B-Planner: Night Bus Route Planning using Large-scale Taxi GPS Traces

Chao Chen[†], Daqing Zhang[†], Zhi-Hua Zhou[‡], Nan Li[‡,♮], Tülin Atmaca[†], and Shijian Li[♭]
[†] CNRS SAMOVAR, Institut Mines-TELECOM/TELECOM SudParis, Evry 91011, France
[‡] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[♮] School of Mathematical Sciences, Soochow University, Suzhou 215006, China
[♭] Department of Computer Science, Zhejiang University, Hangzhou 310027, China

***Abstract*—Taxi GPS traces provide us with rich information about the human mobility pattern in modern cities. Instead of designing the bus route based on inaccurate human survey regarding people's mobility pattern, we intend to address the night-bus route planning issue by leveraging taxi GPS traces. In this paper, we propose a two-phase approach based on the crowd-sourced GPS data for night-bus route planning. In the first phase, we develop a process to cluster "hot" areas with dense passenger pick-up/drop-off, and then propose effective methods to split big "hot" areas into clusters and identify a location in each cluster as a candidate bus stop. In the second phase, given the bus route origin, destination, candidate bus stops as well as bus operation time constraints, we derive several effective rules to build bus routing graph and prune the invalid stops and edges iteratively. We further develop two heuristic algorithms to automatically generate candidate bus routes, and finally we select the best route which expects the maximum number of passengers under the given conditions. To validate the effectiveness of the proposed approach, extensive empirical studies are performed on a real-world taxi GPS data set which contains more than 1.57 million passenger delivery trips, generated by 7,600 taxis for a month in Hangzhou, China.**

***Index Terms*—Taxi GPS Traces; Human Movement Patterns; Bus Routes Planning**

I. Introduction

Buses are a popular and economical way for people to travel around the city, and they are generally "greener" than cars and taxis as they help to decrease traffic congestion, fuel consumption, carbon dioxide emission and travel cost. Thus for sustainable city development, people are encouraged to take public transportation for work, visit, etc. In many cities, the daytime bus transportation systems are usually well designed; however, during late night, most bus systems are out of service, leaving taxis as the only way for getting around. Many cities start to plan night-through bus systems to provide cost-effective and environment friendly transport to citizens.

With the increasingly wide deployment of GPS devices and pervasive sensors, more and more digital traces left by people while interacting with cyber-physical spaces have been accumulated [16], [17], [18]. For instance, taxis in many cities are equipped with GPS devices nowadays, and rich information about the taxis, including where and when passengers are picked-up or dropped-off, which route a taxi takes for a certain trip, can be collected and extracted. This big crowd-sourced data collected using pervasive sensing contains passengers' time-dependent mobility patterns in a city, making it possible to optimally plan night-bus routes by estimating the number of passengers expected along the routes. Previously, bus route planning mainly relied on human surveys to understand the people's mobility patterns [7]. Although this approach was proved to be workable, the time and cost spent in the survey process are quite substantial. Pervasive sensing, communication and computing bring us new ways to sense the pulses of the city at real-time and low cost, understand the situation collectively and quantitatively, and create opportunities to enable new applications in urban planning.

In this paper, we intend to explore the night-bus route design problem leveraging the taxi GPS traces. First of all, we need to identify the candidate bus stops which are associated with locations having big number of taxi passenger pick-up and drop-off records (*PDRs*), the bus stops should be evenly distributed in the "hot" districts to facilitate people's access. After the candidate bus stops are fixed, the next step is to select a bus route which connects the bus origin and a sequence of bus stops to the destination, carrying the maximum number of passengers within a defined time duration. Fortunately, the taxi GPS traces contain quantitative spatial-temporal information about all taxi trips. By mining the taxi GPS data, we can inform where are the "hot" areas for taxi passengers and how many passengers would potentially travel along a certain route. Therefore, the night-bus route design becomes a problem of comparing the number of passengers of all valid bus routes giving certain time constraints.

However, identifying the candidate bus stops from taxi GPS data and enumerating the top-ranked bus routes efficiently are not trivial and straight-forward. To the best of our knowledge, there is still no work reported on candidate bus stop identification and bus route design leveraging taxi GPS data. Consider the taxi GPS trajectories and the converted bus routing graph shown in Fig. 1, seven dense taxi pick-up/drop-off locations (i.e. $C_1 - C_7$) are identified as candidate bus stops, where C_1 and C_7 are designated as the bus origin and destination, respectively. The objective of bus route design is to find a bus route from C_1 to C_7 with maximum number of passengers expected given the bus operation frequency and total travel time. Apparently, to design an effective bus route, we need to address the following research challenges:

First, the taxi passenger pick-up and drop-off points are

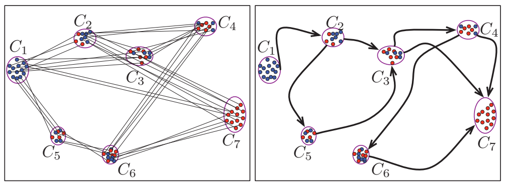

Fig. 1. An illustrative example of the taxi GPS trajectories (left) and the converted bus routing graph (right).

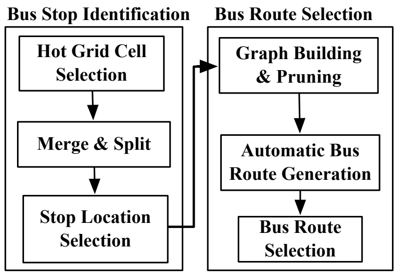

Fig. 2. The two-phase bus route planning framework.

distributed in the whole city, with some areas having more *PDRs* than other areas, but there is no clear guideline about where the bus stops should be put. Thus there needs a method to identify candidate bus stops from taxi passenger pick-up/drop-off distributions.

