# ShareTrace: An Iterative Message Passing Algorithm for Efficient and Effective Disease Risk Assessment on an Interaction Graph

Erman Ayday
Case Western Reserve University and Bilkent University
erman.ayday@case.edu

Youngjin Yoo
Case Western Reserve University
Cleveland, Ohio, USA
youngjin.yoo@case.edu

Anisa Halimi
Case Western Reserve University
Cleveland, Ohio, USA
anisa.halimi@case.edu

## ABSTRACT

We propose a novel privacy-preserving COVID-19 risk assessment algorithm that can make a fundamental contribution to the development of the next generation resilient public health and health care systems. The proposed algorithm, ShareTrace, uses a hyperlocal interaction graph to capture direct and indirect physical interactions among users. Combining user-reported symptoms that are propagated through the hyperlocal interaction graph via a novel message passing algorithm, ShareTrace is able to pick up early warning signals based on the combination of interactions with others and symptoms. The proposed algorithm is inspired by the belief propagation algorithm and iterative decoding of low-density parity-check codes over factor graphs. Our evaluation on synthetic data shows the efficiency and efficacy of the proposed solution.

## CCS CONCEPTS

• **Applied computing** → **Consumer health**; • **Security and privacy** → **Privacy-preserving protocols**; • **Networks** → **Mobile networks**.

## KEYWORDS

COVID-19; digital contact tracing; belief-propagation algorithm; privacy; hyperlocal interaction graph

**ACM Reference Format:**
Erman Ayday, Youngjin Yoo, and Anisa Halimi. 2021. ShareTrace: An Iterative Message Passing Algorithm for Efficient and Effective Disease Risk Assessment on an Interaction Graph. In *12th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '21), August 1–4, 2021, Gainesville, FL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3459930.3469553

## 1 INTRODUCTION

Contact tracing has emerged as an integral tool for containment during an epidemic, such as the current novel coronavirus (COVID-19) crisis. As demonstrated in many countries, smartphone-based digital contact tracing (i.e., proximity tracing) solutions have emerged as a powerful tool to assist government and healthcare authorities to rapidly respond to the public health crisis [3, 4, 8, 17, 19, 21]. While more efficient and scalable than traditional manual contact tracing, existing proximity-based (decentralized) contact tracing solutions (e.g., the ones using Apple and Google's framework [4]) suffer potential shortfalls, mainly in their effectiveness. Particularly, their functionality is severely limited by the inherent architectural design choice that these solutions use to protect user privacy. Most critically, as these approaches rely only on direct contact history, they can be too slow in a highly viral epidemic, like COVID-19.

We believe that using an interaction graph including the direct and indirect contacts of the individuals (rather than only considering direct contact histories of the individuals) for contact tracing is essential for timely and effective containment. On the other hand, contact tracing using an interaction graph is challenging due to (i) its high cost, as it requires to make inference on a large connected graph with a high number of nodes (e.g., individuals); and (ii) privacy concerns, as it requires access to contact histories of the users in order to construct the interaction graph. In this paper, addressing both of these challenges, we propose a distributed algorithm, ShareTrace, that provides contact tracing on an interaction graph efficiently and in a privacy-preserving way.

The proposed algorithm stems from the prior success of iterative message passing algorithms, such as belief propagation [13, 15], in decoding of Low-Density Parity-Check (LDPC) codes [22], reputation management [7], ad-hoc networks [6], and privacy risk quantification [12]. These algorithms rely on graph-based representations of an inference problem, where inference (e.g., decoding LDPC codes or computing reputation scores of parties) can be viewed as message passing between the nodes in the graph. For decoding of LDPC codes, such algorithms are shown to perform at error rates near what can be achieved by the optimal scheme, maximum likelihood decoding, while requiring far less computational complexity. Therefore, we believe that significant benefits offered by these iterative message passing algorithms can be tapped in to benefit efficient, privacy-preserving, and effective contact tracing.

There exists some works that use graphical models to monitor the spread of a virus by reverse analysing its spread. In contrast, our objective is to compute the "exposure risk probability" of each individual due to (i) their symptom risk probabilities (due to their symptoms) and (ii) their contacts with other individuals. Here, risk probability represents the probability that the test result of an individual will be positive (in case of a test at that instant). We formulate this as an inference problem that involves computing the marginal distributions of the exposure risk probabilities from the global joint probability distribution function of many variables. However, such marginalization is challenging when the system includes a high number of parties (e.g., an interaction graph may include millions of individuals). Furthermore, constructing the entire interaction graph


*BCB '21, August 1–4, 2021, Gainesville, FL, USA*

ACM ISBN 978-1-4503-8450-6/21/08…$15.00
https://doi.org/10.1145/3459930.3469553

at a central server has serious privacy implications for the parties, since sensitive information about individuals (such as their location patterns) can be inferred from an interaction graph. The key role of the message passing-based algorithms (e.g., belief propagation) is that we can use them to compute marginal distributions of the exposure risk probabilities with a complexity that grows only linearly with the number of total contacts in the system. The proposed iterative message passing algorithm, ShareTrace, represents the interaction graph as a factor graph and it efficiently and accurately computes the exposure risk probabilities of individuals on this factor graph. ShareTrace's distributed nature (without compromising from accuracy) better protects privacy by eliminating the need to collect the contact information of individuals at a centralized server. Our evaluation results on simulated data show that ShareTrace provides much quicker and more effective notifications to individual users compared to other proximity-based contact tracing solutions.

