A Graph-Based Approach Towards Risk Alerting for COVID-19 Spread

Aibo Guo, QianZhen Zhang, and Xiang Zhao(✉)

Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha, Hunan, China
{aiboguo,zhangqianzhen18,xiangzhao}@nudt.edu.cn

Abstract. With the spiraling pandemic of the Coronavirus Disease 2019 (COVID-19), it has becoming inherently important to disseminate accurate and timely information about the disease. Due to the ubiquity of Internet connectivity and smart devices, social sensing is emerging as a dynamic sensing paradigm to collect real-time contacts between both people and places. For example, we can rely on the Bluetooth signals that smartphones can both send out and receive to collect the real-time user contacts data. Based on the contacts data, in this paper, we investigate to propose an efficient approach to calculate the risk level of each person to have COVID-19. It can help pinpoint the people who need to be isolated. (1) We model the real-time contact data between people as a straming graph, which is a constantly growing sequence of edges. (2) We provide a risk alerting model to find the people who came in contact with someone having COVID-19. (3) In addition, we design efficient algorithms to calculate the risk level of each person and update the levels in real time. (4) Extensive experiments verify the effectiveness and efficiency of our approach.

1 Introduction

Public health experts say tracing who people infected with the coronavirus have been in contact with is a critical step in easing social distancing restrictions. Thanks to the pervasion of smart devices, some softwares, i.e., TWS[1] and TraceTogether[2], have been designed for collecting the real-time contact data between people. The technologies used in them rely on the Bluetooth signals that smartphones can both send out and receive. Using Bluetooth signals can capture the contact records between users and each contact record is used to answer the question "was user A in contact with user B at time T?" Time is important since there maybe multi-times contacts between two users. Based on the collected data, we design efficient algorithms to find the potential users who may have COVID-19.

[1] http://easytws.com/.
[2] https://www.tracetogether.gov.sg/.

L. Qin et al. (Eds.): SFDI 2020/LSGDA 2020, CCIS 1281, pp. 63–69, 2020.
https://doi.org/10.1007/978-3-030-61133-0_5

From the perspective of data management, there may exist two types of solutions—relational and graph-based—to the user-contact data. Using relational databases does not always offer an elegant solution towards efficiently searching, and still lacks best practices currently. In this paper, we design a graph-based solution, attributed to the fact that the user-contact data is a universal graph model of data. As a result, we model the use-contact data as a streaming graph $\mathbb{G}$, which is a constantly growing of edges $\{\sigma_1, \sigma_2, \ldots, \sigma_x\}$ where each σ_i arrives a particular time t_i. Note that, σ_i may have multi-timestamps since σ_i will appear multi-times in $\mathbb{G}$. Specially, we only collect each user's contact data in the prior 14 days. This is because the incubation period of COVID-19 is 1∼14 days.

In order to find the users who came in contact with someone having COVID-19, we propose a risk alerting model to assign each user a risk-level, denoted as RL. That is, when a user tests positive for COVID-19, we assign RL–1, RL–2 and RL–3 to the users who are one-hop, two-hops and three-hops neighbors, respectively. Furthermore, if a user does not have COVID-19 after 14 days or all the neighbors of the user have no risk-level, we can remove the user's risk-level directly. Note that, the time constraints are important in this model, more details will described in Sect. 3.

To achieve real-time responsiveness is the foremost problem we need to face when updating the risk-levels over the streaming graph $\mathbb{G}$; if not, we cannot get efficient updating results over a time span. A naïve method to solve this problem is to recompute risk-levels for the users who have COVID-19. However, it can be prohibitively costly, and we will redo the work. Instead, we design an incremental updating algorithm to calculate the risk-level for each new user-contact record and update the risk-levels for corresponding users only from the newly user who tests positive.

Contributions. In short, we make the following contributions:

- We model the user-contact records collected from the Bluetooth signals of corresponding users' smartphones as a streaming graph.
- Based on the streaming graph, we design a risk alerting model which help to trace the people have been in contact with someone infected with the coronavirus.
- We propose an incremental updating algorithm to update the risk-levels for corresponding users.

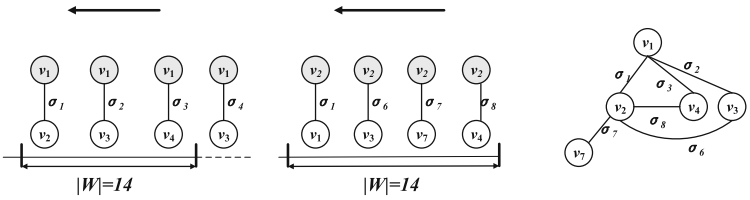
Experiment results demonstrate the effectiveness and efficiency of our techniques.

