Sensor selection for fine-grained behavior verification that respects privacy

Rishi Phatak Dylan A. Shell

***Abstract*—A useful capability is that of classifying some agent's behavior using data from a sequence, or trace, of sensor measurements. The sensor selection problem involves choosing a subset of available sensors to ensure that, when generated, observation traces will contain enough information to determine whether the agent's activities match some pattern. In generalizing prior work, this paper studies a formulation in which multiple behavioral itineraries may be supplied, with sensors selected to distinguish between behaviors. This allows one to pose fine-grained questions, e.g., to position the agent's activity on a spectrum. In addition, with multiple itineraries, one can also ask about choices of sensors where some behavior is always plausibly concealed by (or mistaken for) another. Using sensor ambiguity to limit the acquisition of knowledge is a strong privacy guarantee, a form of guarantee which some earlier work examined under formulations distinct from our inter-itinerary conflation approach. By concretely formulating privacy requirements for sensor selection, this paper connects both lines of work in a novel fashion: privacy—where there is a bound from above, and behavior verification—where sensors choices are bounded from below. We examine the worst-case computational complexity that results from both types of bounds, proving that upper bounds are more challenging under standard computational complexity assumptions. The problem is intractable in general, but we introduce an approach to solving this problem that can exploit interrelationships between constraints, and identify opportunities for optimizations. Case studies are presented to demonstrate the usefulness and scalability of our proposed solution, and to assess the impact of the optimizations.**

I. Introduction

The problems of activity recognition [25], surveillance [15], [20], [24], suspicious and/or anomalous behavior detection [16], fault diagnosis [2], [18], and task monitoring [21] — despite applying to distinct scenarios— all involve the challenge of analyzing behavior on the basis of streams of observations from sensors. Sensor selection and activation problems (as studied by [2], [14], [22], [23]) are concerned with selecting a set of sensors to provide *sufficient* information to reach conclusions that are both unequivocal and correct. Yet, *too much* information may be detrimental — for instance, in elder care and independent living applications (cf. [20]), capturing or divulging sensitive/inappropriate information could calamitous enough to be considered a showstopper.

As a concrete motivating example, consider the house shown in Figure 1. Suppose that it is to be turned, via automation, into a 'smart home' to serve as an assisted living space for an elderly person named Myra. Assume that occupancy sensors triggered by physical presence can be placed in each labelled, contiguous area. We might program a system that uses such sensors to track important properties related to Myra's wellness and health goals so that a carer can be notified if something is amiss. For instance, suppose that to help fend off dementia, Myra does a post-lunch crossword in her study. To determine that Myra has moved through the house and ended up in the study doing her crossword, a single occupancy sensor, STUDY, suffices. Unfortunately, when the pool has just been cleaned, the chlorine negatively affects Myra's sinuses. To ensure that she ends up in the study *and* never visits the swimming pool, we need 2 sensors (STUDY, POOL). The increase makes intuitive sense: we are, after all, now asking for more information about the activity than before. Notice the 3 kinds of behavior that we can now discriminate between: ones that are both safe and desirable (never visiting the pool and ending in the study), ones that are safe but undesirable (never visiting the pool, but also not ending in the study), and ones that are not safe (visiting the chlorinated pool).

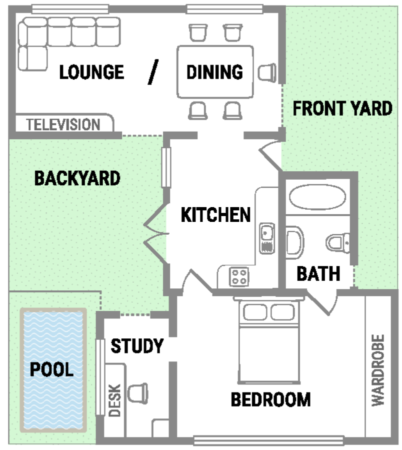


Fig. 1. Myra's assistive living space wherein occupancy detectors can be employed within contiguous areas, corresponding here to eight regions: the POOL, STUDY, BEDROOM, BATHROOM, KITCHEN, LOUNGE/DINING, BACKYARD, and FRONT YARD. Different subsets of detectors result in quite distinct sensing granularities: inadequate sensors will mean that the system is incapable of obtaining information needed about the state of the environment, too many sensors are invasive and cause privacy concerns.

Dinner time is next. We wish to have enough sensing power to tell that Myra has ended up the lounge/dining area, having spent some time in the kitchen. A pair of sensors (KITCHEN, LOUNGE/DINING) will do; and to include the study and pool, these are in addition to the previous 2, giving 4 in total. But alas, now Myra is annoyed: very occasionally, she enjoys a perfectly innocent midnight snack and she feels that any sensor that discloses when she has raided the fridge (and even the frequency of such forays!) is too invasive.[1] She requires that we guarantee that those evenings in which her bedroom is occupied continuously shall appear identical to those in which one (or more) incursions have been made into the kitchen.

Her request, along with the previous requirements, can be met with 5 sensors (LOUNGE/DINING, STUDY, BACKYARD, FRONT YARD, POOL). Though simplistic, this example illustrates

Both authors are affiliated with Dept. of Computer Science & Engineering, Texas A&M University, College Station, TX, USA. {rishi.phatak | dshell}@tamu.edu.

[1]Her concern is not misplaced, given the increasing number of attacks on cloud services in recent years [3] from which stored data may be leaked.

an important idea — it is not enough to reduce the number of sensors to increase privacy, but that sometimes it may be necessary to activate a different and higher cardinality combination of sensors to protect sensitive information.

The present paper re-visits the sensor selection model introduced in the IROS'21 paper of Rahmani et al. [14], advancing and elaborating upon it in order to treat the sort of problem just described. In that paper, the authors consider the setting where a claimant asserts that (future) movements within an environment will adhere to a given itinerary. Then the objective is to select, from some set of sensors at specific locations, a small subset that will detect any deviations from this claim. One of present paper's key advances is the ability to constrain the information obtained from sensors, in order to meet privacy and non-disclosure requirements. Further, the present paper generalizes the problem so that multiple itineraries are considered and, consequently, the objective becomes rather more subtle. In the prior work, the problem is to select sensors that single out the claimed itinerary from all other activity; now, when closely-related itineraries are provided, the sensors selected must have adequate resolving power distinguish fine-grain differentiations (recall the 3 kinds of behavior above).

