Coverage Analysis of Mobile Agent Trajectory via State-Based Opacity Formulations✩

Anooshiravan Saboori

Coordinated Science Laboratory, and Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign

Christoforos N. Hadjicostis∗

Department of Electrical and Computer Engineering, University of Cyprus, and Coordinated Science Laboratory, and Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign

Abstract

This paper performs coverage analysis of mobile agent trajectory utilizing discrete event system models and employing state-based notions of opacity. Non-deterministic finite automata with partial observation on their transitions are used to simultaneously capture both the kinematic model of a mobile agent and the information that becomes available by a set of sensors that are deployed in a given environment. The information provided by the set of sensors is analyzed to track the passage of a mobile agent through certain secret (strategic) locations at specific time instants using state-space notions of opacity, which arise naturally as the way to capture/analyze secrecy and privacy considerations in such settings and to answer related coverage questions. Realistic examples of two-dimensional environments equipped with sensors monitoring the location of a mobile agent that follows a known kinematic model

✩This material is based upon work supported in part by the U.S. National Science Foundation under NSF ITR Award 0426831 and NSF CNS Award 0834409. The research leading to these results has also received funding from the European Commission Seventh Framework Programme (FP7/2007-2013) under grant agreements INFSO-ICT-223844 and PIRG02-GA-2007-224877. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF or EC.

∗Corresponding author’s address: University of Cyprus, 75 Kallipoleos Avenue, P.O. Box 20537, 1678 Nicosia, Cyprus. Fax: +357 22892260

Email addresses: saboori2@uiuc.edu (Anooshiravan Saboori), chadjic@ucy.ac.cy (Christoforos N. Hadjicostis)
are subsequently analyzed by adopting existing tools in the discrete event systems area.

Key words: Discrete Event Systems, System Security, Information Flow, State Estimation, Sensors

1. Introduction

1.1. Overview and Motivation

Modern tactical intelligence, surveillance and reconnaissance (ISR) technologies include a vast assortment of air and ground based radars, unmanned autonomous vehicles, unattended ground-based sensors (UGS) and human intelligence observations. These sensing platforms produce streams of events based on what they detect within their physical range. The resulting event stream must be combined with the physical/logical limitations of the environment in order to identify and track vehicles, aircraft, and/or people. As a result, the problem of tracking in sensor networks has received considerable attention (Crespi et al., 2008; Reid, 1979; Songhwai Oh et al., 2005).

This paper studies the related problem of security and privacy aspects in planar sensor networks. More specifically, the paper studies the problem of tracking the trajectory of the location of a given vehicle (as time goes on) with respect to a certain set of strategic (secret) locations using information from a set of sensors that are deployed in a given planar region. These trajectories can be of interest for a variety of reasons. For example, feasible trajectories can be exploited in order to hide the origin of a trajectory from an observer who is employing the sensor network in an effort to determine whether the vehicle originated from a strategically important location or whether the vehicle passed from this particular set of strategic locations at some instant of time. It is assumed that the locations of the deployed sensors are known and remain constant, but the sensor coverages are allowed to differ (i.e., heterogeneous sensor networks are considered).
In order to study this problem, a security notion called infinite-step opacity (Saboori and Hadjicostis, 2009) is used. This notion is defined assuming that the underlying system is a discrete event system (DES) that can be modeled as a non-deterministic finite automaton with partial observation on its transitions. As shown later in the paper, non-deterministic finite automata with partial observation on their transitions can be used to conveniently model the movement of mobile agents in various terrains; thus, state-based opacity notions (including infinite-step opacity) are ideally suited for coverage analysis and verification of security/privacy properties of interest. It is assumed that the intruder has full knowledge of the system model and is able to track the observable transitions in the system via the observation of the associated labels. Given (partial) knowledge of the initial state of the system, infinite-step opacity requires that entrance of the system state to a subset of system states, called the set of secret states, remain opaque (uncertain) to the intruder. By defining the set of secret states to be the set of strategic locations, infinite-step opacity translates to the ability of the intruder to determine, given all available (past and current) sensor readings, whether the vehicle has gone through some strategic locations at a specific instant in time.

In (Saboori and Hadjicostis, 2009), a verification algorithm for infinite-step opacity using appropriate state estimators was proposed and its computational complexity was analyzed. In this paper, existing tools and appropriate transformations are used to implement the algorithm proposed in (Saboori and Hadjicostis, 2009) and to establish its applicability to coverage analysis of sensor networks. The verification method studied in (Saboori and Hadjicostis, 2009) for infinite-step opacity has $O(2^{N^2})$ state and time complexity but this paper introduces a reduced-complexity verification method which has $O(8^N)$ state and time complexity.

Due to the cost associated with sensor deployment, designers are typically faced with limitations on the number (or location) of deployed sensors. Minimal sensor selection problems aim to find a set of sensors such that: (i) properties of interest about the sensor network hold, and (ii) if any of the sensors in the set is turned off,
the property seizes to hold (Debouk et al., 2002). This paper studies sensor selection problems where the desired property is taken to be the ability of the sensor network to identify the passage of the vehicle through strategic locations; thus, the minimal sensor selection problem translates to the problem of finding a set of sensors (from a given set of available sensors) such that turning off any of the sensors in this set renders the system infinite-step opaque. A top-down algorithm is proposed to solve the minimal sensor selection problem and the correctness of the algorithm is established by showing that lack of infinite-step opacity is a *mask-monotonic* property (Jiang et al., 2003). The effectiveness and applicability of this algorithm are subsequently evaluated via an extensive set of simulations.

1.2. Related Work

The framework considered here for tracking analysis is related to the *weak model* introduced in (Crespi et al., 2008) for studying the *trackability* of sensor networks. A sensor network is trackable if the rate of growth of the number of state sequences that are consistent with a given sequence of observations is *sub-exponential* in the length of the observations. The authors of (Crespi et al., 2008) show that the rate of growth of the number of consistent state sequences is either polynomial or exponential in the size of the observations; they also obtain necessary and sufficient conditions on the placement of the sensors such that this growth is polynomial, i.e., such that the sensor network is trackable. While the model in this paper is similar to that of (Crespi et al., 2008), the two papers study different problems: trackability studies the *number* of state sequences that are consistent with the given sequence of observations, while infinite-step opacity focuses on the state estimates that appear in these state sequences (and is concerned with whether they fall exclusively in the set of secret states at specific points in time).

The authors of (Songhwai Oh et al., 2005) study the problem of *multiple-target* tracking for sensor networks with deterministic deployment subject to communication delays, noisy observations, false alarms, and data packet losses. Assuming that
the noisy observation of the state of the object is measured with a known detection probability (less than one) and that the number of false alarms follows a Poisson distribution, the authors of (Songhwai Oh et al., 2005) use the Markov Chain Monte Carlo Association algorithm to estimate the number of moving objects and their state trajectories. Unlike the framework of (Songhwai Oh et al., 2005), the formulation studied in this paper does not assume probabilities associated with the target movements. Furthermore, the focus in this paper is on characterizing the target trajectories with respect to a set of secret locations.

The sensor selection problem studied here is related to the coverage problem in sensor networks, which has been studied with respect to different objectives and metrics (see, for example, (Lazos and Poovendran, 2006; Xing et al., 2005; Poduri and Sukhatme, 2004)). The characteristic attribute used to classify different approaches to the coverage problem is whether sensor deployment is deterministic or stochastic. In deterministic sensor deployment, the possible locations of the sensors are preselected, whereas in stochastic sensor deployment, sensors are deployed according to a known probability distribution. The coverage problem in the deterministic case boils down to the problem of finding the optimal placement for sensors such that a target coverage objective is met (Xing et al., 2005; Poduri and Sukhatme, 2004). In the stochastic case, the coverage problem reduces to finding the number of sensors that must be deployed, given the sensor deployment distribution, in order for every point in the field of interest to be covered by at least \( k \) sensors with some target probability \( p \) (Lazos and Poovendran, 2006). The sensor selection problem studied here is related to the coverage problem with deterministic sensor deployment (Xing et al., 2005).

The problem of connected coverage has been studied by (Xing et al., 2005), where the authors provide a geometric analysis that relates coverage to connectivity, and also define the necessary conditions for a sensor network to be connected. The authors of (Poduri and Sukhatme, 2004) studied the problem of deterministic coverage under the additional constraint that each sensor must have at least \( k \) neighbors. In its simplest form (full event observation), the minimum sensor selection problem considered here
can be seen as the study of the connectivity of the sensor network graph with respect to the set of secret states (strategic locations).

