1-2024

From a timeline contact graph to close contact tracing and infection diffusion intervention

Yipeng ZHANG

Zhifeng BAO

Yuchen LI
Singapore Management University, yuchenli@smu.edu.sg

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

Xiaoli WANG

Citation

ZHANG, Yipeng; BAO, Zhifeng; LI, Yuchen; ZHENG, Baihua; and WANG, Xiaoli. From a timeline contact graph to close contact tracing and infection diffusion intervention. (2024). *IEEE Transactions on Knowledge and Data Engineering*. 1-12.
Available at: https://ink.library.smu.edu.sg/sis_research/9035

From A Timeline Contact Graph to Close Contact Tracing and Infection Diffusion Intervention

Yipeng Zhang, Zhifeng Bao, Yuchen Li, Baihua Zheng, Xiaoli Wang

Abstract—This paper proposes a novel graph structure to address the problems of information spreading in a real-world, frequently updating graph, with two main contributions at hand: accurately tracing infection diffusion according to fine-grained user movements and finding vulnerable vertices under the virus immunization scenario to mitigate infection diffusion. Unlike previous work that primarily predicts the long-term epidemic trend at the census level, this study aims to intervene in the short-term at the individual level. Therefore, two downstream tasks are formulated to illustrate practicalities: Epidemic Mitigating in Public Area problem (EMA) and Epidemic Maximized Spread in Public Area problem (ESA), where EMA aims to find intervention strategies, and ESA is an adversarial solution against the intervention strategy to test the robustness. Comprehensive experiments are conducted using two real-world datasets with millions of public transport trips, which demonstrate the effectiveness of our approach and highlight the importance of considering the dynamic nature of close contacts in epidemic modelling.

Index Terms—Graph Structure, Infection Diffusion.

1 INTRODUCTION

Graph theory has been a powerful tool for analyzing various network structures, such as social networks, web graphs, and road networks. However, most existing data structures for graph analysis focus on static graphs, which is inadequate for real-world scenarios where graphs change frequently. For example, in a social network, people constantly add and remove followers; in a transportation system, passengers constantly get on and off buses and trains; in road networks, vehicles merge and leave traffic flows. This can make it challenging to track the spread of information, such as a virus or an idea, through the graph.

In this work, we focus on how the information spreads in the temporal/dynamic graph, where the graph structure may change frequently. For instance, tracking the spread of diseases through crowds requires modelling close contact between people when they move over time [1], [2], [3], a regional communication system built spontaneously to connect multiple mobile wireless devices without requiring typical network infrastructure equipment [4], or the ad-hoc radio network between moving vehicles as a local wireless network to exchange information [5], [6]. However, traditional graph data structures are expected to handle

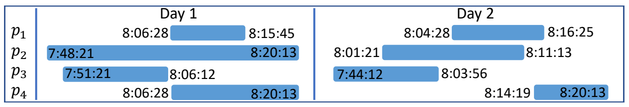


Fig. 1: A two-day check-in record of four persons from p_1 to p_4 at the same location. Each blue bar indicates the period of the corresponding person staying at the location. The two times shown below each bar are the check-in/out times.

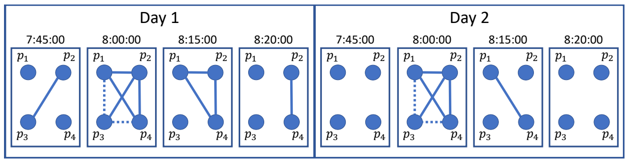


Fig. 2: Eight snapshots are updated every 15 minutes. Each snapshot is generated based on Fig 1, where each line between two dots indicates that the two corresponding persons are staying at one location in that 15 minutes. Particularly, solid lines represent the correct contacting relationships, whereas dot lines are incorrect.

these dynamic connections better. Otherwise, it may lead to inaccuracies and inefficiencies.

To tackle this problem, we propose a new data structure, Timeline Contact Graph (TCG), which is specifically designed to handle frequently changing graphs. Our data structure can be efficiently generated based on city-wide individual movement data and hence capture all contact among persons. This allows us to accurately model the spread of information or diseases through the graph. In a nutshell, each instance, such as a person, a vehicle, or an account in the social network, is a vertex in our data structure. Whenever a change affects the connections between this vertex and its neighbours, we will create a mirror for this vertex connecting to new neighbours, whereas most existing

This project is supported in part by ARC DP220101434 and Natural Science Foundation of Fujian Province of China (No. 2021J01003). The work was conducted when the first author was a PhD student at RMIT.

- *Y. Zhang is with the CSIRO, Melbourne, VIC, Australia. E-mail: yipeng.zhang@data61.csiro.au*
- *Z. Bao is with the School of Computing Technolgies, RMIT University, Melbourne, VIC, Australia. E-mail: zhifeng.bao@rmit.edu.au*
- *Y. Li and B. Zheng are with the School of Computing and Information System, Singapore Management University, Singapore. E-mail: yuchenli/bhzheng@smu.edu.sg*
- *X. Wang is with the School of Informatics, Xiamen University, Xiamen, Fujian, China. E-mail: xlwang@xmu.edu.cn*
- *Corresponding Author: Xiaoli Wang.*

work mainly leverages massive snapshots to maintain the information of the temporal graph [7], [8], [9].

There are some drawbacks to snapshot-based solutions. *First*, there is a dilemma between capturing correct edge connections and maintaining an affordable memory cost. The snapshot-based solution updates with an interval between two snapshots, which may cause the incorrect connection capture when a connection crosses multiple snapshots. For example, in Fig 2, the dot-lines between p_1, p_3, and p_4 on both days are not supposed to be connected. However, decreasing the update interval will significantly increase the number of snapshots. *Second*, even if the hardware is powerful enough to handle massive snapshots, the influence diffusion is hard to split into different snapshots. For any nonlinear diffusion model that measures the influence exchanging between two nodes, it is hard to evaluate the total influence from split segments by simply integrating the influence of each segment with the linear normalization factors. More details are shown in Section 3.1.

Our Contributions:

• *Graph Structure.* In order to deal with the temporal graph with a rapidly changing graph structure and efficiently find how the information spreads through the temporal graph, we propose a novel graph structure, namely Timeline Contact Graph (TCG). It can be created directly based on vertices information such as the public transport system record, or transform the enormous discrete graph structures (e.g., snapshots) into a TCG graph, to avoid creating enormous snapshots and hence overcome the dilemma between losing the accuracy of changing edges among vertexes and suffering the explosive amount of data (Section 3).

• *Evaluation Downstream Tasks.* In order to illustrate how our work helps overcome the disadvantage of snapshot-based solutions on the huge temporal graph, we design two downstream takes under the Virus Immunization scenarios for concreteness. The general applicability of our graph structure will be illustrated later in Section 7.

Task 1. Tracing Infection Diffusion in Short-term. The virus infection is rapidly spreading with cascading diffusion and exponentially growing close contacts of infected individuals. However, these close contacts between people change constantly. In this context, we aim to identify all potential infections caused by close contact with known infected persons. Given a set of infected individuals and a temporal network representing check-in/out records, our goal is to find all potentially infected individuals based on a diffusion model (Section 4).

Task 2. Detecting Vulnerable vertices. Understanding a network's vertex vulnerability is crucial for addressing issues like virus immunization and influence maximization. In our study, we focus on identifying the network's most vulnerable vertices. Our goal is twofold: firstly, we aim to identify highly influential individuals or 'super spreaders' contributing to pandemic spread (Section 5), and secondly, we seek to locate the most dangerous points of interest (POIs) for effective virus intervention (Section 6). By identifying these key vertices, we can better understand network dynamics and devise robust virus control strategies. We prove the task's NP-hardness theoretically and introduce our algorithm to address the problems.

• *Simulator & Experiment.* We carefully design experiments based on two large-scale real-world movement datasets to evaluate all baselines. The experimental result shows that (1) Simulation: our solution is up to 20 times more accurate than baselines; (2) Intervention: our solution outperforms the best baselines about three times by average; (3) Scalability: our solution is scalable to handle four weeks of movement data with tens of millions trips. s

2 RELATED WORK

In this section, we review related work in the two most closing domains of virus immunization and facility location. Overall, the contributions and related work are connected through their shared focus on addressing the challenges posed by temporal graphs in the context of virus control and prevention.

2.1 Virus Immunization

The Virus Immunization (VI) work aims to study the prevention and diffusion of the virus. The three most common compartments in all epidemic models are Susceptible (S), Infected (I), and Recovered (R) [10]. S represents the set of people who are healthy but susceptible to being infected. I represents the set of people who are infected but are able to recover. R represents the set of people who have recovered but may be infected again based on different problem settings. Our work is fundamentally different from most VI studies from the perspective of the objectives, and the methodology, respectively, as illustrated below.

Objective: Most studies [2], [3], [11], [12], [13], [14] in VI focus on the question of "will the pandemic happen at long-term at *population-level*?" The difference is three-fold: time period, analytical granularity, and goal. For instance, given a graph G, the *goal* of the work [13] is to find a threshold of the first eigenvalue λ^*, such that the virus will diminish when the time approaches infinity (*time period*) if λ of G is smaller than λ^*, otherwise it will become a pandemic. Each node in the graph represents a city (*analytical granularity*). Nadini et al. [15] investigate the effects of modular and temporal connectivity patterns on epidemic spreading among communities, and the interval of snapshots is one month.

