Identifying Likely-Infected Nodes and Reconstructing the Infection Trace for COVID-19

Vidushi Vashishth
Georgia Institute of Technology
Atlanta, Georgia
vvashishth3@gatech.edu

Abhinav Gupta
Georgia Institute of Technology
Atlanta, Georgia
agupta931@gatech.edu

Kasturi Gottivedu Shriniwas
Georgia Institute of Technology
Atlanta, Georgia
kshriniwas3@gatech.edu

ABSTRACT

It is a challenging problem in epidemiology to map out the exact trace of the spread of infection during pandemics such as Covid-19 due to the extremely high rates at which these viruses spread and infect people. In this work, we aim to reconstruct the propagation of infection for Covid-19 in South Korea. We attempt to recover the exact trace of infection, which includes identifying other likely-infected nodes that are unreported, as well as the seeds or culprits of the network. We employ CuLT [6] which maps this reconstruction task into the classic directed Steiner-tree problem. We construct the temporal network consisting of patient interactions and an initial list of reported infections from the DS4C dataset [3]. We subsequently use CuLT to recover the flow of infection in South Korea during the Covid-19 pandemic, whilst identifying the missing infections. This work showcases the robustness and scalability of approaches like [6] on real-world epidemiological datasets to successfully recover the flow of spread of infection.

KEYWORDS

epidemiology, covid-19

ACM Reference Format:
Vidushi Vashishth, Abhinav Gupta, and Kasturi Gottivedu Shriniwas. 2021. Identifying Likely-Infected Nodes and Reconstructing the Infection Trace for COVID-19. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Modeling the spread of a virus in a population is a data-intensive task. Even official numbers from accredited institutes are bound to be biased or incomplete owing to the complex nature of the problem. A lack of awareness about new diseases, lack of infrastructure for adequate testing, population hesitancy, and numerous other reasons lead to biased data and reports about the rate of infections. Similarly, even in social media, there are approximately 6,000 tweets sent out every second and it's difficult to collect complete cascades of data. Hence, we seldom have access to the exact trace of infection and our data is noisy and incomplete. Any model of an epidemic inferred using this data is bound to carry these biases and deviate from the ground truth. This project is an effort to estimate the ground truth by zeroing in on the missed infections. Modeling the spread of trends on social media is also a useful application area where this study will help. Data on viral blogs/social media posts/videos/hashtags also suffers from biases. Privacy settings of users, lack of network connectivity are some bottlenecks to

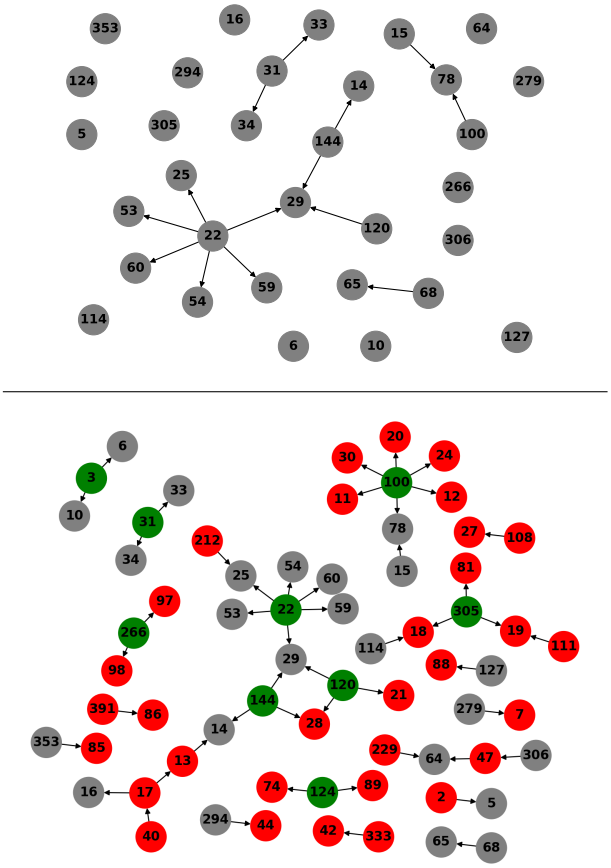

Figure 1: Recovering the infection trace on the DS4C: South Korea dataset using CuLT [6]. Top: the reported infection nodes in the network. Bottom: the recovered trace of infection after employing CuLT. The nodes in red are the likely-infected nodes that were not reported in the figure on top. The nodes in green are the seed nodes identified while reconstructing the propagation of infection.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

collecting accurate data. Our project will help identify missed users that helped fuel a trend or a pandemic.

Towards this end, we have used NETFILL [8] and CuLT[6] to find the missed nodes of infection in the spread of COVID-19 in Korea as well as Singapore during the year 2020. We hence had the opportunity to compare the performance of these algorithms. This was especially challenging because it required understanding the accepted input formats of the algorithms as well as the datasets thoroughly, and zeroing in on the pre-processing demands of these algorithms. Handling the Singapore COVID dataset was difficult as it has been presented in the form of a graphic dashboard which makes it difficult to scrape programmatically. We tried to attack this problem by starting with a small subgraph of infection of these datasets and giving it as input to both algorithms. We then iteratively increased the size of the subgraph to see how these algorithms scale on the datasets.

