PlaceMap: Discovering Human Places of Interest Using Low-Energy Location Interfaces on Mobile Phones

Kuldeep Yadav
Xerox Research Centre, India
kuldeep.r@xerox.com

Vinayak Naik
IIIT-Delhi, India
naik@iiitd.ac.in

Abhishek Kumar
IIIT-Delhi, India
abhishek1202@iiitd.ac.in

Prateek Jassal
IIIT-Delhi, India
prateek1270@iiitd.ac.in

ABSTRACT

Emerging class of context-aware mobile applications, such as Google Now and Foursquare require continuous location sensing to deliver different location-aware services. Existing research, in finding location at higher abstraction, use GPS and WiFi location interfaces to discover places, which result in high power consumption. These interfaces are also not available on all feature phones that are in majority in developing countries.

In this paper, we present a framework *PlaceMap* that discovers different places and routes, solely using GSM information, i.e., Cell ID. *PlaceMap* stores and manages all the discovered places and routes, which are used to build spatio-temporal mobility profiles for the users. *PlaceMap* provides algorithms that can complement GSM-based place discovery with an initial WiFi-based training to increase accuracy. We performed a comprehensive offline evaluation of *PlaceMap* algorithms on two large real-world diverse datasets, self-collected dataset of 62 participants for 4 weeks in India and MDC dataset of 38 participants for 45 weeks in Switzerland. We found that *PlaceMap* is able to discover up to 81% of the places correctly as compared to GPS. To corroborate the potential of *PlaceMap* in real-world, we deployed a life-logging application for a small set of 18 participants and observed similar place discovery accuracy.

Categories and Subject Descriptors

C.3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: Real-time and embedded systems

Keywords

Place discovery; GSM; Energy-efficient; Places of Interest; GPS; WiFi

1. INTRODUCTION

Mobile applications consider user context to provide personalized services to the mobile users. For instance, Google Now tracks user's location continuously to push information such as weather, traffic, bus arrival time, etc. A user location context is high level information, e.g., places visited by users in their day-to-day lives and routes taken by users between the places, more than fine-grained latitude and longitude. Accurately finding everyday places from a user's mobility has critical importance for many context aware applications, as people spend about $80-90\%$ of the time indoors [12]. For example, many context-aware applications make use of place information to enable geo-reminders [9], participatory sensing [7], content-sharing decisions [21], advertisements [8], etc. Popular social networks, such as Foursquare and Facebook Places, also use place information for automatic checkins. Similarly, there are services, which need path travelled by a user. These include pollution exposure estimation [7], healthcare (i.e. activity logging), traffic estimation, ride-sharing, and advertisements.

A mobility profile for a user consists of all the places visited by her with accurate arrival and departure time for those places. Much of the current work use GPS and WiFi [17, 12, 13] to continuously track user's locations and find places by applying different clustering algorithms on those location coordinates. Current research work on finding places have limited impact in developing countries due to following reasons.

1. **Constrained by Feature Phones** A large number of phones are feature phones, especially in developing countries, which have limited capabilities, e.g., they lack location interfaces of GPS and WiFi [31, 14]. Due to their limited capabilities, feature phones are unable to use context aware applications, which require user context.

2. **High Energy Consumption** Finding place information using GPS and WiFi requires continuous tracking of location, which drains the phone battery. Yohan et al [10] observed that the reduction of battery life for GPS, WiFi, and GSM was 72%, 45%, and 18% respectively in comparison to baseline for 1400 mAh battery, while location information was sampled every minute.

3. **Limited Coverage** GPS is not available indoors. Further, there is a lack of *city-scale WiFi infrastructure* in many developing countries and hence, it is not possible to discover places or routes using WiFi data alone.

Due to above-described limitations, there is a need to design new ways of finding places. Many applications do not require high accuracy in case of place discovery. A place is inherently bigger than a location coordinate. Therefore, use of GSM information is a deserving alternative. In this paper, we propose a framework

ACM DEV-5 (2014), December 5–6, 2014, San Jose, CA, USA.
Copyright 2014 ACM 978-1-4503-2936-1/14/12 ...$15.00.
http://dx.doi.org/10.1145/2674377.2674386.

PlaceMap to discover places and routes visited by mobile users using only GSM information. In a nutshell, our framework keeps track of Cell IDs continuously and then uses a clustering algorithm to segregate Cell IDs according to physical places. One of the main challenges encountered by Cell ID clustering is that Cell IDs keep on changing even if user stays at a same place. Our clustering algorithm is able to deal with this noisy data. Apart from finding places, *PlaceMap* finds routes between two different places taken by mobile user with Cell ID information only.

We provide a version of *PlaceMap* with WiFi, if high accuracy in finding places is needed. Even in that case, *PlaceMap* uses WiFi only for the initial training. After the training is over, WiFi is not needed. For the initial training, *PlaceMap* compares places resulted from GSM-based place discovery with the more accurate information generated using WiFi and corrects its mistakes, such as merged or divided places. WiFi in itself is not used for finding location. Unlike traditional approaches, *PlaceMap* uses WiFi only to improve accuracy of locating places, which are found using GSM-based algorithms. If a user has repeated visits to same places, WiFi is not needed after the initial training. GSM-based place discovery algorithms suffices. The collected place signature information is used to track revisits to places and movements between different places. This signature information can also be shared with feature phone users to increase the accuracy of place discovery. This way, the phones without WiFi also provide high accuracy in finding places. The contribution of this paper are as follows:

1. A graph-based clustering algorithm *GCA* to discover places solely from Cell ID information across different users. To further improve accuracy of *GCA* for indoors, we developed an algorithm, which uses an initial training of WiFi data to learn places and later uses Cell ID data only. Based on extracted places using GSM data, we estimate the arrival and departure time as well as the routes that she takes between two places.
2. Evaluation of the proposed algorithms on two publicly available large mobility datasets both from developed country and developing country. We found that *PlaceMap* can discover nearly 69% places correctly using only GSM data in the Indian dataset as compared to that using WiFi. In MDC dataset, we found that *PlaceMap* was able to discover nearly 81% of the places correctly with the initial training of WiFi. *PlaceMap* records 80% of arrival and departure times to the revisited places with in a delay of 10 minutes. Further, it estimates the distance and duration of routes travelled by users with a median error of 1.47Km and 6.71minutes respectively.
3. We implement proposed framework *PlaceMap* as a Cloud service, which uses the above-mentioned algorithms. The cloud service is used to discover all the places visited by a user, arrival/departure time at those places, and frequent routes undertaken by her. As a proof-of-concept, we designed and deployed a life-logging mobile application which uses *PlaceMap* cloud service to 18 student participants for 2 weeks and observe that, it provides similar aggregate accuracy as when evaluated for the two large datasets.

In essence, the current focus of *PlaceMap* is to enable discovery of places using low energy location interfaces and not necessarily tagging them with their semantic labels. There are various approaches proposed by researchers, which can take place information from *PlaceMap* and assign semantic labels to each one of them [29].

2. DEFINITIONS

In this section, we define Place, Route, and Mobility Profile.

