Sensor selection for detecting deviations from a planned itinerary

Hazhar Rahmani   Dylan A. Shell   Jason M. O'Kane

Abstract—**Suppose an agent asserts that it will move through an environment in some way. When the agent executes its motion, how does one verify the claim? The problem arises in a range of contexts including validating safety claims about robot behavior, applications in security and surveillance, and for both the conception and the (physical) design and logistics of scientific experiments. Given a set of feasible sensors to select from, we ask how to choose sensors optimally in order to ensure that the agent's execution does indeed fit its pre-disclosed itinerary. Our treatment is distinguished from prior work in sensor selection by two aspects: the form the itinerary takes (a regular language of transitions) and that families of sensor choices can be grouped as a single choice. Both are intimately tied together, permitting construction of a product automaton because the same physical sensors (i.e., the same choice) can appear multiple times. This paper establishes the hardness of sensor selection for itinerary validation within this treatment, and proposes an exact algorithm based on an integer linear programming (ILP) formulation that is capable of solving problem instances of moderate size. We demonstrate its efficacy on small-scale case studies, including one motivated by wildlife tracking.**

I. INTRODUCTION

Determining how agents within an environment are behaving and understanding that behavior, for instance by recognizing whether it fits some pattern, is crucial to the problem of situational awareness, which broadly encompasses agent detection and tracking, activity modeling, general sense-making, and semantically-informed surveillance. It forms an important capability for intelligent systems and is a topic of interest for the robotics community for at least three reasons. First, as information consumers: such information could enhance the ability of a robot to act within context, improving the responsiveness and appropriateness of robot actions to other events. Secondly, as information producers: we may wish to task a robot with providing raw sensor information to enable coverage and facilitate such situational awareness. The third reason is one shared of technical interest: the methods and algorithms that enable such situational awareness have substantial overlap with those used for estimation on-board robots and, historically, cross-pollination between the two has been fruitful. The present paper fits within the vein of work concerned with guarding an environment [5], though is closer to the minimalist spirit of [11] both in terms of sensors—we adopt a simple model well suited to information-impoverished sensors such as occupancy and
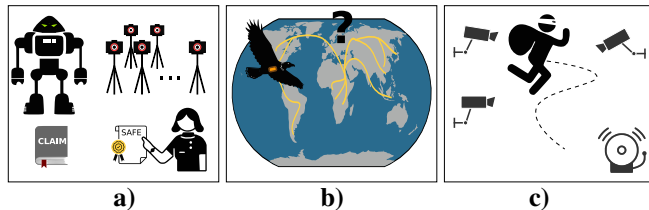

Fig. 1. Three quite distinct settings but which can be treated via the formulation described in this paper. **a)** Wishing to ensure that claims made about a particular robot's behavior will hold, an engineer chooses to instrument her laboratory's testing environment (in this case, with a set of motion tracking cameras) so she can subject it to adequate scrutiny. **b)** An ornithologist would like to test a new hypothesis concerning the migration patterns of a species of bird, so assesses the cost of acquiring tracking capabilities with adequate power of discernment. **c)** Rather than purchasing sensors to maximize coverage, if benign activities can be precisely characterized, then negating that description permits surveillance which is sufficient to identify anomalous activities but with far fewer sensors.

beam sensors—but also in the use of combinatorial filters—for fusing sequential observations and estimating state.

Our work was inspired by Yu and LaValle [14], who consider the question of validating a story: given a polygonal environment, a claimant provides a sequence of locations which they assert to have visited and the system is tasked with determining whether a given sequence of sensor readings is consistent with that claim. That is, does the sensor history contain any evidence that the given sequence of locations was in fact not visited? First in [14], and then, with several refinements and under weaker assumptions, in the follow-up [15], Yu and LaValle provide an efficient method for this problem. Their approach is sound and complete in that it identifies inconsistencies between the story and the sensed history if and only if such inconsistencies exist. However, the strength of the validation (or, more correctly, the method's *in*ability to *in*validate) must be understood modulo sensor data. The faculty to detect contradictions depends critically on sensor history, on the evidence that the sensors provide. The more limited the sensing, the fewer fibs you can catch.

A natural concern, then, is how to choose sensors. Suppose that the given story describes a pre-declared itinerary, a future path or structured collection of possible paths through an environment. Now, given a set of possible sensors one could deploy, when some path has been executed, which ones suffice to detect deviations from the itinerary? The present paper considers an optimization variant where we ask for a minimal set of sensors that can accomplish this. Fig. 1 illustrates the breadth of use cases for this scenario, in which the goal is to ensure detection of all deviations. It is important to note that with a given environment, given itinerary, and set of sensors, this may be impossible—the whole set of sensors
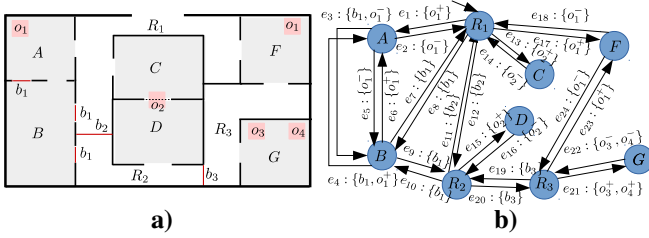
Fig. 2. **a)** An example of an environment, which shows the floor map of a department. This environment is guarded by beam sensors b_1, b_2, and b_3, and by occupancy sensors o_1, o_2, o_3, and o_4. **b)** A multigraph representing the environment. Each edge e is labeled by a world-observation, a set of events received when an agent moves from the region represented by the source vertex of e to the region represented by the target vertex of e.

may be inadequate. A concrete sort of application, based on selection of a suite of beam and occupancy sensors in an indoor environment, appears in Fig. 2.

This paper starts by formalizing the problem of optimal sensor selection to detect deviations from a disclosed itinerary (Section III). As itineraries are claims about the future, it seems useful to permit rather more flexibility than the stories of Yu and LaValle allow, and so our treatment does. Further, motivated by the physical placement of sensors which effectively guard multiple areas at once, it models circumstances where multiple sensors may be obtained together as a single logical unit. We then establish the computational hardness of the problem (in Section V), and show how it may be treated via integer linear programming (in Section VI). Experimental results in Section VII show that realistic size instances can be practically solved in this way.

II. RELATED WORK

The problem of reducing the sensor readings needed to establish some property has been the subject of extensive study in the discrete event systems literature (see the survey by Sears and Rudie [8]). That literature distinguishes sensor selection (e.g., [2]) from sensor activation (e.g., [1], [12]). The former considers, as studied in the present paper, a one-shot decision at initialization of whether to adopt a sensor or not; the latter is an online variant that switches sensors on/off across time. This paper aims to fill the niche between such sensor-oriented work and the problem of story validation (as exemplified by Yu and LaValle [14], [15]).