## 2 RELATED WORK

The most prominent automated contact tracing that is being used is the one proposed by Apple and Google [4]. Some popular existing apps (e.g., SwissCovid [21], CoronaWarn) are based on this solution. In such apps, Bluetooth signals are used to identify the contacts between users. Then, when a user is diagnosed positive, this information is broadcasted to all the other users in a privacy-preserving way and only the users who were in touch with the infected patient get a warning. Some other works [5, 10, 18, 20, 23, 24] use different cryptographic components, such as private set intersection, trusted execution environment, secure multiparty computation, blockchain approach, and zero-knowledge to achieve the same goal with stronger privacy guarantees. Different from existing solutions, ShareTrace computes the exposure risk probabilities of the users using the entire interaction graph in an efficient and privacy-preserving way, and hence it is more effective to control the spread of the virus compared to other solutions.

## 3 PROPOSED SOLUTION

We assume that smartphones locally and continuously generate Bluetooth signals and each user's contacts (generated similar to the existing contact tracing solutions [4]) are stored in the corresponding user's local device or personal cloud (we discuss the potential real-life deployment in Section 5.2). Users use the IDs of their contacts to establish the links in the proposed distributed solution (in Section 3.3). We assume that users locally update their symptoms using a symptom risk calculation algorithm (e.g., [14]). "Symptom risk probabilities" of users are computed in their local devices (i.e., symptom risk probability is the probability that the test result of a user will be positive [14]). The computed symptom risk probabilities (over the last 14 days, which is the quarantine period based on CDC recommendations), along with the contact vector of a user $i$, $C_i$ (includes the contact of the user over the previous 14 days), are stored in their local devices or personal clouds.

In the following, for simplicity, we describe how the proposed ShareTrace algorithm works in a centralized setting (where symptom risk probabilities and contact vectors of individuals are collected at a central server). Then, in Section 3.3, we discuss how the proposed algorithm works in a distributed setting.

### 3.1 ShareTrace: Message Passing-Based Contact Tracing on an Interaction Graph

The goal of ShareTrace is to compute the posterior risk probabilities of the users, given the prior risk probability for each user (computed using the user's symptoms or the diagnosis). Our proposed iterative message passing algorithm is inspired by earlier work on the improved iterative decoding algorithm of LDP codes in the presence of stopping sets [16, 22] and our earlier work on the use of belief propagation algorithm for reputation management [7] and privacy risk quantification [12]. For instance, in iterative decoding of LDPC codes, every check-vertex (in the graph representation of the code) has some opinion of what the value of each bit-vertex should be. The iterative decoding algorithm would then analyze the collection of these opinions to decide, at each iteration, what value to assign for the bit-vertex under consideration. Once the values of the bit-vertices are estimated, in the next iteration, those values are used to determine the satisfaction of the check-vertex values. The novelty of this work stems from the observation that a similar approach can be adapted to determine an individual's exposure risk probability for an infectious disease considering his/her symptoms, behaviors, and interactions with others.

We represent the set of users in the system as $\mathbb{S}$, where $|\mathbb{S}| = s$. Let $r_j$ be a random variable representing the exposure risk probability of user $j$ ($j \in \mathbb{S}$). Similar to symptom risk probability, exposure risk probability also represents the probability that the test result of the user will be positive. Let also $\mathbb{R} = \{r_j : j \in \mathbb{S}\}$ be the collection of variables representing the exposure risk probabilities of the users in the system. We assume $\mathbb{T}$ is an $s \times s$ matrix keeping the contact times between the users over the last 14 days (an entry may have multiple contact times if there are multiple contacts between two users over the last 14 days). Also, $\mathbb{D}$ is another $s \times s$ matrix keeping the contact duration between the users over the last 14 days (an entry may have multiple contact durations if there are multiple contacts between two users over the last 14 days). Finally we let $\mathbb{L}$ be a vector (of size $s$) representing the symptom risk probabilities of the users.