The report has been further divided into the following sections. The comments made on the mid-term milestone report have been addressed in Section 2. The details of the algorithms employed and related literature survey have been detailed in Section 3. Section 4 defines the problem statement more formally. Section 6 describes the experiments and related results. We conclude our findings and discuss the future scope of work in Section 7.

2 RESPONSE TO MILESTONE COMMENTS

As per the comments on our milestone report, we have explained the problem statement and what we are trying to solve more formally in Section 4. In that section, we have also detailed upon what all information is provided to us by the datasets under consideration, and how we modeled this to cater to our algorithms.

3 RELATED WORK AND SURVEY

In the current literature around reverse engineering of an epidemic, the focus has been more on finding the first patients or culprit nodes in the population. NETSLEUTH [5] explores the problem of identifying the source or seed node (or the 'culprit') from which the infection began to spread to other nodes in the graph, for the Susceptible-Infected (SI) model. This work employs the Minimum Description Length (MDL) [2] principle to identify the set of seed nodes and the virus propagation ripple, which best explains the graph snapshot. The NETSLEUTH algorithm first identifies likely sets of seed nodes from the graph and then subsequently optimizes the virus propagation ripple by maximizing the likelihood. The algorithm not only identifies the seed nodes (the nodes from which the epidemic started) but also finds the number of culprit seed nodes responsible for the epidemic. This work was an important contribution to the epidemiology literature as the identification of seed nodes in network infection models was understudied at the time. There were existing works that focussed on identifying a single source node. But this was the first work to identify multiple seed nodes and also automatically determine their number.

Lappas et al [4] also work on a similar problem statement of identifying k active nodes that best explain the observed activation state of the graph. The activation state is simply the state of nodes in the network - whether they're infected or not. They refer to these nodes as 'effectors' and define the 'k-effectors' problem, where the goal is to find a set of k active nodes such that, had the information propagation started from them, it would cause the activation state that is currently being observed in the network. Hence, these effectors are nothing but seeds or culprits. The authors put forth an efficient dynamic-programming algorithm that can solve the k-EFFECTORS problem optimally in polynomial time. This work was also very crucial to the literature as it was the first to consider the k-effectors problem in networks.

Approach	Linear or Near-Linear?	Can identify seed nodes	Automatically determine 'k'?	Recover missing nodes?
Lappas et al	✗	✓	✗	✗
NETSLEUTH	✓	✓	✓	✗
Sadikov et al	✗	✗	✗	✓
NETFILL	✓	✓	✓	✓
CuLT	✓	✓	✓	✓

Figure 2: Algorithm Comparison

There are some key distinctions between NETSLEUTH [5] and the work done by Lappas et al [4]. Both wish to identify k-seed nodes that cause the observed activation state, but the information propagation models are different. NETSLEUTH studies this under the Susceptible-Infected model while the latter studies it under an independent cascade model. Moreover, NETSLEUTH has a linear time complexity whereas [4] has quadratic time complexity. But most importantly, the biggest difference in these works is that [4] cannot automatically determine the number of seed nodes. The value of k has to be pre-defined for the algorithm to work. But NETSLEUTH finds the number of seed or culprit nodes automatically and does not need the value of k as input.

One of the works which address the problem of identifying the missing nodes is [7]. In this work, Sadikov et al. attempt to correct the missing data in information cascades by using a k-tree model of cascades. Using the proxy-K-tree model, they infer characteristics such as the size and number of edges of the resulting cascades. They empirically prove that the properties estimated via a proxy cascade are much closer to the properties of the complete cascade than that of a cascade with missing data. The proposed work is effective in detecting the missing nodes in cascades but it may fail for severely imbalanced cascade trees. Further, it only considers broad statistical features like average size and depth for recovering original cascades. In contrast, [8] presents a novel approach to correct the missing data at a per-node level.

In their paper [8] Sundareisan et al. are the first ones to focus on identifying missing infections *along with* possible culprits in a given network. To this end, they proposed the algorithm NETFILL which considers the spread of the disease to follow the SI model. NETFILL [8] identifies both, the missing infections as well as the source nodes (or culprits) of the epidemic. It is also near-linear in

	patient_id	sex	age	country	province	city	infection_case	infected_by	contact_number	symptom_onset_date	confirmed_date	released_date
0	1000000001	male	50s	Korea	Seoul	Gangseo-gu	overseas inflow	NaN	75	2020-01-22	2020-01-23	2020-02-05
1	1000000002	male	30s	Korea	Seoul	Jungnang-gu	overseas inflow	NaN	31	NaN	2020-01-30	2020-03-02
2	1000000003	male	50s	Korea	Seoul	Jongno-gu	contact with patient	2002000001	17	NaN	2020-01-30	2020-02-19
3	1000000004	male	20s	Korea	Seoul	Mapo-gu	overseas inflow	NaN	9	2020-01-26	2020-01-30	2020-02-15
4	1000000005	female	20s	Korea	Seoul	Seongbuk-gu	contact with patient	1000000002	2	NaN	2020-01-31	2020-02-24

