

Optimal sensor deception in stochastic environments with partial observability to mislead a robot to a decoy goal

Hazhar Rahmani Mukulika Ghosh Syed Md Hasnayeen

Abstract—Deception is a common strategy adapted by autonomous systems in adversarial settings. Existing deception methods primarily focus on increasing opacity or misdirecting agents away from their goal or itinerary. In this work, we propose a deception problem aiming to mislead the robot towards a decoy goal through altering sensor events under a constrained budget of alteration. The environment along with the robot’s interaction with it is modeled as a Partially Observable Markov Decision Process (POMDP), and the robot’s action selection is governed by a Finite State Controller (FSC). Given a constrained budget for sensor event modifications, the objective is to compute a sensor alteration that maximizes the probability of the robot reaching a decoy goal. We establish the computational hardness of the problem by a reduction from the 0/1 Knapsack problem and propose a Mixed Integer Linear Programming (MILP) formulation to compute optimal deception strategies. We show the efficacy of our MILP formulation via a sequence of experiments.

I. INTRODUCTION

Deception is a common adversarial strategy that can involve concealing intent from adversarial agents [12], [15] or misleading them through sensor manipulation [17], [21]. This paper focuses on the latter, where an autonomous agent or robot is deliberately misled toward predetermined decoy goals via systematic sensor alterations. Since modifying sensor information incurs a cost, the system optimizes deception within a constrained budget. Unlike existing approaches that focus on optimizing the placement of decoy goals or honeypots [2], [14], [18], our approach assumes a fixed set of decoy goals and achieves deception by optimally swapping sensor observations (readings). This method has broad applications in autonomous system planning, including security applications and adversarial environments in multi-agent systems. For example, consider a strategic defense scenario in an environment shown in Fig. 1. An intruder ground robot, attempts to infiltrate a protected area located in upper right cell, while avoiding detection. The environment contains 6 beacons, S_1 to S_6 , producing distinct colors, which the ground robot relies on for localization and planning to move toward the protected area. The robot dynamic has stochastically, and so, when it does one of the actions North, East, South, and West, its actuators guarantee with a probability less than 1 that they will move the robot to the intended cell. The defense system knows the strategy the robot uses for action selection. Accordingly, it alters the beacon identities by spoofing them, effectively misdirecting the robot away from the ammunition area and towards the

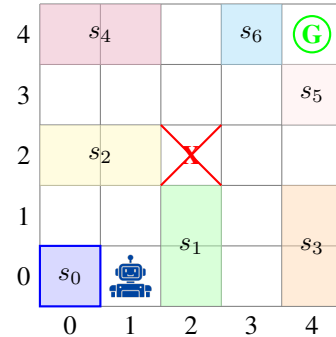


Fig. 1: An example of sensor deception. The agent at bottom left cell is tasked to navigate to the goal at the top right cell. Because of photoelasticity in the robot’s dynamic, the current state, the position of the robot, is not observable to the robot. The range sensors provide partial observability. The system will alter the sensors to mislead the agent to the decoy goal state in the middle cell containing the security.

middle cell containing a guard tower. This spoofing is not online and instead is offline and it is performed only one time, before the agent’s execution. It can simply swap the colors produced by different beacons. The cost of beacon spoofing can include energy expenditure or the risk posed to authorized agents, and hence, there is a limited budget for spoofing. In this paper, we answer the question that in this kind of scenarios, how we alter or swap the colors produced by some beacons to maximize the probability of directing the intruder toward the decoy goal. Our problem can also help identify vulnerabilities in a sensor network, where a malicious agent manipulates sensors to mislead the system to an unsafe or hazardous state.

In this paper, we consider the sensor deception problem in a stochastic environment by modeling the interaction between an autonomous agent (or robot) and its environment as a Partially Observable Markov Decision Process (POMDP). The robot lacks direct knowledge of its true state and relies on sensor-generated observations. Those observations can be strategically altered by an agent at a cost. The robot selects actions based on its current observations and memory, which we model as a Finite State Controller (FSC) [1]. The objective is to maximize the probability of misleading the robot to the predetermined decoy goal when it has limited capability of modifying the sensor readings. We establish the computational hardness of our problem by a reduction from the 0/1 Knapsack problem. Furthermore, we formulate the optimal sensor deception problem as a Mixed Integer

The authors are with the Dept. of Computer Science, Missouri State University, Springfield, MO, USA. {hrahmani, mghosh, sh3739s}@missouristate.edu

Linear Programming (MILP) model, offering a structured and scalable solution for synthesizing deception strategies that achieve optimal cost-efficiency.

The contributions of this paper are summarized as follows:

- A novel deception problem formulation, which aims to compute a sensor alteration maximizing the probability of misdirecting a robot to a predetermined decoy goal, under a limited cost budget of sensor alteration.
- The NP-hardness of our sensor alteration problem, proved by reduction from Knapsack problem.
- A scalable MILP-based solution of the sensor alteration problem.

After discussing related work in Section II, we provide basic definitions and present our problem formulation in Section III. In Section IV, we present verification algorithm. In Section V, we present our NP-hardness result, and in Section VI, we provide our MILP formulation. Section VII reports results of our experiments, and Section VIII, we conclude the paper and draw future research directions.

II. RELATED WORK

Deception of adversary or opponent by altering sensor information has been achieved in form of opacity [7], [9] and deceptive control [18], [22]. Deceptive control is implemented through adding, removal or altering sensors or observations. In this paper, we focus on deception planning. Optimal placement of decoy targets is considered in [14], [18] with linear program solution to optimize deception with [14] utilizing probabilistic attack graphs compared to deterministic one in [18]. Goal obfuscation considers hiding true goal among fake goals to implement deception [4], [12]. Savas et. al. use Linear Programming to maximize adversary teams' deception about its true goal while ensuring the highest probability of achieving its true goal in [20]. The observer's predictions are formulated as probability distributions over true goals, guided by the maximum entropy principle. Game theory based approaches have been used in deploying decoys for deception in hypergames [10], [11], [13]. Unlike the approaches mentioned above, we model a sensor deception by the alterations of sensors to lead the agent to a finite set of predetermined decoy targets while maintaining the cost incurred by sensor alterations under a limited budget.