Then, the problem of computing the exposure risk probabilities can be viewed as finding the marginal probability distributions of each variable in $\mathbb{R}$, given the observed data (i.e., symptoms) and the interactions between the users. We formulate the problem by considering the global function $p(\mathbb{R}|\mathbb{T}, \mathbb{D}, \mathbb{L})$, which is the joint probability distribution function of the variables in $\mathbb{R}$ given the contact times and contact durations between the users. Then, each marginal probability function $p(r_j|\mathbb{T}, \mathbb{D}, \mathbb{L})$ may be obtained as follows:

$$p(r_j|\mathbb{T}, \mathbb{D}, \mathbb{L}) = \sum_{\mathbb{R}\setminus\{r_j\}} p(\mathbb{R}|\mathbb{T}, \mathbb{D}, \mathbb{L}), \quad (1)$$

where $\mathbb{R}\setminus\{r_j\}$ implies all variables in $\mathbb{R}$ except $r_j$.

The number of terms in (1) grows exponentially with the number of variables (users), making the computation infeasible for large-scale systems. Thus, inspired by earlier work on iterative message passing algorithms in decoding of LDPC codes [22], reputation management [7], and privacy risk quantification [12], we propose to factorize (1) to local functions using a factor graph and utilize a message passing algorithm to calculate the marginal probability distributions in linear complexity. A factor graph is a bipartite graph containing two sets of nodes (corresponding to variables and

factors) and edges incident between two sets [13]. In our interaction graph, each factor node represents the contact (interaction) between two users $i$ and $j$, and hence the degree of each factor node $c(i,j)$ is 2. On the other hand, each variable node $j$ represents a random variable representing the exposure risk probability $r_j$ of each user. In Figure 1, we show the factor graph representation corresponding to an interaction graph with 4 users.

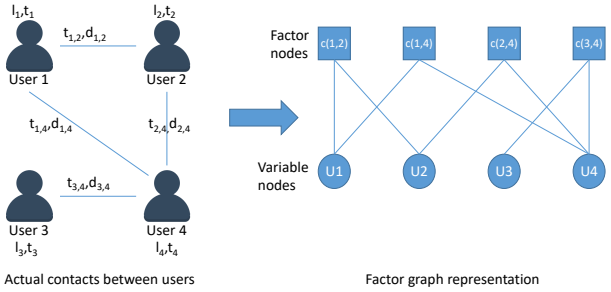



**Figure 1: Factor graph representation of the interaction between 4 users.**

Next, we suppose that the global function $p(\mathbb{R}|\mathbb{T},\mathbb{D},\mathbb{L})$ factors into products of several local functions, each having a subset of variables from $\mathbb{R}$ as arguments as follows:

$$p(\mathbb{R}|\mathbb{T},\mathbb{D},\mathbb{L}) = \frac{1}{Z}\prod_{i,j\in\mathbb{S}} f_{i,j}(\mathcal{R}_{i,j},\mathbb{T}_{i,j},\mathbb{D}_{i,j},\mathbb{L}_{i,j}), \tag{2}$$

where $Z$ is the normalization constant and $\mathcal{R}_{i,j}$ is a subset of $\mathbb{R}$ including $r_i$ and $r_j$. Also, $\mathbb{T}_{i,j}$ keeps the contacts times, $\mathbb{D}_{i,j}$ keeps the contact durations, and $\mathbb{L}_{i,j}$ keeps the symptom risk probabilities of users $i$ and $j$. Hence, each factor node $c(i,j)$ is associated with a local function $f_{i,j}$ and each local function $f_{i,j}$ represents the transmission of the disease risk between two users $i$ and $j$ (given the exposure and symptom risk probabilities of the users and the duration of their contact).

We now introduce the messages between the factor and the variable nodes to compute the marginal distributions using the proposed message passing algorithm. We denote the messages from the variable nodes to the factor nodes and from the factor nodes to the variable nodes as $\mu$ and $\lambda$, respectively. The message $\mu^{(\nu)}_{i\to c(i,j)}(\vec{r_i})$ is a vector of size $m$, and it includes the maximum $m$ values of $r_i$, at the $\nu$−th iteration. On the other hand, $\lambda^{(\nu)}_{c(i,j)\to i}(r_i)$ denotes the value of $r_i$, at the $\nu$−th iteration given the contact details between users $i$ and $j$ and the exposure risk probability of user $j$ at that iteration.

