Resource Bounded Secure Goal Obfuscation

Anagha Kulkarni[1] and **Matthew Klenk**[2] and **Shantanu Rane**[2*] and **Hamed Soroush**[2*]

[1]Computer Science Department, Arizona State University, Tempe, AZ, USA
anaghak@asu.edu

[2]Palo Alto Research Center, Palo Alto, CA, USA
firstname.lastname@parc.com

Abstract

An AI system that is operating in an adversarial environment should be able to provide safeguards for its internal information. In other words, an adversary should not be able to perform diagnosis on the AI system's internal information based on the resulting observations during plan execution. The system should be able to produce a plan that achieves the desired objective while minimizing leakage about its internal information. In this paper, we present an approach that allows an AI agent to securely obfuscate its true goal (i.e., agent's internal information) for as long as possible using a subset of candidate goals. By making all the candidate goals equally likely for as long as possible, the agent's true goal is kept secured. The AI agent may have to incur an additional cost to reach its true goal, but this cost buys the obfuscation guarantee. Given a larger resource budget, greater obfuscation is possible. We present empirical evaluations of our approach using IPC domains to show its scope and feasibility.

1 Introduction

In an adversarial environment, an implicit requirement for an AI agent is to minimize information leakage while achieving its objectives. If the agent's activities are not secure, an adversarial observer can use diagnosis to infer internal information and interfere with the agent's objectives. Consider the following domains: in military planning, adversaries observe troop movements to infer possible targets; in corporate strategy, competitors predict each others future directions by observing potential partnerships; in product design, component specifications often portend new product's functionality. In such settings, the agent should generate behaviors that can obfuscate its actual objective for as long as possible. The execution of obfuscated behaviors may be more expensive. This leads to the agent trading-off its available resources in order to ensure privacy of the sensitive information.

Our problem setting considers two agents, an actor and an observer. The actor can perform actions in the environment. The observations, known to the actor and the observer, are the result of the actor's activities. The observations can be partially or fully observable. Partially observable observations reveal some information about the actor's actions and state but do not reveal it exactly, whereas the fully observable ones convey the exact action taken by the actor and the corresponding state transition. While the observer knows a set of goals the actor could be pursing, the true goal is not known to the observer. Therefore the objective is to securely obfuscate the true goal while the actor is trying to achieve it. In this work, we present approaches that the actor can utilize to obfuscate its true goal for as long as the observation model allows and in a cost-effective manner.

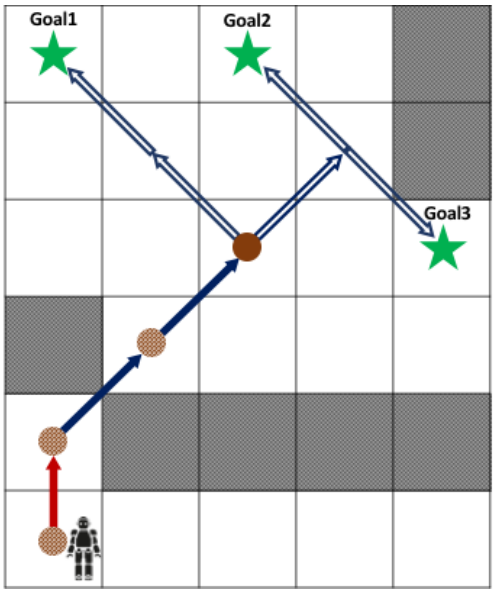

Figure 1: A `gridworld` example illustrating a stronger notion of privacy. With an observation model that distinguishes diagonal and orthogonal actions, the observer sees the same sequence of observations for all the three goals regardless of the agent's true goal.