Second, to deliver the maximum number of passengers, the best bus route should leave from the bus origin C_1, go through all the intermediate bus stops, and finally reach the destination C_7. That is, the bus route would follow the sequence of bus stops as $C_1 \rightarrow C_2 \rightarrow C_5 \rightarrow C_3 \rightarrow C_4 \rightarrow C_6 \rightarrow C_7$. However, the problem for this route is that the whole trip would take a very long time to complete, which is intolerable if the number of candidate bus stops is big. Thus, a non-trivial trade-off has to be made between the number of passengers expected along the route and the total time travelled.

Third, as there is no taxi passenger travelling from C_4 to C_7 in Fig. 1 (left), if we design the bus route as $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_7$, then the significant passenger flow in paths $C_2 \rightarrow C_4$ and $C_3 \rightarrow C_4$ cannot be accommodated. Alternatively, by including C_4 in the planned bus route as $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4 \rightarrow C_7$, all the passenger flows in $C_2 \rightarrow C_4$, $C_3 \rightarrow C_4$, $C_2 \rightarrow C_7$ and $C_3 \rightarrow C_7$ are accommodated with the cost of adding one more stop. Therefore, even the path $C_4 \rightarrow C_7$ has no passenger flow, but adding it to the planned bus route would lead to a better solution due to passenger flow accumulation from all previous stops and paths.

Finally, besides the passenger flow accumulation along the bus route, we also need to consider that the passenger flows are usually different from time to time. For instance, the passenger flow during 23:00-24:00 might be very different from that during 3:00-4:00. Thus we have to consider the passenger flows accumulated and the total number of passengers in all concerned timeslots while selecting the planned bus route.

In this paper, we propose a two-phase approach to address the above-mentioned challenges. In the first phase, we identify the candidate bus stops leveraging the taxi GPS data. In the second phase, with all the candidate bus stops identified and the bus route OD designated, we develop effective rules and heuristic algorithms to generate the bus route with maximum number of passengers expected under the time constraints, with the process shown in Fig. 2. In summary, the main contributions of this paper include:

First, we propose a two-phase approach to tackle the night-bus route design problem leveraging the taxi GPS data. To the best of our knowledge, this is the first work on night bus route design using the taxi travel speed, time and *PDRs* information.

Second, we develop a novel process with effective methods to cluster “hot” areas with dense passenger pick-up/drop-off, split big “hot” areas into walkable size ones and identify candidate bus stops. It is verified that the proposed method outperforms the popular k-means method in terms of soundness of selected bus stop location and evenness of selected bus stop distribution.

Third, we derive several effective rules to build the directed bus routing graph where nodes and edges represent the candidate bus stops and valid connections among stops, respectively. To ensure the bus would reach destination in the end and reduce the computation complexity, we also develop an iterative process to remove invalid nodes and edges.

Finally, we propose two heuristic algorithms for automatically generating valid bus routes. One is the *probability based spreading algorithm* which randomly selects the next stop among the possible candidate stops in each step, giving the candidate stop with high accumulated passenger flow a big probability for random selection; the other is the *top-k spreading algorithm* which selects k nodes with high accumulated passenger flows as candidate stops in each step. It is verified that the *probability based spreading algorithm* outperforms the *top-k* approach in the selection of best bus routes.

II. Related work

Here, we briefly review the related work which can be grouped into two categories. The first category is about making use of taxi GPS traces for urban planning and traffic management. The existing work includes automatic map construction [3], detecting hot spots and frequent travel patterns [8], predicting road traffic conditions [4], informing land use and function distribution [11], [13], uncovering inefficient road network connectivity [18], planning optimal driving route [14] and various applications such as next passenger finding [15], anomalous trajectory discovery [17]. Among the taxi GPS trace related papers, the work addressing “hotspots” and frequent travel OD patterns are relevant to our work for identifying candidate bus stops and providing passenger flow data among potential bus stops, but there is no paper except one [2] using those data for bus route planning. The main goal of [2] is to mine historic taxi GPS trips to suggest a flexible bus route. The work first clusters trips with similar starting time, duration, origin and destination; it then attempts to identify the route that connects multiple dense taxi trip clusters. The work is different from ours as it only chooses the route which maximizes the sum of each connected trip cluster. In another word, it does not consider the time constraints and the accumulated effects among connection stops, thus it would never include the path like $C_4 \to C_7$ of Fig. 1 in the planned bus route, while our approach might include the path as long as the route expects the maximum number of accumulated passengers and the total travel time constraint can be met.

The second category is about the bus network design, which is an intensive studied area in urban planning and transportation field. The bus network design is known to be a complex, non-linear, non-convex, multi-objective NP-hard problem [10], [9]. The aim is to determine bus routes and operation frequencies that achieve certain objectives, subject to the constraints and passenger flows. The popular objectives include shortest route, shortest travel time, lowest operation cost, maximum passenger flow, maximum area coverage and maximum service quality while the constraints include time, capacity and resources. However, the selection of the objectives should take care of the operator as well as user requirements which are often conflicting, leading to design trade-off rather than an optimal solution. As noted in [5], early bus network design is mainly based on human survey to get passenger flows and user demands, it relies heavily on heuristics and intuitive principles developed by a designer's own experience and practice. Recent work on bus network design also assumes that the passenger flows are given by user survey or population estimation, the best solving algorithms are based on heuristic procedures to find sub-optimal solutions. A detailed review about route network design can be found in [7].

Despite the renewed attention for bus network design, there is still no work addressing the night-bus route design problem leveraging the taxi passenger OD flow data. Different from existing research, our work aims to find a bus route with a fixed frequency, maximizing the number of passengers expected along the route subject to the total travel time constraint. This problem is different from the traditional Travelling Salesman Problem (TSP) [1] in nature, which aims to find the shortest path that visits each given location (node) exactly once. TSP evaluates different routes with exact N locations, which means all candidate stops should be included in the route. Our problem is also different from the shortest path finding problem [12], which intends to get the shortest path for a given OD pair. In our case, we have to consider the accumulated effect (passenger flows) from all previous stops to current stop for choosing the bus route.

