CSE 8803 EPI: Data Science for Epidemiology, Fall 2022

Lecturer: B. Aditya Prakash
Scribe: Shen En Chen, Keerthan Ramnath

September 29, 2022
Lecture 11: Outbreak Detection (II)

---

1 Lecture Overview

Early detection and modeling of a contagious epidemic can provide important guidance on controlling contagion spread and designing effective countermeasures. To this end, designing social network sensors (e.g. monitoring a handful of individuals to forecast emergence of an epidemic) has attracted much attention. In Lecture 10, we saw that the problem of selecting the set of sensor nodes that maximize the rewards (e.g. peak lead time) can be modeled with submodular functions. Despite its NP-hardness, we can effectively solve for the submodular set function with greedy hill climbing. We prove that the greedy hill climbing algorithm gives a $(1-e^{-1})$-approximation by lower-bounding the gain at each step.

Another scenario of deploying social network sensors is the prevention and control of healthcare-associated infection (HAI) outbreaks in hospitals. Different from general population, social networks in hospitals are often modeled by 2-mode models, which consider not only individuals but also locations/objects such as door knobs as nodes in the network. Due to the heterogeneity of nodes and resource constraints, the objective optimization changes from selecting the optimal set of nodes to place sensors on to calculating the optimal probability vector of which each dimension represents the per day rate of monitoring for a particular type of agents or locations (e.g., patients, technicians, door knobs, etc.). The goal is to find the probability vector that maximizes the expected number of scenarios detected. Similarly, the problem is NP-hard, and we can leverage submodularity to approximate with a greedy algorithm.

2 Greedy Hill Climbing for Submodular Functions

Recall that, for the general sensor optimization problem, we can abstract the problem as set functions $f(S)$ that takes a set of nodes as input an returns a reward.

2.1 Submodular Set Functions

The set function $f(S)$ is sub-modular if:

$$(S \cup \{u\}) - f(S) \geq f(S \cup \{u\}) - f(S), \forall S \subseteq T$$

We also note that sum of submodular functions is also a submodular function.

2.2 Optimization of Set Function

Our goal is to find:

$$S^* = \arg\max_S f(S) \text{ s.t. } |S| = k$$

which is NP-hard. But we can use hill climbing to give an approximation.

2.3 Greedy Hill Climbing

The greedy hill climbing works as the follows:

- We begin with an empty set $S_0 = \{\}$.
- At each iteration, pick the node that gives the best marginal gain

$$S_i = S_{i-1} \cup \arg\max_u f(S_{i-1} \cup u)$$

- If we have a constraint k on the set size, stop when $|S| = k$. Otherwise, continue adding nodes to the set.
- Due to the monotonicity of modular functions, in the absence of k, adding all nodes to the set would yield the maximum gain/reward.

  Compared to the brute-force approach, which has exponential time for the n-choose-k operation, the greedy hill climbing is $O(n)$ at each iteration, leading to a total time complexity of $O(kn)$ after k iterations.

2.4 Why Hill Climbing Works

2.4.1 Claim 1: The marginal gain of adding an arbitrary set of nodes B to a given set A is upper bounded by the sum of the individual gain of adding each node in B independently to A

Given an arbitrary set of nodes $B = \{b_1, b_2, \ldots, b_k\}$, Claim 1 states:

$$f(A \cup B) - f(A) \leq \sum_i^k [f(A \cup \{b_i\} - f(A))]$$

2.4.2 Proof of Claim 1

Let $B_i = b_1, \ldots, b_i$ (the set of all the B_j for $1 \leq j \leq i$) and $B_0 = \{\}$. Then:

$$f(A \cup B) - f(A) = \sum_{i=1}^k [f(A \cup B_i) - f(A \cup B_{i-1})] \quad (1)$$

$$= \sum_{i=1}^k [f(A \cup B_{i-1} \cup \{b_i\}) - f(A \cup B_{i-1})] \quad (2)$$