2. Preliminaries and Background

2.1. Notation

2.1.1. Languages and Automata

Let Σ be an alphabet and denote by Σ* the set of all finite-length strings of elements of Σ, including the empty string ε. A language $L \subseteq \Sigma^*$ is a subset of finite-length strings from strings in Σ*. For a string ω, $\overline{\omega}$ denotes the prefix-closure of ω and is defined as $\overline{\omega} = \{ t \in \Sigma^* | \exists s \in \Sigma^*: ts = \omega \}$. The post-string $\omega/t$ of ω after $t \in \overline{\omega}$ is defined as $\omega/t = \{ s \in \Sigma^* | ts = \omega \}$. For any string ω, $|\omega|$ denotes the length of ω and the length of ε is taken to be $|\epsilon| = 0$ (Cassandras and Lafortune, 2008; Wonham, 2009).

A DES is modeled in this paper as a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, where $X = \{0, 1, \ldots, N - 1\}$ is the set of states, $\Sigma$ is the set of events, $\delta : X \times \Sigma \rightarrow 2^X$ (where $2^X$ is the power set of $X$) is the non-deterministic state transition function, and $X_0 \subseteq X$ is the set of possible initial states. The function $\delta$ can be extended from the domain $X \times \Sigma$ to the domain $X \times \Sigma^*$ in the routine recursive manner: $\delta(i, ts) := \bigcup_{j \in \delta(i, t)} \delta(j, s)$, for $t \in \Sigma$ and $s \in \Sigma^*$ with $\delta(i, \epsilon) := i$. The behavior of DES $G$ is captured by $L(G) := \{ s \in \Sigma^* | \exists i \in X_0, \delta(i, s) \text{ is non-empty} \}$.

In general, only a subset $\Sigma_{obs}$ of the events can be observed. Typically, one assumes that $\Sigma$ can be partitioned into two sets, the set of observable events $\Sigma_{obs}$ and the set of unobservable events $\Sigma_{uo}$. The natural projection $P_{\Sigma_{obs}} : \Sigma^* \rightarrow \Sigma_{obs}^*$ can be used to map any trace executed in the system to the sequence of observations associated with it. This projection is defined recursively as $P_{\Sigma_{obs}}(ts) = P_{\Sigma_{obs}}(t)P_{\Sigma_{obs}}(s)$, $t \in \Sigma$, $s \in \Sigma^*$, with

$$P_{\Sigma_{obs}}(t) = \begin{cases} t, & \text{if } t \in \Sigma_{obs}, \\ \epsilon, & \text{if } t \in \Sigma_{uo} \cup \{\epsilon\}, \end{cases}$$
Figure 1: (a) DES $G$; (b) Current-state estimator $G_{0,obs}$ for DES $G$ discussed in Example 1.

(Cassandras and Lafortune, 2008; Wonham, 2009). In the sequel, the subscript $\Sigma_{obs}$ in $P_{\Sigma_{obs}}$ will be dropped if it is clear from the context.

2.1.2. Current-State Estimator

Assume a known non-deterministic finite automaton $G = (X, \delta, \Sigma, X_0)$ under some natural projection map $P_{\Sigma_{obs}}$ (with respect to the set of observable events $\Sigma_{obs} \subseteq \Sigma$) is used to model a DES and the observations it generates. Given a sequence of observations $\omega$, the state of the system might not be identifiable uniquely due to the lack of precise knowledge about the initial state, the non-determinism that is present in the system, and the partial observation of events. The set of states that the system might reside in given that $\omega$ was observed is referred to as the current-state estimate. The current-state estimator is a deterministic automaton $G_{0,obs}$ which captures these estimates and can be constructed as follows (Caines et al., 1991). Each state of $G_{0,obs}$ is associated with a unique subset of states of the original DES $G$ (so that there are at most $2^{|X|} = 2^N$ states). The initial state of $G_{0,obs}$ is associated with $X_0$, representing the fact that the initial state could be any state in $X_0$. At any state $Z$ of the estimator ($Z \subseteq X$), the next state upon observing an event $\sigma \in \Sigma_{obs}$ is the unique state of $G_{0,obs}$ associated with the set of states that can be reached from (one or more of) the states in $Z$ via a string of events that generates the observation $\sigma$ (if this set is empty then $\sigma$ is not possible from this particular state of $G_{0,obs}$). This automaton is denoted by $G_{0,obs} = AC(2^X, \Sigma_{obs}, \delta_{obs}, X_0)$ where its state set $2^X$ is the power set of $X$, its event set $\Sigma_{obs}$ is the set of observable events of $G$, $\delta_{obs}$ is the
(deterministic) transition function, its initial state $X_0$ is the set of initial states for $G$, and $AC$ is the part of the automaton that is accessible from initial state $X_0$. If the notation $X_{obs} \subseteq 2^X$ is used to denote the reachable states from $X_0$ under $\delta_{obs}$, then $AC(2^X, \Sigma_{obs}, \delta_{obs}, X_0) = (X_{obs}, \Sigma_{obs}, \delta_{obs}, X_0)$. The following example clarifies this construction.

**Example 1.** Consider the DES $G$ in Figure 1-a. Assuming that $\Sigma_{obs} = \{\alpha, \beta, \theta\}$ and $X_0 = X$ (i.e., the initial state could be any state), the current-state estimator $G_{0, obs}$ in Figure 1-b can be constructed as follows: starting from the initial state $X_0$ and observing $\beta$, the current state is any of the states in $\{3, 4, 5\}$; at this new state, the possible transitions for $G_{obs}$ are the transitions possible in $G$ from at least one of the states in $\{3, 4, 5\}$. Following this procedure, $G_{0, obs}$ can be completed as in Figure 1-b. Clearly, $X_{obs}$ in this case is $\{\{1\}, \{4\}, \{5\}, \{1, 2\}, \{4, 5\}, \{3, 4, 5\}, \{0, 1, 2, 3, 4, 5\}\}$.

2.1.3. Initial-State Estimator

Assume a known non-deterministic finite automaton $G = (X, \delta, \Sigma, X_0)$ under some natural projection map $P_{\Sigma_{obs}}$ (with respect to the set of observable events $\Sigma_{obs} \subseteq \Sigma$) is used to model a DES and the observations it generates. Given a sequence of observations $\omega$, the initial-state estimate is the set of all states that belong to the set of initial states $X_0$, and which could have generated the sequence of observations $\omega$. The work in (Saboori and Hadjicostis, 2008) introduced a (deterministic) finite automaton called initial-state estimator (ISE) which can be used to obtain initial-state estimates (much like the current-state estimator introduced in the previous section can be used to obtain current-state estimates).

The ISE uses the notion of a state mapping to capture all the information included in any sequence of observations (of finite but arbitrary length) regarding the initial state of the system. A state mapping $m \in 2^{X^2}$ is a set whose elements are pairs of states: the leftmost component of each element (pair) is the starting state and the rightmost component is the ending state. (For state mapping $m \in 2^{X^2}$, $m(1)$ is used to denote the set of starting states and $m(0)$ to denote the set of ending states.) The
composition operator $\circ : 2^{X^2} \times 2^{X^2} \rightarrow 2^{X^2}$ for state mappings $m_1, m_2 \in 2^{X^2}$ is defined in (Saboori and Hadjicostis, 2008) as

$$m_1 \circ m_2 := \{(j_1, j_3) \mid \exists j_2 \in X, (j_1, j_2) \in m_1, (j_2, j_3) \in m_2\}.$$

In (Saboori and Hadjicostis, 2008), the $\omega$-induced state mapping is defined for $\omega \in \Sigma_{\text{obs}}^*$ as

$$M(\omega) = \{(i, j) \mid i, j \in X, \exists t \in \Sigma^* \{P(t) = \omega, j \in \delta(i, t)\}\}.$$

Note that $M(\omega) = \emptyset$ denotes the fact that the sequence of observations $\omega$ is not feasible in DES $G$ regardless of its initial state. Finally, for any $Z \subseteq X$, the operator $\odot : 2^X \rightarrow 2^{X^2}$ is defined in (Saboori and Hadjicostis, 2008) as $\odot(Z) = \{(i, i) \mid i \in Z\}$.

