T2CBS: Mining Taxi Trajectories for Customized Bus Systems

Yan Lyu*, Chi-Yin Chow*, Victor C. S. Lee*, Yanhua Li† and Jia Zeng‡
*Department of Computer Science, City University of Hong Kong, Hong Kong
†Department of Computer Science, Worcester Polytechnic Institute, MA, USA
‡Huawei Noah's Ark Lab, Hong Kong
*{yanlv2-c@my., chiychow@, csvlee@}cityu.edu.hk, †yli15@wpi.edu, ‡zeng.jia@huawei.com

***Abstract*—A customized bus (CB) system is a new emerging public transportation that provides flexible demand-oriented transit services for city commuters. Existing CB systems encounter two challenges of 1) collecting travel demands and discovering travel patterns effectively and efficiently and 2) planning profitable bus lines based on travel patterns. In this paper, we propose a bus line planning framework, called T2CBS, by taking full advantage of taxi trajectory data. In T2CBS, similar travel demands are discovered from passenger trajectories with a clustering algorithm, and CB stops are deployed at pick-up and drop-off points of trajectory clusters with integer linear programming. To plan profitable CB lines, we propose a profit estimation model, by considering the number of taxi passengers who can be attracted to CB buses. A routing algorithm (CBRouting) and a timetabling algorithm (CBTimetabling) are proposed to generate a CB line that can achieve the maximum profit for each trajectory cluster. We conduct experiments on one-month taxi trajectory data in Nanjing, China. Experimental results demonstrate that our T2CBS can generate CB lines with higher profit compared with baseline methods, and the moderate increase in travel time along the CB lines is significantly dominated by the savings in bus fare.**

I. Introduction

In recent years, a new innovative public transport mode, called Customized Bus (CB) systems [10], has been springing up across China. A CB system provides flexible demand-oriented transport service for city commuters. It collects travel demands by online surveys, aggregates similar travel demands, and publishes candidate bus lines for users to reserve seats or purchase tickets. With the advantages of congestion alleviation, environmental friendliness as well as better travel experience, the CB system enjoys high popularity in more and more cities of China [10].

CB systems significantly differ from traditional bus systems because they aim to serve groups of passengers with similar travel demands, by providing direct and efficient transit services. A typical CB line, as illustrated in Fig. 1, has the following features: 1) multiple bus stops are deployed in the origin and destination areas, in order to provide CB services within a short walk distance; 2) no or very few intermediate stops are deployed between its origin area and destination area; 3) its timetable matches the demands of target passengers. On the contrary, traditional bus systems aim to serve the majority of population. A traditional bus line usually has a lot of intermediate stops, and its timetable is also designed to satisfy the majority's travel demands, in order to serve as many people as possible. Therefore, the existing bus line planning models [4], [7] are not suitable for planning CB lines.

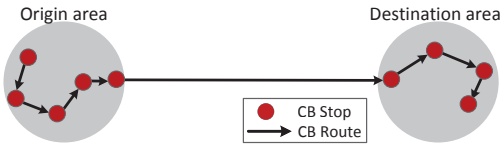


Fig. 1. Illustration of a typical customized bus line.

Existing CB systems conduct online surveys to collect travel demands of potential passengers, aggregate similar demands, and generate CB lines by transit operators. This strategy suffers the following problems: (1) The travel data collected by online surveys are just the travel intention of a limited number of potential commuters who participated in the surveys, so the collected data may not be reliable or reflect the reality [8]. (2) It is hard to detect changes of travel patterns timely by conducting online surveys. (3) It is very tedious, inefficient and costly to plan CB lines by analyzing the survey data manually [10]. Therefore, it is a timely topic to study how to discover similar travel demands and plan bus lines for CB systems in a more reliable and cost-efficient manner.

In this paper, we are motivated to propose a framework, **T2CBS**, to design CB lines by analyzing the trajectories of taxi passengers. Taxi trajectories record the detailed travel information of passengers, such as departure time and location, arrival time and location. Such comprehensive travel data of the large number of real passengers reveal the actual travel demands citywide [3], [12], [13], [14], so they are more reliable than the survey data. Also, it is possible for T2CBS to detect any significant change of travel pattern by using the latest travel data regularly, thus allowing necessary adjustment to existing CB services.