Figure 3: DS4C dataset

confirmed_date	infected_by	patient_id	Info1	Info2	Info3	Info4	Info5	Info6
2020-01-30	2002000001	1000000003	1	1	0	0	0	0
2020-01-31	1000000002	1000000005	1	1	0	0	0	0
2020-01-31	1000000003	1000000006	1	1	0	0	0	0
2020-01-31	1000000003	1000000007	1	1	0	0	0	0
2020-02-05	1000000003	1000000010	1	1	0	0	0	0
2020-02-16	1000000017	1000000013	1	1	0	0	0	0
2020-02-16	1000000013	1000000014	1	1	0	0	0	0
2020-02-19	1000000070	1000000078	0	0	0	0	0	0
2020-02-19	1000000070	1000000079	0	0	0	0	0	0
2020-02-19	1000000070	1000000087	0	0	0	0	0	0
2020-02-19	1000000070	1000000088	0	0	0	0	0	0
2020-02-19	1000000070	1000000089	0	0	0	0	0	0

Figure 4: Processed D4SC Dataset for CULT

complexity in most cases and automatically detects the number of seed nodes in the network. We used NETFILL on the Singapore COVID dataset as well Korea COVID dataset to zero in on the missed infections. The details of this experiment formulation and related results have been discussed in Section 6

Rozenshtein et al [6] consider the problem of reconstructing an epidemic over time and recovering the exact flow of spread, which includes identifying the missing nodes as well as the seed nodes! This work puts forth an effective and scalable framework called "CuLT" which formulates this problem as a classic temporal directed Steiner-tree computation. A Steiner tree is a tree in a graph that spans at least all the nodes in a terminal set. It may also contain nodes that are in the graph but not in the terminal set, but it will most definitely span all the nodes in the terminal set. We explored the idea of using such trees for finding missing infections in Homework-2. This work [6] also considers the same fundamental idea and maps the reconstruction task into the classic directed Steiner-tree problem. The novelty of this approach lies in the fact that it explicitly takes into account the exact time that nodes interact, resulting in more accurate reconstructions of the epidemic and the identification of missing nodes. Rozenshtein et al had run the CuLT algorithm on the Flixter dataset. Flixter was modeled on a social network, where nodes, signifying people, rate different movies. The timestamp of this rating helps establish a directed edge, signifying the spread of infection, between different user nodes. So a directed edge will be established between nodes u and v (neighbors in the social network), if both these nodes rate the same movie within a gap of 7 days. We have similarly adopted the CuLT algorithm on the Korea COVID dataset. The details of this experiment, the underlying assumptions taken for this modelling, and the discussion of the derived results are mentioned in the Section 6.

4 PROBLEM DEFINITION

4.1 NETFILL

Given a connection network between different people and the state of infection they are in (Susceptible or Infected), our goal is to figure out the culprits of the pandemic as well as any missed infected nodes using NETFILL. More formally,

Problem 1. *Given connection network graph G = (V, E), where V denotes people in the social network and edges E establish proximity or contact (possibility of infection spread) between the people. V can either have the color red, signifying its infected state, or green, signifying its susceptible state. The goal is to find the green-colored vertex set MISSED which has vertices that should ideally have been colored red. Another goal is to find the set SEED which has red-colored nodes that were the starting point of the spread of the pandemic.*

4.2 CuLT

Given a temporal network and a sample of reported infections, CULT aims to recover the epidemic by reconstructing the flow of spread, including identifying the seed nodes and identifying likely-infected nodes that were not reported. This approach explicitly takes into account the exact time that nodes interact, which leads to more accurate reconstructions. It recovers the infection trace by mapping this reconstruction problem into the classic directed Steiner-tree problem and subsequently applying modified and faster versions of known approximation algorithms. We leverage this algorithm to recover the trace and identify missing infections in the DS4C [3] dataset. We mathematically define the problem as

Problem 2. *Given temporal network $G = (V, E)$, where E denotes a history of interactions between nodes V, and given reports $R = (u, t)$, our goal is to determine the infection paths P such that the cost $(P|R)$ is minimized. Intuitively, we need to find P which best explains the*

temporal distribution of R.Further, for a given set of candidate seed nodes C, we determine the paths P such that $S(P) \subseteq S$ where $S(P)$ is a set of seed nodes determined by the paths P.

5 DATASETS AND DATA PREPROCESSING

5.1 DS4C: The South Korea Dataset

The primary dataset we utilize for our project is the Data Science for Covid-19 (DS4C) dataset, which contains epidemiological data of COVID-19 patients in South Korea [3]. It contains the ID of the patient, the number of contacts with other people, and most importantly, the ID of who infected the patient! The data also contains the date of the onset of symptoms for each patient, the 'confirmed' and 'released' dates and whether the infection occurred due to an overseas inflow or due to contact with a patient. Hence, we preprocess this dataset to build the contact network and the patient interactions and use it to identify the missed nodes in an infection trace. A snippet of this dataset is shown in figure 3

We wish to use the South Korea dataset for reconstructing the infection trace with CuLT. To this end, we preprocess the dataset and build the temporal interaction network from the data, which contains information about when two nodes interacted, whether they infected one another and whether either of them was a seed/culprit node. We majorly make two assumptions while constructing the temporal network:

- Given that n_1 tested positive at timestamp $\hat{t}$ and n_2 infected n_1, we assume that n_1 might have interacted with n_2 at $t = \hat{t} - 7$, i.e 7 days prior to the date when n_1 tested positive.
- Given that $n_1, n_2 \in c$, where c denotes a group case and a group case accounts for a large number of infections that originated from a given location l such as a church or a market, we assume that all the nodes $n \in c$ might have interacted with each other around 7 days prior to the date when the first infection $i \in c$ was reported.