This paper establishes the computational hardness of sensor selection and optimization under this richer setting (see Section V) giving a nuanced description of its relation to the constraints introduced to modulate the collected information. Then, although the problem is worst-case intractable in general, we introduce an exact method in Section VI which treats the sensor selection problem using automata theoretic tools (an approach quite distinct from the ILP of [14]). Multiple itineraries are provided as input and their interrelationships express constraints — we examine opportunities to exploit aspects of this structure, which leads us to propose some optimizations. The empirical results we present in Section VII show that the improvements obtained from the optimizations are significant, and demonstrate how they help improve the scalability of our proposed solution.

Some detail has necessarily been omitted, the reader is encouraged to refer to [12] for the full authoritative version of the paper.

II. Related works

So far, no single model for robotic privacy has yet emerged. A useful taxonomy dealing with privacy for robots (and associated intelligent systems) appears in [17]. Perhaps most visible candidate is that of differential privacy, used by such works as [4], [13]. There, the underlying formulation builds upon a notion of nearness (originally with a static database of multiple records), and is a less natural fit to treat the problem of altering the processes by which data are acquired. The present work tackles how privacy (of even a single entity) may be preserved without any need for addition of noise if they can exert some degree of control on the tools used to collected that data.

The idea of obscuring or concealing information is another candidate and is prevalent in the control community's notion of opacity: an excellent overview for Discrete Event Systems (DES) is by Jacob, Lesage, and Faure [6]. A DES is said to be opaque if a secret has some level of indistinguishability, a concept very close to the conflation constraints we define in Section III. For further reading in the role of opacity in DES, the reader is referred to [8], [26] and [7].

Previous work by Masopust and Yin affirms that the properties of detectability and opacity are worst case intractable in general [9]. In particular, Cassez et. al. [1] showed that determining the opacity of static and dynamic masks was PSPACE-Complete via formulation of so-called 'state-based' and 'trace-based' opacities. In our work, importantly, simply obfuscating states is not enough, as how that particular state was reached also plays a role. A second factor which differentiates our work is that we allow specifications of constraints between two specified behaviors, instead of making them binary, one-versus-all decisions. An important subtlety, moreover, is that the conflation constraints are directed (cf., also [11]), implying that a more fine grained designation of obfuscation is allowed without necessarily running in both directions. Thus, we find it more suitable to reduce directly from the inclusion problem than universality.

III. Problem statement and definitions

The environment in which some agent of interest moves is modelled as a discrete structure called the *world graph*:

Definition 1 (World Graph [14])**.** A world graph is an edge-labelled, directed multigraph $\mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda)$:$V$ is a non-empty vertex set; E is a set of edges; $\mathrm{src} : E \to V$ and $\mathrm{tgt} : E \to V$ are source and target functions, respectively, identifying a source vertex and target vertex for each edge; $v_0 \in V$ is an initial vertex; $S = \{s_1, s_2, \ldots, s_k\}$ is a nonempty finite set of sensors; $\mathbb{Y} = \{Y_{s_1}, Y_{s_2}, \ldots, Y_{s_k}\}$ is a collection of mutually disjoint event sets associated to each sensor; $\lambda : E \to \wp(Y_{s_1} \cup Y_{s_2} \cup \cdots \cup Y_{s_k})$ is a labelling function, which assigns to each edge, a world-observation a set of events. (Here powerset $\wp(X)$ denotes all the subsets of X.)

The usefulness of the world graph is that it governs two major aspects of the agent's locomotion: how it may move, and what would happen if it moved in a certain way. The agent is known to start its movements at v_0 and take connected edges.

However, the agent cannot make any transitions that are not permitted by the world graph. Myra, for example, cannot jump from the BEDROOM to the LOUNGE/DINING without first going through the KITCHEN. Thus, the collection of all paths that can physically be taken by the agent is defined as follows:

Definition 2 (Walks [14])**.** A string $e_1 e_2 \ldots e_n \in E^*$ is a walk on the world graph if and only if $\mathrm{src}(e_1) = v_0$ and for all $i \in \{1, \ldots, n-1\}$ we have that $\mathrm{tgt}(e_i) = \mathrm{src}(e_{i+1})$. The set of all walks over $\mathcal{G}$ is denoted $\mathrm{Walks}(\mathcal{G})$

Next, we seek to understand what role the sensors play when an agent interacts with the world. Whenever an edge is crossed, it causes a 'sensor response' described by the label on that edge: those sensors which are associated with the sensor values in the label (and are turned on/selected) will emit those values. Returning to the home in Figure 1, assume there are sensors in the BEDROOM and STUDY which measure occupancy. Then, when Myra starts in the bedroom and moves to the study, we would obtain the event $\{\text{BEDROOM}^-, \text{STUDY}^+\}$ for the transition, with the plus superscript representing an event triggered by detection, and minus the inverse. The model also allows sensors other than those which detect occupancy (e.g., non-directed traversals via break beams), see [14] too.

To understand the sensor values generated when crossing a single edge where sensors may be turned off, we use a sensor labelling function:

Definition 3 (Sensor labelling function)**.** Let $\mathcal{G} = (V, E, \text{src}, \text{tgt}, v_0, S, \mathbb{Y}, \lambda)$ be a world graph, and $M \subseteq S$ a sensor selection from it. For selection M, the set of all events that could be produced by those sensors will be denoted $\mathbf{Y}(M) = \bigcup_{s \in M} Y_s$. Then the *sensor labelling function* is for all $e \in E$:

$$\lambda_M(e) = \begin{cases} \lambda(e) \cap \mathbf{Y}(M) & \text{if } \lambda(e) \cap \mathbf{Y}(M) \neq \varnothing, \\ \epsilon & \text{otherwise.} \end{cases}$$

(Note that ϵ here is the standard empty symbol.) Later in the paper, Figure 2 forms an example of an environment with a world graph whose edges bear appropriate sensor labels.

Now, we may formally define the signature function for a walk and a given sensor set as follows:

Definition 4 (Signature of a walk [14])**.** For a world graph $\mathcal{G} = (V, E, \text{src}, \text{tgt}, v_0, S, \mathbb{Y}, \lambda)$ we define function $\beta_\mathcal{G}$: $\text{Walks}(\mathcal{G}) \times \wp(S) \to (\wp(\mathbf{Y}(S)) \setminus \{\varnothing\})^*$ such that for each $r = e_1 e_2 \dots e_n \in \text{Walks}(\mathcal{G})$ and $M \subseteq S$, $\beta_\mathcal{G}(r, M) = z_1 z_2 \dots z_n$ in which for each $i \in \{1, \dots, n\}$, we have that $z_i = \lambda_M(e_i)$.