T2CBS mines taxi trajectory data to identify taxi passengers' trajectories as an indicator of travel demands for the CB system, discovers travel patterns from the passenger trajectories, and plans profitable CB lines based on the discovered travel patterns. Specifically, we first extend the density based clustering algorithm DBSCAN to cluster passenger trajectories with similar origins, destinations and departure times to discover travel patterns shared by a large number of commuters. And then, CB stops are deployed at the pick-up and drop-off points

of trajectory clusters through integer linear programming, with an objective of minimizing the number of stops that can cover a certain percentage of pick-up or drop-off points within an acceptable walk distance. To plan a CB line for each trajectory cluster, we first propose a profit estimation model, which takes walk distance and waiting time into consideration in determining the number of taxi passengers that can be attracted to the CB line. With this model, a CB line with the maximum profit is generated with a routing algorithm (CBRouting) and a timetabling algorithm (CBTimetabling).

The rest of this paper is organized as follows. Section II defines the problem. Section III and IV provides the detailed methodology of T2CBS. Section V presents evaluation results. Finally, we conclude this paper in Section VI.

II. Problem Statement

In this section, we first present two key concepts: *passenger trajectory* and *CB line*, define the problem of planning CB lines, and outline the components of T2CBS.

Passenger trajectory. A GPS-equipped taxi periodically reports its location with a timestamp and working status (i.e., occupied or vacant). A *passenger trajectory* is a sub-sequence of spatial points that the taxi travels over time. Each point consists of the taxi ID, latitude, longitude, timestamp, and occupied status. The *pick-up point* and *drop-off point* are the points where a pick-up activity ("vacant to occupied") and its immediate subsequent drop-off activity ("occupied to vacant") occur, respectively.

CB line. A *CB line*, denoted by L, has two key components: (1) A CB route, denoted by R, is a sequence of bus stops, i.e., $R = (b_1, b_2, \cdots, b_r)$, where r is the number of stops and b_i is the i-th stop along the route, $1 \leq i \leq r$. (2) A CB timetable, denote by $\mathbf{t}(R)$, is a sequence of start time of each *trip*, i.e., $\mathbf{t}(R) = (t^1, t^2, \cdots, t^s)$, where a *trip* is one ride of a bus that starts at the first stop along the route R, and s is the number of bus *trips* in a day.

Problem Definition. Given a set of passenger trajectories, we aim to discover similar travel demands, find optimal locations to deploy CB stops, and design routes and timetables for CB lines, so as to maximize their profit.

T2CBS framework. Our T2CBS consists of two phases: (1) *CB stop deployment phase.* In this phase, travel patterns are discovered by clustering passenger trajectories. And CB stops are deployed to cover the pick-up and drop-off points of each cluster using integer linear programming. (2) *CB line planning phase.* A CB line with the maximum profit is generated for each trajectory cluster using a routing algorithm and a timetabling algorithm in this phase.

III. CB stop deployment

In this section, we first discover the travel patterns by clustering trajectories of taxi passengers, and then deploy CB stops at pick-up and drop-off points of trajectory clusters.

A. Clustering trajectories

CB buses aim to serve groups of passengers who have similar origins, destinations and departure times. Such travel patterns can be discovered by clustering passenger trajectories with close pick-up points, drop-off points and pick-up timestamps.

Traditional density based clustering algorithm DBSCAN [6] clusters data points based on the spatial distance between them. It cannot be used directly for clustering trajectories with the spatial and temporal features. Therefore, we extend it to an algorithm T-DBSCAN, for clustering trajectories with nearby pick-up and drop-off points and similar pick-up timestamps.

We employ a spatial radius c_d and a temporal radius c_t to define the spatio-temporal neighbors of a trajectory. Specifically, the spatio-temporal neighborhood of a trajectory T_i, denoted by $N_{c_d,c_t}(T_i)$, is defined as a set of trajectories that 1) the spatial distance from their pick-up points and drop-off points to the corresponding points of T_i are within c_d, and 2) the temporal distance between their pick-up stamps and that of T_i are within c_t, i.e.,

$$\begin{aligned} N_{c_d,c_t}(T_i) = \{T_j \in \mathcal{T} | & dist(o_{T_i}, o_{T_j}) \leq c_d, \\ & dist(d_{T_i}, d_{T_j}) \leq c_d, |t_{T_i} - t_{T_j}| \leq c_t\}, \end{aligned} \tag{1}$$

where $\mathcal{T}$ is the set of passenger trajectories, $dist(\cdot, \cdot)$ denotes the road network distance of two spatial points; o_{T_i}, d_{T_i}, and t_{T_i} denote the pick-up point, drop-off point, and pick-up timestamp of T_i, respectively.

We also employ the minimum number of neighbor trajectories $MinTrs$ to constrain the density of a trajectory cluster. If the number of neighbor trajectories is more than $MinTrs$, i.e., $|N_{c_d,c_t}(T_i)| \geq MinTrs$, the trajectory T_i is considered as a *core trajectory*. The three parameters c_d, c_t and $MinTrs$ determine the density of clusters, and we describe a heuristic method for determining the parameters in Section V-C.