$$\leq \sum_{i=1}^k [f(A \cup B_{i-1}) - f(A)] \quad (3)$$

- Eq. (1): **The gain of adding the whole set B to A is equivalent to the sum of marginal gains of gradually adding each element of B.** For example, we first add b_i to A to form $A \cup \{b_1\}$; the marginal gain for this addition is $f(A \cup \{b_1\}) - f(A) = f(A \cup B_1) - f(A \cup \{\}) = f(A \cup B_1) - f(A \cup B_0)$. Similarly, the marginal gain of adding b_2 following the addition of b_1 is $f(A \cup B_2) - f(A \cup B_1)$. We continue adding in elements of B until we get $\sum_{i=1}^k [f(A \cup B_i) - f(A \cup B_{i-1})]$.

- Eq. (2): Rewriting $B_i = B_{i-1} \cup \{b_i\}$.
- Eq. (3): Applying the **property of diminishing return** on Eq. (2). We can view each term $f(A \cup B_{i-1} \cup \{b_i\}) - f(A \cup B_{i-1})$ in the summation in Eq. (2) as the margin gain of adding b_i to $A \cup B_{i-1}$. Because $A \subseteq A \cup B_{i-1}$, the marginal gain of adding b_i to the larger set $A \cup B_{i-1}$ is smaller or equal to that of adding to the smaller set A. Thus, each term in the summation in Eq. (2) is smaller or equal to its corresponding term in Eq. (3).

2.4.3 Main Proof (Part 1)

Given S_i, where S_i is the set we have at step i during hill climbing, and an arbitrary set T of size k that follows similar notation of set B in Section 2.4.1, we have the following:

$$
\begin{aligned}
f(T) &\leq f(S_i \cup T) && (4)\\
&= f(S_i \cup T) - f(S_i) + f(S_i) && (5)\\
&\leq \sum_{j=1}^{k} [f(S_i \cup \{t_j\}) - f(S_i)] + f(S_i) && (6)\\
&\leq \sum_{j=1}^{k} \delta_{i+1} + f(S_i) && (7)\\
&= f(S_i) + k\delta_{i+1} && (8)
\end{aligned}
$$

Rearranging the inequality above, we obtain:

$$
\delta_{i+1} \geq \frac{1}{k} [f(T) - f(S_i)] \tag{9}
$$

- Eq. (4): The monotonicity of modular function.
- Eq. (5): Subtracting $f(S_i)$ and adding $f(S_i)$ on purpose for the following derivation.
- Eq. (6): Applying Claim 1.
- Eq. (7): Rewriting the marginal gain $f(S_i \cup \{t_j\}) - f(S_i) = f(S_{i+1}) - f(S_i)$ as δ_{i+1}.
- Eq. (8): Simplifying the summation because δ_{i+1} is independent of the index j for T.

Implication of Eq. (9): the marginal gain at each step i during hill climbing cannot be too bad given the lower bound.

Following Eq. (9), we can write the reward of $f(S_{i+1})$ as:

$$
\begin{aligned}
f(S_{i+1}) &= f(S_i) + \delta_{i+1}\\
&\geq f(s_i) + \frac{1}{k} [f(T) - f(S_i)] && (10)\\
&= (1 - \frac{1}{k}) f(S_i) + \frac{1}{k} f(T) && (11)
\end{aligned}
$$

- Eq. (10): Applying Eq. (9).
- Eq. (11): Rewriting Eq. (10).

2.4.4 Claim 2: $f(S_i)$ has a lower bound $\left[1-(1-\frac{1}{k})^i\right] f(T)$.

Given S_i, where S_i is the set we have at step i during hill climbing, and an arbitrary set T of size k that follows similar notation of set B in Section 2.4.1, Claim 2 states:

$$f(S_i) \geq \left[1-(1-\frac{1}{k})^i\right] f(T), \quad \forall i \geq 0 \tag{12}$$

2.4.5 Proof of Claim 2

We will prove Claim 2 by induction.