III. Candidate Bus Stop Identification

In the proposed two-phase bus route planning framework, the objective of phase one is to identify candidate bus stops by exploiting the taxi *PDRs*. The whole process consists of three steps: (1) Divide the whole city into small equal-sized grid cells, mark those "hot" grid cells with high taxi *PDRs* for further processing; (2) Merge the adjacent "hot" grid cells to form "hot" areas, divide each big area into "walkable size" cluster; (3) Choose one grid cell as the candidate bus stop location in each walkable size "hot" cluster, by assuming that passengers from the same cluster would easily walk to the stop to take the bus.

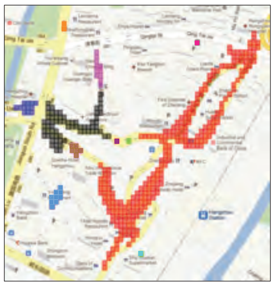

Fig. 3. City partitions near Hangzhou Railway Station. Each city partition is marked with a different color.

A. Hot Grid Cells and City Partitions

In this work, we first divide the city into equal-sized grid cells, with each cell about $10m \times 10m$. In such a way, the whole city is partitioned into 5000×2500 cells in total. Out of all the grid cells, over 95% of them contain no taxi passenger *PDRs* as they are either lakes, mountains, buildings, and highways that cannot be reached or stopped by taxis, or suburb areas that people seldom travel to. Only 0.11% of them have more than 0.2 *PDRs* per hour on average if we only count the *PDRs* in late night. And we name these grid cells as "hot" ones.

As each grid cell has maximum eight neighbors, if we define the connectivity degree (*CD*) of a "hot" grid cell as the number of "hot" neighboring cells, the *CD* of any grid cell will range from 0 to 8, where the "hot" grid cell with *CD* equal to 0 is called isolated cell. As the city is composed of mixed hot grid cells and common grid cells, both hot cells and common cells form irregular "hot areas" and "common areas" as a consequence of same type of cells being adjacent to each other. These "hot areas" are also called city partitions, as shown in Fig.3. Apparently, some small partitions (like the small ones in Fig. 3) can be very close to some big ones (like the black and red ones in Fig. 3). It would be necessary to consider all the city partitions globally in order to plan the bus stop locations, thus city partitions close to each other had better merge to form big clusters for better overall bus stop distribution. In the next section, we propose a simple strategy to merge the close partitions into bigger clusters.

B. Cluster Merging and Splitting

We present the cluster merging and splitting approach in Algorithm 1. After obtaining all city partitions, we sort them in a descending order according to the number of *PDRs* (Line1). To merge the partitions close to each other iteratively, we propose to use the hottest partition to *absorb* its nearby partitions according to the descending order of *PDRs*, until no more nearby partitions meet the merging criteria (Line8). Then we choose the next hottest partition to repeat the same

Algorithm 1 Merge Algorithm

```
Input: List of partitions $\{P_i\}$
Output: List of clusters $\{C_i\}$
 1: $P \leftarrow sort(P)$, $(i = 1, 2, \cdots, n)$ // Sort $P$ according to amount of its PDRs by
    descending order
 2: $i = 1$;// Initialization
 3: while $P \neq \emptyset$ do
 4:     $C_i = \{P_1\}$;
 5:     $P = P \backslash \{P_1\}$ // Remove $P_1$ from $P$
 6:     $k = |P|$ ;
 7:     for $j := 1$ to $k$ do
 8:         if $dist(C_i, P_j) < th$ (we set $th$ to 150 $m$) then
 9:             $C_i = C_i \cup P_j$ //absorb the closer partition
10:             $P = P \backslash \{P_j\}$ //Remove $P_j$ from $P$
11:         end if
12:     end for
13:     $i = i + 1$;
14: end while
```

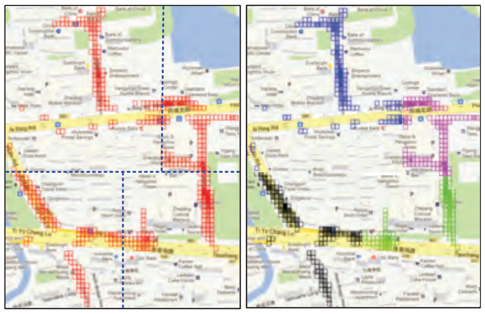

Fig. 4. Illustrative example of splitting. Big cluster formed via merging (left). Big cluster split into 4 walkable size clusters (right, in four different colors).

process until all the partitions are checked (Line8$\sim$Line12). The location of each partition is first initialized by computing the weighted average location of all grid cells using the Eq. 1.

$$loc(P) = \frac{\sum_{i=1}^{N}(PDRs(g_i) * loc(g_i))}{\sum_{i=1}^{N} PDRs(g_i)} \tag{1}$$

where $loc(g_i)$ refers to the longitude/latitude of the member grid cell g_i.

After merging one partition, the location of the combined cluster is updated (Line9) and the absorbed partition is removed from the partition list (Line 10). The *dist* function refers to the distance between two given partitions. The algorithm will be terminated until no partitions can be merged to a new cluster (Line3).

However, some merged clusters may be too large, which can set up more than one bus stop. Thus proper *splitting* should be done for these big clusters. In general, the merged clusters can be classified into three groups according to their size (the size of cluster is defined as minimal rectangle which covers all the grid cells): 1) with both height and width are greater than $500m$; 2) with either height or width is greater than $500m$; and 3) with both height and width are less than $500m$. We find that only a very small number of clusters (around 10%, group 1 and 2) need further *splitting* operations.

As for large clusters (group 1 and 2), we adopt a simple strategy to split them. Specifically, for clusters in group 1, we split the big cluster into a number of sub-clusters, aiming to minimize the difference of *PDRs* of the resulted clusters both in horizonal and vertical directions, while for clusters in group 2, we only need to split the cluster in one direction. Fig. 4 shows an illustrative example of splitting a cluster into four sub-clusters with the proposed splitting strategy. The initial cluster belongs to group 1 (Fig. 4 (left)), the splitting is first done in horizontal direction to produce two sub-clusters with similar *PDRs*. After the first splitting, two sub-clusters with width greater than $500m$ are generated, thus both sub-clusters require a further splitting in vertical direction. The final result with four split sub-clusters is shown in Fig. 4 (right).