There have been recent works (Kulkarni, Srivastava, and Kambhampati 2018; Masters and Sardina 2017; Keren, Gal, and Karpas 2016b) which explore the problem of privacy preservation and deception. We explore the goal obfuscation problem introduced in Kulkarni, Srivastava, and Kambhampati (2018). Our main contribution is twofold: introduction of a stronger definition of privacy with respect to the observation sequence generated, and an approach to trade-off additional cost incurred with the amount of obfuscation

*The authors appear in alphabetical order

achieved. The problem of goal obfuscation (Kulkarni, Srivastava, and Kambhampati 2018) involves hiding the true goal of the agent from the observer by making the observation sequence "consistent" with v candidate goals. Essentially, consistent with v candidate goals means that, there exists a plan with the same observation sequence to achieve each of the v goals. However, we introduce a stronger definition of consistency, such that, when the solution computation mechanism is run with either of the v goals as a "true goal", we get the same observation sequence. Thus, our solution observation sequence makes all the v goals equally likely for as long as possible. This preserves the privacy of the true goal even when the observer has access to the computation mechanism. Also, our approach can compute the cheapest solution that satisfies the stronger privacy property. In the following paragraph, we present an example that achieves obfuscation with respect to v goals and satisfies the stronger privacy property.

Consider the `gridworld` in Figure 1. Assume each cell allows movement in all 8 adjacent cells with equal cost. The initial state is at cell (0, 0). The candidate goals are given by Goal1, Goal2, Goal3 and Goal4, and Goal2 is the actor's true goal. With $v = 3$, we select Goal1, Goal2 and Goal3 because they have higher landmark similarity than any other combination of 3 goals (inclusive of true goal). The observation model emits two types of observation symbols for each cell: o_1 (marked by blue arrow) if the next cell is diagonally adjacent and o_2 (marked by red arrow) otherwise. In the Figure 1, we can see three plans where each reaches a candidate goal. Each of these plans produce the same observation sequence given by $\{o_2, o_1, o_1, o_1, o_1\}$. Each of these plans is a secure goal-obfuscated plan, where $v = 3$. The observation sequence is the same regardless of the true goal. A secure goal-obfuscated plan can be achieved by finding the cheapest path to a closest point that is equidistant (in terms of remaining steps) from each of the goals. In the example, the equidistant state at cell (3, 4), is 2 steps away from each goal. From this equidistant state, we check if there exists an observation sequence which is common to the 3 goals and reaches the goals in only 2 steps. From cell (3, 4), the observation sequence $\langle o_1, o_1 \rangle$ satisfies the requirement. The four equidistant states for the three candidate goals are marked in brown color.

If the observation model allows finer observations, such that, each grounded action is mapped to a unique observation symbol, then a fully obfuscated plan might not be possible. In such cases, our aim is to obfuscate the true goal for as long as possible by computing a plan that achieves an equidistant state, such that the cost from the equidistant state to each of the goals is minimized. We also discuss a method for choosing v goals among a set of n candidate goals, such that, we can minimize the solution cost and maximize the overall obfuscation. In the following sections, we discuss our approach in detail. We also establish upper and lower bounds on the cost of solutions which satisfy our definition of obfuscation. We present evaluation of our approaches using three International Planning Competition (IPC) domains, namely, `Blocksworld`, `Logistics` and `Zenotravel`.

2 Background and Preliminaries

We use the formulation of goal obfuscation planning problem (Kulkarni, Srivastava, and Kambhampati 2018) to define essential properties for a secure goal obfuscation problem. We also list a few cryptography assumptions that help specify the behavior of the agents and our algorithm.

2.1 Classical Planning

A classical planning problem can be defined as a tuple $\mathcal{P} = \langle \mathcal{D}, \mathcal{I}, G \rangle$, where $\mathcal{D} = \langle \mathcal{F}, \mathcal{A} \rangle$ is the actor's domain with $\mathcal{F}$ representing a finite set of fluents that define the state of the world $s \subset \mathcal{F}$, $\mathcal{A}$ is a finite set of actions available to the actor, and $\mathcal{I}, G \subset \mathcal{F}$ representing the initial state and goal subformulae. Each action $a \in \mathcal{A}$ is a tuple of the form $\langle c(a), pre(a), eff^+(a), eff^-(a) \rangle$ where $c(a)$ denotes the cost of an action, $pre(a) \subseteq \mathcal{F}$ is a set of preconditions for the action a, $eff^+(a) \subseteq \mathcal{F}$ is a set of positive effects and $eff^-(a) \subseteq \mathcal{F}$ is a set of negative effects, i.e., $\Gamma(s, a) \models \perp$ if $s \not\models pre(a)$; else $\Gamma(s, a) \models s \cup eff^+(a) \setminus eff^-(a)$ where $\Gamma(\cdot)$ is the transition function. The cumulative transition function is given by $\Gamma(s, \langle a_1, a_2, \ldots, a_n \rangle) = \Gamma(\Gamma(s, a_1), \langle a_2, \ldots, a_n \rangle)$. The solution to $\mathcal{P}$ is a *plan* or a sequence of actions $\pi = \langle a_1, a_2, \ldots, a_n \rangle$, such that, $\Gamma(\mathcal{I}, \pi) \models G$, i.e., starting from the initial state to the goal. The cost of the plan is, $cost(\pi) = \sum_{a_i \in \pi} c(a_i)$.

