. Locating discretionary service facilities, ii: maximizing market size, minimizing inconvenience. *Operations Research*, 43(4):623–632, 1995.

[2] O. Berman, R. C. Larson, and N. Fouska. Optimal location of discretionary service facilities. *Transportation Science*, 26(3):201–211, 1992.

[3] M. Boccia, A. Sforza, and C. Sterle. Flow intercepting facility location: Problems, models and heuristics. *Journal of Mathematical Modelling and Algorithms*, 8(1):35–79, 2009.

[4] Z. Chen, Y. Liu, R. C.-W. Wong, J. Xiong, G. Mai, and C. Long. Efficient algorithms for optimal location queries in road networks. In *SIGMOD*, pages 123–134, 2014.

[5] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090. ACM, 2011.

[6] Z. Drezner. *Facility location: a survey of applications and methods*. Springer, 1995.

[7] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Computer and System Sciences*, 31(2):182–209, 1985.

[8] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[9] H. W. Hamacher and Z. Drezner. *Facility location: applications and theory*. Springer, 2002.

[10] W. He, D. Li, T. Zhang, L. An, M. Guo, and G. Chen. Mining regular routes from gps data for ridesharing recommendations. In *SIGKDD*, pages 79–86, 2012.

[11] V. Kolar, S. Ranu, A. P. Subramainan, Y. Shrinivasan, A. Telang, R. Kokku, and S. Raghavan. People in motion: Spatio-temporal analytics on call detail records. In *COMSNETS*, pages 1–4, 2014.

[12] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604. ACM, 2007.

[13] C. Long, R. C.-W. Wong, and H. Jagadish. Direction-preserving trajectory simplification. *Proceedings of the VLDB Endowment*, 6(10):949–960, 2013.

[14] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *GIS*, pages 352–361, 2009.

[15] S. Mitra, S. Ranu, V. Kolar, A. Telang, A. Bhattacharya, R. Kokku, and S. Raghavan. Trajectory aware macro-cell planning for mobile users. In *INFOCOM*, 2015.

[16] S. Mitra, R. Sharma, P. Saraf, A. Bhattacharya, and S. Ranu. Netclus: A scalable indexing framework for trajectory-aware placement of services. In *Under Review at VLDB*, 2016.

[17] W. S. Sarle. Finding groups in data: An introduction to cluster analysis. *Journal of the American Statistical Association*, 86(415):830–833, 1991.

[18] R. Song, W. Sun, B. Zheng, and Y. Zheng. Press: A novel framework of trajectory compression in road networks. *Proceedings of the VLDB Endowment*, 7(9):661–672, 2014.

[19] T.-H. Wu and J.-N. Lin. Solving the competitive discretionary service facility location problem. *European Journal of Operational Research*, 144(2):366–378, 2003.

[20] X. Xiao, B. Yao, and F. Li. Optimal location queries in road network databases. In *ICDE*, pages 804–815, 2011.

[21] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *KDD*, pages 316–324, 2011.

[22] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *SIGSPATIAL GIS*, 2010.