Chapter 1

Risk Propagation

Risk propagation is a message-passing algorithm that estimates an individual's infection risk by considering their demographics, symptoms, diagnosis, and contact with others. Formally, a *risk score* s_t is a timestamped infection probability where $s \in [0, 1]$ and $t \in \mathbb{N}$ is the time of its computation. Thus, an individual with a high risk score is likely to test positive for the infection and poses a significant health risk to others. There are two types of risk scores: *symptom scores*, or prior infection probabilities, which account for an individual's demographics, symptoms, and diagnosis (Menni et al., 2020); and *exposure scores*, or posterior infection probabilities, which incorporate the risk of direct and indirect contact with others.

Given their recent risk scores and contacts, an individual's exposure score is derived by marginalizing over the joint infection probability distribution. Naively computing this marginalization scales exponentially with the number of variables (i.e., individuals). To circumvent this intractability, the joint dis-

tribution is modeled as a factor graph, and an efficient message-passing procedure is employed to compute the marginal probabilities with a time complexity that scales linearly in the number of factor nodes (i.e., contacts).

Let $G = (V, F, E)$ be a *factor graph* where V is the set of variable nodes, F is the set of factor nodes, and E is the set of edges incident between them (Kschischang et al., 2001). A *variable node*

$$v : \Omega \to \{0, 1\}$$

is a random variable that represents the infection status of an individual, where the sample space is $\Omega = \{healthy, infected\}$ and

$$v(\omega) = \begin{cases} 0 & \text{if } \omega = healthy \\ 1 & \text{if } \omega = infected. \end{cases}$$

Thus, $p_t(v_i) = s_t$ is a risk score of the i-th individual. A *factor node*

$$f : V \times V \to [0, 1]$$

defines the transmission of infection risk between two contacts. Specifically, contact between the i-th and j-th individual is represented by the factor node $f(v_i, v_j) = f_{ij}$, which is adjacent to the variable nodes v_i, v_j. This work and Ayday et al. (2021) assume risk transmission is a symmetric function, $f_{ij} = f_{ji}$. However, it may be extended to account for an individual's susceptibility and transmissibility such that $f_{ij} \neq f_{ji}$. Figure 1.1 depicts a factor graph that

reflects the domain constraints.

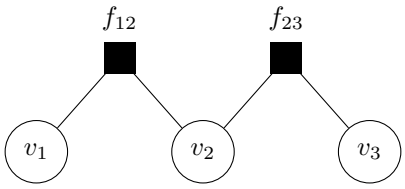

Figure 1.1: A factor graph of 3 variable nodes and 2 factor nodes.

1.1 Synchronous Risk Propagation

Ayday et al. (2021) first proposed risk propagation as a synchronous, iterative message-passing algorithm that uses the factor graph to compute exposure scores. The first input to Risk-Propagation is the set family S, where

$$S_i = \{\, s_t \mid \tau - t < T_s \,\} \in S \tag{1.1}$$

is the set of recent risk scores of the i-th individual. The second input to Risk-Propagation is the contact set

$$C = \{\, (i, j, t) \mid i \neq j, \tau - t < T_c \,\} \tag{1.2}$$

such that (i, j, t) is the *most recent* contact between the i-th and j-th individual that occurred from time t until at least time $t+\delta$, where $\delta \in \mathbb{N}$ is the *minimum contact duration*[1]. Naturally, risk scores and contacts have finite relevance, so

[1]While Ayday et al. (2021) require contact over a δ-contiguous period of time, the Centers for Disease Control and Prevention (2021) account for contact over a 24-hour period.

(1.1) and (1.2) are constrained by the *risk score expiry* $T_s \in \mathbb{N}$ and the *contact expiry* $T_c \in \mathbb{N}$, respectively. The *reference time* $\tau \in \mathbb{N}$ defines the relevance of the inputs S, C and is assumed to be the time at which RISK-PROPAGATION is invoked.

For notational simplicity in RISK-PROPAGATION, $\max X = 0$ if $X = \emptyset$.

1.1.1 Variable Message Computation

The current exposure score of the i-th individual is defined as $\max S_i$. Hence, a *variable message* $\mu_{ij}^{(n)}$ from the variable node v_i to the factor node f_{ij} during the n-th iteration is the set of maximal risk scores $R_i^{(n-1)}$ from the previous $n-1$ iterations that were not derived by f_{ij}. In this way, risk propagation is reminiscent of the max-sum algorithm; however, risk propagation aims to maximize *individual* marginal probabilities rather than the joint distribution (Bishop, 2006, pp. 411–415).

1.1.2 Factor Message Computation

A *factor message* $\lambda_{ij}^{(n)}$ from the factor node f_{ij} to the variable node v_j during the n-th iteration is an exposure score of the j-th individual that is based on interacting with those at most $n-1$ degrees separated from the i-th individual. This population is defined by the subgraph induced in G by

$$\{\, v \in V \cap F \setminus \{v_j, f_{ij}\} \mid d(v_i, v) \leq 2(n-1) \,\},$$

where $d(u, v)$ is the distance between the nodes u, v. The computation of a factor message assumes the following.

1. Contacts have a nondecreasing effect on an individual's exposure score.

2. A risk score s_t is *relevant* to the contact (i, j, t_{ij}) if $t \leq t_{ij} + \beta$, where $\beta \in \mathbb{N}$ is a *time buffer* that accounts for delayed reporting of symptom scores and contacts.