Note that, we assume all the actions in the planning problem are of uniform cost equal to 1. In other words, the length of the plan is equal to the cost of the plan and an optimal plan is a shortest length plan for the given planning problem.

2.2 Goal Obfuscation Planning Problem

The goal obfuscation planning problem, as defined in Kulkarni, Srivastava, and Kambhampati (2018), involves two agents: an actor planning in the environment and an observer. The observer has access to the domain model of the actor, which includes the set of fluents, actions and the initial state of the actor. The observer also has access to the set of candidate goals but is *unaware* of the true goal of the actor. Whenever the actor performs an action in the environment, an observation is emitted. The observation model available to the observer is deterministic and is accessible to both the agents.

Definition 1. *(Kulkarni, Srivastava, and Kambhampati 2018) A **goal obfuscation planning problem** is a tuple, $\mathcal{P}_{\mathcal{GO}} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G}, \Omega, \mathcal{O} \rangle$, where,*

- *$\mathcal{G} = \{G_A \cup G_1 \cup G_2 \ldots \cup G_{n-1}\}$ is a set of n candidate goal conditions, and G_A is the true goal of the actor.*
- *$\Omega = \{o_i | i = 1, \ldots, m\}$ is a finite set of observations symbols corresponding to the domain.*
- *$\mathcal{O} : (\mathcal{A} \times \mathcal{S}) \rightarrow \Omega$ is the observation function which allows either partial or full observability mapping the pair: action taken and state reached to observation symbols*

When the observation model allows partial observability, multiple $\langle a, s' \rangle$ pairs correspond to the same observation. When it allows full observability, each observation corresponds to a unique $\langle a, s' \rangle$ pair. A plan solution to a goal-obfuscation planning problem achieves the true goal while

obfuscating it from an adversarial observer for as long as possible.

2.3 Secure Goal-Obfuscation

An important point to note is that the observer only receives the observation sequence associated with the plan executed by the actor. Despite having access to the agent's goal set $\mathcal{G}$ in $\mathcal{P}_{\mathcal{GO}}$, the observer is unaware of the agent's true goal. A secure goal-obfuscated plan is a solution to $\mathcal{P}_{\mathcal{GO}}$, that minimizes the bias of the observation sequence towards the true goal. With respect to a chosen set of v goals, such that, $\mathcal{G}_v \subseteq \mathcal{G} \wedge G_A \in \mathcal{G}_v$, if we can make the observation sequence associated with the actor's plan identical for all the v goals, then we can say that the true goal is securely obfuscated. The value of v can be $v \leq n$. We will now distill three important properties of a secure goal-obfuscated plan: **(1)** For each goal in $\mathcal{G}_v$, there should exist a plan which achieves the goal in l number of steps, **(2)** The observation sequence associated with v plans that each achieve a goal in $\mathcal{G}_v$ should be the same, **(3)** The observation sequence should not be biased by any goal in $\mathcal{G}_v$. The first and second properties are straight-forward. The third property states that the actor should be able to compute the same observation sequence while generating a plan solution to any of the v goals. If the above mentioned properties are followed, the observation sequence associated with the plan does not disclose any specific information about the true goal.

2.4 Cryptography Assumptions

We now list a few standard assumptions in cryptography and secure multi-party computation literature presented in several prior works (Lindell 2005; Štolba 2017). The properties listed in the Subsection 2.3 can be interpreted through these assumptions. The following assumptions specify the properties on the behavior of the agents and the algorithm.