A factor node $c(i,j)$ (whose neighbors are variable nodes $i$ and $j$) forms its message to a variable node $i$ as follows. For simplicity, we assume users $i$ and $j$ have met only once, and hence $\mathbb{T}_{i,j} = t_{i,j}$ and $\mathbb{D}_{i,j} = d_{i,j}$ (in case of multiple interactions, $\mathbb{T}_{i,j}$ and $\mathbb{D}_{i,j}$ have multiple values, representing these interactions). To form its message, factor node $c(i,j)$ uses (i) contact time between user $i$ and user $j$ ($t_{i,j}$), (ii) duration of the contact, $d_{i,j}$, and (iii) the message it received from variable node $j$ in the $(\nu-1)$th iteration, $\mu^{(\nu-1)}_{j\to c(i,j)}(\vec{r_j}) = ((r_j^1,t_j^1),(r_j^2,t_j^2),(r_j^3,t_j^3),\ldots,(r_j^m,t_j^m))$. $\mu^{(\nu-1)}_{j\to c(i,j)}(\vec{r_j})$ includes the top $m$ risks that variable node $j$ received from its neighbors (neighboring factor nodes) in the $(\nu-1)$th iteration along with the time $t$ associated with those risks. Factor node $c(i,j)$ first selects the entry $(r_j^k,t_j^k)$ from $\mu^{(\nu-1)}_{j\to c(i,j)}(\vec{r_j})$ which has the highest risk probability ($r_j^k$) that is associated with a time before $t_{i,j}$ (i.e., $t_j^k \le t_{i,j}$). Then, the factor node computes the risk probability of user $i$ due to user $j$ based on $r_j^k$ and $d_{i,j}$: If $d_{i,j}$ is greater than a threshold (15 minutes according to CDC guidelines), computed risk of $i$ is equal to $r_j^k \cdot \tau$ (where $\tau$ is the transmission probability and it is equal to 0.8 according to CDC guidelines [9]). otherwise computed risk of $i$ is equal to 0. If the user $i$ and $j$ have more than one contact (i.e., if they met more than once), we compute the risk for each contact like above and send the highest risk to the variable node $i$. More formally, the message from the factor node $c(i,j)$ to the variable node $i$ at the $\nu$−th iteration is formed as

$$\lambda^{(\nu)}_{c(i,j)\to i}(r_i) = \max_{\eta} f_{i,j}(r_i,\mu^{(\nu-1)}_{j\to c(i,j)}(\vec{r_j})(\eta),\mathbb{T}_{i,j},\mathbb{D}_{i,j},\mathbb{L}_{i,j})\mu^{(\nu-1)}_{j\to c(i,j)}(\vec{r_j})(\eta), \tag{3}$$

where, $\mu^{(\nu-1)}_{j\to c(i,j)}(\vec{r_j})(\eta)$ represents the $\eta$−th element in $\mu^{(\nu-1)}_{j\to c(i,j)}(\vec{r_j})$. This computation must be performed for every neighbor of each factor nodes. This finishes the first half of the $\nu$−th iteration.

The risk probability of a user is typically determined based on the highest-risk event the user was exposed to in the last 14 days. For instance, if the user had 2 contacts and the risk probability due to one of these contacts is high, then the risk probability of the user is determined based on this high-risk contact. Thus, as opposed to traditional belief propagation (which multiplies the messages at the variable nodes in order to form a new message), our proposed message passing-based algorithm uses a "max" function at the variable nodes. Therefore, the message from a variable node $i$ to a factor node $c(i,j)$ in iteration $\nu$ is computed as follows. Variable node $i$ receives messages from all of its neighboring factor nodes in the $(\nu-1)$th iteration. To form its message to its neighboring factor node $c(i,j)$, it creates a vector $\mu^{(\nu)}_{i\to c(i,j)}(\vec{r_i})$ (including the highest $m$ risks received along with their corresponding times, as described before) and sends the entire vector to $c(i,j)$. Note that to avoid self-bias, $\mu^{(\nu)}_{i\to c(i,j)}(\vec{r_i})$ formed for factor node $c(i,j)$ does not include the risk ($r_i$) received from factor node $c(i,j)$ in the previous iteration. The size of vector $\mu^{(\nu)}_{i\to c(i,j)}(\vec{r_i})$ is $m$, which is a design parameter. For the most accurate results, we set $m$ as long as the total contact history of user $i$.

At the beginning of the algorithm, the exposure risk probabilities of all users are set to their symptom risk probability. Algorithm starts at the variable nodes and each variable node sends (i) its symptom risk probabilities and (ii) corresponding times (at which the symptom risk probabilities are computed) to all its neighboring factor nodes. At the end of each iteration, each variable node computes its exposure risk probability as the maximum of (i) risk probabilities it receives from its neighboring factor nodes and (ii) its symptom risk probabilities. The algorithm converges either when the computed risk probabilities at the variable nodes stop changing (i.e., the change in the users' risks between two consecutive iterations is less than 0.00001) or after a predefined number of iterations (100 iterations). In the experiments shown in Section 4, the algorithm takes at most 5-6 iterations to converge.

## 3.2 Toy Example

Here, we provide a toy example for the proposed message passing-based risk inference algorithm for the users in the interaction graph in Figure 1. For simplicity, we assume each user has one symptom risk probability and $l_i$ represents the symptom risk probability of user $i$ (computed from its symptoms and/or diagnosis, as discussed before). If users have more than one symptom risk probabilities (computed at different times over the last 14 days), they are integrated to the algorithm similarly. Also, let $t_i$ be the last update time for the symptom risk probability of user $i$ (last time the user updated its symptoms and/or diagnosis).