In contrast, we focus on accurately tracking the influence of diffusion in a graph within a short time, where the diffusion is based on the physical distance. Intuitively, under the texture of the influence diffusion, we answer the question of "who will be influenced exactly?"

Methodology: Existing VI studies fall into two categories, based on either the betweenness centrality [16] or the eigenvalues [17], [18]. However, most existing solutions do not take the dynamic graph into consideration, whereas, the time-varying connectivity pattern of networks is essential to the epidemic process [19], [20]. A few studies [9], [15], [18], [21] that support the varying graph or temporal graph are suffering an accuracy and efficiency issues since they are the snapshot-based solution.

2.2 Facility Location

Due to the ever-growing city scale, the fierce competition among businesses, and residents' strong willingness to chase a better quality of life, it is crucial for companies and governments to devote more effort to city planning that will benefit neighbourhoods and the people who live there. It aims to model complex real-world scenarios and solve various facility location applications in promoting social benefits or saving costing.

Ali et al. [22] study two problem, k Best Facility Trajectory Search (kBFT) and k Best Coverage Facility Trajectory Search (kBCovFT). The former one aims to find the top-k locations from a given dataset that each of them covers the maximum number of trajectories generated from a set of users than the rest locations, while the latter one aims to find a subset of locations with a cardinality k from the dataset, such that the number of trajectories covered by this subset is maximized. Several work [23], [24] has been published focusing on the similar ideal of maximizing the total number of covered trajectories.

Yilmaz et al. [25] consider a scenario where the locations of the customers are unknown. They propose an optimal location predictor that accepts partial information about customer locations and returns a location for the new facility. Ahmadian et al. [26] define a problem where points are referred to as clients and clusters are defined by the assignment of clients to centres. The goal is to find a feasible solution that minimizes the maximum radius or the clustering cost of the total k facilities. Gomez-Rodriguez al. [27] focus on inferring the probability of each edge, given the graph structure and initial/final states of nodes.

Most facility location research focuses on maximizing the number of covered customers. While one of our contributions involves deploying facilities at POIs, maximizing customers is insufficient for our problem of preventing information from spreading in a temporal graph among continuously moving individuals. Therefore, the facility location problem cannot address the challenges posed by our dynamic graph scenario.

3 Timeline Contact Graph

In this section, we will introduce a novel Timeline Contact Graph (TCG). We first raise three challenges of snapshot-based solutions, then introduce essential notations and define TCG, describe its construction from temporal graphs, and demonstrate its equivalence to all temporal graphs.

3.1 Pain Points

Challenge 1 - Incorrect connection capture. Challenge 1 - Incorrect connection capture. In a temporal graph, edges and vertices can be added or removed anytime. Therefore, a snapshot-based solution may include edges between vertices that have since been removed. For example, in Fig 2, for each snapshot, we connect two persons if they stay in the same period. Since they all are at the same location from 8:00:00 to 8:14:59 on day 1, they are connected in the third snapshot, which lead to incorrect connections (the dot lines) as p_3 does not contact with p_1 and p_4.

Challenge 2 - High memory cost. To record the correct check-in/out data shown in Fig 1, we need a higher update frequency. Instead of setting a concise duration, a more reasonable solution is event-based snapshotting that creates new snapshots whenever check-in/out happens. However, this event-based snapshotting captures precise graph states but can lead to prohibitive memory costs at a city-wide scale, as millions of check-in/out happen daily, which poses a dilemma between accuracy and memory efficiency.

TABLE 1: Important notations

Notation	Description
t_{start}, t_{end}	Start time, end time
$\overrightarrow{t}$	A duration from from t_b to t_e
s	A location
p_n	A person. Each Person has multiple records $p_n.Tr = \{tr_i, \cdots, tr_j\}$.
p_n^i	A movement record. p_n^i and $p_n.tr_i$ are interchangeable.
$tr_i \in \mathcal{R}$	A movement formed as a tuple $\{s, t_b, t_e\}$, where t_b/t_e is the check-in/out time
$\mathcal{G} = (\mathcal{V}, \mathcal{E}_{t_{start}, t_{end}})$	A temporal graph at the duration from t_{start} to t_{end}
$e = (u, v, \overrightarrow{t})$	A temporal edge connects vertices u and v among duration $\overrightarrow{t}$
$R(\overrightarrow{t}_1, \overrightarrow{t}_2)$	The overlap time of $\overrightarrow{t}_1$ and $\overrightarrow{t}_2$

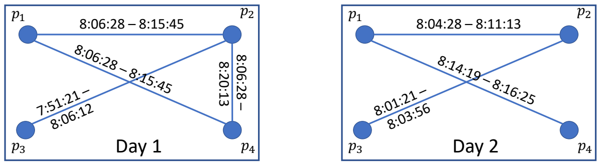


Fig. 3: A temporal graph generated according to Fig 1. Noting that we illustrate the temporal graph with two figures to show multiple edges between the same pairs of persons distinctly, not because they are two snapshots.

Challenge 3 - Splitting diffusion. Segmenting diffusion into snapshots complicates the accurate modeling of influence, particularly in nonlinear diffusion models. For example, if an exposure of 13.75 minutes between p_2 and p_4, as shown in Fig 1, is split by a snapshot, simple addition of the resulting probabilities from segments 9.28 and 4.46 minutes ($f(9.28) + f(4.46)$) is misleading for any nonlinear $f()$. Such nonlinear behaviors, common in various information propagation contexts like epidemic modeling [28], information spreading [29] and influence maximization [30], require careful normalization of influence among snapshots to maintain accuracy.

3.2 Preliminary

3.2.1 *Check-in/out Data*

Location. Given a location database $\mathcal{S}$, each location $s \in \mathcal{S}$ can be checked in/out by persons.
Movement. Given a movement database $\mathcal{R}$, where each movement $tr_i \in \mathcal{R}$ is a tuple $\{s, t_b, t_e\}$. Particularly, s_b/s_e denotes the check-in/out location, and t_b/t_e is the check-in/out time.
Person. Given a person database $\mathcal{P}$, where each person $p_n \in \mathcal{P}$ has a set of trips $p_n.Tr = \{tr_i, ..., tr_j\}$. For simplifying, let p_n^i denote the trip tr_i of p_n. We use p_n^i and $p_n.tr_i$ interchangeably.

3.2.2 *Temporal Graph*

The temporal graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}_{t_{start}, t_{end}})$, where from t_{start} to t_{end} finite period. A temporal edge

$e = (u, v, \overrightarrow{t})$ connects temporal vertices u and v, where $u, v \in \mathcal{V}$, and $\overrightarrow{t}$ is a duration from t_b to t_e. Each temporal edge e only exists in the period $e.\overrightarrow{t} = [t_b, t_e)$ from t_b to t_e. $\forall e \in \mathcal{E}$, $e.t_b \geq t_{start}$ and $e.t_e \leq t_{end}$. Noting that v does not have temporal features, as all vertices are stable. Therefore, we do not remove any vertex from the temporal graph if no edge is connected to it in a particular period. Let $R(\overrightarrow{t}_1, \overrightarrow{t}_2)$ denote the overlap time between two durations. For any two edges $e_1 = (u, v, \overrightarrow{t}_1)$ and $e_2 = (u, v, \overrightarrow{t}_2)$, there is a duration overlap, if and only if $R(\overrightarrow{t}_1, \overrightarrow{t}_2) > 0$.

It is easy to build a temporal graph based on any given movement database. Fig 3 illustrate a temporal graph generated from Fig 1. As every two vertices may have multiple temporal edges, for illustrating more clearly, we illustrate the temporal graph with two separate figures. We first build a graph $\mathcal{G}$ containing four vertices $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$, as we have four persons. For every pair of persons p_i and p_j, we build an edge (v_m, v_n) if and only if $\exists tr_i \in p_m.Tr, tr_j \in p_n.Tr$ such that $tr_i^m.s = tr_j^n.s$ and $R(tr_i^m, tr_j^n) > 0$.

3.3 Timeline Contact Graph

Definition 1. A Timeline Contact graph (TCG) is an ordered pair $G = (V, E)$, where V refers to a set of vertices, with each $v \in G$ representing a state of $v \in \mathcal{G}$. The set E comprises edges defined as $\{(v_i, v_j) | v_i, v_j \in V \; and \; v_i \neq v_j\}$, indicating direct connections between vertices.

This definition introduces TCG that addresses the challenges identified with snapshot-based methods. This structure enables us to capture all relevant close contacts effectively. Fig 4 illustrates a TCG generated based on Fig 1, which is intricately designed to capture all close contacts associated with check-ins and check-outs. In this graph, movements are denoted by light blue pairs, where the left and right vertices symbolize check-in and check-out points, respectively. Dotted arrows within or between these pairs represent self-connections, signifying the continuous infection status of an individual. Conversely, solid arrows depict close contacts between different individuals, highlighting potential new infection pathways.

3.4 Generate TCG based on a temporal graph

Algorithm 1 illustrates how to generate TCG based on a temporal graph. Fig 4 exemplifies a simple TCG generated based on the movement record shown in Fig 1.