The NP-completeness of minimal sensor selection for the properties of observability and diagnosability [7] was established by Yoo and Lafortune [10]. Recently, Yin and Lafortune [13] proposed a general approach to optimizing sensor selection that applies to a very wide set of problems and subsumes several previous methods. Their approach is capable of enforcing what they term 'information state-based properties,' essentially arbitrary predicates defined on states, which allow one treat a variety of fault detection and diagnosis tasks (including observability and diagnosability). Our itinerary validation problem, however, doesn't fall within this class as it is not enough to ask whether a state is visited or not; specific transitions between states matter too.

The characteristic that differentiates itinerary validation is that the basic mathematical objects under consideration

are trajectories. This emphasizes an important distinction between language- and automata-based properties (cf. our [6]), with fairly subtle implications for our model and approach. Itineraries will be taken to describe sequences of edges and in order to relate the world's structure to those sequences, we will form a product. That product may partially unroll or unfold the world, so that the choice of a sensor can affect elements less 'locally' than one might naturally expect. One implication for the model is that we directly treat situations where multiple sensors may be selected as a single logical unit as this occurs in the product anyway. Thence, the further implication is that our computational complexity (hardness) result utilizes a cover selection problem directly.

III. DEFINITIONS AND PROBLEM STATEMENT

This section formalizes our sensor selection problem.

A. Modeling the environment

In our approach, the environment is modeled using a discrete structure, called a world graph, defined as follows.

Definition 1: *[World graph] A* world graph *is an edge-labeled directed multigraph* $\mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda)$ *in which*

- V *is a nonempty 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, which identify the source vertex and target vertex of 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, and*
- $\lambda : E \to \wp(Y_{s_1} \cup Y_{s_2} \cup \cdots \cup Y_{s_k})$ *is a labeling function, which assigns to each edge, a* world-observation— *a set of events.*

(Here $\wp(X)$ denotes the set of all subsets of X.)

The intuition is that a world graph describes an environment through which agent might move, along with the available sensors in the environment and also the sensor events that agent may trigger along its motion, provided those sensors are active. Vertices in the graph represent regions within the environment of interest. Edges represent feasible transitions between regions, each labeled with a set of sensor events that happen simultaneously when the system makes the transition corresponding to that edge.

In this model, for each sensor $s_i \in S$, Y_{s_i} is the set of all events produced by s_i. Notice that Definition 1 stipulates that distinct sensors produce disjoint events, that is, for each $s_i, s_j \in S$, if $s_i \neq s_j$, then $Y_{s_i} \cap Y_{s_j} = \varnothing$. The labeling function λ is used to indicate for each edge, the set of events produced by the sensors when the agent makes a transition corresponding to that edge in the environment.

B. Example: Beam and occupancy sensors

Though the technical results to follow apply for any world graph that satisfies Definition 1, to keep the description reasonably concrete, we present examples focused upon

occupancy sensors (which detect the presence of the agent in a region) and beam sensors (which detect the passage of an agent between adjacent regions).

To illustrate, consider the simple environment in Fig. 2a, which is guarded by four occupancy sensors o_1, o_2, o_3, and o_4 and three beam sensors b_1, b_2, and b_3. Fig. 2b shows the world graph corresponding to this environment, as constructed via the algorithm of Yu and LaValle [14]. Each state of this graph represents a room or a region within the environment guarded by the same set of sensors. Each edge shows a transition between two neighboring regions.

Notice that, in the example, some pairs of rooms are connected by multiple doors, each of which are guarded by different sensors. This explains why Definition 1 uses a multigraph structure for world graphs. Also note that in this environment, an agent cannot directly move between rooms C and D because those two rooms are separated by a window rather than a door.

In this example, an occupancy sensor o, which detects the presence of an agent in a region X, is activated when the agent enters X and is deactivated once the agent exits X. Accordingly, each occupancy sensor o in this example produces two events, o^+, which occurs when o is activated, and o^-, which happens when o is deactivated. A beam sensor is activated when a mobile agent physically crosses (or breaks) the beam and then instantly deactivated. Because the agent is mobile and a beam sensor is deactivated immediately after it was activated, we model each beam sensor b with a single event b in Y_b. When beam b is broken, the system knows that the physical line segment was crossed, but not the direction of crossing. For the environment in Fig. 2: for each of the occupancy sensors o_i, $i \in \{1,2,3,4\}$, $Y_{o_i} = \{o_i^+, o_i^-\}$; for each of the beam sensors b_j, $j \in \{1,2,3\}$, $Y_{b_j} = \{b_j\}$.

When an agent makes a transition between two regions, it is possible that several events happen simultaneously, and in fact, the system observes all those events at the same time. For example, when an agent makes a transition from room A to room B from the left door, two events o_1^- and b_1 happen simultaneously, and thus, edge e_3 in the world graph is labeled with the world-observation $\{b_1, o_1^-\}$.

Finally, notice that a world graph can readily represent scenarios in which a single sensor guards multiple transitions. In the example, B has four doors, three of which are guarded. The beams that guard those doors are assumed to be a single beam sensor $b_1 \in S$. When an agent crosses any of those beams, the system knows that one of them is crossed but it does not know which one it was. Likewise, rooms C and D are guarded by a single occupancy sensor o_2, which is located on the window between those two rooms. Thus, if an agent enters any of those two rooms, o_2 is activated, but by observing o_2^+, the system cannot tell if the agent entered room C or room D.

C. Itinerary DFA

In our story validation problem, an agent takes a tour in the environment along a continuous path. This path is represented over the world graph $\mathcal{G}$ by a *walk*, which is

defined as a finite sequence of edges $e_1 e_2 \cdots e_n \in E^*$ in which $\mathrm{src}(e_1) = v_0$ and for each $i \in \{1, 2, \ldots, n-1\}$, $\mathrm{tgt}(e_i) = \mathrm{src}(e_{i+1})$. The set of all walks over $\mathcal{G}$ —that is, the set of all not-necessarily-simple paths one can take in the environment— is denoted $\mathrm{Walks}(\mathcal{G})$.

The agent claims that its tour will be one of those words specified by a deterministic finite automaton (DFA):

Definition 2: *[Itinerary DFA] An* itinerary DFA *over a world graph* $\mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda)$ *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 all accepting (final) states.*

For each finite word $r = e_1 e_2 \cdots e_n \in E^*$, there is a unique sequence of states $q_0 q_1 \cdots q_n$ for which q_0 is the initial state and for each $i \in \{0, 1, \ldots, n-1\}$, $\delta(q_i, e_i) = q_{i+1}$. Word r is *accepted* by the DFA if $q_n \in F$. The language of $\mathcal{I}$, denoted $L(\mathcal{I})$, is the set of all finite words accepted by $\mathcal{I}$, i.e., $L(\mathcal{I}) = \{r \in E^* \mid r \text{ is accepted by } \mathcal{I}\}$. The robot claims its tour will be one of the words $r \in L(\mathcal{I})$. Note that each word accepted by this DFA is a walk over the world graph, and accordingly, the robot's itinerary not only specifies the sequence of locations the agent visits but it also identifies the specific transitions (i.e. the doors between the rooms) through which the agent moves.