- **Place** A *Place* is defined as a location, where a user stays for non-trivial amount of time, e.g., "Home" and "Workplace". Burbey et al [19] considered a location as a place if the user has spent more than 10 minutes at that location. Depending on different location interfaces, a place can constitute a set of Cell IDs (i.e. c_i) or a set of WiFi APs (i.e. w_i), or a pair of GPS coordinates.
  $P_i = \{c_1,c_2,c_3,c_4,c_5\}$ or $P_i = \{w_1,w_2,w_3,w_4\}$ or
  $P_i = \{\text{latitude, longitude}\}$
- **Route** A *Route* is defined as a travel path taken by a user between two places. In the context of this paper, a route constitutes of a series of timestamp ordered GPS coordinates or a set of time ordered Cell IDs observed during travel duration.
  $R_i = \{(c_1,t_1),(c_2,t_2),\cdots,(c_k,t_k)\}$ or
  $R_i = \{(g_1,t_1),(g_2,t_2),\cdots,(g_k,t_k)\}$
  where $\{latitude, longitude\} \in g_i$ and t_i represents timestamp, when Cell ID or GPS coordinates is recorded.
- **Mobility Profile** A *Mobility Profile* is defined as a set containing (a) visited places along with their respective arrival and departure timestamps and (b) routes taken with their start times and end times. Mobility profile for a user X is represented as follows:
  $M_X = \{(P_1,a_1,d_1),(P_2,a_2,d_2),\cdots,(P_n,a_n,d_n)\} \cup \{(R_1,s_1,e_1),(R_2,s_2,e_2),\cdots,(R_m,s_m,e_m)\}$

3. PLACEMAP ALGORITHMS

In this section, we describe algorithms for building mobility profile, which is combination of places, their respective arrival and departure time, and route information.

3.1 Place Discovery

GSM APIs in majority of phones provide access to only one Cell ID to which the phone is connected and its corresponding RSSI (Received signal strength indication) [11]. WiFi-based place discovery methods assume access to multiple base stations or access points [17]. Hence, the signal fingerprinting method that works in case of WiFi does not work for majority of GSM phones. Previous work [16] has shown that even if a user stays at the same place, the Cell ID may change due to reasons, such as network load, small time signal fading, and inter-network ($2G$ to $3G$ or vice versa) handoffs. We call this change in Cell ID while the user is at the same place as "oscillating effect". To deal with this effect, we need a place discovery algorithm to cluster oscillating Cell IDs.

PlaceMap organizes GSM information, which is a tuple of MCC[1], MNC[2], LAC[3], and Cell ID, with timestamps. LAC is an identifier assigned to a group of nearby cell base stations. This tuple can uniquely identify a cell base station globally. To simplify description of algorithms, henceforth we will represent every base station only with its Cell ID and the corresponding timestamp. As an example, if a user's movement pattern in terms of Cell IDs is $\{c_1,c_2,c_2,c_1,c_1\}$ at time $\{t_1,t_2,t_3,t_4,t_5\}$ respectively, the information is represented as set of cell records, $\{(c_1,t_1),(c_2,t_2),\cdots,(c_1,t_5)\}$. The process of discovering places in *PlaceMap* is divided into two phases, the first is construction of clusters from day-specific cell records and the second is to map clusters to places

[1]Mobile Country Code
[2]Mobile Network Code
[3]Location Area Code

and keep learning this mapping as each day is passed. As part of first phase, we present three potential clustering algorithms for *PlaceMap*, which can cluster Cell IDs observed in a day's time period according to places. As a user's movement pattern is likely to be redundant across days, we consider a day's data as an input for clustering algorithms. Second phase involves continuous place discovery process, which utilizes day-specific clustering algorithms' output to maintain a global list of all discovered places of a user.

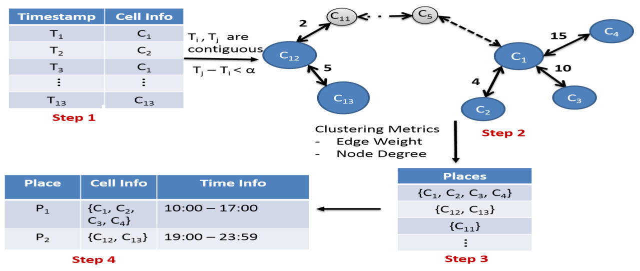
3.1.1 LAC-based Clustering Algorithm (LCA)

Cellular network provides same LAC identifier to a set of co-located cell base stations [18]. Therefore, the most naive approach is to use LAC to cluster oscillating Cell IDs. Cell ID clusters are produced from the given cell records, where each cell cluster contains Cell IDs belonging to the same LAC. Each of the LAC-based clusters (CC_l) can have one or more Cell IDs.

3.1.2 Graph-based Clustering Algorithm (GCA)

Let $\{(c_1, t_1), (c_2, t_2), \cdots, (c_k, t_k)\}$ be distinct time-ordered Cell IDs observed in a day. From the given day specific cell records, we build an undirected graph, called as *movement graph* $G(V, E)$, where $\forall_{i \in \{1,k\}} c_i \in V$ and there exist an edge $e(c_i, c_j)$ between c_i and c_j if c_i and c_j are contiguous in time ordered cell records and time difference between start time of c_j and end time of c_i is less than α.

As an example, in step 1 of Figure 1, c_1 and c_2 occurred contiguously and t_2-$t_1 \leq \alpha$, so there will be an edge between c_1 and c_2 in the corresponding movement graph of the user. Multiple edges between c_i and c_j are merged into a single edge with weight equal to that of number of edges between c_i and c_j. α ensures that an edge occurs only across successive Cell IDs. There are scenarios, in which two Cell IDs are contiguous, but the time difference between them is high due to switching off of the phone, unavailability of the network, or loss of location updates. Such Cell IDs may not be close to each other. In those scenarios, α is able to prune them effectively. An example of a representative movement graph created from given cell records is shown in step 2 of Figure 1.



Timestamp	Cell Info
T_1	C_1
T_2	C_2
T_3	C_1
⋮	⋮
T_{13}	C_{13}

Places
$\{C_1, C_2, C_3, C_4\}$
$\{C_{12}, C_{13}\}$
$\{C_{11}\}$
⋮

Place	Cell Info	Time Info
P_1	$\{C_1, C_2, C_3, C_4\}$	10:00 – 17:00
P_2	$\{C_{12}, C_{13}\}$	19:00 – 23:59

Figure 1: Various steps to discover places from the raw cell records and subsequently, building mobile profile using the place information.

We observed that Cell IDs, which belong to same cluster have high number of oscillations and thus high edge weight between them or forms a star like topology as shown in step 2 of figure Figure 1. We propose a three phase algorithm as described in Algorithm 1, which uses a oscillation threshold parameter η to cluster Cell IDs based on edge weight and an another threshold parameter η' to identify star topology in movement graph. *GCA* takes movement graph as an input and produces Cell ID clusters as an output, where each cluster of Cell IDs represents a different place as shown in step 3 of Figure 1.