The behavior of the agent will be specified with respect to a given world graph and these specifications will describe sequences of edges the agent may decide to take in the world graph. Following the convention of [14], each is called an itinerary. Subsequent definitions will involve the use of multiple itineraries in order to constrain what information about the agent's behavior the sensors are allowed to obtain.

Definition 5 (Itinerary DFA [14])**.** An itinerary DFA over a world graph $\mathcal{G} = (V, E, \text{src}, \text{tgt}, v_0, S, \mathbb{Y}, \lambda)$ is a DFA $\mathcal{I} = (Q, E, \delta, q_0, F)$ in which Q is a finite set of states; E is the alphabet; $\delta : Q \times E \to Q$ is the transition function; q_0 is the initial state; and F is the set of accepting (final) states.

With the basic elements given, the next four definitions formalize the different classes of constraints we desire a set of sensors to satisfy. Conflation constraints allow one type of behavior to 'appear' similar to another, while discrimination constraints specify that two behaviors must be distinguishable.

Definition 6 (Conflation constraint)**.** A conflation constraint on a world graph $\mathcal{G}$ is an ordered pair of itineraries $(\mathcal{I}_a, \mathcal{I}_b)^\boxminus$.

Definition 7 (Discrimination constraint)**.** A discrimination constraint on a world graph $\mathcal{G}$ is an unordered pair of itineraries $[\mathcal{I}_1, \mathcal{I}_2]^\boxtimes$.

Both types will combined within a graph:

Definition 8 (Discernment designation)**.** A *discernment designation* is a mixed graph $\mathcal{D} = (I, I_D, I_C)$, with vertices I being a collection of itineraries, along with undirected edges I_D which are a set of discrimination constraints, and arcs (directed edges) I_C which are a set of conflation constraints.

And, finally, we can state what a satisfying selection entails:

Definition 9 (Satisfying sensor selection)**.** Given some discernment designation $\mathcal{D}$, a sensor set $M \subseteq S$ is a *satisfying sensor selection for* $\mathcal{D} = (I, I_D, I_C)$ if and only if both of the following conditions hold:

- For each $[\mathcal{I}_1, \mathcal{I}_2]^\boxtimes \in I_D$ we have that there exist no $w_1 \in \text{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_1)$ and $w_2 \in \text{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_2)$ where $\beta_\mathcal{G}(w_1, M) = \beta_\mathcal{G}(w_2, M)$.
- For each $(\mathcal{I}_a, \mathcal{I}_b)^\boxminus \in I_C$ we have that for every $w \in \text{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_a)$, there exists $c_w \in \text{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_b)$ where $\beta_\mathcal{G}(w, M) = \beta_\mathcal{G}(c_w, M)$.

In the above definition, the '$\boxtimes$' constraints correspond to *discrimination* requirements, while '$\boxminus$' require *conflation*. The importance of the set intersections is that the only things that can really happen are walks on the world graph. When there is a discrimination constraint, there are no walks from the one itinerary that can be confused with one from the other itinerary. When there is a conflation constraint, any walk from the first itinerary has at least one from the second that appears identical. Conflation models privacy in the following sense: any putative claim that the agent followed one itinerary can be countered by arguing, just as plausibility on the basis of the sensor readings, that it followed the other itinerary. While the discrimination constraint is symmetric, the second need not be. (Imagine: $\{\beta_\mathcal{G}(w, M) | w \in \text{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_1)\} = \{a, b, c, d\}$ while $\{\beta_\mathcal{G}(w', M) | w' \in \text{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_2)\} = \{a, b, c, d, e\}$. Then $(\mathcal{I}_1, \mathcal{I}_2)^\boxminus$ is possible, while $(\mathcal{I}_2, \mathcal{I}_1)^\boxminus$ is not.)

Now, we are ready to give the central problem of the paper:

Decision Problem: Minimal sensor selection to accommodate a discernment designation in itineraries (MSSADDI)

Input: A world graph $\mathcal{G}$, a discernment designation $\mathcal{D}$, and a natural number $k \in \mathbb{N}$.

Output: A satisfying sensor selection $M \subseteq S$ for $\mathcal{D}$ on $\mathcal{G}$ with $|M| \leq k$, or 'INFEASIBLE' if none exist.

IV. SIGNATURE AUTOMATA

To understand how we may begin solving MSSADDI and what its theoretical complexity is, we introduce the concept of a signature automaton. Signature automata are produced from the product automata of an itinerary with the world graph:

Definition 10 (Product automaton [14])**.** Let $\mathcal{G} = (V, E, \text{src}, \text{tgt}, v_0, S, \mathbb{Y}, \lambda)$ be a world graph and $\mathcal{I} =$

(Q, E, δ, q_0, F) be an intinerary DFA. The product $\mathcal{P}_{\mathcal{G},\mathcal{I}}$ is a partial DFA $\mathcal{P}_{\mathcal{G},\mathcal{I}} = (Q_\mathcal{P}, E, \delta_\mathcal{P}, q_0^\mathcal{P}, F_\mathcal{P})$ with $Q_\mathcal{P} = Q \times V$; $\delta_\mathcal{P} : Q_\mathcal{P} \times E \to Q_\mathcal{P} \cup \{\perp\}$ is a function such that for each $(q, v) \in Q_\mathcal{P}$ and $e \in E$; $\delta_\mathcal{P}((q, v), e)$ is defined to be $\perp$ if $\mathrm{src}(e) \neq v$, otherwise, $\delta_\mathcal{P}((q, v), e) = (\delta(q, e), \mathrm{tgt}(e))$; $q_0^\mathcal{P} = (q_0, v_0)$, and $F_\mathcal{P} = F \times V$.

The language of this product automaton, as a DFA, is the collection of (finite-length) sequences from E that can be traced starting at $q_0^\mathcal{P}$, never producing a $\perp$, and which arrive at some element in $F_\mathcal{P}$. The language recognized is the set of walks that are within the itinerary $\mathcal{I}$, i.e., $\mathcal{L}(\mathcal{P}_{\mathcal{G},\mathcal{I}}) = \mathrm{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I})$.