C. *Candidate Bus Stop Location Selection*

After *merging* and *splitting* operations, we obtain a big number of "hot" clusters with the size smaller than $500m \times 500m$. The next step is to select a *representative* grid cell in each cluster to serve as the candidate bus.

$$\arg\max \left[w_1 \times \frac{CD(i)+1}{9} + w_2 \times \frac{PDRs(i)}{\sum_{i=1}^{n}(PDRs(i))} \right] \tag{2}$$

To select this *representative* grid cell, both connectivity degree (*CD*) and the number of *PDRs* of each cell in the cluster are taken into consideration. While the *CD* of a grid cell characterizes the accessibility of the cell, the number of *PDRs* is an indicator of its "hotness". The grid cell having the maximum value defined in the Eq. 2 in each cluster is selected as the "center" of the cluster, marked as the location of the candidate bus stop. We set $w_1 = w_2 = 0.5$ in the evaluation, and totally we get 579 candidate bus stops in the city.

IV. Bus Route Selection

After fixing the candidate bus stops in phase one, the aim of phase two is to find the best bus route for a given OD, expecting to maximize the expected number of passengers under the time constraints.

Here, we first approximate the passenger flow and travel time between any two candidate stops using taxi GPS traces, then we present the bus route selection method which contains the following three-steps: 1) Build the bus routing graph and remove invalid nodes and edges iteratively based on certain criteria; 2) Automatically generate candidate bus routes with two proposed heuristic algorithms; 3) Select the bus route by comparing the expected number of passengers under the same total travel time constraint.

A. *Passenger Flow and Travel Time Estimation*

We record the travel demand and time information in two matrix, named passenger flow matrix (*FM*) and bus travel time matrix (*TM*). Each element in the matrix refers to the number of passengers and bus travel time from one stop (ith) to another stop (jth, $i \neq j$). We count the total taxi trips from ith cluster to jth cluster as each stop is responsible for its cluster. We set the maximum waiting time for passengers at the stop is 30 minutes, so any pick-up or drop-off events taking place in this time window are counted. We simply assume the passenger flows among candidate bus stops remain unchanged during each 30-minutes duration. The final *FM* is got by averaging all flow matrix at different bus frequencies. We also assume *TM* keeps unchanged across the night time. $tm(s_i, s_j)$ is the average travel time multiply by α, which is a constant. We set $\alpha = 1.5$ to consider the speed difference between taxis and bus. For the paths having no taxi trip occurring in history (for instance, nobody travels by taxi due to too short distance), we use *Ddist*$(s_i,s_j)/v$ to approximate $tm(s_i, s_j)$, where *Ddist*(s_i, s_j) is the driving distance between s_i and s_j, and v is set to 50 km/h.

B. Bus Routing Graph Building and Pruning

Selecting the best bus route is a very challenging problem as two conflicting requirements must be met: one is to ensure that the bus route would traverse intermediate stops and finally reach the destination within a limited time; the other is to maximize the number of passengers accumulated along the route. For example, if we choose the stop with the heaviest passenger flow from the origin as the first node, and then keep choosing the next stop following the heaviest passenger flow principle, then we might neither reach the destination, nor achieve the objective of having the maximum number of passengers accumulated along the route. To meet the above two requirements and follow the intuitive principles in bus route design, some basic criterion should be set for bus routing graph building and candidate bus route selection.

1) Routing graph building criteria: Obviously, there would be numerous stop combinations for a given OD pair, and only a small proportion of them meet the first or second requirement. To reduce the search space of possible stops and routes, we can build a bus routing graph starting from origin to destination using heuristic rules. For instance, from the shortest travel time perspective, the bus route should extend from origin towards the direction of destination, it can be further converted into three rules: each new selected stop should be farther from the origin, closer to destination, and farther from previous stop. From the intuitive bus route design principle, the bus stops should not be too far from each other, also the bus route should not comprise sharp zig-zag paths. These can also be translated into two criterion in building the bus routing graph. Specifically, given the OD pair (s_1, s_n) and the candidate route $R = \langle s_1, s_2, \cdots, s_n \rangle$, we should obey the following criterion when building the bus routing graph.

- *Criteria 1: Adequate stop distance*

$$dist(s_{i+1}, s_i) < \delta \quad (i = 1, 2, \cdots, n-1)$$

where δ is a user-specified parameter. We set $\delta = 1.5\ km$, which means the maximum distance between two consecutive stops should be no more than 1.5 km.

- *Criteria 2: Move forward*

$$x_{new}(i+1) > x_{new}(i) \quad (i = 1, 2, \cdots, n-1)$$
$$x_{new}(i) = x(i) \cos\theta + y(i) \sin\theta$$
$$\theta = \tan^{-1} \frac{y(n)}{x(n)}$$

$(x(i), y(i))$ of the s_i is got by simply subtracting the longitude and latitude value to that of s_1. x_{new} is the value of X-axis of stop in the new coordination which is with s_1 as the origin, and from s_1 to s_n as the direction of X-axis (see the left panel in Fig. 5). This criteria guarantees the bus will always move forward along the OD direction.

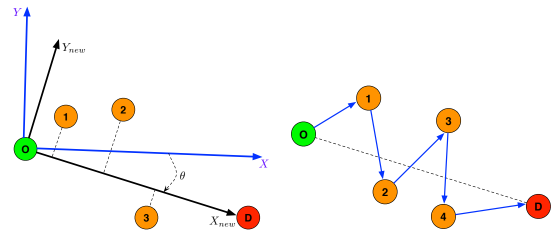

Fig. 5. Demonstration of Criteria 2 (left) and Criteria 5 (right).