We begin by initializing a TCG G and a hash-set $H < v, (v_1, v_2) >$. Here, a temporal vertex from $\mathcal{G}$ serves as the key, and the set of pairs (v_1, v_2) are the values (line 1.1). The first and last vertices for each individual are added as the start and end states, denoted as v_{begin} and v_{end}, respectively (orange dots in Fig 4) (lines 1.2-1.4). Next, each temporal edge contributes two pairs of vertices to the hash-set H (lines 1.5-1.9). Each pair corresponds to a single vertex of $\mathcal{G}$. Thus, two pairs are needed for the two vertices of the corresponding temporal edge (line 1.6). These pairs are added to H under the relevant $u' \in \mathcal{G}$ (line 1.7). We also assign the beginning and ending times to each vertex of a pair (line 1.8). In the subsequent step, pairs belonging to the same person are merged if their durations overlap (lines 1.10 - 1.13). Then, all pairs $(v_i, v_j) \in H(v')$ associated with one person are ordered in ascending order based on $v_i.t$ (lines 1.14 - 1.16). For an ordered pairs list of one person, denoted as $H(v') = \{(v_i, v_j)_1, \cdots, (v_i, v_j)_k\}$, we add self-connecting edges that connect all adjacent pairs from the end of one (i.e., v_j of $(v_i, v_j)_k \in H(v')$) to the beginning of the next (i.e., v_i of $(v_i, v_j)_{k+1} \in H(v')$), which are depicted as the dotted arrow between pairs in Fig 4. In the event of vertex activation or infection, the state is maintained probabilistically. Thus, we use self-connections to sustain the state within each pair and between pairs associated with an individual. The weights or probabilities assigned to these self-connection edges can be tailored to meet the specific needs of different applications.

Algorithm 1: Timeline Contact Graph Generation

```
    Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}_{t_{start}, t_{end}})$
    Output: $G$
1.1 Initialize $G$ and $H < v, (v_1, v_2) >$
1.2 for $v' \in \mathcal{V}$ do
1.3     Add the beginning/ending vertex $v_{begin}/v_{end}$ into $G$
1.4     Set $v_{begin}.t = t_{start}$, $v_{end}.t = t_{end}$
1.5 for $e = (u', v', \overrightarrow{t}) \in \mathcal{E}$ do
1.6     Add two pairs vertices $(v_i, v_j)$ and $(v_m, v_n)$ to $G$ presenting the beginning and ending time (i.e., $e.t_b$ and $e.t_e$) of $u'$ and $v'$, respectively.
1.7     $H.add(u', (v_i, v_j))$, $H.add(v', (v_m, v_n))$
1.8     Set $v_i.t = v_m.t = e.t_b$, $v_j.t = v_n.t = e.t_e$
1.9     Add edges $(v_i, v_j)$, $(v_m, v_n)$, $(v_i, v_m)$, $(v_m, v_i)$
1.10 for $v' \in H.key$ do
1.11     while $\exists (v_i, v_j), (v_m, v_n) \in H(v')$, such that $R((v_i.t, v_j.t), (v_m.t, v_n.t)) > 0$ do
1.12         Add a new pair $(v_p, v_q)$ and set $v_p.t = min(v_i.t, v_m.t)$, $v_q.t = max(v_j.t, v_n.t)$
1.13         Relocate all edge connecting to $v_i$ and $v_m$ to $v_p$
1.14 for $v' \in H.key$ do
1.15     Order $(v_i, v_j) \in H(v')$ ascendingly based on $v_i.t$
1.16     From $(v_i, v_j)_1$ to $(v_i, v_j)_{|H(v')|}$, add edges linking $v_j$ in previous one to $v_i$ in next one
```

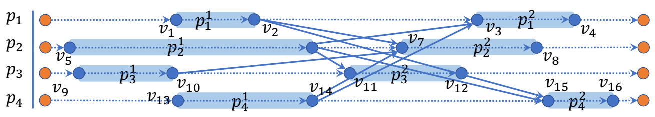


Fig. 4: TCG based on the movement record of Fig 1.

4 Diffusion On Timeline Contact Graph

Modelling influence diffusion in networks is an important challenge. Depending on the different types of applications, the influence diffuses differently. For instance, influence within social networks is spread via user interactions, such as following relationships, and propagated through actions like posts or shares. Similarly, in physical spaces, information spreads through direct interactions between individuals or vehicles. To examine TCG, we design two downstream tasks in the VI problem, which traditionally suffers the unaffordable memory cost when the graph structure changes rapidly. First, we need to define how the infection spread, and how to utilize TCG to model this spreading.

4.1 Infection Propagation Model

In this work, we adopt the Susceptible-Infected (SI) model to introduce the virus spreading, as it is the most classical disease model. Individuals are born into the simulation with no immunity (susceptible). Once infected, individuals stay infected and remain in contact with the susceptible population. To demonstrate that our work can take any infection probability $f()$ as an input, we adopt two different $f()$, the EC_{50} model and $InfTre$ model. In this part, we first introduce the EC_{50} model. Another one will be introduced in Section 7.

The EC_{50} model is a well-known model in the virus domain [31], [32], [33]. The infection probability increases when the duration increases. Formally, it is defined as $f(R) = 1/(1+(EC_{50}/R)^{\tau})$, where EC_{50} is a time threshold that a healthy person will have a 50% probability of being infected if she stays in the same POI closely with another infected person; τ is used to adjust the slope of $f()$.

Let $Pr(p_n^i, p_m^j) = f(R(p_n^i, p_m^j))$ be the infection probability between two persons after one contact (i.e., contact between i-th check-in of p_n and j-th check-in of p_m). Here, $R()$ is the exposure duration of one contact, defined as $R(p_n^i, p_m^j) = max((min(tr_i.t_e, tr_j.t_e) - max(tr_i.t_b, tr_j.t_b)), 0)$ if $tr_i.s = tr_j.s$, and $R(p_n^i, p_m^j) = 0$ otherwise; $f()$ is the virus diffusion model measures the infection probability based on the exposure duration, which is EC_{50} model at here. When an infected passenger p_n is making a trip tr_i, p_n will infect another passenger p_m with a probability $Pr(p_n^i, p_m^j)$. This process can be viewed as random events of flipping a coin with a probability $Pr(p_n^i, p_m^j)$.

We describe the following routine to simulate the infection propagation process.

Step 1. We initialize an empty priority queue of trip records, with all the trips sorted in ascending order of t_b.

Step 2. Given a set of infected passengers P, we assume the remaining passengers (i.e., $\mathcal{P} \backslash P$) are healthy. We push the first trips of all the infected passengers (i.e., $\cup_{p_n \in P} p_n^1$) into the queue.

Step 3. The queue pops a record tr_i iteratively. While checking in a POI via tr_i, p_n infects other passengers in the same POI with a probability $Pr(p_n^i, p_m^j)$.

Step 4. We push all the trips that have not been infected but will be infected by tr_i into the queue.

Step 5. Repeat steps 3 and 4 until the queue is empty.

The TCG has two advantages. *First*, by utilizing the TCG, the enormous discrete graph structures (e.g., snapshots) can be transformed into a single graph that contains all the information, which avoids the dilemma of whether losing the accuracy of changing edges among vertexes or suffering from the explosive amount of data. *Second*, according to the government's rules, the levels of strict intervention may change, and hence the definition of close contact varies. The TCG handles this scenario by involving a threshold for pushing infected trips. For example, for a loose intervention rule that contacts for more than 10 minutes will be counted as close contact, we only push trips where the infection probability is larger than $f(10 \ mins)$ into the queue.

4.1.1 *Incubation Period*

There is a special condition when we study virus diffusion, the incubation period, which makes the virus diffusion problem more complex than the influence maximization problem in the social network. When a person is infected, she/he might be not able to infect other healthy persons immediately because of the incubation period. We define the incubation period as γ hours. For instance, if p_n is infected at the check-in record tr_i, then p_n will be infectious and detectable from $tr_i.t_b + \gamma$ onward, i.e., γ hours after tr_i.

Accordingly, we need to modify Algorithm 1 to become compatible with the incubation period. Technically, instead of linking the two pairs based on their close contact in line 1.10 (i.e., v_i, v_m and v_m, v_i), we will connect v_i to the v_m of a pair that is behind the current one. For example, let γ be 12 hours. In Fig 4, instead of connecting pairs on the first day (i.e., $(v_1, v_2)_1$, $(v_5, v_6)_1$, $(v_9, v_{10})_1$ and $(v_{13}, v_{14})_1$), we actually connect the pairs in the first day to the corresponding pairs in the second day. For instance, instead of connecting $(v_1, v_2)_1$ and $(v_5, v_6)_1$, we connect $(v_1, v_2)_1$ to $(v_7, v_8)_2$ and $(v_5, v_6)_1$ to $(v_3, v_4)_2$. Consequently, if p_1 is infected at $(v_1, v_2)_1$ and successfully diffuse to p_2, p_2 will become activated at $(v_3, v_4)_2$ in second day, which is the 12 hours after p_1 and p_2 have a close contact.

5 <u>E</u>PIDEMIC <u>M</u>ITEGATING IN PUBLIC <u>A</u>REA

In this section, we explore the problem of <u>E</u>pidemic <u>M</u>itigation in Public <u>A</u>reas (EMA) from a defensive perspective, focusing on developing optimal checkpoint deployment strategies to effectively mitigate infection diffusion. The motivation is clear: by detecting and intercepting infected individuals, identified as p_i, before they access transit stations, we can prevent them from infecting others on public transport and further quarantine them to curb subsequent virus spread. To facilitate this, checkpoints are established at stations, enabling the implementation of preventative measures and identification of potentially infected passengers during boarding.