3. Risk transmission between contacts is incomplete. Thus, a risk score decays exponentially along its transmission path in G at a rate of $\log(\alpha)$, where $\alpha \in (0, 1)$ is the *transmission rate*.

To summarize, a factor message $\lambda_{ij}^{(n)}$ is the maximum relevant risk score in the variable message $\mu_{ij}^{(n)}$ (or 0) that is scaled by the transmission rate α.

Ayday et al. (2021) assume that the contact set C may contain (1) multiple contacts between the same two individuals and (2) *invalid* contacts, or those lasting less than δ time. However, these assumptions introduce unnecessary complexity. Regarding assumption 1, suppose the i-th and j-th individual come into contact q times such that $t_k < t_\ell$ for $1 \leq k < \ell \leq q$. Let M_k be the set of relevant risk scores, according to the contact time t_k, where

$$M_k = \{\, \alpha \cdot s_t \mid s_t \in \mu_{ij}^{(n)}, t \leq t_k + \beta \,\}.$$

Then $M_k \subseteq M_\ell$ if and only if $\max M_k \leq \max M_\ell$. Therefore, only the most recent contact time t_q is required to compute the factor message $\lambda_{ij}^{(n)}$. With respect to assumption 2, there are two possibilities.

1. If an individual has at least one valid contact, then their exposure score is computed over the sugraph induced in G by their contacts that define the neighborhood N_i of the variable node v_i.

2. If an individual has no valid contacts, then their exposure score is $\max S_i$ or 0, if all of their previously computed risk scores have expired.

In either case, a set C containing only valid contacts implies fewer factor nodes and edges in the factor graph G. Consequently, the complexity of Risk-Propagation is reduced by a constant factor since fewer messages must be computed.

1.1.3 Termination Detection

Risk-Propagation terminates when the normed difference between maximum exposure scores of the previous iteration and current iteration is less than the tolerance $\epsilon \in \mathbb{R}$. Note that $\mathbf{r}^{(n)}$ is the vector of maximum risk scores in the the n-th iteration such that $r_i^{(n)}$ is the i-th component of $\mathbf{r}^{(n)}$. The ℓ^1 and ℓ^∞ norms are sensible choices for detecting convergence. Ayday et al. (2021) use the ℓ^1 norm, which ensures that an individual's exposure score changed by at most ϵ after the penultimate iteration.

RISK-PROPAGATION(S, C)

1: $(V, F, E) \leftarrow$ FACTOR-GRAPH(C)
2: $n \leftarrow 1$
3: **for each** $v_i \in V$
4: $\quad R_i^{(n-1)} \leftarrow$ top K of S_i
5: $\quad r_i^{(n-1)} \leftarrow \max R_i^{(n-1)}$
6: $\quad r_i^{(n)} \leftarrow \infty$
7: **while** $\|\mathbf{r}^{(n)} - \mathbf{r}^{(n-1)}\| > \epsilon$
8: $\quad$ **for each** $\{v_i, f_{ij}\} \in E$
9: $\quad\quad \mu_{ij}^{(n)} \leftarrow R_i^{(n-1)} \setminus \{\, \lambda_{ji}^{(k)} \mid k \in [1 \,..\, n-1] \,\}$
10: $\quad$ **for each** $\{v_i, f_{ij}\} \in E$
11: $\quad\quad \lambda_{ij}^{(n)} \leftarrow \max \{\, \alpha \cdot s_t \mid s_t \in \mu_{ij}^{(n)}, t \leq t_{ij} + \beta \,\}$
12: $\quad$ **for each** $v_i \in V$
13: $\quad\quad R_i^{(n)} \leftarrow$ top K of $\{\, \lambda_{ji}^{(n)} \mid f_{ij} \in N_i \,\}$
14: $\quad$ **for each** $v_i \in V$
15: $\quad\quad r_i^{(n-1)} \leftarrow r_i^{(n)}$
16: $\quad\quad r_i^{(n)} \leftarrow \max R_i^{(n)}$
17: $\quad n \leftarrow n + 1$
18: **return** $\mathbf{r}^{(n)}$

1.2 Asynchronous Risk Propagation

While straightforward to specify, the description of risk propagation in Section 1.1, is not a viable implementation for real-world application. RISK-PROPAGATION is an *offline algorithm* because it requires all individuals' recent

contacts and risk scores to run. As Ayday et al. (2021) note, this is not privacy-preserving. An offline design is also suboptimal with respect computational efficiency and human safety. Specifically, most exposure scores may not change across invocations of RISK-PROPAGATION, which implies communication overhead and redundant computation. As a mitigation, RISK-PROPAGATION can only be executed once or twice daily, which was originally proposed by Ayday et al. (2020). However, this causes significant delay in reporting to individuals their new exposure risk; and in the face of a pandemic, timely information is essential for individual and collective health.

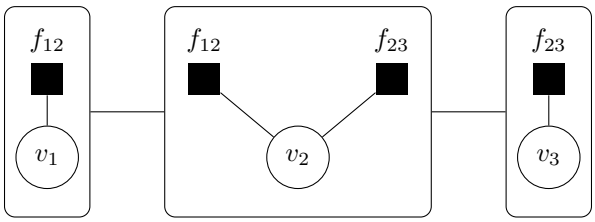

Figure 1.2: One-mode projection onto the variable nodes in Figure 1.1.