2 Preliminaries

A typical data schema for the *topology of user-contact records* consists of a number of vertices representing users, and links between the nodes representing contacts between them. This schema naturally translates to a vertex-labeled undirected graph $g = (V, E, L)$.

Definition 1 (Streaming graph). *A* Streaming graph $\mathbb{G}$ *is a constantly growing sequence of undirected edges* $\{\sigma_1, \sigma_2, \ldots, \sigma_x\}$ *where each* σ_i *arrives at a particular time* t_i *(*$t_i < t_j$ *when* $i < j$*).* t_i *is also referred to as the timestamp of* σ_i*. Each edge* σ_i *has two labelled vertices and two edges are connected if and only if they share one common endpoint.*

Since two users may have multiple contacts with each other, there may be multi-edges between two vertices in $\mathbb{G}$ representing contact records between them in different timestamps. For each user's contact records, we use the *time-based sliding window model*, where a sliding window W defines a timespan with fixed duration $|W|$. Here, we set $|W| = 14$ since the incubation period of COVID-19 is $1 \sim 14$ days. An example of a streaming graph $\mathbb{G}$ is shown in Fig. 1(a). For each vertex, i.e., v_1 or v_2, we record the corresponding edges within 14 days.



(a) Graph stream under time window of size 14 for each vertex (b) A snapshot of the streaming graph

Fig. 1. An example of the streaming graph

Definition 2 (A Snapshot of a Streaming Graph). *Given a streaming graph* $\mathbb{G}$ *at current time point* t*, the current snapshot of* $\mathbb{G}$ *is a graph* $\mathbb{G}_t = (\mathbb{V}_t, \mathbb{E}_t)$ *where* $\mathbb{E}_t$ *is the set of edges that occurs on each vertex at time* t*.*

Figure 1(b) shows the snapshot of Fig. 1(a) at time point σ_1.

3 Risk Alerting Model

In this section, we design a risk alerting model for COVID-19 spread, namely, RAMC, to trace the people who have been in contact with someone infected with the coronavirus. The model consists of two steps: (1) assign each user in the streaming graph a risk-level according to the users who have COVID-19; (2) eliminate the risk-level of a user whose status becomes safe.

3.1 Risk-Level Assignment

We rely on the BFS search algorithm to assign each user who has been in contact with someone infected with the coronavirus directly or indirectly a risk-level.