In the following, we first introduce the problem definition of EMA and prove that EMA is NP-hard. Next, based on our TCG, we propose a concept called *Contact Paths Tracing (CPT)* that is generated based on the latest trip of a passenger p_n. It presents a set of contacts that have the potential to infect the passenger p_n. Last, we propose our solution to address the EMA problem based on CPT.

5.1 Problem Definition of EMA

First, we formally define the checkpoint as follows:

Definition 2. Checkpoint: Given a checkpoint s and a passenger p_n who has a trip tr_i at s, p_n is allowed to board and alight if she is healthy and meanwhile passes the screening test implemented at s; p_n will be isolated if she is infected (and accordingly, the following trips of p_n ,$\{p_n^{i+1}, \cdots, p_n^{|Tr|}\}$, are removed from $\mathcal{T}$).

We introduce $W(s_n) = \cup_{tr_i \in \partial s_n} I(p_n^i)$ to denote the coverage of the checkpoint s_n, where ∂s_n refers to a set of trips such that $\forall tr_i \in \partial s_n$, $tr_i.s = s_n$, and $I(p_n^i)$ denotes all close contacts of p_n^i. Now, we are ready to formally present the problem definition for EMA.

Definition 3. The EMA Problem: Given a passenger dataset $\mathcal{P}$, a trip dataset $\mathcal{T}$, a checkpoint candidates dataset $\mathcal{S}$, a budget k, and an incubation threshold γ, EMA is to find a set of stations $S \subseteq \mathcal{S}$ to set up checkpoints so as to maximize the coverage of checkpoints.

$$S = \underset{|S| \leq k}{\operatorname{argmax}} \; W^*_{\mathcal{P},\mathcal{T},\gamma}(S) = | \cup_{s_n \in S} W(s_n)|. \qquad (1)$$

A straightforward solution is to select the top-k stations with the largest number of trips. However, a station with a larger number of trips does not necessarily have a higher vulnerability. For example, considering a POI with many check-in records that come from a fixed crowd with a constant movement pattern, whereas another POI with fewer check-in records from a group of highly mobile persons, the latter POI is more vulnerable. The experimental results to be reported in Section 7.5 confirm this observation. As compared to the checkpoint deployment strategy that always picks the top-k stations (based on the volume of trips), our solution can improve the coverage by at most three times.

Theoretically speaking, an optimal checkpoint deployment strategy should well mitigate the virus from spreading among all situations, which is equivalent to minimizing the maximum number of finally infected passengers no matter who the initially infected passengers are [34], [35]. However, to the best of our knowledge, no existing work could be applied to address our problem because of two main reasons: (1) They are unable to handle a highly-frequent changing graph structure caused by people's movements. (2) The scale of the trips considered by existing work is significantly smaller than ours. For instance, existing work [35] proposes a dynamic index data structure, namely Weighted Cascade (WC) model. It is designed for influence analysis when some of the edges or nodes are removed from the graph. With a dataset that is one order of magnitude smaller, WC needs more than one hour to build the dynamic index data structure and update the index after removing edges/nodes in each iteration. In contrast, our solution costs less than 300 seconds in the default settings for finding 50 stations as checkpoints to eliminate all infected passengers.

5.2 Hardness of EMA

Theorem 1. The EMA problem is NP-hard.

Proof 2. We prove it by reducing the Minimum k-union Problem (MinKU) [36]. In the MinKU problem, given a collection of set $\mathcal{S}' = \{s'_1, s'_2, \cdots, s'_m\}$ where each set s'_i is a subset of a given ground set $\mathcal{P}'$, an integer $0 \leq k \leq m$ and $0 \leq \tau \leq |\mathcal{P}'|$, it aims to find $S' \subseteq \mathcal{S}'$, such that $|S'| = k$ and $| \cup_{s'_i \in S'} s'_i| \leq \tau$. We map the MinKU problem to the EMA problem with the following process: (1) In EMA problem, let each person has two check-ins only; let $\gamma = 0$; let $f(R()) = 100\%$ if $R() > 0$. (2) We map each check-in p in EMA to each element $p' \in \mathcal{P}'$ in MinKU, and map each candidate location $s \in \mathcal{S}$ in EMA to each set $s' \in \mathcal{S}'$ in MinKU. (3) For any check-in p_n^i, if $Pr(p_n^i) > 0$, we connect p_n to the location $p_n^i.s$, and connect the corresponding p'_n to s'. Clearly the mapping can be done in polynomial time. Consequently, EMA is equivalent to deciding, given k and τ, whether MinKU problem can find k subsets S' such that $| \cup_{s'_i \in S'} s'_i| \leq \tau$. If the answer is Yes, then EMA can find $|\mathcal{S}| - k$ locations from S that saves $|\mathcal{P}| - | \cup_{s_i \in S} S_i|/2 \geq |\mathcal{P}| - \tau/2$ persons. This is because, for a checkpoint, people who connect to it will be healthy as they only have two check-ins. Therefore, for a set of locations that are not checkpoints S, the maximum number of infected persons is $| \cup_{S_i \in S'} S_i|/2$. Since MinKU is NP-complete, the decision problem of EMA is NP-complete. Hence, the optimization problem of EMA is NP-hard.

5.3 CPT-based Station Selection (CPT-SS)

According to Definition 3, given a set of checkpoints S, we have $W(S)$ that captures the set of people who can be covered (i.e., saved from being infected) if checkpoints are placed at the set S. Accordingly, we define $W(S|s_i) = |W(S \cup \{s_i\})| - |W(S)|$ as the marginal gain of adding s_i into S. A naive greedy-based solution will iteratively select a POI with the maximum $W(S|s_i)$. However, the challenge is how to efficiently evaluate $W(S|s_i)$. Hence, we introduce the CPT-based POI Selection (CPT-SS).

Definition 4. Given a passenger $p_n \in \mathcal{P}$, the *Contact Paths Tracing* (CPT) of p_n is the set of trips that can infect p_n.

Intuitively, a Contact Paths Tracing (CPT) is generated based on the latest trip made by a passenger $p_n \in \mathcal{P}$ and it includes all the trips tr that have a certain probability ($> 0\%$) of infecting the passenger p_n based on the given infectious model. CPT is generated as follows:

Generation of one CPT.
Step 1. Randomly select $p_n \in \mathcal{P}$; initialize an empty queue Q and an empty CPT;
Step 2. Add tr_i into both Q and CPT where $tr_i \in p_n.Tr$ refers to the latest trip of p_n;
Step 3. Pop the top trip tr_i out of Q;
Step 4. For each incoming edge of tr_i (from node p_m^j), flip a coin with $Pr(p_n^i, p_m^j)$ probability;
Step 5. If true and $tr_j \notin CPT$, add tr_j into both Q and CPT.
Step 6. Repeat steps 3 to 5 until Q is empty.

According to Definition 4, given a CPT of p_n, all trips in this CPT can infect p_n. It has been proven that the probability of CPT overlapping with any set of trips T is equivalent to the probability of T infecting p_n [37], [38], [39]. Thus, the number of times that a trip tr_i appears in the CPT of different passengers indicates the potential impact of tr_i in terms of the capability to spread the virus, e.g., the trip appearing in the largest number of CPT contributes the most to the spread of the virus and hence shall be isolated first. The last question is how many CPT we need to accurately evaluate $W(S|s_i)$. Let θ be the number of CPT sets that we need, $\mathcal{F}(S)$ be the fraction of CPT sets over all CPT sets covered by a given set of checkpoints S. The existing work [39] shows that, $n\mathcal{F}(S)$ is an accurate estimator of $W(S)$, when θ is sufficiently large, i.e., $\theta \geq n(8 + 2\epsilon)(l \log n + \log \binom{n}{k} + \log 2)/(OPT \cdot \epsilon^2)$, where n is the number of trips and ϵ is the approximate ratio. OPT can be estimated by $n(1 - (1 - w(CPT))^k)$, where $w(CPT)$ is the number of edges of CPT. For more details of deriving θ, please refer to the work [39].

Algorithm 2: CPT-based Station Selection (CPT-SS)

```
Input: Checkpoint number $k$, Timeline Graph $G$, $\theta$,
       Location database $\mathcal{S}$
Output: Checkpoint set $S$
2.1 Based on $G$, generate $\theta$ CPT set into $\mathcal{R}$
2.2 while $|S| \leq k$ do
2.3     Select $s_i \in \mathcal{S}$ that covers the most CPT sets in $\mathcal{R}$
2.4     $S \leftarrow S \cup \{s_i\}$
2.5     $\mathcal{S} \leftarrow \mathcal{S} \backslash \{s_i\}$
2.6     Remove all CPT sets covered by $s_i$ from $\mathcal{R}$
```

Algorithm 2 presents the pseudo-code of CPT-SS. To minimize the number of infected passengers, we need to find stations with the most extensive coverage, i.e., block the most dangerous trips that may infect a large number of passengers. Since the most dangerous trip is expected to appear in the greatest number of CPT sets, we generate CPT sets and then find checkpoints that can detect the largest number of dangerous trips. We first generate θ CPT sets as $\mathcal{R}$ based on the timeline contact graph G (Line 2.1). In the selection loop, we select a station s_i that covers the most CPT sets in $\mathcal{R}$ (Lines 2.3-2.5) and remove all CPT sets that are covered by s_i from $\mathcal{R}$ (Line 2.6). The loop terminates when $|S| = k$ and S is returned.

