12-2019

Study group travel behaviour patterns from large-scale smart card data

Xiancai TIAN
Singapore Management University, shawntian@smu.edu.sg

Baihua ZHENG
Singapore Management University, bhzheng@smu.edu.sg

Citation

TIAN, Xiancai and ZHENG, Baihua. Study group travel behaviour patterns from large-scale smart card data. (2019). *2019 IEEE International Conference on Big Data: December 9-12, Los Angeles: Proceedings*. 1232-1237.

Available at: https://ink.library.smu.edu.sg/sis_research/4614

Study Group Travel Behaviour Patterns From Large-Scale Smart Card Data

Xiancai Tian, Baihua Zheng
Living Analytics Research Centre, Singapore Management University, Singapore
{shawntian, bhzheng}@smu.edu.sg

***Abstract*—In this paper, we aim at studying the group travel behaviour (GTB) patterns from large-scale auto fare collection (AFC) data. GTB is defined as two or more commuters intentionally and regularly traveling together from an origin to a destination. We propose a method to identify GTB accurately and efficiently and apply our method to the Singapore AFC dataset to reveal the GTB patterns of Singapore commuters. The case study proves that our method is able to identify GTB patterns more accurately and efficiently than the state-of-the-art.**

***Index Terms*—smart card data; spatial and temporal systems; group travel behaviour; Bloom Filter**

I. Introduction

In this paper, we aim at identifying *group travel behaviour (GTB)* patterns of commuters who are travelling using the public transport system (e.g., buses and/or metro), which is a relatively new problem. We fully utilize the city-scale data captured by an *automated fare collection (AFC)* system to perform our study.

Without loss of generality, we assume most, if not all, AFC systems capture the trip information tp_i in the form of $\langle id, s_o, t_o, s_d, t_d, inf \rangle$. Here, id refers to the identity of the smart card, s_o and s_d refer to the boarding location (i.e., the origin) and the alighting location (i.e., the destination) respectively in terms of the respective bus stop or metro station, t_o and t_d stand for the boarding time stamp and the alighting time stamp respectively, and inf refers to other key information captured (e.g., commuting type in terms of bus ride or metro ride, commuter category in terms of child, adult, senior and so on) which could vary from system to system. When the context is clear, we use the term *AFC data* interchangeably with the term *smart card data* or *trip records*, and we use the term tap in/tap out interchangeably with the term boarding/alighting.

In this paper, GTB is defined as two or more commuters *intentionally* and *regularly* traveling together from one location to another. Implied by previous psychological studies [1, 2] and our daily experience, group travellers tend to swipe their cards one right after another. Motivated by this, we propose a smart card data-driven approach to identify GTB in the context of the public transport network. The basic idea is that: if two or more commuters tap in at the same metro station/bus stop within a very short time window and tap out at the same metro station/bus stop within a very short time window, and such synchronization occurs at least a certain number of times, they are identified as *group travellers*. In other words, given two commuters, we determine whether they are group travellers based on *intentionality* and *regularity* of the trips they make together. We quantify the intentionality based on the temporal distance and the spatial distance of the swiping (i.e., tapping in or tapping out) card behavior observed; and we measure the regularity based on the number of times the almost-synchronized swiping card behaviors happened.

There are multiple ways to implement the above mentioned idea to identify whether two or more commuters are group travellers. For example, we could store a trip record as a point in a four-dimensional space (i.e., the dimensions of t_o, t_d, s_o and s_d). For a given trip record tp_i, we could adopt a brute-force approach to scan the trip records to locate those neighboring trips tp_j such that $tp_i.id \neq tp_j.id \wedge |tp_i.t_o - tp_j.t_o| \leq \tau \wedge |tp_i.t_d - tp_j.t_d| \leq \tau \wedge tp_i.s_o = tp_j.s_o \wedge tp_i.s_d = tp_j.s_d$[1]. Parameter τ refers to the temporal threshold which shall be set to a very small value (e.g., 5 seconds in this paper). Obviously, this approach might need to scan many trip records in the collection. Given the fact that the daily ridership of public transport systems in a metropolitan city is in the scale of millions or even more, this exhaustive search is not practical and suffers from very poor scalability. Indexes are available to speedup the search performance. However, given the fact that commuting records are generated in a continuous fashion, updates are expected to happen extremely frequent and the high index update cost makes indexes not an ideal solution.