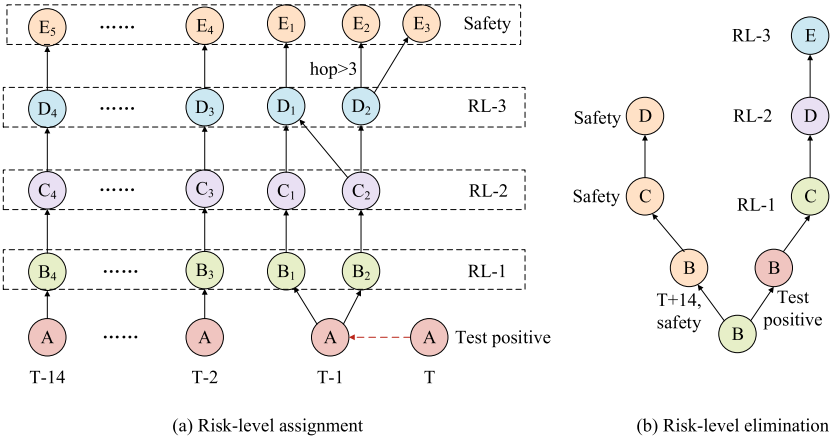


Fig. 2. Risk alerting model

Given a vertex $\mathcal{V}$ in $\mathbb{G}$ that represents a user who tests positive for COVID-19 at timepoint T, we can obtain corresponding contact records of $\mathcal{V}$ from T-1 to T-14. Here, we only assign the risk-levels for 1-hop to 3-hops neighbors of $\mathcal{V}$. In detail, for each neighbor $\mathcal{V}'$ of $\mathcal{V}$, we first set $\mathsf{RL}(\mathcal{V}') = 1$ where $\mathsf{RL}(\mathcal{V}')$ represents the risk-level of $\mathcal{V}'$. Note that, there may be multi-edges between $\mathcal{V}$ and $\mathcal{V}'$ with different timepoints. The earliest timestamp, denoted as $\mathsf{ET}(\mathcal{V}')$, will be used in the BFS process to calculate the risk-levels for other vertices. Then for each unvisited neighbor $\mathcal{V}''$ of $\mathcal{V}'$, we check whether there is an edge $\langle \mathcal{V}', \mathcal{V}'' \rangle$ with timestamp $t_{v''}$ such that $t_{v''} \geq \mathsf{ET}(\mathcal{V}')$; if so, we set $\mathsf{RL}(\mathcal{V}'') = 2$. What's more, we set $\mathsf{ET}(\mathcal{V}'') = t_{v''}$ if $t_{v''}$ is the earliest timestamp that can confirm above condition. Specially, there may be another neighbor $\mathcal{V}^*$ of $\mathcal{V}''$ and $\mathsf{RL}(\mathcal{V}^*) = 1$. As a result, we need also calculate another value for $\mathsf{ET}(\mathcal{V}'')$ based on the edges between $\mathcal{V}''$ and $\mathcal{V}^*$. In this case, we set $\mathsf{ET}(\mathcal{V}'')$ as the smallest value between all the values. Finally, we calculate the risk-levels for the 3-hops neighbors of $\mathcal{V}$ in a similar manner. Omitted in the interest of space, we do not describe here.

Figure 3(a) gives the example to calculate corresponding users' risk-levels when the user A tests positive for COVID-19 at time T.

3.2 Risk-Level Elimination

In our model, a user's risk-level will be eliminated if we can make sure the status of the user is safe.

For each vertex $\mathcal{V}$ in $\mathbb{G}$ with its $\mathsf{RL}(\mathcal{V}) = 1$ at timestamp T, if (1) the user represented by the vertex $\mathcal{V}$ has no symptoms of COVID-19 at timestamp T+14; and (2) there is no user represented by the neighbor of $\mathcal{V}$ who tests positive for COVID-19 between time T and T+14, we can eliminate the risk-level of $\mathcal{V}$. As for other vertices $\{\mathcal{V}'\}$, we check whether (1) there exists a neighbor $\mathcal{V}''$ with its $\mathsf{RL}(\mathcal{V}'') > \mathsf{RL}(\mathcal{V}')$; and (2) there is an edge $\langle \mathcal{V}', \mathcal{V}'' \rangle$ with its timestamp later

than $\mathsf{ET}(\mathcal{V}'')$. If not, we can eliminate the risk-level of $\mathcal{V}'$. Figure 3(b) shows the elimination process when the status of becomes safe.

4 Incremental Algorithms

In this section, we propose an effective algorithm, namely, updateRL, to update the risk-levels of corresponding users when a new contact record is added into the streaming graph $\mathbb{G}$.

Now we explain updateRL, which is invoked for each edge insertion $\langle v, v' \rangle$ with timestamp t_1. Firstly, updateRL checks the risk-levels of v and v', respectively. Note that, each user who tests positive for COVID-19 will not have new contact record. As a result, we have the following two cases that may cause the update of the risk-levels.

① **From RL -3 to RL -2.** Suppose that $\mathsf{RL}(v') = 3$ and $\mathsf{RL}(v) = 1$. If $t_1 > \mathsf{ET}(v)$, updateRL transits $\mathsf{RL}(v')$ from 3 to 2 and sets $\mathsf{ET}(v') = t_1$.

② **From safety to RL -2 (or RL -3).** Suppose that the status of v' is safety and $\mathsf{RL}(v) = 1$ (or $\mathsf{RL}(v) = 2$). If $t_1 > \mathsf{ET}(v)$, updateRL sets $\mathsf{RL}(v') = 2$ (or $\mathsf{RL}(v') = 3$) and sets $\mathsf{ET}(v') = t_1$.

5 Experiments

In this section, we report experiment results and analyses.

5.1 Experiment Setup

The proposed algorithms were implemented using C++, running on a Linux machine with two Core Intel Xeon CPU 2.2 Ghz and 32 GB main memory. Particularly, three algorithms were implemented: (1) RAMC, our algorithm to assign and eliminate corresponding users' risk levels; (2) updateRL, our update algorithm for newly added contact record; (3) updateRL-R, our algorithm that recomputes the users' risk-levels from the uses who have COVID-19.