Markov Decision Process (MDP) has been widely used in deceptive planning [5], [8], [15], [20]. Karabag et al. [8] explore deception by an agent through partial observability of the supervisor, modeling the environment as an MDP. The agent optimizes the generation of a deceptive policy, quantifying deception using Kullback-Leibler (KL) divergence to ensure that the agent's actions remain indistinguishable from the supervisor's expectations while achieving its covert objective. In [19], the method is extended to team deception, where each agent is modeled as an MDP, and a centralized deceptive policy synthesis is applied while maintaining decentralized execution. Ma et al. [15] investigate covert deception planning in a stochastic environment, modeled as an MDP, where the agent aims to maximize the discounted

total reward while maintaining covert behavior. They demonstrate that finite-memory policies can outperform Markovian policies in this constrained MDP framework and develop a primal-dual gradient-based method for synthesizing optimal and covert Markov policies. In our approach, we model the interaction of the deceived agent with the environment as partially observable Markov Decision process (POMDP) and the selection of action by the agent as finite state controller.

Another closely related work by Meira-Góes et al. in [16] investigates sensor manipulation in a stochastic environment by modeling the system as a probabilistic automaton. The authors leverage concepts from stochastic games to synthesize an attack policy that maximizes the likelihood of the system being in an unsafe state. In [17], the authors extend their work by modeling the interaction between the attacker and the system as an MDP, and also incorporating penalties into the attack model as a second problem. While these work maximizes the unsafety of the system, our proposed work focuses on sensor manipulation to maximize the likelihood of the agent landing in decoy state.

III. DEFINITIONS AND PROBLEM STATEMENT

To model the interaction of the robot with the environment we use a discrete structure defined as follows.

Definition 1: A partially observable Markov decision process (POMDP) is a tuple $\mathcal{M} = (S, A, \mathbf{P}, s_0, \Omega, O)$ in which

- S is a finite set of states,
- A is a finite set of actions,
- $\mathbf{P} : S \times A \times S \rightarrow [0, 1]$ is a transition probability function such that for each $s, s' \in S, a \in A, \mathbf{P}(s, a, s')$ is the probability that the system transitions to state s' by doing action a at state s , and it holds that $\sum_{s' \in S} \mathbf{P}(s, a, s') = 1$ for all $s \in S$ and $a \in A$,
- s_0 is the initial state,
- Ω is a finite set of observations,
- $O : S \rightarrow \Omega$ is the observation function such that for each state $s \in S, O(s)$ is the observation the system emits when it enters s .

A POMDP defines how the state of the system changes in response to the robot's actions. At each step, the robot receives an observation produced by the system based on the current state. Since the same observation could be produced by multiple states, the robot cannot directly observe the current state. The robot's action selection process is governed by a finite-state controller (FSC), formally defined as follows.

Definition 2: A finite-state controller for a POMDP $\mathcal{M} = (S, A, \mathbf{P}, s_0, \Omega, O)$ is a tuple $\mathcal{C} = (N, n_0, \gamma, \delta)$ in which

- N is a finite set of memory nodes,
- n_0 is the initial node,
- $\gamma : N \times \Omega \rightarrow A$ is the action selection function where for each node $n \in N$ and observation $o \in \Omega$, it chooses an action $\gamma(n, o)$ when the controller is in state n and the robot has perceived an observation o ,
- $\delta : N \times \Omega \rightarrow N$ is the memory update function, telling the controller to transition to node $\delta(n, o)$ by receiving observation o at node n .

The robot uses the finite-state controller $\mathcal{C} = (N, n_0, \gamma, \delta)$ to choose actions at every time-step. At the initial time step 0, the POMDP is in state s_0 and the controller has memory n_0 . Because of partial observability, the robot generally does not explicitly know the current state of the POMDP. However, based on the perceived observation associated with the current state, it can reduce uncertainty about the state of the POMDP. At each time-step $k \geq 0$, the controller generates action $a_k = \gamma(n_k, O(s_k))$, which must be executed by the robot. Upon executing this action, the POMDP transitions from state s_k to s_{k+1} at time step $k + 1$. When the POMDP enters s_{k+1} , an observation $O(s_{k+1})$ is emitted. State s_{k+1} is produced randomly from every state s' for which $\mathbf{P}(s_k, a_k, s') > 0$.

In this paper, we study a problem in which the sensors have been attacked by an adversarial agent, causing them to produce incorrect observations. The purpose of the sensor attack is to mislead the robot into a decoy goal. We consider only *alteration* attacks.

Definition 3: [Observation alteration] An observation alteration is a function $\alpha : \Omega \rightarrow \Omega$ where for each $o \in \Omega$, the robot receives observation $\alpha(o)$ instead of o whenever it is supposed to receive o .

Consider that the sensor attack is in a sense an offline attack rather than an online attack, meaning that it is performed before any system execution. Each attack incurs a cost, defined by a function: Given the observation alteration cost function $c : \Omega \times \Omega \rightarrow \mathbb{R}_{\geq 0}$, where for each $o_1, o_2 \in \Omega$, $c(o_1, o_2)$ is the cost of altering observation o_1 to observation o_2 .

Let \mathbf{A} be the set of all sensor alterations defined over Ω . The cost associated with a sensor alteration is identified using function $C : \mathbf{A} \rightarrow \mathbb{R}_{\geq 0}$, assigning cost $C(\alpha) = \sum_{o \in \Omega} c(o, \alpha(o))$ to sensor alteration α .

The adversary's objective is to make an observation alteration that maximizes the probability of misleading the agent to the decoy while ensuring the cost of the observation alteration is no greater than a cost budget.

Problem: Optimal Sensor Alteration for Decoy Goal Misleading (OSA_DGM)

Input: A POMDP $\mathcal{M} = (S, \alpha, \mathbf{P}, s_0, \Omega, O)$, a decoy goal $S_D \subseteq S$, a reference controller $\mathcal{C} = (N, n_0, \gamma, \delta)$, a cost alteration function c , and a cost budget $B \in \mathbb{R}_{\geq 0}$.