Assumption 1. ***Adversary knows the actor's algorithm***. *This means that the adversarial observer has access to the algorithm and knows how the algorithm works. For example, the adversary knows what states are generated by the algorithm.*

Assumption 2. ***Independence of inputs***. *This means that the adversary can run the algorithm with different inputs. For example, the adversary can set different goals as true goal to check the variability of the output.*

Assumption 3. ***Delivery of observations is fair and in-order***. *This means that when the actor takes an action and reaches a corresponding state, only then the adversary receives the corresponding observation. And the delivery of each observation occurs in the order in which it was emitted by the actor's plan execution.*

Assumption 4. ***Semi-Honest Adversary***. *This means that the adversary is a passive adversary and follows the algorithm and the protocol correctly but may glean information from the execution to learn about the private information of the actor.*

All the four assumptions are commonly found in cryptography literature. Assumptions 1 and 2 delineate the essential properties of a good obfuscating algorithm. Assumption 1 states that the agent should not rely on the privacy of the algorithmic mechanism itself. Assumption 2 states that the adversary may rerun the algorithm with different inputs, therefore the algorithmic mechanism should be immune to reruns. Assumptions 3 and 4 outline the behavior of adversary. Assumption 3 states that the adversary can not manipulate the execution of plan and delivery of the observations. Assumption 4 states that our adversary is a passive adversary that follows the protocol. These assumptions are significantly different from those typically taken in the AI community in that they do not rely on assumptions about the adversary's goal recognition model.

3 Computation of Secure Goal-Obfuscated Plans

In this section, we will discuss our approach for computing a secure solution to a goal obfuscation planning problem. Given the properties in Subsection 2.3 and assumptions in Subsection 2.4, we can say the following:

Definition 2. *A* ***secure goal obfuscated planner****, computes a plan solution $\pi_{\mathcal{P}_{\mathcal{GO}}}$, and an observation sequence $O_{\mathcal{P}_{\mathcal{GO}}}$, to a $\mathcal{P}_{\mathcal{GO}} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G}_v, \Omega, \mathcal{O} \rangle$, where, $\Gamma(\mathcal{I}, \pi_{\mathcal{P}_{\mathcal{GO}}}) \models G_A$, such that, $\forall G' \in \mathcal{G}_v$, if G' is set as the true goal of the actor, there exists a plan $\pi'_{\mathcal{P}_{\mathcal{GO}}}$, and an observation sequence $O'_{\mathcal{P}_{\mathcal{GO}}}$, where $\Gamma(\mathcal{I}, \pi_{\mathcal{P}_{\mathcal{GO}}}) \models G' \wedge O'_{\mathcal{P}_{\mathcal{GO}}} = O_{\mathcal{P}_{\mathcal{GO}}}$.*

Therefore, given a set of v goals, a secure goal-obfuscated planner will generate the same observation sequence when any of the v goals is set as a true goal. For a given problem, if there does not exist a secure goal obfuscated plan (say the observation model doesn't allow for it), then the planner computes a partial plan starting from initial state that satisfies the privacy assumptions for as long as possible. This partial plan guarantees secure obfuscation for the entirety of its length (although not until the goal). In such cases, we compute partial secure plans which minimize the distance to the true goal in which revealing observations occur. The process of computing secure goal-obfuscated plans consists of two phases: **(1)** selection of v goals from the set of n candidate goals and **(2)** computation of a secure goal-obfuscated plan and observation sequence. The first phase selects v candidate goals (inclusive of true goal) that have higher similarity with each other. And then, the second phase computes a plan whose observation sequence is consistent with all of the goals present in the chosen set of v goals.

3.1 Decoy Goal Selection

We choose a set of $v-1$ candidate goals such that these goals have higher similarity with the true goal and with each other. We use a landmark based measure to compute the similarity between the candidate goals. There is a prior work that has made use of disjunctive landmarks for diverse planning (Bryce 2014). In automated planning, for a given problem instance each landmark is a subset of fluent instantiations that every plan must satisfy in order to solve the problem. The intuition behind using landmarks is that, the goals with common landmarks will go through similar states/actions

and thereby introduce inherent ambiguity in the plans that achieve the candidate goals. We can choose $\mathcal{G}_v$ as follows:

1. For each goal, $G \in \mathcal{G}$, extract the set of landmarks and add each distinct landmark to a set $\mathcal{L}$.
2. Initialize each landmark $L \in \mathcal{L}$ with an empty list.
3. For each distinct landmark L, if it belongs to a goal G, add G to L's list.
4. Order the landmarks in $\mathcal{L}$ in decreasing order of the number of associated goals.
5. Select all the landmarks in $\mathcal{L}$ that are associated with at least v goals. If no such landmarks exist then starting from the first select all landmarks until there are v unique goals.
6. Order the unique goals in the decreasing order of their frequency in the selected landmarks.
7. If the true goal appears in first half then starting from the first goal (otherwise, starting from the last goal), divide the goals in groups of unique v goals.
8. Select the group of v goals that includes the true goal.

Given the set of v goals, we can obtain a lower bound on the cost of our secure goal-obfuscated plan.

Proposition 1. *For given $\mathcal{G}_v$, let π_{G_i} be the optimal plan to reach the farthest G_i in $\mathcal{G}_v$. If π_v is a secure goal obfuscated plan that solves $\mathcal{P}_{\mathcal{GO}}$ then,*

$$cost(\pi_v) \geqslant cost(\pi_{G_i})$$

The above proposition states that the cost of a solution to $\mathcal{P}_{\mathcal{GO}}$ cannot be cheaper than the optimal cost to reach the farthest goal in the set of v goals. This proposition can be useful in a resource sensitive setting, the decoy goals can be chosen such that the lower bound of the solution cost is smaller than the available cost-bound. Note that in our setting, all the actions have unit cost.

3.2 Secure Goal-Obfuscated Plan

Once we have chosen the set of v goals, the next step is to compute a secure goal-obfuscated plan. As stated in the properties listed in Subsection 2.3, the observation sequence should not be biased by any particular goal in $\mathcal{G}_v$. In order to achieve that, we do the following: **(1)** we first compute a set of states that are equidistant to each of the goals in $\mathcal{G}_v$ and **(2)** then we compute a bounded length plan from the equidistant state to a goal, such that the observation sequence is same for plans reaching other goals from the equidistant state. We call it bounded length belief plan, this idea is similar to that of "k-ambiguous plan" in Kulkarni, Srivastava, and Kambhampati (2018). Formally, an *equidistant state* and a *bounded length belief plan* are defined as follows:

Definition 3. *An **equidistant state**, e, is a state in the state space of a $\mathcal{P}_{\mathcal{GO}}$ from which there are d number of steps/observations remaining to each of the v goals.*

Each of the v goals can be achieved in equal number of steps from an equidistant state. When the observation model allows partial observability, the observer operates in the belief space. For every observation emitted, the observer's belief space is updated to reflect all the possible states consistent with the observation sequence. For example, in Figure 1, the observation $o1$, is consistent with the actor being in any of the 4 diagonally adjacent cells. On the other hand, when the observation model allows full observability, there is no belief space. Note that, as mentioned before, in case of full observability only partial plan, that is plan up to the closest (to the goals) equidistant state is secure. In the case of partial observability, we compute bounded length belief plan.

Algorithm 1: Computation of Equidistant States

```
1  procedure Equidistant_Computation(D, G_v, Ω, O)
2      open ← Priority_Queue()                          ▷ Open list
3      equidistant ← Priority_Queue()  ▷ Equidistant states
4      closed ← {}                                      ▷ Closed list
5      h_diff, h_max ← Heuristic_Computation(G_v)
6      open.push(I, h_max)
7      if h_diff = 0 then
8          equidistant.push(I, h_max)
9      end
10     while open ≠ ∅ do
11         s ← open.pop()
12         closed ← closed ∪ s
13         for a, s' ∈ successors(s) do
14             g(s') ← g(s) + cost(a)
15             h_diff, h_max ← Heuristic_Computation(G_v)
16             if s' ∉ open and s' ∉ closed then
17                 open.push(s', g(s') + h_max)
18                 if h_diff = 0 then
19                     equidistant.push(I, g(s') + h_max)
20                 end
21             else if g(s') < g^prev(s') then
22                 if s' ∉ open then
23                     closed ← closed \ s'
24                     open.push(s', g(s') + h_max))
25                     if h_diff = 0 then
26                         equidistant.push(I, g(s') + h_max)
27                     end
28                 else
29                     update priority to g(s') + h_max
30                 end
31             end
32         end
33     end
34     return equidistant
35 procedure Heuristic_Computation(G_v)
36     h_{G_v} ← {}
37     for G ∈ G_v do
38         compute h_G
39         h_{G_v} ← h_{G_v} ∪ h_G
40     end
41     h_diff ← max(h_{G_v}) − min(h_{G_v})
42     return (h_diff, max(h_{G_v}))
```