- *Criteria 3: Source-farther*

$$dist(s_{i+1}, s_1) > dist(s_i, s_1) \quad (i = 1, 2, \cdots, n-1)$$

This ensures that the bus will move away from the origin s_1 farther in each step.

- *Criteria 4: Destination-closer*

$$dist(s_{i+1}, s_n) < dist(s_i, s_n) \quad (i = 1, 2, \cdots, n-1)$$

This ensures the bus will move closer to the destination s_n in each step.

- *Criteria 5: No zigzag route*

$$\underbrace{\arg\min}_{s_j}(dist(s_{i+1}, s_j)) = s_i \quad (j = 1, 2, \cdots, i)$$

Criteria 5 ensures the smoothness of the route. There would be no sharp zigzag path along the OD direction. The route demonstrated in the right panel of Fig. 5 should not happen, as it violates the *no zigzag route* criteria. We can see $\arg\min(dist(s_3, s_j)) = s_1 \neq s_2 \quad (j = 1, 2)$, also $\arg\min(dist(s_4, s_j)) = s_2 \neq s_3 \quad (j = 1, 2, 3)$.

2) Graph building and pruning: The aim of graph building is to construct a directed graph with nodes and links given an OD pair, in which the nodes are the stops, and edges link the stop to its next possible stops, regardless of passenger flows among them. While the goal of graph pruning is to remove invalid edges and nodes according to the proposed criterion.

Graph Building: Given the bus route OD, their locations are firstly used to narrow down the choice of valid candidate stops, only the candidate stops lying between them are under consideration. For each stop within the range, we determine links to its next possible stops according to the proposed *Criteria 1*$\sim$*4*. As *Criteria 5* is related to all stops in one bus route, so we use it to prune the routing graph after it is built. Fig. 6 (left) shows an illustrated example about a generated bus routing directed graph.

Graph Pruning: Some nodes and edges can be further pruned because they are not useful for valid candidate bus route selection. To be specific, for nodes without in-coming edges (if not origin) or out-going edges (if not destination),

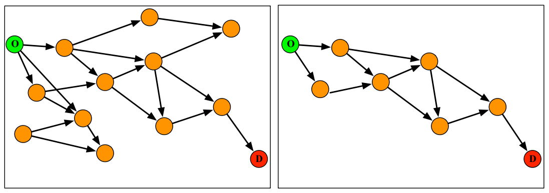

Fig. 6. A bus routing directed graph for a given OD.

as they will not form any valid routes with the bus route OD pair, these edges and nodes should be deleted.

We first calculate all the nodes' in-coming and out-going degrees. And nodes (exclude the given OD) and corresponding edges would be iteratively deleted from graph if their in-coming or out-going degree is zero. At last graph with only one zero in-coming degree (given origin) and one zero out-going degree (given destination) would be generated. After graph pruning, all the bus routes starting from the source and following the edges in the graph would eventually reach the destination. Fig. 6 (right) displays the resulted graph after applying pruning to the graph in Fig. 6 (left).

C. Automatic Candidate Bus Route Generation

Probability based Spreading Algorithm: Though we have removed invalid nodes and edges through graph pruning, the problem of enumerating all possible routes from given source to destination is proved to be NP hard. Indeed, it is also unnecessary to enumerate all possible routes and compare them all, because most of routes are *dominated* by few others.

DEFINITION 1. We say R_i **dominates** R_j iif: 1) $T(R_i) \leq T(R_j)$; 2) $Num(R_i) > Num(R_j)$.
where T and Num are the total travel time and number of expected delivered passengers, which is computed based on Eq. 3 and 4.

$$T = \sum_{i=1}^{n-1} tm(s_{i+1}, s_i) + (n-2) \times t_0 \tag{3}$$

$$Num = \sum_{i;j(j>i)}^{n} fm(s_i, s_j) \tag{4}$$

where t_0 is the bus waiting time at each stop, and we set it to 1.5 minutes. The definition is similar to the *skyline routes* in [6], and the rational behind is that only routes with less travel time but larger number of passengers should be selected. *Skyline* detector [18] will prune the routes which are dominated by skyline routes in the candidate set. Thus, the comparison can be done among detected *skyline routes*.

The key idea of our proposed *probability based spreading algorithm* is to randomly select the next stop among the possible candidate stops in each step, where the candidate stops having high accumulated passenger flow with previous stops are given high probability for random selection. We describe the approach in Algorithm 2. The spreading starts from the given source (Line3). The next stop in the candidate route is chosen based on Eq. 5.

Algorithm 2 Probability based Spreading

```
Input: $G(S, E)$; *flow* Matrix
Output: $\mathcal{R}^*$
1: $\mathcal{R} = \emptyset$
2: Repeat
3:   currentR = $s_1$
4:   Choose the next stop $s_i^*$ with respect to currentR according to Eq. 5
5:   $R$ = currentR$\cdot s_i^*$
     //$\cdot$ operation appends $s_i$ to currentR
6:   Repeat Lines 4~5 Until $s_i^* = s_n$
7:   $\mathcal{R} = \mathcal{R} \cup R$
8:   Get corresponding skyline routes $\mathcal{R}^*$
9: Until $\mathcal{R}^*$ keeps unchanged
```

$$P(s_i^* | \langle s_1, s_2, \cdots, s_j \rangle) = \frac{\sum_{m=1}^{j} fm(s_m, s_i^*)}{\sum_{i=1}^{|S^*|} \sum_{m=1}^{j} fm(s_m, s_i^*)} \tag{5}$$

where $fm(s_m, s_i^*)$ is the passenger flow from s_m to s_i^*, and S^* contains the next possible stops of s_j (child nodes of s_j in routing graph).

We can see the selection of next stop in the candidate route is not only determined by the current stop, but also all the previous stops. The output of this algorithm is one candidate bus route with the number of stops associated with the number of spreading steps. The spreading would be terminated when the given destination is reached (Line6). For each run, we get either a repeated route or new route, thus the candidate route set $\mathcal{R}$ would increase as the spreading algorithm is activated. Then a question arises: *how many times are sufficient to run the spreading algorithm and get the best results ?* Based on *Definition* 1 about the skyline routes, we should consider if the skyline route set $\mathcal{R}^*$ remains changed or unchanged.