To address the efficiency issue effectively, we fully utilize the fact that the number of bus stops and metro stations is fixed and trip records are captured by the AFC system according to the chronological order naturally. Accordingly, we strategically store the trip records using tap in chunks and tap out chunks. Records in the same tap in chunks share the same tap in location (i.e., tap in at the same bus stop or metro station), while records in the same tap out chunks have the same tap out locations (i.e., tap out at the same bus stop or metro station). To ease the measurement of the temporal distance, we further partition the records in each chunk using blocks, with records in the same block having their tap in time stamps or tap out time stamps fallen within a very short time window. As to be detailed later, this arrangement effectively reduces the search space of a trip's neighboring trips from the initial N trips to a much smaller number of trips in a few blocks, with N the total number of trip records. We also adopt *Bloom Filter*,

[1]Please refer to Definition 3.1 for the formal definition of neighboring trip.

a probabilistic data structure, to perform the checking which could further improve the performance. In addition, we carry out a case study using real dataset captured by the AFC system in Singapore within one month. We report the performance of our algorithm and share the meaningful insights about group travelling.

The remainder of this paper is organized as follows. Section II reviews previous studies on several related topics, including group walking behaviour. Section III presents our method, including the GTB identifying algorithm BEEP and some relevant techniques used. Section IV reports performance of our algorithm and our findings based on one-month trip records captured by the Singapore AFC system. Section V concludes this paper.

II. Literature Review

The availability of smart card data provides enormous opportunities for public transport research [3]. Much of the existing literature has sought to propose various methods to investigate travel behavior using smart card data. Nevertheless, most of the AFC data-related travel behavior analysis tasks focus on isolated *individual travel behaviour (ITB)*, and few of them pay attention to the study of GTB. One of the most well-studied types of GTB is *group walking behavior (GWB)*. In line with most travel behavior research, early studies on walking behavior have treated pedestrians as isolated individuals, each having a desired speed and direction of motion [2].

More recently, GWB has received substantial attention [2, 4, 5, 6]. Among these studies, identification of pedestrian groups is usually done manually using data collected by video recordings [2, 4]. Other methods have also been adopted, like 3D laser range sensors [6] and accelerometer sensors [7]. While these studies help us understand pedestrian behavior from a group perspective, GWB has so far only been analyzed at a micro scale or in a relatively small area, like a commercial street or a metro station. Thus, these approaches have not been able to provide an understanding of the characteristics of GWB versus *individual walking behavior (IWB)* at a larger spatial scale, such as a neighborhood, a town or even the entire city.

The only piece of work that studies a similar problem is presented in [8], which develops a naive method to identify GTB, using one-week smart card data generated by the metro system in Beijing. It proposes a feasible but very *inefficient* method, which scans the trip records one by one to find those corresponding to the same boarding station and the same alighting station and meanwhile being close to each other in terms of boarding time stamp and alighting time stamp. It does not consider regularity at all so the identified group travellers might not be real group travellers. For example, the subway system in Beijing is well known to be overcrowded during peak hours, and many commuters might tap in the station within a short time window especially when there are multiple gantries in each station. We introduce the concept of regularity into the definition of GTB, which provides a support to the identified group travellers.

Different from existing solutions, we adopt a smart card data-driven approach to identify GTB patterns from large-scale dataset both accurately and efficiently. To our best knowledge, this is the first work on identifying GTB patterns from AFC data that considers the accuracy, the efficiency and also the scalability.

III. Solution Algorithm

In this section, we first formulate the problem of GTB, introduce the data structure *Bloom Filter* that will serve as a key building block of our solution, and then present the solution.

A. Problem Formation

Before presenting our solution algorithm in detail, we introduce two core definitions that will be used throughout the paper in Definition 3.1 and Definition 3.2 respectively.

***Definition** 3.1:* **Neighboring Trip.** For a given trip tp, trip tp' made by a different commuter is considered as a neighboring trip of tp, iff trips tp' and tp board at the same metro station/bus stop within an time interval of τ, and they meanwhile alight at the same metro station/bus stop within a time interval of τ, i.e., $tp'.id \neq tp.id \wedge |tp'.t_o - tp.t_o| \leq \tau \wedge |tp'.t_d - tp.td| \leq \tau \wedge tp'.s_o = tp.s_o \wedge tp'.s_d = tp.s_d$. Here, τ is a threshold controlling how close two neighboring trips should be in the temporal dimension (e.g., we set $\tau = 5s$ in our experiments).