Algorithm 1: Pseudocode of *GCA*

```
Algorithm: Graph-based Clustering Algorithm (GCA)
Input: Movement Graph $G(V, E)$ where $V$ is set of vertices
       and $E$ is the set of edges
Output: Set of Cell ID Clusters $CG$
begin
    /* First Phase */
    Rank all the edges in $E$ into decreasing order of their weight;
    $CG = \phi$ ;
    while $(\forall e_k \in E)$ AND $w(e_k) \geq \eta$ do
        if $v_i \in CG_j$ where $v_i \in e_k, i \in (1, 2), CG_j \in CG$ then
            $CG_j = CG_j \cup v_{k1} \cup v_{k2}$;
        else
            Create new cluster $CG_n = v_{k1} \cup v_{k2}$ and add it to $CG$ ;
    /* Second Phase */
    while $(\forall v_j \in CG_k)$ where $CG_k \in CG$ do
        if $(\Delta(v_j) \geq \eta')$ then
            $CG_k = CG_k \cup neighbors(v_j)$ ;

    /* Third Phase */
    $CG' = \phi$ ;
    while $(\forall CG_i \in CG)$ do
        $isExist = false$ ;
        while $(\forall CG_j \in CG')$ do
            if $(CG_i \cap CG_j) \neq \phi$ then
                $CG_i = CG_i \cup CG_j$ ;
                $isExist = true$ ;
                break ;
        if $\neg(isExist)$ then
            Add $CG_i$ to $CG'$ ;
    while $(\forall v_i \in V)$ do
        if $(v_i \notin \exists CG_k)$ where $CG_k \in CG'$ then
            Create new cluster $CG_n = v_i$ and add it to $CG'$ ;
    Copy $CG'$ into $CG$ ;
    return $CG$ ;
```

1. In the first phase of *GCA* , we start with initializing a cluster set CG to ϕ. All edges in the movement graph $G(V, E)$ are then sorted in descending order according to their weights. After sorting, *GCA* selects one edge at a time which has edge weight equal or more than η. If at least one of its two vertices is already in one of the cluster sets, say $CG_i \in CG$. If it is found, both the vertices of that edge are merged into CG_i. Otherwise, a new cluster is created containing only those two vertices and added to CG.

2. $\Delta(v)$ represents the degree of an vertex. In the second phase, we consider each vertex within the set of clusters (CG) and if $\Delta(v)$ is higher than or equal to η', all the neighboring vertices of v are also added to the cluster, which contains v, if they are not already included. We use η' as a threshold to classify vertices, which are central to different Cell ID transitions as compared to others. As an example, vertex represented by Cell ID c_1 in Figure 1 is central to Cell ID transitions with c_2, c_3 and c_4.

3. In the third phase, to create distinct and non-overlapping clusters, we combine all the clusters from CG, with a common vertex, to produce a new set of clusters CG' that does not have any common vertices across different clusters. All the left over vertices are added as a separate cluster in CG'. For instance, if a vertex v_i does not belong to any of the

clusters in CG', we create a new cluster CG_n that contains v_i and add it to set of clusters CG'. Finally, all clusters in CG' are added to CG.

As defined earlier, each vertex in every cluster $CG_i \in CG$ corresponds to a unique Cell ID and each cluster in CG represents a different place visited by the user in a day. We compare the accuracy of *GCA* with ground truth as well as LCA-based clustering in evaluation section.

3.1.3 *WiFi Trained Cell ID Clustering Algorithm (WTCA)*

GCA and LCA work on an assumption that a user will see different Cell IDs at different places. However, multiple places can be in close proximity, e.g., a student staying in a dorm that is close to academic building. If coverage of some cellular towers encompass both the dorm and the academic building, the user's phone may see overlapping set of Cell IDs in both the buildings. The cluster produced by *GCA* are mutually exclusive. In other words, the algorithm will merge the two nearby but different places and show them as one, even if there exists a single common Cell ID observed at each of the two places. This merging effect of *GCA* is also observed in our collected data as some of the participants live in dorms, which are close to academic building. For instance, if a user's phone observes Cell IDs $\{c_1,c_2,c_3,c_4\}$ at P_1 and Cell IDs $\{c_4,c_5,c_6\}$ at place P_2, Cell ID c_4 is common across places P_1 and P_2.

While such geographically close places may have overlapping Cell IDs, it is unlikely that they will have overlapping WiFi APs due to latter's comparatively smaller coverage. In the afore-mentioned example, if the WiFi APs seen at P_1 are disjoint from that seen at P_2, this information can be used to associate APs seen at P_1 with $\{c_1,c_2,c_3,c_4\}$ and those seen at P_2 with $\{c_4,c_5,c_6\}$. In this case, we can take into account disjoint Cell IDs $\{c_1,c_2,c_3\}$ and $\{c_5,c_6\}$ to distinguish between different places. We use this insight to extend *GCA* by training it with cell clusters found using mobility profile created using WiFi data. Building mobility profile using WiFi data has been extensively studied in previous research work [12, 17, 13]. We have used the UIM clustering algorithm [17] to discover places using WiFi data. This knowledge is then used to cluster Cell IDs. In particular, *PlaceMap* performs the following steps:

1. Use UIM clustering algorithm to build mobility profile MP_w using WiFi data. The resulting profile has all the places with their respective arrival and departure time information for a given day d. MP_{wd} = $\{(P_1, a_1, d_1), (P_2, a_2, d_2), \cdots, (P_n, a_n, d_n)\}$

2. For each place P_i in MP_w, find all the Cell IDs observed by the user and create a cluster with all those Cell IDs. If a user visits the same place at two different time intervals, we take a union of Cell IDs seen in both of these time intervals, to create a single Cell ID cluster corresponding to P_i.

3. For a given day d, once a cell cluster is found for each place in MP_w, all of them are put into a set of clusters CW.

After, *PlaceMap* collects WiFi data for d number of days, it computes WiFi-based Cell ID clusters observed for each day, say CW = $\{CW_1, CW_2, \cdots, CW_d\}$. A Cell ID is said to be conflicting if it belongs to two different Cell ID clusters for the same day. Such conflicting Cell IDs essentially belong to two different places and hence can not be relied upon when performing clustering. We create a separate conflicting set, C_C, that contains all such conflicting Cell IDs, which exist in CW.

Now, we have to cluster remaining non-conflicting Cell IDs in CW according to unique places. We define a support metric $s(c_i, c_j)$ for every non-conflicting Cell ID pair $c_i, c_j \in CW$ as $s(c_i, c_j) = \frac{O(c_i,c_j)}{min(O(c_i),O(c_j))}$, where $O(c_i, c_j)$ denotes the number of joint occurrences in days of c_i and c_j within the same cluster and $O(c_i)$ denotes all the occurrences in days of c_i, irrespective of whether c_j was in the same cluster as c_i or not. Two Cell IDs c_i and c_j are strongly connected and are in the same cluster, if $s(c_i, c_j) \geq \gamma$, where γ is system defined threshold. The high value of support metric i.e. $s(c_i, c_j)$ indicates that c_i and c_j are observed together in the same cluster for a large number of days. For instance, if we find $s(c_i, c_j) = 0.5$, it means that a pair of Cell IDs were observed in the same cluster in half of the total number of days in training period. On the same lines, if we use value of γ equal to 0.5, it means that a pair of Cell IDs belong to the same cluster more than half the total number of days. In that case, they are termed as strongly connected.

Once, support metric is computed for each non-conflicting Cell IDs using WiFi-based cell clusters. After that, GCA-based Cell ID clusters CG are refined to remove the merging effect with the help of following steps:

1. For each cluster $CG_i \in CG$, remove the Cell IDs that overlap with the conflicting set C_C and insert them into a separate cluster C_S. Correspondingly, CG is modified to a cluster set CG' for which Cell IDs across all the clusters are non-overlapping.

2. Correspondingly, for each cluster $CG'_i \in CG'$, if it was modified in the previous step then it is taken out from CG' and all its Cell IDs are added into a single cluster CG_n. Thereafter, Algorithm 2 is used to return one or more strongly connected clusters in CG_n, called SC.

3. Further, for each of the strongly connected clusters, add back the corresponding conflicting Cell IDs from C_S to each of its components, to create the modified strongly connected clusters SC'.

4. Final modified cluster set CG is obtained by combining clusters in SC' and CG'.

Using the WiFi training data, *WTCA* corrects the merging errors of GCA and subsequently forms a new set of Cell ID clusters, which minimize the number of merged places.