As described in Algorithm 1, our T-DBSCAN takes the set of passenger trajectories $\mathcal{T}$, the parameters c_d, c_t and $MinTrs$ as inputs, and discovers a trajectory cluster with two steps. It first chooses an arbitrary trajectory from $\mathcal{T}$ satisfying the core trajectory conditions as a seed (Lines 4 to 10). Then, it retrieves all the trajectories in the neighborhood of the seed (Line 11), and expands the neighborhood by exploring each neighbor trajectory with its own neighborhood (Lines 12 to 24). The cluster containing the seed is formed until trajectories in the expanded neighborhood are all explored. We employ k-d tree [2] for indexing in T-DBSCAN.

B. Deploying CB stops

The pick-up and drop-off points of a trajectory cluster indicate the exact locations that the passengers in this cluster want to start and end their trips. Therefore, for each cluster, we deploy CB stops to cover these pick-up and drop-off points, respectively.

To provide better service quality, the CB stops should cover as many pick-up/drop-off points as possible with an acceptable walk distance. To achieve this goal, more stops should be deployed. However, too many stops will increase

Algorithm 1 T-DBSCAN for trajectory clustering

Input: (1) A set of trajectories $\mathcal{T}$ and (2) three parameters c_d, c_t and $MinTrs$.
Output: A set of clusters $\mathcal{C} = \{C_1, C_2, \cdots, C_{|\mathcal{C}|}\}$

Set c_id to be 1;
Mark all the trajectory in $\mathcal{T}$ as unvisited
for each $T \in \mathcal{T}$ **do**
  if T is unvisited **then**
    Mark T as visited
    Get the neighborhood of T, i.e., $N_{c_d,c_t}(T)$, by Eq. 1
    if $|N_{c_d,c_t}(T)| < MinTrs$ **then**
      Mark T as a non-core trajectory
    else
      Mark T as a core trajectory and put T into C_{c_id}
      Put the trajectories in the neighborhood of T into set $\mathcal{Q}$, i.e., $\mathcal{Q} \leftarrow N_{c_d,c_t}(T) - \{T\}$
      for each $T_{nb} \in \mathcal{Q}$ **do**
        if T_{nb} is unvisited **then**
          Mark T_{nb} as visited and put T_{nb} into C_{c_id}
          Get neighborhood $N_{c_d,c_t}(T_{nb})$
          if $|N_{c_d,c_t}(T_{nb})| \geq MinTrs$ **then**
            Insert $N_{c_d,c_t}(T_{nb}) - \{T_{nb}\}$ into $\mathcal{Q}$
          end if
        end if
        if T_{nb} is a non-core trajectory **then**
          Put T_{nb} into C_{c_id}
        end if
        Remove T_{nb} from $\mathcal{Q}$
      end for
      Increase c_id by 1
    end if
  end if
end for

the travel time, which will in turn damage the service quality. Therefore, we formulate the stop deployment problem as a facility location problem [9]: given a coverage radius of a stop $CovRad$, and coverage percentage $CovPct$, the problem aims to find the minimum number of stops and their optimal locations, such that these stops can cover at least $CovPct$ percentage of passengers within the radius of $CovRad$.

We employ integer linear programming (ILP) to find the best locations to deploy bus stops. The pick-up or drop-off points indicate available locations for picking up or dropping off passengers, so a bus stop should be deployed at a certain pick-up or drop-off point. Let $L = \{l_i\}, (1 \leq i \leq |L|)$ be a set pick-up or drop-off locations from a trajectory cluster. We denote $X = [X_{i,j}]$ as a neighborhood indicator, with each $X_{i,j} = 1$ or 0 representing whether or not the location l_j is within the radius $CovRad$ of l_i, $1 \leq i, j \leq |L|$. Let $y = [y_i]$ denote the deployment configuration, with each $y_i = 1$ or 0 representing a bus stop should or should not be deployed at l_i. Let $z = [z_j]$ be a 1/0 indicator, representing whether or not the l_j could be covered by a stop within the coverage radius.