To ease the presentation, we define a Boolean function NEIG(tp_i, tp_j). If trip tp_i is a neighboring trip of trip tp_j, it returns 1; otherwise, it returns 0. Note that neighboring trip relationship is commutative, i.e., NEIG(tp_i, tp_j) = NEIG(tp_j, tp_i). Based on the concept of neighboring trips, we define group traveller in Definition 3.2. Note that $\sum_{\forall tp_i \in Tp_i, tp_j \in Tp_j} \text{NEIG}(tp_i, tp_j)$ captures the total number of neighboring trips commuters c_i and c_j make together. Parameter ρ is the minimum support whose value determines the confidence when two commuters are reported as group travellers. In general, a larger ρ corresponds to a higher confidence level with fewer commuter pairs reported as group travellers, and a smaller ρ corresponds to a lower confidence level with more commuter pairs reported as group travellers.

***Definition** 3.2:* **Group Traveller.** Given commuters c_i and c_j, let Tp_i and Tp_j represent the sets of trips made by c_i and c_j respectively. Commuter c_j is considered a group traveller of c_i iff $\sum_{\forall tp_i \in Tp_i, tp_j \in Tp_j} \text{NEIG}(tp_i, tp_j) \geq \rho$. Here, ρ is a threshold controlling the confidence when two commuters are reported as group travellers.

Again, to ease the presentation, we introduce a Boolean function GROUP(c_i, c_j) that returns 1 if commuter c_i is a group traveller of c_j, and returns 0 otherwise. Group traveller relationship is commutative too. On the other hand, group traveller relationship is not transitive, e.g., if c_2 is a group traveller of c_1 and c_3 is a group traveller of c_2, commuter c_3 may or may not be a group traveller of c_1. This is because the set of neighboring trips commuters c_1 and c_2 travel together

could be different from the set of neighboring trips commuters c_2 and c_3 travel together.

B. Bloom Filter

As proposed in [9], a Bloom filter B is a compact data structure to represent a set $S = \{s_1, s_2, \cdots, s_n\}$ of n elements, in the form of m-bits vector $\langle B[0], B[1], \cdots, B[m-1]\rangle$. It chooses k independent hash functions h_1, h_2, $\cdots$, and h_k, each with a range of $[0, m-1]$. Initially, all the bits in the bloom filter B are set to 0. Thereafter, for each element $s_i \in S$, the bits $B[h_1(s_i)]$, $B[h_2(s_i)]$, $\cdots$, and $B[h_k(s_i)]$ in the bloom filter are set to 1.

To check whether a given element e is in S, we could just compare the bloom filter of S and that of e. To be more specific, we generate a m-bits vector B_e for element e, using the same set of k hash functions h_1, h_2, $\cdots$, and h_k used for constructing the bloom filter B for the set S. We perform bit-wise and operation (denoted as &&) to determine whether the 1-bits in B_e are also set to 1 in B, i.e., whether $B\&\&B_e = B_e$. There are only two possible outputs, i.e., a match or a mismatch. If $B\&\&B_e \neq B_e$ (i.e., a mismatch), element e is guaranteed to be NOT in S, as bloom filter does not cause any false negative (i.e., $B\&\&B_e \neq B_e \Rightarrow e \notin S$); otherwise (i.e., $B\&\&B_e = B_e$), there is a high probability that e is in S. In other words, the Bloom filter admits controlled false positive rates but no false negatives, with its false positive rate f being $(1-(1-\frac{1}{m})^{kn})^k$.

C. Identifying Group Travelers

According to Definition 3.2, we understand that the efficiency of calculating $\sum_{\forall tp_i \in Tp_i, tp_j \in Tp_j} \text{NEIG}(tp_i, tp_j)$ plays a key role in studying GTB. As explained before, to make the search faster, we need to control the number of trip pairs (tp_i, tp_j) we need to evaluate. As the output of $\text{NEIG}(tp_i, tp_j)$ is either 1 or 0, only those trip pairs (tp_i, tp_j) with $\text{NEIG}(tp_i, tp_j) = 1$ actually affect the value of $\sum_{\forall tp_i \in Tp_i, tp_j \in Tp_j} \text{NEIG}(tp_i, tp_j)$. In other words, we shall *filter out* the trip pairs with their corresponding $\text{NEIG}(tp_i, tp_j) = 0$ from evaluation as many as possible.