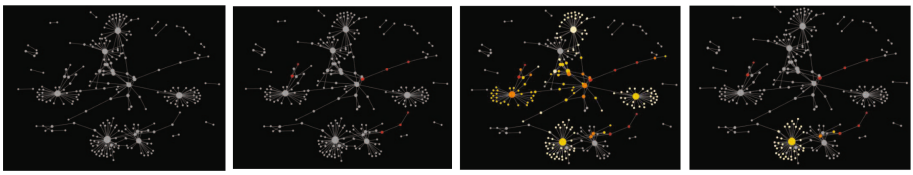


Fig. 3. The mainly process of RAMC

Since we do not have the real-life user-contact records, we use two human contact temporal networks, i.e., HS [1] and PS [1] to simulate the user-contact

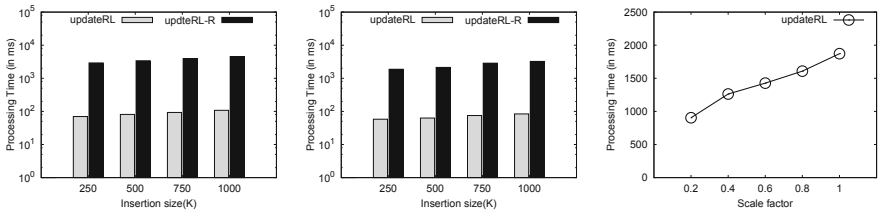
records. HS contains 2,367,984 triples while the edge insertions consist of 225,124 triples. PS contains 1,254,132 triples while the edge insertions consists of 112,607 triples. We also use a synthetic streaming social graph data LSBench which contains 23,549,621 triples.

5.2 Evaluating the Effectiveness of RAMC

In this subsection, we evaluate the effectiveness of our proposed risk alerting model. We ran experiments on HS and PS and randomly set 1000 vertices as the users who test positive for COVID-19 on both datasets. According to the experiment results, we find that the risk-levels of corresponding users can be efficient calculated within 600ms on both datasets. Figure 3 shows the partial visualization results in our experiment by using HS dataset. In detail, the first picture shows the partial initial graph; the second picture shows some users who have COVID-19 are emerged in the graph; the third graph shows the risk-level assignment process and the last picture shows the risk-level elimination process.

5.3 Varying the Edge Insertion Size

In this subsection, we evaluate the impact of edge insertions on the performance of updateRL and updateRL-R. We vary the number newly-inserted triples from 25K ($= 25 \times 10^3$) 100K in 25K increments on both datasets. Figure 4(1) and Fig. 4(2) shows the processing time for each algorithm. We see that updateRL has consistently better performance than updateRL-R. What's more, the figure reads a non-exponential increase as edge insertion size grows. Specially, updateRL outperforms updateRL-R by up to 42.78 times.



(1) HS (2) PS (3) LSBench

Fig. 4. Experiment results

5.4 Varying Data Sizes

We evaluate the scalability of updateRL on LSBench dataset. We randomly sampled about 20% to 100% from the LSBench dataset so that the data and result distribution remain approximately the same with the whole dataset. Figure 4(3)

reads a non-exponential increase as data size grows. In generally, the processing time grows at no more than twice the speed of growth in the size of the dataset. The scalability suggests that updateRL can handle reasonably large real-life graphs as those existing algorithms for deterministic graphs.

6 Related Work

Representative algorithms for pattern matching/search from the streaming graph include TurFlux [2] and TreeMat [3], etc. However, these work are about continuous subgraph matching and cannot be used in our model. To the best of our knowledge, this is among the first attempts to design a risk alerting for COVID-19 spread based on a graph search algorithm. We believe that this work will benefit for fighting COVID-19.

7 Conclusion

In this paper, we have investigated a systematic graph-based approach to risk alerting for COVID-19 spread. We design a risk alerting model to help trace the people who have been in contact with someone infected with the coronavirus and propose an incremental updating algorithm to update the risk-levels.

References

1. Fournet, J., Barrat, A.: Contact patterns among high school students. CoRR, vol. abs/1409.5318 (2014)
2. Kim, K., et al.: TurboFlux: a fast continuous subgraph matching system for streaming graph data. In: Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, 10–15 June 2018, pp. 411–426 (2018)
3. Zhang, Q., Guo, D., Zhao, X., Guo, A.: On continuously matching of evolving graph patterns. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, 3–7 November 2019, pp. 2237–2240 (2019)