Theorem 1 below ensures that when the *skyline route* set stays unchanged with the increase of spreading algorithm runs, then the best route has been discovered.

Theorem 1. *$\mathcal{R}_1^*$ and $\mathcal{R}_2^*$ are the detected skyline routes from $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively. If $\mathcal{R}_1 \subseteq \mathcal{R}_2$, then we have: $\forall R_i \in \mathcal{R}_1^*$, $\exists R_j \in \mathcal{R}_2^*$; $R_i = R_j$ or R_i is dominated by R_j.*

In Algorithm 2, we have $\mathcal{R}_{t_1} \subseteq \mathcal{R}_{t_2}$ if the running time $t_1 < t_2$, and the algorithm would be stopped when no better skyline routes are returned with the increase of running times, that is $\mathcal{R}_{t_1}^* = \mathcal{R}_{t_2}^*$ (Line9). Instead of choosing only one stop randomly at each spreading step like in the *probability based spreading algorithm*, an intuitive way is to select top-k stops each time, where those k nodes should have highest accumulated passenger flow with previous stops; In such a way, the first step selects top-k nodes, thus leading to k routes from the origin to those nodes; In the second step, each k nodes would select another top-k nodes, thus the total candidate routes would be k^2; Assume reaching the destination needs n step spreading, then the total candidate routes generated would be k^n in the end. We use this **top-k spreading** method as the baseline.

D. Bus Route Selection

Given the bus operation frequency (once every 30 minutes) and the taxi passenger flow from 21:30 to 5:30, we obtain the

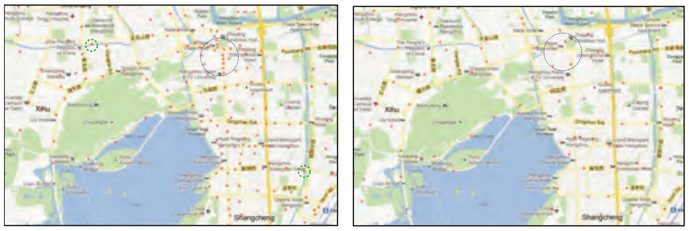
Fig. 7. Comparison results with *k*-means. Results got by *k*-means (left) and results got by our method (right).

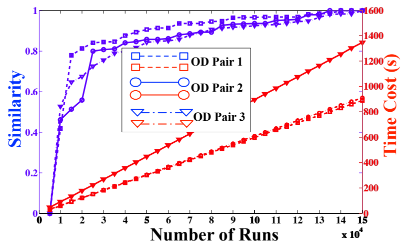

Fig. 8. Convergence study of the proposed spreading algorithm.

TABLE I
DETAILED INFORMATION ABOUT STUDIED OD PAIRS

	OD Pairs	Distance (*km*)	Number of Stops
1	ZJU - Railway	5.70	104
2	Railway - East Railway	5.86	75
3	East Railway - ZJU	8.80	144

candidate bus routes for a given OD pair and maximum travel time using the two different heuristic spreading algorithms, the skyline route which achieves the maximum expected number of passengers will be selected as the operating route. With the planned bus route consisting of the selected bus stops, the next step is to find a physical bus route in the real setting, which consists of road segments. The selection of each road segment is done by following the dense and fine trajectories of taxis if they allow buses to operate; Otherwise similar bus routes near the planned ones can be adopted as a refined solution.

V. EXPERIMENTAL EVALUATION

Here, we validate the proposed approach with a large-scale real-world taxi GPS dataset which was generated from 7,600 taxis in a large city in China (Hangzhou) for one month, with more than 1.57 million of night passenger-delivering trips.

A. *The Identification of Candidate Bus Stops*

We compare the bus stop results generated with our method with that generated by the popular *k*-means clustering method. We set $k = 579$, which is the same as our method. We adopt the Eulerian distance as the similarity metric. The centroid of each cluster is selected as the stop. Fig. 7 shows the comparison results. Comparing with the popular *k*-means approach, our proposed candidate bus stop identification method has two advantages: 1) the centroid of each cluster got by *k*-means is the average location of all its members, and it may fall into non-reachable places like river, as highlighted by the green circle in Fig. 7 (left). In our proposed method, both hotness and connectivity of each grid cell is considered for bus stop location selection, and the selected bus stops are meaningful and stoppable places; 2) Several identified stops by *k*-means fall into a small area (highlighted by the blue circles) as the size of clusters got by *k*-means is very different, while our proposed method generates candidate bus stops that are evenly distributed in the hot areas, which meets better the commonsense design criteria of bus stops.

B. *Evaluation of the Probability based Spreading Algorithm*

We evaluate our probability based spreading approach. We first show the convergence of the proposed algorithm. Then we perform a quantitative statistical analysis of all the candidate routes generated for three given OD pairs. We will also give the computed *skyline route* results. Finally, we validate that our proposed bus route generation approach outperforms the baseline approach. Table I shows the details of three OD pairs for night-bus route design experiment, where more than 70 candidate bus stops are in the candidate bus route selection list.

1) Convergence Study: As illustrated in Algorithm 2, our proposed bus route generation process would be terminated if the resulted *skyline routes* keep unchanged. We study the similarity of consecutively generated *skyline routes* from 5,000 to 150,000 runs, with a constant interval of 5,000 runs. We measure the similarity (sim) of two sets A and B as follows:

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{6}$$

The similarity results of the consecutively generated skyline routes with a 5,000 run interval are shown in Fig. 8, the time cost is put in the diagram as well. In this study, we can see sim values gradually reach 1 with the increase of runs for all three OD pairs, meaning that in all three cases the best bus route converges to one. Also the time cost is almost linearly increased with the number of runs, suggesting that the spreading time cost at each run is almost constant. It is also noted that the three curves for three OD pairs have different slopes, the reason is probably because the bus routes corresponding to different ODs have different lengths and varied number of candidate bus stops, thus the spreading time and candidate bus stop selection time should be also different.