Base Case $i = 0$:

$$f(S_0) = f(\phi) = 0 = \left[1-(1-\frac{1}{k})^0\right] f(T)$$

Inductive Step:

Inductive Claim: Assume $f(S_i) \geq \left[1-(1-\frac{1}{k})^i\right] f(T)$ holds for $i = m$.

Then, following Eq. (11):

$$\begin{aligned}
f(S_{m+1}) &= (1-\frac{1}{k}) f(S_m) + \frac{1}{k} f(T) \\
&\leq (1-\frac{1}{k}) \left[1-(1-\frac{1}{k})^m\right] f(T) + \frac{1}{k} f(T), \quad \text{(Inductive Claim)} \\
&= \left[1-\frac{1}{k}-(1-\frac{1}{k})^m + \frac{1}{k}(1-\frac{1}{k})^m + \frac{1}{k}\right] f(T) \\
&= \left[1-(1-\frac{1}{k})^m + \frac{1}{k}(1-\frac{1}{k})^m\right] f(T) \\
&= \left[1-(1-\frac{1}{k})(1-\frac{1}{k})^m\right] f(T) \\
&= \left[1-(1-\frac{1}{k})^{m+1}\right] f(T)
\end{aligned}$$

By induction, we have proved $f(S_i) \geq \left[1-(1-\frac{1}{k})^i\right] f(T), \forall i \geq 0$.

2.4.6 Main Proof (Part 2)

Given Claim 2, we have:

$$f(S) = f(S_k) \geq \left[1 - (1 - \frac{1}{k})^k\right] f(T)$$
$$= f(S_k) \geq (1 - \frac{1}{e}) f(T)$$

We have thus proven that the greedy hill climbing algorithm gives a $(1 - \frac{1}{e})$-approximation.

Note: $(1 - \frac{1}{k})^k = \frac{1}{e}$ comes from the well known inequality $1 - x < e^{-x}$, where $x = \frac{1}{k}$ in our case.

3 Detection in 2-Mode Models

The greedy hill climbing algorithm proven above applies to general network models that model each individuals in the populations as nodes. However, in some cases, we need a domain-aware framework to effectively model the environment. At hospitals, we often see a 2-mode model used to monitor whether there is a healthcare-associated infection (HAI) outbreak within the hospital. 2-mode model provides more flexibility by taking into account location-to-person and person-to-location transmissions and considering both people and locations/objects (e.g., door knobs) as nodes that can come into contact with contagions.

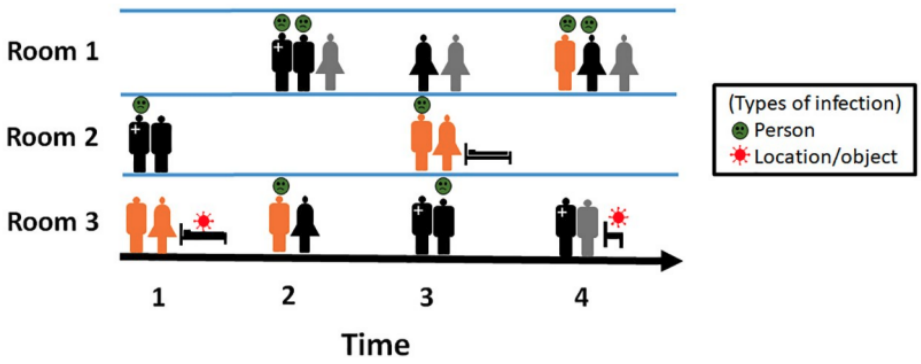

Fig 1. Visualization of a possible HAI spread. The only healthcare worker in the example is marked with a '+' sign. The rest of the humans are patients and visitors. Each individual is assigned a unique combination of color and silhouette pair. Infected humans are indicated by the green emoji and contaminated locations are indicated by the bacteria emoji. As human agents move through various locations, they infect other agents and contaminate the locations.