The ISE constructed in (Saboori and Hadjicostis, 2008) utilizes state mappings as follows: each state of the ISE is associated with a unique state mapping and, since the initial state of the system belongs to $X_0$, the mapping associated with the initial state is the mapping $m_0 = \odot(X_0)$. Given a new observation, the ISE current state transitions into an ISE state whose associated state mapping is the composition of the state mapping associated with the ISE current state and the state mapping induced by the new observation. Continuing in this way, one can build a (well-defined) structure which, at any given time, gives information about the current state and the initial state through the state mappings associated with each of its states. Note that this structure is guaranteed to be finite and has at most $2^{N^2}$ states where $N$ is the number of states of DES $G$. The initial-state estimator for a given non-deterministic automaton $G = (X, \Sigma, \delta, X_0)$ is denoted by $G_{\infty, \text{obs}} = AC(2^{X_0}, \Sigma_{\text{obs}}, \delta_{\infty, \text{obs}}, X_{\infty, 0})$ where $2^{X^2}$ (the power set of $X^2$, where $X$ is the set of states of $G$) is the state set, $\Sigma_{\text{obs}}$ is the set of observable events, $\delta_{\infty, \text{obs}}$ is the (deterministic) transition function, $X_{\infty, 0} = \odot(X_0)$ is the initial state, and $AC$ is the part of the automaton that is accessible from initial state $X_{\infty, 0}$. The notation $X_{\infty, \text{obs}} \subseteq 2^{X^2}$ is used to denote the reachable states from $X_{\infty, 0}$ under $\delta_{\infty, \text{obs}}$ so that $AC(2^{X_0}, \Sigma_{\text{obs}}, \delta_{\infty, \text{obs}}, X_{\infty, 0}) = (X_{\infty, \text{obs}}, \Sigma_{\text{obs}}, \delta_{\infty, \text{obs}}, X_{\infty, 0})$.

The following example illustrates the construction of the initial-state estimator.
Example 2. The ISE for the DES $G$ of Figure 1-a with $\Sigma_{obs} = \{\alpha, \beta, \theta\}$ is shown in Figure 2. The initial uncertainty is assumed to be equal to the state space and hence the initial state of the ISE is the state mapping $m_0 = \{(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\}$. Upon observing $\beta$ the next state of the ISE becomes $m_1 = \{(0, 3), (0, 4), (1, 4), (2, 5), (3, 5), (5, 5)\} = m_0 \circ M(\beta)$, where $M(\beta) = \{(0, 3), (0, 4), (1, 4), (2, 5), (3, 5), (5, 5)\}$ is the state mapping induced by observation $\beta$ (on the right of Figure 2 a graphical way is used to describe the state mapping associated with the states of the ISE). If, instead of $\beta$, one initially observes $\alpha$, the state mapping that is induced by observing $\alpha$ is $m_0 \circ M(\alpha) = M(\alpha)$, where $M(\alpha) = \{(4, 5)\}$. To take into account the observation $\beta$ followed by observation $\alpha$, one needs to compose the state mapping $m_1$ with $M(\alpha)$ which results in $\{(0, 5), (1, 5)\} \equiv m_5$. Using this approach for all possible observations (from each state), the ISE construction can be completed as shown in Figure 2. To avoid cluttering the diagram, the state that corresponds to the empty state mapping (and that is reached via sequences of observations that cannot be generated by $G$) is not included in the figure.

2.2. Infinite-Step Opacity

This section formally defines the notion of infinite-step opacity (refer to Figure 3 for a graphical representation).
Definition 1 (Infinite-Step Opacity). Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map $P$ with respect to the set of observable events $\Sigma_{obs}$ ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, automaton $G$ is infinite-step opaque with respect to $S$ and $P$ (or $(S, P, \infty)$-opaque), if for all $t \in \Sigma^*$, $t' \in \bar{t}$, and $i \in X_0$,

$$\{\exists j \in S \{ j \in \delta(i, t'), \delta(j, t/t') \text{ is non-empty} \} \} \Rightarrow$$

$$\{\exists s \in \Sigma^*, \exists s' \in \bar{s} \{ P(s) = P(t), P(s') = P(t'), \exists i' \in X_0, \exists j' \in \delta(i', s') \{ j' \in X - S, \delta(j', s/s') \text{ is non-empty} \} \} \}. \blacksquare$$

For $t, s \in L(G)$ with $P(s) = P(t)$ one says that $t$ passes through state $j$ when $s$ passes through state $j'$ if there exist $t' \in \bar{t}$, $s' \in \bar{s}$, and $i, i' \in X_0$ such that $j \in \delta(i, t')$, $j' \in \delta(i', s')$ while $P(t') = P(s')$, and both $t/t'$ and $s/s'$ have continuations from states $j$ and $j'$, respectively. According to Definition 1, DES $G$ is $(S, P, \infty)$-opaque if for every string $t$ in $L(G)$ that visits a state $j$ in $S$ (with string $t$ having a continuation from state $j$), there exists a string $s$ in $L(G)$ with $P(s) = P(t)$ such that when string $t$ passes through the state $j$ in $S$, string $s$ passes through a state $j'$ in $X - S$ (with string $s$ having a continuation from state $j'$).

Remark 1. Infinite-step opacity is a special case of the more general notion of opacity which aims at determining whether a given system’s secret behavior (i.e., a subset of the behavior of the system that is considered critical and is usually represented by a predicate) is kept opaque to outsiders (Bryans et al., 2008). More specifically, this requires that the intruder (modeled as a passive observer of the system’s behavior) never be able to establish the truth of the predicate. Opacity was first introduced in (Mazari, 2004) for secure protocols and then was extended to generalized transition systems.
in (Bryans et al., 2008). The work in (Saboori and Hadjicostis, 2007, 2008, 2009) studies this problem for the case of finite automata and uses state-based predicates for opacity. Due to the restriction that the underlying system is a non-deterministic finite automaton with partial observation of its transitions, systematic methodologies for verifying opacity properties of interest are obtainable. This might not possible in general; for example, the problem of verifying infinite-step opacity is decidable for non-deterministic finite automata but this is not the case for the general framework of (Bryans et al., 2008). Finite automata with partial observation on their transitions can be used to conveniently model the movement of mobile agents in various terrains; furthermore, as established later in this paper, state-based notions of opacity translate to certain interesting security-related properties about the trajectories that these agents follow.

### 2.2.1. Verifying Infinite-Step Opacity Using a Bank of ISE

One way to verify that a system is infinite-step opaque is to verify that at any point during the observation process, knowing the sequence of observations before reaching that point, in addition to any future observation sequence (that is possible from that point onward), does not (and will not) allow the intruder to determine that the set of possible states at that point is a subset of the set of secret states (Saboori and Hadjicostis, 2009). Using this intuition, the verification methodology proposed in (Saboori and Hadjicostis, 2009) consists of two phases: (i) find all possible estimates of the system’s current state along any possible sequence of observations, and (ii) for each point in this trajectory (set of possible system states), calculate the information that can be gained about the state at that point by possible observation sequences from that point onward. The first phase can be achieved via a standard current-state estimator which captures the estimate of the current state given a sequence of observations. The second phase requires the construction of an ISE-like state estimator for each possible uncertainty about the current-state estimate (which is now used as the initial state estimate for the ISE-like state estimator). In other words, for
each set of state estimates \( Z \subseteq X \) provided in the first phase, one constructs an ISE whose initial state is associated with the state mapping \( \odot(Z) \). Clearly, if any of these ISEs contains a state with associated state mapping \( m \) such that its set of starting states (is non-empty and) contains elements only in \( S \) (i.e., if \( m(1) \subseteq S \) and \( m \neq \emptyset \)), then DES \( G \) is not infinite-step opaque. The following theorem from (Saboori and Hadjicostis, 2008) formalizes this discussion.

**Theorem 1.** Consider a non-deterministic finite automaton \( G = (X, \Sigma, \delta, X_0) \), a natural projection map \( P \) with respect to the set of observable events \( \Sigma_{obs} \) \( (\Sigma_{obs} \subseteq \Sigma) \), and a set of secret states \( S \subseteq X \). For each set of current-state estimates \( Z_n \) associated with a state of its current-state estimator \( G_{0,obs} \), construct the initial-state estimator \( G_{\infty,obs}^{(n)} = AC(2^{X^2}, \Sigma_{obs}, \delta_{\infty,obs}^{(n)}, X_{\infty,0}^{(n)}) \equiv (X_{\infty,obs}^{(n)}, \Sigma_{obs}, \delta_{\infty,obs}^{(n)}, X_{\infty,0}^{(n)}) \) by setting its initial state \( X_{\infty,0}^{(n)} \) to be \( \odot(Z_n) \). Then, DES \( G \) is \( (S, P, \infty) \)-opaque if and only if

\[
\forall n, \forall m \in X_{\infty,obs}^{(n)} : m(1) \not\subseteq S \text{ or } m = \emptyset, \tag{1}
\]

where \( X_{\infty,obs}^{(n)} \) is the set of states in \( G_{\infty,obs}^{(n)} \) that are reachable from \( X_{\infty,0}^{(n)} = \odot(Z_n) \) and \( m(1) \) denotes the set of starting states of state mapping \( m \).

