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 distribution 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 $G = (X, F, E)$ be a *factor graph* where X 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*

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

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

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

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

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

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 $f(x_i, x_j) = f_{ij}$, which is adjacent to the variable nodes x_i, x_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.

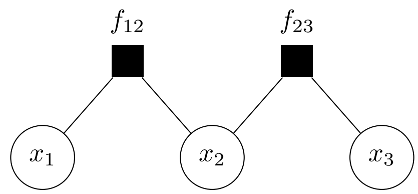

Figure 1.1: A factor graph of 3 variable nodes and 2 factor nodes.

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 and is assumed to be the time at which RISK-PROPAGATION is invoked. For notational simplicity in RISK-PROPAGATION, let X be a set. Then $\max X = 0$ if $X = \emptyset$.

1.1.1 Variable Messages

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 x_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 Messages

A *factor message* $\lambda_{ij}^{(n)}$ from the factor node f_{ij} to the variable node x_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

$$\{ v \in X \cap F \setminus \{x_j, f_{ij}\} \mid d(x_i, v) \leq 2(n-1) \},$$

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 $t < t_{ij} + \beta$, where $\beta \in \mathbb{N}$ is a *time buffer* that accounts for delayed reporting of symptom scores and contacts. The expression $t_{ij} + \beta$ is called the *buffered contact time*.

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 m times such that $t_k < t_\ell$ for $1 \leq k < \ell \leq m$. Let Λ_k be the set of relevant risk scores, according to the contact time t_k, where

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

Then $\Lambda_k \subseteq \Lambda_\ell$ if and only if $\max \Lambda_k \leq \max \Lambda_\ell$. Therefore, only the most recent contact time t_m 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 subgraph induced in G by their contacts that define the neighborhood N_i of the variable node x_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

To detect convergence, the normed difference between the current and previous exposure scores is compared to the threshold $\epsilon \in \mathbb{R}$. Note that $\mathbf{r}^{(n)}$ is the vector of exposure 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)

$(X, F, E) \leftarrow$ Create-Factor-Graph(C)

$n \leftarrow 1$

for each $x_i \in X$

  $R_i^{(n-1)} \leftarrow \text{top } K \text{ of } S_i$

  $r_i^{(n-1)} \leftarrow \max R_i^{(n-1)}$

  $r_i^{(n)} \leftarrow \infty$

while $\|\mathbf{r}^{(n)} - \mathbf{r}^{(n-1)}\| > \epsilon$

  for each $\{x_i, f_{ij}\} \in E$

    $\mu_{ij}^{(n)} \leftarrow R_i^{(n-1)} \setminus \{\, \lambda_{ji}^{(k)} \mid k \in [1 \mathinner{..} n-1] \,\}$

  for each $\{x_i, f_{ij}\} \in E$

    $\lambda_{ij}^{(n)} \leftarrow \max \{\, \alpha s_t \mid s_t \in \mu_{ij}^{(n)}, t < t_{ij} + \beta \,\}$

  for each $x_i \in X$

    $R_i^{(n)} \leftarrow \text{top } K \text{ of } \{\, \lambda_{ji}^{(n)} \mid f_{ij} \in N_i \,\}$

  for each $x_i \in X$

    $r_i^{(n-1)} \leftarrow r_i^{(n)}$

    $r_i^{(n)} \leftarrow \max R_i^{(n)}$

  $n \leftarrow n + 1$

return $\mathbf{r}^{(n)}$

1.2 Asynchronous Risk Propagation

While straightforward to specify, Risk-Propagation, is not a viable implementation for real-world application, because it is an *offline algorithm* that requires all individuals' recent contacts and risk scores to run. As Ayday et al.

(2021) note, the centralization of health and contact data is not privacy preserving. An offline design is also computational inefficient and risks human safety. Specifically, most exposure scores may not change across invocations of Risk-Propagation, which implies communication overhead and computational redundancy. As a mitigation, Ayday et al. (2020) suggest running Risk-Propagation only once or twice daily. However, this causes substantial delay in reporting to individuals their exposure scores; and in the face of a pandemic, timely information is essential for individual and collective health.

To address the privacy limitations of Risk-Propagation, Ayday et al. (2021) propose decentralizing the factor graph such that the processing entity (e.g., mobile device or "personal cloud") associated with the i-th individual maintains the state of the i-th variable node and its neighboring factor nodes. But for real-world application, the proposal leaves key questions unanswered:

1. Is message passing synchronous or asynchronous?
2. How does message passing terminate?
3. Are any optimizations used to reduce communication cost?
4. How do processing entities exchange messages over the network?

Key observations:

- Asynchronous message passing amongst stateful entities is the essence of the actor model, so we can use it to describe risk propagation as an online algorithm - By applying one-mode projection, the factor graph is equivalent to a contact network. Thus, we can extend the concepts of reachability in temporal network to account for message-passing semantics and measure the communication complexity of risk propagation.