D. Itinerary validation

We seek to enable a minimal subset $M \subseteq S$ of sensors such that, when the agent finishes its tour within the environment, the system can determine with full certainty whether the agent followed its itinerary or not. At the completion of the agent's tour, the system does not know the exact tour the agent took in the environment, instead receiving only a sequence of world-observations. Each item in the sequence is generated by the system when a set of sensors were activated or deactivated simultaneously as a result of the agent's moving in the environment.

Let us mildly abuse notation and use Y_M to denote the set of all events produced by sensors in some $M \subseteq S$, i.e., $Y_M = \bigcup_{s \in M} Y_s$. If, from all sensors S, only the sensors in M are turned on, then when the agent transitions across e in $\mathcal{G}$, the system receives world-observation $\lambda(e) \cap Y_M$. That is, when transitioning across an edge, the system observes precisely those events that are both associated with that edge and enabled by one of set M's selected sensors. Where $\lambda(e) \cap Y_M = \varnothing$, this must be handled slightly differently: in this case, the system produces no symbol at all (not a symbol reporting that some un-sensed event occurred—which would itself be a tacit sort of information). We make this precise next.

The agent's walk over the world graph generates a sequence of non-empty world-observations. For a world graph $\mathcal{G} = (V, E, \mathrm{src}, \mathrm{tgt}, v_0, S, \mathbb{Y}, \lambda)$, we define function $\beta_{\mathcal{G}} : \mathrm{Walks}(\mathcal{G}) \times \wp(S) \to (\wp(Y_S) \setminus \varnothing)^*$ in which for each $r \in \mathrm{Walks}(\mathcal{G})$ and subset of sensors $M \subseteq S$, $\beta_{\mathcal{G}}(r, M)$ gives the sequence of world-observations the system receives when the agent takes walk r and precisely the sensors in M are

turned on. Formally, for each $r = e_1 e_2 \cdots e_n \in \text{Walks}(\mathcal{G})$, $\beta_{\mathcal{G}}(r, M) = z_1 z_2 \cdots z_n$ in which for each $i \in \{1, \ldots, n\}$, $z_i = \lambda(e_i) \cap Y_M$ if $(\lambda(e_i) \cap Y_M) \neq \varnothing$, and $z_i = \epsilon$ otherwise, where ϵ is the (standard) *empty symbol*.

Based on this function, we make a definition that formulates conditions under which a set of sensors are able to tell whether the agent adhered to its claimed itinerary or not.

Definition 3: *[Certifying Sensor Selection] Let $M \subseteq S$ be a subset of sensors. We say M certifies itinerary $\mathcal{I}$ on world graph $\mathcal{G}$ if there exist no $r \in L(\mathcal{I}) \cap \text{Walks}(\mathcal{G})$ and $t \in \text{Walks}(\mathcal{G}) \setminus L(\mathcal{I})$ such that $\beta_{\mathcal{G}}(r, M) = \beta_{\mathcal{G}}(t, M)$.*

Intuitively, if M is a certifying sensor selection for $\mathcal{I}$, then based on the sequence of world-observations $\beta_{\mathcal{G}}(r, M)$ the system perceives from the environment, the system can tell whether r was within the claimed itinerary or not, that is, whether $r \in L(\mathcal{I})$ or not. In fact, if for each $r \in L(\mathcal{I}) \cap \text{Walks}(\mathcal{G})$, there is no $t \in (\text{Walks}(\mathcal{G}) \setminus L(\mathcal{I}))$ such that $\beta_{\mathcal{G}}(t, M) = \beta_{\mathcal{G}}(r, M)$, then the system can tell with full certainty if r was within the claimed itinerary or not. Thus, the system must choose a sensor set M to turn on that is certifying for $\mathcal{I}$ on $\mathcal{G}$.

We formalize our minimization problem as follows.

Problem: Minimal sensor selection to validate an itinerary (MSSVI)

Input: A world graph $\mathcal{G} = (V, E, \text{src}, \text{tgt}, v_0, S, \mathbb{Y}, \lambda)$ and an itinerary DFA $\mathcal{I} = (Q, V, \delta, q, F)$.

Output: A minimum size certifying sensor selection $M \subseteq S$ for $\mathcal{I}$ on $\mathcal{G}$, or 'INFEASIBLE' if no such certifying sensor selection exists.

Before showing this problem to be NP-hard, we describe a construction that turns out to be useful in what follows.

IV. World graph-itinerary product automata

In this section, we describe how to use the inputs of the MSSVI problem to construct a product automaton that captures the interactions between a world graph and an itinerary DFA. We use this construction for both a hardness result about MSSVI (in Section V) and a practical solution of MSSVI via integer linear programming (in Section VI). This product automaton is defined as follows.

Definition 4: *[Product automaton] Let $\mathcal{G} = (V, E, \text{src}, \text{tgt}, v_0, S, \mathbb{Y}, \lambda)$ be a world graph and $\mathcal{I} = (Q, E, \delta, q_0, F)$ be an itinerary DFA. The product automaton $\mathcal{P}_{\mathcal{G}, \mathcal{I}}$ 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*

- $Q_{\mathcal{P}} = Q \times V$,
- $\delta_{\mathcal{P}} : Q_{\mathcal{P}} \times E \nrightarrow Q_{\mathcal{P}}$ *is a partial function such that for each $(q, v) \in Q_{\mathcal{P}}$ and $e \in E$, $\delta_{\mathcal{P}}((q, v), e)$ is undefined if $\text{src}(e) \neq v$, otherwise, $\delta_{\mathcal{P}}((q, v), e) = (\delta(q, e), \text{tgt}(e))$,*
- $q_0^{\mathcal{P}} = (q_0, v_0)$, *and*
- $F_{\mathcal{P}} = F \times V$.