2) Candidate Routes Statistics: Fig. 9 shows the statistical information about the number of stops of candidate routes. Several interesting observations can be made:

- For OD pair 1, routes with 8∼10 stops take up over 80% of the cases (both origin and destination are included). Few routes can reach the destination by traversing only 4 stops, or passing more than 11 stops.
- For OD pair 2, over 60% of the routes contain 9 or 10 stops. Similar to the case of OD pair 1, some routes can reach the destination by passing 4 stops.
- For OD pair 3, most of the routes contain 10 to 18 stops due to the longer OD distance, and almost half of the

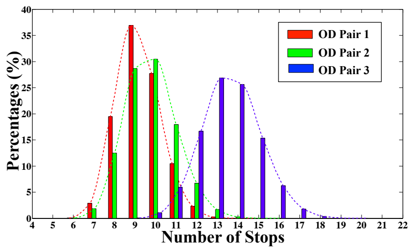

Fig. 9. Statistical information about candidate route stops for 3 OD pairs.

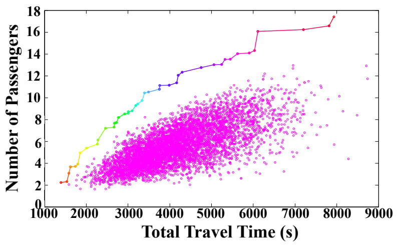

Fig. 10. Detected *skyline routes* and other candidate routes.

routes include 13 or 14 stops.

- The statistical results comply with the intuition that the longer distance of a given OD pair, the more stops the route would contain.

3) Skyline Routes: We show the *skyline routes* for the OD pair 2 in Fig. 10. Each point in the plane represents a candidate route. The x-axis stands for the total travel time of candidate route, while the y-axis represents the expected number of passengers. From Fig. 10, we can see that the *skyline routes* are connected to form a curve above all the points representing common routes, and over 99% of the routes are dominated by the few skyline routes. Specifically, we get 40 skyline routes across all the travel time frames, out of hundreds of thousands of routes for the case of OD pair 2. Similar phenomena have been observed for other two cases as well.

4) Comparison with top-k spreading algorithm: In top-k spreading algorithm, the selection of k is vital to the skyline routes generated as well as the time needed to generate all the candidate routes. In particular, when $k_1 < k_2$, we have $\mathcal{R}_{k_1} \subseteq \mathcal{R}_{k_2}(k_1 \leq k_2)$. Theorem 1 guarantees that a bigger k would lead to a better set of skyline routes. However, the greater k also results in significantly increase of time cost. We compare the *skyline routes* generated from the *probability based spreading* method with that from the *top-k spreading* method with different k value, which is shown in Fig. 11. We can see that the *probability based spreading* approach outperforms the *top-k algorithms* even when k is set to 5.

C. Comparison with Real Routes

As the taxi GPS dataset we have was collected from April 2009 to March 2010, we are very interested in knowing if there was any new night-bus route created during this year and how the planned bus route generated with our approach compares with the manually created route. Fortunately we were told that a night-bus route was created in February 2010, we could

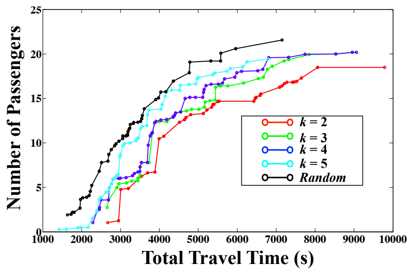

Fig. 11. Comparison results with baseline under different k values.

access all the taxi passenger flows before the route started date. It is noted that the route is designed by local experts and the user demands are obtained from expensive human survey. We first draw the newly started night-bus route on Google map as shown in Fig. 12 (left bottom), then we draw our proposed night-bus route R_1 in Fig. 12 (left top). Through comparison we see that they are quite different. With the newly started route, we decide to take a similar route in our selected candidate bus routes (not the best one), and we find R_2 as shown in Fig. 12 (left top). It is noted that the main difference between R_2 and the newly started route is that R_2 includes an additional Stop J in the route. By comparing the passenger flow in segment I$\rightarrow$K with that in segment J$\rightarrow$K at different time slots, it is found that the passenger flow in path J$\rightarrow$K is even greater than I$\rightarrow$K in the first two time slots, as shown in Fig. 12 (right top). Considering further the accumulation effects, including Stop J in the bus route would significantly increase the expected number of passengers along the route. Thus, our candidate bus route R_2 would outperform the newly added bus route, at the cost of adding one more bus stop and more travel time.

We also compare our proposed best route R_1 with the candidate route R_2. The difference between R_1 and R_2 lies in two different paths taken from C to H. While R_2 passes the famous shopping street (Yan'an Road) in Hangzhou ($C \rightarrow E \rightarrow F \rightarrow H$), R_1 traverses the famous night-club areas along the West Lake. If we compare the number of passengers in R_1 and R_2, it can be seen from Fig. 12 (right bottom) that the passenger flow of R_2 is heavier than that of R_1 only around 22:00, and it is much lighter soon after 23:00. With the rest of the stops being the same for both R_1 and R_2, there is no doubt about why R_1 has been selected as the best night-bus route. If we take a closer look at R_1, R_2, and the newly started route, as R_1 takes a much shorter route than R_2 and needs similar travel time as the newly started route does, but R_1 expects much more passengers than R_2 and the newly started route, thus it is reasonable to conclude that the selected night-bus route with our proposed approach is better than the current route-in-service in terms of travel time as well as expected number of passengers.