In this paper, we organize the trip records via a very novel concept *chunk*. There are two types of chunks, namely *tap in chunks* and *tap out chunks*, denoted as $C^{in}_{s_i}$ and $C^{out}_{s_i}$ respectively. The former captures all the tap-in actions observed at a specific metro station/bus stop s_i, and the latter refers to the tap-out actions observed at s_i, both following the chronological order. Each record in $C^{in}_{s_i}$ is in the form of $\langle id, s_i, t_o\rangle$, and each record in $C^{out}_{s_i}$ is in the form of $\langle id, s_i, t_d\rangle$. A complete trip record consists of a record $\langle id, s_i, t_o\rangle \in C^{in}_{s_i}$ and a record $\langle id, s_j, t_d\rangle \in C^{out}_{s_j}$ such that $\nexists \langle id, s'_j, t'_d\rangle \in \cup C^{out}_{s_j}$ such that $t'_d < t_d \wedge t'_d > t_o$. That is to say a tap in record and its immediate tap out record form a complete trip record.

The size of $C^{in}_{s_i}$ or $C^{out}_{s_i}$ keeps increasing as new trips are made by commuters. As neighboring trips only refer to two trips happening within a very small temporal window controlled by parameter τ, we further partition the records in $C^{in}_{s_i}$ or $C^{out}_{s_i}$ into small blocks, with each block spanning a temporal window of size τ.

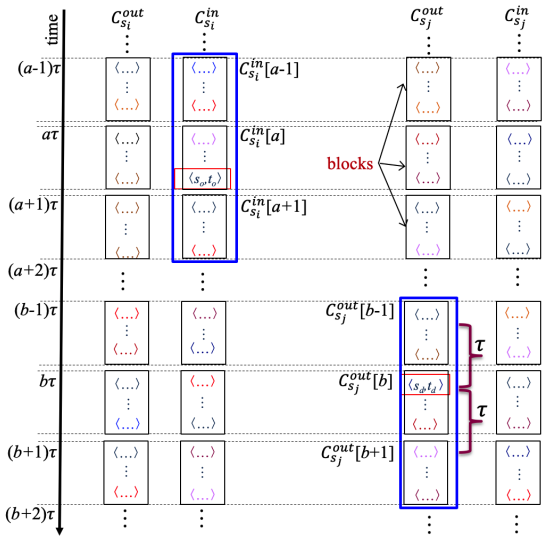

Fig. 1: Chunk storage of AFC records ($t_{ini} = 0$)

Fig. 1 provides a visualization to facilitate the understanding of chunks and blocks. Each black colored rectangular box containing a set of $\langle \cdots \rangle$ is a block, and blocks of the same column form a chunk. Assume an AFC system starts working at an initial time stamp t_{ini}. Then, all the tap-in records captured by each station s_i within the time window $[t_{ini}, t_{ini}+\tau)$ are preserved by the first block of $C^{in}_{s_i}$, denoted as $C^{in}_{s_i}[0]$, all the tap-in records captured at each station s_i within the time window $[t_{ini}+\tau, t_{ini}+2\tau)$ are preserved by the second block in $C^{in}_{s_i}$, denoted as $C^{in}_{s_i}[1]$, and so on. The same applies to tap-out records. We further assume $t_{ini} = 0$ for brevity in the rest of the paper. Note that there could be many empty blocks.

When trip records are preserved by chunks/blocks as explained above, we can efficiently conduct a search to locate all the neighboring trips tp_j for a given trip $tp_i = \langle id, s_o, t_o, s_d, t_d\rangle$, as stated in Lemma 1. The notation $C^{in}_{s_o}[x:y]$ with $x < y$ refers to the union of blocks $C^{in}_{s_o}[x] \cup C^{in}_{s_o}[x+1] \cup \cdots \cup C^{in}_{s_o}[y]$.

Lemma 1: For a given trip $tp_i = \langle id^i, s^i_o, t^i_o, s^i_d, t^i_d\rangle$, let $a = \lfloor \frac{t^i_o}{\tau} \rfloor$ and $b = \lfloor \frac{t^i_d}{\tau} \rfloor$. Assume $tp_j = \langle id^j, s^j_o, t^j_o, s^j_d, t^j_d\rangle$ is a neighboring trip of tp_i. It is guaranteed that $\langle id^j, s^j_o, t^j_o\rangle \in C^{in}_{s^i_o}[a-1:a+1]$ and $\langle id^j, s^j_d, t^j_d\rangle \in C^{out}_{s^i_d}[b-1:b+1]$.