**Example 3.** This example shows that DES \( G \) in Figure 1-a is not \( (\{2\}, P, \infty) \)-opaque. To verify infinite-step opacity, one needs to first construct the current-state estimator \( G_{0,obs} \) as in Figure 1-b. This state estimator has seven states \( Z_0 = \{1\}, \ Z_1 = \{4\}, \ Z_2 = \{5\}, \ Z_3 = \{1, 2\}, \ Z_4 = \{4, 5\}, \ Z_5 = \{3, 4, 5\}, \ Z_6 = \{0, 1, 2, 3, 4, 5\} \); hence, one needs to construct seven ISEs with initial states \( \{(1, 1)\}, \{(4, 4)\}, \{(5, 5)\}, \{(1, 1), (2, 2)\}, \{(4, 4), (5, 5)\}, \{(3, 3), (4, 4), (5, 5)\} \), and \( \{(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\} \), respectively. However, among these initial states (i.e., state mappings) only state mapping \( \{(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\} \), which corresponds to \( Z_6 \), and state mapping \( \{(1, 1), (2, 2)\} \), which corresponds to \( Z_3 \), contain the secret state 2. Since the set of possible initial states can only decrease with additional observations (Caines et al., 1991), one only needs to construct the two ISEs that have these two state mappings as their initial states. (i) The ISE \( G_{\infty,obs}^{(6)} \) with initial state mapping corresponding to \( Z_6 \) is indeed the initial-state estimator in Figure 2 which was con-
Figure 4: ISE $G^{(3)}_{\infty,obs}$ corresponding to state $Z_3 = \{1, 2\}$ in the current-state estimator of the DES $G$ in Figure 1-a.

Considered in Example 2. It can be easily verified that the set of starting states of all state mappings associated with this ISE has states outside the set of secret states. (ii) The ISE $G^{(3)}_{\infty,obs}$ with initial state corresponding to $Z_3$ is depicted in Figure 4 (again, the figure does not include the state that corresponds to the empty state mapping and is reached via sequences of observations that cannot be generated by $G$). State $m_3 = \{(2, 5)\}$ in $G^{(3)}_{\infty,obs}$ violates $(\{2\}, P, \infty)$-opacity since its set of starting states only contains state 2 which is a secret state. State $m_3$ is reachable in $G^{(3)}_{\infty,obs}$ via $\beta\beta$ from $m_0$. Moreover, $m_0$ in this ISE corresponds to the state in $G_{0,obs}$ (in Figure 1-b) that is reached via observation $\theta$. Putting these two pieces of information together, one can conclude that observing $\theta\beta\beta$ reveals that the system has gone through state 2, which is a secret state.

Note that using both the ISE and the current-state estimator for verifying infinite-step opacity requires that for each state of the current-state estimator, an ISE-like state estimator be constructed. Since there are at most $2^N$ states for the current-state estimator, the complexity of this method is $O(2^N \times 2^{N^2})$ or equivalently $O(2^{N^2})$. This exponential complexity is not desirable for implementation purposes; in Section 5.4 it is shown that for a fixed set of secret states $S$ this complexity can be reduced to $O(8^N)$. Keep in mind, however, that verifying infinite-step opacity for DES $G$...
under a natural projection map \( P_{\Sigma_{obs}} \) with \(|\Sigma_{obs}| > 1\) is PSPACE-hard (Saboori and Hadjicostis, 2009) and it is unlikely that any algorithm can verify this property in polynomial time.

3. Security and Privacy Considerations for Mobile Agent Trajectories

3.1. Modelling Framework

Consider a vehicle capable of moving in a two-dimensional space that is modeled as a rectangular grid (Figure 5-a illustrates the modeling approach via a toy example of a \(2 \times 2\) grid). The state of the vehicle corresponds to the coordinates \((x, y)\) (or cell number) of its location in the grid, and any trajectory that the vehicle follows corresponds to a sequence of states. Clearly, the origin of the trajectory is captured by the initial state of the vehicle.

In order to model vehicle movement limitations (due to obstacles or logical/physical limitations in the vehicle motion), these movements are assumed to be describable via a kinematic model (a finite automaton) whose states are associated with the state (position) of the vehicle and whose transitions correspond to the possible movements of the vehicle at this position (up, right, diagonal, etc.). Figure 5-b depicts an example of a kinematic model \(H\) for a vehicle that moves in the grid in Figure 5-a.

It is assumed that a sensor network is deployed in the region where the vehicle moves and provides partial information about its location at any given time. If each sensor detects the presence of the vehicle in a cell or in some aggregation of cells, then when the vehicle passes through a cell within the coverage of a sensor, this sensor emits a signal to indicate this event. However, since sensors cannot determine the exact cell (within their coverage area) at which the vehicle resides, the kinematic model can be enhanced to encompass this information by assigning label \(\alpha\) to all transitions that end in a cell covered by sensor \(\alpha\). The label of transitions ending

\[\text{If a single sensor has the capability to distinguish the location of the vehicle with respect to a partition of cells in its coverage area, this can be captured by using multiple sensors to model each of the signals that might be provided by this single sensor.}\]
in areas covered by more than one sensor can be chosen to be a special label that indicates the set of all sensors covering that location. Assuming that the coverage area of sensor $\alpha$ only includes cell 2 and that the coverage area of sensor $\beta$ includes cells 0, 2 and 3, Figure 5-c depicts the non-deterministic automaton $G$ that models both the kinematic model of the vehicle and the corresponding sensor readings. Dotted arrows correspond to (unobservable) transitions that lead to locations not covered by any sensor.

3.2. Security and Privacy Considerations

One of the questions that might arise in the above context is that of characterizing all trajectories (sequences of states) that a vehicle can follow such that the origin of each trajectory or the location visited at some past point along the trajectory remains ambiguous to the sensor network. These trajectories can be of interest for a variety of reasons. For example, they can be employed to hide the origin of a trajectory from an observer who is employing the sensor network (i.e., who is observing the labels in Figure 5-c) to identify whether the origin belongs to a set of secret (strategically important) locations or to determine whether the vehicle passed from this particular set of locations at some instant of time. It is not hard to see that the passage of the vehicle through the set of strategic locations is not exposed to the intruder if and only if the automaton modeling the sensor network is infinite-step opaque with respect to the set of strategic locations (i.e., the set of secret states is taken to be the set of strategic locations).
4. Sensor Selection Related Questions

Due to the cost associated with deploying sensors, designers are typically faced with limitations on the number (or location) of deployed sensors. Minimum sensor selection problems aim to find the minimum number of sensors (or, more generally, a set of sensors of minimum cost) such that certain properties about the sensor network hold (Debouk et al., 2002). These properties vary depending on the underlying application and include (among others) observability, normality, diagnosability, and co-observability (Yoo and Lafortune, 2002; Jiang et al., 2003). In this paper, by letting the desired property be the ability of the sensor network to identify the passage of the vehicle through some set of strategic cells $S$, the minimum sensor selection problem is defined as the problem of finding (among a given set of available sensors) the set of sensors that has the minimum possible cardinality and guarantees that the system is not infinite-step opaque with respect to $S$. This problem is defined formally below.

**Definition 2 (Minimum Sensor Selection Problem).** Consider an instance of the sensor coverage problem, i.e., an $n_1 \times n_2$ 2-dimensional grid, the kinematic model $H$ of the vehicle that moves in this grid, a set of sensors with sensor coverage areas expressed in terms of aggregations of cells (for each sensor in the given set), and a set of strategic (secret) locations $S$. Find a subset of the given set of sensors that has minimum cardinality and ensures that the system is not infinite-step opaque.

Clearly, the solution to the minimum sensor selection problem can be obtained by searching through all sensor configurations to obtain the set of sensors (of minimum cardinality) that results in a system that is not infinite-step opaque. The problem has exponential complexity in the number of available sensors (because there is an exponential number of sensor configurations).

Another version of sensor selection problem is the minimal sensor selection problem which aims at finding a minimal solution, i.e., a set of sensors that have the following two properties: (i) if all sensors in the set are selected, the system is not
infinite-step opaque, (ii) by turning off any one sensor in this set, the system becomes infinite-step opaque. This problem is defined formally below.