As it takes 7 days on average for the onset of covid symptoms, these assumptions are intuitive in nature. Now, we define an interaction I between n_1 and n_2 at timestamp t as (n_1, n_2, t) and create a temporal network by formulating a history of interactions E. The input to the CuLT algorithm consists of interactions along with potential seeds. We define candidate seeds S as the nodes which were the first ones to test positive in a group infection case. Finally, for every interaction, we define that every activated node has a β probability to be reported. Note that we could have considered all nodes as reported because all nodes are essentially reported according to our data set. But, in order to evaluate the efficacy of the CuLT algorithm in recovering the epidemic with missing information, we consider only a sample of reported nodes. Therefore, we preprocess the DS4C dataset and extract useful information to create an input I for the CuLT algorithm as shown in figure 4. Each row of Input I can be represented by $(t, n_1, n_2, i_1, i_2, r_1, r_2, s_1, s_2)$. Here,
(t, n_1, n_2) represent an interaction I
i_1 denotes if n_1 is infected at time t
i_2 denotes if n_2 is infected at time t
r_1 denotes if n_1 is reported
r_2 denotes if n_2 is reported
s_1 denotes if n_1 is a seed node
s_2 denotes if n_2 is a seed node.

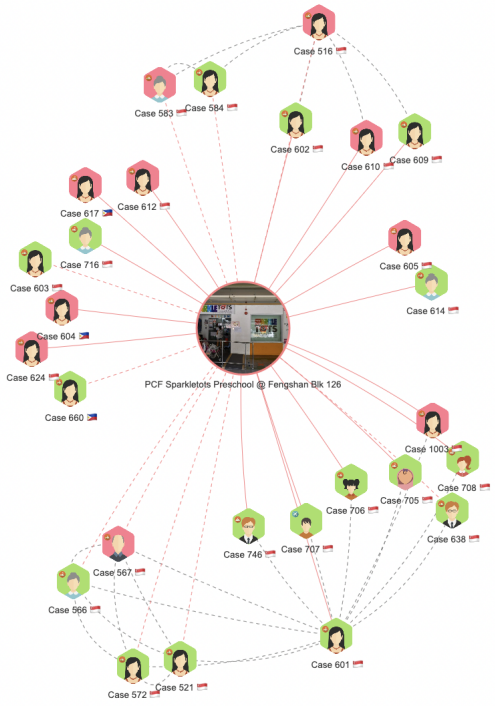

Figure 5: Singapore COVID spread subgraph

5.2 Singapore Dataset

The Covid-19 Virus Outbreak dataset for Singapore [1] is an open-source dataset created by Upcode Academy. This meticulously maintained dataset keeps track of all positive cases of the Covid-19 disease in Singapore, further distributed by gender, nationality, and age. It also keeps track of the infection sources - whether it was an imported case from a country other than Singapore, or a local transmission within the community. The dataset creates distinct clusters depending on the location of the infection and the network graph it subsequently computes. The network graph consists of various nodes (corresponding to an anonymized patient) and the edges connect it to the location where the infection occurred, which act as cluster representatives. Figure 5 displays a subgraph of the Singapore Dataset. There are certain dashed edges in the network which signify contact between two nodes or familial relationship between them. But these edges are in minority and not enough to establish a proper connection network between people.

There are certain problems in the way the Singapore dataset has been designed. A very small subset of the graph has peer-to-peer connections. Majority of the nodes have been grouped together geographically, which suggests one of the infected nodes in this region may have been a seed. To circumvent this issue we randomly selected a node in the cluster to be the seed of infection. Another big issue with this dataset is its pictorial format. Pre-processing a pictorial graph programmatically is not easy and an error-bound process. We hence started by manually creating adjacency lists of small subgraphs from Singapore Dataset, and feeding them to