Note that the transition function of this DFA is partial. We will write $\delta_{\mathcal{P}}(p, e) = \bot$ to mean that $\delta_{\mathcal{P}}$ is undefined for (p, e). The extended transition function $\delta_{\mathcal{P}}^* : Q_{\mathcal{P}} \times E^* \nrightarrow$

$Q_{\mathcal{P}}$—which for each $q \in Q_{\mathcal{P}}$ and $r \in E^*$, $\delta_{\mathcal{P}}^*(q, r)$ denotes the state to which the DFA reaches by tracing r from state q—is also partial. For a word $r = e_1 e_2 \cdots e_n \in E^*$, we use $\delta_{\mathcal{P}}^*(q_0^{\mathcal{P}}, r) = \bot$ to mean that this DFA crashes when it traces r from the initial state, that is, there is a unique state sequence $q_0 q_1 \cdots q_k$ for some $k < n$ such that $\delta_{\mathcal{P}}(q_{i-1}, e_i) = q_i$ for all $i \in \{1, 2, \ldots, k-1\}$ but $\delta_{\mathcal{P}}(q_k, e_k) = \bot$. A word $r \in E^*$ is *trackable* by this DFA if $\delta_{\mathcal{P}}^*(q_0^{\mathcal{P}}, r) \neq \bot$.

Our purpose in constructing this product automaton is revealed by the following result.

Lemma 1: Let $\mathcal{G}$, $\mathcal{I}$, and $\mathcal{P}_{\mathcal{G}, \mathcal{I}}$ be the structures in Definition 4. A subset $M \subseteq S$ of sensors is a certifying sensor selection for $\mathcal{I}$ if and only if for each $r, r' \in E^*$ such that $\delta_{\mathcal{P}}^*(q_0^{\mathcal{P}}, r) \in F_{\mathcal{P}}$ and $\delta_{\mathcal{P}}^*(q_0^{\mathcal{P}}, r') \in Q_{\mathcal{P}} \setminus F_{\mathcal{P}}$, it holds that $\beta_{\mathcal{G}}(r, M) \neq \beta_{\mathcal{G}}(r', M)$.

Proof: The construction yields two direct observations:
(1) Every word trackable by $\mathcal{P}$ is a walk over $\mathcal{G}$ and vice versa, i.e., $\{r \in E^* \mid \delta_{\mathcal{P}}^*(q_0^{\mathcal{P}}, r) \neq \bot\} = \text{Walks}(\mathcal{G})$.
(2) The accepting states of $\mathcal{P}$ correspond to the accepting states of the itinerary DFA, so $L(\mathcal{P}_{\mathcal{G}, \mathcal{I}}) = L(\mathcal{I})$.
Taken together, (1) and (2) imply that $\text{Walks}(\mathcal{G}) \setminus L(\mathcal{I}) = \{r \in E^* \mid \delta_{\mathcal{P}}^*(q_0^{\mathcal{P}}, r) \in Q_{\mathcal{P}} \setminus F_{\mathcal{P}}\}$. This means that each walk over the world graph that is not within the language of the itinerary DFA, reaches a non-accepting state in this DFA. But, because every word in the the language of the itinerary DFA reaches to an accepting state in this DFA, if there are two words r and r' such that r reaches to an accepting state, r' reaches to a non-accepting state, and r and r' both yield the same sequence of world-observations by $\beta_{\mathcal{G}}$ under M, i.e., $\beta_{\mathcal{G}}(r, M) = \beta_{\mathcal{G}}(r', M)$, then M is not a certifying sensor selection for $\mathcal{I}$. Contrariwise, if there are no such words r and r', then M is a certifying sensor selection. This completes the proof. ∎

As a result, given a sensor selection M as a feasible solution to MSSVI with inputs $\mathcal{G}$ and $\mathcal{I}$, one can use the product automaton $\mathcal{P}_{\mathcal{G}, \mathcal{I}}$ to check if M is a certifying sensor selection for $\mathcal{I}$ or not by testing whether such r and r' described in the proof of this lemma can be found or not. The next section makes this idea clear.

V. Hardness of MSSVI

Next, we present a hardness result for minimal sensor selection, starting by casting MSSVI as a decision problem.

Decision Problem: Minimal sensor selection to validate an itinerary (MSSVI-DEC)

Input: A world graph $\mathcal{G} = (V, E, \text{src}, \text{tgt}, v_0, S, \mathbb{Y}, \lambda)$, an itinerary DFA $\mathcal{I} = (Q, V, \delta, q_0, F)$, and integer k.

Output: Yes if there is a certifying sensor selection $M \subseteq S$ such that $|M| \leq k$; No otherwise.

We prove that MSSVI-DEC is NP-complete, by showing that it is both in NP and NP-hard. First, we show that MSSVI-DEC can be verified in polynomial time.

Lemma 2: MSSVI-DEC $\in$ NP.

Proof: We need to show that, using a given sensor selection $M \subseteq S$ as a certificate, we can verify in polynomial

time both (1) whether $|M| \leq k$ and (2) whether M is a certifying sensor selection in the sense of Definition 3 or not. As (1) is trivially verifiable, we turn to (2).

Recall that by Lemma 1, if for any words $r, r' \in E^*$ for which $\delta_{\mathcal{P}}^*(v_0, r) \in F_{\mathcal{P}}$ and $\delta_{\mathcal{P}}^*(v_0, r') \in Q_{\mathcal{P}} \setminus F_{\mathcal{P}}$, it holds that $\beta_{\mathcal{G}}(r, M) \neq \beta_{\mathcal{G}}(r', M)$, then M is a certifying sensor selection for the given itinerary DFA $\mathcal{I}$, otherwise M is not a certifying sensor selection for $\mathcal{I}$. Thus, to check if M is certifying, we compute, using a fixed point algorithm described presently, a relation R that relates all pairs of states q and p that are reachable from the initial state by two words (walks) that yield the same sequence of world-observations by $\beta_{\mathcal{G}}$ under M. Then we check whether R relates any pairs of states q and p such that one of them is accepting while the other is non-accepting. If R relates such a pair, then M is not a certifying sensor selection for $\mathcal{I}$. Otherwise, M is certifying for $\mathcal{I}$. To construct R, begin with R initially assigned to $\{(q_0^{\mathcal{P}}, q_0^{\mathcal{P}})\}$. Then iteratively update R according to the following equation until the iteration reaches a fixed point, with no additional tuples added to R.

$$R \leftarrow R \cup \bigcup_{(q,p) \in R} \begin{pmatrix} \bigcup_{\substack{e,d \in E: \\ (\lambda(e) \cap Y_M) = \\ (\lambda(d) \cap Y_M) \\ \text{and } \delta_{\mathcal{P}}(q,e) \neq \bot \\ \text{and } \delta_{\mathcal{P}}(p,d) \neq \bot}} (\delta_{\mathcal{P}}(q,e), \delta_{\mathcal{P}}(p,d)) \end{pmatrix}$$
$$\cup \bigcup_{(q,p) \in R} \begin{pmatrix} \bigcup_{\substack{e \in E: \\ (\lambda(e) \cap Y_M) = \epsilon \\ \text{and } \delta_{\mathcal{P}}(q,e) \neq \bot}} (\delta_{\mathcal{P}}(q,e), p) \end{pmatrix}.$$