Definition 4. *A **bounded length belief plan**, π_{G_A}, is a plan of length, d, associated with an observation sequence, O_v, where $\Gamma(e, \pi_{G_A}) \models G_A$, then $\forall\ G \in \mathcal{G}_v\ \exists\ \pi_G$ of length d and associated with observation sequence O_v, where $\Gamma(e, \pi_G) \models G$. Here d is the number of remaining steps from e to each of the goals.*

From an equidistant state, we perform a blind search for

bounded length d in the belief space. After d steps if all the goals are found such that the observation sequence is same then we output the bounded length belief plan and its observation sequence.

A secure goal-obfuscated plan is then generated by computing a cheap plan to an equidistant state, e and then computing a bounded length belief plan from e to the goals. The computation of the first part is done by performing a state space search. The state space is searched to compute a list of equidistant states. If from a given state, the nearest and the farthest of the v goals have the same number of remaining steps then we add it to the list of equidistant states. The details of this procedure are given in Algorithm 1. The computation of second part constitutes performing a bounded length blind search in the belief space starting from each equidistant state until the solution is found. The equidistant states are then processed one by one, starting from the closest equidistant state to the goals. For each equidistant state, the bounded length belief search is run. The details of this procedure are given in Algorithm 2.

Given the set of equidistant states in the entire state space of $\mathcal{P}_{\mathcal{GO}}$, we can obtain a upper bound on the cost of our secure goal-obfuscated plan.

Proposition 2. *For given $\mathcal{G}_v$, let $\mathcal{E} = \{e_1, \ldots, e_n\}$ be the set of all the equidistant states for $\mathcal{P}_{\mathcal{GO}}$, such that, $\forall e_i \in \mathcal{E}, \pi^{\mathcal{I}}_{e_i}$ be the plan from initial state to e_i and $\pi^{e_i}_{G}$ be the plan from e_i to a $G \in \mathcal{G}_v$. If π_v is a secure goal obfuscated plan that solves $\mathcal{P}_{\mathcal{GO}}$ then,*

$$cost(\pi_v) \leqslant \underset{e \in \mathcal{E}}{\operatorname{argmax}}\ cost(\pi^{\mathcal{I}}_e) + cost(\pi^{e}_G)$$

The above proposition states that the cost of a solution to $\mathcal{P}_{\mathcal{GO}}$ is bounded by the cost of a plan via the costliest equidistant state. This proposition can be useful in a resource sensitive setting, the set of equidistant states can be filtered further before starting belief space search such that the upper bound on the solution cost is smaller than or equal to the available cost-bound.

4 Empirical Evaluation

In this section, we evaluate the scope and usability of our approach. Our empirical evaluation measures the following objectives:

1. The impact of different observation models on the extent of obfuscation.
2. The trade-off between additional cost and extent of obfuscation possible.
3. The comparison between run time and plan costs for goal obfuscation planning versus optimal planning.

In the following subsections, we will discuss the domains used, and the setup of the problems and observation models for each of these. Then we will discuss the observations and results derived for each of the three evaluation objectives.

4.1 Domains and Experimental Setup

We use three IPC domains, namely `Blocksworld`, `Logistics` and `Zenotravel` to evaluate our approach.