6 Epidemic Maximized Spread in Public Area

In this section, from the offensive perspective, we study the ESA problem that aims to find k "super-spreaders", the passengers who will spread the virus the most, thus possibly leading to a pandemic. This helps the government to evaluate the robustness of checkpoint deployment strategies in the extreme case that the virus is spread widely.

6.1 Problem Definition of ESA

We begin by introducing the Epidemic Maximized Spread in Public Area (ESA) problem. The primary objective here is to assess the efficacy of a given checkpoint deployment strategy under the worst-case scenario. In specific terms, given a certain checkpoint deployment strategy, we aim to identify a set of k passengers who, if initially infected, could potentially infect the maximum number of other passengers.

Taking checkpoints into account, the infection probability between two trips can be expressed as follows:

$$Pr(p_n^i, p_m^j) = \begin{cases} f(R(\cdot)) & otherwise \\ 0 & if \;\; tr_i.s_b \in S \vee tr_j.s_b \in S \end{cases} \quad (2)$$

where S denotes a set of checkpoints. Conceptually, when an infected passenger attempts to pass a checkpoint, all of her remaining trips are immediately suspended. As a result, these trips are no longer taken into account in the simulation of virus diffusion. Building on Equation 2, we formally present the ESA problem as follows:

Definition 5. The ESA Problem: Given a passenger dataset $\mathcal{P}$, a trip dataset $\mathcal{T}$, a set of infected passengers $P \in \mathcal{P}$, a set of checkpoints S, a budget k, and an incubation period γ, ESA is to find at most k passengers $P \in \mathcal{P}$ such that P can maximize the following equation:

$$P = \text{argmax}_{|P| \leq k} \; I_{\mathcal{P},\mathcal{T},\gamma,S}(P). \quad (3)$$

6.2 Hardness of ESA

Theorem 3. The ESA problem is NP-hard.

Proof 4. We prove it by reducing the Set Cover problem to the ESA problem. In the Set Cover problem, given a collection of sets $S' = \{s'_1, s'_2, \cdots, s'_m\}$ where each set s'_i is a subset of a given ground set $\mathcal{P}' = \{p'_1, ..., p'_n\}$, it aims to find whether there exist k of the subsets whose union is equal to $\mathcal{P}'$. We reduce the Set Cover problem to ESA with the following process: (1) We map each element $p'_i \in \mathcal{P}'$ in the Set Cover problem to each person $p_i \in \mathcal{P}$ in the ESA problem. (2) We map each set $s'_i \in S'$ in the Set Cover Problem to a set of persons $\{p_i, \cdots, p_j\} = I(p_n) \subseteq \mathcal{P}$ in the ESA problem, where $p'_m \in s'_n$ maps to the person $p_m \in I(p_n)$ that will be infected by p_n. Consequently, the Set Cover problem is equivalent to deciding whether there are the k-sized set of initially infected persons $P \in \mathcal{P}$, such that $I(P) = |\mathcal{P}|$. Since the Set Cover problem is NP-complete, the decision problem of EMA is NP-complete. Hence, the optimization problem of EMA is NP-hard.

6.3 CPT-based Passengers Selection (CPT-PS)

Now, we are ready to present the solution, CPT-based Passengers Selection (CPT-PS), for the ESA problem. The main objective is to find the set of super-spreaders, with each being able to infect many healthy passengers. Similar to the EMA problem, a super-spreader is a passenger whose trips appear in the greatest number of CPT sets. Since the solution of ESA is similar to that of EMA, we omit the pseudocode but highlight the key steps and differences only. Based on the timeline contact graph G, we initialize and generate θ number of CPT sets as $\mathcal{R}$ first. Then, a passenger p_i that covers the largest fraction of CPT sets is added into P; and all CPT sets that are covered by p_i are removed from $\mathcal{R}$. The loop terminates when $|P| = k$, and P is returned. It is worth noting that there is one difference between ESA and EMA; the process of generating a CPT. According to Definition 2, if an infected passenger p_n checks-in at s via tr_i, p_n will be isolated and $p_n.Tr = \{tr_i, \cdots, tr_{|Tr|}\}$ will be removed from $\mathcal{T}$). Moreover, the CPT of p_n is the set of trips in $\mathcal{T}$ that can infect $p_n^{|Tr|}$. Therefore, an infected trip is not able to infect p_n for the CPT of p_n. As a result, in the process of generating CPT, we change Step 4 to: for each incoming edge of tr_i, if $tr_i.s \notin S$, flip a coin with $Pr(p_n^i, p_m^j)$ probability, where S is the set of checkpoints.

7 Experiment

In this section, we first introduce the datasets in Section 7.1. Then, we introduce two infection probability measurements that are commonly used in the epidemic field and experiment setup in Section 7.2. Finally, we evaluate the proposed graph structure in Section 7.3, evaluate how well the various algorithms could mitigate viruses for EMA problem in Section 7.4, and the robustness of each algorithm for ESA problem in Section 7.5, respectively. All codes are available online[1] for reproducibility.

TABLE 2: Dataset Information

Dataset	Trip	Passenger	Station
EZ-LINK	2×10^6	5.2×10^4	302
BART	4.2×10^6	6.8×10^5	50

TABLE 3: Parameter settings

k	25, **50**, 75
γ (Hours)	24, **72**, 120
EC_{50} (Seconds)	1800, **3600**, 7200
Period (Days)	**14**, 21, 28
P_s	0.01%, 0.05%, **0.1%**, 0.2%, 0.5%, 1%

7.1 Datasets

We conduct experiments based on EZ-LINK (default) and BART datasets. Details on this dataset are summarized in Table 2. The EZ-LINK data contains comprehensive movement records within a city-scale public transport system, as close contact and high user density offer ideal conditions for studying virus transmission dynamics, which are crucial for tracing infection pathways. Notably, unlike many larger datasets where each movement record is anonymous, EZ-LINK provides individual identification, enabling the tracing of specific infected individuals over time and accurately assessing the spread of infection through direct contacts.

Due to privacy constraints and the rarity of similar datasets with individual tracking, we synthesize the BART dataset based on Bart data from the San Francisco Bay Area Rapid Transit District (BART) system[2]. The original BART data only contains orientation destination (OD) matrix. To synthesize the BART dataset based on EZ-LINK data, we first extracted the trip frequency distribution per passenger from EZ-LINK. We then calculate a scale factor based on average numbers of trips per passenger for both datasets, and use it to adjust the trip frequencies of BART. Next, trips are assigned to unique passenger IDs for the RABT dataset. For each trip, select origin and destination stations based on the probabilities from the RABT OD matrix, simulating a realistic spatial distribution of trips. Finally, we randomly generate exact boarding and alighting times, ensuring each trip reflects realistic variations in travel times.

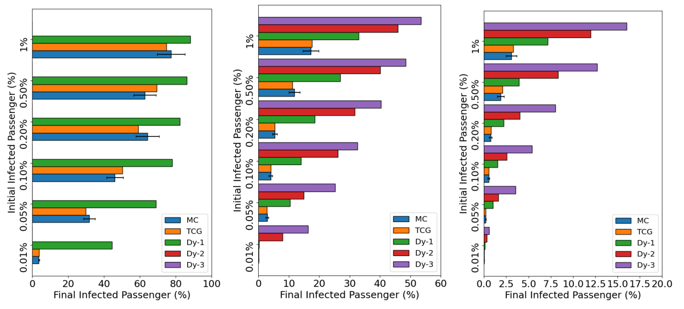
7.2 Experiment Setup

All key parameters are summarized in Table 3. In each set of experiments, we vary only one parameter and set the rest of the parameters to their default values highlighted in bold.

Incubation period γ. γ is a period between when a passenger is able to infect others and when she is infected. It varies from disease to disease. For instance, the average γ of Influenza is two days [40], and that of COVID-19 is five days [41], [42]. Hence, we set γ to 24 hours (1 day), 72 hours (3 days), and 120 hours (5 days), to represent short, normal, and long incubation periods to cover most cases.

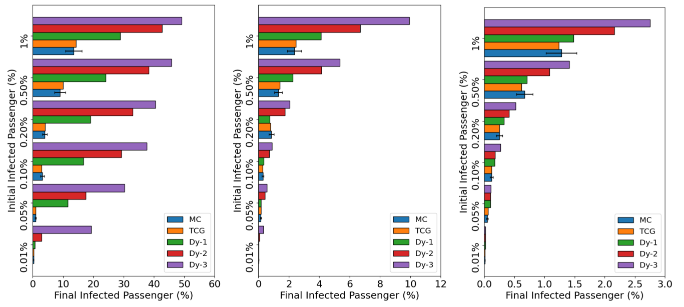
Percentage of Initially Infected Passengers P_s. Since a fixed number is meaningless without considering the total number of passengers, we set P_s as the percentage of the passengers that are initially infected. Then, the number of

1. https://github.com/e1qjmis5/VirusDiffusionCode
2. https://www.bart.gov/about



(a) EC_{50} = 1800s (b) EC_{50} = 3600s (c) EC_{50} = 5400s

Fig. 5: Graph competition of varying EC_{50} (Seconds) when the incubation period $\gamma = 72$ hours within 14 days



(a) γ = 24 hours (b) γ = 72 hours (c) γ = 120 hours

Fig. 6: Graph competition of varying γ when the incubation period $EC_{50} = 3600$ seconds within 14 days

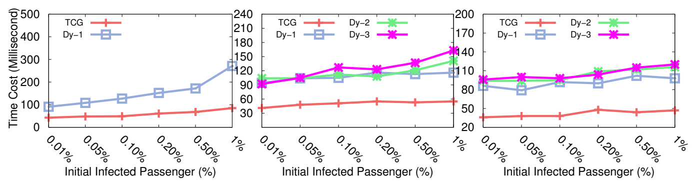
initially infected passengers will be $P_s \times |\mathcal{P}|$, where $|\mathcal{P}|$ is the total number of passengers. In each experiment, we set P_s to 0.01%, 0.05%, 0.1%, 0.2%, 0.5%, and 1%, which represent the cases from a few initially infected persons to a small group of initially infected persons.