We aim to find y, indicating the best locations to set up the minimum number of stops. The problem is formally formulated as below.

$$\text{minimize} \quad \sum\nolimits_{\forall l_i \in L} y_i, \tag{2}$$

$$\text{subject to} \quad \sum\nolimits_{l_j \in L} z_j \geq CovPct \cdot |L|, \qquad \forall l_i \in L, \tag{3}$$

$$z_j \leq \sum\nolimits_{l_i \in L} y_i \cdot X_{i,j}, \qquad \forall l_i, l_j \in L, \tag{4}$$

$$y_i, z_j = \{0, 1\}, \qquad \forall l_i, l_j \in L, \tag{5}$$

Eq. 2 captures the total number of deployed stops. The constraint in Eq. 3 presents that at least $CovPct$ of pick-up or drop-off points should be covered by the deployed stops within the radius of $CovRad$. The constraint in Eq. 4 guarantees the validity of the assignments, that is, if there are no bus stops in the neighborhood of location l_j, i.e., $\sum_{l_i \in L} y_i \cdot X_{i,j} = 0$, then l_j cannot be counted into the total number of covered points, i.e., $z_j = 0$. The constraints in Eq. 5 state that each y_i or z_j has to be either 0 or 1.

The optimal locations of the minimum number of stops, i.e., $y_i = 1$, can be discovered by solving the above ILP problem. We solve this problem with the commercial optimization solver, Gurobi Optimizer 5.0.

IV. CB LINE PLANNING

In this section, we first propose a profit estimation model for a CB line. And then, for each trajectory cluster with its deployed CB stops, we design a CB line with the maximum profit using a routing algorithm and a timetabling algorithm.

A. Profit Estimation Model

To estimate the profit of a CB line with route $R = (b_1, b_2, \cdots, b_r)$ and timetable $\mathbf{t}(R) = (t^1, t^2, \cdots, t^s)$, we first estimate the travel demands for the CB line, count the number of potential passengers onboard, and calculate its profit.

Travel demands estimation. Since the trajectories of taxi passengers reflect real travel demands of city dwellers [1], [5], we estimate the travel demands of CB lines from taxi trajectories by employing two parameters: walk distance $walkD$ and waiting time $waitT$. Specifically, we assume that a taxi passenger can be attracted to take a CB bus if 1) his/her pick-up point and drop-off point are within the distance of $walkD$ to the nearest stops of the line, and 2) the waiting time for a bus is less than $waitT$.

On-board passengers counting. We count the on-board passengers for each trip of the CB line based on $walkD$ and $waitT$. In the k-th bus trip ($1 \leq k \leq s$), given the start time t^k, we estimate the arrival time at each stop as $t_1^k, t_2^k, \cdots, t_r^k$, based on the speed of a CB bus, which is similar to taxis [10], and the dwell time at a stop, which is set to 1 minute. The number of passengers expected to take the CB bus at b_i, denoted as $N_i^+(t_i^{k-1}, t_i^k)$, can be estimated by summing the number of passengers who will drop off at any one of the stops after b_i, during the time t_i^{k-1} to t_i^k, (let $t_i^0 = 0$), i.e.,

$$N_i^+(t_i^{k-1}, t_i^k) = \sum\nolimits_{j=i+1}^{r} \sum\nolimits_{t=\max\{t_i^{k-1}, t_i^k - waitT\}}^{t_i^k} n(b_i, b_j, t),$$

where $n(b_i, b_j, t)$ is the number of taxi passengers that start at locations within $walkD$ from b_i, arrive at b_i at time t, and end at locations within $walkD$ from b_j. We only count the passengers arrive at b_i from $\max\{t_i^{k-1}, t_i^k - waitT\}$ to t_i^k.

Let N_i^- be the number of passengers dropped off at b_i, which can be obtained by recording destinations of on-board passengers. The number of on-board passengers at b_i, denoted as N_i, can be estimated as:

$$N_i(t_i^{k-1}, t_i^k) = N_{i-1}(t_{i-1}^{k-1}, t_{i-1}^k) - N_i^- + N_i^+(t_i^{k-1}, t_i^k), \tag{6}$$

Algorithm 2 CBRouting

Input: (1) A set of CB stops of a cluster $B = \{b_i\}$, $(1 \le i \le |B|)$, (2) $len(b_i, b_j)$, $(b_i, b_j \in B)$.
Output: Optimal CB route $R(B)$.

$Len(\{b_0\}, b_0) = 0$
for s = 1 to $|B|$ **do**
  for all subsets S of size s, $S \subseteq B$ **do**
    $S = \{S, b_0\}$, $Len(S, b_0) = \infty$
    for $b_j \in S$ **do**
      Check the validity of (S, b_j) by the precedent constraint
      if (S, b_j) is valid **then**
        Obtain the shortest route length $Len(S, b_j)$ and the route $R(S, b_j)$ by Eq. 8-9
      end if
    end for
  end for
end for
$R(B) = R(B, b_j^*)$, $b_j^* = \text{argmin}_{b_j \in B} Len(B, b_j)$

Algorithm 3 CBTimetabling

Input: A trajectory cluster, and its CB route R.
Output: Optimal timetable $\mathbf{t}^*(R)$, and the maximum daily profit $P^*(R)$.