The only purpose of a factor vertex is to compute and relay messages between variable vertices. Thus, one-mode projection onto the variable vertices can be applied such that variable vertices $v_i, v_j \in V$ are adjacent if the factor vertex $f_{ij} \in F$ exists Zhou et al. (2007). Figure 1.2 shows the modified topology.

To send a message to variable vertex v_i, variable vertex v_j applies the computation associated with the factor vertex f_{ij}. This modification differs from the distributed extension of risk propagation Ayday et al. (2021) in that we do not create duplicate factor vertices and messages in each user's PDS. By storing the contact time between users on the edge incident to their variable

vertices, this modified topology is identical to the *contact-sequence* representation of a *contact network*, a kind of *time-varying* or *temporal network* in which a vertex represents a person and an edge indicates that two persons came in contact:

$$\{\, (v_i, v_j, t) \mid v_i, v_j \in V; v_i \neq v_j; t \leq 0 \,\}, \tag{1.3}$$

where a triple (v_i, v_j, t) is called a *contact* Holme and Saramäki (2012). Specific to risk propagation, t is the time at which users u and v *most recently* came in contact (see Section 1.2).

The usage of a temporal network in this work differs from its typical usage in epidemiology which focuses on modeling and analyzing the spreading dynamics of disease Craft (2015); Danon et al. (2011); Koher et al. (2019); Lokhov et al. (2014); Riolo et al. (2001); Zino and Cao (2021); **?**. In contrast, this work uses a temporal network to infer a user's MPPI. As a result, Section 1.3 extends temporal reachability to account for both the message-passing semantics and temporal dynamics of the network. As noted by Holme and Saramäki (2012), the transmission graph provided by Riolo et al. (2001) "cannot handle edges where one vertex manages to not catch the disease." Notably, the usage of a temporal network in this work allows for such cases by modeling the possibility of infection as a continuous outcome. Factor graphs are useful for decomposing complex probability distributions and allowing for efficient inference algorithms.

However, as with risk propagation, and generally any application of a factor graph in which the variable vertices represent entities of interest (i.e., of which

the marginal probability of a variable is desired), applying one-mode projection is a .

1.2.1 ShareTrace Actor System

As a distributed algorithm, risk propagation is specified from the perspective of an *actor*, which is equivalent to the paradiagm of *think- like-a-vertex* graph-processing algorithms McCune et al. (2015). Some variation exists on exactly how actor behavior is defined Agha (1985); **?**. Perhaps the simplest definition is that the *behavior of an actor* is both its *interface* (i.e., the types of messages it can receive) and *state* (i.e., the internal data it uses to process messages) **?**. An *actor system*[2] is defined as the set of actors it contains and the set of unprocessed messages[3] in the actor mailboxes. An expanded definition of an actor system also includes a *local states function* that maps mail addresses to behaviors, the set of *receptionist actors* that can receive communication that is external to the actor system, and the set of *external actors* that exist outside of the actor system Agha (1985). Practically, a local states function is unnecessary to specify, so the narrower definition of an actor system is used. The remainder of this section describes the components of the ShareTrace actor system.

[2]This is technically referred to as an *actor system configuration*.

[3]Formally, a *message* is called a *task* and is defined by a *tag*, a unique identifier; a *target*, the mail address to which the message is delivered; and a *communication*, the message content Agha (1985).

1.2.1.1 User Actor Behavior

Each user corresponds to an actor that participates in the message-passing protocol of risk propagation. Herein, the user of an actor will only be referred to as an *actor*. The following variant of the concurrent, object-oriented actor model is assumed to define actor behavior **?**.

- An actor follows the *active object pattern* Lavender and Schmidt (1996); **?** and the *Isolated Turn Principle* **?**. Specifically, the state change of an actor is carried out by instance- variable assignment, instead of the canonical BECOME primitive that provides a functional construct for pipelining actor behavior replacement Agha (1985); **?**. The interface of a user actor is fixed in risk propagation, so the more general semantics of BECOME is unnecessary.

- The term "name" Agha (1985); **?** is preferred over "mail address" Agha (1985); **?** to refer to the sender of a message. Generally, the mail address that is included in a message need not correspond to the actor that sent it. Risk propagation, however, requires this actor is identified in a risk-score message. Therefore, to emphasize this requirement, "name" is used to refer to both the identity of an actor and its mail address.

- An actor is allowed to include a loop with finite iteration in its behavior definition; this is traditionally prohibited in the actor model Agha (1985); **?**.

- Because the following pseudocode to describe actor behavior mostly follows the conventions of Cormen et al. (2022) (see **??**), the behavior

definition is implied by all procedures that take as input an actor.

The INITIALIZE-ACTOR operation initializes an actor, which is equivalent to the *new expression* Agha (1985) or CREATE primitive **?** with the exception that it only specifies the attributes (i.e., state) of an actor. As mentioned earlier, the behavior description of an actor is implied by the procedures that require an actor as input. Every actor A has the following attributes.

- $A.name$: an identifier that enables actors to communicate with it Agha (1985); **?**.

- $A.contacts$: a dictionary (see **??**) that maps an actor name to a timestamp. That is, if user i of actor A_i comes in contact with user j of actor A_j, then the *contacts* of A_i contains the name of A_j along with the most recent contact time for users i and j. The converse holds for user j. This is an extension of the *acquaintance vector* **?** or *acquaintance list* Agha (1985).

- $A.score$: the user's current exposure score. This attribute is either a default risk score (see), a risk score sent by an actor of some contacted user[4], or a symptom score of the user.