Infection Probability Measurement. To demonstrate that our work can take any infection probability $f()$ as an input, we adopt two different models with unique $f()$ for illustration. **(1)** EC_{50} model. It is widely used in measuring the concentration required for the drug, antibody, or toxicant, to measure infection probability [31], [32], [33]. We set $f(R) = 1/(1 + (EC_{50}/R)^{\tau})$, where EC_{50} indicates the duration of the close contact that will lead to a 50% probability of being infected; τ is used to adjust the slope of $f()$. We name it EC_{50} model, and set $\tau = 3$. **(2)** $InfTre$ model. $InfTre$ model is based on a threshold of exposure duration. Specifically, given a threshold of exposure duration, $f() = 100\%$ if $R()$ is not shorter than the threshold, and $f() = 0\%$ otherwise [43]. Please refer to the technical report for the experimental results of $InfTre$.

Performance Metrics. *The total number of infected passengers* and *the running time* are employed as the effectiveness metric and the efficiency metric respectively. For each experiment, we report the average result of 20 runs.

Compared Methods. The comparison is conducted from two perspectives, w.r.t. two different goals presented earlier.

1) **Task 1. Tracing Infection Diffusion:** We compare our TCG with the dynamic graph [18], namely Dy-x. Here, x refers to the number of hours between two consecutive snapshots (i.e., a smaller x value indicates that more



(a) EC_{50} = 1800s (b) EC_{50} = 3600s (c) EC_{50} = 5400s

Fig. 7: Running time of varying EC_{50} (Seconds) when the incubation period $\gamma = 72$ hours within 14 days

snapshots will be generated based on a shorter interval). The edge between two close contact passengers is added according to the probability $f(R)$, where R is the shortest time that v_i or v_j stays at the location. For this task, we employ the Monte Carlo (MC) method to provide the ground truth, enabling us to determine which baseline is more accurate in terms of simulating virus diffusion. The detail of the MC method is presented in Section 7.3.

2) **Task 2. High-risk Detection:** (1) For the EMA problem, we compare four deployment strategies: I) Tk_1: Top-k checkpoints covering the largest number of trips; Tk_2: Top-k checkpoints covering the largest number of 100 super-spreaders found by ESA, II) CPT-SS: the CPT-based checkpoints selection (Section 5.3), and iii) Eig: the eigenvalue-based solution presented in Section 2. In each iteration, we select a node that maximizes the decrease of the first eigenvalue. (2) For the ESA problem, we compare three initial infection settings: I) $Rand$: P_s randomly-selected passengers, II) $Max_{1/2}$: P_s super-spreaders identified by CPT-PS (Section 6.3), and III) Max_3: Top-k check-in passengers.

7.3 Evaluation of The Timeline Contact Graph

The tracing of infection diffusion is compared between our TCG and the dynamic graph approach [18], referred to as Dy-x, where x signifies the hours between two successive snapshots. Thus, a smaller x value indicates a faster snapshot updating. The weight of the edge in the graph represents the infection probability, measured as $f(R)$, where R is the shortest duration that either v_i or v_j stays at a specific location. Since there is no ground truth data to evaluate the accuracy of different simulating methods, we introduce the MC method to simulate the ground truth results. The detail of MC is shown as follows:

Step 1. Initialize an empty priority queue Q, ordered by the check-in time t_b of trips.
Step 2. Add all trips from the initial infectious persons to Q.
Step 3. Pop a trip p_n^i from Q and identify its close contacts.
Step 4. For each close contact, calculate $Pr(p_n^i, p_m^j)$. If a random event with this probability occurs, add the corresponding trip p_m^j to Q.
Step 5. Repeat Steps 3-4 until Q is empty.
Step 6. Count the number of unique infected persons.

We first compare the accuracy and efficiency of our TCG and the dynamic graph in modelling infection diffusion without checkpoints under default settings with randomly selected initial infections. We vary two parameters, transmissibility EC_{50} and incubation period γ. The lower the EC_{50} and γ, the faster the disease outbreak. Thus, a small EC_{50} and γ require precise modelling.

Figs 5 - 6 show the experimental results of varying EC_{50} and γ. We have two main observations.

First, TCG models virus diffusion more accurately. For instance, in Fig 6b, Dy-1, Dy-2, and Dy-3 overestimate the number of infections by three, nine, and twelve times compared to TCG, respectively. This is because snapshot-based solutions fail to capture accurate close contacts.

Second, the advantage of TCG becomes more significant when EC_{50} becomes smaller. For example, when $EC_{50} = 1,800$s, in terms of the number of infected passengers, TCG outperforms Dy-1 by at most 20 times as shown in Fig 5a. It is because the small EC_{50} will lead to a high infection probability and hence cause an explosive increase in the number of infected passengers and make the dynamic graph (based on snapshots) mistakenly consider non-contact passengers as having contact. Therefore, it is easy for a dynamic graph to overestimate the number of infected passengers.

Last, the experimental results reported in Fig 6 demonstrate a similar trend. In the case of Dy-2 and Dy-3 in Fig 5a and Fig 6a, we have excluded the results, as both of them exceeded the memory limits. This was a consequence of their generation of an extensive number of edges, primarily attributable to inaccurate contact recording.

We also evaluate the efficiency of different methods and the results are reported in Fig 7. Results of Dy-2 and Dy-3 are not reported, as they are out of memory when generating a massive number of edges due to inaccurate contact recording. Consistent with our expectation, TCG requires significantly less time, as compared to dynamic graph. One interesting point is that, with a larger EC_{50}, the increase of P_s shows a smaller impact on the running time. It is because a large EC_{50} can avoid the explosive growth of the final infection number. Therefore, the larger the EC_{50} is, the smaller the impact of P_s on running time is.

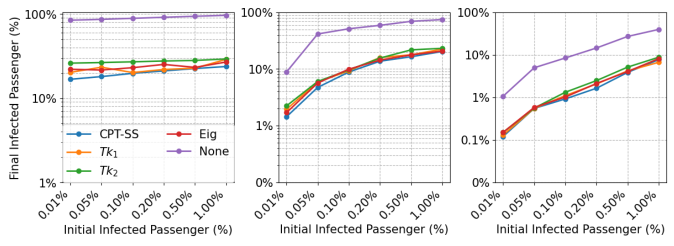
7.4 Evaluation of Checkpoints Deployment Strategies to Address the EMA Problem

7.4.1 Effectiveness Study

In this part, we first evaluate *how well all algorithms can mitigate the infection diffusion under different settings of γ and EC_{50}* on EZ-LINK dataset. Specifically, we simulate three different categories corresponding to the virus of low/medium/high danger, which leads to nine scenarios. For each scenario, we vary the percentage of initially infected passengers, P_s, from 0.01% to 1%, to simulate how serious the epidemic is. We additionally involve "None" as the none-checkpoint strategy to show the worst case. Figs 8-10 show the experimental results where we make five main observations. Lastly, we conduct experiments on BART dataset.

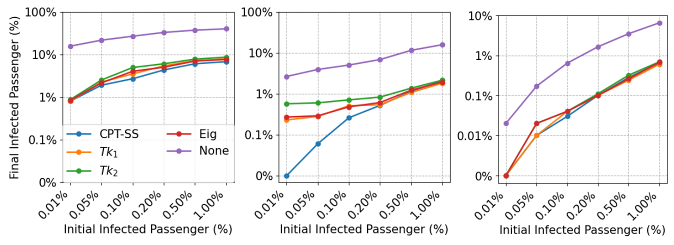
First, with a larger P_s, all the algorithms result in a larger number of finally infected passengers. It is clear that early adoption of the measures to control the virus spread could help mitigate the epidemic. Moreover, when both γ and EC_{50} are small, the virus is highly contagious, and hence a few infected passengers will lead to a huge number of infected passengers.

Second, with a fixed EC_{50}, the smaller the incubation period γ is, the bigger the number of finally infected passengers will be. This is because when γ is small, a passenger will soon be able to infect others after she is infected.



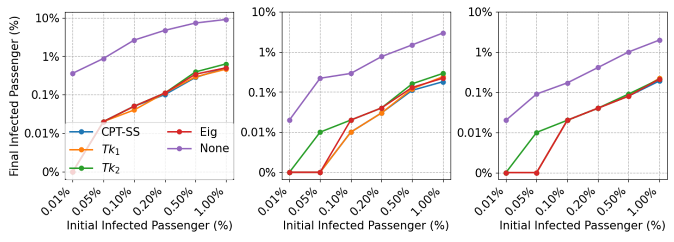
(a) γ = 24 hours (b) γ = 72 hours (c) γ = 120 hours

Fig. 8: Infection of varying incubation γ (EC_{50} = 1800s)



(a) γ = 24 hours (b) γ = 72 hours (c) γ = 120 hours

Fig. 9: Infection of varying incubation γ (EC_{50} = 3600s)



(a) γ = 24 hours (b) γ = 72 hours (c) γ = 120 hours

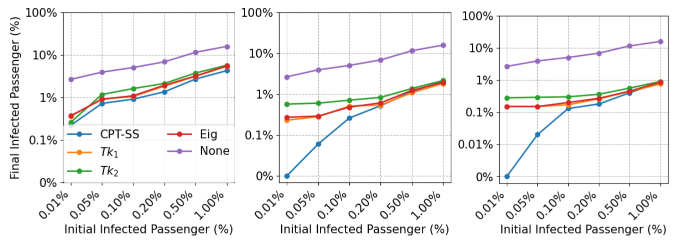
Fig. 10: Infection of varying incubation γ (EC_{50} = 5400s)

Third, with a fixed γ, when EC_{50} becomes larger, the numbers of finally infected passengers under all the algorithms become smaller. This is because a larger EC_{50} indicates that healthy passengers will be less likely infected.