$P(t_f) = 0$, and set optimal timetable before and at time t_f as empty, i.e., $\mathbf{t}(t_f) = \emptyset$
for $t = t_f + 1$ to t_l **do**
  Calculate $P(t) = \max_{1 \le \tau \le t - t_f} P(t - \tau) + p(t - \tau, t)$
  Let $\tau^* = \text{argmax}_{1 \le \tau \le t - t_f} P(t - \tau) + p(t - \tau, t)$, and get the previous timetable $\mathbf{t}(t - \tau^*)$
  if $p(t - \tau^*, t) > 0$ **then**
    Timetable before and at time t: $\mathbf{t}(t) = (\mathbf{t}(t - \tau^*), t)$
  else
    $P(t) = P(t - \tau^*)$, $\mathbf{t}(t) = \mathbf{t}(t - \tau^*)$
  end if
end for
$P^*(R) = P(t_l)$, and $\mathbf{t}^*(R) = \mathbf{t}(t_l)$

where $1 < i \le r$. And for $i = 1$, the number of on-board passengers at stop b_1 is $N_1(t_1^{k-1}, t_1^k) = N_1^+(t_1^{k-1}, t_1^k)$. Note that the number of on-board passengers cannot exceed the bus capacity, so we only count the on-board passengers as the bus capacity U when $N_i > U$, i.e., $N_i(t_i^{k-1}, t_i^k) = \min\{U, N_i(t_i^{k-1}, t_i^k)\}$.

Profit estimation. We estimate the daily profit of a CB line by subtracting the operational cost along the CB line from the fare revenue. The profit from the k-th bus trip along route R, denoted as $p(t^{k-1}, t^k)$, can be calculated as

$$p(t^{k-1}, t^k) = \sum_{i=1}^{r-1} (N_i(t_i^{k-1}, t_i^k) \cdot \rho_t - \rho_o) \cdot len(b_i, b_{i+1}), \quad (7)$$

where ρ_t is the ticket price per kilometer, ρ_o is the operational cost per kilometer of a bus, $len(b_i, b_{i+1})$ is the route length from b_i to b_{i+1} along route R, t^k denotes the start time for the k-th trip, and $t^0 = 0$. Note that $p(t^{k-1}, t^k)$ could be below 0, i.e., the k-th bus trip could lose money. In this case, we consider this trip as a dummy trip with no profit.

The daily profit of a bus line, denoted as $P(R, \mathbf{t}(R))$, can be calculated by summing up the profit from each trip in a day, i.e., $P(R, \mathbf{t}(R)) = \sum_{k=1}^{s} p(t^{k-1}, t^k)$.

B. Routing

We first discuss basic features a CB route should have. (1) Hamiltonian path. Normally each stop cannot be visited more than once in one trip, indicating that the CB route should be a Hamiltonian path. (2) Precedent constraint. Pick-up stops should be visited before their corresponding drop-off stops along the route. Therefore, a CB route can be defined as the shortest Hamiltonian path that meets the precedent constraint.

Note that searching the shortest Hamiltonian path for a set of stops with the precedent constraint is a typical travel salesman problem (TSP) [11], which is NP-hard. We developed a dynamic programming algorithm, called CBRouting, to generate a CB route for each trajectory cluster.

Let B be the set of deployed CB stops from a cluster. We employ a dummy CB stop b_0, that has the following properties: 1) no passengers come from or to this stop, and 2) the distance between b_0 and any other stops is 0. We set b_0 as the origin stop for all routes. For a subset of stops $S \subseteq \{b_0, B\}$ and $b_0, b_j \in S$, let (S, b_j) be the set of possible paths that start at b_0, visit each node in S exactly once, and finish in b_j. (S, b_j) can only be valid if it satisfies the precedent constraint, i.e.,

- Precedent constraint: if b_j is a drop-off stop, then all its corresponding pick-up stops should be in S; if b_j is a pick-up stop, then any of its drop-off stops cannot be in S.

For a valid set (S, b_j), let $R(S, b_j)$ be the shortest route, and $Len(S, b_j)$ be the length of $R(S, b_j)$. We obtain $R(S, b_j)$ by picking the best second-to-last stop b_i, such that the route length from b_0 to b_i, (i.e., $Len(S - \{b_j\}, b_i)$), plus the length of the final route segment $len(b_i, b_j)$, is minimized:

$$Len(S, b_j) = \min_{b_i \in S: i \ne j,} Len(S - \{b_j\}, b_i) + len(b_i, b_j). \quad (8)$$

We denote b_i^* as the best second-to-last stop, then

$$R(S, b_j) = (R(S - \{b_j\}, b_i^*), b_j). \quad (9)$$

Note that $(S - \{b_j\}, b_i)$ should satisfy the precedent constraint, we prune the invalid sets for each iteration of Eq. 8 and 9.