- $A.scores$: a dictionary that maps a time interval to a risk score; used to tolerate synchronization delays between a user's device and actor (see Section 1.2.1.2).

[4]As discussed later (see HANDLE-CONTACT-MESSAGE in Section 1.2.1.1), it possible for an actor i to send a message to an actor j but not be a contact of actor j.

INITIALIZE-ACTOR

1: $A.name \leftarrow$ GENERATE-NAME
2: $A.contacts \leftarrow \emptyset$
3: $A.scores \leftarrow \emptyset$
4: $A.score \leftarrow$ NULL-RISK-SCORE(A)
5: **return** A

NULL-RISK-SCORE(A)

1: $s.value \leftarrow 0$
2: $s.time \leftarrow 0$
3: $s.expiry \leftarrow 0$
4: $s.sender \leftarrow A.name$
5: **return** s

Risk scores and contacts have finite relevance, which is parametrized by a *liveness* or *relevance duration* $T_s, T_c > 0$, respectively. The relevance of risk scores and contacts is important, because it influences how actors pass messages. For example, actors do not send irrelevant risk scores or relevant risk scores to irrelevant contacts. The *time-to-live* (TTL) of a risk score or contact is the remaining duration of its relevance. In the following operations, $s.time$ denotes the time at which the risk score was originally computed, $c.time$ is the contact time, and τ is the current time. The interface of a user actor is

SCORE-TTL(s)

1: **return** $T_s - (\tau - s.time)$

defined by two types of messages: contact messages and risk-score messages. A *contact message* c contains the name $c.name$ of the actor whose user was

CONTACT-TTL(c)

1: **return** $T_c - (\tau - c.time)$

contacted and the contact time $c.time$. A *risk-score message* s is simply a risk score along with the actor's name $s.sender$ that sent it. A risk score previously defined as the ordered pair (r, t) (see Section 1.2) is represented as the attributes r and $s.time$. The following sections discuss how a contact message and risk-message are processed by an actor.

1.2.1.1.1 Handling Contact Messages There are two ways in which a user actor can receive a contact message. The first, technically correct approach is for a receptionist actor to mediate the communication between the user actor and the PDS so that the user actor can retrieve its user's contacts. The second approach is to relax this formality and allow the user actor to communicate with the PDS directly[5].

The HANDLE-CONTACT-MESSAGEoperation defines how a user actor processes a contact message. A contact is assumed to have finite relevance which is parametrized by the *contact time-to-live* $T_c > 0$. A contact whose contact time occurred no longer than T_c time ago is said to be *alive*. Thus, a user actor only adds a contact if it is alive. Regardless of whether the contact is alive, the user actor attempts to send a risk-score message that is derived from its current exposure score or a cached risk score (see Section 1.2.1.2):

[5]If the PDS itself is an actor, then a push-oriented dataflow could be implemented, where the user actor receives contact messages (and symptom-score messages). This would be more efficient and timely than a pull-oriented dataflow in which the PDS is a passive data store that requires the user actor or receptionist to poll it for new data.

```
HANDLE-CONTACT-MESSAGE(A, c)
 1: if CONTACT-TTL(c) > 0
 2:     c.key ← c.name
 3:     CACHE-INSERT(A.contacts, c)
 4: c.threshold ← NULL-RISK-SCORE(A)
 5: CACHE-REFRESH(A.scores)
 6: s ← CACHE-MAX(A.scores, c.time + β) or NULL-RISK-SCORE(A)
 7: SEND(A, c, s)
```

Like contacts, each risk score is assumed to have finite relevance that is

```
SEND(A, c, s)
 1: REFRESH-THRESHOLD(A, c)
 2: if SHOULD-RECEIVE(c, s)
 3:     s′.value ← s.value · α
 4:     SEND(c, s′)
 5:     SET-THRESHOLD(c, s′)
```

```
SHOULD-RECEIVE(c, s)
 1: return c.threshold.value < s.value
      and c.time + β > s.time
      and c.name ≠ s.sender
```

```
SET-THRESHOLD(c, s)
 1: s′.value ← γ · s.value
 2: c.threshold ← s′
```

REFRESH-THRESHOLD(A, c)

1: **if** $c.threshold.value > 0$ **and** SCORE-TTL$(c.threshold) \leq 0$

2: CACHE-REFRESH$(A.scores)$

3: $s \leftarrow$ CACHE-MAX$(A.scores, c.time + \beta)$ **or** NULL-RISK-SCORE(A)

4: $s'.value \leftarrow s.value \cdot \alpha$

5: SET-THRESHOLD(c, s')

parametrized by the *score time-to-live* $T_s > 0$ and evaluated by the operation IS-SCORE-ALIVE. To send an actor's current exposure score, the contact must be sufficiently recent. It is assumed that risk scores computed after a contact occurred have no effect on the user's exposure score.

To account for the disease incubation period, a delay in reporting symptoms, or a delay in establishing actor communication, a time buffer $\beta \geq 0$ is considered. That is, a risk score is not sent to a contact if IS-CONTACT-RECENT returns FALSE.

The TRANSMITTED operation is used to generate risk scores that are sent to other actors. It is assumed that contact only implies an incomplete transmission of risk between users. Thus, when sending a risk score to another actor, the value of the risk score is scaled by the *transmission rate* $\alpha \in (0, 1)$. Notice that the time of the risk score is left unchanged; the act of sending a risk-score message is independent of when the risk score was first computed.