**Definition 3 (Minimal Sensor Selection Problem).** Consider an instance of the sensor coverage problem, i.e., an $n_1 \times n_2$ 2-dimensional grid, the kinematic model $H$ of the vehicle that moves on the grid, a set of sensors with sensor coverage areas expressed in terms of aggregations of cells (for each sensor in the given set), and a set of strategic (secret) locations $S$. Find a subset of the given set of sensors which is minimal and ensures that the system is infinite-step opaque, i.e., find a set of sensors such that (i) when all sensors in the set are selected the system is not infinite-step opaque, and (ii) if any one of the sensors in this set is turned off, the system becomes infinite-step opaque.

The authors of (Jiang et al., 2003) study minimal solutions to the sensor selection problem for a general property $P$, and propose a top-down algorithm for finding a minimal solution with complexity polynomial in the number of available sensors and in the time needed to verify property $P$. They also argue that as long as property $P$ is mask-monotonic, the top-down algorithm obtains a minimal solution. For the property $P$ to be mask-monotonic the following needs to be true: if the given setting satisfies $P$ with set of deployed sensors $I$; then the given setting satisfies $P$ with any other set of deployed sensors $I'$ that includes $I$ (i.e., $I \subseteq I'$).

In the sequel, it is shown that lack of infinite-step opacity is mask-monotonic. First, for simplicity it is assumed that in Definitions 2 and 3 sensor coverage areas are not overlapping. In this case, turning on a sensor $\alpha$ can be modeled as adding event $\alpha$ to the set of observable events $\Sigma_{\text{obs}}$ (and removing it from the set of unobservable events). Assume that the system is not infinite-step opaque for a given set of deployed sensors $I_1$ denoted by projection map $P_{\Sigma_{\text{obs}}}$; then, there exists a string $s$ that passes through the set of secret states such that no other string $t$ with the same projection, $P_{\Sigma_{\text{obs}}}(s) = P_{\Sigma_{\text{obs}}}(t)$, passes through a non-secret state when $s$ passes through the set of secret states. Next, one turns on more sensors, and denotes the set of deployed
sensors by $I_2 \supseteq I_1$ and the corresponding projection map by $P_{\Sigma_2}$. Since $\Sigma_o_1 \subseteq \Sigma_o_2$,

$$P_{\Sigma_2}(s) = P_{\Sigma_2}(t) \Rightarrow P_{\Sigma_1}(s) = P_{\Sigma_1}(t),$$

which implies that strings $s$ and $t$ with $P_{\Sigma_2}(s) = P_{\Sigma_2}(t)$ will also satisfy $P_{\Sigma_1}(s) = P_{\Sigma_1}(t)$; thus, string $t$ has to pass through secret states when string $s$ passes through secret states even under mapping $P_{\Sigma_2}$. This implies that lack of infinite-step opacity is mask monotonic as long as sensor coverage is not overlapping.

Next, the case when sensors are overlapping is considered. Recall that the label of transitions ending in cells which are covered by more than one sensor is chosen to be a special label that indicates the set of all sensors covering that cell. Therefore, if the set of sensors that cover a cell changes — due to turning on more sensors — then the associated label with transitions ending in that cell will also change. However, if the transitions that end in a specific cell have identical label before turning on a sensor, they will also have identical label after turning on that sensor (although the associated label may now be different). Thus if $s$ and $t$ are such that $P_{\Sigma_2}(s) = P_{\Sigma_2}(t)$, then it also holds that $P_{\Sigma_1}(s) = P_{\Sigma_1}(t)$. The rest of the proof is similar to the previous case.

The above discussion implies that lack of infinite-step opacity is mask monotonic and one can use the top-down algorithm proposed in (Jiang et al., 2003) to find the minimal solution to the sensor selection problem in Definition 2. The top-down algorithm works as follows: one starts by selecting all sensors and switches sensors off one at a time (in some arbitrary order), until the system becomes infinite-step opaque. The last sensor is kept on so that the resulting set of selected sensors ensures that the system is not infinite-step opaque and, from this set of sensors, one tries to find a sensor to turn off such that the system does not become infinite-step opaque (e.g., by going through the sensors in some predefined order). If this is possible, that sensor is turned off and one continues until no sensor can be turned off, obtaining in this way a minimal solution to the sensor selection problem in Definition 2.
It should be noted that there may exist more than one minimal solutions and that the minimal solution does not necessarily correspond to the set of sensors that has minimum cardinality. This also explains why obtaining a minimal solution is generally easier than obtaining the minimum solution.

5. Simulation Studies

5.1. Generating Kinematic Models and Sensor Coverages

To study the verification process of infinite-step opacity for tracking problems in the context of sensor networks one first needs to obtain a system model. The approach described in Section 3.1 is used in this section to model the terrain that the vehicle is moving on as a grid of \(n \times n\) cells, which are identified (for different values of \(n\)) as follows:

\[
\begin{bmatrix}
1 & 2 & \ldots & n \\
\vdots & \vdots & \ddots & \vdots \\
(n-1)n+1 & (n-1)n+2 & \ldots & n^2
\end{bmatrix}
\]

This implies that the kinematic model \(H\) (which describes the limitations on the vehicle movements) has \(n^2\) states. It is assumed that the vehicle in each cell can only move to neighboring cells so that from each state \(i\) \((1 \leq i \leq n^2)\) of the kinematic model \(H\) at most\(^3\) 9 transitions are possible: \(i \rightarrow i-n-1, i \rightarrow i-n, i \rightarrow i-n+1, i \rightarrow i-1, i \rightarrow i, i \rightarrow i+1, i \rightarrow i+n-1, i \rightarrow i+n,\) and \(i \rightarrow i+n+1.\)

In order to show the applicability of the results to typical kinematic models on an \(n \times n\) grid, the possible transitions (movements) between neighboring cells are chosen randomly. This is accomplished by creating for each state \(i, 1 \leq i \leq n^2,\) a random binary vector of size 9, each element of which corresponds to one of the (at most) 9 possible state transitions for state \(i.\) The index of the element is the index of the state transition (in the ordering described above) and the randomly obtained states on the edges of the grid have less possible transitions.\(^3\)
binary value indicates whether the given transition is possible (value 1) or not (value 0). Matlab is used to create this random binary vector (0 and 1 are chosen with equal probability) and the resulting information is stored in a text file (in a format appropriate for the UMDES Library tool (Lafortune, 2009) which is described in more detail in the following section).

It is assumed that there are $m$ sensors in the sensor network and various values of $m$ are tried at the beginning of each simulation. The coverage of each sensor is assigned randomly by creating a rectangular coverage region for each sensor. More specifically, the top left and bottom right vertices of the coverage region are chosen with probability that is uniform in the given grid; however, sensor coverage areas are not allowed to be overlapping. Once sensor coverage areas are chosen as described above, labels are assigned to the transitions in the kinematic model $H$ to obtain automaton $G$ (as described in Section 3.1). Again, Matlab is used to create the sensor coverage area and store the resulting automaton $G$ in a text file in a format appropriate for the UMDES Library tool.

5.2. Implementation of Verification Algorithms for Tracking Problems

5.2.1. Using the UMDES Library to Verify Infinite-Step Opacity

In order to implement the verification method for infinite-step opacity (Saboori and Hadjicostis, 2009) described in Section 2.2.1, the UMDES Library is used. UMDES is a library of C routines written at the University of Michigan (Lafortune, 2009) for studying discrete event systems modelled as finite automata.

In the UMDES Library, an automaton with $N$ states is stored as a text file with extension .fsm with a header which denotes the number of states of the given automaton, followed by exactly $N$ paragraphs which describe the state transitions out of each state. As will be explained later, Matlab is used to call all UMDES Library routines and to manipulate their output.

The first step in verifying infinite-step opacity (as described in Theorem 1) requires constructing the current-state estimator. The UMDES Library provides a routine for
creating the current-state estimator for a given automaton. Note that this routine is
developed assuming that the given automaton has one (unique) initial state but one
can overcome this limitation by adding a dummy state $x$ to $G$ and by connecting it to
all states in $X_0$ using a dummy unobservable transition. It is not hard to see that the
current-state estimator $\hat{G}_{0,\text{obs}}$ constructed for this new automaton $\hat{G}$ (with state $x$
as its unique initial state) is the same\(^4\) as the current-state estimator $G_{0,\text{obs}}$ constructed
for automaton $G$ with $X_0$ as its initial state.