3.1.4 *Continuous Place Discovery*

One of the clustering algorithms i.e. *LCA*, *GCA*, or *WTCA* can be used to discover places from single day's cell records. Since, most of the time user visit same places, *PlaceMap* needs to have a continuous process to discover new places across different days and for different users. On the first day, *PlaceMap* initializes a list of places CC with the places discovered on that day using one of aforementioned clustering algorithms. After that, *PlaceMap* keeps using clustering algorithms to discover new day-specific places CG and updates existing list of places CC using them. *PlaceMap* performs the update using two different operations *merge* and *add*.

In *merge* operation, similarity between newly discovered places in CG is measured with existing places in CC. Here, similarity check is performed to find out whether a place has already been discovered or not. To compare similarity between places, we defined a metric, i.e.. cluster similarity index, which is computed as ratio of number of common Cell IDs to minimum size of among the two clusters. Size of a Cell ID cluster is equal to number of Cell

Algorithm 2: Pseudocode of Strongly Connected Clustering Algorithm

```
1  Algorithm: Strongly Connected Clusters Algorithm
   Input: $CG_i$ is a cluster of Cell IDs
   Output: Strongly connected clusters set $SC$
2  begin
3      $SC = \phi$ ;
4      while ($\forall c_j \in CG_i$) do
5          while ($\forall c_k \in CG_i$) do
6              if $s(c_j, c_k) > \gamma$ then
7                  if ($c_j \in SC_m$ OR $c_k \in SC_m$ where $SC_m \in SC$) then
8                      $SC_m = SC_m \cup c_j \cup c_k$ ;
9                  else
10                     Create a new cluster with $(c_j, c_k)$ and add it to $SC$ ;
11     return $SC$ ;
```

IDs contained in it. For any two places in CG and CC, it is computed as follows: $Cluster\ Similarity\ Index\ (CG_i, CC_j) = \frac{CG_i \cap CC_j}{min(|CG_i|, |CC_j|)}, where\ CG_i \in CG\ and\ CC_j \in CC$ If cluster similarity index is greater than ρ, it signifies that there is high similarity between cell clusters and CG_i is removed from CG and it is merged with CC_j. This process is followed for all the places, which are part of CG. In case of *WTCA* , conflicting Cell IDs are removed from both the clusters before measuring their mutual similarity.

PlaceMap bootstraps *add* operation after merge operation and all the leftover places of CG are added to CC. Essentially, CC contains all the discovered places at a given time in *PlaceMap* and it can be used for finding other information, such as arrival time discussed in subsequent subsection.

3.2 Measurement of Place Arrival and Departure Times

For creation of mobility profile of a user, *PlaceMap* needs to track her arrival and departure time information for every discovered place. In case of GSM-based place recognition, signature information consists of a set of Cell IDs. This signature information can be used to detect arrival and departure time of a user from a given place in real time. *PlaceMap* stores all the visited places' signatures and continuously tracks Cell ID information with sampling interval of 1 minute. If currently sensed Cell ID information belongs to one of the places' signature information for continuously t_a minutes, it records arrival at that place. *PlaceMap* uses a threshold of t_a minutes to reduce effect of occasional fluctuation among Cell IDs.

To detect departure from a place, *PlaceMap* keeps track of current Cell ID information to see if it belongs to a place signature, where it has recorded last arrival. Once, it detects a Cell ID, which does not belong to the last recorded place's signature information for consecutive t_d minutes, departure is recorded for that place. In the same way, arrival and departure time information of all the places are recorded in mobility profile of a user. Stay time of a user at a given place is the time difference between her arrival and departure times at that place.

3.3 Route Discovery

Route taken by a user to travel between a set of places is an important information for applications, e.g., PIER [7]. *PlaceMap*'s route finding algorithm takes help of arrival and departure time information to extract route information from mobility data. *PlaceMap* gives a flexibility to specify granularity of route tracking to mobile applications. Based on the application requirements, *PlaceMap* has two modes of route tracking, low accuracy mode and high accuracy mode. In low accuracy mode, once a user departs from a source place, *PlaceMap* starts tracking of current Cell ID information at a sampling interval of 1 minute. Route tracking will be on till user arrives at destination place and it will result in a sequence of Cell IDs. Apart from Cell IDs, route information will consist of start time of the route, which is equal to departure time of source place, and end time of route, which is equal to arrival time at destination place. Low accuracy route information is sufficient for applications, such as route based advertisements.

Some applications, such as crowd-sourced traffic information, need highly accurate route information, which can only be obtained by GPS. While working with such applications, *PlaceMap* opportunistically activates GPS, i.e., whenever a person departs from place, to obtain highly accurate tracking information similarly to approach presented in SenseLoc [12]. A high accuracy mode consumes higher energy as compared to that of low accuracy mode.

4. DATASETS FOR EVALUATION OF ALGORITHMS

We have used two large datasets to evaluate accuracy of *PlaceMap* algorithms. Both the datasets are publicly available.

1. **Self-collected Dataset:** We developed a data collection app for Android phones and deployed it on 62 participants's phones in New Delhi, India. The participants include students and staff members of our institute and they were made to sign a consent form before their participation in the study. The participants were selected using convenience sampling and only criteria used for recruitment was availability of Android phones.

   Our data collection app scans and logs GSM information every 1 minute and visible WiFi APs information every 10 minutes. Scanning intervals for WiFi is more than that of scanning GSM information because it results in higher energy consumption. GPS data was not collected due to higher energy consumption as compared to WiFi. Table 1 shows descriptive statistics about the dataset. To the best of our knowledge, it is first of its kind of data collection in India [15]. We have made the dataset publicly available after anonymization of personally identifiable information[4]. Spatial diversity of the dataset is high as it contains 11847 unique Cell IDs and 7717 unique WiFi APs.

Data	**Self Dataset**	**MDC Dataset**
Total GSM records	$1,131,509$	$8,029,388$
Total WiFi records	$109,286$	$2,856,858$
Total GPS records		$1,553,154$

Table 1: Descriptive statistics about both the datasets.

2. **MDC Dataset:** It is also a publicly available dataset, which was released as part of Nokia's Mobile Data Challenge (MDC) 2012 [26]. This dataset was collected in Switzerland from 2009 to 2011 using Nokia $N95$ smartphones and they have publicly released data of 38 participants. Dataset contains following different types of data, mobility data in terms of GPS, WiFi, and GSM interfaces, social interactions in terms of SMS and Bluetooth connections and application usage data.

[4]http://muc.iiitd.edu.in/datasets/

We only considered mobility data for our analysis. GSM information was scanned every 1 minute, WiFi scanning was performed every 2 minutes, and GPS coordinates were sampled every 10 seconds. Descriptive statistics of the dataset is given in Table 1. Spatial diversity of the dataset was high due to large duration of data collection and large number of participants. There were $18,321$ unique observed Cell IDs and $126,968$ unique WiFi APs.

5. EVALUATION OF ALGORITHMS

In this section, we evaluate algorithms for finding each of the mobility profile constituents using the aforementioned datasets. The output of proposed algorithms are compared against ground truth. Previous studies used manual inputs to collect baseline (ground truth), which is then compared with the outputs of the place discovery algorithms. However, most of those studies were for a short duration with a few participants. Due to limited scale of their data, it was feasible to collect diary based manual inputs from participants [12]. However in our datasets, given the number of participants and duration of the data collection, it is not scalable to collect human inputs.