First, we construct the interaction graph (in Figure 1) from the contact histories and the corresponding time stamps of the users. $t_{i,j}$ and $d_{i,j}$ represent the time and duration of the contact between users $i$ and $j$, respectively. Next, we construct the factor graph (for the proposed message passing-based algorithm) as in Figure 1. In the following, we focus on the message exchange between user $U4$ and its neighbors for the first 2 iterations of the algorithm. For simplicity of the presentation, we rename the factor nodes as follows: $c(1,4) = a$, $c(2,4) = b$, $c(3,4) = c$.

As discussed, the algorithm starts from the variable nodes. Thus, as the first message, variable node $U4$ sends $(l_4, t_4)$ to all its neighboring factor nodes ($a$, $b$, and $c$). Factor node $c$ receives $(l_3, t_3)$ from $U3$ and $(l_4, t_4)$ from $U4$ in the first iteration. To generate its message to $U4$, factor node $c$ uses $(l_3, t_3)$, $t_{3,4}$, and $d_{3,4}$. If $t_3$ is before $t_{3,4}$ (i.e. if the contact between $U3$ and $U4$ happened after $U3$'s recent symptom risk update), $U4$'s exposure risk probability may be updated due to $U3$. Similarly, if $t_3$ is after $t_{3,4}$, $U4$'s exposure risk probability will not be affected due to $U3$. We represent the message from $c$ to $U4$ as $\lambda^{(1)}_{c\to U4}(r_{U4}) = (r_{4,3}, t_{3,4})$. Thus, if $t_3$ is before $t_{3,4}$ and $d_{3,4}$ is more than 15 minutes, $r_{4,3} = l_3 \cdot \tau$ (where $\tau$ is the transmission probability, as discussed before) and if $t_3$ is after $t_{3,4}$, $r_{4,3} = 0$. Factor node $c$ also generates its message for $U3$ in a similar way. This completes the first iteration of the algorithm.

As a result of the first iteration, $U4$ receives messages $\lambda^{(1)}_{a\to U4}(r_{U4})$, $\lambda^{(1)}_{b\to U4}(r_{U4})$, and $\lambda^{(1)}_{c\to U4}(r_{U4})$ from its neighbors. To form its next message to factor node $c$, $U4$ generates its message $\mu^{(2)}_{U4\to c}(\vec{r_{U4}})$ as $\mu^{(2)}_{U4\to c}(\vec{r_{U4}}) = ((\lambda^{(1)}_{a\to U4}(r_{U4}), t_a), (\lambda^{(1)}_{b\to U4}(r_{U4}), t_b), (l_4, t_4))$, where $t_a$ and $t_b$ are the times associated to the risk probabilities sent by factor nodes $a$ and $b$, respectively. After receiving $\mu^{(2)}_{U4\to c}(\vec{r_{U4}})$, processing at the factor node $c$ in the second iteration is similar to before. To calculate the final risk probability (when the algorithm converges), variable node $U4$ receives all messages from its neighbors and it picks the one with the maximum risk probability and that becomes the posterior risk probability of user $U4$.

## 3.3 Extension to Distributed Architecture

The algorithm in Section 3.1 can easily be extended for a distributed setting to provide a privacy-preserving risk calculation algorithm for the users. To achieve this, the factor graph (in Figure 1) can be structured between the users (either using users' end devices or between their local clouds, as discussed more in Section 5.2) in a distributed way as in Figure 2. For this, we assign duplicate factor nodes in the graph. For instance $c(1,2)$ and $c(2,1)$ in the figure represent the same interaction between variable nodes $U1$ and $U2$.

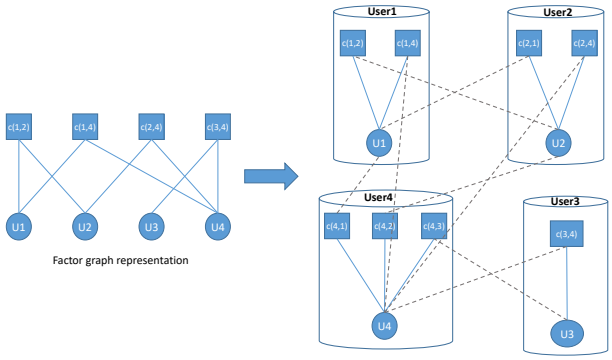



**Figure 2: Distributed interaction graph between 4 users. The factor graph (on the left) is constructed in the user spaces in a distributed way (on the right). The dashed lines represent the communication between the users during the proposed message passing algorithm.**

For message passing, each variable node sends its messages to both the factor nodes within its own user space and also to the ones (duplicate ones) that are in other user's space. This will allow us to support message passing on an interaction graph in a distributed and privacy-preserving way. To reduce the amount of information that is provided from one user's space to another, we will explore utilizing the differential privacy concept [11] in future work.