Definition 11 (Signature automaton)**.** Let $\mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda)$ be a world graph, let $M \subseteq S$ be a sensor selection on it, $\mathcal{I} = (Q, E, \delta, q_0, F)$ be an itinerary DFA, and $\mathcal{P}_{\mathcal{G},\mathcal{I}}$ be their product. A signature automaton $\mathcal{S}_{\mathcal{G},\mathcal{I},M} = (Q_\mathcal{P}, \Sigma, \delta_\mathcal{S}, q_0^\mathcal{P}, F_\mathcal{P})$ is a nondeterministic finite automaton with ϵ-moves (NFA-ϵ) with

- $\Sigma = \{\lambda_M(e) \mid e \in E, \lambda_M(e) \neq \epsilon\}$
- $\delta_\mathcal{S} : Q_\mathcal{P} \times \Sigma \cup \{\epsilon\} \to \wp(Q_\mathcal{P})$ is a function defined for each $(q, v) \in Q_\mathcal{P}$ and $\sigma \in \Sigma \cup \{\epsilon\}$ such that

$$\delta_\mathcal{S}\big((q, v), \sigma\big) = \Big\{\delta_\mathcal{P}\big((q, v), e\big) \;\Big|\; e \in E, \delta_\mathcal{P}\big((q, v), e\big) \neq \perp, \lambda_M(e) = \sigma\Big\}.$$

The signature automaton produces all the signatures that could result from following a path in the world graph conforming to the given itinerary. Formally, we have the following:

Lemma 1. *For world graph $\mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda)$, sensor selection $M \subseteq S$, and itinerary $\mathcal{I} = (Q, E, \delta, q_0, F)$, if their signature automaton is $\mathcal{S}_{\mathcal{G},\mathcal{I},M}$, then:*

$$\mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I},M}) = \{\beta_\mathcal{G}(w, M) \mid w \in \mathrm{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I})\}.$$

Proof. For all $w \in \mathrm{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I})$ there is a unique sequence of states $q_0^\mathcal{P}, \ldots, q_n^\mathcal{P}$ in $\mathcal{P}_{\mathcal{G},\mathcal{I}}$ such that $q_n^\mathcal{P} \in F_\mathcal{P}$. Following that sequence through the signature automaton returns signature $\beta_\mathcal{G}(w, M)$. Similarly, any string that is accepted by $\mathcal{S}_{\mathcal{G},\mathcal{I},M}$ has a sequence of states $q_0^\mathcal{P}, \ldots, q_n^\mathcal{P}$ in $\mathcal{S}_{\mathcal{G},\mathcal{I},M}$ such that $q_n^\mathcal{P} \in F_\mathcal{P}$. Following those states through $\mathcal{P}_{\mathcal{G},\mathcal{I}}$ returns the walk conforming to the itinerary which produced it. ☐

Note that signature automaton simply replaces the alphabet E of the product automaton with the alphabet Σ. This introduces nondeterminism in the automaton because two outgoing edges from a vertex in the world graph may produce the same (non-empty) sensor values. Moreover, certain transitions may be made on the empty symbol if no sensor values are produced upon taking an edge in the world graph too.

The preceding is useful owing to the next pair of lemmas.

Lemma 2. *Given world graph $\mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda)$ and itinerary DFAs: $\mathcal{I}^1 = (Q^1, E, \delta^1, q_0^1, F^1)$ and $\mathcal{I}^2 = (Q^2, E, \delta^2, q_0^2, F^2)$, a subset of sensors $M \subseteq S$ is a satisfying sensor selection for constraint discrimination of itineraries $\mathcal{I}^1$ and $\mathcal{I}^2$ if and only if $\mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M}) \cap \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M}) = \varnothing$.*

Proof. Assume that M satisfies the constraint $[\mathcal{I}_1, \mathcal{I}_2]^\boxtimes$. This implies that there exist no w_1 and w_2, with $w_1 \in \mathrm{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_1)$ and $w_2 \in \mathrm{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_2)$, where $\beta_\mathcal{G}(w_1, M) = \beta_\mathcal{G}(w_2, M)$. The previous fact along with Lemma 1 implies $\mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M}) \cap \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M}) = \varnothing$. The other way: if such w_1 and w_2 can be found, then letting $c = \beta_\mathcal{G}(w_1, M) = \beta_\mathcal{G}(w_2, M)$, we have that $\{c\} \subseteq \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M}) \cap \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M})$. ☐

Notice that if $\mathcal{L}(\mathcal{I}_1) \cap \mathcal{L}(\mathcal{I}_2) \neq \varnothing$ then any walks $w_1 = w_2$ taken from this intersection must have $\beta_\mathcal{G}(w_1, M) = \beta_\mathcal{G}(w_2, M)$. Any two itineraries with overlapping languages, and whose overlap falls (partly) within the set of walks, will yield a sensor selection problem that must be infeasible when these itineraries are given as a discrimination constraints.

A similar lemma follows for the conflation constraints.

Lemma 3. *Given world graph $\mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda)$ and itinerary DFAs: $\mathcal{I}^1 = (Q^1, E, \delta^1, q_0^1, F^1)$ and $\mathcal{I}^2 = (Q^2, E, \delta^2, q_0^2, F^2)$, a subset of sensors $M \subseteq S$ is a satisfying sensor selection for constraint conflation of itineraries $\mathcal{I}^1$ and $\mathcal{I}^2$ if and only if $\mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M}) \subseteq \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M})$.*

Proof. Assume that M satisfies the constraint $(\mathcal{I}_1, \mathcal{I}_2)^\boxminus$. This implies that every $w \in \mathrm{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_1)$ has a $c_w \in \mathrm{Walks}(\mathcal{G}) \cap \mathcal{L}(\mathcal{I}_2)$ with $\beta_\mathcal{G}(w, M) = \beta_\mathcal{G}(c_w, M)$. The previous fact along with Lemma 1 implies $\mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M}) \subseteq \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M})$. In the opposite direction, if there exists a w for which no c_w can be found, we know that $\mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M}) \not\subseteq \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M})$ since $\beta_\mathcal{G}(w, M) \in \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M})$ but $\beta_\mathcal{G}(w, M) \notin \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M})$. ☐

V. Complexity of MSSADDI

A. Background and preliminaries

Before we prove the hardness of MSSADDI, we state some known facts from automata and complexity theory.

Lemma 4 (Savitch's Theorem [19])**.** *In the context of complexity classes, we have that* PSPACE = NPSPACE.

Lemma 5 (NFA intersection [5])**.** *Given two non-deterministic finite automata (NFAs) $\mathcal{A}$ and $\mathcal{B}$, it can be determined in polynomial time if $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}) = \varnothing$.*

Lemma 6 (NFA inclusion [10])**.** *Given two non-deterministic finite automata (NFAs) $\mathcal{A}$ and $\mathcal{B}$, it is* PSPACE*-Complete to determine if $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$.*

B. Hardness of MSSADDI

Next, we investigate the hardness of the problem formulated above. Since the original original MSSVI problem [14] is NP-Complete (it essentially involves a single itinerary and its complement, one discrimination constraint, and zero conflation constraints), we naturally expect the problem to be NP-Hard. And this is indeed true (though the direct proof is straightforward and, hence, omitted). For the full problem, the question is whether the conflation constraints contribute additional extra complexity. The answer is in the affirmative, under standard computational complexity assumptions:

Lemma 7. *MSSADDI is in* PSPACE.