To expand R, this equation uses two rules, one in the first line and the other in the second line. Fig. 3 illustrates those two rules. The first rule states that if $(q, p) \in R$, then for any edges $e, d \in E$ such that q has an outgoing transition for e and p has an outgoing transition for d, if e and d yield the same world-observation under M, then we must add $(\delta_{\mathcal{P}}(q), \delta_{\mathcal{P}}(p))$ to R as well, which means that states $\delta_{\mathcal{P}}(q, e)$ and $\delta_{\mathcal{P}}(p, d)$ are reachable from the initial state of $\mathcal{P}_{\mathcal{G},\mathcal{I}}$ by two words (walks) that yield the same world observation by $\beta_{\mathcal{G}}$ under M. The second rule enforces that if $(q, p) \in R$, then for any edge $e \in E$ such that q has an outgoing transition for e, if e yields the empty world-observation under M (that is, if e yields ϵ by $\beta_{\mathcal{G}}$ under M), then $(\delta_{\mathcal{P}}(q, e), p)$ must be added to R too. This is because here both states $\delta_{\mathcal{P}}(q, e)$ and p are reachable from the initial state, respectively, by a pair of words (walks) r and r' that, respectively, reached q and p, while yielding a single sequence of world-observations. Using an appropriate implementation, this algorithm takes time which is polynomial in the size of $\mathcal{P}$ because there are $\mathcal{O}(|Q_{\mathcal{P}}|^2)$ pairs in R and thus $\mathcal{O}(|Q_{\mathcal{P}}|^2)$ stages of updates, each checking at most $|E|$ edges. This shows that MSSVI-DEC $\in$ NP. ∎

The practical import of this lemma is that, in polynomial time, we can decide whether a sensor set is certifying for a given itinerary or not.

Next, we prove that MSSVI-DEC is computationally hard. To do so, we shall reduce from a well-known problem.
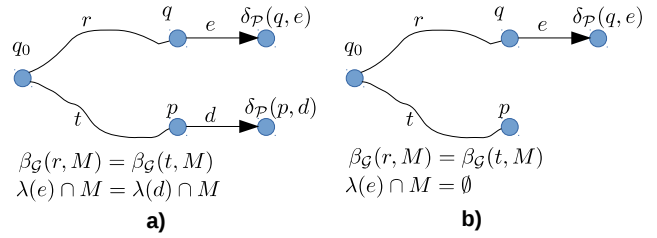

Fig. 3. Two cases in the construction of the relation R in the proof of Lemma 2. In each case, R is expanded, given that states q and p are already related by R. **a)** In this case, pair $\delta_{\mathcal{P}}(q, e)$ and $\delta_{\mathcal{P}}(p, d)$ are added to R. **b)** In this case, pair $\delta_{\mathcal{P}}(q, e)$ and p are added to R.

Decision Problem: Set cover (SETCOVER-DEC)

Input: A finite set U, called the universe, a collection of subsets $\mathbf{O} = \{O_1, O_2, \ldots, O_m\}$ in which, for each $i \in \{1, 2, \ldots, m\}$, $O_i \subseteq U$ and $\bigcup_{i \in \{1,2,\ldots,m\}} O_i = U$, and an integer k.

Output: Yes if there is a sub-collection $\mathbf{N} \subseteq \mathbf{O}$ such that $\bigcup_{O \in \mathbf{N}} O = U$ and $|\mathbf{N}| \leq k$, and No otherwise.

This problem is known to be NP-complete [3]. We reduce from SETCOVER-DEC to prove the following result.

Theorem 1: MSSVI-DEC *is* NP-*hard*.

Proof: We prove this result by a polynomial reduction from SETCOVER-DEC to MSSVI-DEC. Given a SETCOVER-DEC instance

$$x = \langle U = \{u_1, u_2, \ldots, u_n\}, \mathbf{O} = \{O_1, O_2, \ldots, O_m\}, k \rangle,$$

construct an MSSVI-DEC instance

$$f(x) = \langle \mathcal{G} = (V, E, \text{src}, \text{tgt}, v_0, S, \mathbb{Y}, \lambda), \mathcal{I}, k' \rangle,$$

as illustrated in Fig. 4 and detailed below.

– For the vertices of the world graph, create $2n + 2$ states, denoted $V = \{C_0, C_1, \ldots, C_{n+1}\} \cup \{u_1, u_2, \ldots, u_n\}$. Note that the u_i elements correspond directly to the elements of the universe in the SETCOVER-DEC instance. The idea is that the u_i's represent rooms arranged in sequence, each accessible from a shared corridor composed of the C_i's.

– For the edges of the world graph, create $4n + 2$ edges, denoted $E = \{e_0, e_1, \ldots, e_n\} \cup \{e_0', e_1', \ldots, e_n'\} \cup \{d_1, d_2, \ldots, d_n\} \cup \{d_1', d_2', \ldots, d_n'\}$. The src and tgt functions are defined so that each e_i connects C_i to C_{i+1}, each e_i' connects C_{i+1} to C_i, each d_i connects C_i to u_i, and each d_i' connects u_i to C_i.

– Create a set of $n+m+1$ sensors, $S = \{b_1, b_2, \ldots, b_{n+1}\} \cup \{O_1, O_2, \ldots, O_m\}$, in which the b_i's are beam sensors and the O_i's are occupancy sensors. The event set corresponds to these sensors in the usual way, with one event for each beam sensor and two events for each occupancy sensor, so for each $j \in \{1, 2, \ldots, n + 1\}$, $Y_{b_j} = \{b_j\}$, and for each $i \in \{1, 2, \ldots, m\}$, $Y_{O_i} = \{O_i^+, O_i^-\}$, and then, $\mathbb{Y} = \{Y_{b_0}, Y_{b_1}, \ldots, Y_{d_{n+1}}, Y_{O_1}, Y_{O_2}, \ldots, Y_{O_m}\}$.

– For the events labeling each edge, define $\lambda(e_i) = \lambda(e_i') = b_{i+1}$ for the e_i and e_i' edges, $\lambda(d_i) = \{O_j^+ \mid u_i \in O_j\}$ for the d_i edges, $\lambda(d_i') = \{O_j^- \mid u_i \in O_j\}$ for the d_i' edges. This models one or more occupancy sensors in
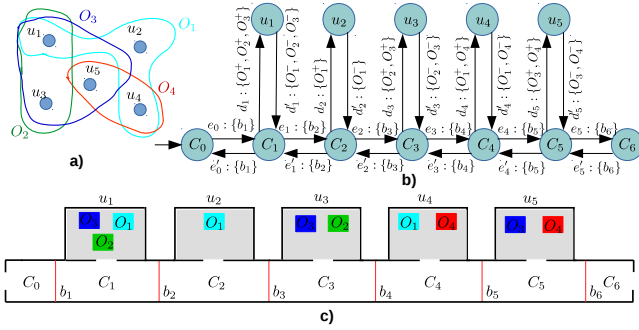
Fig. 4. **a)** An instance of the set cover problem. **b)** The instance of the MSSVI problem for this set cover instance. **c)** A physical environment representing the MSSVI instance.

each of the u_i rooms, according to the subsets within the SETCOVER-DEC instance, and beam sensors along the corridor between each room.