Algorithm 2: Computation of Bounded Length Belief Plan and Observation Sequence

```
1  procedure belief_plan_Computation(equidistant, D, G_v, Ω, O)
2      h_min ← ∞
3      farthest_equi ← ∅
4      while equidistant ≠ ∅ do
5          s_equi, h_equi ← equidistant.pop()
6          if h_equi < h_min then
7              farthest_equi ← s_equi
8          end
9          b_equi ← {s_equi}                          ▷ Initial belief
10         open ← Priority_Queue()                    ▷ Open list
11         closed ← {}                                ▷ Closed list
12         open.push(⟨s_equi, b_equi⟩)
13         closed ← closed ∪ b_equi
14         while open ≠ ∅ do
15             s, b ← open.pop()
16             if g(s) > g(s_equi) + h_equi then
17                 break
18             end
19             if b ⊨ GOAL-TEST(G_v) then
20                 return obs-sequence for s
21             end
22             for a, s' ∈ successors(s) do
23                 o ← O(a, s')
24                 b' ← Belief-Generation(b, a, o)
25                 g(s') ← g(s) + cost(a)
26                 if b' ∉ closed then
27                     open.push(⟨s', b'⟩)
28                     closed ← closed ∪ b'
29                 end
30             end
31         end
32         return obs-sequence for farthest_equi
33 procedure Belief_Generation(b, a, o)
34     b' ← {}
35     for ŝ ∈ b do
36         for â ∈ A do
37             if ŝ ⊨ pre(â) and O(â, Γ(ŝ, â)) = o then
38                 b' ← b' ∪ Γ(ŝ, â)
39             end
40         end
41     end
42     return b'
```

For each of the domains, we randomly generated 50 problem instances. For the `Blocksworld` domain, we generated problems with 4 to 8 blocks and towers of maximum height 4 for both initial and goal states. For the `Logistics` domain, we generated problems with goals consisting of 2 to 6 facts. For the `Zenotravel` domain, we generated problems with goals consisting of 2 to 6 facts. We generated 5 random candidate goals (n=5) for each problem.

In order to explore the first evaluation objective, we generated both partially and fully observable observation models for all the three domains. The partially observable models have many-to-one mapping of action-state pairs to observation symbols. For the sake of simplicity, we used lifted action names as observation symbols. Therefore, the grounded actions taken and associated states are mapped to the cor-

Domain	Metrics	Observation Models	
		Many-to-One	One-to-One
Blocksworld	Avg obfuscated length (%)	81.50	24.31
	Avg run time	46.85	58.39
	SD run time	79.95	73.24
Logistics	Avg obfuscated length (%)	84.06	62.93
	Avg run time	244.68	256.18
	SD run time	249.46	250.12
Zenotravel	Avg obfuscated length (%)	73.75	48.65
	Avg run time	216.14	224.14
	SD run time	216.49	213.68

Table 1: Empirical evaluation for different types of observation models. We report average percentage of obfuscated plan length, and the average and standard deviation of time taken (in seconds) to compute the obfuscated plan.

Domain	Metrics	Cost-bounds	
		+20%	+50%
Blocksworld	Avg obfuscated length (%)	71.90	77.75
Logistics	Avg obfuscated length (%)	73.25	83.58
Zenotravel	Avg obfuscated length (%)	65.81	69.87

Table 2: Empirical evaluation to explore the cost versus obfuscation trade-off. We examine the extent of obfuscation for different cost-bounds. We report average percentage of obfuscated plan length.

responding lifted action names. For the `Blocksworld` domain, the observation symbols were *pickup, putdown, stack, unstack*. For the `Logistics` domain, the observation symbols were *load-truck, unload-truck, load-airplane, unload-airplane, drive-truck, fly-airplane*. Finally for the `Zenotravel` domain, the symbols were *board, debark, fly, zoom, refuel*. The fully observable models have one-to-one mapping, that is the observer is aware of the actions taken and the states reached by the agent.

In order to balance the run-time between the state space search and the belief space search, we go back and forth between these two searches. For a given problem, when the heuristic value reaches some threshold (say, reduces by 50% of its value from the initial state), we start processing the states in the equidistant queue. We then run the belief space search for the states in equidistant queue until they are exhausted. Once the equidistant queue has been exhausted, we restart the state space search again to find further equidistant states exhaustively. After which, we again start with belief space search. We can modulate the heuristic threshold according to the problem types and desired objectives. This strategy can be especially helpful when the state space is large, if the planner spends all its time resource in exploring the state space, there will be no resource left for exploring the belief space. For the experiments, we kept the threshold of 50%.