In order for a scalable comparison, we derive the baseline using GPS and WiFi because they are considered more accurate compared to GSM [17, 25, 13]. The necessary condition is that the data from the two interfaces are collected simultaneously with GSM data. *PlaceMap* has two different type of algorithms, i.e., one which uses only GSM data and other one uses GSM with an initial training from WiFi data. Due to absence of GPS data, we evaluate our GSM-only algorithms with the baseline generated using WiFi data for self collected dataset. However, we use GPS data as baseline to evaluate both types of algorithms in case of MDC dataset.

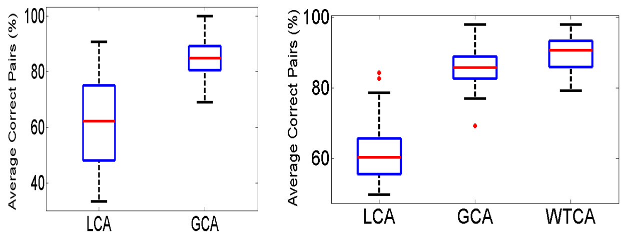
5.1 Evaluation of Place Discovery Algorithms

Place discovery algorithms of *PlaceMap* have two phases, day-specific clustering and continuous place discovery mechanism. We perform evaluation for both of these phase separately. However, accuracy of continuous place discovery mechanism is directly proportional to the accuracy of clustering algorithms *LCA*, *GCA*, and *WTCA*. For creating baseline, we implemented UIM algorithm [17] to build WiFi-based mobility profile and found corresponding day-based Cell ID clusters as presented in *WTCA* description. The set of clusters originated using WiFi are called as WiFi-based cell clusters and are used for evaluation of clustering algorithms.

In case of GPS, we use Kang et al [25] algorithm for clustering of GPS coordinates according to physical places. Kang et al's algorithm needs a time t and distance d threshold parameter for clustering, we use t=5 minutes and d = 200 meters in our settings. Similar to WiFi, we build GPS-based mobility profile and compute Cell ID cluster corresponding to each place in GPS based mobility profile. The set of Cell ID clusters computed using GSM based mobility profile are called as GPS-based cell clusters and are used to evaluate our clustering algorithms. The evaluation results of both the phases of the place discovery mechanism are as follows:

5.1.1 Evaluation of Clustering Algorithms

We represent Cell ID clusters produced using *LCA*, *GCA*, and *WTCA* algorithms as CL, CC, CWT respectively. For baseline, we represent WiFi-based and GPS-based Cell ID clusters as CW and CG. As shown in Table 2, we empirically find η and η' to be equal to 3 and use it for performing all experiments related to GCA. We define a metric *correct pair* for comparison between baseline and clusters produced by different clustering algorithms. A Cell ID pair C_i and C_j is counted as *Correct Pair*, if their occurrence within



(a) Self Dataset (b) MDC Dataset

Figure 2: Accuracy of clustering algorithms for all the participants in self and MDC datasets. *GCA* outperformed *LCA* by giving more correct pairs. *WTCA* improved upon GCA by correcting merged places.

the same or across different clusters in CW is reflected accordingly in the Cell ID based clustering approach evaluated. For instance, if C_i and C_j belongs to same cluster in CW then they should be in same cluster for the scheme under evaluation.

It is common to have holes in mobility data. For instance, WiFi may not be available at some places especially in developing countries. For fairness, we take precaution that comparison is done only when there is sufficient baseline data available for the day. Percentage of correct pairs found in evaluated scheme, say CC, is computed out of total pairs of Cell IDs in baseline CW for every day. After that, we compute average % correct pairs that is an average of all days of % correct pairs. Average % correct pairs depicts the final accuracy of an evaluated clustering algorithm and it is calculated for each participant separately. Similar process is followed to compute the accuracy of all clustering algorithms.

Figure 2 presents the distribution of all clustering algorithms' performance across all participants. Mobility characteristics of each participant is different, therefore, accuracy of clustering algorithms for users is likely to differ. As shown in box plot of Figure 2a, *GCA* produced 84.93% average correct pairs, while *LCA* produced 62.23% average correct pairs. For few participants, LCA provided good accuracy up to 90.75%. It is when a person visits places that are in distinct LAC areas. Errors in *GCA* occurred due to merging of places that are geographically close to each other. We did not evaluate *WTCA* clustering algorithm on self dataset due to unavailability of GPS data.

In case of MDC dataset, GCA produced 85.72% average correct pairs, while LCA produced 60.24% average correct pairs. Although both of these datasets were collected in two different countries and varied demographics, we find that improvement in clustering accuracy with *GCA* is consistent across both the datasets. This shows the generalizability of the *GCA* algorithm. *WTCA* is designed to correct the errors in *GCA* by identifying merged places and segregating those with the initial d days of training provided by WiFi-based cell clusters. We empirically found that $d = 8$ gives maximum possible accuracy as shown in Table 2 and used this value for all experiments. For calculating strongly connected components in *WTCA* , we empirically find that $\gamma = 0.5$ provides maximum possible accuracy. $\gamma = 0.5$ for each Cell ID pair means that they should be seen together in same cluster for at least half the number of training days. While calculating correct pairs using *WTCA* , we ignore Cell ID pairs containing conflicting Cell IDs because they can belong to any of those places. As shown in Figure 2b, *WTCA* provided 90.66% average correct pairs compared to 85.72% of *GCA* in MDC dataset. This improvement was recorded because *WTCA* split places, which *GCA* wrongfully merged, and put them into different clusters using the WiFi-based training. We observe that *WTCA* fails to correct some of merged places, when there is no distinct Cell ID for those places, i.e., all the Cell IDs are observed at both the places belongs to conflicting set.

From the evaluation of *PlaceMap* clustering algorithms, we find that *GCA* was able to build day-specific Cell ID clusters, which can distinguish between places visited by a user. However, *GCA* is prone to merge nearby places which are corrected using WiFi training by *WTCA* .

Algorithm	Parameter Values
GCA	$\eta = 3, \eta' = 3$
WTCA	$d = 8, \gamma = 0.5$
Continuous Place Discovery	$\rho = 0.55$
Measurement of Place Arrival and Departure Time	t_a=3, $t_d = 3$

Table 2: Consolidated list of tunable parameters and their values used for evaluation of *PlaceMap* algorithms

5.1.2 Evaluation of Continuous Place Discovery Mechanism

Here, we evaluate *PlaceMap*'s performance in discovering places for complete data collection duration unlike day-specific evaluation presented earlier. The places, which are discovered by *PlaceMap* , are called as *discovered places* and those discovered by GPS/WiFi baseline are called as *baseline places*. Baseline places, which are also discovered by *PlaceMap* are *baseline-discovered places* and the places which could not be discovered by *PlaceMap* are *missed* places as shown in Figure 3.

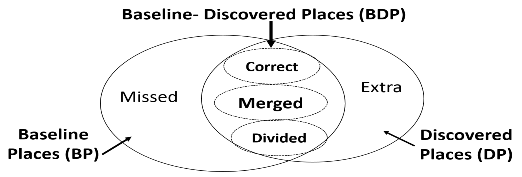


Figure 3: Relationship between different places discovered by baseline and *PlaceMap*

For comparison purpose, we build a mapping between baseline places and discovered places. For each place in baseline, we find a set of corresponding Cell IDs observed by user during her stay at that place across all days. The set of Cell IDs for each place in baseline produces a Cell ID cluster. We find a mapping between these two sets of clusters, i.e., one that is built using baseline called CW and second that is discovered using *PlaceMap* called CC. For every Cell ID cluster in baseline, we find all the corresponding similar clusters from CC using cluster similarity index described earlier. We consider two Cell ID clusters to be similar if value of Cluster Similarity Index is greater than a threshold δ. In case of *WTCA* , we remove conflicting Cell IDs before computing cluster similarity index score to minimize their impact. We define a mapping of places, between those found by baseline and *PlaceMap*. Following is an example of such mapping.