Proof 1: Let's prove by contradiction, and assume the above statement is not true. That is to say there is at least one trip tp_j with $\text{NEIG}(tp_i, tp_j) = 1$ such that $\langle id^j, s^j_o, t^j_o\rangle \notin C^{in}_{s^i_o}[a-1:a+1]$ or $\langle id^j, s^j_d, t^j_d\rangle \notin C^{out}_{s^i_d}[b-1:b+1]$. If $\langle id^j, s^j_o, t^j_o\rangle \notin C^{in}_{s^i_o}[a-1:a+1]$, $\langle id^j, s^j_o, t^j_o\rangle \in C^{in}_{s^i_o}[0:a-2] \cup C^{in}_{s^i_o}[a+2:\infty]$. That is to say, $t^j_o < (a-1)\tau$ or $t^j_o \geq (a+2)\tau$. As $t^i_o \in [a\tau, (a+1)\tau)$, $|t^j_o - t^i_o| > \tau$ which contradicts with the fact that $\text{NEIG}(tp_i, tp_j) = 1$. Similarly, if $\langle id^j, s^j_d, t^j_d\rangle \notin C^{out}_{s^i_d}[b-1:b+1]$, we have $|t^i_d - t^j_d| > \tau$ which also contradicts with the fact that $\text{NEIG}(tp_i, tp_j) = 1$. Consequently,

our assumption is invalid and the above statement is true. □

Although Lemma 1 could effectively shrink the search space of neighboring trips for a given trip tp_i, it could be still expensive to check whether a tap in record r_{in} appearing in $C^{in}_{s^i_o}[a-1:a+1]$ and a tap out record r_{out} appearing in $C^{out}_{s^i_d}[b-1:b+1]$ could form a neighboring trip of trip tp_i. If r_{in} and r_{out} could form a neighboring trip of trip tp_i, they must have the same ID, i.e., the tap in action and the tap out action are indeed performed by the same commuter. Motivated by this observation, we propose to take the advantage of bloom filter to facilitate the checking.

Given a target trip tp_i in the form of $\langle id^i, s^i_o, t^i_o, s^i_d, t^i_d\rangle$, let tap out record $r_{out} = \langle id, s_d, t_d\rangle$ be one tap out record with $s_d = s^i_d \wedge |t_d - t^i_d| \leq \tau \wedge id \neq id^i$. We need to verify whether there is a tap-in record r_{in} belonging to id present in $C^{in}_s[x]$ such that the tap in record r_{in} and the tap out record r_{out} could form a trip tp_j that is a neighboring trip of trip tp_i.

To make the above mentioned search faster with the help of bloom filter, we, for each non-empty block $C^{in}_s[x]$, build a bloom filter B^x_s to represent the IDs corresponding to all the tap-in records preserved by $C^{in}_s[x]$. To check whether there is a record $r_{in} \in C^{in}_s[x]$ representing the tap in action of r_{out}, we first need to guarantee that $r_{out}.id$ appears in $C^{in}_s[x]$. Accordingly, we construct a bloom filter for $r_{out}.id$, denoted as B_{id}, and check whether $B^x_s \&\& B_{id} = B_{id}$, as explained in Section III-B. If $B^x_s \&\& B_{id} \neq B_{id}$, it is guaranteed that there is no tap in record preserved by the block $C^{in}_s[x]$ that is related to r_{out}. Consequently, the tap out record r_{out} can not be a part of any neighboring trips of trip tp_i, and hence could be safely pruned. Otherwise, we scan the records in $C^{in}_s[x]$ one by one. There are two purposes behind the scanning. First, we need to verify that whether there is a record $r_{in} \in C^{in}_s[x]$ with $r_{in}.id = r_{out}.id$ as bloom filter might cause false positive. Second, if there is a such record (i.e., true positive), we need to compare $r_{in}.t_o$ with the tap in time stamp of trip tp_i to make sure it is within τ away from t^i_o.

Now we are ready to introduce the *block-based neighboring trip search algorithm* (in short BEEP), with its pseudo code presented in Algorithm 1. It locates all the neighboring trips, if any, for a given trip record tp_i (i.e., the target trip). First, we locate the blocks $C^{in}_{s^i_o}[a]$ and $C^{out}_{s^i_d}[b]$ that accommodate the tap in record and tap out record of the target trip tp_i respectively, and initialize $result$ to be an empty set (Line 1). Note, $result$ is a set to store all the neighboring trips of the target trip. Next, we check each tap out record r_{out} in the form of $\langle id, s_d, t_d\rangle$ in $C^{out}_{s^i_d}[b-1:b+1]$, as guided by Lemma 1 (Lines 3-9). For a tap out record r_{out}, if its id is the same as the id of the target trip, or the temporal difference between its tap out timestamp and that of target trip is larger than τ, it can be safely pruned. Otherwise, we continue the evaluation. We generate a bloom filter B_{id} based on $r_{out}.id$ (Line 4). Note the function BLOOMFILTER() is the construction algorithm employed to generate bloom filters $\cup B^x_s$ corresponding to all the tap-in blocks. We compare B_{id} with that of tap in blocks. If the bloom filter indicates a match, we invoke the function