– For the itinerary DFA $\mathcal{I}$, construct a DFA accepting the singleton language $L(\mathcal{I}) = \{e_0 d_1 d_1' e_1 d_2 d_2' e_2 \ldots d_n d_n' e_n\}$ as a linear chain of states.

– For the bound on the number of sensors allowed, choose $k' = k + n + 1$.

In the MSSVI-DEC instance constructed in this way, notice that, for each subset $O \in \mathbf{O}$, the construction makes a corresponding occupancy sensor O and puts that sensor in all rooms u for which $u \in O$. Moreover, the itinerary specifies a single walk $e_0 d_1 d_1' e_1 d_2 d_2' e_2 \cdots d_n d_n' e_n$, which indicates that the sequence of regions the agent intends to visit is $C_0 C_1 u_1 C_1 C_2 u_2 C_2 \cdots C_{n+1}$. That is, the itinerary calls for the agent to travel down the corridor, visiting each room exactly once in the specific order $u_1, u_2, \ldots, u_n$, without backtracking within the corridor. For the system to be able to tell that the agent has visited a room, at least one occupancy sensor in each room must be turned on. Also, each of the beam sensors must be turned on so that the system can assure that the agent did not vacillate back-and-forth between cells in the corridor.

The construction clearly takes polynomial time, so it remains only to show that the reduction is correct, i.e. that the original SETCOVER-DEC instance has a set cover of size at most k if and only if the constructed MSSVI-DEC has a valid sensor set of size k'.

($\Rightarrow$) Suppose there exists a set cover $\mathbf{N} \subseteq \mathbf{O}$ such that $|\mathbf{N}| \leq k$ and $\sum_{O \in \mathbf{N}} O = U$. In this case, based on our discussion, the sensor selection $M = \mathbf{N} \cup \{b_1, b_2, \ldots, b_{n+1}\}$ is for itinerary $\mathcal{I}$, a certifying sensor selection of size $|M| = |\mathbf{N}| + n + 1 \leq k + n + 1 = k'$.

($\Leftarrow$) Conversely, suppose there exists for $\mathcal{I}$, a certifying sensor selection $M \subseteq S$ for which $|M| \leq k'$. As argued above, because M is certifying, it must contain each of the $n+1$ beam sensors. Thus, there are at most $k' - (n+1) = k$ occupancy sensors in M. Recall, however, that for this construction, every certifying sensor selection includes at least one occupancy sensor within each room. Thus, the occupancy sensors in M form a set cover of size at most k for the original SETCOVER-DEC instance. ∎

Finally, the following results follow immediately from Theorem 1 and Lemma 2.

Theorem 2: MSSVI-DEC *is* NP-*complete.*

Corollary 1: MSSVI *is* NP-*hard.*

As a result, assuming $P \neq NP$, one cannot find a certifying sensor selection with minimum size in polynomial time.

VI. MSSVI via Integer Linear Programming

In this section, we present an exact solution to MSSVI using an Integer Linear Programming formulation of the problem. First, we cast the MSSVI problem into mathematical programming form, and then linearize its constraints.

A. Mathematical programming formulation of MSSVI

For each sensor $s \in S$, we introduce a binary variable u_s, which receives value 1 if sensor s is chosen to be turned on, and receives 0 otherwise. For each tuple of states $(q, p) \in Q_{\mathcal{P}} \times Q_{\mathcal{P}}$, we introduce a binary variable $a_{q,p}$, which receives value 1 if and only if there exist two finite words $r, r' \in E^*$, such that $\delta_{\mathcal{P}}^*(q_0^{\mathcal{P}}, r) = q$, $\delta_{\mathcal{P}}^*(q_0^{\mathcal{P}}, r') = p$, and and $\beta_{\mathcal{G}}(r, M) = \beta_{\mathcal{G}}(r', M)$. For each edge $e \in E$ and event $e \in Y$, we introduce a binary variable $b_{e,y}$ which is assigned a value 1 if and only if the label of e contains y and the sensor that produces y is chosen to be turned on. More precisely, if $y \in \lambda(e)$ and $u_{\eta(y)} = 1$, then $b_{e,y}$ receives value 1, otherwise it receives value 0, where $\eta(y)$ is the sensor that produces event y, i.e., $\eta(y) = s$ such that $y \in Y_s$. In terms of these variables, an MSSVI instance can be expressed as follows.

Minimize:
$$\sum_{s \in S} u_s \tag{1}$$

Subject to:
$$a_{q_0^{\mathcal{P}}, q_0^{\mathcal{P}}} = 1 \tag{2}$$

For each $q \in F_{\mathcal{P}}, p \in Q_{\mathcal{P}} \setminus F_{\mathcal{P}}$,
$$a_{q,p} = 0 \tag{3}$$

For each $q \in Q_{\mathcal{P}} \setminus F_{\mathcal{P}}, p \in F_{\mathcal{P}}$,
$$a_{q,p} = 0 \tag{4}$$

For each $e \in E$ and $y \in Y$ s.t. $y \notin \lambda(e)$,
$$b_{e,y} = 0 \tag{5}$$

For each $e \in E$ and $y \in Y$ s.t. $y \in \lambda(e)$,
$$b_{e,y} = u_{\eta(y)} \tag{6}$$

For each $q, p \in Q_{\mathcal{P}}$ and $e, d \in E$ such that $\delta_{\mathcal{P}}(q, e) \neq \perp$ and $\delta_{\mathcal{P}}(p, d) \neq \perp$,
$$a_{q,p} = 1 \wedge (\forall y \in Y, b_{e,y} = b_{d,y}) \Rightarrow a_{\delta_{\mathcal{P}}(q,e), \delta_{\mathcal{P}}(p,d)} = 1 \tag{7}$$

For each $q, p \in Q_{\mathcal{P}}$ and $e \in E$ such that $\delta_{\mathcal{P}}(q, e) \neq \perp$,
$$a_{q,p} = 1 \text{ and } (\forall y \in Y, b_{e,y} = 0) \Rightarrow a_{\delta_{\mathcal{P}}(q,e), p} = 1 \tag{8}$$