Figure 1: Visualization of an example HAI spread in hospitals [1]

In contrast to general SIR model, such model also adds more complexities to model the different states of people and locations.

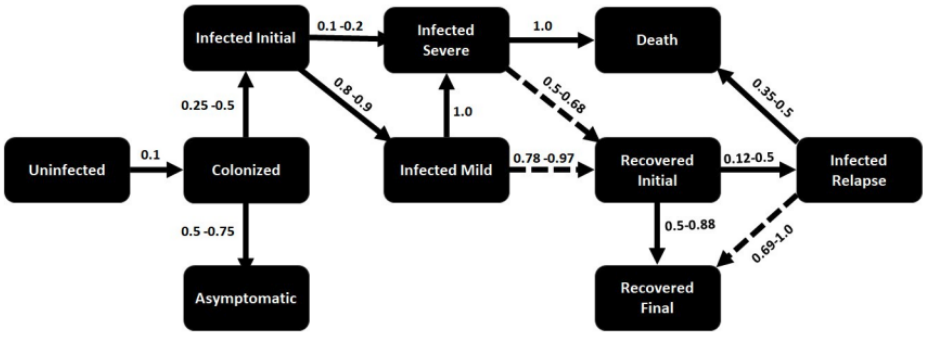

Fig 2. Human infection model for *C. difficile*. Each state in the finite state machine shown above indicates the stages in the infection/recovery process. The arrows indicate possible transition in state and the weight on the arrows indicate the transition probability. The dashed arrows represent transition under medication.

Figure 2: Infection states for people in a realistic two-mode disease model for *C. Diff.* [1]

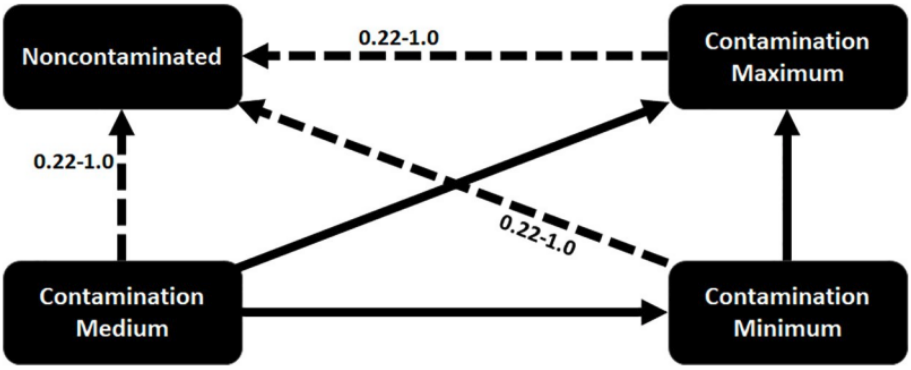

Fig 3. Fomite contamination model for *C. difficile*. Each state in the finite state machine shown above indicates the stages in the contamination/decay process. The solid arrows indicate possible transition in state. The transition between the states depend on number of infected people in the location. The dashed arrows represent transition assuming cleaning.

Figure 3: Infection states for locations in a realistic two-mode disease model for *C. Diff.* [1]

3.1 Optimization Objective

Different from the case of general social network sensors, in which we aim to find the set of sensor nodes that give us the maximum rewards such as peak lead time, in 2-mode models, **the objective is to calculating the optimal probability vector**. Each dimension represents the per day rate of monitoring for a particular type of agents or locations.