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

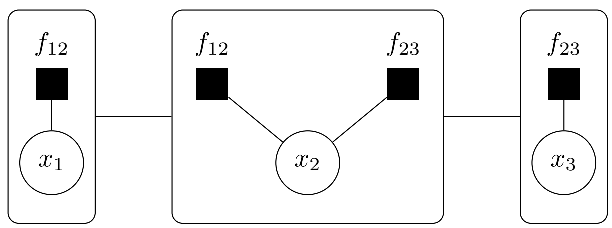

Figure 1.2: One-mode projection onto the variable nodes in Figure 1.1.

To send a message to variable node x_i, variable node x_j applies the computation associated with the factor node 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 node represents a person and an edge indicates that two persons came in contact:

$$\{ (u, v, t) \mid u, v \in V; u \neq v; t \in \mathbb{N} \}, \tag{1.3}$$

where a triple (u, v, 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 node 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*. 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.

1.2.2 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

[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).

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); **?**.

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

The Create-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. An actor a has the following attributes.

- $a.name$: the actor's name; used by other actors to communicate with it (Agha, 1985; **?**).

- $a.contacts$: a dictionary (see Appendix B) that maps an actor's name to a time. That is, if the i-th individual comes in contact with the j-th individual, then $a_i.contacts$ contains $a_j.name$ and the time of contact. This is an extension of the concept of *acquaintances* (Agha, 1985; **?**).

- $a.scores$: a dictionary that maps a time interval to a risk score; used to tolerate synchronization delays between a user's device and actor (see **??**).
- $a.exposure$: the actor's exposure score. This attribute is either a symptom score, a risk score sent by another actor, or the null risk score (see Null-Risk-Score).

```
Create-Actor
1: a.name ← Generate-Name
2: a.contacts ← ∅
3: a.scores ← ∅
4: a.exposure ← Null-Risk-Score(a)
5: return a
```

```
Null-Risk-Score(a)
1: s.value ← 0
2: s.t ← 0
3: s.sender ← a.name
4: 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, $s.t$ denotes the time at which the risk score was originally computed, $c.t$ is the contact time, and τ is the current time. The interface of a user actor is defined by

```
RISK-SCORE-TTL(s)
1: return T_s − (τ − s.t)
```

```
CONTACT-TTL(c)
1: return T_c − (τ − c.t)
```

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 contacted and the contact time $c.t$. 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 (s, t) (see Section 1.2) is represented as the attributes s and $s.t$. The following sections discuss how a contact message and risk-message are processed by an actor.

1.2.2.1 Contacts

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[4].

The HANDLE-CONTACToperation 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,

HANDLE-CONTACT(a, c)

if CONTACT-TTL$(c) > 0$
  $c.key \leftarrow c.name$
  CACHE-INSERT$(a.contacts, c)$
  $s \leftarrow$ CACHE-MAX-OLDER-THAN$(a.scores, c.t + \beta)$
  APPLY-RISK-SCORE(a, c, s)

the user actor attempts to send a risk score message that is derived from its current exposure score or a cached risk score (see **??**): Like contacts,

APPLY-RISK-SCORE(a, c, s)

REFRESH-SEND-THRESHOLD(a, c)
if SHOULD-CONTACT-RECEIVE(c, s)
  $s'.value \leftarrow s.value \cdot \alpha$
  $c.buffered \leftarrow s'$
  SET-SEND-THRESHOLD(c, s')

each risk score is assumed to have finite relevance that is parametrized by the

[4] 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.

```
Should-Contact-Receive(c, s)
return c.threshold.value < s.value
  and c.t + β > s.t
  and c.name ≠ s.sender
```

```
Set-Send-Threshold(c, s)
s′.value ← γ · s.value
c.threshold ← s′
```

```
Refresh-Send-Threshold(a, c)
if c.threshold.value > 0 and Risk-Score-Ttl(c.threshold) ≤ 0
    s ← Cache-Max-Older-Than(a.scores, c.t + β)
    s′.value ← s.value · α
    Set-Send-Threshold(c, s′)
```

```
Handle-Flush-Timeout(a)
for each c ∈ a.contacts
    if c.buffered ≠ nil
        Send(c.name, c.buffered)
        c.buffered ← nil
Cache-Refresh(a.contacts)
```

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.

1.2.2.2 Risk Scores

Upon receiving a risk score message, an actor executes the following operation.

The Update-Actor operation is responsible for updating an actor's state,

```
Handle-Risk-Score(a, s)
if Risk-Score-Ttl(s) > 0
    s.key ← [s.t, s.t + T_s)
    Cache-Insert(a.scores, s)
    Update-Exposure-Score(a, s)
    for each c ∈ a.contacts
        Apply-Risk-Score(a, c, s)
```

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.

Assuming a finite number of actors, any positive send coefficient γ guar-