Proof. The full detailed proof appears in [12]. □

Next, for showing hardness, we reduce from the NFA inclusion problem. One can think of this intuitively as showing that conflation constraints, in solving the inclusion problem on signature automata, cover worst-case instances.

Lemma 8. *MSSADDI is* PSPACE*-Hard*

Proof. We reduce from NFA Inclusion, known to be PSPACE-Complete (Lemma 6). Given an NFA Inclusion Problem instance $x = \langle \mathcal{A} = (Q_A, \Sigma, \delta_A, q_0^A, F_A), \mathcal{B} = (Q_B, \Sigma, \delta_B, q_0^B, F_B)\rangle$ we form an instance of MSSADDI $f(x) = \langle \mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda), \mathcal{D} = (I, I_D, I_C), k\rangle$.

Every state of $\mathcal{A}$ and $\mathcal{B}$ will be assumed to reachable from their respective start states (unreachable states do not contribute to the NFA's language, and are easily trimmed). We construct $\mathcal{G}$ as follows:—

1) Let the vertex set be $V = \{v_0\} \cup Q_A \cup Q_B$ where v_0 is a new vertex not in either Q_A or Q_B.
2) Let the edge set be $E = \{e_A, e_B\} \cup \{e_1, e_2, \ldots, e_n, e_{n+1}, e_{n+2}, \ldots, e_{n+m}\}$. Here e_A is an edge that connects v_0 to q_0^A and e_B is an edge connecting v_0 to q_0^B. Assuming there are n transitions in $\mathcal{A}$ of the form $q_j^A \in \delta_A(q_i^A, \sigma)$, we produce an edge e_k for some $1 \le k \le n$ which connects q_i^A to q_j^A for every such σ. Similarly, if there are m transitions in $\mathcal{B}$ of the form $q_j^B \in \delta_B(q_i^B, \sigma)$, we would have an edge e_{n+k} for some $1 \le k \le m$ connecting q_i^B to q_j^B for each σ. The src and tgt functions are defined appropriately for all edges.
3) Let sensor set $S = \{s_1, \ldots, s_{|\Sigma|}\}$ where each sensor produces exactly one event so that if $\Sigma = \{\sigma_1, \ldots, \sigma_{|\Sigma|}\}$ then $Y_{s_i} = \{\sigma_i\}$ and $\mathbb{Y} = \{Y_{s_1}, \ldots, Y_{s_{|\Sigma|}}\}$.
4) The edge labelling function is defined as follows. First, let $\lambda(e_A) = \lambda(e_B) = \varnothing$. Then, for each transition in $\mathcal{A}$ of the form $q_j^A \in \delta_A(q_i^A, \sigma)$, if $\sigma = \epsilon$, label that edge with $\varnothing$, otherwise label it with the singleton set $\{\sigma\}$ for all such σ. Follow the same procedure again for $\mathcal{B}$. Note that, by construction, a single sensor may cover an edge from both $\mathcal{A}$ and $\mathcal{B}$. This is natural as the given NFAs share the alphabet Σ. Importantly: this does not violate the assumption that sensors have pairwise distinct readings. Turning some sensor on, means we receive its readings from both regions—that constructed from $\mathcal{A}$ *and* $\mathcal{B}$—or, when turned off, from neither.

The following define $\mathcal{D}$, the discernment designation:—

1) In the world graph $\mathcal{G}$ constructed in the previous step, let there be $p \le n+m$ edges collected as $\{e_{i_1}, e_{i_2}, \ldots, e_{i_p}\}$ where we have that each of them has a non-empty label, i.e., $e_{i_k} \in E$, and $\lambda(e_{i_k}) \ne \varnothing$ for every $1 \le k \le p$. Then let the set of itineraries I be $\{I_{e_{i_1}}, I_{e_{i_2}}, \ldots, I_{e_{i_p}}\} \cup \{I_{e^+_{i_1}}, I_{e^+_{i_2}}, \ldots, I_{e^+_{i_p}}\} \cup \{I_A, I_B\}$, where we will give the language accepted by each DFA. The first $2p$ elements have a language with a single string: for $1 \le k \le p$, to determine the languages $\mathcal{L}(I_{e_{i_k}})$ and $\mathcal{L}(I_{e^+_{i_k}})$, run a breadth first search (BFS) from v_0 on $\mathcal{G}$. This co-routine will return the shortest path (consisting of specific edges) from v_0 to $\mathrm{src}(e_{i_k})$. This path is the only string accepted by $I_{e_{i_k}}$, and the same path but with e_{i_k} appended is the only string accepted by $I_{e^+_{i_k}}$.
Next, itinerary DFA I_A is to be defined so it accepts a string $e_{i_1} e_{i_2} \ldots e_{i_r}$ where $e_{i_k} \in E$ for all $1 \le k \le r$ if and only if $\mathrm{tgt}(e_{i_r}) \in F_A$. Similarly, define DFA I_B so that it accepts a string $e'_{i_1} e'_{i_2} \ldots e'_{i_q}$ where $e'_{i_k} \in E$ for all $1 \le k \le q$ if and only if $\mathrm{tgt}(e'_{i_q}) \in F_B$. Note that we are not asking for the given NFAs $\mathcal{A}$ and $\mathcal{B}$ to be converted to DFAs — instead, we are simply constructing a DFA which recognizes that some *path* of an accepting string arrives at an accepting state in the NFA. The construction of such a DFA is simple: For I_A, define two states q_0 and q_1, with only q_1 accepting. Then, define transitions from q_0 to q_1 and q_1 to q_1 for all $e \in E$ such that $\mathrm{tgt}(e)$ is a final state in $\mathcal{A}$. Similarly, define transitions from q_0 to q_0 and q_1 to q_0 for all $e \in E$ such that $\mathrm{tgt}(e)$ is not a final state in $\mathcal{A}$. Doing the same for $\mathcal{B}$ gives I_B.
2) Define $I_D = \left\{[I_{e_{i_1}}, I_{e^+_{i_1}}]^{\boxtimes}, \ldots, [I_{e_{i_p}}, I_{e^+_{i_p}}]^{\boxtimes}\right\}$.
3) Finally, define $I_C = \left\{(I_A, I_B)^{\boxminus}\right\}$.