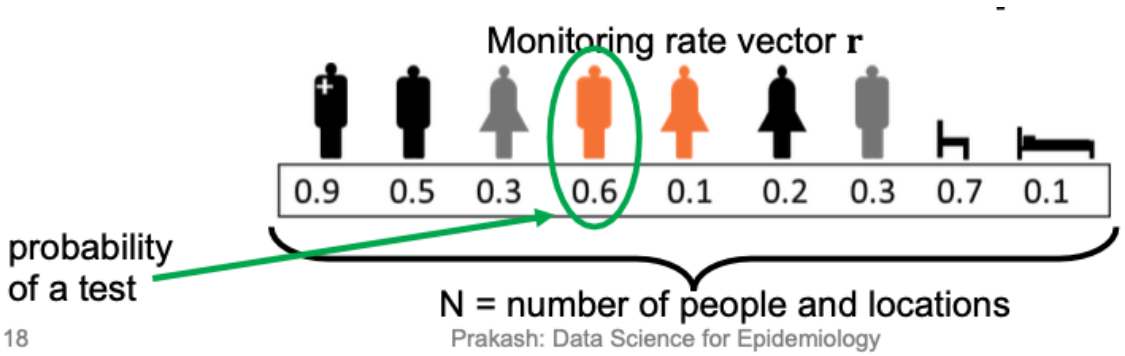

Figure 4: The objective in 2-mode model detection is a vector representing the monitoring probabilities for different types of agents and locations [2]

3.2 Model Calibration and Scenario Generation

We can leverage the floor plan of the hospital and mobility logs of personnels in the building as observations of cascades. However, such data is often sparse and not enough to fit the model. As a result, we need to:

1. **Observation Collection:** obtain mobility log and other sparse observations.
2. **Calibration:** Estimate the parameters of the two-mode disease model using the sparse observations.
3. **Scenario Collection:** Use the calibrated model and the mobility log to produce possible outbreak scenarios