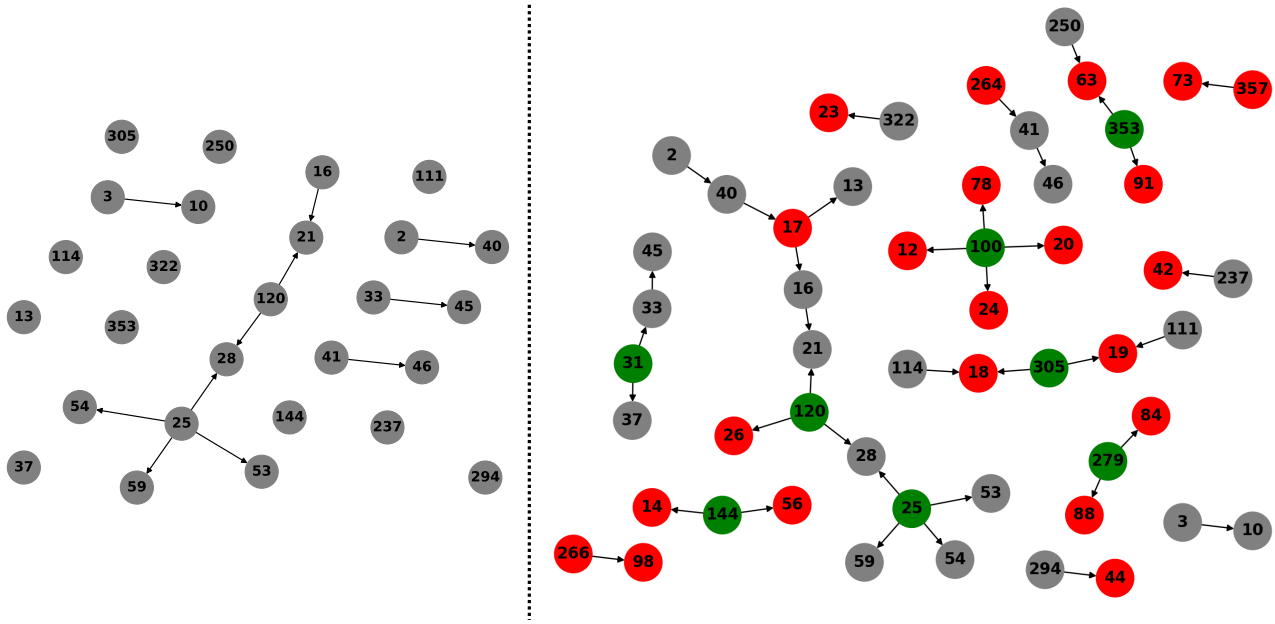

Figure 6: Recovering the infection trace on DS4C using CuLT. The graph on the left show the initially reported infection nodes (all in grey). The figure on the right is the recovered trace using CuLT [6] where the likely-infected nodes (that were not reported in the graph on the left) are shaded in red. The seed nodes are shaded in green and are identified by checking whether their in-degree is zero.

NETFILL to compute missed infections and seeds. The results of this experiment have been discussed in Section 6.2.1

6 EXPERIMENTS AND RESULTS

6.1 CuLT

Our main objective is to reconstruct the propagation of the activity of the network in order to identify the missing nodes in the Covid-19 infection trace. To this end, we use CuLT [6] to recover the flow of the spread of the Covid-19 pandemic in South Korea. CuLT formulates the problem of reconstructing an epidemic as that of a temporal Steiner-tree computation. It is an effective algorithm that improves the running time of the directed Steiner-tree problem to make it scalable to large networks. We use this approach on the data from [3], which has been preprocessed to obtain individual ripples in the trace, as explained in the previous section.

Our input consists of a temporal network which we extract from the South Korea Covid-19 dataset [3]. This contains information about when two nodes interacted, whether either of them successfully infected the other and if either of them was a seed in the infection trace. We use CuLT on this dataset to solve the temporal Steiner tree problem and recover trace of the pandemic in Korea, thereby identifying the missing nodes in the network and 'who infected whom'. Given the input temporal network and a set of 'reported nodes' - those that we know are infected for sure, we reconstruct the epidemic using CuLT and recover many missed nodes, as shown in 1.

In order to identify missing infections that are truly not known, we can consider the entire set of reported infections as the input to the algorithm. Unfortunately, all the patients in the DS4C dataset are reported infections. It does not contain information about the neighbors or friends of the patients and hence, we are unable to establish a 'friendship network' of the patients in the dataset. But if such a network can be identified, we can use CuLT to find the truly missed infections in the graph (that are likely-infected but not reported in DS4C).

The nodes in white are the initial known infected nodes which were reported. After using the CuLT algorithm, we obtain the missing nodes that were also most likely infected but not reported, which are shown in red. Hence, we are able to successfully identify missing nodes from the infection trace! Furthermore, the CuLT algorithm can also identify the starting or the seed nodes in the infection trace that are the 'culprits' by analyzing the in-degree of the nodes. Those nodes in the trace which have an in-degree of zero are ones that were not infected by anyone else in the network and hence, are the culprit nodes, as shown in green.

6.1.1 ***Evaluating CuLT's Performance for South Korea:*** In order the evaluate the performance of CuLT and how accurately it is able to reconstruct the pandemic's propagation, we sample a small subset of the infected nodes from the South Korea dataset and only 'report' them as being infected. By doing this, we consider the entire Korea dataset to be the ground-truth (what we truly know to be infected) and assume that we only know a random sample of these nodes are reported initially. We subsequently use CuLT to recover the trace and compare its results with the ground truth to analyze its performance on our dataset. The recovered trace of