Algorithm 1: Block-based Neighboring Trip Search Algorithm (BEEP)

Input: trip records preserved by blocks $\cup C^{in}_s$ and $\cup C^{out}_s$ Bloom filters $\cup B^x_s$ for all the tap-in blocks target trip $tp(id^i, s^i_o, t^i_o, s^i_d, t^i_d)$ and parameters τ

Output: all the neighboring trips of tp

1 $a \leftarrow \lfloor \frac{t^i_o}{\tau} \rfloor$, $b \leftarrow \lfloor \frac{t^i_d}{\tau} \rfloor$, $result \leftarrow \emptyset$
2 **for** *each tap out record* $r_{out} \in C^{out}_{s^i_d}[b-1:b+1]$ **do**
3 **if** $r_{out}.id \neq id^i \wedge |r_{out}.t_d - t^i_d| \leq \tau$ **then**
4 $B_{id} \leftarrow$ BLOOMFILTER($r_{out}.id$)
5 **for** $(x \leftarrow a-1; x < a+1; x \leftarrow x+1)$ **do**
6 **if** $B_{id} \&\& B^x_{s^i_o} = B_{id}$ **then**
7 $temp \leftarrow$ SCANNING(tp, $r_{out}.id$, $C^{in}_{s^i_o}[x]$, τ))
8 **if** $temp \neq \emptyset$ **then**
9 $result \leftarrow result \cup temp$, **break**
10 **return** the set of neighboring trips $result$

SCANNING() to perform the detailed examination (Line 6-7). The examination of a tap out record r_{out} could be safely terminated once it is reported that r_{out} is part of a neighboring trip of the target trip (Lines 8-9). The search completes when all the tap out records in $C^{out}_{s^i_d}[b-1:b+1]$ have been evaluated. We return the result set $result$ to end the process (Line 10).

With the help of BEEP algorithm, we can find the group travellers of any target commuter easily. Let id be the smart card id of our target commuter c_i, and we want to locate all the group travellers for c_i based on AFC data collection X. First, we can locate all the trips Tp_i made by the commuter c_i. For each trip $tp \in Tp_i$, we look for its neighboring trips via BEEP, and assume the returned result is stored by a set $neig_{tp}$. If $neig_{tp}$ is not empty, we scan the trips in $neig_{tp}$ one by one. Each trip $tp' \in neig_{tp}$ increases the counter associated with the pair of commuters $\langle id, tp'.id\rangle$ by one. When a counter reaches the parameter ρ, $tp'.id$ will be reported as a group traveller of id.

The above mentioned group traveller identification algorithm can also be easily adjusted to the online setting where new trip records keep coming. Let C be the set of IDs of our target commuters. When new tap in actions and tap out actions are performed, we preserve all the records using blocks (i.e., new blocks are generated, and new bloom filters are constructed). If a tap in action is taken by one of the target commuters say $c_i \in C$, we monitor the immediate tap out action. Once the tap out action is performed, we have the complete information of the latest trip made by commuter c_i. We then invoke BEEP algorithm to locate the neighboring trips and use the neighboring trips to update the counter values of qualified commuter pairs. Because of the superior efficiency of BEEP algorithm, we are able to support real-time identification of group travellers in a city scale (i.e., C could contain millions

of IDs).

IV. Case Study

To verify our solution, we perform a case study using the trip records collected in the month of April 2016 by the Singapore AFC system. For illustration purposes, we randomly pick one million smart cards and apply our algorithm solution to identify group travelers for each smart card. We first report the search performance of our algorithm solution and then share our findings.

Commuters' travel patterns depend on multiple factors, e.g., the day of the week, the time of the day, and the commuter category. This study investigates the underlying GTB patterns among four age groups of commuters, i.e., children, students, adults, and seniors[2].

A. Efficiency Evaluation

One of the key contributions of this work is an efficient search algorithm that can detect the group travellers in real-time on the city scale. To demonstrate the superior search performance of BEEP based search algorithm, the first set of experiments is to evaluate the search performance. In the following, we first study the sensitivity of parameter ρ and then report the search performance.