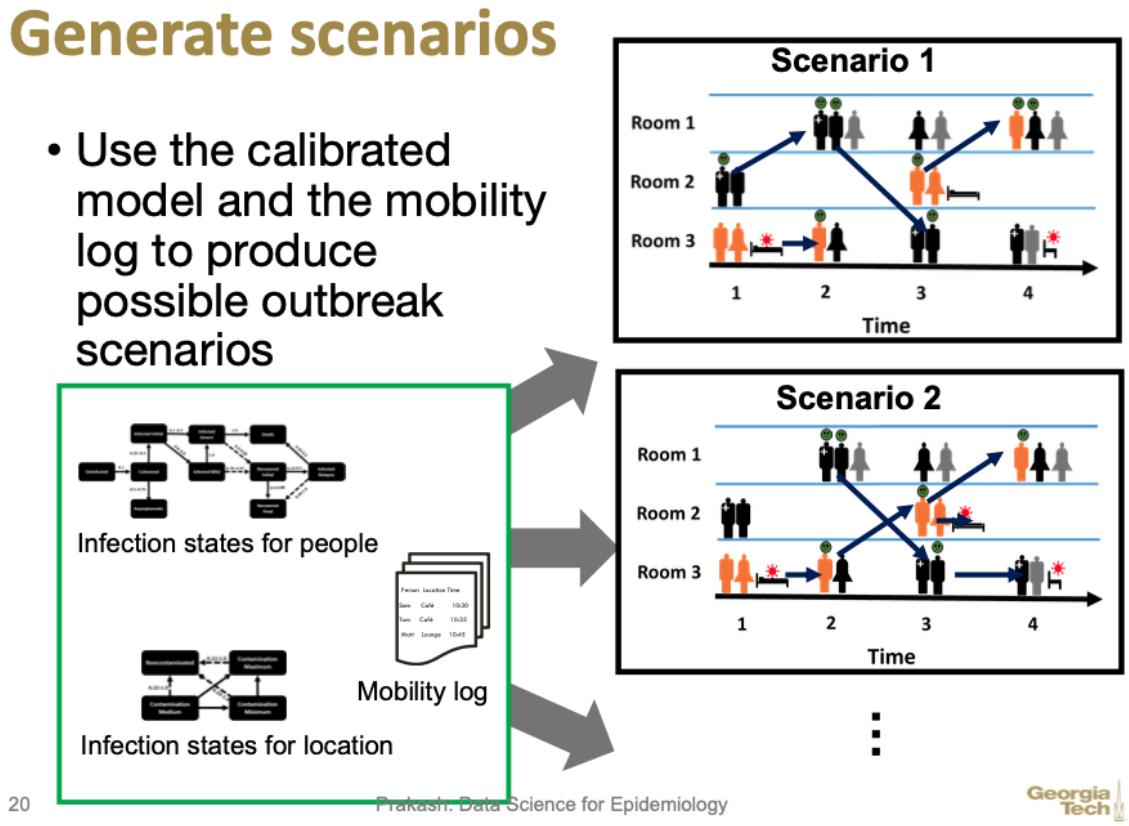

Figure 5: To prepare training data for our model for optimizing monitoring vector, we (1) collect sparse observations, (2) calibrate the model parameters with the observations (3) generate all possible outbreak scenarios. [2]

3.3 Example Approach: HaiDetect

HaiDetect [1] is designed specifically for HAI outbreak monitoring scenarios.

3.3.1 Formal Optimization Problem

The optimization is described formally below:

- **Given:**
  - A set of calibrate scenarios $\mathcal{I}$ (all possible scenarios)
  - The budget on the total number of people and locations to monitor each day
- **Find:**
  - A per day rate vector $\mathbf{r}$ with probability to monitor each node
- **Such that:**
  - The expected number $\varepsilon(\mathcal{I}|\mathbf{r})$ of scenarios detected is **maximized**.

The **final objective** can be formalized as:

$$\mathbf{r}^* = \arg\max_{\mathbf{r}} \varepsilon(\mathcal{I}|\mathbf{r}) \tag{13}$$

However, this is **NP-hard**.

3.3.2 Greedy Algorithm of HaiDetect

To solve the NP-hard problem above, HaiDetect employs a greedy algorithm:

Algorithm 1 HaiDetect

Require: I, budget B

1: **for** each feasible initial vector $\mathbf{r}_0$ **do**
2: Initialize the rate vector $\mathbf{r} = \mathbf{r}_0$
3: **while** $\sum_v \mathbf{r}[v] \cdot \mathbf{c}[v] < B$ **do**
4: Find a node v and rate r maximizing average marginal gain
5: Let $\mathbf{r}[v] = r$
6: Remove all candidate pairs of nodes and rates which are not feasible
7: Return the best rate vector $\mathbf{r}$

Figure 6: HaiDetect is a greedy algorithm. [1]

3.3.3 Why HaiDetect's Greedy Algorithm Works

The greedy algorithm works because the $\varepsilon(\mathcal{I}|\mathbf{r})$ in our final objective function is shown to be a discrete **submodular lattice function**. The definition of the discrete lattice function is out of the scope of the lecture but the lattice function has a property of:

$$f(\mathbf{a}) + f(\mathbf{b}) \geq f(\mathbf{a} \vee \mathbf{b}) + f(\mathbf{a} \wedge \mathbf{b}) \tag{14}$$

where $\mathbf{a}$ and $\mathbf{b}$ are vectors belonging to the same domain space, and $\mathbf{a} \vee \mathbf{b}$ and $\mathbf{a} \wedge \mathbf{b}$ represent the element-wise maximum and minimum of $\mathbf{a}$ and $\mathbf{b}$ respectively.

Note: the equation above strongly resembles the requirement for submodular set functions:

$$(S \cup \{u\}) - f(S) \geq f(S \cup \{u\}) - f(S), \forall S \subseteq T$$

Key: Showing the function is submodular proves the optimality of the greedy approach.

3.3.4 Experiments: Dataset and Setting

After generating all possible scenarios, we partition them into two sets: one is used to select the rate vectors $\mathbf{r}$ using the greedy algorithm and the other is used to measure the success of returned $\mathbf{r}$ on the unseen outbreak scenarios.