VI. Conclusion and Future Work

In this paper, we have investigated the problem of night-bus route design by leveraging the taxi GPS traces, which

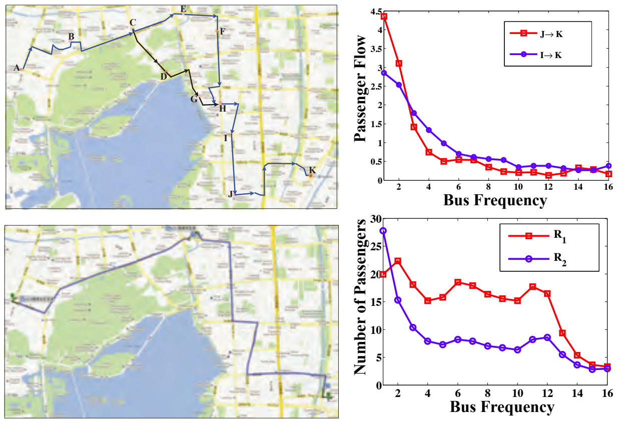

Fig. 12. Results comparison. Planned routes (top left); Passenger flow comparison of two segments at different frequency (top right); Opened night-bus routes (bottom left); Number of delivered passengers at different frequency (R_1 vs R_2, bottom right).

is motivated by the needs of applying pervasive sensing, communication and computing technology for sustainable city development. To solve the problem, we propose a two-phase approach for night-bus route planning. In the first phase, we devise method to identify locations of the candidate bus stops, with the passenger pick-ups and drop-offs as inputs. In the second phase, with the bus route origin, destination, and candidate bus stops as inputs, we develop two heuristic algorithms to automatically generate candidate bus routes, and finally we select the best route which expects the maximum number of passengers under the given conditions. On a real-world dataset which contains more than 1.57 million passenger delivery trips, we compare our proposed candidate bus stop identification method with the popular k-means clustering method and show that our method can generate more reasonable and meaningful results. We further extensively evaluate our proposed *probability based spreading algorithm* for automatic bus route generation and validate its effectiveness as well as its superior performance over the heuristic top-k spreading algorithm. Further more, we show the selected night-bus route with our proposed approach is better than a newly started night-bus route-in-service in Hangzhou, China.

This work is among our initial attempts to apply pervasive computing techniques in achieving better urban planning of smart cities. With the constraints of the approach and the acquired data, this work still has several limitations: 1) Human mobility patterns uncovered by taxi GPS traces are biased and incomplete, thus the designed bus route might not be the best, considering the fact that taxi traffic does not reflect the whole passenger flow and people taking taxis might not be willing to switch to public transport for various reasons; 2) The work makes some assumptions in problem formulation and evaluation, for example, for bus stop determination we set the walkable distance as 500 meters and the threshold for cluster split and merge is 150 meters, apparently the choice of those parameters would affect the results, but a study of the parameter sensitivity is not provided due to the space limitation; 3) Currently we only explore the issue of designing the best bus route for a given OD pair, the proposed approach is still unable to tackle the design of a bus route network.

In the future, we plan to broaden and deepen this work in several directions. First, we attempt to investigate the optimal bus route design with more real-life assumptions such as varied bus operation frequency and limited bus capacity; Second, we plan to study the impact of parameter change on the bus route design and the added bus route on the taxi services along the bus route; Third, we would like to develop practical systems leveraging on taxi GPS traces, enabling a series of pervasive smart transportation services.

ACKNOWLEDGEMENTS

We would like to thank the shepherd, Dr. Petteri Nurmi, and the reviewers for their constructive suggestions. This research was partially supported by the Paris-Region SYSTEM@TIC Smart City "AQUEDUC" Program, NSF of China (No. 61105043), and JiangsuSF (BK2011566).

REFERENCES

[1] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2007.
[2] F. Bastani, Y. Huang, X. Xie, and J. W. Powell. A greener transportation mode: flexible routes discovery from GPS trajectory data. In *Proc. of GIS*, pages 405–408, 2011.
[3] L. Cao and J. Krumm. From GPS traces to a routable road map. In *Proc. of the ACM GIS*, pages 3–12, 2009.
[4] P. S. Castro, D. Zhang, and S. Li. Urban traffic modelling and prediction using large scale taxi GPS traces. In *Proc. of Pervasive Computing*, pages 57–72, 2012.
[5] T. A. Chua. The planning of urban bus routes and frequencies: A survey. *Transportation*, 12(2):147–172, 1984.
[6] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In *Proc. of ACM SIGKDD*, pages 899–908, 2010.
[7] V. Guihaire and J.-K. Hao. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42(10):1251 – 1273, 2008.
[8] L. Liu, C. Andris, and C. Ratti. Uncovering cabdrivers' behavior patterns from their digital traces. *Computers, Environment and Urban Systems*, 34(6):541 – 548, 2010.
[9] B. M.H. and M. H.S. TRUST: A Lisp program for the analysis of transit route configurations. *Transportation Research Record*, 1283:125–135, 1990.
[10] G. F. Newell. Some issues relating to the optimal design of bus routes. *Transportation Science*, 13(1):20–35, 1979.
[11] G. Pan, G. Qi, Z. Wu, D. Zhang, and S. Li. Land-use classification using taxi GPS traces. *IEEE Trans. on ITS*, (99):1–11, 2012.
[12] Z. Wang and J. Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *SIGCOMM Comput. Commun. Rev.*, 22(2):63–71, 1992.
[13] J. Yuan, Y. Zheng, and X. Xie. Discovering regions of different functions in a city using human mobility and POIs. In *Proc. of ACM SIGKDD*, pages 186–194, 2012.
[14] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proc. of the GIS*, pages 99–108, 2010.
[15] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun. Where to find my next passenger. In *Proc. of UbiComp*, pages 109–118, 2011.
[16] D. Zhang, B. Guo, and Z. Yu. The emergence of social and community intelligence. *Computer*, 44(7):21 –28, 2011.
[17] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li. iBAT: detecting anomalous taxi trajectories from GPS traces. In *Proc. of UbiComp*, pages 99–108, 2011.
[18] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *Proc. of UbiComp*, pages 89–98, 2011.