$Place\ Mapping = \{CW_i \rightarrow (CC_j); CW_m \rightarrow (CC_n);$
$CW_j \rightarrow (CC_i, CC_k); CW_k \rightarrow (CC_j); CW_l \rightarrow ()\}$

The place mapping is used to further classify baseline-discovered places into following different categories. These categories help in building evaluation metrics as mentioned in [12, 22].

- **Missing Place** A baseline place is said to be missed if it does not have any corresponding mapping found in discovered places, i.e., CW_l in above example.
- **Merged Place** A place is said to be merged if two different baseline places point to a single discovered place, i.e., CW_i and CW_k maps to the single place CC_j in above example.
- **Divided Place** A place is said to divided if a baseline place maps to two or more discovered places, i.e., CW_j represents a divided place in above example.
- **Correct Place** A place is called as correct if there is a single mapping of baseline place to discovered place, i.e., CW_m represents correct place in above example.

The value of ρ is crucial in continuous place discovery mechanism as lower value results in *divided* places where as higher value will result in *merged* places. Using empirical analysis, we derived $\rho = 0.55$ as shown in Table 2 and used the same value for all the experiments. For both the datasets, we build mapping between baseline and discovered places for each participant and then find instances of missing, correct, merged, and divided places. We do not find any instances of missing place in both the datasets as all the baselines places existed in some form among discovered places. The places, which were discovered by *PlaceMap* but did not exist in baseline, are called extra places. Extra places are due to holes in the baseline data, i.e., some locations may not have GPS/WiFi coverage.

In total, number of baseline places were 228 in self dataset and 1123 in MDC dataset. In self dataset, while using *PlaceMap*'s *GCA* clustering, about 69% places were found to be correct and 24% places were merged when compared with the baseline. For some participants (nearly 26%), correct places were over 90% because the mutual distance between their places was high and *PlaceMap* could easily distinguish them with the Cell IDs. As described earlier, range of cellular tower is large and therefore nearby places may observe similar Cell IDs. Hence, there were high instances of merged places in the self dataset. There were only 6.11% number of places, which were divided by *PlaceMap*. A division of a place happens primarily due to clustering errors.

For MDC dataset in comparison to the self dataset, there were only 20% merged places even while using *GCA* as shown in Figure 4. This is one of side effect of using GPS for baseline because place clustering using GPS can not distinguish between places, which are close by [12]. However, the number of correct places increased to 81% from 75%, while using *WTCA* clustering. Our place evaluation results demonstrate that *PlaceMap* with *WTCA* is able to discover up to 81% of places correctly. In case of MDC dataset too, for some of the participants (nearly 31%), correct places were over 90% due to high mutual distance between their visited places.

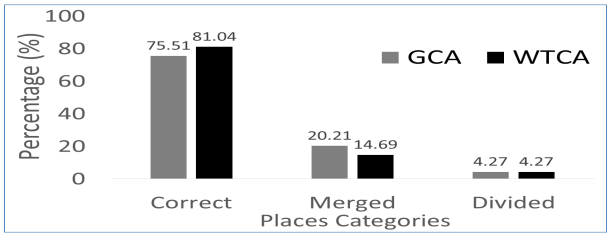
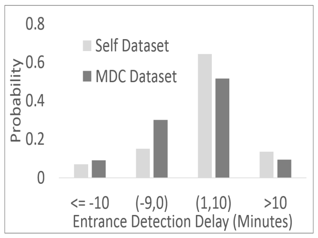
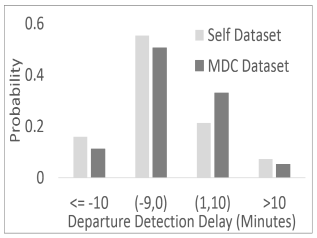


Figure 4: Places discovered by *GCA* and *WTCA* in MDC dataset across all participants; Nearly 81% places were found to be correct using *WTCA* as compared to baseline (GPS)

We find that *PlaceMap* with *GCA* clustering leads to merging of nearby places. Some of these merged places are corrected by



(a)



(b)

Figure 5: (a) Distribution of arrival detection delay; (b): Distribution of departure detection delay

WTCA clustering however, we conclude that *PlaceMap* is effective in discovering places with a granularity of a building and not necessarily room level granularity provided by WiFi-based techniques [12].

5.2 Evaluation of Arrival and Departure Time Estimation

We find arrival and departure time of a user for each visited place using baseline data. The baseline mobility profile of a user for a day is represented as $M_b = \{(P_1, a_1, d_1), (P_2, a_2, d_2), \cdots, (P_n, a_n, d_n)\}$, whereas mobility profile created by *PlaceMap* is represented as M_p = $\{(P_1, a_1, d_1), (P_2, a_2, d_2), \cdots, (P_k, a_k, d_k)\}$.

After that, $PlaceMapping(PM)$ is built between places in M_b and M_p as described in Section 5.1.2. Assuming, $P_i \in M_b$ and $P_j \in M_p$ are found to be same, we use following two metrics to evaluate accuracy of *PlaceMap*.

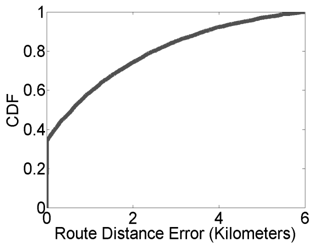
Arrival Detection Delay = Baseline Arrival Time (P_i) - *PlaceMap* Arrival Time (P_j)
Departure Detection Delay = Baseline Departure Time (P_i) - *PlaceMap* Departure Time (P_j)

For every place in M_b and M_p, which is discovered correctly, we find out the arrival detection delay and departure detection delay. For the places, which are merged in *PlaceMap* due to clustering, we assume those as a single place in baseline too and subsequently, compute their arrival and departure detection delays. Figure 5a shows the distribution of arrival detection delay for the all the participants in both the datasets. The negative values represents that *PlaceMap* detected the place after arrival of a user at a place. Nearly 80% of total place arrivals were detected within a delay of 10 minutes by *PlaceMap* for both the datasets. In nearly 20% of the cases, arrival detection delay is more than 10 minutes, which is due to clustering errors or missing data. As we considered merged places by *PlaceMap* service as single place for this evaluation, we did not see any noticeable difference among *PlaceMap* variants i.e. *GCA* and *WTCA* in estimating arrival and departure times.

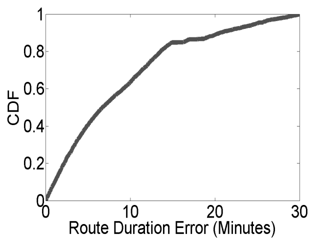
Departure detection delay for 76% of the total departures was less than 10 minutes in case of self dataset. In MDC dataset, nearly 83% place departures had a detection delay of less than 10 minutes. The median stay duration across all places were around 60 minutes. Hence, we believe that *PlaceMap* can be used in application scenarios, which can tolerate inaccuracy of few minutes in detecting and arrival of a user at a place.