For each $q, p \in Q_{\mathcal{P}}$ and $e \in E$ such that $\delta_{\mathcal{P}}(p, e) \neq \perp$,
$$a_{q,p} = 1 \text{ and } (\forall y \in Y, b_{e,y} = 0) \Rightarrow a_{q, \delta_{\mathcal{P}}(p,e)} = 1 \tag{9}$$

The objective (1) is to minimize the number of sensors turned on. Constraint (2) asserts that there exists a world-

	Itinerary	Description of itinerary	Computed solution	Comp. time (sec)
1	$Walks(\mathcal{G})$	All location sequences	$\varnothing$	2.7
2	$E^* \setminus Walks(\mathcal{G})$	No location sequence	$\varnothing$	2.3
3	$Walks(\mathcal{G}) \setminus \{r \in E^* \mid e_{21} \notin r\}$	Do not enter G	$\{o_3\}$	2.8
4	$e_1 e_5 e_9 e_{15}$	$R_1 A B R_2 D$	INFEASIBLE	2.5
5	$e_1 e_5 e_9 e_{15} + e_1 e_5 e_7 e_{13}$	$R_1 A B R_2 D$ or $R_1 A B R_1 C$	$\{o_1, b_1, b_2, o_2, b_3\}$	2.9
6	$e_{12}(e_{11}e_{12} + e_{20}e_{19})^* e_{20} e_{21}$	$R_1 R_2 (R_1 R_2 + R_3 R_2)^* R_3 G$	$\{o_1, o_2, o_3, b_1\}$	4.1

observation sequence, (the empty string, ϵ) by which both states of the tuple $(q_0^{\mathcal{P}}, q_0^{\mathcal{P}})$ are reachable from the initial state. Constraint Sets (3) and (4) ensure that the sensor selection is certifying. Constraint Sets (5) and (6) encode which sensors affect the label of each edge. Constraint Set (7) asserts that if two states q and p are both reachable by a world-observation sequence, then for any edges e and d, if those two edges receive the same world-observation under the chosen sensors, then it means states $\delta_{\mathcal{P}}(q,e)$ and $\delta_{\mathcal{P}}(p,d)$ are also reachable by at least one sequence of world-observations under the chosen sensors. In fact, these constraints implement the case shown in Fig. 3a. Similarly, Constraint Sets (8) and (9) implement the case in Fig. 3b.

This formulation would be a 0–1 integer linear programming model if Constraint Sets (7), (8), and (9) were linear. Thus, the next section shows how to linearize them.

B. Integer linear programming formulation of MSSVI

To linearize Constraint Set (7), first we introduce a binary variable $j_{e,d,y}$ for each $e, d \in E$ and $y \in Y$, which receives its value from the following constraints.

For all $e, d \in E$ and all $y \in Y$,

$$b_{e,y} - b_{d,y} \leq j_{e,d,y}, \quad (10)$$
$$b_{d,y} - b_{e,y} \leq j_{e,d,y}, \quad (11)$$
$$j_{e,d,y} \leq b_{e,y} + b_{d,y}, \text{ and} \quad (12)$$
$$j_{e,d,y} \leq 2 - b_{e,y} - b_{d,y}. \quad (13)$$

These linear constraints assign value 0 to $j_{e,d,y}$ if $b_{e,y} = b_{d,y}$; otherwise, they assign value 1 to $j_{e,d,y}$. Hence, Constraint Set (7) is replaced by the following linear constraints.

For each $q, p \in Q_{\mathcal{P}}$ and $e, d \in E$ s.t. $\delta_{\mathcal{P}}(q,e) \neq \bot$ and $\delta_{\mathcal{P}}(p,d) \neq \bot$,

$$(1 - a_{q,p}) + \sum_{y \in Y}(j_{e,d,y}) + a_{\delta_{\mathcal{P}}(q,e), \delta_{\mathcal{P}}(p,d)} \geq 1. \quad (14)$$

We also replace Constraint Set (8) by linearized forms:

For each $q, p \in Q_{\mathcal{P}}$ and $e \in E$ s.t. $\delta_{\mathcal{P}}(q,e) \neq \bot$,

$$(1 - a_{q,p}) + \sum_{y \in Y} b_{e,y} + a_{\delta_{\mathcal{P}}(q,e), p} \geq 1. \quad (15)$$

Similarly, we linearize Constraint Set (9) as follows.

For each $q, p \in Q_{\mathcal{P}}$ and $e \in E$ s.t. $\delta_{\mathcal{P}}(p,e) \neq \bot$,

$$(1 - a_{q,p}) + \sum_{y \in Y} b_{e,y} + a_{q, \delta_{\mathcal{P}}(p,e)} \geq 1. \quad (16)$$

Now, we have an ILP formulation of MSSVI, which can

be solved directly by any of the many existing highly-optimized ILP solvers. This ILP formulation not only can be used for obtaining solutions to MSSVI but also to compute feasible (rather than optimal) solutions for large instances of world graphs for which exact solutions to MSSVI cannot be computed in a reasonable amount of time.

VII. CASE STUDIES

In this section, we present case studies, using the ILP formulation from the previous section to solve some representative instances of MSSVI. All trials were performed on an Ubuntu 16.04 computer with a 3.6 GHz processor.

A. Case study 1: Computer Science Department

Recall Fig. 2, with a map of a small computer science department. For this world graph, we executed several instances of the MSSVI with different itineraries to verify the algorithm's correctness. Table I shows results on those instances. The first two instances consider extreme itineraries as boundary test cases. For the first, the itinerary consists of all walks on the world graph, including the empty string; in the second instance, the itinerary does not have any walks over the world graph. In both of these instances, the optimal solution (which has size 0, i.e. no sensor is needed) was correctly found. The third scenario considers an itinerary moving any way, other than entering room G. To certify this itinerary, it suffices to turn on only one of the sensors o_3 and o_4, both located in room G. Again, our implementation found this solution correctly. The fourth itinerary specifies a single sequence in which the agent enters room A from R_1, then it enters room B from the right door between A and B, then passes through R_2 to enter D. There is no certifying sensor selection for this itinerary because there is another walk, $e_1 e_5 e_7 e_{13}$, that is not within the claimed itinerary but which produces the same sequence of world-observations, for any selection of sensors. In contrast, the fifth scenario, whose itinerary includes both the walk from the fourth scenario and additionally $e_1 e_5 e_7 e_{13}$, can be solved with no need to turn on sensors o_3 and o_4. The last itinerary specifies all walks in which the agent does not enter any room but only room G at the end after traveling along the corridor between any of regions R_1, R_2 and R_3. For this itinerary, it is required to turn on sensors o_1, o_2 and b to verify that the robot did not enter any of rooms A, B, C, D, F. It also requires either o_3 or o_4 be turned on to ensure the robot enters room G at the end. Our program took less than 5 seconds to compute a minimal certifying sensor selection for each of these itineraries.