If line **??** of [] evaluates to FALSE, then the actor attempts to retrieve the maximum cached risk-score message based on the buffered contact time. If such a message exists and is alive, a risk-score message is derived and sent to the contact. The SEND operation follows the semantics of the SEND-TO

primitive Agha (1985); **?**.

1.2.1.1.2 Handling Risk-Score Messages Upon receiving a risk-score message, an actor executes the following operation. The UPDATE-ACTOR

```
HANDLE-RISK-SCORE-MESSAGE(A, s)
1: if SCORE-TTL(s) > 0
2:     s.key ← [s.time, s.expiry)
3:     CACHE-INSERT(A.scores, s)
4:     UPDATE-EXPOSURE-SCORE(A, s)
5:     CACHE-REFRESH(A.contacts)
6:     PROPAGATE(A, s)
```

```
UPDATE-EXPOSURE-SCORE(A, s)
1: if A.score.value < s.value
2:     A.score ← s
3: else if SCORE-TTL(A.score) ≤ 0
4:     CACHE-REFRESH(A.scores)
5:     A.score ← CACHE-MAX(A.scores) or NULL-RISK-SCORE(A)
```

operation is responsible for updating an actor's state, based on a received risk-score message. Specifically, it stores the message inside the actor's interval cache $A.scores$, assigns the actor a new exposure score and send coefficient (discussed below) if the received risk-score value exceeds that of the current exposure score, and removes expired contacts.

In previous work Ayday et al. (2021), risk propagation assumes synchronous message passing, so the notion of an iteration or inter-iteration difference

threshold can be used as stopping conditions. However, as a streaming algorithm that relies on asynchronous message passing, such stopping criteria are unnatural. Instead, the following heuristic is applied and empirically optimized to minimize accuracy loss and maximize efficiency. Let $\gamma > 0$ be the *send coefficient* such that an actor only sends a risk-score message if its value exceeds the actor's *send threshold* $A.threshold$ (line **??** in PROPAGATE).

Assuming a finite number of actors, any positive send coefficient γ guarantees that a risk-score message will be propagated a finite number of times. Because the value of a risk score that is sent to another actor is scaled by the transmission rate α, its value exponentially decreases as it propagates away from the source actor with a rate constant $\log \alpha$.

As with the [] operation, a risk-score message must be alive and relatively recent for it to be propagated. As in previous work Ayday et al. (2021), factor marginalization is achieved by not propagating the received message to the actor who sent it. The logic of PROPAGATE differs, however, in two ways. First, it is possible that no message is not propagated to a contact. The intent of sending a risk-score message is to update the exposure score of other actors. However, previous work Ayday et al. (2021) required that a "null" message with a risk-score value of 0 is sent. Sending such ineffective messages incurs additional communication overhead. The second difference is that only the *most recent* contact time is used to determine if a message should be propagated to a contact. Contact times determine what messages are relevant. Given two contact times t_1, t_2 such that $t_1 \leq t_2$, then any risk score with time $t \leq t_1 + \beta$ also satisfies $t \leq t_2 + \beta$. Thus, storing multiple

contact times is unnecessary.

```
PROPAGATE(A, s)
1: for each c ∈ A.contacts
2:     SEND(A, c, s)
```

1.2.1.2 Risk Score Caching

1.3 Message Reachability

A fundamental concept of reachability in temporal networks is the *time-respecting path*: a contiguous sequence of contacts with nondecreasing time. Thus, vertex v is *temporally reachable* from vertex u if there exists a time-respecting path from u to v, denoted $u \to v$ Moody (2002). The following quantities are derived from the notion of a time-respecting path and help quantify reachability in a time-varying network Holme and Saramäki (2012).

- The *influence set* $I(v)$ *of vertex* v is the set of vertices that v can reach by a time-respecting path.
- The *source set* $S(v)$ *of vertex* v is the set of vertices that can reach v by a time-respecting path.
- The *reachability ratio* $f(G)$ *of a temporal network* G is the average influence-set cardinality of a vertex v.

Generally, a message-passing algorithm defines a set of constraints that determine when and what messages are sent from one vertex to another. Even

if operating on a temporal network, those constraints may be more or less strict than requiring temporal reachability. As a dynamic process, message passing on a time-varying network requires a more general definition of reachability that can account for network topology *and* message-passing semantics Barrat and Cattuto (2013).

Formally, the *message reachability from vertex u to vertex v* is the number of edges along the *shortest path P* that satisfy the message passing constraints,

$$m(u, v) = \sum_{(i,j) \in P} f(u, i, j, v),$$

where

$$f(u, i, j, v) = \begin{cases} 1 & \text{if all constraints are satisfied} \\ 0 & \text{otherwise.} \end{cases}$$

Vertex v is *message reachable* from vertex u if there exists a shortest path such that $m(u, v) > 0$. The *message reachability of vertex u* is the maximum message reachability from vertex u:

$$m(u) = \max\{m(u, v) \mid v \in V\}. \tag{1.4}$$

The temporal reachability metrics previously defined can be extended to

message reachability by only considering the message-reachable vertices:

$$I_m(u) = \{v \in V \mid m(u,v) > 0\}$$
$$S_m(v) = \{u \in V \mid m(u,v) > 0\}$$
$$f_m(G) = \sum_{v \in V} |I_m(v)| \cdot |V|^{-1}.$$