Our CBRouting is depicted in Algorithm 2. Given a set of CB stops B from one cluster, we iteratively search the shortest route for each subset S with Eq. 8-9, and check the precedent constraint. The CBRouting finally generates the shortest route after the set B is searched, i.e.,

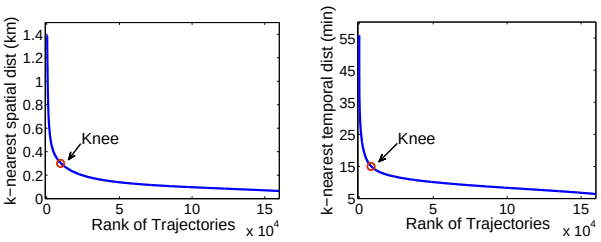
$$R(B) = R(B, b_j^*), \; b_j^* = \text{argmin}_{b_j \in B} Len(B, b_j). \quad (10)$$

C. Timetabling

For each trajectory cluster and its CB route R, we schedule an optimal timetable $\mathbf{t}^*(R)$, i.e., the sequence of start time of each trip along the route in a day, with a dynamic programming algorithm, called CBTimetabling, in order to maximize the total profit of all the trips.

Let t_f and t_l be the pick-up time of the first and last trajectory in this cluster in a day respectively. Then we only need to search the timetable within the time interval of $[t_f, t_l]$.

Suppose from time t_f to t, $t_f < t \le t_l$, we have set a certain number of trips between t_f to t, and t is the start time of the last trip. Let $\mathbf{t}(t)$ be the optimal timetable whose last trip starts at t, and $P(t)$ be the maximum total profit of the trips with $\mathbf{t}(t)$. We obtain $P(t)$ by picking the best start time of the second-to-last trip $t - \tau$, such that the total profit of the



(a) sorted k-spatial-dist (b) sorted k-temporal-dist

Fig. 2. Sorted k-Nearest Spatial and Temporal Distances (k = 4).

trips from time t_f to $t-\tau$, (i.e., $P(t-\tau)$), plus the profit of the last trip $p(t-\tau, t)$, is maximized:

$$P(t) = \max_{1 \le \tau \le t - t_f} P(t-\tau) + p(t-\tau, t), \qquad (11)$$

where $t-\tau$ is the start time of the second-to-last trip, and $p(t-\tau, t)$ can be calculated by Eq. 7. The maximum total profit along the route R, denoted by $P^*(R)$, is obtained by iteratively searching $P(t)$ from time t_f to t_l, i.e., $P^*(R) = P(t_l)$.

Our CBTimetabling is depicted in Algorithm 3. We record the start times of the trips that contribute profit to the maximum total profit as the optimal timetable $\mathbf{t}^*(R)$ (Lines 4 to 9).

V. Experiments

We evaluate the performance of our T2CBS using a large-scale real taxi trajectory data set. In this section, we describe the data set, two baseline algorithms and parameter setting, and present experimental results.

A. Passenger trajectories

The taxi trajectory data set was collected in Nanjing, China from June 1st to 29th, 2010. There are 7,476 taxis with a total of 540,577,652 GPS records. The fields in each record that are related to our study include the taxi ID, timestamp, latitude, longitude, and working status of the taxi. We extract 5,456,051 passenger trajectories from the data set, by detecting the pick-up and drop-off activities. Considering the different travel patterns in weekdays and weekends & holidays, we split the one-month data into two subsets, namely, trajectories in the 20 weekdays and trajectories in the 9 weekends & holidays, and conduct experiments on the two data subsets, separately.

B. Baseline Algorithms

To the best of our knowledge, there is no similar framework proposed for planning bus lines for CB systems in the literature. So we design the following two baseline algorithms:

Top density Stop Deployment (TopSD). TopSD divides the area of pick-up or drop-off points from a trajectory cluster into equal-sized grids, selects the top-K grids with the largest number of pick-up or drop-off points, and deploys a stop at the centroid point in each selected grid. The side length of a grid is set as 2 times the coverage radius of a stop (i.e., $2 \cdot CovRad$). K is the minimum number of stops discovered by our T2CBS. The CB line planning methods in TopSD remain the same as in T2CBS.