5.3 Evaluation of Route Discovery

Using baseline of GPS, *PlaceMap* records information about the route. The route information consists of series of time-sorted GPS geo-coordinates, the first being start time and the last being end time of the route. The process for finding routes from GPS data works exactly opposite to the process of finding stay points [25]. We compute the total distance travelled and time duration from for



(a)



(b)

Figure 6: (a) Distribution of arrival detection delay; (b): Distribution of departure detection delay

each route. In *PlaceMap*, a route consists of time-ordered Cell IDs observed during the travel. We convert these Cell IDs into corresponding geo-coordinates with the help of Cell ID databases [5] and compute the distance traveled for each route. Routes found by *PlaceMap* are then compared with baseline using two metrics i.e. *route distance error* which is absolute difference between route distance measured by baseline and that estimated using *PlaceMap* and *route duration error* which is absolute difference between route duration measured by baseline and that estimated using *PlaceMap* .

In total, we find non-distinct $7,258$ routes using baseline data for all the participants in MDC dataset. The median route duration is 17.58 minutes and median route distance is 8.03 Km. We observe that whenever a user travels between nearby places, path tracking is not enabled by *PlaceMap* due to failure of departure time detection, possibly because of place merging effect originated from clustering errors described in Section 5.1.2. The routes, which are missed by *PlaceMap* are called *missing routes*. We find that *PlaceMap* missed nearly 35% of routes as compared to the baseline. As the routes are not distinct, we observe that most of these routes are for a short distance (90% percentile of route distance was 2.52 Km.)and they occur very frequently given redundancy in user's mobility.

For the routes, which are detected by *PlaceMap* and existed in baseline, we compute the route distance error and route duration error as shown in Figure 6. As shown in Figure 6a, median route distance error is 1.47 Km and $75th$ percentile error is 2.83 Km. Route distance error is introduced by crowdsourced geo-coordinates of Cell IDs because range of a Cell ID in urban area could be up to few Kms. As shown in Figure 6b, *PlaceMap* has median route duration error of 6.71 minutes and $75th$ percentile error is 12.44 minutes. Route duration error in *PlaceMap* is introduced by errors in detecting departure and arrival time based only on Cell ID. We do not provide any evaluation results on self dataset due to lack of GPS data. Based on our evaluation results on two datasets, we conclude that *PlaceMap* can make errors of few Kms in distance estimation and several minutes in case of estimating route duration. Most of these errors are induced by inaccuracies in place arrival and departure detection.

6. CASE STUDY

There are a set of mobile applications, which automatically log personal mobility history using location interfaces, such as WiFi and GPS [3, 1]. For instance, LifeMap [1] provides a visualization of places visited by a person, average time spent on those places, etc. However, LifeMap primarily takes help of WiFi APs to learn places in a user's mobility profile. A continuous scans of WiFi APs consume significant energy, a limitation observed by many of reviewers of LifeMap application. We implemented *PlaceMap* as an Azure cloud service and subsequently, designed and developed a mobile application for personal mobility history logging that uses

PlaceMap cloud service. A beta version of the application is publicly available on Google Play[2]. We have extended this application to add day-to-day diary logging features. The modified mobile application was deployed with 18 student participants and they were asked to use it for history logging of their personal mobility.

Figure 7 presents snapshots of aforementioned mobile application. Apart from automatic place discovery, participants were advised to tag their places. For every place tag, participants need to record arrival and departure time from a place. We requested participants to provide human inputs to capture their perception about places and to understand semantic meaning of places using their tags. As shown in Figure 7a, user is able to visualize all the visited places on a map interface. She has flexibility to tag a place with a customized name and icon, which appears into list of places as shown in Figure 7b. *PlaceMap* has a capability to store and manage long-term mobility history of a user. Our mobile application uses that capability to present fine-grained information to the user about her stay time at visited places and visiting days. Figure 7c presents a snapshot of context-based advertisements pushed to the user's device based on her mobility profile and preferences.

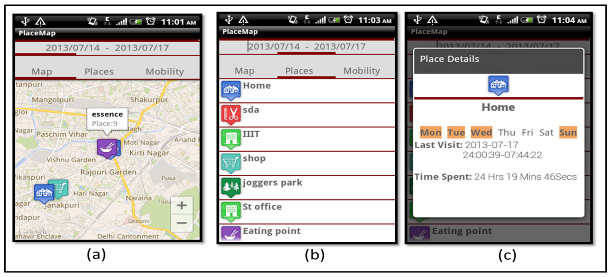


Figure 7: Snapshots of mobility history logging mobile application that uses *PlaceMap* cloud service.

Out of 18 participants, we ignore data of 2 participants as they did not tag even a single place in the given duration of 2 weeks. There were total 123 places discovered by *PlaceMap* from rest of the participants. In total, participants tagged 85 places. However, some of the tagged places did not contain departure information. From the rest of the 62 places, we found that *PlaceMap* using *WTCA* was able to correctly discover 79.03% of the places, merged 14.52% of places, and divided 6.45% of places. From this user study evaluation, we conclude that *PlaceMap* has shown effectiveness in discovering places in the wild. These results are comparable to the earlier analysis done on large datasets. Further, we observed that most of merged places in *PlaceMap* were very close to each other i.e. academic building and library. We are planning large-scale deployment going forward. We will also be taking help of users in reducing percentage of merged places.

7. RELATED WORK

In this section, we describe related work, which deals with energy-efficient continuous location sensing and finding mobility profile in terms of places and routes. In continuous location sensing, many mobile systems adaptively duty cycle GPS using different approaches i.e. with the help of movement detection using sensors such as accelerometer [11, 24], Cell ID sequencing [27], pre-assessment of different application requirements [24] and by incorporating street segment data [23].

Some approaches use GPS, WiFi, and GSM data individually or collectively to discover places for a user. Kang et al [25] designed a clustering algorithm to find places using GPS coordinates based on temporal and spatial stay threshold. Zhou et al [22] uses Density and Join (DJ) clustering algorithm to discover places using GPS coordinates. In earlier section, we used algorithm proposed by Kang et al [25] as baseline for comparing *PlaceMap* place discovery algorithms. Jyotish [17] proposes an algorithm that clusters WiFi APs into physical places. We use Jyotish to cluster WiFi APs according to places for creating baseline and compare it with places generated using *PlaceMap*. Senseloc [12] uses repetitive WiFi scans to learn about arrival and departure from a place. To track travel paths, Senseloc uses GPS whenever it detects that user is traveling. SmartDC [13] uses a three level triggered sensing scheme to discover places in a user's mobility profile. The levels are LAC, WiFi, and GPS. Unlike Sensloc, SmartDC duty cycles location sensors based on the past mobility, thereby saving energy. However, there are large number of phones, which do not have GPS and WiFi sensors. WiFi-based place sensing schemes require WiFi APs infrastructure, which is not widely available in many parts of the world.

Demirbas et al [16] uses GSM data to generate spatio-temporal mobility profile in reality mining dataset [4]. Most of the Cell IDs in reality mining dataset have a place label attached, provided by the participants during data collection process. Place labels have been used for clustering Cell IDs with respect to different places along with a circular subsequence algorithm to recognize oscillating Cell IDs. In practice, it is infeasible to get place labels in large scale deployments and also, Demirbas et al did not provide any evaluation of the produced clusters. Lassonen et al [28] presented a Cell ID clustering algorithm based on cell graph. The algorithm is similar to movement graph in *PlaceMap* without any edge weights. Their cluster merging algorithm combines Cell ID clusters with even one common Cell ID. This may unnecessarily merge distinct places.