Let $\mathbf{M}$ be the $|V| \times |V|$ *message-reachability matrix* of the temporal network G such that vertices are enumerated $1, 2, \ldots, |V|$ and for each $m_{ij} \in \mathbf{M}$,

$$m_{ij} = \begin{cases} 1 & \text{if } m(i,j) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then the cardinality of the influence set for vertex i is the number of nonzero elements in the ith row of $\mathbf{M}$:

$$|I_m(i)| = \sum_{j=1}^{|V|} m_{ij}. \tag{1.5}$$

Similarly, the cardinality of the source set for vertex j is the number of nonzero elements in the jth column of $\mathbf{M}$:

$$|S_m(j)| = \sum_{i=1}^{|V|} m_{ij}. \tag{1.6}$$

For risk propagation, let $H(x)$ be the *Heaviside step function*,

$$H(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then message reachability is defined as

$$m(u, v) = \sum_{(i,j) \in P} f_r(u, i) \cdot f_c(u, i, j) \tag{1.7}$$

where P is the set of edges along the shortest path $u \to v$ such that the actors are enumerated $0, 1, \ldots, |P| - 1$ and

$$f_r(u, i) = H(\alpha^i \cdot r_u - \gamma \cdot r_i) \tag{1.8}$$

$$f_c(u, i, j) = H(t_{ij} - t_u + \beta) \tag{1.9}$$

are the value and contact-time constraints in the SHOULD-RECEIVE operation (see Section 1.2.1.1), where (r_i, t_i) is the current exposure score for actor i and t_{ij} is the most recent contact time between actors i and j.

The value of (1.7) can be found by associating with each symptom score a unique identifier. If each actor maintains a log of the risk scores it receives, then the set of actors that receive the symptom score or a propagated risk score thereof can be identified. This set of actors defines the induced subgraph on which to compute (1.7) using a shortest-path algorithm Johnson (1977).

Regarding efficiency, (1.4) to (1.6) provide the means to quantify the communication overhead of a given message-passing algorithm on a temporal net-

work. Moreover, because such metrics capture the temporality of message passing, they can better quantify complexity than traditional graph metrics.

By relaxing the constraint (1.9), it is possible to estimate (1.7) with (1.8). The *estimated message reachability of vertex u to vertex v*, denoted $\hat{m}(u, v)$, is defined as follows. Based on (1.8),

$$\alpha^{\hat{m}(u,v)} \cdot r_u \leq \gamma \cdot r_v,$$

where the left-hand side is the value of the propagated symptom score for actor u when $\hat{m}(u, v) = 1$, and the right-hand side is the value required by some message-reachable actor v to propagate the message received by actor u or some intermediate actor. Solving for $\hat{m}(u, v)$,

$$\hat{m}(u, v) \leq f(u, v), \tag{1.10}$$

where

$$f(u,v) = \begin{cases} 0 & \text{if } r_u = 0 \\ |P| & \text{if } r_v = 0 \\ \log_\alpha \gamma + \log_\alpha r_v - \log_\alpha r_u & \text{otherwise.} \end{cases}$$

Equation (1.10) indicates that a lower send coefficient γ will generally result in higher message reachability, at the cost of sending possibly ineffective messages (i.e., risk scores that do not update the exposure score of another actor). Equation (1.10) also quantifies the effect of the transmission rate α. Unlike the send coefficient, however, the transmission rate is intended to be derived

from epidemiology to quantify disease infectivity and should not be optimized to improve performance.

Given the multivariate nature of message reachability, it is helpful to visualize how it with various combinations of parameter values. Figure 1.3 includes several line plots of estimated message reachability $\hat{m}(u, v)$ with respect to the initial risk score magnitude of actor u.