The next step in verifying infinite-step opacity (as described in Theorem 1) re-
quires constructing ISEs with their initial state set to $\odot(Z_0)$ for each state $Z_0$ of the
current-state estimator which contains a state in the set of secret states $S$. As there
is no support for constructing ISEs in the UMDES Library, one employs a transfor-
mation that enables the use of the routine for current-state estimation in order to
construct ISEs. Specifically, to construct the ISE associated with $G = (X, \Sigma, \delta, X_0)$
the following steps can be performed:

(i) For each $i \in X_0$, one constructs $G_i = (\{i\} \times X, \Sigma, \delta_i, \{i\} \times X_0)$ where for
$j \in X, \alpha \in \Sigma$: $\delta_i((i, j), \alpha) \equiv (i, \delta(j, \alpha))$. In words, $G_i$ is the automaton that is
obtained by annotating each state of $G$ with label $i$ and assuming that the automaton
starts from state $i$.

(ii) Then, automaton $\hat{G} = (\bigcup_{i \in X_0} X_i, \Sigma, \hat{\delta}, \odot(X_0))$ is constructed, with $X_i, i \in X_0$
denoting the set of states of $G_i$ and $\hat{\delta}((i, j), \alpha) = \delta_i((i, j), \alpha)$. In words, $\hat{G}$ is obtained
by taking the union of all $G_i, i \in X_0$, and setting its set of initial states equal to
$\odot(X_0)$.

(iii) By constructing the current-state estimator for $\hat{G}$, one can obtain the ISE for
$G$ as described earlier in this paper.

Remark 2. In Step (i), in order to annotate automaton $G$, one uses the product
routine provided in the UMDES Library and the auxiliary automaton $H_i$ depicted in
Figure 6. In Step (ii), one cannot use the union routine provided in the UMDES Library

\(^4\)Apart from the initial state which includes not only the unobservable reach of $X_0$ but also $X$.\)
to obtain $\hat{G}$. The problem is that the union routine provided by the UMDES Library acts on two non-deterministic automata $G_1$ and $G_2$, and returns the deterministic automaton $G'$ that is the determinization of the non-deterministic automaton $G_1 \cup G_2$ (constructed using the subset construction (Cassandras and Lafortune, 2008)). This implies that the output $G'$ of the union operator and $G_1 \cup G_2$ can have different sets of states. In Step (ii) of the transformation described above, however, the set of states (and non-determinism) of the automaton from the union operator needs to be intact; therefore, the union routine provided in the UMDES Library cannot be used here. This issue is solved by using Linux capabilities to manipulate files. As mentioned earlier, each $G_i$ is stored as a text file, e.g., as $G_i.fsm$. In order to construct the union $\cup G_i$ without modifying the set of states, the header from each file $G_i.fsm$ (corresponding to each $G_i$, $i \in X_0$) is removed, and then all of these files are merged into one file with its header (i.e., number of states) equal to $|X_0| \times N$ (since the number of automata of the form $G_i$ that are merged is $|X_0|$, and each $G_i$ has $N$ states). The automaton corresponding to this file is $\hat{G}$ in Step (ii) of the above transformation.

The above transformation obtains the ISE by constructing the current-state estimator for an annotated system model but, compared to the method described in the previous section (which constructs the ISE using state mappings), it is not as efficient neither in terms of memory nor in terms of computational time. One obvious reason for this is that many states in $\hat{G}$ that are not reachable from its set of initial-states are nevertheless constructed and stored as part of the algorithm. This issue was ignored for the sake of being able to employ existing routines from the UMDES Library. The reader, however, should keep in mind that the effect of this inefficiency on the time required for simulation is large and therefore, the reported times are not indicative of the actual time required for verifying infinite-step opacity (they are longer than
the actual time); instead, these timing measurements should be used as a relative measure of difference between simulations with different parameters.

The last step in verifying infinite-step opacity (as described in Theorem 1) requires checking whether the set of starting states of any of the state mappings in the constructed ISEs lies entirely within the set of secret states. This checking is accomplished using Matlab code, which is also used as a wrapper for the program. The code first creates a random kinematic model with associated sensor coverage as described in Section 5.1. The result is stored in a text file readable by UMDES Library routines as described in this section. Then, the current-state estimator and the required initial-state estimators based on the transformation procedure described in this section are constructed.

5.3. Simulation Results

The first part of the simulation (small example) studies the effect that the number of deployed sensors has on infinite-step opacity and hence indirectly solves the (minimal) sensor selection problem. It is assumed that the vehicle is moving on a $6 \times 6$ grid, and that $S = \{5, 16, 27\}$ and $X_0 = X$. The kinematic model used throughout this part of the simulation is depicted in Figure 7. It is assumed that up to seven sensors are available for deployment. The coverage of each sensor was chosen to be non-overlapping as depicted in Figure 7. One starts assuming that all sensors are selected. At each step, sensors are turned off one at a time, and one checks whether the system becomes infinite-step opaque. Sensors are turned off in the same order as their identifier with the largest one (i.e., 7) turned off first. The results are summarized in Table 1: after turning off sensors 7, 6, and 5, turning off any additional sensor violates infinite-step opacity; therefore, the solution to the minimal sensor selection problem is the set of sensors $\{1, 2, 3, 4\}$. As can be observed, the algorithm runs relatively fast.

The solution to the minimum sensor selection problem was also studied via exhaustive search over all possible sensor configurations. For this part, one starts by turning off all sensors. The proposed algorithm first enumerates all seven singleton
Figure 7: Example of a $6 \times 6$ sensor grid.
subsets of set $I = \{1, 2, 3, 4, 5, 6, 7\}$ using the function `nchoosek` in Matlab. For each such subset, the corresponding sensor is turned on and one verifies whether the system is infinite-step opaque. If the system is not infinite-step opaque, the algorithm stops and reports the chosen sensor as the solution to the minimum sensor selection problem. Otherwise, that sensor is turned off and the algorithm proceeds with the next subset. Once each sensor corresponding to a sensor subset of size 1 is turned on and the system is verified to be infinite-step opaque, the algorithm proceeds by enumerating all subsets of set $I$ of size 2, checking whether the selected subset makes the system infinite-step opaque. As soon as the algorithm finds a sensor configuration for which the system is not infinite-step opaque, the algorithm stops. The proposed algorithm stops after enumerating the set $\{1, 2, 3, 4\}$. This implies that the solution to the minimal sensor selection obtained previously is also a solution to the minimum sensor selection problem.

In the second part of the simulation, one studies the minimal solution to the sensor selection problem in Definition 2 for randomly generated grids with randomly selected sensor coverage areas that are not allowed to be overlapping. For this, one randomly creates 100 grids of size $6 \times 6$ as described in Section 5.1. Again, it is assumed that up to seven sensors can be deployed, and the sensor coverage is chosen randomly as described in Section 5.1. Table 2 describes the solution to the minimal sensor selection problem in Definition 2 for these randomly generated grids. This table should be interpreted as follows: out of 100 grids, 72 needed 3 sensors to violate

<table>
<thead>
<tr>
<th>Deployed sensors</th>
<th>1:7</th>
<th>1:6</th>
<th>1:5</th>
<th>1:4</th>
<th>1,2,3</th>
<th>1,2,4</th>
<th>1,3,4</th>
<th>2,3,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>X_{obs}</td>
<td>$</td>
<td>53</td>
<td>37</td>
<td>17</td>
<td>11</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>max $</td>
<td>X_{\infty,obs}</td>
<td>$</td>
<td>2451</td>
<td>915</td>
<td>322</td>
<td>193</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Time (Sec)</td>
<td>3.55</td>
<td>1.9</td>
<td>2.4</td>
<td>3.8</td>
<td>3.4</td>
<td>3.8</td>
<td>3.4</td>
<td>3.17</td>
</tr>
<tr>
<td>Infinite-Step Opaque?</td>
<td>No</td>
<td>No</td>
<td>2.4</td>
<td>3.8</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: Infinite-step opacity for the $6 \times 6$ grid in Figure 7. $|X_{obs}|$ denotes the number of current-state estimator states and max $|X_{\infty,obs}|$ denotes the maximum number of ISE’s states. Here $i : j$ means sensors $i, i + 1, \ldots, j$ are deployed.

5The order follows the order of the resulting vector from function `nchoosek`.
infinite-step opacity, 20 needed 4 sensors, and so forth. As can be seen from this table, on average 3 sensors suffice to violate infinite-step opacity.

The third part of the simulation studies the effect of the grid size on the number of states of the generated ISE’s. For this, one doubled and tripled the size of the grid. More specifically, \( n = 6, 8, 10 \) were used, each time choosing \( S \) randomly such that it includes 3% of system states and also spreads over the grid. One also assumed that \( X_0 = X \). For each value of \( n \), five sensors were considered with location and coverage chosen randomly (as described earlier). The results are summarized in Table 3. As the number of cells in the grid triples (from 36 to 100), the size of constructed ISE’s also triples. Also observe that (not surprisingly) the algorithm takes a relatively long time for large size grids.