Lastly, let $k = |\Sigma|$.

This three-piece mapping is accomplished in polynomial time since the size of the world graph is $O(1+|\mathcal{A}|+|\mathcal{B}|)$ and the size of $\mathcal{D}$ (i.e., the number of constraints) is $O(|\mathcal{A}|+|\mathcal{B}|)$.[2] Since BFS runs in polynomial time on $\mathcal{G}$, all the discrimination requirements need polynomial time to construct and each is of polynomial size. For the itineraries in the conflation constraints, the DFAs have 2 states and $|E|$ transitions.

Finally, to prove correctness: there must be a satisfying sensor selection of size at most k if and only if $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$.

($\Longrightarrow$) Assume that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. Then the sensor selection $M = S$ is a satisfying sensor selection because, firstly, $|M| = |\Sigma| = k$. Secondly, note that all the discrimination constraints are satisfied because all the sensors are turned on. Lastly, the conflation constraint is also satisfied by reasoning as follows: any walk beginning at v_0 first going to q_0^A and ending at some $v \in F_A$ has a signature $\{\sigma_1\}\{\sigma_2\}\ldots\{\sigma_m\}$ for which $\sigma_1\sigma_2\ldots\sigma_m \in \mathcal{L}(\mathcal{A})$ which implies $\sigma_1\sigma_2\ldots\sigma_m \in \mathcal{L}(\mathcal{B})$. But, by construction, one can take a path in the world graph, taking a first step from v_0 to q_0^B without producing any sensor value, and then follow exactly the same path that is accepting in $\mathcal{B}$ through the world graph, and this path will produce signature $\{\sigma_1\}\{\sigma_2\}\ldots\{\sigma_m\}$.

($\Longleftarrow$) Assume there exists some satisfying sensor selection of size less than or equal to $k = |\Sigma|$. Firstly, no sensor may be turned off since doing so would violate the discrimination constraint between the singleton itineraries involving the edge(s) labelled with the disabled sensor's value. Thus, the sensor selection has size exactly k. Secondly, the conflation constraint is also met implying that, for all signatures $\{\sigma_1\}\{\sigma_2\}\ldots\{\sigma_m\}$ produced by taking v_0 to q_0^A and ending at some $v_i \in F_A$, there exists a path from v_0 to q_0^B ending at $v_j \in F_B$ such that its

[2]Here, $|\cdot|$ gives the number of transitions or states, whichever is greater.

signature is also $\{\sigma_1\}\{\sigma_2\}\ldots\{\sigma_m\}$. Since no sensor is turned off, the paths that obtain the signatures in the world graph can be taken in $\mathcal{A}$ and $\mathcal{B}$ as well, so $\sigma_1\sigma_2\ldots\sigma_m \in \mathcal{L}(\mathcal{A})$ implies $\sigma_1\sigma_2\ldots\sigma_m \in \mathcal{L}(\mathcal{B})$, thus $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. □

Theorem 1. *MSSADDI is* PSPACE*-Complete.*

Proof. Follows from Lemmas 7 and 8. □

VI. Algorithm Description

Having proved the theoretical complexity class of MSSADDI, we now turn to a description of the algorithm we used to solve it. Although the algorithm is not polynomial time (as, assuming P $\neq$ PSPACE, it couldn't be) we introduce several optimizations to help ameliorate its running time.

A. Baseline Algorithm

The approach we chose for solving MSSADDI was a complete enumeration of subsets, with some shortcutting. The pseudo-code, based directly on the automata theoretic connections identified in the preceding, appears in Algorithm 1.

Algorithm 1 Complete Enumeration for MSSADDI

Inputs: A world graph $\mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda)$ and a discernment designation $\mathcal{D} = (I, I_D, I_C)$
Output: The minimum satisfying sensor selection, if it exists, otherwise **null**

1: $M^* \leftarrow$ **null** ▷ *The current best sensor set*
2: **for** $k = |S|$ **down to** 0 **do**
3:     **for** M in COMBINATIONS(S, k) **do**
4:         **for** $[\mathcal{I}^1, \mathcal{I}^2]^{\boxtimes} \in I_D$ **do**
5:             $\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M} \leftarrow$ SIGNATUREAUTOMATON$(\mathcal{G}, \mathcal{I}^1, M)$
6:             $\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M} \leftarrow$ SIGNATUREAUTOMATON$(\mathcal{G}, \mathcal{I}^2, M)$
7:             **if** $\mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M}) \cap \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M}) \neq \varnothing$ **then**
8:                 **Continue** to next M ▷ *Check next combination*
9:         **for** $(\mathcal{I}_1, \mathcal{I}_2)^{\boxminus} \in I_C$ **do**
10:             $\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M} \leftarrow$ SIGNATUREAUTOMATON$(\mathcal{G}, \mathcal{I}^1, M)$
11:             $\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M} \leftarrow$ SIGNATUREAUTOMATON$(\mathcal{G}, \mathcal{I}^2, M)$
12:             **if** $\mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^1,M}) \nsubseteq \mathcal{L}(\mathcal{S}_{\mathcal{G},\mathcal{I}^2,M})$ **then**
13:                 **Continue** to next M ▷ *Check next combination*
14:         **if** All I_D and I_C satisfied **then**
15:             $M^* \leftarrow M$
16:             **Continue** to next k ▷ *Now try sets of size $k-1$*
17:     **if** No M where $|M| = k$ satisfies all I_D **then**
18:         **return** M^* ▷ *Prior solution was smallest feasible one*
19: **return** M^* ▷ *Final exit*

It is a top down search over all subsets of S where we attempt to check each constraint by constructing its signature automaton and verifying the intersection and subset properties, lines 7 and 12, respectively, as in the previous sections. Discrimination constraints are checked first (lines 4–8) because we expect them to be easier to check than conflation constraints (Lemmas 5 and 6).

We take advantage of one more property of sensor sets in relation to discrimination constraints to define our baseline algorithm. Since we stipulate that different sensors produce different sensor outputs, it follows that if $M \subseteq S$ does not satisfy a discrimination constraint, then neither can any subset of M. Therefore, when no combination of sensors of size k satisfies *all* the discrimination constraints, the search is ended, and the current best satisfying sensor set returned (line 18).

Next, we propose two optimizations over the baseline algorithm just described. While each does involve a different trade-off, neither sacrifices the correctness guarantee.

B. The Caching Optimization

Notice how the signature automaton is constructed each time an itinerary is encountered in a constraint (lines 5–6 and 10–11). This seems to be wasteful if an itinerary appears in multiple constraints (as it can be with several). The signature automaton can be cached after it is constructed should the same itinerary appear in another constraint, allowing it to be retrieved without the need for additional computation.