1.4 System Model

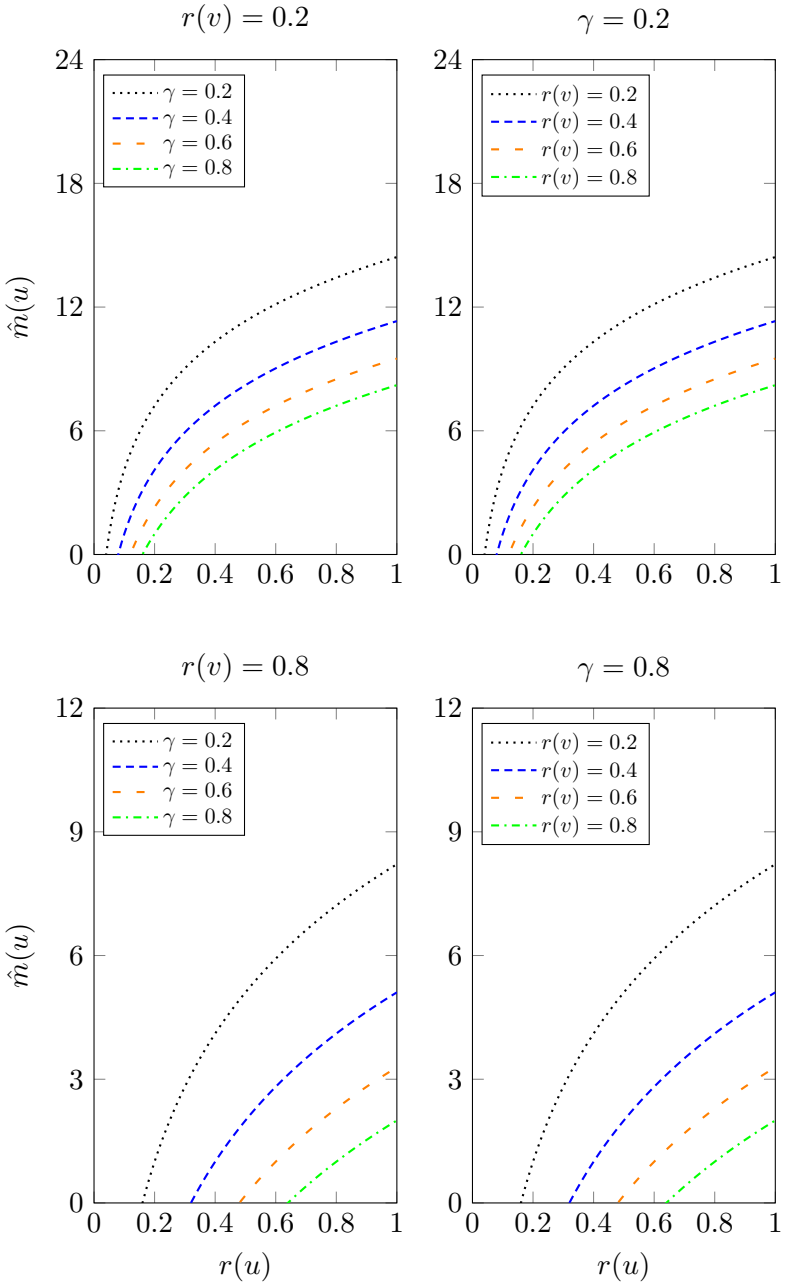

Figure 1.3: Estimated risk propagation message reachability $\hat{m}(u)$ for different values of the transmission rate α, the send tolerance γ, and the initial magnitude $r(v) = r_0(v)/\alpha$ of the destination user v with respect to the initial magnitude $r_u = r_0(u)/\alpha$ of the source user u.

Appendix A

Pseudocode Conventions

The pseudocode conventions in this work are mostly consistent with Cormen et al. (2022).

- Indentation indicates block structure.
- Looping and conditional constructs have similar interpretations to those in standard programming languages.
- Composite data types are represented as *objects*. Accessing an *atttribute* a of an object o is denoted $o.a$. A variable representing an object is a *pointer* or *reference* to the data representing the object. The special value NIL refers to the absence of an object.
- Parameters are passed to a procedure *by value*: the "procedure receives its own copy of the parameters, and if it assigns a value to a parameters, the change is *not* seen by the calling procedure. When objects are passed, the pointer to the data representing the object is copied, but the

attributes of the object are not." Thus, attribute assignment "is visible if the calling procedure has a pointer to the same object."

- A **return** statement "immediately transfers control back to the point of call in the calling procedure."
- Boolean operators **and** and **or** are *short circuiting*.

The following conventions are specific to this work.

- Object attributes may be defined *dynamically* in a procedure.
- Variables are local to the given procedure, but parameters are global.
- The "$\leftarrow$" symbol is used to denote assignment, instead of "=".
- The "=" symbol is used to denote equality, instead of "==", which is consistent with the use of "$\neq$" to denote inequality.
- The "$\in$" symbol is used in **for** loops when iterating over a collection.
- Set-builder notation $\{ x \in X \mid \textsc{Predicate}(x) \}$ is used to create a subset of a collection X in place of constructing an explicit data structure.

Appendix B

Data Structures

Let a *dynamic set* X be a mutable collection of distinct elements. Let x be a pointer to an element in X such that $x.key$ uniquely identifies the element in X (Cormen et al., 2022). Let a *dictionary* be a dynamic set that supports insertion, deletion, and membership querying (Cormen et al., 2022):

- $\textsc{Search}(X, k)$ returns a pointer x to an element in the set X such that $x.key = k$; or NIL, if no such element exists in X.
- $\textsc{Insert}(X, x)$ adds the element pointed to by x to X.
- $\textsc{Delete}(X, x)$ removes the element pointed to by x from X.
- $\textsc{Minimum}(X)$ and $\textsc{Maximum}(X)$ return a pointer x to the minimum and maximum element, respectively, of the totally ordered set X; or NIL, if X is empty. Unlike Cormen et al. (2022), this work permits attributes other than $x.key$ to be used in the element ordering of X.

Appendix C

Actor Model

The *actor model* is a local model of concurrent computing that defines computation as a strict partial ordering of events (Hewitt, 1977; Hewitt and Baker, 1977). An *event* is defined as "a transition from one local state to another" (Hewitt and Baker, 1977); or, more concretely, "a *message* arriving at a computational agent called an *actor*." The actor that receives the message of an event is called the *target* of the event.

Upon receipt, an actor may send messages to itself or other actors, update its local state, or create other actors.

May send messages to other actors (Clinger) Update its local state (Clinger) Create other actors

Events caused

An event may *activate* subsequent events

Ordering laws:

Passing messages between actors is the only means of communication. Thus, control and data flow are inseparable in the actor model (Hewitt et al., 1973).

A minimal specification of an actor includes its *name* and *behavior*, or how it acts upon receiving a message. Clinger (1981) explains,

> An actor's name is a necessary part of its description because two different actors may have the same behavior. An actor's behavior is a necessary part of its description because the same actor may have different behaviors at different times.

acquaintances, or a finite collection of names of the actors that it knows about (Hewitt et al., 1973).