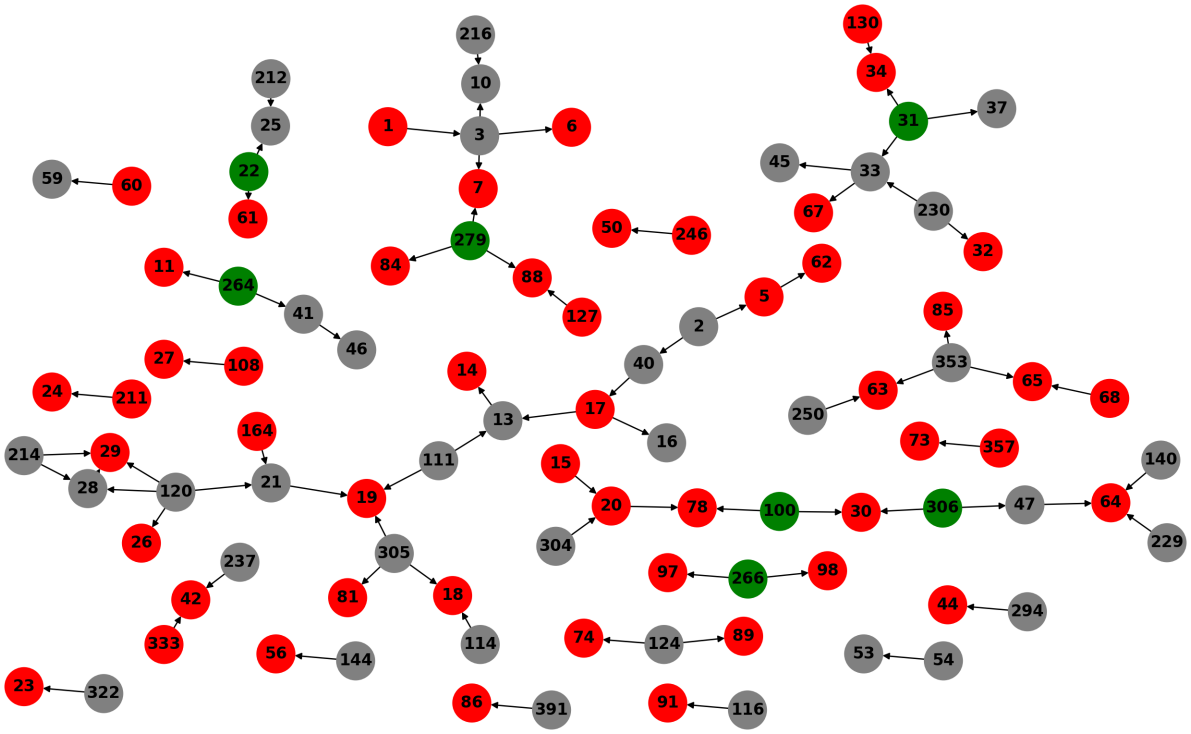

Figure 7: The entire infection trace in the DS4C dataset, which we consider as the ground truth to evaluate CuLT's [6] performance. All the nodes in the graph are actually reported as infected. However, in order to analyze CuLT's performance, we only consider a subset of these as reported - which are indicated by the grey nodes in this figure. The ones in red are the 'missing' nodes while those in green are the seeds, both of which we intend to identify while recovering the trace of the pandemic using CuLT.

infection is shown in 6 and the ground truth (actual set of reported infections) is shown in 7.

To evaluate the results, we compare the set of nodes in the Steiner trees with the ground-truth set of infected nodes and use metrics such as the Matthews correlation coefficient (MCC), F1 Score, and accuracy. We compare CulT with two baselines as done in [6]. The first is straightforward, we simply account for the given reports R (Reports). The second baseline (Baseline) returns the one-hop-cascade from the given reports. That is, given a reported infection (u, t), Baseline assumes that every future interaction (u, v, t) leads to a successful infection. The infection is not propagated further than one-hop-neighbors of a reported node as that unduly harms precision. Although none of the two baselines returns a collection of k temporal Steiner trees, we can still evaluate their accuracy against a ground-truth set of activated nodes (note that Reports has precision 1.0).

6.1.2 ***Limitations of CuLT:.*** The CuLT algorithm put forth by Rozenshtein et al [6] further improves the running time of the Steiner-tree algorithm in an attempt to make it scalable. However, the approach fails to scale for enormous and gigantic networks. For instance, when we test CuLT for the entire South Korea dataset (which consists of about 42k interactions within the friendship network), the algorithm fails to converge. The results we show are for smaller subsets of this dataset of about 5k interactions. Hence, a future direction of study is to further try to improve the performance of this approach to make it scalable for massive networks as usually discovered in real life.

6.1.3 ***Analysis of Seed Nodes Detected by CuLT.*** For every set of randomly selected reports R, CuLT ascertains 7 seed nodes on an average. We observe an interesting trend in the seed nodes detected.

- Majority of the seed nodes fall in the age group 40-60.
- Majority of the seed nodes got infected due to foreign travel.
- Majority of the seed nodes belong to Jongno-gu city or Dongdaemun city.

Therefore, we can infer that the outbreak is likely to have been started from a middle-aged person belonging to Jongno-gu city or Dongdaemun city. Further, this person is likely to have been infected due to foreign travel.