# 4 EVALUATION

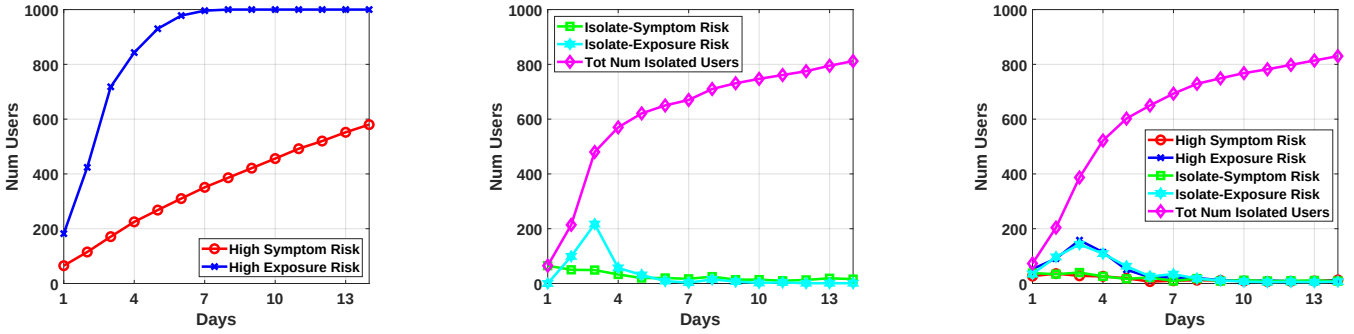
We evaluate the proposed message passing-based algorithm in terms of its efficacy and compare it with the contact-tracing algorithms that adopt Apple/Google (A/G)'s framework [4]. For this evaluation, we first create synthetic data to simulate the mobility patterns (and hence contacts) and symptoms of individuals.

## 4.1 Data Generation

Our synthetic dataset consists of 1000 users ($s = 1000$) that interact with each-other during a period of 14 days. To simulate the fact that in real-life some users have more contacts than others, we used a power law distribution ($f(x,a) = ax^{a-1}$). We determined the number of contacts for each user via the power law distribution and then generated their contact vectors randomly. We generated 54376 interactions between all the 1000 users using $a = 0.05$. We also tried other "$a$" values in the range $[0.0005 - 0.1]$ and obtained similar results for the efficacy and in terms of the advantage of ShareTrace compared to A/G's framework. For each user, we generated a random symptom vector for each day (using the symptoms in the symptom risk calculation algorithm [14]). We assume that each day, symptoms either remain the same or new ones appear, but they do not disappear. Thus, the symptom risk probabilities of each user only increases during this 14-day period (to represent the worst case scenario).

## 4.2 Efficacy

Here, we study the efficacy of ShareTrace to stop the spread of the virus. We assume that all users update their symptoms every day. Figure 3(a) shows the number of users that have a high symptom risk probability (higher than 0.5) and a high exposure risk probability (higher than 0.5) each day if none of the users isolate. As




(a) Number of users with high symptom and exposure risk probabilities

(b) All users with a high symptom or exposure risk probability isolate each day

(c) Half of the users with a high symptom or exposure risk probability isolate each day

**Figure 3: Efficacy of ShareTrace to stop the spread of the virus.**

the days pass, the virus is spread through the individuals and the number of users with high symptom and/or exposure risk probability increases. In Figure 3(b), we show that if users isolate due to their high symptom risk probability or their high exposure risk probability (computed using ShareTrace), the spread of the virus would stop in a few days. As shown in the figure, the number of users who isolate each day significantly decreases after day 4, which means that after day 4, number of high-risk probability users is low, and hence the spread can be controlled. Since in real-life, it is not realistic to assume that all users would isolate (quarantine) themselves as a result of high risk probability (computed using ShareTrace), we also did the same evaluation when only half of the users isolate themselves as a result of their high risk probability. In Figure 3(c), we show that even if half of the users that have a high symptom or exposure risk probability would isolate each day, still the spread of the virus is controlled in 7 days.

## 4.3 Comparison with A/G

To compare the proposed algorithm to traditional contact tracing, we made a few assumptions. We assume that all the users are actively using the apps (both ShareTrace and apps using A/G framework). If a user has a symptom risk probability higher than 0.5, we assume that they get tested. The outcome of the test is probabilistically modelled based on users' symptom risk probability on the day of the test (i.e., symptom risk probability is the probability that the test result will be positive [14]). If a user's diagnosis is positive, then the symptom risk probability of the user becomes 1 and that user isolates. In traditional contact tracing, users' direct contacts in the last 14 days are informed immediately and they get tested the next day. In ShareTrace, the exposure risk probabilities of the users (directly and indirectly contacted users) are updated via the proposed message passing-based algorithm and the users that have an exposure risk probability higher than 0.5 are tested the next day. In favor of the A/G framework, we assume that (i) the test results are 100% accurate and (ii) test results are immediately available. Also, all users are tested each day in the background to determine their real diagnosis regardless of them meeting the conditions of getting tested. In this way, we know the total number of users that are positive (i.e., that have COVID-19).