As mentioned in Section III, the value of ρ determines the confidence of identified group travellers. To find out the influence of ρ on the number of identified group travellers, we conduct a sensitivity analysis to determine the optimal value of ρ. As reported in Fig 2, we sample the trips made by $10,000$ out of the 1 million sample commuters in Singapore within one month's period and report the ratio of group travelers, with ρ varied from 1 to 10. Based on the elbow criterion, 4 or 5 would be most suitable for identifying GTB in the Singapore case. In what follows, we have therefore set ρ to the value of five to identify GTB, i.e., two commuters shall take at least five neighboring trips together within one month in order to be reported as group travellers.

Next, we compare the *execution time* required by our algorithm solution and a baseline solution to identify group travellers, with the same experimental setup. A filtering-and-refinement alike model is implemented as the baseline, where all the trip records are sorted based on tap-in time stamp (i.e., t_o) in ascending order. Given a trip tp, the baseline searches for all the trips that are within τ distance to tp in the t_o dimension based on binary search to form a candidate set. It then evaluates each trip in the candidate set to test if it meets the neighboring trip criteria in other dimensions. Note this model performs much better than the brute-force approach mentioned in Section I.

Fig. 3 reports the execution time with the size of cards varied. It can be observed that the execution time of both models increases linearly as the number of cards becomes larger. However, the execution time of the two models grows at very different speeds. On average, our model is running five times faster than the baseline approach and it demonstrates much better scalability. We also test the time required to find all the neighboring trips of a trip in a real-time online setting. On average, it takes BEEP 15 milliseconds to retrieve all the neighboring trips of a given trip during a peak hour. That is to say, the throughput is about 66 trips per second at one server (i.e., a server can find the neighbouring trips for 66 independent trips within one second) which is far beyond the tap out rate at any station/bus stop. This further demonstrates that the proposed approach can handle the detection of GTB in a city scale efficiently.

B. Characteristics of Group Travelling Patterns

1) Ratio of group travellers: We plot the ratio of group travellers for each commuter category in Fig. 4. As we can observe clearly from the plot, the ratio of group travellers differs significantly across commuter categories, i.e., children and students have a much higher ratio of group travelers than adults and seniors. One possible reason is that children (i.e., kids below 7 years old) and some students (e.g., primary students) are too young to travel alone, particularly when making long-distance trips, and are usually accompanied by at least one caregiver. On the other hand, seniors have a higher ratio of group traveller than adults, which is consistent with our expectation that seniors have more free time and are more actively engaged in group events (e.g., community events, group exercise classes and celebration events) than working adults, which may contribute to the higher ratio of group travellers.

2) Ratio of group trips: For two commuters c_i and c_i that are identified as the group travellers (i.e., $\text{GROUP}(c_i, c_j) = 1$), we name the neighboring trips made by c_i and c_j as the *group trips*. In this set of experiments, we study, for each commuter c_i that is identified as a group traveller, the rate of the number of group trips made by c_i with any of his group travellers to the total number of trips c_i makes, namely *group trip ratio*. The result is plotted in Fig. 5.

We could observe that children have a much higher group trip ratio than the other three categories of commuters. The group trip ratio for most children is around 80%, which is reasonable since children are almost always supervised by someone else when traveling. Interestingly, the kernel density plot for student commuters displays a double-peak style. The second peak at the end of higher group trip ratios is very likely to be contributed by younger students, e.g., primary school students who are still expected to travel together with caregivers, while the elder students (e.g., secondary school students and above) can travel alone. Adults and senior travellers complete most of their trips on their own, which is reflected by their right-skewed distributions of the group trip ratios.

3) Group size: Next, we study the size of groups, which is defined as the number of commuters involved in the same group trip. The overall group size distribution is reported in Fig. 6. Obviously, group trips by two commuters dominate the share of all group trips for all the four categories of commuters. In particular, seniors travel in pairs in more than

[2]*Commuter category* is an attribute associated with each smart card and meanwhile captured by the AFC system in Singapore.