Output: A sensor alteration α with cost at most B for which, $\mathbf{Pr}_{\text{reach}}(S_D, \alpha)$ is maximum.

IV. MISLEADING PROBABILITY OF A SENSOR ALTERATION

In this section, we introduce a product automaton construction, which can be used for a hardness result presented in the next section, and for computing the probability of misleading the robot to a decoy goal on a sensor alteration.

Definition 4: The product of POMDP $\mathcal{M} = (S, A, \mathbf{P}, s_0, \Omega, O)$ and finite-state controller $\mathcal{C} =$

(N, n_0, γ, δ) under sensor alteration $\alpha : \Omega \rightarrow \Omega$ for decoy goal $S_D \subseteq S$ is a tuple $\mathcal{P} = (Q, q_0, \mathbf{T}, Q_G)$ in which

- $Q = S \times N$ is the state space,
- $q_0 = (s_0, n_0)$ is the initial state,
- $Q_D = S_D \times N$ is the set of goal states, and
- $\mathbf{T} : Q \times Q \rightarrow [0, 1]$ is the transition function such that for each states $(s, n), (s', n') \in Q$,

$$\mathbf{T}((s, n), (s', n')) = \begin{cases} \mathbf{P}(s, s', a) & \text{if } \delta(n, \alpha(O(s))) = n' \\ & \text{and} \\ & \gamma(n, \alpha(O(s))) = a \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Note that this automaton is, in fact, a *goal Markov chain*—a Markov chain with the set of goal states Q_D . Each state of it is a tuple (s, n) in which s is the current state of the world and n is current memory node of the controller. The system produces observation $O(s)$ and the attacker alters this sensor reading to $\alpha(O(s))$. The robot receives the modified observation $\alpha(O(s))$ and uses it to select action $a = \gamma(n, \alpha(O(s)))$, generated by the controller. The controller then transitions to node $n' = \delta(n, \alpha(O(s)))$, and the POMDP transitions from state s to a state s' stochastically, based on $\mathbf{P}(s, n, \cdot)$.

To compute the probability of reaching the decoy, one can introduce a variable z_q for each $q \in Q$, and set $z_q = 1$ if $q \in Q_D$, and otherwise,

$$z_q = \sum_{q' \in Q} \mathbf{T}(q, q') z_{q'}.$$

By solving this Bellman equation using standard methods, such as the method in Chapter 10 of [3], the probability of misleading the robot to the decoy goal is given by z_{q_0} .

V. HARDNESS RESULTS

In this section, we present our hardness result.

First, we consider the decision variant of our problem.

Decision Problem: Optimal Sensor Alteration for Decoy Goal Misleading (OSA_DGM-DEC)

Input: A POMDP $\mathcal{M} = (S, A, \mathbf{P}, s_0, \Omega, O)$, a decoy goal $S_D \subseteq S$, a reference controller $\mathcal{C} = (N, n_0, \gamma, \delta)$, a cost alteration function c , a cost budget $B \in \mathbb{R}_{\geq 0}$, and a real number $r \in \mathbb{R}_{\geq 0}$.

Output: Yes, if there exists a sensor alteration α such that $C(\alpha) \leq B$ and $\mathbf{Pr}_{\text{reach}}(S_D, \alpha) \geq r$; and No, otherwise.

Lemma 1: OSA_DGM-DEC \in NP.

Proof: Let $x : \langle \mathcal{M} := (S, A, \mathbf{P}, s_0, \Omega, O), \mathcal{C} := (N, n_0, \gamma, \delta), S_D, B, r \rangle$ be an instance of OSA_DGM-DEC. We assume the sensor alteration α is given as the certificate. We need to prove that in polynomial time to the size of x we can verify if $C(\alpha) \leq B$ and $\mathbf{Pr}_{\text{reach}}(S_D, \alpha) \geq r$.

We construct \mathcal{P} using the construction in Definition 4. The running time of this construction is $\mathcal{O}(|S||N||A||\Omega| + |S|^2|N|^2)$, which is polynomial to the size x . Then, it takes a

time polynomial to the size of \mathcal{P} , to compute the probability of reaching Q_G in \mathcal{P} , which represented $\mathbf{Pr}_{\text{reach}}(S_D, \alpha)$. Trivially, checking whether $\mathbf{Pr}_{\text{reach}}(S_D, \alpha) \geq r$ and $C(\alpha) \leq B$ takes a polynomial time, and this completes the proof. ■

Next, we consider a well known problem.

Decision Problem: 0/1 Knapsack Problem (0/1-KNAPSACK-DEC)

Input: n items with weights $W = [w_1, w_2, \dots, w_n]$ and values $V = [v_1, v_2, \dots, v_n]$, a knapsack with capacity P , a positive real number $L \geq 0$.

Output: Yes if there is a set $I \subseteq \{1, 2, \dots, n\}$ such that $\sum_{i \in I} w_i \leq P$ and $\sum_{i \in I} v_i \geq L$, and No otherwise.

In words, this problem asks whether the knapsack can be filled, either fully or partially, with a subset of items whose total weight is at least L .

Next, we show that our problem is computationally hard.

Theorem 1: OSA_DGM-DEC \in NP-hard.

Proof: By reduction from the 0/1 Knapsack problem.