```
Update-Exposure-Score(a, s)
if a.exposure.value < s.value
    a.exposure ← s
else if Risk-Score-Ttl(a.exposure) ≤ 0
    a.exposure ← Cache-Max(a.scores)
```

antees 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 $\log \alpha$.

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, node v is *temporally reachable* from node u if there exists a time-respecting path from 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 node v is the set of nodes that v can reach by a time-respecting path.
- The *source set* S_v of node v is the set of nodes 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 node v.

Generally, a message-passing algorithm defines a set of constraints that determine when and what messages are sent from one node 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 node u to node 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

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

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

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

The temporal reachability metrics previously defined can be extended to message reachability by only considering the message-reachable vertices:

$$I_u = \{\, v \in V \mid m(u,v) = |P| \,\}$$
$$S_v = \{\, u \in V \mid m(u,v) = |P| \,\}$$
$$f(G) = \sum_{v \in V} |I_v| \cdot |V|^{-1} \,.$$

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

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

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

$$|I_i| = \sum_{j=1}^{|V|} m_{ij}. \tag{1.5}$$

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

$$|S_j| = \sum_{i=1}^{|V|} m_{ij}. \tag{1.6}$$

For risk propagation, let P is the set of edges along the shortest path from u to v such that the actors are enumerated $1, \ldots, |P|$. Then message reachability is defined as

$$m(u,v) = \sum_{(i,j)\in P} [\alpha^i s_u > \gamma \alpha s_{ij}] \cdot [t_u < t_{ij} + \beta] \tag{1.7}$$

are the value and contact-time constraints in the Should-Contact-Receive operation (see Section 1.2.2), where (s_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 network. Moreover, because such metrics capture the temporality of message passing, they can better quantify complexity than traditional graph metrics.

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

$$\alpha^{\hat{m}(u,v)} \cdot s_u \le \gamma \cdot s_v,$$

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 $\hat{m}(u,v)$,

$$\hat{m}(u,v) \leq f(u,v), \tag{1.8}$$

where

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

Equation (1.8) 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 (1.8) 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.

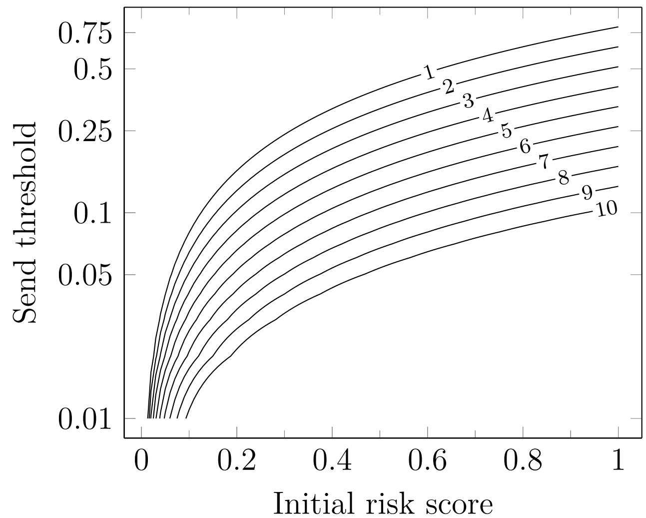

Figure 1.3: Estimated message reachability. Contour lines are shown for integral reachability values. Given an initial risk score and message reachability, a contour line indicates an upper bound on the permissible send threshold.

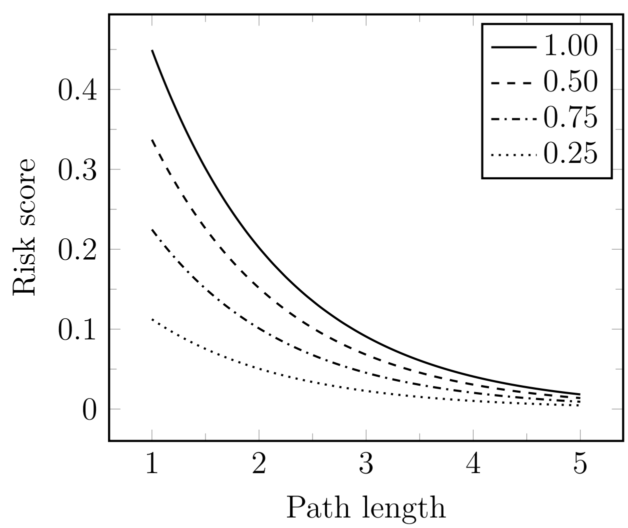

Figure 1.4: Exponential decay of risk scores.

Appendix A

Pseudocode Conventions

Pseudocode conventions 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. 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.

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.

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`.

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.

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.