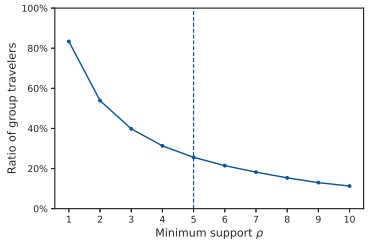

Fig. 2: Sensitivity analysis of ρ

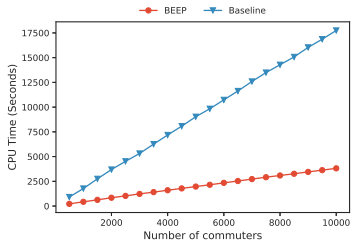

Fig. 3: Efficiency comparison

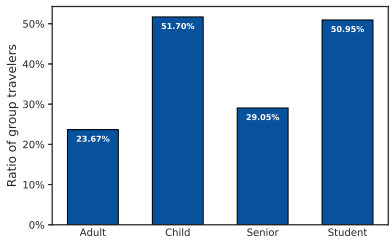

Fig. 4: Ratio of group travelers

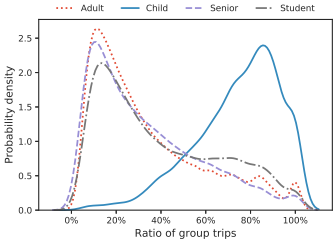

Fig. 5: Ratio of group trips

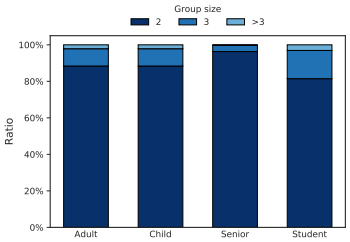

Fig. 6: Group size

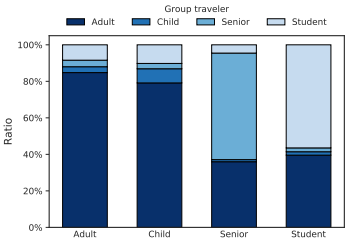

Fig. 7: Who travel with me

95% of group trips. In addition, we find that students are more likely to travel in bigger groups, i.e., group trips with 3 or more commuters.

4) Who travel with whom?: Last but not least, we are interested in finding out who travels with whom, with the results plotted in Fig. 7. On the one hand, the most popular category of group travelers for both adults and children is adults, since about 80% of their group travelers are adults. On the other hand, seniors like to take public transport with fellow seniors, and students travel together very frequently with students, which are possibly their classmates or schoolmates.

V. Conclusion

In this paper, we have madethe following main contributions. First, we demonstrate the potential of using smart card records to gain insights into GTB patterns of public transport users. Second, we propose an efficient method to identify GTB patterns on large-scale datasets. Third, we report insights of GTB patterns in the context of Singapore public transport. We are confident that the proposed method does shed the first light on the pattern of GTB at the metropolitan scale. In the near future, we plan to extend the model to study the group behaviors of activities beyond commuting (e.g., shopping, dining).

Acknowledgments

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative. The authors would like to thank the Land Transport Authority (LTA) of Singapore for providing the data. However, we declare that all the findings shared in this paper represent the opinions of the authors but not LTA.

References

[1] James A Cheyne and Michael G Efran. "The effect of spatial and interpersonal variables on the invasion of group controlled territories". In: *Sociometry* (1972), pp. 477–489.

[2] Mehdi Moussaıd et al. "The walking behaviour of pedestrian social groups and its impact on crowd dynamics". In: *PloS one* 5.4 (2010), e10047.

[3] Xiancai Tian and Baihua Zheng. "Using Smart Card Data to Model Commuters' Responses Upon Unexpected Train Delays". In: *IEEE Big Data'18*. 2018, pp. 831–840.

[4] Ursula Polzer. "Nonverbal behavior in public space as a function of density and group size". PhD thesis. uniwien, 2011.

[5] Giuseppe Vizzari, Lorenza Manenti, and Luca Crociani. "Adaptive pedestrian behaviour for the preservation of group cohesion". In: *Complex Adaptive Systems Modeling* 1.1 (2013), p. 7.

[6] Francesco Zanlungo, Dražen Brščić, and Takayuki Kanda. "Pedestrian group behaviour analysis under different density conditions". In: *Transportation Research Procedia* 2 (2014), pp. 149–158.

[7] Kleomenis Katevas et al. "Walking in Sync: Two is Company, Three's a Crowd". In: *ACM WPA-15*. 2015, pp. 25–29.

[8] Yongping Zhang, Karel Martens, and Ying Long. "Revealing group travel behavior patterns with public transit smart card data". In: *Travel Behaviour and Society* 10 (2018), pp. 42–52.

[9] B. Bloom. "Space/Time Trade-Offs in Hash Coding with Allowable Errors". In: *Communications of the ACM* 13.7 (1970).