Given an instance

$$x = \langle W := [w_1, \dots, w_n], V := [v_1, \dots, v_n], P, L \rangle$$

of the 0/1-KNAPSACK-DEC problem, we construct an instance

$$y = \langle \mathcal{M}, S_D := \{s_{\perp}\}, \mathcal{C}, c, B, r \rangle$$

of OSA_DGM-DEC in which for $\mathcal{M} = (S, A, \mathbf{P}, s_0, \Omega, O)$,

- $S = \{s_0\} \cup \{s_i \mid i \in \{1, \dots, n\}\} \cup \{s_{\clubsuit}\} \cup \{s_{\top}\} \cup \{s_{\perp}\}$
- $A = \{a, b\}$
- $\Omega = \{o_0\} \cup \{o_i \mid i \in \{1, \dots, n\}\} \cup \{o_{\clubsuit}\} \cup \{o_{\top}\} \cup \{o_{\perp}\}$
- for each $j \in \{0\} \cup \{1, \dots, n\} \cup \{\perp, \top, \clubsuit\}$,

$$O(s_j) = o_j,$$

- for each $s, s' \in S$ and $t \in A$,

$$\mathbf{P}(s, t, s') = \begin{cases} \frac{v_i}{2 \sum_{j=1}^n v_j} & \text{if } (s = s_0, t = a, s' = s_i) \text{ for } 1 \leq i \leq n \\ \frac{1}{2} & \text{if } (s = s_0, t = a, s' = s_{\clubsuit}) \\ 1 & \text{if } (s = s_0, t = b, s' = s_0) \\ 1 & \text{if } (s = s_i, t = a, s' = s_{\top}) \text{ for } 1 \leq i \leq n \\ 1 & \text{if } (s = s_i, t = b, s' = s_{\perp}) \text{ for } 1 \leq i \leq n \\ 1 & \text{if } (s = s_{\clubsuit}, t = b, s' = s_{\top}) \\ 1 & \text{if } (s = s_{\clubsuit}, t = a, s' = s_{\perp}) \\ 1 & \text{if } (s = s_{\top}, s' = s_{\top}, t \in A) \\ 1 & \text{if } (s = s_{\perp}, s' = s_{\perp}, t \in A) \\ 0 & \text{otherwise} \end{cases}$$

and for $\mathcal{C} = (N, n_0, \gamma, \delta)$, we have

- $N = \{n_0, n_1, n_2\}$,

- For each $n \in N$ and $o \in \Omega$,

$$\gamma(n, o) = \begin{cases} a & \text{if } n = n_0, o = o_0 \\ a & \text{if } n = n_1, o = o_i \text{ for } 1 \leq i \leq n \\ b & \text{otherwise.} \end{cases}$$

- δ is defined as follows: For each $n \in N$ and $o \in \Omega$,

- $\delta(n, o) = n_1$ if n_0 and $o = o_0$,
- and otherwise, $\delta(n, o) = n$

- for each $n \in N$ and $o \in \Omega$,

$$\delta(n, o) = \begin{cases} n_1 & \text{if } n = n_0, o = o_0, \\ n_2 & \text{otherwise} \end{cases}$$

$$S_D = \{s_{\perp}\},$$

we define c such that for each $o, o' \in \Omega$,

$$c(o, o') = \begin{cases} 0 & \text{if } o = o' \\ w_i & \text{if } o = o_i \text{ for } 1 \leq i \leq n, o' = o_{\clubsuit} \\ \infty & \text{otherwise} \end{cases}$$

and we set $B = P$ and $r = \frac{L}{2 \sum_{i=1}^n v_i}$.

Clearly, this reduction takes a polynomial time, and hence, we only need to show that the reduction is correct, i.e., 0/1-KNAPSACK-DEC produces Yes for x iff OSA_DGM-DEC produces Yes for y .

(\Rightarrow) Assume that 0/1-KNAPSACK-DEC produces Yes for x , that is, there exists set $I \subseteq \{1, \dots, n\}$ such that $\sum_{i \in I} w_i \leq P$ and $\sum_{i \in I} v_i \geq L$. We construct the sensor alteration α such that for each observation $o \in \Omega$,

$$\alpha(o) = \begin{cases} a_{\clubsuit} & \text{if } o = o_i \text{ for an integer } i \in I \\ o & \text{otherwise} \end{cases}$$

This means when the system enters any state s_i for $i \in I$, the robot is deceived to think it is in s_{\clubsuit} , causing the controller to give action b instead of a to the robot to execute, and by executing that action, the system enters s_{\perp} . Because s_{\perp} is reached only by doing action b from the s_j 's for $j \in \{1, \dots, n\}$, the probability of reaching the decoy goal $\{s_{\perp}\}$ under sensor alteration α is computed

$$\mathbf{Pr}_{\text{reach}}(S_D, \alpha) = \sum_{i \in I} \frac{v_i}{2 \sum_{j=1}^n v_j} = \frac{\sum_{i \in I} v_i}{2 \sum_{j=1}^n v_j}.$$

, and because $\sum_{i \in I} v_i \geq L$,

$$\mathbf{Pr}_{\text{reach}}(S_D, \alpha) \geq \frac{L}{2 \sum_{j=1}^n v_j} = r. \quad (2)$$

Also,

$$\begin{aligned}
C(\alpha) &= \sum_{o \in \Omega} c(o, \alpha(o)) \\
&= \sum_{o \in \Omega \setminus \{s_i | i \in I\}} c(o, \alpha(o)) + \sum_{i \in I} c(o_i, \alpha(o)) \\
&= \sum_{o \in \Omega \setminus \{s_i | i \in I\}} c(o, o) + \sum_{i \in I} c(o_i, o_{\clubsuit}) \\
&= \sum_{o \in \Omega \setminus \{s_i | i \in I\}} 0 + \sum_{i \in I} w_i \\
&= \sum_{i \in I} w_i, \tag{3}
\end{aligned}$$

and because $\sum_{i \in I} w_i \leq P$ and that $B = P$, it holds that $C(\alpha) \leq B$. This combined with (2) implies that OSA_DGM-DEC produces Yes for y .