Fourth, CPT-SS significantly outperforms all baselines, achieving results that are approximately three times better than both Tk_1 and Tk_2, and ten times better than Eig. The underwhelming performance of Tk_1 indicates that merely setting up checkpoints based on high passenger traffic is ineffective. This suggests that blocking the top-k stations with the highest passenger flow or targeting top-k 'super-spreaders' with the most infectiousness is an inefficient strategy, creating vulnerabilities in the public transportation network and necessitating a focus on systemic resilience.

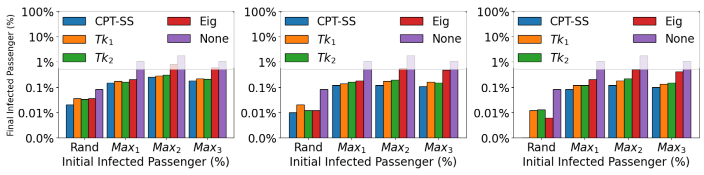
Fifth, the effectiveness of CPT-SS is pronounced in scenarios where the epidemic is more severe (i.e., Figure 8a). This is because highly contagious viruses require careful planning in the deployment of checkpoints to prevent widespread outbreaks. Conversely, when the virus's spreading ability is weakened (larger γ and EC_{50}), all algorithms show the capacity to control the epidemic effectively.

Fig 11 illustrates how varying the number of checkpoints k affects the total number of infected passengers. We observe three main points: First, an increase in checkpoints leads to a reduction in the number of infected passengers. Second, with a higher number of checkpoints ($k \geq 50$),



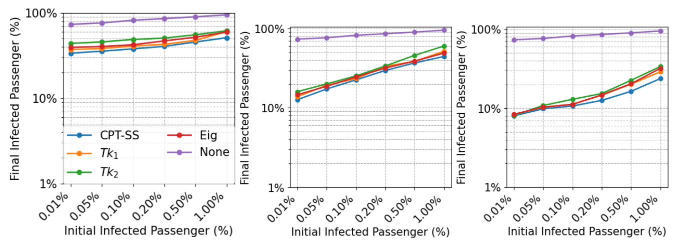
(a) $k = 25$ (b) $k = 50$ (c) $k = 75$

Fig. 11: Infection of varying the checkpoint number k



(a) $k = 25$ (b) $k = 50$ (c) $k = 75$

Fig. 12: Robustness evaluation



(a) γ = 24 hours (b) γ = 72 hours (c) γ = 120 hours

Fig. 13: Infection of varying the checkpoint number k using BART dataset

the benefits of CPT-SS become more pronounced. This is because checkpoints can only limit virus spread to a certain extent. At higher k, since high-traffic stations are often in urban centres with overlapping coverage, less busy areas might be overlooked. Third, when the initially infected group is smaller (i.e., $P_s \leq 0.5\%$), the advantages of CPT-SS are more significant. With fewer initial infections, precise control can almost completely halt virus spread. If checkpoints are not strategically placed, however, and resources are wasted on redundant coverage, some areas may be neglected, leading to more infections.

Experiments using BART dataset. Fig 13 depicts the efficacy of various algorithms at managing virus spread with checkpoints, comparing results from BART and EZ-LINK datasets. We observe two main points: First, even with 20 checkpoints, which represent 40% of the total stations, performance is significantly lower compared to EZ-LINK due to the BART dataset's higher passenger density. Second, despite our solution outperforming all baselines, over 25% of passengers are infected with 20 checkpoints at $P_s = 1\%$, indicating the potential need for city-wide lockdowns in dense urban areas.

7.4.2 *Efficiency Study*

Table 4 shows the efficiency results. We have two observations. First, the increasing EC_{50} will increase the time-cost. According to existing studies [37], [38], [39], a more

TABLE 4: Efficiency Study of CPT-SS method

	EC_{50}		
k	1800	3600	5400
25	72.17 Seconds	179.73 Seconds	191.5 Seconds
50	61.26 Seconds	133.8 Seconds	143.84 Seconds
75	66.49 Seconds	127.85 Seconds	139.36 Seconds

negligible infection probability leads to a smaller average width of CPT (contact paths tracing). Theoretically, to maintain the same error bound, more samples are needed to trade off the small width of CPT, which in turn requests a larger number of CPT to evaluate each trip's infection range. Consequently, it is more time-consuming. Second, the number of checkpoints only slightly affects the time-cost of CPT-SS. The reason is that, the more stations are selected, the less number of CPT is needed. Intuitively, when the number of checkpoints increases, we do not need a highly accurate evaluation of the coverage of checkpoints since a larger number will "remedy" the accurate loss.

7.5 Robustness Evaluation for the ESA Problem

As outlined in Section 3, we identify 100 'super-spreaders' - a small group of passengers who could potentially cause a pandemic. We then test the robustness of checkpoint deployment strategies by setting these 'super-spreaders' as the initial carriers.

We have four strategies to illustrate the impact of 'super-spreaders'. Specifically, Max_1 and Max_2, based on Algorithm 2. Max_1 generates 'super-spreaders' without knowing the checkpoint locations, simulating a general worst-case scenario. Conversely, Max_2 generates 'super-spreaders' with the knowledge of the checkpoint locations, simulating a targeted attack. Max_3 finds the infected individuals who have checked in the most at non-checkpoint locations as 'super-spreaders'. We also implement a random selection strategy, $Rand$. The results, shown in Fig 12, lead to three observations.

First, among all solutions, 'super-spreaders' indeed have a higher risk of causing a pandemic, as evidenced by the fact that the number of finally infected passengers in the $Rand$ strategy is smaller than that in the Max methods.

Second, the effectiveness of our algorithm increases with the number of checkpoints. In situations where checkpoints are scarce, we cannot protect the entire population even when we successfully locate the 'super-spreaders'. However, as the number of checkpoints increases, our algorithm becomes more effective at deploying those checkpoints to block the 'super-spreaders'. Specifically, when comparing the results from $k = 25$ to 75, our solution outperforms top-k and Eig by about 18.9%, 31.2%, 44.8%, and 52.8%, 65.1%, 64.3%, respectively.

Third, all solutions show improved performance in Max_3 in terms of controlling the final number of infected passengers. This can be attributed to two reasons. First, passengers who have checked in the most at non-checkpoint locations may concentrate in the same locations, making it easier to isolate and control the spread of infection. Second, Max_3 uses a linear equation (Top-k) to evaluate the infectious probability. Despite these challenges, our algorithm still manages to outperform others, demonstrating its robustness and effectiveness.

8 CONCLUSION

In this work, we study how to trace and mitigate infection diffusion according to fine-grained users' movement records. We introduce an infection diffusion simulator based on TCG to model the infection diffusion according to users' movements. Two downstream tasks ESA and EMA are proposed to illustrate how our work assists governments in mitigating the epidemic from the defensive and offensive perspectives. Finally, we conduct experiments using two real-world datasets from public transport systems in Singapore and San Francisco, containing millions of trip records. The results demonstrate that our TCG graph evaluates the infection diffusion model more accurately compared to a snapshot-based dynamic graph. Additionally, in terms of checkpoint deployment strategies, our solution consistently outperforms the baselines for randomly infected passengers and super-spreaders across various virus infectiousness levels and different periods.

REFERENCES

[1] S. Chang, E. Pierson, P. W. Koh, J. Gerardin, B. Redbird, D. Grusky, and J. Leskovec, "Mobility network models of covid-19 explain inequities and inform reopening," *Nature*, pp. 1–8, 2020.

[2] S. Ghamizi, R. Rwemalika, M. Cordy, L. Veiber, T. F. Bissyandé, M. Papadakis, J. Klein, and Y. L. Traon, "Data-driven simulation and optimization for covid-19 exit strategies," in *SIGMOD*. ACM, 2020, pp. 3434–3442.

[3] Q. Hao, L. Chen, F. Xu, and Y. Li, "Understanding the urban pandemic spreading of COVID-19 with real world mobility data," in *SIGMOD*. ACM, 2020, pp. 3485–3492.

[4] B. S. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter, "Deterministic broadcasting in ad hoc radio networks," *Distributed Comput.*, vol. 15, no. 1, pp. 27–38, 2002.

[5] G. V. Rossi, Z. Fan, W. H. Chin, and K. K. Leung, "Stable clustering for ad-hoc vehicle networking," in *2017 IEEE Wireless Communications and Networking Conference, WCNC 2017, San Francisco, CA, USA, March 19-22, 2017*. IEEE, 2017, pp. 1–6.