### 5.4. Verification of Infinite-Step Opacity for a Fixed Set of Secret States

In this section a reduced complexity method is developed for verifying infinite-step opacity for a specific set of secret states \( S \). Verifying infinite-step opacity using a bank of initial-state estimators (ISE) as described in Section 2.2.1 is not tailored to a particular \( S \) (i.e., the ISE can be used for any set of secret states, even if the set of secret states is modified or is to be designed). If \( S \) is invariant (fixed over time), one can potentially simplify the verification method since the exact set of possible system states at a particular point in time is not needed; instead, one only requires knowledge of whether this set consists exclusively of secret states. In this section, it is shown that

<table>
<thead>
<tr>
<th>Grid size</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 6 \times 6 )</td>
</tr>
<tr>
<td>( 8 \times 8 )</td>
</tr>
<tr>
<td>( 10 \times 10 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of current-state estimator states</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
</tr>
<tr>
<td>44</td>
</tr>
<tr>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of ISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Number of ISE’s states</th>
</tr>
</thead>
<tbody>
<tr>
<td>2243</td>
</tr>
<tr>
<td>1935</td>
</tr>
<tr>
<td>5070</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
</tr>
<tr>
<td>30.5</td>
</tr>
<tr>
<td>1048</td>
</tr>
</tbody>
</table>

Table 2: Solution to the minimal sensor selection problem for 100 randomly generated \( 6 \times 6 \) grids with seven available sensors.

Table 3: Number of ISE’s states as a function of grid size.
for a given (invariant) set of secret states $S$ the complexity of the verification method can be reduced from exponential in the square of the number of states ($O(2^{N^2})$) to exponential in the number of states ($O(8^N)$), where $N = |X|$ denotes the number of states of the underlying automaton $G$. Note that the price paid for this reduction is that each set of secret states $S$ would require separate verification.

Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ with set of states $X = \{0, 1, \ldots, N-1\}$, a natural projection map $P$ with respect to the set of observable events $\Sigma_{\text{obs}}$ ($\Sigma_{\text{obs}} \subseteq \Sigma$), and a set of secret states $S \subseteq X$. Note that in order to verify infinite-step opacity, using the approach based on a bank of ISEs described in Section 2.2.1, for each set of potential initial states (indicated by a corresponding state of the current-state estimator), one only needs to know whether (following a sequence of observations $\omega \in \Sigma^*_\text{obs}$), a possible current state is reachable from secret initial states (in $S$) or not (outside $S$) (Saboori and Hadjicostis, 2008). Therefore, instead of associating a set of starting states with each ending state (as done with state mappings), one can simply assign to each ending state one of four labels to indicate whether this ending state is reachable (i) exclusively from secret initial states (label $S$), (ii) exclusively from non-secret ones (label $NS$), (iii) from a mixture of both types of states (label $M$), or (iv) not reachable from any initial state (label $NR$). One can capture this via a set of $N$ pairs of the type $(Y(i), i)$ where $i \in X$ and $N = |X|$. This set of pairs is called a state-status mapping $q$ and is of the form

$$q = \{(Y(0), 0), (Y(1), 1), \ldots, (Y(N-1), N-1) | Y(i) \in \{S, NS, M, NR\} \forall i \in X\}.$$  

The set of all state-status mappings is denoted by $\{S, NS, M, NR\}^X \equiv \Delta^X$ and has cardinality $4^N$.

Each time a new observation is made, the label $Y$ associated with (ending) states in the pairs of state-status mapping $q$ can be easily propagated along with the ending state estimates, according to the following rules: (i) When composing the current state-status mapping with the state mapping induced by the new observation, if two
or more (ending) state estimates with two different labels from the set \( \{S, NS, M\} \) merge to an identical ending state estimate, the label \((M)\) is assigned to this new (ending) state estimate; this indicates that the new ending state can be reached from at least one secret and at least one non-secret initial state. (ii) If, on the other hand, the merging involves one or more states with a single label from the set \( \{S, NS, M\} \) and zero or more states with label \( NR \), then the label \( S, NS, \) or \( M \) propagates intact. (iii) Finally, if the last observation cannot occur from any of the ending states or the merging involves only states with label \( NR \), the label \( NR \) is assigned to it. The above idea was formulated in (Saboori and Hadjicostis, 2008) via the composition operator \( \otimes: \Delta^X \times 2^{X^2} \rightarrow \Delta^X \) defined for a state-status mapping \( q \in \Delta^X \) and a state mapping \( m \in 2^{X^2} \) as

\[
q \otimes m = \{(Y(i), i) | i \in X\},
\]

where

\[
Y(i) = \begin{cases} 
S, & \text{if (i) } \exists i' \in X \{ (i', i) \in m \}, \text{ and (ii) } \forall i' \in X \{ (i', i) \in m \Rightarrow \} \\
& \{ (S, i') \in q \text{ or } (NR, i') \in q \}, \\
NS, & \text{if (i) } \exists i' \in X \{ (i', i) \in m \}, \text{ and (ii) } \forall i' \in X \{ (i', i) \in m \Rightarrow \} \\
& \{ (NS, i') \in q \text{ or } (NR, i') \in q \}, \\
M, & \text{if either: (i) } \exists i' \in X \{ (i', i) \in m \} \text{ and } (M, i') \in q \} \text{ or} \\
& \exists i', i'' \in X \{ (i', i), (i'', i) \in m \} \text{ and } (S, i'), (NS, i'') \in q \}, \\
NR, & \text{if either: (i) } \not\exists i' \in X \{ (i', i) \in m \} \text{ or } \\
& \forall i' \in X \{ (i', i) \in m \Rightarrow (NR, i') \in q \}.
\end{cases}
\]

In this way a deterministic finite automaton \( G_{\text{verifier}} \), called \( \text{verifier} \), can be constructed. The verifier was introduced in (Saboori and Hadjicostis, 2008) for verifying the notion of initial-state opacity. Under a sequence of observations \( \omega \in \Sigma_{\text{obs}}^* \), \( \omega \neq \epsilon \), the verifier reaches a state associated with a state-status mapping for which each state \( i \) is associated with a unique label \( Y(i) \) from the set \( \Delta \equiv \{S, NS, M, NR\} \). Label
\(\mathbb{Y}(i)\) indicates whether \(i\) is reachable in \(G\) via sequences of events (with projection \(\omega\)) that start exclusively from secret initial states (\(\mathbb{Y}(i) = S\)), or exclusively from non-secret initial states (\(\mathbb{Y}(i) = NS\)), or from a mixture of secret and non-secret initial states (\(\mathbb{Y}(i) = M\)), or not reachable at all (\(\mathbb{Y}(i) = NR\)). The state-status mapping associated with the verifier initial state is \(q_0 = \{(\mathbb{Y}_0(i), i) | i \in X\}\) where

\[
\mathbb{Y}_0(i) = \begin{cases} 
S, & \text{for } i \in X_0 \cap S, \\
NS, & \text{for } i \in X_0 - S, \\
NR, & \text{for } i \in X - X_0.
\end{cases}
\]

Upon observing an event \(\alpha \in \Sigma_{obs}\), both components in each pair of the state-status mapping associated with the verifier current state need to be updated; this is accomplished by defining the next state under input \(\alpha\) to be the state associated with state-status mapping \(q_0 \otimes M(\alpha)\), where \(M(\alpha)\) is the state mapping induced by observation \(\alpha\). The construction of the deterministic automaton \(G_{\text{verifier}}\) in (Saboori and Hadjicostis, 2008) continues in this way and is stated formally below for completeness.