(\Leftarrow) Assume OSA_DGM-DEC produces Yes for y , meaning that, there is a sensor alteration α for which $\Pr_{\text{reach}}(S_D, \alpha) \geq r$ and that $C(\alpha) \leq B$. State s_{\perp} can be reached either by doing action b at the s_i 's for $i \in \{1, \dots, n\}$ or by doing a at s_{\clubsuit} . The cost of altering o_{\clubsuit} to any other observation is ∞ . Therefore, if the system enters s_{\clubsuit} , it will certainly perform action b , causing it to transition to s_{\top} rather than s_{\perp} in the next time step. Thus, s_{\perp} cannot be reached from s_{\clubsuit} . The system enters s_{\perp} from a state s_i for $i \in \{1, \dots, n\}$ only when o_i is mapped to o_{\clubsuit} by α . Because $\Pr_{\text{reach}}(S_D, \alpha) \geq r$, there must be a set $I \subseteq \{1, \dots, n\}$ such that among all the o_i 's, only those for which $i \in I$, it has been set $\alpha(o_i) = o_{\clubsuit}$. By assumption $C(\alpha) \leq B$, and since $\sum_{i \in I} w_i$ by (3), and that $B = P$, it holds that $\sum_{i \in I} w_i \leq P$. Also, because

$$\Pr_{\text{reach}}(S_D, \alpha) = \sum_{i \in I} \frac{v_i}{\sum_{j=1}^n w_j} \geq r = \frac{L}{\sum_{j=1}^n w_j}, \tag{4}$$

it holds that $\sum_{i \in I} v_i \geq L$. This combined with that $\sum_{i \in I} w_i \leq P$ proves that I yields Yes for instance x of 0/1-KNAPSACK-DEC. \blacksquare

Therefore, the following result is implied from Lemma 1 and Theorem 1.

Theorem 2: OSA_DGM-DEC \in NP-complete.

Proof: Combine Lemma 1 and Theorem 1. \blacksquare

Corollary 1: OSA_DGM \in NP-hard.

Hence, under the assumption that $P \neq NP$, we cannot find in polynomial time, a sensor alteration with a given cost budget that maximizes the probability of misleading the robot to the decoy goal.

VI. OSA_DGM VIA MIXED INTEGER LINEAR PROGRAMMING

In this section, we provide a mixed integer linear programming approach to solve the optimal sensor alteration problem.

The idea of our programming formulation is to have binary variables by assigning values to which, a sensor

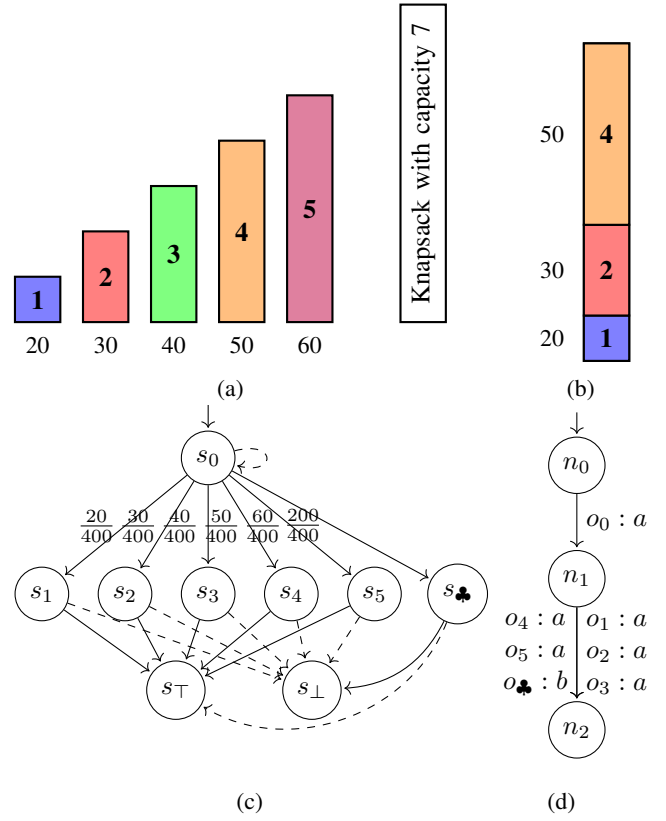


Fig. 2: (a) An instance of the 0/1 knapsack problem. There are 5 items with weights $W = [1, 2, 3, 4, 5]$ and values $V = [20, 30, 40, 50, 60]$. The capacity of the knapsack is 7 (b) Optimal solution to the instance of the 0/1 knapsack problem. The knapsack's total weight is 7, and the total value is 100. (c) The POMDP of the instance of our problem, the OSA_DGM-DEC problem, constructed by our reduction for the instance of the 0/1-KNAPSACK-DEC in Part (a) of this figure. The solid edges are transitions that take place with action a and the dashed arrows are transitions for action b . All the transitions missing probability labels, use probability 1. We omitted those labels to reduce visual clutter. States s_{\perp} and s_{\top} are absorbing states. Their outgoing transitions are omitted to reduce visual clutter. (d) The finite-state controller of the instance of our problem constructed by our reduction for the instance of the 0/1-KNAPSACK-DEC in Part (a) of this figure. All the missing transitions enters n_2 and choose action b .

alteration is synthesized, and then to have certain variables each representing the probability of reaching to the decoy goal by the sensor alteration from a state of a product automaton. Each state of this product automaton is a tuple (s, n, o) representing a situation when the POMDP is in state s , current memory node of the controller is n , and observation $O(s)$ is altered to o by the sensor alteration.

More precisely, we introduce the following variables:

- A binary variable $x_{o,o'}$ for each $o, o' \in \Omega$: Variable $x_{o,o'}$ will receive 1 iff observation o is altered to observation o' by the sensor alteration.

- A continuous variables $z_{s,n,o} \in [0, 1]$ for each state $s \in S$, node $n \in N$, and observation $o \in \Omega$: The value of $z_{s,n,o}$ will be the probability of misleading the robot to the decoy goal when the system is in state s , the controller is in node n , and the observation emitted when the system is in s , $O(s)$, is altered to o under the sensor alteration.