Fixed Time Interval (FTI). FTI adopts a timetable with a fixed headway for a route. We set the headway as the waiting time of $waitT$, i.e., the time interval between every two successive bus trips along a route is $waitT$. The stop deployment and routing methods remain the same as in T2CBS.

TABLE I
Default parameter setting

Parameter	Value
Coverage radius of a stop $CovRad$	500 m
Coverage percentage of stops $CovPct$	95%
Walk distance $walkD$	$\{100, 200, 300, 400, 500\}$ m
Waiting time $waitT$	$\{10, 15, 20, 25, 30\}$ min
Capacity of a CB bus U	40
Ticket fare for each passenger ρ_t	2 RMB/km
Operational cost of a bus ρ_o	10 RMB/km

C. Parameter setting

Parameter setting for trajectory clustering. Our T-DBSCAN needs three parameters, c_d, c_t and $MinTrs$. To choose their values, we adapt the heuristic suggested for DBSCAN in [6].

We first define the top-k neighbors of a trajectory with the two steps: (1) *Spatial and temporal sorting*. The neighbors of the trajectory are sorted in ascending order of the spatial distance and the temporal distance separately. The spatial distance is the average distance of pick-up point and drop-off point from a neighbor to the trajectory. And the temporal distance is the pick-up time difference between a neighbor and the trajectory. Each neighbor has its spatial rank and temporal rank. (2) *Spatial-temporal ranking*. We sum up the spatial and temporal ranks for each neighbor, and generate the top-k neighbors.

Among the top-k neighbors of each trajectory, we find the maximum spatial distance and maximum temporal distance, named as "k-nearest spatial distance" and "k-nearest temporal distance", respectively. These two distances define the neighborhood of a trajectory with exactly k neighbors.

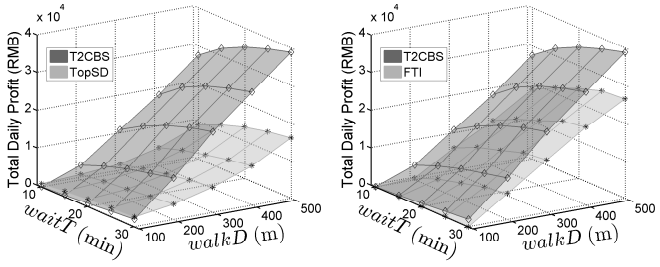
We sort all the trajectories in descending order of their k-nearest spatial distance and k-nearest temporal distance, and form the *sorted k-spatial-dist* graph and the *sorted k-temporal-dist* graph, respectively. As illustrated in Fig. 2, the two graphs reveal the spatial and temporal distributions of trajectories, in which the knees indicate the values of the spatial radius c_d and the temporal radius c_t.

In experiments, we adopt the empirical value of $k = 4$ and set $MinTrs = 4$ as suggested by [6]. Fig. 2 plots the sorted 4-spatial-dist graph and the sorted 4-temporal-dist graph for the weekday data set. We obtain the knee trajectories of the two graphs, and get the 4-nearest spatial distance of the knee from Fig. 2(a) as 300 meters, and the 4-nearest temporal distance of the knee from Fig. 2(b) as 15 minutes. Therefore, we set $c_d = 300$ meters, and $c_t = 15$ minutes. We also get the same parameter values for the weekend & holiday data set.

Other parameter setting. We summarize the parameters in Table. I.

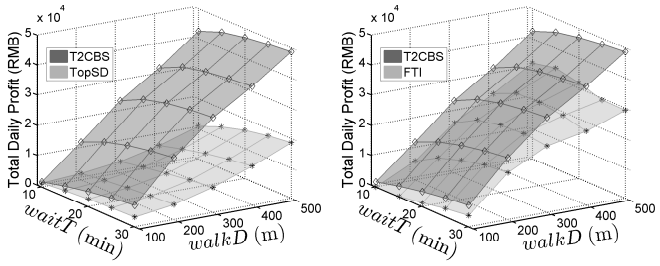
D. Profit Evaluation

We measure the effectiveness of the whole framework using the total daily profit of planned CB lines. Figs. 3-4 present the results, where we observe that our T2CBS always performs



(a) Comparing TopSD (b) Comparing FTI

Fig. 3. Total daily profit of CB lines in weekdays.



(a) Comparing TopSD (b) Comparing FTI

Fig. 4. Total daily profit of CB lines in weekends & holidays.

better than the baseline algorithms for different values of $waitT$ and $walkD$. The details are listed as follows.