3.3.5 Performance: Detection Probability

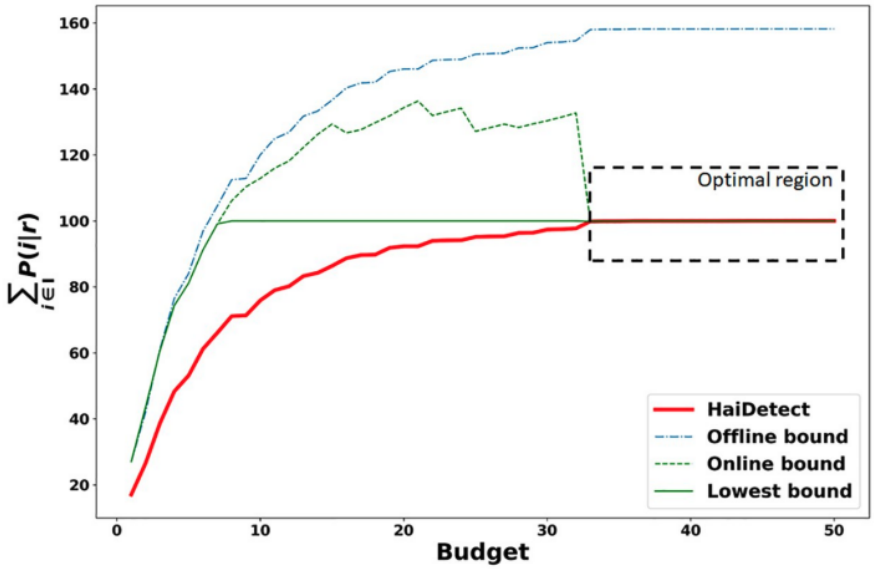

Fig 5. The objective value due to the solution returned by HaiDetect and the bounds for various budgets. The region in the dashed box shows where HaiDetect is optimal.

Figure 7: HaiDetect [1] is optimal in observed cascades.

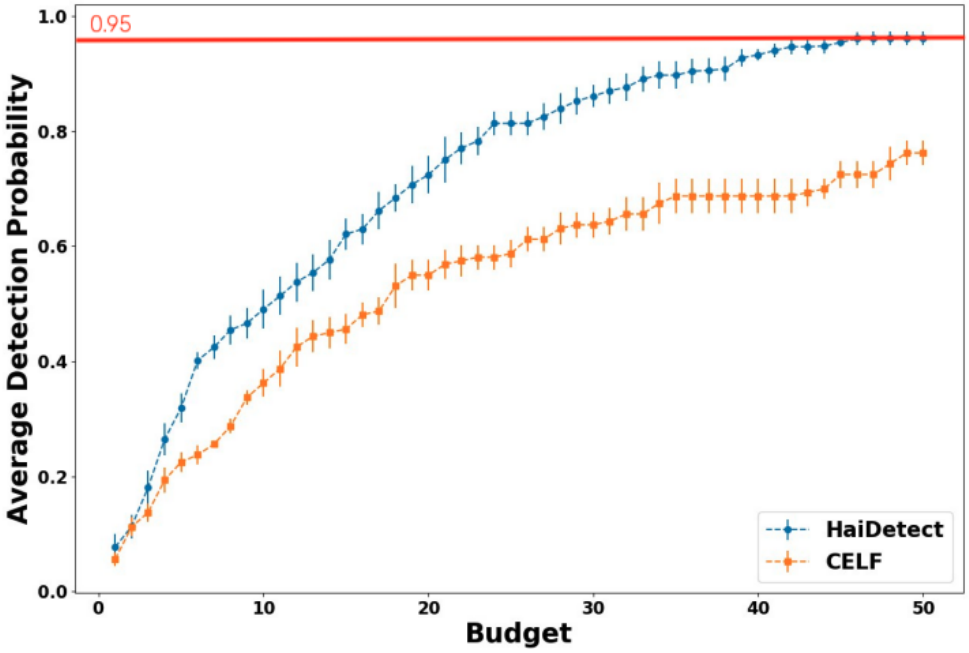

Fig 6. Probability of detecting future outbreaks (normalized) for different budgets. HaiDetect significantly outperforms Celf implying that monitoring sensors selected by HaiDetect have higher chance of detecting an outbreak.