Thus, the specification of an actor system consists of the following (Hewitt, 1977).

- Deciding on the natural kinds of actors (objects) to have in the system to be constructed.
- Deciding for each kind of actor what kind of messages it should receive.
- Deciding for each kind of actor what it should do when it receives each kind of message.

Bibliography

Gul Agha. *Actors: A model of concurrent computation in distributed systems.* PhD thesis, MIT, 1985. URL `http://hdl.handle.net/1721.1/6952`.

Erman Ayday, Fred Collopy, Taehyun Hwang, Glenn Parry, Justo Karell, James Kingston, Irene Ng, Aiden Owens, Brian Ray, Shirley Reynolds, Jenny Tran, Shawn Yeager, Youngjin Yoo, and Gretchen Young. Sharetrace: A smart privacy-preserving contact tracing solution by architectural design during an epidemic. Technical report, Case Western Reserve University, 2020. URL `https://github.com/cwru-xlab/sharetrace-papers/blob/main/sharetace-whitepaper.pdf`.

Erman Ayday, Youngjin Yoo, and Anisa Halimi. ShareTrace: An iterative message passing algorithm for efficient and effective disease risk assessment on an interaction graph. In *Proc. 12th ACM Con. Bioinformatics, Comput. Biology, Health Inform.*, BCB 2021, 2021.

Alain Barrat and Ciro Cattuto. Temporal networks of face-to-face human interactions. In Petter Holme and Jari Saramäki, editors, *Temporal Netw.*, Underst. Complex Syst. Springer, 2013. doi:10.1007/978-3-642-36461-7_10.

Christopher M. Bishop. Pattern recognition and machine learning. In M. I. Jordan, Robert Nowak, and Bernhard Schoelkopf, editors, *Inf. Sci. Stat.* Springer, 2006.

William Clinger. *Foundations of actor semantics*. PhD thesis, MIT, 1981. URL `http://hdl.handle.net/1721.1/6935`.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, fourth edition, 2022.

Meggan E. Craft. Infectious disease transmission and contact networks in wildlife and livestock. *Phil. Trans. R. Soc. B*, 370, 2015. doi:10.1098/rstb.2014.0107.

Leon Danon, Ashley P. Ford, Thomas House, Chris P. Jewell, Gareth O. Roberts, Joshua V. Ross, and Matthew C. Vernon. Networks and the epidemiology of infectious disease. *Interdiscip. Perspect. Infect. Dis.*, 2011, 2011. doi:10.1155/2011/284909.

Centers for Disease Control and Prevention. Quarantine and isolation, 2021. `https://www.cdc.gov/coronavirus/2019-ncov/your-health/quarantine-isolation.html`.

Carl Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(3):323–364, 1977. doi:10.1016/0004-3702(77)90033-9.

Carl Hewitt and Henry Baker. Laws for communicating parallel processes. Working paper, MIT Artificial Intelligence Laboratory, 1977. URL `http://hdl.handle.net/1721.1/41962`.

Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, pages 235–245, 1973. URL `https://dl.acm.org/doi/10.5555/1624775.1624804`.

Petter Holme and Jari Saramäki. Temporal networks. *Phys. Rep.*, 519, 2012. doi:10.1016/j.physrep.2012.03.001.

Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24, 1977. doi:10.1145/321992.321993.

Andreas Koher, Hartmut H. K. Lentz, James P. Gleeson, and Philipp Hövel. Contact-based model for epidemic spreading on temporal networks. *Phys. Rev. X*, 9, 2019. doi:10.1103/PhysRevX.9.031017.

Frank R. Kschischang, Brendan J. Frey, and Hans A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, 47, 2001. doi:10.1109/18.910572.

R. Greg Lavender and Douglas C. Schmidt. Active object – an object behavioral pattern for concurrent programming, 1996. `https://csis.pace.edu/~marchese/CS865/Papers/Act-Obj.pdf`.

Andrey Y. Lokhov, Marc Mézard, Hiroki Ohta, and Lenka Zdeborová. Inferring the origin of an epidemic with a dynamic message-passing algorithm. *Phys. Rev. E*, 90, 2014. doi:10.1103/PhysRevE.90.012801.

Robert McCune, Tim Weninger, and Greg Madey. Thinking like a vertex: A

survey of vertex-centric frameworks for large-scale distributed graph processing. *ACM Comput. Surveys*, 48, 2015. doi:10.1145/2818185.

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, Alessia Visconti, Pirro Hysi, Ruth C E Bowyer, Massimo Mangino, Mario Falchi, Jonathan Wolf, Sebastien Ourselin, Andrew T Chan, Claire J Steves, and Tim D Spector. Real-time tracking of self-reported symptoms to predict potential COVID-19. *Nat. Med.*, 26, 2020. doi:10.1038/s41591-020-0916-2.

James Moody. The importance of relationship timing for diffusion. *Soc. Forces.*, 81, 2002.

Christopher S. Riolo, James S. Koopman, and Stephen E. Chick. Methods and measures for the description of epidemiologic contact networks. *J. Urban Health*, 78, 2001. doi:10.1093/jurban/78.3.446.

Tao Zhou, Jie Ren, Matú š Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. *Phys. Rev. E*, 76, 2007.

Lorenzo Zino and Ming Cao. Analysis, prediction, and control of epidemics: A survey from scalar to dynamic network models. *IEEE Circuits Syst. Mag.*, 21, 2021. doi:10.1109/mcas.2021.3118100.