Using these variables, we make the following programming model.

$$\text{Maximize: } z_{s_0, n_0, O(s_0)} \quad (5)$$

Subject to:

- $$\sum_{o, o' \in \Omega} x_{o, o'} \cdot c(o, o') \leq B \quad (6)$$

- $$x_{O(s_0), O(s_0)} = 1 \quad (7)$$

- For each $s \in S_D$, $n \in N$, and $o \in \Omega$,
$$z_{s,n,o} = x_{O(s), o} \quad (8)$$

- For each $s \in S \setminus S_D$, $n \in N$, and $o \in \Omega$
$$z_{s,n,o} = \sum_{\substack{s' \in S, \\ n' \in N, \\ o' \in \Omega}} x_{O(s), o} \mathbf{T}((s, n, o), (s', n', o')) z_{s', n', o'} \quad (9)$$

- For each $o, o' \in \Omega$,
$$x_{o, o'} \in \{0, 1\} \quad (10)$$

- For each $s \in S$, $n \in N$, and $o \in \Omega$,
$$z_{s,n,o} \in [0, 1] \quad (11)$$

The objective (5) is to maximize the probability of misleading the robot to the decoy goal S_D . Constraint (6) ensures that the cost of the alteration is no greater than the cost budget B . Constraint (7) asserts that the observation associated with the initial state of the POMDP cannot be altered. This is because the robot know that any execution of the POMDP starts from s_0 . Constraints (8) sets the base case for the Bellman equation by setting to be 1, the probability of reaching the decoy for a state (s, n, o) where s is a decoy state and observation $O(s)$ is altered to o by the sensor alteration. Constraints (16) simply implement the Bellman equation.

This programming would be an MILP if Constraints (16) were linear. Note that although $\mathbf{T}((s, n, o), (s', n', o'))$, because both $x_{O(s), o}$ and $z_{s', n', o'}$ variables, the product of these three is not a linear term. Thus, we need to linear these constraints.

To linearize them, we first introduce a continuous variable $l_{s,o,s',n',o'}$ for each $s, s' \in S$, $n, n' \in N$, and $o, o' \in \Omega$. The range of this variable is $[0, 1]$. Then, we introduce the following additional constraints to make $l_{s,o,s',n',o'}$ receive value $x_{O(s), o} \cdot z_{s', n', o'}$.

- For each $s, s' \in S$, $n' \in N$, and $o, o' \in \Omega$,
$$l_{s,o,s',n',o'} \geq 0 \quad (12)$$

$$l_{s,o,s',n',o'} \leq z_{s', n', o'} \quad (13)$$

$$l_{s,o,s',n',o'} \leq x_{O(s), o} \quad (14)$$

$$l_{s,o,s',n',o'} \geq z_{s', n', o'} - (1 - x_{O(s), o}) \quad (15)$$

Finally, we replace Constraints 16 with the following constraints:

- For each $s \in S \setminus S_D$, $n \in N$, and $o \in \Omega$

$$z_{s,n,o} = \sum_{\substack{s' \in S, \\ n' \in N, \\ o' \in \Omega}} \mathbf{T}((s, n, o), (s', n', o')) \cdot l_{s,o,s',n',o'} \quad (16)$$

This MILP is complete, but needs to be improved. Note that given a pair of state $s, s' \in S$, the MILP introduces all variables $l_{s, \cdot, s', \cdot}$ even if $\mathbf{P}(s, a, s') = 0$ for all $a \in A$. We improve the MILP by introducing those variables only when for at least an action $a \in A$, $\mathbf{P}(s, a, s') > 0$, but if for all $a \in A$, it holds that $\mathbf{P}(s, a, s') = 0$, then we do not have to even create the variables $l_{s, \cdot, s', \cdot}$ and the constrains involving them.

This MILP formulation not only can be used for computing optimal solutions to OSA_DGM but also for computing sub-optimal solutions for problem instances for which optimal solutions cannot be computed under limited time budgets. The presented MILP can be directly solved by a variety of highly-optimized MILP solvers.

VII. CASE STUDIES

In section, we present results of our implementation of the MILP for several instances of the OSA_DGM problem. We implemented our program in Python and used the Python interface of Gurobi [6] to solve the MILP instances. All experiments were performed on a system with Windows 11, Core i-9 (2000 Mhz) processor, and 32GB memory.

A. The Reduction Example

Our implementation for the OSA_DGM instance in the reduction, Figure 2c, compute the sensor alteration in which

$$o_4 \rightarrow o_{\clubsuit}, o_2 \rightarrow o_{\spadesuit}, o_1 \rightarrow o_{\heartsuit}.$$

The MILP for this instance had 59,373 variables and 236,477 constraints. It took 0.099 for the solver to find an optimal solution. This experiment verifies the correctness of our algorithm.

B. Grid Environment

This case study considers a robot operating in a $n \times n$ grid environment. An instance where $n = 5$, is shown in Figure 3. The robot is tasked with delivering an item from the starting position $(0, 0)$ to the goal position $(n-1, n-1)$. The robot has four actions: N, S, E, W , which respectively command the robot's actuators to move to the cell in the North, South, East, and West side of its' current position. The robot's dynamic is stochastic: The actuators of the robot guarantee that they move the robot to the intended cell specified by the action command with probability 0.8. The probability that the robot's actuators move the robot in either of the two unintended directions orthogonal to its current

cell is 0.1 each, provided movement in both directions is possible. Otherwise, if only one unintended direction is possible (for example, when robot is in one of the corner cells of the environment), the probability of movement in that direction is 0.2. Figure 3(bottom) illustrate these two situations. Generally, the robot is unaware about its own position, i.e., the current state is not observable, but using 7 range sensors s_0 through s_6 , it has partial observability of the current state of the world. Each sensor s_i produces an observation o_i when the robot enters a cell guarded by s_i . Therefore, there are 7 observations, each produced by a sensor, along with one additional observation, which we denoted b , produced by the cells that are not guarded by any sensors. With these in mind, the problem is formulated as a POMDP, which has a state for each potential position of the robot—each cell in the grid. Cell (2, 2), which in the figure is denoted by X in red, is hazardous, and a policy governing the robot movement should help the robot to not enter that cell. In the POMDP, the states corresponding to the goal and the hazardous position are *absorbing*—states that have self-loops for all actions.

The robot uses the finite-state controller shown in Figure 3. The controller tells the robot to do action E when the robot receives any of the observations o_0, o_1, o_4 , and o_6 , and to do action N when any of the observation o_2, o_3 , and o_4 , are received. The action produced for the blank observation, b , will be E if the last non-blank observation was any of the observations o_0, o_1, o_4 , and o_3 , and otherwise, it will be N .