Our work differs from these work in several aspects.
First, *PlaceMap* uses a novel clustering approach that creates a edge weighted movement graph to model Cell ID fluctuations and subsequently discovers clusters of Cell IDs. Second, we provide algorithms for identifying and segregating nearby places, which otherwise can get merged due to large coverage of cell towers, with the help of training provided by WiFi. Third, we define metrics to compare Cell ID clusters with baseline and presents evaluation results on two diverse long duration datasets.

8. CONCLUSION AND DISCUSSION

Many context-aware mobile applications and services require continuous location sensing to infer person-specific mobility profile. Currently available solutions, based on GPS and WiFi, are limited in their reach to people and result in high energy consumption. Further, for discovering places, a significant number of mobile applications do not require fine-grained accuracy, such as at the room-level, as provided by WiFi. This paper tries to fill this gap and proposes *PlaceMap*, which uses widely available GSM interface to discover places and routes to build a users' mobility profiles.

As part of *PlaceMap*, we propose a graph-based clustering algorithm *GCA* to discover places visited by a person solely from Cell ID information. If an application wants accuracy at the room or floor level, *PlaceMap* can use limited duration of WiFi-based initial training to learn the places. Further, *PlaceMap* has mechanism to keep discovering new places along with tracking revisits to existing places with their arrival and departure times. *PlaceMap* provides an algorithm to discover routes travelled by users and estimates routes' durations and distances. We performed a comprehensive evaluation of *PlaceMap* algorithms using two publicly available large datasets collected in diverse settings. Our evaluation results show that *PlaceMap* is able to correctly discover up to 81% of total places visited by users and detects nearly 80% of place visits as compared to GPS within 10 minutes of arrival at the place. Also, for nearly one third of users in both the datasets, *PlaceMap* provides more than 90% of the correct places. *PlaceMap* provides high accuracy whenever mutual distance between places visited by a user is relatively high. In case of routes, we found that *PlaceMap* can estimate route distance and duration with a median error of 1.47Km and 6.71 minutes respectively.

PlaceMap has a modular architecture and can adapt according to different applications' requirements. For instance, applications can use only GSM-based place discovery, if they are executing on feature phones. We also envision that the WiFi-based initial training can be crowdsourced to share it with features phone users who visits same places. As part of this work, we present a life logging application, which uses *PlaceMap* to discover and manage user's mobility profiles. We believe that *PlaceMap* can help in bootstrapping several of these applications, which need to track user's context continuously. Also, we are working to provide privacy guarantees to the users of *PlaceMap* by designing a privacy preserving matching service.

9. ACKNOWLEDGMENTS

This work is partially sponsored by Microsoft Research PhD Fellowship, Nokia Research, and Information Technology Research Academy (ITRA) grants. We would also like to thank all the participants who contributed their mobility data into our datasets.

10. REFERENCES

[1] LifeMap Google Play Application, `https://play.google.com/store/apps/details?id=com.mobed.lifemap`

[2] PlaceMap Google Play Application, `https://play.google.com/store/apps/details?id=com.iiitd.muc.placemap`

[3] Moves Google Play Application, `https://play.google.com/store/apps/details?id=com.protogeo.moves`

[4] MIT Reality Mining Dataset, `http://realitycommons.media.mit.edu/realitymining.html`

[5] Open Cell ID Database, `www.opencellid.org`

[6] Barabi A., Understanding individual human mobility patterns," Nature 453, 779-782.

[7] Mun, Min, et al. "PEIR, the personal environmental impact report, as a platform for participatory sensing systems research." ACM MobiSys'09.

[8] Khan, A. J., Subbaraju, V., Misra, A., & Seshan, S. Mitigating the true cost of advertisement-supported free mobile applications. ACM HotMobile'12.

[9] Ludford, P. J., et al. Because I carry my cell phone anyway: functional location-based reminder applications. ACM CHI 2006.

[10] Chon, Yohan, Wanchang Ryu, and Hojung Cha. "Predicting smartphone battery usage using cell tower ID monitoring." Pervasive and Mobile Computing (2013).

[11] Jeongyeup P., Kim J., and Govindan R. "Energy-efficient rate-adaptive gps-based positioning for smartphones." ACM MobiSys'10.

[12] Kim, Donnie H., et al. "SensLoc: sensing everyday places and paths using less energy." ACM SenSys'10.

[13] Chon, Yohan, et al. "SmartDC: Mobility Prediction-based Adaptive Duty Cycling for Everyday Location Monitoring.", IEEE Transactions on Mobile Computing, 2013.

[14] Yadav K., Naik V., Singh A., Singh P.,Kumaraguru P., and Chandra U.,Challenges and novelties while using mobile phones as ICT devices for Indian masses: short paper, NSDR'10.

[15] K. Yadav, A. Kumar, A. Bharti, and V.Naik . Characterizing Mobility Patterns of People in Developing Countries using Their Mobile Phone Data, COMSNETS 2014.

[16] Bayir M.A., Demirbas M.,PRO, and Eagle N.: Discovering spatiotemporal mobility profiles of cellphone users, WOWMOM'09.

[17] Vu L., Do Q., and Nahrstedt K., Jyotish: Constructive approach for context predictions of people movement from joint Wifi/Bluetooth trace, IEEE PerCom'11.

[18] Ficek M., Kencl L., Spatial extension of the Reality Mining Dataset. MASS 2010.

[19] Burbey I.E., Predicting Future Locations and Arrival Times of Individuals. Doctoral Thesis, Blacksburg, Virginia, April 2011.

[20] Thiagarajan, A. et al. Accurate, low-energy trajectory mapping for mobile devices. USENIX NSDI 2011.

[21] Yadav, K., Naik, V., & Singh, A. (2012). MobiShare: cloud-enabled opportunistic content sharing among mobile peers. IIIT-D Technical Report, IIITD-TR-2012-009.

[22] Changqing Z. et al. "Discovering personally meaningful places: An interactive clustering approach." ACM Transactions on Information Systems (TOIS) 25, no. 3 (2007): 12.

[23] Wang, H. et al. "WheelLoc: Enabling Continuous Location Service on Mobile Phone for Outdoor Scenarios.", IEEE INFOCOM'13.

[24] Kaisen L., Kansal A., Lymberopoulos D., and Zhao F. "Energy-accuracy trade-off for continuous mobile device location." ACM MobiSys 2010.

[25] Kang J., Welbourne W., Stewart B., and Borriello G. "Extracting places from traces of locations." ACM workshop on Wireless mobile applications and services on WLAN hotspots'04.

[26] Laurila, Juha K. et al. "The mobile data challenge: Big data for mobile computing research." In Proceedings of the Workshop on the Nokia Mobile Data Challenge, Pervasive 2012.

[27] Jeongyeup, Kim K, Singh J., and Govindan R. "Energy-efficient positioning for smartphones using cell-id sequence matching." ACM MobiSys 2011.

[28] Kari L. et al. "Adaptive on-device location recognition." In Pervasive Computing, pp. 287-304.

[29] Do, T., and Gatica-Perez D.. "The Places of Our Lives: Visiting Patterns and Automatic Labeling from Longitudinal Smartphone Data." IEEE Transactions on Mobile Computing, 2013.

[30] Iacopo C. et al., "Matador: Mobile Task Detector for Context-Aware Crowd-Sensing Campaigns.", PERCOM 2013

[31] Yadav, Kuldeep, et al. "Low Energy and Sufficiently Accurate Localization for Non-Smartphones." IEEE MDM'12.