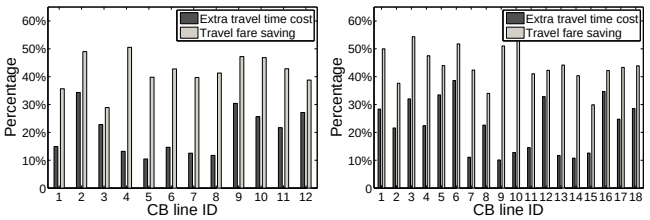
- As $waitT$ and $walkD$ increases, the total daily profit increase for all the methods in both data sets. This is because, with greater $waitT$ and $walkD$, the estimated travel demands for each CB line become greater, leading to greater total profit. We can observe that the profit of our T2CBS increases faster than the baseline algorithms.
- Figs. 3(a) and 4(a) demonstrate the superiority of our T2CBS in deploying CB stops. This shows T2CBS can achieve larger service coverage than TopSD with the same number of deployed stops. Therefore CB lines consisting of these CB stops can make more profit.
- Figs. 3(b) and 4(b) demonstrate the superiority of our T2CBS in timetabling. This is because FTI assigns the same timetable with a fixed headway for all the CB lines, while our T2CBS generates the optimal timetable for each CB line by maximizing its daily profit.

E. Travel experience validation

To validate the travel experience of the CB lines generated by our T2CBS, we compare the travel time and fare of CB buses with that of taxis. Fig. 5 shows that the percentage of travel fare saving is larger than the percentage of extra travel time cost for each CB line in both data sets. For example, in Fig. 5(a), passengers who choose CB line 1 would spend 15% longer travel time than that of taxis, but spend 35% less money. This demonstrates that even though passengers would sacrifice in travel time when choosing CB buses, they still benefit more in travel fare.

VI. Conclusion

In this paper, we propose T2CBS: a holistic framework which aims to strategically plan bus lines for customized bus (CB) systems by mining the taxi GPS trajectory data. To discover travel patterns, we cluster trajectories of taxi passengers with the T-DBSCAN algorithm. And then CB stops are deployed at pick-up and drop-off points of trajectory clusters, with integer linear programming. To plan profitable CB lines, a profit estimation model is proposed. A CB line that achieves the maximum profit in each cluster is generated with a routing algorithm (CBRouting) and a timetabling algorithm (CBTimetabling). Extensive experiments are conducted on one-month taxi trajectory data in Nanjing, China. Experimental results demonstrate that our T2CBS can generate CB lines with higher profit compared with baseline methods, and the moderate increase in travel time along the CB lines is significantly dominated by the savings in bus fare.



(a) Weekdays (b) Weekends & holidays

Fig. 5. Percentage of extra travel time cost and travel fare saving of CB buses compared with taxis.

References

[1] F. Bastani, Y. Huang, X. Xie, and J. W. Powell. A greener transportation mode: flexible routes discovery from gps trajectory data. In *ACM SIGSPATIAL*, 2011.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[3] P. S. Castro, D. Zhang, C. Chen, S. Li, and G. Pan. From taxi gps traces to social and community dynamics: A survey. *ACM Computing Surveys*, 46(2):17, 2013.

[4] A. Ceder. *Public transit planning and operation: theory, modeling and practice*. Elsevier, Butterworth-Heinemann, 2007.

[5] C. Chen, D. Zhang, Z.-H. Zhou, N. Li, T. Atmaca, and S. Li. B-planner: Night bus route planning using large-scale taxi gps traces. In *PerCom*, 2013.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.

[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] S. Lefever, M. Dal, and A. Matthiasdottir. Online data collection in academic research: advantages and limitations. *British Journal of Educational Technology*, 38(4):574–582, 2007.

[9] Y. Li, J. Luo, C.-Y. Chow, K.-L. Chan, Y. Ding, and F. Zhang. Growing the charging station network for electric vehicles with trajectory data analytics. In *ICDE*, 2015.

[10] T. Liu and A. A. Ceder. Analysis of a new public transport service concept: customized bus in china. *Transport Policy*, 39(0):63 – 76, 2015.

[11] A. Mingozzi, L. Bianco, and S. Ricciardelli. Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints. *Operations Research*, 45(3):365–377, 1997.

[12] H. Wang, W. Yang, J. Zhang, J. Zhao, and Y. Wang. Start: Status and region aware taxi mobility model for urban vehicular networks. In *INFOCOM WKSHPS*, 2015.

[13] N. J. Yuan, Y. Zheng, X. Xie, Y. Wang, K. Zheng, and H. Xiong. Discovering urban functional zones using latent activity trajectories. *TKDE*, 27(3):712–725, 2015.

[14] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *ACM Ubicomp*, 2011.