Figure 4 shows the total number of users with positive diagnosis per day (based on the test results done in the background), the number of users with positive diagnosis that were detected by traditional contact tracing schemes (i.e., apps using A/G framework), and the ones that were detected by ShareTrace. Our results show that the proposed ShareTrace algorithm can detect users that got infected with COVID-19 faster than the contact tracing schemes using A/G framework. It has an advantage of 12% over A/G framework on the third day and this advantage goes up to 25% on the fifth day. The advantage of ShareTrace increases as the number of users' daily contacts increases. Even if less users get tested per day (i.e., not all the users that have a symptom risk probability or exposure risk probability higher than 0.5), ShareTrace still has an advantage over A/G framework since users can proactively decide to isolate based on their symptom and exposure risk probability. We believe that this gap will be higher under more realistic settings, where tests are not available to everyone instantly, when the test results are not immediately available, and when the test results are not 100% accurate because users can decide to isolate based on the symptom or exposure risk probability.

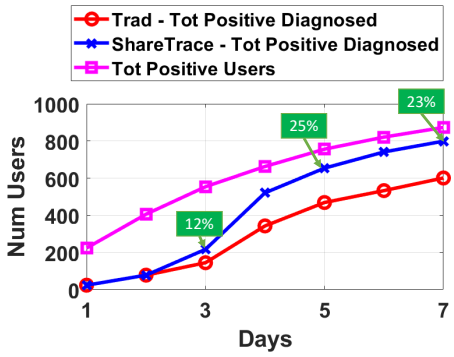



**Figure 4: Comparison of ShareTrace to traditional contact tracing (i.e., apps using A/G framework).**

## 5 DISCUSSION

Here, we briefly discuss the real-life deployment and scalability of the proposed algorithm.

### 5.1 Properties of the Proposed Algorithm

The proposed algorithm does not heavily rely on the graph structure or contact patterns between individuals. In simulations, we used the power law distribution (with different parameters) just to provide a proof of concept implementation. The proposed distributed message passing-based inference algorithm works with any other graphical model and contact distribution. Besides, the superiority of the proposed algorithm with respect to the baseline still holds under different contact distribution models (e.g., in dense and sparse populations); only the efficiency of our model changes depending on the contact distributions (as discussed in Section 5.3).

### 5.2 Real-life Deployment

The proposed distributed algorithm (in Section 3.3) can be implemented in real-life by using existing architectures that provide decentralized personal data accounts (PDAs). For instance, HAT Microserver (HAT) [2] is one way to implement such PDAs. The HAT is a personal data server, where the user is the legal and functional owner of the data and confers data rights to users. The HAT Microservers sit within the HATDeX platform operated by Dataswift [1], and the platform enables data storage, computation, and contracting to different applications, ShareTrace being one of them. A PDA in HAT ecosystem is, therefore, a cloud-based technology for data rights, mobility, and control for individuals, making it fully interoperable with all other applications. PDAs also enable individuals to install pre-trained tools (supplied by organizations or data scientists) to generate "edge" analytics and private AI insights, and hence the proposed distributed message passing-based algorithm can be implemented in real-life between the PDAs of the users. The distributed nature of the proposed algorithm makes it privacy-preserving (which is a crucial property for contact tracing) and this is also a significant add-on with respect to other graph-based algorithms to track the spread of a virus.

### 5.3 Complexity

If we have $s$ users in the system, the proposed algorithm will have $s$ variable nodes and at most $s^2$ factor nodes. Thus, the computational complexity of the proposed algorithm is $O(s^2)$ in the worst case. However, this worst-case scenario occurs only when each user contacts every other user (which is an unrealistic scenario in practice). Therefore, in practice, the interaction graph is a sparse one, and hence the complexity of the proposed message passing-based algorithm is close to linear with the number of users in the network. Furthermore, when the algorithm is implemented in a distributed way (as in Section 3.3), the complexity of the algorithm per user can be represented as the total number of unique contacts of a user over 14 days (which is typically significantly less than $s$).

## 6 CONCLUSION

In this paper, we have proposed a novel message passing-based COVID-19 risk assessment algorithm, ShareTrace, on an interaction graph. ShareTrace is an iterative algorithm on an interaction graph motivated by prior success on message passing techniques and belief propagation algorithms for decoding LDPC codes, reputation management, and privacy risk quantification. We have also shown how the proposed graph-based algorithm can be implemented in a distributed setting to provide a privacy-preserving risk assessment tool. Our results show that ShareTrace significantly outperforms traditional contact-tracing algorithms in terms of its efficacy.