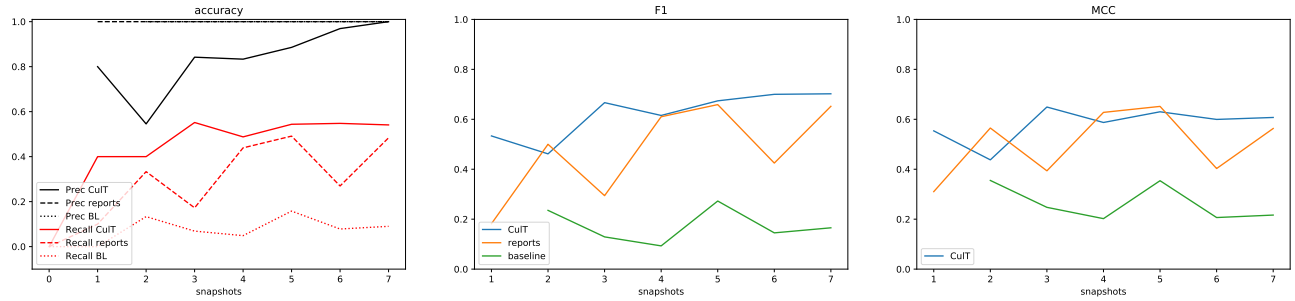

Figure 8: CuLT Performance: We evaluate how well we recover the temporal order of infections by comparing the set of nodes in the Steiner trees with the ground-truth set of all reported infection nodes. Here, we measure the quality using three metrics: Accuracy (ACC), F1 score (F1) and the Matthews correlation coefficient (MCC).

Figure 9: Subgraph of 118 nodes for South Korea Dataset

6.2 NETFILL

6.2.1 ***NETFILL on Singapore Dataset:*** As established in section 5, we use manually created subgraphs of the Singapore Dataset as input to the NETFILL algorithm. NETFILL also takes the set of the ground truth (D) of infections and currently reported set of infections (SD) as input. So we purposefully skipped some nodes from the set D to create the set SD for the subgraph sampled from the Singapore Dataset. The intuition behind this was to see if the output of the set of infected nodes *MISSED* by NETFILL includes the nodes $D - SD$. This would have been helpful in establishing the credibility of the algorithm as well. We carried this experiment with manually created subgraphs of sizes 3, 12, and 27. The outputs of all these experiments are shown in Figures 10, 11, and 12. We had computed the 3 node graph output by the milestone deadline and the results seemed decent as the choice of seeds and the correctly identified infected nodes make sense. The same cannot be said for the output of 12 and 27 node graphs. As can be seen in Figures 11 and 12, some of the identified false-negative nodes are not adjacent to the identified seeds. The reason for this was the sparsity of the graphs. The following warning was thrown while running NETFILL on the input : ***Warning: First input matrix is close to singular or badly scaled. RCOND = 3.812579e-18. Results may be inaccurate.*** Since it is non-trivial to scale the size of the subgraph from the Singapore dataset to a significant number manually, we decided to input a subgraph from the Korea dataset instead as we already had a pipeline to pre-process that graph programmatically. The results of this experiment are explained in the next section.

6.2.2 ***NETFILL on Korea Dataset:*** To run NETFILL on the DS4C dataset, we first create an adjacency matrix of nodes that have potentially interacted with each other. We construct an undirected contact network with the help of the columns 'patient_id' and