Our implementation indicated that the probability of reaching the goal position using this controller without sensor alteration (i.e., a sensor alteration with a cost budget 0) for the instance with $n = 5$ is 0.915 and the probability of entering the hazardous position is 0.085. The purpose of the attacker is to mislead the robot to enter the hazardous state, and hence, the decoy goal contains only that state. For a cost budget 1, our implementation of the MILP formulation computed the sensor alteration that changes o_1 to o_3 . This is consistent with the problem instance because altering o_1 to o_3 maximizes the likelihood of misleading the robot into entering the decoy when only one observation can be altered. Our program indicated that the probability of reaching the decoy under this sensor alteration is 0.720, which is significantly higher than 0.085, the probability of misleading without a sensor attack. To interpret the impact of this sensor alteration, consider that under this sensor alteration, the probability of following the path $(0, 0) \rightarrow (1, 0) \rightarrow (2, 0) \rightarrow (2, 1) \rightarrow (2, 2) \rightarrow (2, 2)$, alone is $0.8^4 = 0.4096$.

We repeated the experiment for other cost budgets greater than 1. For a cost budget of 2, our implementation computed to do the alteration $o_1 \rightarrow o_5$ and $o_2 \rightarrow o_0$. This is consistent with the positions of s_1 and s_2 and the action produced for o_1, o_5, o_2 , and o_0 by the controller. The probability of misleading the robot to the decoy under this sensor alteration is 0.861. For a cost budget of 3, our algorithm decided to alter o_6 to o_2 , along with $o_1 \rightarrow o_5$ and $o_2 \rightarrow o_0$. Considering the stochastic nature of the robot’s dynamics and controller, altering o_6 affects the robot’s decision by

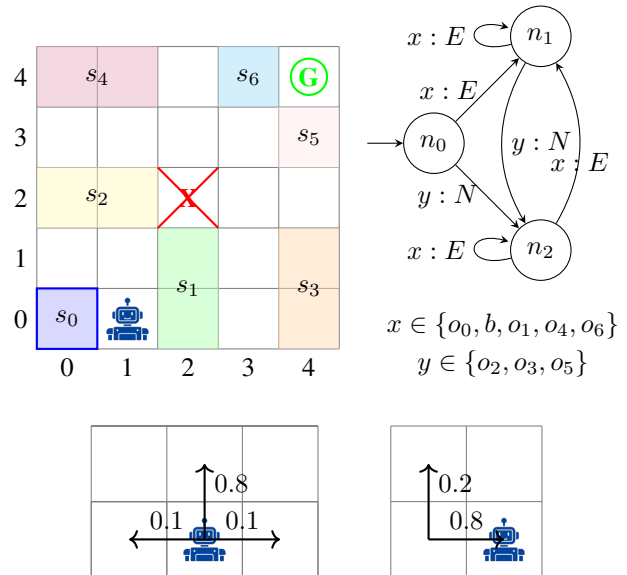


Fig. 3: **Top-left**) A grid environment guarded by 7 range sensors s_0 through s_7 . The robot is tasked to deliver an item from the starting location (0, 0) to the goal location, (4, 4). Cell (2, 2) is hazardous and must be avoided. That cell considered a decoy and the attacker’s purpose is to mislead the robot to that cell. **Top-right**) A finite-state controller the robot uses. **Bottom-left**) The robot’s dynamic when it performs action N , standing for going to North. **Bottom-right**) The robot’s dynamic when it performs action E , standing for going to East.

causing it to take action N instead of E upon observing o_6 . As a result, the robot may be directed to move through the blank cells to the left and bottom-left of s_6 before entering the hazardous state, which could be reached by an infinite number of paths. The probability of following those paths is $\sum_{n=1}^{\infty} 0.001(0.01)^n \approx 0.00101$. Our implementation indicated that under the computed sensor alteration, the probability of reaching the decoy is 0.862, rounded to three decimal places. For a cost budget 4, the computed sensor alteration included $o_2 \rightarrow o_6, o_5 \rightarrow o_1, o_1 \rightarrow o_2$, and $o_6 \rightarrow o_3$. Under this alteration, the probability of reaching the decoy was 0.864 (rounded up). This probability remained the same for cost budgets greater than 4. In each case, the actions assigned by the computed sensor alteration for the observations produced by sensors s_4 and s_3 did not change. This is reasonable because for example, there is no advantage in choosing action N instead of E upon observing o_4 .

We performed a scalability experiment by creating 6 instances of an $n \times n$ grid for $n \in \{5, 15, 25, 35, 45\}$. Each instance maintained the same topology as the case with $n = 5$, shown in Figure 3, the same number of sensors as and the same sensor range sizes. For each we measured the number variables of the MILP, the number of constraints of the MILP, the time to construct the MILP, and the time to solve the MILP. Results of this experiment are shown in Figure 4. As expected, both computation and execution times

were increased as the size of the grid increased.

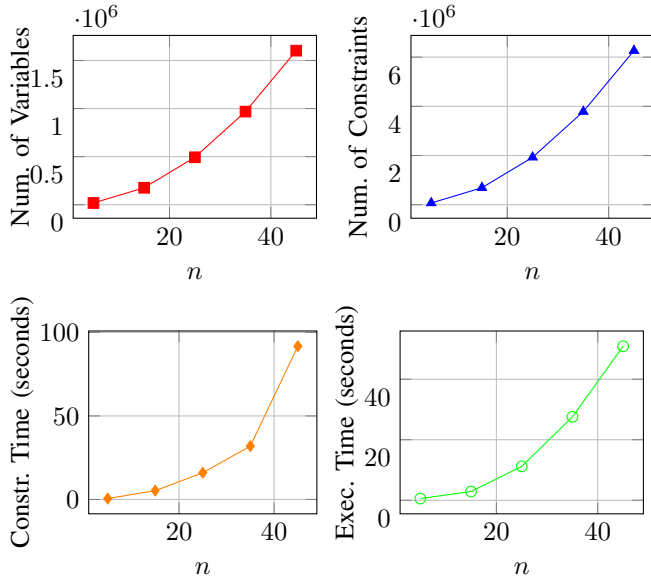


Fig. 4: Results of our scalability experiment for grids similar to the grid in Figure 3a of size $n \times n$, $n \in \{5, 15, 25, 35, 45\}$. Note that for the top two graphs, the values on the y-axis are in millions (e.g., the instance for $n = 45$ has more than 1.6 million variables and more than 6.2 million constraints).