[6] K. D. Singh, P. Rawat, and J. Bonnin, "Cognitive radio for vehicular ad hoc networks (cr-vanets): approaches and challenges," *EURASIP J. Wirel. Commun. Netw.*, vol. 2014, p. 49, 2014.

[7] M. Then, T. Kersten, S. Günnemann, A. Kemper, and T. Neumann, "Automatic algorithm transformation for efficient multi-snapshot analytics on temporal graphs," *Proc. VLDB Endow.*, vol. 10, no. 8, pp. 877–888, 2017.

[8] J. Gao, C. Zhou, and J. X. Yu, "Toward continuous pattern detection over evolving large graph with snapshot isolation," *VLDB J.*, vol. 25, no. 2, pp. 269–290, 2016.

[9] B. Wilder, S. Suen, and M. Tambe, "Preventing infectious disease in dynamic populations under uncertainty," in *AAAI*. AAAI Press, 2018, pp. 841–848.

[10] H. W. Hethcote, "The mathematics of infectious diseases," *SIAM Rev.*, vol. 42, no. 4, pp. 599–653, 2000.

[11] M. Minutoli, P. Sambaturu, M. Halappanavar, A. Tumeo, A. Kalyanaraman, and A. Vullikanti, "Preempt: scalable epidemic interventions using submodular optimization on multi-gpu systems," in *2020 SC20*. IEEE Computer Society, 2020, pp. 765–779.

[12] J. Liu, P. E. Paré, A. Nedic, C. Y. Tang, C. L. Beck, and T. Basar, "Analysis and control of a continuous-time bi-virus model," *IEEE Trans. Autom. Control.*, vol. 64, no. 12, pp. 4891–4906, 2019.

[13] S. Gracy, P. E. Paré, H. Sandberg, and K. H. Johansson, "Analysis and distributed control of periodic epidemic processes," *IEEE Trans. Control. Netw. Syst.*, vol. 8, no. 1, pp. 123–134, 2021.

[14] H. Li, C. Xia, T. Wang, S. Wen, C. Chen, and Y. Xiang, "Capturing dynamics of information diffusion in SNS: A survey of methodology and techniques," *ACM Comput. Surv.*, vol. 55, no. 2, pp. 22:1–22:51, 2023.

[15] M. Nadini, K. Sun, E. Ubaldi, M. Starnini, A. Rizzo, and N. Perra, "Epidemic spreading in modular time-varying networks," *Scientific reports*, vol. 8, no. 1, pp. 1–11, 2018.

[16] C. M. Schneider, T. Mihaljev, S. Havlin, and H. J. Herrmann, "Suppressing epidemics with a limited amount of immunization units," *Physical Review E*, vol. 84, no. 6, p. 061911, 2011.

[17] C. Chen, H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos, "Eigen-optimization on large graphs by edge manipulation," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 4, pp. 49:1–49:30, 2016.

[18] B. A. Prakash, H. Tong, N. Valler, M. Faloutsos, and C. Faloutsos, "Virus propagation on time-varying networks: Theory and immunization algorithms," in *PKDD*, ser. Lecture Notes in Computer Science, vol. 6323. Springer, 2010, pp. 99–114.

[19] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Analysis and control of epidemics: A survey of spreading processes on complex networks," *IEEE Control Systems Magazine*, vol. 36, no. 1, pp. 26–46, 2016.

[20] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic processes in complex networks," *Reviews of modern physics*, vol. 87, no. 3, p. 925, 2015.

[21] J. Zhan, T. Rafalski, G. Stashkevich, and E. Verenich, "Vaccination allocation in large dynamic networks," *Journal of Big Data*, vol. 4, no. 1, p. 2, 2017.

[22] M. E. Ali, S. S. Eusuf, K. Abdullah, F. M. Choudhury, J. S. Culpepper, and T. Sellis, "The maximum trajectory coverage query in spatial databases," *Proc. VLDB Endow.*, vol. 12, no. 3, pp. 197–209, 2018.

[23] P. Zhang, Z. Bao, Y. Li, G. Li, Y. Zhang, and Z. Peng, in *SIGKDD*. ACM, 2018, pp. 2748–2757.

[24] Y. Zhang, Y. Li, Z. Bao, S. Mo, and P. Zhang, "Optimizing impression counts for outdoor advertising," in *SIGKDD*. ACM, 2019, pp. 1205–1215.

[25] E. Yilmaz, S. Elbasi, and H. Ferhatosmanoglu, "Predicting optimal facility location without customer locations," in *SIGKDD*. ACM, 2017, pp. 2121–2130.

[26] S. Ahmadian, A. Epasto, R. Kumar, and M. Mahdian, "Clustering without over-representation," in *SIGKDD*. ACM, 2019, pp. 267–275.

[27] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf, "Structure and dynamics of information pathways in online media," in *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, S. Leonardi, A. Panconesi, P. Ferragina, and A. Gionis, Eds. ACM, 2013, pp. 23–32.

[28] R. M. Anderson, B. Anderson, and R. M. May, *Infectious diseases of humans: dynamics and control*. Oxford university press, 1992.

[29] M. Farajtabar, J. Yang, X. Ye, H. Xu, R. Trivedi, E. B. Khalil, S. Li, L. Song, and H. Zha, "Fake news mitigation via point process based intervention," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 1097–1106.

[30] Q. Zhan, J. Zhang, S. Wang, P. S. Yu, and J. Xie, "Influence maximization across partially aligned heterogenous social networks," in *PAKDD*, ser. Lecture Notes in Computer Science, vol. 9077. Springer, 2015, pp. 58–69.

[31] C. for Disease Control and P. (CDC), "Principles of epidemiology in public health practice," U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES, Tech. Rep., 2006.

[32] K. Lin, D. Y.-T. Fong, B. Zhu, and J. Karlberg, "Environmental factors on the sars epidemic: air temperature, passage of time and multiplicative effect of hospital infection," *Epidemiology & Infection*, vol. 134, no. 2, pp. 223–230, 2006.

[33] C. Ritz, F. Baty, J. C. Streibig, and D. Gerhard, "Dose-response analysis using r," *PloS one*, vol. 10, no. 12, p. e0146021, 2015.

[34] A. Logins, Y. Li, and P. Karras, "On the robustness of cascade diffusion under node attacks," in *WWW*. ACM / IW3C2, 2020, pp. 2711–2717.

[35] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi, "Dynamic influence analysis in evolving networks," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1077–1088, 2016.

[36] S. A. Vinterbo, "A note on the hardness of the k-ambiguity problem," *Technical Report DSG-T R-2002-006*, 2002.

[37] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *SODA*. SIAM, 2014, pp. 946–957.

[38] Q. Guo, S. Wang, Z. Wei, and M. Chen, "Influence maximization revisited: Efficient reverse reachable set generation with bound tightened," in *SIGMOD*. ACM, 2020, pp. 2167–2181.

[39] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: near-optimal time complexity meets practical efficiency," in *SIGMOD*. ACM, 2014, pp. 75–86.

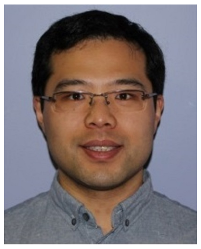
[40] C. for Disease Control and P. (CDC), "Clinical signs and symptoms of influenza," U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES, Tech. Rep., 2020.

[41] S. A. Lauer, K. H. Grantz, Q. Bi, F. K. Jones, Q. Zheng, H. R. Meredith, A. S. Azman, N. G. Reich, and J. Lessler, "The incubation period of coronavirus disease 2019 (covid-19) from publicly reported confirmed cases: estimation and application," *Annals of internal medicine*, vol. 172, no. 9, pp. 577–582, 2020.

[42] W. H. Organization, "Advice on the use of masks in the context of covid-19," World Health Organization, Tech. Rep., 2020.

[43] ——, "Contact tracing in the context of covid-19," World Health Organization, Tech. Rep., 2020.

Yipeng Zhang received the PhD degree in computer science from RMIT University in 2023. He is currently working in CSIRO. His research interests include combinatorial optimization, database system, large language model, and machine learning. He received the best paper award runner-up in SIGKDD 2019.



Zhifeng Bao received the PhD degree in computer science from the National University of Singapore in 2011 as the winner of the Best PhD Thesis in school of computing. He is currently a professor with the RMIT University and leads the big data research group at RMIT. He is also an Honorary Fellow with University of Melbourne in Australia. His current research interests include data usability, spatial database, data integration, and data visualization.

Yuchen Li is an assistant professor at the School of Computing and Information Systems, Singapore Management University (SMU). He received double BSc degrees in applied math and computer science (both with first-class honors) and a Ph.D. degree in computer science from the National University of Singapore (NUS), in 2013 and 2016, respectively. His research interests include graph analytics and heterogeneous computing. He received best paper awards in KDD and a Lee Kong Chian fellowship.

Baihua Zheng received the PhD degree in computer science from the Hong Kong University of Science and Technology, China, in 2003. She is currently a professor with the School of Information Systems, Singapore Management University, Singapore. Her research interests include mobile/pervasive computing, spatial databases, and big data analytics.

Xiaoli Wang received the BS degree from the Northeastern University of China, and the PhD degree from the National University of Singapore, all in computer science. She is currently an associate professor with the School of Informatics, Xiamen University. Her research interests mainly focus on database and data mining relevant issues, including indexing and query processing on complex structures, data cleaning and integration, textual data processing, big healthcare data, and social media computing.