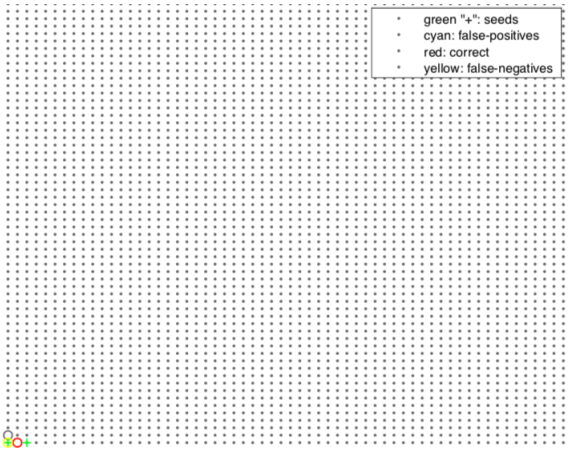

Figure 10: COVID spread on a 3 node subgraph for Singapore Dataset

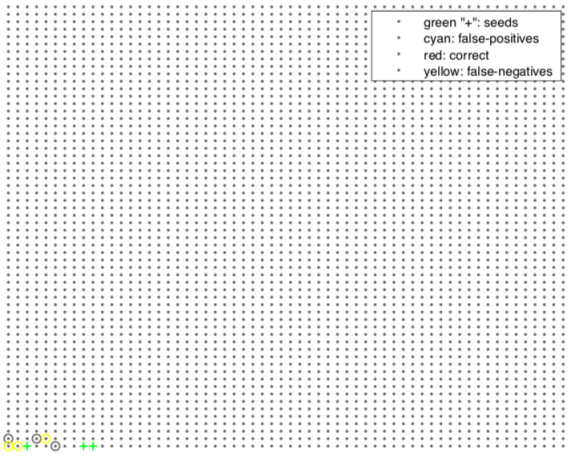

Figure 11: COVID spread on a 12 node subgraph for Singapore Dataset

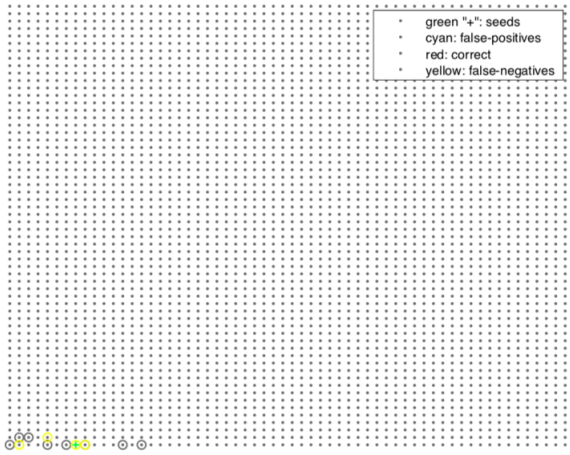

Figure 12: COVID spread on a 27 node subgraph for Singapore Dataset

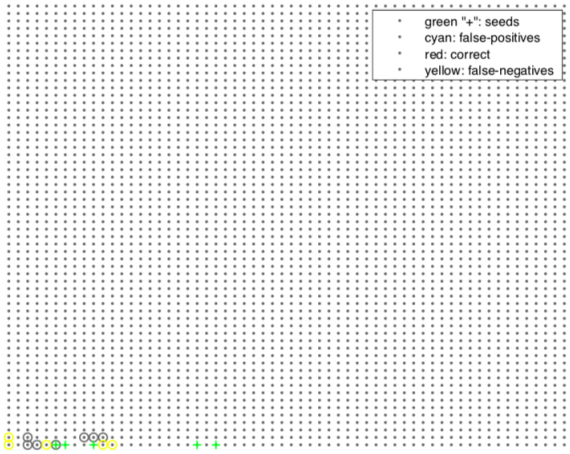

Figure 13: COVID spread on a 25 node subgraph for South Korea Dataset

'infected_by' of the dataset. We create an undirected edge (n_1, n_2) such that n_2 infects n_1 because they must have interacted with each other in order to spread the infection among themselves. We first consider the entire dataset, but we observe a complex relationship between the nodes. Hence, we begin our experimentation with a smaller subset of the contact network and observe the performance of NETFILL as we increase the number of nodes, n. We start with n=25 and consider all the nodes to be infected. Hence, the set of actual infections, D, consists of all 25 nodes. We randomly sample 6 nodes from D to create a set of reported infections, SD. Figure 13shows the result for n=25. We don't get desirable results for this sub-graph because of its small size. Hence, we run NETFILL on a subgraph with 118 nodes shown in figure 9. Due to the sparse nature of the contact network and the small size of the individual connected components, we do not obtain desirable results. In fact, in this case, the NETFILL algorithm fails during MDL calculation.

7 CONCLUSION AND DISCUSSION

In this work, we consider the problem of reconstructing the spread of the Covid-19 pandemic in South Korea. We use CuLT [6] to map this reconstruction task into a directed Steiner-tree problem and recover the flow of the spread, including identifying the missing nodes in the trace and the seed nodes. We preprocess the DS4C: South Korea dataset to construct a temporal network consisting of interactions between the Covid-19 patients and a sample of initial reported infections. We feed this information into CuLT and successfully recover the trace of the infection. We further sample out a subset of the nodes, recover the flow of spread, and compare it with the ground truth (all the reported infections). We analyze its performance using various metrics such as MCC, accuracy, and precision. Apart from CuLT, we also use other methods such as NETFILL [8] on similar datasets and analyze their performance.

REFERENCES

[1] UpCode Academy. 2020. Dashboard of the COVID-19 Virus Outbreak in Singapore. (2020). https://www.againstcovid19.com/singapore/dashboard
[2] Peter D. Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press Books, Vol. 1. The MIT Press. https://ideas.repec.org/b/mtp/titles/0262072815.html
[3] Jang S. Lee W. Lee J. K. Jang D. H. Kim, J. [n. d.]. *DS4C Patient Policy Province Dataset: a Comprehensive COVID-19 Dataset for Causal and Epidemiological Analysis.* arXiv:cmu.edu/dietrich/causality/CameraReadys-accepted%20papers/55%5CCameraReady%5Cpaper.pdf https://www.kaggle.com/kimjihoo/coronavirusdataset
[4] Theodoros Lappas, Evimaria Terzi, Dimitrios Gunopulos, and Heikki Mannila. 2010. Finding effectors in social networks. *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010).
[5] B. Aditya Prakash, Jilles Vreeken, and Christos Faloutsos. 2012. Spotting Culprits in Epidemics: How Many and Which Ones?. In *2012 IEEE 12th International Conference on Data Mining*. 11–20. https://doi.org/10.1109/ICDM.2012.136
[6] Polina Rozenshtein, Aristides Gionis, B. Aditya Prakash, and Jilles Vreeken. 2016. Reconstructing an Epidemic Over Time. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1835–1844. https://doi.org/10.1145/2939672.2939865
[7] Eldar Sadikov, Montserrat Medina, Jure Leskovec, and Hector Garcia-Molina. 2011. Correcting for missing data in information cascades. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 55–64.
[8] Shashidhar Sundareisan, Jilles Vreeken, and B. Aditya Prakash. [n. d.]. *Hidden Hazards: Finding Missing Nodes in Large Graph Epidemics.* 415–423. https://doi.org/10.1137/1.9781611974010.47 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611974010.47