Note, however, the trade-off being made here: while the running time reduced, the space requirements increased. Typical library implementations allow for language intersection and subset properties to be checked only on DFA's which, when converted, can result in an exponential increase in space requirements.

C. The Adaptive Weights Optimization

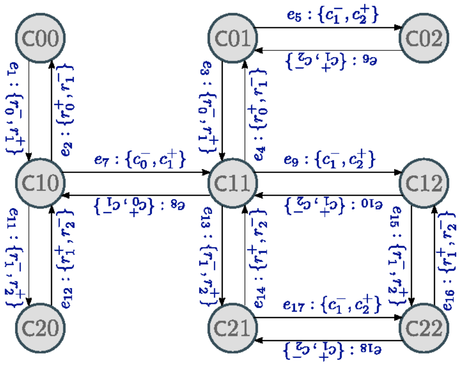
The second optimization we introduce is a dynamic reordering of constraints. Inspired by classical methods in AI for constraint satisfaction problems (CSP's) which seek to make the current assignment *fail fast*, we devised an adaptive weighting mechanism for the desired discernment graph.

Seeking to end the search as fast as possible, discrimination constraints are checked first in the hopes that if none of the sensor sets of cardinality k satisfies the discrimination constraints, then the search can be declared hopeless and ended immediately. Once a satisfying sensor set is found for the discrimination constraints, though, the following strategy is used. Whenever a particular constraint fails to be satisfied, that sensor set 'votes' the erring constraint up so that future sets know which constraint is likely to fail. Thus, after a few iterations, enough data is collected so that a sensor set checks that constraint first which most of the sets before it failed on. The idea is the more demanding (or stringent) constraints are learned and propagated upward for prioritization.

VII. Experimental results

The following experiments were all performed on a computer running Windows 11 with an Intel i7 CPU having 16 GB RAM using Python 3.

As a basic sanity check, we ran the baseline algorithm on the problems presented in Section I. For these problems, the algorithm correctly provided the optimal solutions in less than 1 s. Next, to test the scalability of the proposed approach and to assess the impact of the optimizations, we ran the experiments that follow.



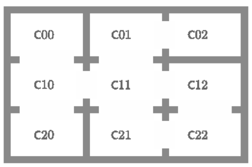


Fig. 2. An example world for testing the algorithm's scaling. This is the 3×3 case, with blueprint of the map at left, and labelled world graph shown above. There are 3 row and 3 column sensors available which detect the agent's presence. The problem is choose a subset of these.

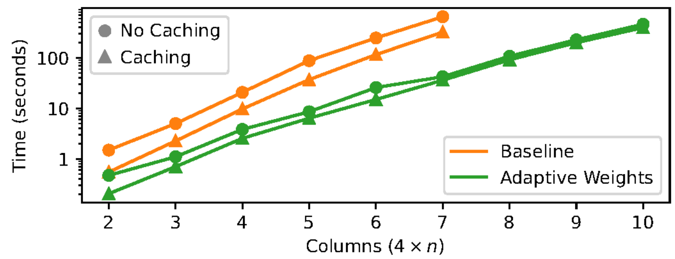


Fig. 3. Effect of all the optimizations for various grid sizes.

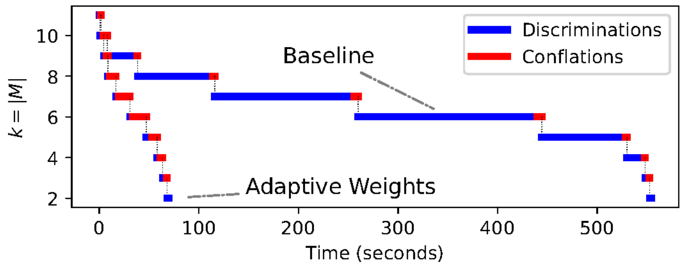


Fig. 4. Effect of dynamically reordering constraints when checking each sensor combination. The horizontal axis shows the progression of time and the vertical axis the size of the sensor set being checked.

A. Test cases

The test cases we propose are designed such that they are parameterized: we use an $m \times n$ grid-type world graph. An example with $m = n = 3$ is shown in Figure 2, with the scaled versions adding cells rightward and downward (without any missing edges unlike the figure). There is a sensor in each row that registers the fact that agent is present within the associated row. Similarly, a column sensor detects when the agent is within that column. Sensor set S consists of $m + n$ sensors, one for each row and each column. The figure shows the labelled world graph, this small instance with 18 edges, the arcs each bearing their λ-based labelling. These follow a simple pattern: for example, r_2^+ means that row 2's sensor has triggered, going from the unoccupied to occupied state; while c_1^- means that column 1's sensor has gone from the occupied to unoccupied.

Finally, we construct an itinerary for every state in the world graph where the language accepted by the DFA for the itinerary describes following any edge in the world graph any number of times followed by an edge incoming to this state. Essentially, the itinerary DFA for that state accepts a string of edges if and only if the last edge that was taken in that walk was an incoming edge to that state.

The number of constraints are proportional to the number of states in the world graph. We add mn discrimination constraints each by randomly selecting any 2 itineraries which describe ending in two states which are in a different column *and* in a different row. Similarly, we also add m conflation constraints per column, each between 2 random itineraries that describe ending in different rows in that column. Thus, in expectation, each itinerary is in 2 discrimination constraints and 2 conflation constraints.

B. Solutions

From the description of the problem above, it should be clear that activating either only the row sensors or only the column sensors should be a satisfying sensor selection for the discrimination constraints alone. After all, ending in a different row and column can be distinguished on the basis of either information provided by a row sensor or a column sensor. However, when considering both the discrimination and conflation constraints, only one of these options becomes feasible — namely, that involving only activating the column occupancy sensors. Activating a row sensor could potentially violate some conflation constraints which describe ending in that row. Note that we see another detail of MSSADDI reiterated here — that when $n > m$, it may be necessary to activate more sensors (i.e., column sensors as opposed to only the row sensors) to satisfy the both upper and lower information bounds as opposed to the lower bounds alone.

C. Analysis

The basic scaling plot for various grid sizes is shown in Figure 3. As can be seen in that plot, using the caching optimization alone led on average to a $53.5\,\%$ reduction in the running time. For our purposes, all the signature automata were able to be cached, and memory did not seem to be an issue (i.e., we never received an out-of-memory exception). Thus, time, not space, seemed to be a dominating factor in solving this problem with current resources.