## ACKNOWLEDGEMENT

This work was supported in part by LG TCA (Technology Center America). The views expressed are those of the authors only.

## REFERENCES

[1] [n.d.]. *Dataswift.* https://www.dataswift.io/
[2] [n.d.]. *Hub-of-All-Things.* https://www.hubofallthings.com/
[3] [n.d.]. *PEPP-PT Documentation.* https://github.com/pepp-pt/pepp-pt-documentation
[4] [n.d.]. *Privacy-Preserving Contact Tracing.* https://www.apple.com/covid19/contacttracing
[5] [n.d.]. *SafeTrace.* https://github.com/enigmampc/SafeTrace
[6] Erman Ayday and Faramarz Fekri. 2011. An iterative algorithm for trust management and adversary detection for delay-tolerant networks. *IEEE Transactions on Mobile Computing* 11, 9 (2011), 1514–1531.
[7] Erman Ayday and Faramarz Fekri. 2012. Iterative Trust and Reputation Management Using Belief Propagation. *IEEE Transactions on Dependable and Secure Computing* 9, 3 (May 2012), 375–386.
[8] Justin Chan, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Sudheesh Singanamalla, Jacob Sunshine, et al. 2020. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing. *arXiv preprint arXiv:2004.03544* (2020).
[9] S County, L Hamner, P Dubbel, I Capron, A Ross, A Jordan, J Lee, J Lynn, and A Ball. 2020. High SARS-CoV-2 attack rate following exposure at a choir practice. *Morb Mortal Wkly Rep High* 69, 19 (2020), 606–610.
[10] Didem Demirag and Erman Ayday. 2020. Tracking and controlling the spread of a virus in a privacy-preserving way. *arXiv preprint arXiv:2003.13073* (2020).
[11] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation.* Springer, 1–19.
[12] Mathias Humbert, Erman Ayday, Jean-Pierre Hubaux, and Amalio Telenti. 2013. Addressing the concerns of the lacks family: quantification of kin genomic privacy. In *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security.* 1141–1152.
[13] F. Kschischang, B. Frey, and H. A. Loeliger. 2001. Factor Graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* (2001).
[14] Cristina Menni, Ana M Valdes, Maxim B Freidin, Carole H Sudre, Long H Nguyen, David A Drew, Sajaysurya Ganesh, Thomas Varsavsky, M Jorge Cardoso, Julia S El-Sayed Moustafa, et al. 2020. Real-time tracking of self-reported symptoms to predict potential COVID-19. *Nature medicine* (2020), 1–4.
[15] J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc.
[16] H. Pishro-Nik and F. Fekri. 2007. Results on Punctured Low-Density Parity-Check Codes and Improved Iterative Decoding Techniques. *IEEE Transactions on Information Theory* 53, 2 (Feb. 2007), 599–614.
[17] Ramesh Raskar, Isabel Schunemann, Rachel Barbar, Kristen Vilcans, Jim Gray, Praneeth Vepakomma, Suraj Kapa, Andrea Nuzzo, Rajiv Gupta, Alex Berke, et al. 2020. Apps gone rogue: Maintaining personal privacy in an epidemic. *arXiv preprint arXiv:2003.08567* (2020).
[18] Leonie Reichert, Samuel Brack, and Björn Scheuermann. 2020. Privacy-Preserving Contact Tracing of COVID-19 Patients. *IACR Cryptol. ePrint Arch.* (2020), 375.
[19] Ronald L Rivest, Jon Callas, Ran Canetti, Kevin Esvelt, Daniel Kahn Gillmor, Yael Tauman Kalai, Anna Lysyanskaya, Adam Norige, Ramesh Raskar, Adi Shamir, et al. 2020. The PACT protocol specification. *Private Automated Contact Tracing Team, MIT, Cambridge, MA, USA, Tech. Rep. 0.1* (2020).
[20] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. 2020. Epione: Lightweight contact tracing with strong privacy. *arXiv preprint arXiv:2004.13293* (2020).
[21] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, et al. 2020. Decentralized privacy-preserving proximity tracing. *arXiv preprint arXiv:2005.12273* (2020).
[22] Badri N Vellambi and Faramarz Fekri. 2007. Results on the improved decoding algorithm for low-density parity-check codes over the binary erasure channel. *IEEE Transactions on Information Theory* 53, 4 (2007), 1510–1520.
[23] Dong Wang and Fang Liu. 2020. Privacy Risk and Preservation in Contact Tracing of COVID-19. *CHANCE* 33, 3 (2020), 49–55.
[24] Hao Xu, Lei Zhang, Oluwakayode Onireti, Yang Fang, William Bill Buchanan, and Muhammad Ali Imran. 2020. BeepTrace: Blockchain-enabled Privacy-preserving Contact Tracing for COVID-19 Pandemic and Beyond. *arXiv preprint arXiv:2005.10103* (2020).