VIII. CONCLUSION

In this paper, we studied a sensor deception problem the aim of which is to find a sensor alteration that can maximize the probability of misleading an agent to a predetermined decoy goal, under limited capability for sensor alteration. The environment is modeled by a POMDP and the agent's actions are governed by a Finite State Controller (FSC). We proved that our problem is NP-hard, and provided an algorithm based on MILP. We showed through experiments that our algorithm is capable of computing optimal solutions for problems of moderate size. Future work can focus on improving the MILP or introducing a new one with less variables and constraints. The standard techniques to deal with NP-hard problems, such as introducing heuristic and approximate solutions, as well as identifying instances that can be solved in polynomial time, can be considered for future work. Also, it might be useful to study the problem for more general kinds of models and strategies.

REFERENCES

- [1] M. Ahmadi, R. Sharan, and J. W. Burdick, "Stochastic finite state control of pomdps with ltl specifications," *arXiv preprint arXiv:2001.07679*, 2020.
- [2] A. H. Anwar, C. A. Kamhoua, N. O. Leslie, and C. Kiekintveld, "Honey-pot allocation for cyber deception under uncertainty," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3438–3452, 2022.
- [3] C. Baier and J.-P. Katoen, *Principles of model checking*, 2008.
- [4] S. Bernardini, F. Fagnani, S. Franco *et al.*, "An optimization approach to robust goal obfuscation," in *PROCEEDINGS-INTERNATIONAL CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING*, vol. 1. International Joint Conference on Artificial Intelligence (IJCAI), 2020, pp. 118–128.

- [5] S. Chen, Y. Savas, M. O. Karabag, B. M. Sadler, and U. Topcu, "Deceptive planning for resource allocation," in *2024 American Control Conference (ACC)*. IEEE, 2024, pp. 4188–4195.
- [6] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: <https://www.gurobi.com>
- [7] R. Jacob, J.-J. Lesage, and J.-M. Faure, "Overview of discrete event systems opacity: Models, validation, and quantification," *Annual Reviews in Control*, vol. 41, pp. 135–146, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1367578816300189>
- [8] M. O. Karabag, M. Ornik, and U. Topcu, "Exploiting partial observability for optimal deception," *IEEE Transactions on Automatic Control*, vol. 68, no. 7, pp. 4443–4450, 2023.
- [9] C. Keroglou and C. N. Hadjicostis, "Probabilistic system opacity in discrete event systems," in *2016 13th International Workshop on Discrete Event Systems (WODES)*, 2016, pp. 379–384.
- [10] A. N. Kulkarni, J. Fu, H. Luo, C. A. Kamhoua, and N. O. Leslie, "Decoy allocation games on graphs with temporal logic objectives," in *Decision and Game Theory for Security: 11th International Conference, GameSec 2020, College Park, MD, USA, October 28–30, 2020, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 168–187. [Online]. Available: https://doi.org/10.1007/978-3-030-64793-3_9
- [11] A. N. Kulkarni, H. Luo, N. O. Leslie, C. A. Kamhoua, and J. Fu, "Deceptive labeling: Hypergames on graphs for stealthy deception," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 977–982, 2021.
- [12] A. Kulkarni, S. Srivastava, and S. Kambhampati, "Signaling friends and head-faking enemies simultaneously: Balancing goal obfuscation and goal legibility," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, p. 1889–1891.
- [13] L. Li, H. Ma, A. N. Kulkarni, and J. Fu, "Dynamic hypergames for synthesis of deceptive strategies with temporal logic objectives," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 334–345, 2023.
- [14] H. Ma, S. Han, C. Kamhoua, and J. Fu, "Optimal resource allocation for proactive defense with deception in probabilistic attack graphs," in *Decision and Game Theory for Security*, J. Fu, T. Kroupa, and Y. Hayel, Eds. Cham: Springer Nature Switzerland, 2023, pp. 215–233.
- [15] H. Ma, C. Shi, S. Han, M. R. Dorothy, and J. Fu, "Covert planning against imperfect observers," in *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '24. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2024, p. 1319–1327.
- [16] R. Meira-Góes, R. Kwong, and S. Lafortune, "Synthesis of sensor deception attacks for systems modeled as probabilistic automata," in *2019 American Control Conference (ACC)*, 2019, pp. 5620–5626.
- [17] R. Meira-Góes, R. H. Kwong, and S. Lafortune, "Synthesis of optimal multiobjective attack strategies for controlled systems modeled by probabilistic automata," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 2873–2888, 2022.
- [18] S. Milani, W. Shen, K. S. Chan, S. Venkatesan, N. O. Leslie, C. Kamhoua, and F. Fang, "Harnessing the power of deception in attack graph-based security games," in *Decision and Game Theory for Security*, Q. Zhu, J. S. Baras, R. Poovendran, and J. Chen, Eds. Cham: Springer International Publishing, 2020, pp. 147–167.
- [19] C. Probine, M. O. Karabag, and U. Topcu, "A decentralized shotgun approach for team deception," in *Decision and Game Theory for Security*, A. Sinha, J. Fu, Q. Zhu, and T. Zhang, Eds. Cham: Springer Nature Switzerland, 2025, pp. 177–197.
- [20] Y. Savas, C. K. Verginis, and U. Topcu, "Deceptive decision-making under uncertainty," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 5, pp. 5332–5340, Jun. 2022. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/20470>
- [21] R. Su, "Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations," *Automatica*, vol. 94, pp. 35–44, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0005109818301912>
- [22] Q. Zhang, C. Seatzu, Z. Li, and A. Giua, "Selection of a stealthy and harmful attack function in discrete event systems," *Scientific Reports*, vol. 12, no. 1, p. 16302, 2022.