Though small, this case study suggests the correctness of our algorithm. The next section tests its scalability.

B. Case study 2: Where eagles soar

Ornithologists employ cellular-network devices to track migratory patterns of larger birds, allowing new insights to be gleaned [4]. Fig. 5a shows aggregated tracking data (from [9]) for journeys made by eagles over a year. The surprisingly infrequent flights across open water might lead one to hypothesize that eagles circle the Caspian Sea (the region made visually salient in the figure). To validate this hypothesis would require purchasing data roaming capabilities from cellular-network operators across multiple countries in this region. Fig. 5b shows an approach to model the problem of minimizing these costs. The map is divided into subregions, each representing a vertex of a world graph. Edges of the world graph are between adjacent hexagons. When whole subregions fall substantially within a single country, they have been assigned a color representing the potential of purchasing data service for that country. There are 10 colors, representing the sensor set S. We model the hypothesis of circling the Caspian Sea by an itinerary containing walks that visit III·II·I·III, III·I·II·III, or their extra cyclic permutations. The DFA describing this itinerary consists of 7 states.

The observations provided by the cellular network are akin to the occupancy sensors in the previous example, with events triggered when the eagle enters or leaves each hexagonal cell. Our decomposition, shown in Fig. 5b, has 36 colored cells. Accordingly, there are 72 events. An additional 9 cells are uncolored. It took 734.83 seconds for our implementation to form the ILP model and then it took 270.17 seconds to find an optimal solution. In this solution, only 6 out of the 10 sensors (the colors listed in the caption of Fig. 5) were turned on. Before finding an optimal solution, the program found feasible solutions of sizes 10, 9, and 8 respectively in 83.07, 90.91, and 176.13 seconds.

VIII. CONCLUSION

We have examined the question of selecting the fewest sensors subject to the requirement that they have adequate distinguishing power to differentiate motions conforming to an itinerary from those that do not. This optimization question fits the resource minimization concern that underlies several useful applications. Our formulation of this problem allows for the possibility that when a sensor is selected, it can provide readings for events in potentially multiple places. To solve the problem, rather than proposing a custom algorithm, we give an ILP formulation for it, leveraging decades of optimization on such solvers. This approach is seen to solve instances of moderate size — including a small-scale case study motivated by wildlife tracking.

For future work, the steps that have become standard when dealing with NP-hard problems remain: seeking special-cases that possess some additional structure making them easier, understanding the problem using more nuanced parameterization (i.e., fixed parameter tractability approaches),
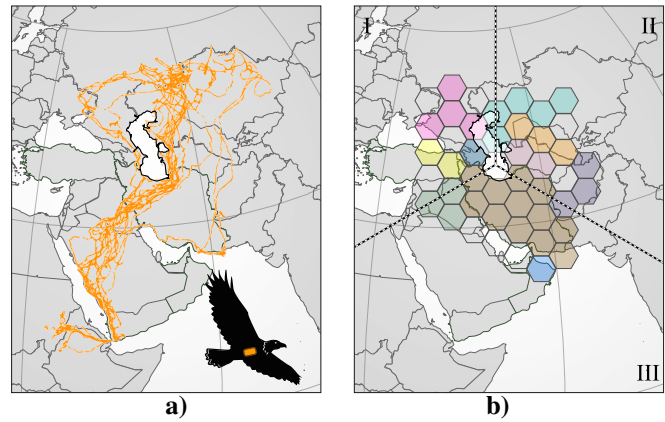

Fig. 5. **a)** An aggregate of tracking data, reproduced from [9], showing the journeys across parts of North Africa and Central Asia made by eagles over a period of one year. Fig. **b)** The decomposition of the map into subregions. Subregions who fall substantially within a single country have been assigned a color representing the potential of purchasing data roaming service from that country (there are 10 colors, representing sensors set S). The optimal sensor selection to certify the hypothesis of circling the Caspian Sea has six sensors, consisting of the sets of hexagons of the following colors: ⬡, ⬡, ⬡, ⬡, ⬡, and ⬡.

and custom heuristics and approximation algorithms. Also, there might be room to improve the ILP to make it more effective.

REFERENCES

[1] F. Cassez and S. Tripakis, "Fault diagnosis with static and dynamic observers," *Fundamenta Informaticae*, vol. 88, no. 4, pp. 497–540, 2008.

[2] A. Haji-Valizadeh and K. A. Loparo, "Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems," *IEEE Trans. on Auto. Control*, vol. 41, no. 11, pp. 1579–1593, 1996.

[3] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, 1972, pp. 85–103.

[4] B.-U. Meyburg, P. Paillat, and C. Meyburg, "Migration routes of steppe eagles between asia and africa: a study by means of satellite telemetry," *The Condor*, vol. 105, no. 2, pp. 219–227, 2003.

[5] J. O'Rourke, *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[6] H. Rahmani, D. A. Shell, and J. M. O'Kane, "Planning to chronicle," in *Algorithmic Found. of Robotics XIV*, 2021, pp. 277–293.

[7] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. on Auto. Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[8] D. Sears and K. Rudie, "Minimal sensor activation and minimal communication in discrete-event systems," *Discrete Event Dynamic Systems*, vol. 26, no. 2, pp. 295–349, Jun. 2016.

[9] J. Stewart, "Incredible Map Shows Flight Routes of Eagles Over the Course of One Year," March 6, 2019, Last access: Feburary 27, 2021. [Online]. Available: https://mymodernmet.com/eagle-tracking-map/

[10] Tae-Sic Yoo and S. Lafortune, "NP-completeness of sensor selection problems arising in partially observed discrete-event systems," *IEEE Trans. on Auto. Control*, vol. 47, no. 9, pp. 1495–1499, 2002.

[11] B. Tovar, F. Cohen, and S. M. LaValle, "Sensor beams, obstacles, and possible paths," in *Algorithmic Foundations of Robotics VIII*, 2009, pp. 317–332.

[12] W. Wang, C. Gong, and D. Wang, "Optimizing sensor activation in a language domain for fault diagnosis," *IEEE Trans. on Auto. Control*, vol. 64, no. 2, pp. 743–750, 2018.

[13] X. Yin and S. Lafortune, "A general approach for optimizing dynamic sensor activation for discrete event systems," *Automatica*, vol. 105, pp. 376–383, 2019.

[14] J. Yu and S. M. LaValle, "Cyber detectives: Determining when robots or people misbehave," in *Algorithmic Foundations of Robotics IX*, 2010, pp. 391–407.

[15] ——, "Story validation and approximate path inference with a sparse network of heterogeneous sensors," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4980–4985.