The results are even more impressive for the adaptive weights optimization. As compared to the baseline algorithm, it led on average to a $87.6\,\%$ improvement in running time. When both optimizations are applied together, however, caching the signature automata seems to have little effect when adaptive weights are already in use. This makes sense because the adaptive weights allow a sensor set to be determined as unsatisfiable fast, lowering the probability that the same itinerary will be checked more than once.

Seeking to understand how the mix of constraints checked changes when adaptive weights are used, we decided to analyze the time spent by the algorithm in different parts of the code for the 6×5 world graph grid. We measured the wall clock every time the algorithm started checking subsets of size k (see line 2 in Algorithm 1). Furthermore, we also kept count of the number of discrimination and conflation constraints checked for each sensor set aggregated over size k before it failed. The results, including a visualization of the constraint type, appear in the stepping chart in Figure 4.

Notice, first, how the optimization leads to a greater proportion of conflation constraints being checked. For our case, conflation constraints tend to fail more often when the sensor set is of high cardinality since they are likely to include row sensors. Thus, a greater proportion (or sometimes even absolutely more) of them are checked, as compared to baseline. We see that the decision, on the basis of Lemmas 5 and 6, to place lines 4–8 before lines 9–13 may be mistaken, on average.

Secondly, observe how the algorithm is able to terminate after concluding that no set of size $k = 2$ will satisfy all the discrimination constraints. The minimum satisfying sensor set in this case turned out to be 3 column sensors.

VIII. Conclusion and Future Works

This paper tackled the sensor selection problem for multiple itineraries while also allowing for considerations of privacy. We also provided strong reasoning for why merely minimizing selected sensors does not lead to satisfaction of specific privacy requirements. We formulated this problem and proved that it was worst-case intractable. Further, we provided an algorithm (based on automata-theoretic operations) to solve the problem and considered a few optimizations over the naïve implementation. In the process, we realized that the gains from those optimizations were significant owing to an inclination for wanting incorrect solutions to fail fast.

In the future, research might seek a direct reduction from the problem we proposed to canonical PSPACE-Complete problems such as QSAT. Other approaches common to solving computationally hard problems such as random algorithms, and improved heuristics may also be fruitful.

A. Acknowledgements

This material is based upon work supported in part by the National Science Foundation under grant IIS-2034097 and DoD Army Research Office under award W911NF2120064.

References

[1] Franck Cassez, Jérémy Dubreil, and Hervé Marchand. Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40:88–115, 2012.

[2] Franck Cassez and Stavros Tripakis. Fault diagnosis with static and dynamic observers. *Fundamenta Informaticae*, 88(4):497–540, 2008.

[3] Te-Shun Chou. Security threats on cloud computing vulnerabilities. *International Journal of Computer Science & Information Technology*, 5(3):79, 2013.

[4] Jorge Cortés, Geir E Dullerud, Shuo Han, Jerome Le Ny, Sayan Mitra, and George J Pappas. Differential privacy in control and network systems. In *IEEE 55th Conference on Decision and Control (CDC)*, pages 4252–4272, 2016.

[5] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 3 ed., 2006.

[6] Romain Jacob, Jean-Jacques Lesage, and Jean-Marc Faure. Overview of discrete event systems opacity: Models, validation, and quantification. *Annual reviews in control*, 41:135–146, 2016.

[7] Stéphane Lafortune, Feng Lin, and Christoforos N Hadjicostis. On the history of diagnosability and opacity in discrete event systems. *Annual Reviews in Control*, 45:257–266, 2018.

[8] Feng Lin. Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503, 2011.

[9] Tomáš Masopust and Xiang Yin. Complexity of detectability, opacity and a-diagnosability for modular discrete event systems. *Automatica*, 101:290–295, 2019.

[10] Albert R Meyer and Larry J Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *SWAT*, volume 72, pages 125–129, 1972.

[11] Jason M. O'Kane and Dylan A. Shell. Automatic Design of Discreet Discrete Filters. In *IEEE International Conference on Robots and Automation (ICRA)*, Seattle, WA, May 2015.

[12] Rishi Phatak and Dylan A Shell. Sensor selection for fine-grained behavior verification that respects privacy (extended version). *IEEE IROS - arXiv preprint arXiv:2307.13203*, 2023.

[13] A. Prorok and V. Kumar. Privacy-Preserving Vehicle Assignment for Mobility-on-Demand Systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, 2017.

[14] Hazhar Rahmani, Dylan A. Shell, and Jason M. O'Kane. Sensor selection for detecting deviations from a planned itinerary. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6511–6518, Online, 2021.

[15] Raúl Gómez Ramos, Jaime Duque Domingo, Eduardo Zalama, and Jaime Gómez-García-Bermejo. Daily human activity recognition using non-intrusive sensors. *Sensors*, 21(16):5270, 2021.

[16] Neil C. Rowe. Detecting suspicious behavior from positional information. In *Modelling Others from Observations Workshop at IJCAI. Edinburgh, Scotland*, 2005.

[17] Matthew Rueben, Cindy M Grimm, Frank J Bernieri, and William D Smart. A taxonomy of privacy constructs for privacy-sensitive robotics. *arXiv preprint arXiv:1701.00841*, 2017.

[18] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. on Auto. Control*, 40(9):1555–1575, 1995.

[19] Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[20] Md Zia Uddin, Weria Khaksar, and Jim Torresen. Ambient sensors for elderly care and independent living: a survey. *Sensors*, 18(7):2027, 2018.

[21] Rui Wang, Yingxia Wei, Houbing Song, Yu Jiang, Yong Guan, Xiaoyu Song, and Xiaojuan Li. From Offline Towards Real-Time Verification for Robot Systems. *IEEE Trans. on Industrial Informatics*, 14(4):1712–1721, 2018.

[22] Weilin Wang, Chaohui Gong, and Di Wang. Optimizing sensor activation in a language domain for fault diagnosis. *IEEE Trans. on Auto. Control*, 64(2):743–750, 2018.

[23] Xiang Yin and Stéphane Lafortune. Minimization of sensor activation in decentralized discrete-event systems. *IEEE Trans. on Auto. Control*, 63(11):3705–3718, 2017.

[24] Jingjin Yu and Steven M LaValle. Cyber detectives: Determining when robots or people misbehave. In *Algorithmic Foundations of Robotics IX*, pages 391–407. 2010.

[25] Jingjin Yu and Steven M LaValle. Story validation and approximate path inference with a sparse network of heterogeneous sensors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4980–4985, 2011.

[26] Janan Zaytoon and Stéphane Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2):308–320, 2013.