**Definition 4 (Verifier).** Consider a non-deterministic finite automaton \(G = (X, \Sigma, \delta, X_0)\), with set of states \(X = \{0, 1, \ldots, N - 1\}\), a natural projection map \(P\) with respect to the set of observable events \(\Sigma_{obs}\) (\(\Sigma_{obs} \subseteq \Sigma\)), and a set of secret states \(S \subseteq X\). The verifier is defined as the deterministic finite automaton \(G_{\text{verifier}} = AC(\Delta^X, \Sigma_{obs}, \delta_{\text{verifier}}, X_{\text{verifier},0})\) with set of labels \(\Delta = \{S, NS, M, NR\}\), set of states \(\Delta^X\), event set \(\Sigma_{obs}\), initial state \(X_{\text{verifier},0} = \{(\mathbb{Y}_0(i), i) | i \in X\}\), and state transition function \(\delta_{\text{verifier}} : \Delta^X \times \Sigma_{obs} \rightarrow \Delta^X\) defined for \(\alpha \in \Sigma_{obs}\) as

\[
\delta_{\text{verifier}}(q, \alpha) := q \otimes M(\alpha).
\]

Recall that \(AC\) denotes the accessible part of this automaton starting from state \(X_{\text{verifier},0}\). If \(X_{\text{verifier}} \subseteq \Delta^X\) is used to denote the set of verifier states reachable from the verifier initial state \(X_{\text{verifier},0}\) under the state transition mapping \(\delta_{\text{verifier}}\).
then $G_{\text{verifier}} = (X_{\text{verifier}}, \Sigma_{\text{obs}}, \delta_{\text{verifier}}, X_{\text{verifier},0})$. 

**Remark 3.** In (Saboori and Hadjicostis, 2008), it is shown that the verifier state $q$ that is reachable from the verifier initial state $X_{\text{verifier},0}$ via a finite-length string $\omega \in \Sigma_{\text{obs}}^*, \omega \neq \epsilon$, is associated with a state status mapping $q \in 2^{\Delta \times X}$ such that:

(i) $(Y, i) \in q$ (for some unique label $Y \in \{S, NS, M\}$) if and only if system state $i$ is reachable in system $G$ from some initial state in $X_0$ via a string $s$ such that $P(s) = \omega$;

(ii) for each $(Y, i) \in q$, $Y = S$ ($Y = NS$) if and only if all of the system initial states from which system state $i$ is reachable via a string $s$, such that $P(s) = \omega$, are secret (non-secret) states; (iii) for each $(Y, i) \in q$, $Y = M$ if and only if there exist at least one secret and at least one non-secret initial state from which system state $i$ is reachable via a string $s$, such that $P(s) = \omega$; finally, $(NS, i) \in q$ if and only if there is no initial state from which system state $i$ is reachable via a string $s$ such that $P(s) = \omega$.

In order to verify that a system is infinite-step opaque, it is required to verify that at any point during the observation process, any possible future current-state estimate associated with a sequence of observations after that point is reachable from both a secret state and a non-secret state in the set of possible states at that point. Also, Remark 3 implies that the verifier has all the information concerning the secrecy of the originating state (in the set of initial states $X_0$) for any sequence in the system; therefore, by taking $X_0$ to be the set of possible states at this point in time, one can verify the secrecy of the system state at this point (and all other points in the observation process which result in the same system current-state as this point of interest). Since for infinite-step opacity to hold this needs to be verified for all points along the observation process, one can construct a bank of verifiers (instead of a bank of ISEs) associated with each possible system current-state estimate, such that for each verifier in this bank, the system set of initial states equals the associated set of current-state estimates. The following definition is needed in order to formalize this with a theorem.
Definition 5. Given a set of states $X$ and a set of labels $\Delta = \{S, NS, M, NR\}$, a state-status mapping $q \in \Delta^X$ is $NR$-certain if
\[
\forall i \in X \{ Y(i) = NR \}.
\]
A state-status mapping that is not $NR$-certain is $L$-certain ($L \in \{N, NS, M\}$) if
\[
\forall i \in X \{ Y(i) = L \text{ or } Y(i) = NR \}.
\]

Theorem 2. Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map $P$ with respect to the set of observable events $\Sigma_{obs}$ ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$. For every set of current-state estimates $Z_n$ associated with a state of its current-state estimator $G_{0,obs}$, construct the verifier $G_{verifier}^{(n)} = AC(\Delta^X, \Sigma_{obs}, \delta_{verifier}^{(n)}, X_{verifier,0}^{(n)}) \equiv (X_{verifier}^{(n)}, \Sigma_{obs}, \delta_{verifier}^{(n)}, X_{verifier,0}^{(n)})$ with set of labels $\Delta = \{S, NS, M, NR\}$, by setting
\[
X_{verifier,0}^{(n)} = \{(Y, i) | (Y = S, i \in S \cap Z_n) \lor (Y = NS, i \in Z_n - S) \lor (Y = NR, i \in X - Z_n)\}.
\]
Then, $DES G$ is $(S, P, \infty)$-opaque if and only if
\[
\forall n, \forall q \in X_{verifier}^{(n)} : q \text{ is not } S\text{-certain}, \quad (2)
\]
where $X_{verifier}^{(n)}$ is the set of states in $G_{verifier}^{(n)}$ that is reachable from its initial state $X_{verifier,0}^{(n)}$.

Proof. First observe that in an infinite-step opaque system, at any point during the observation process, any possible future current-state estimate associated with a sequence of observations after that point is reachable from both a secret state and a non-secret state in the set of possible states at that point. In other words, after observing any sequence of observations $\omega \in \Sigma_{obs}^*$ with the associated current-state estimate $Z_n$, for any string $t \in \Sigma^*$ that originates from a secret state $i$ in the set
there exists another string \( s \in \Sigma^* \) that originates from a non-secret state \( j \) in the set \( Z_n \) and has the same projection as \( t \), i.e., \( P(s) = P(t) \equiv \omega \Omega \) for some \( \Omega \in \Sigma_{obs}^* \). By Remark 3, each state of the verifier \( G_{verifier} \) with initial state set to be
\[
\{(Y, i) |(Y = S, i \in S \cap Z_n) \lor (Y = NS, i \in Z_n - S) \lor (Y = NR, i \in X - Z_n)\},
\]
the associated state status mapping captures the secrecy of the origin of any sequence in the system in the set \( Z_n \), i.e., it captures whether the sequence is reachable from a secret state in the set \( Z_n \) or not. Therefore, existence of a verifier state (reachable via the string \( \Omega \)) for which the associated state status mapping is \( S \)-certain is equivalent to the existence of a string \( \Omega \in \Sigma_{obs}^* \), \( \Omega \neq \epsilon \), such that, there does not exist any string \( t \in \Sigma^* \) that originates from a non-secret state \( j \) in the set \( Z_n \) and satisfies \( P(t) = \Omega \). Finally, note that any sequence in the system that generates the sequence of observations \( \omega \Omega \) passes through the set of secret states when the sequence of observations \( \omega \omega \) was observed. This violates infinite-step opacity and completes the proof. ■

The verifier introduced in Definition 4 has state complexity \( O(4^N) \) where \( N = |X| \) denotes the number of states of the underlying automaton \( G \). Checking for infinite-step opacity using Theorem 2 requires at most \( 2^N \) such verifiers and as a result, has space and time complexity \( O(8^N) \).

6. Conclusions

This paper studied the application of the notion of infinite-step opacity to coverage analysis of mobile agent trajectory. Infinite-step opacity requires that the entrance of the system to a set of secret states \( S \), at any time during the operation of the system, remain opaque to outsiders. Existing tools were employed and appropriate transformations were devised to verify infinite-step opacity using a current-state estimator and a bank of initial-state estimators. The proposed algorithm was shown to be effective for relatively small values of \( n \). The minimal sensor selection problem was also studied by obtaining a minimal set of sensors that must be used to ensure that
the system is not infinite-step opaque and such that, by removing any sensor in this minimal set, the system becomes infinite-step opaque. Key to these developments was the observation that lack of infinite step opacity is a mask monotonic property with respect to a given set of sensors.

Interesting future work includes tracking problems involving two (or more) mobile agents with possibly different kinematic models or kinematic models with common patterns (in case where the mobile agents share group formations) (Reid, 1979; Bar-Shalom et al., 1995; Yang and Sikdar, 2003). In this case, sensor readings can be triggered by any of the vehicles which adds uncertainty to the problem and can potentially be handled by using projection mappings more general than natural projection. The study of this problem and variations of it, along with potential applications of modular verification techniques Saboori and Hadjicostis (2010a) will be part of future work. Another direction for future research is to implement (without necessarily using existing tools) the reduced complexity algorithm for verifying infinite-step opacity that was introduced in this paper.

Finally, it would also be interesting to use techniques from supervisory control (Wonham, 2009) to design minimally restrictive supervisors that allow a given mobile agent to choose its movements in ways that ensure infinite-step opacity (while enabling the agent to move between different locations on the grid).

References


URL http://www.eecs.umich.edu/umdes/toolboxes.html


Wonham, W., 2009. Supervisory control of discrete event systems. Systems and Control Group, Department of Electrical and Computer Engineering, University of Toronto. Available at www.utoronto.ca/DES.