Figure 8: HaiDetect [1] detects 95% of unobserved cascades.

3.3.6 Performance: Detection Time

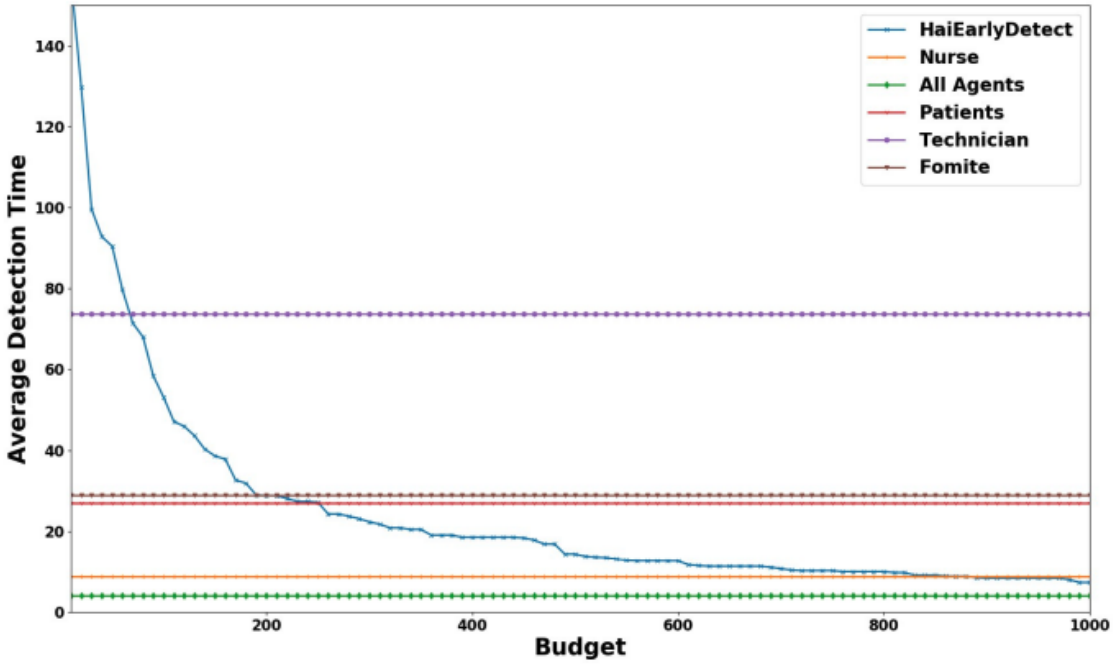

Fig 9. Average dectection time (in days) for various budgets. The flat lines are the detection time for monitoring all members of different categories of agents. Note that for a budget of 1000, monitoring sensors selected by HaiEarlyDetect detect future outbreaks earlier than monitoring all nurses.

Figure 9: HaiDetect [1] outperforms current hospital monitoring practices with much shorter mean detection time.

References

[1] B. Adhikari, B. Lewis, A. Vullikanti, J. M. Jiménez, and B. A. Prakash. Fast and near-optimal monitoring for healthcare acquired infection outbreaks. *PLoS Comput Biol*, 15(9):e1007284, 09 2019.

[2] A. P. B. Cse 8803: Epi data science for epidemiology - lecture 11: Outbreak detection (ii). https://www.dropbox.com/sh/jg48r4y9489oulj/AADakPzG1sH2Icax6AWnKRQPa/?preview=lecture-11.pdf, oct 2022.