4.2 Results

The evalution results are presented in Tables 1, 2 and 3. We modified the STRIPS planner pyperplan (Alkhazraji et al. 2016) to implement our algorithms. To compute equidistant states we used `lmcut` heuristic[1]. We ran the experiments with $v = 3$ for all the domains. We ran our experiments on 2.7 GHz Intel Core i7 processor with 16 GB RAM. The experiments were run with 15 minutes time-out for each problem instance. The following number of problems reached time-out before a solution could be found: 4 out of 50 problems in the `Blocksworld` domain, 19 out of 50 problems in `Logistics` and 22 out of 50 problems in `Zenotravel`. These problems were not considered in the results.

Domain	Metrics	Optimal Plan	Obfuscated Plan
Blocksworld	Avg plan cost	5.64	6.57
	Avg run time	0.11	46.85
	SD run time	0.05	79.95
Logistics	Avg plan cost	26.40	27.94
	Avg run time	14.87	244.68
	SD run time	23.77	249.46
Zenotravel	Avg plan cost	8.3	10.79
	Avg run time	10.18	216.14
	SD run time	13.03	216.49

Table 3: Empirical evaluation to explore differences in optimal plan to goal and obfuscated plan to goal. We report average plan cost and average, standard deviation of time taken in seconds to compute the solution to the goal.

Table 1 examines how the different observation models affect the length of obfuscated part of the plan solution. It is not always possible to find a completely secure goal-obfuscated plan. In such cases, we output a partial solution. Here the obfuscated solutions were found without any resource constraints. For the fully observable observation models, we see that the `Blocksworld` domain had the least obfuscated solutions. For the partially observable observation models, we see that `Logistics` domain had the most obfuscated solutions.

Table 2 evaluates the impact of constraining the additional cost on the length of obfuscated parts of the solution. We ran experiments with cost bound of 1.2 and 1.5 times the optimal cost of the true goal. By only incurring 20% additional cost, we are able to generate plans with substantial portions obfuscated. Increasing the additional cost does not dramatically improve the obfuscation performance. This highlights the importance of the observation model and candidate goal selection in the resource bounded goal obfuscation problem.

Table 3 compares the run-time and cost differences between optimal and goal-obfuscated plans. For all the three domains, the plan cost of obfuscated plans is slightly higher than optimal plans. Although the main difference lies in the amount of time taken to compute the obfuscated plans. Given there is enough time available to compute obfuscated plans, the plan cost incurred for the obfuscation guarantee is not too high.

[1]By using `lmcut` heuristic, we approximate the equidistant states since it is not the same as the actual optimal cost $h^*(\cdot)$.

5 Related Work

Planning and diagnosis have long been recognized for their inferential similarities (Sohrabi, Baier, and McIlraith 2010). When planning systems are deployed, diagnosis is seen as a mechanism to refine its models of its own actions (Kuhn et al. 2008), the environment (Molineaux, Kuter, and Klenk 2012), and of adversary capabilities (Birnbaum et al. 1990). The combination of planning, execution, diagnosis, and learning underpins much of the research in the Metacognition (Cox 2005) and Goal Reasoning (Aha 2018) communities. From this perspective, our work is an extension to understanding how planning, goal reasoning, and execution interact in an adversarial world.

Recently, there has been some interest in exploring notions of privacy preservation, deception, etc., in adversarial scenarios (Kulkarni, Srivastava, and Kambhampati 2018; Pozanco et al. 2018; Masters and Sardina 2017; **?**; Keren, Gal, and Karpas 2016b). In Kulkarni, Srivastava, and Kambhampati (2018), the authors introduce the problem of goal obfuscation and provide a satisficing solution to the problem, which is not guaranteed to be secure. In Pozanco et al. (2018), the authors explore a setting, where an agent plans to prevent another agent from achieving certain goal. In this case, both the agents are actively planning towards a goal. However the authors do not use any obfuscating strategy to hide their intent of blocking. In Masters and Sardina (2017), the authors present an approach to obfuscate goals by making one goal more likely than the other, however their approach does not support deception when the adversary knows their algorithm. In Keren, Gal, and Karpas (2016b), the authors obfuscate a goal with only one candidate goal that shares maximal non-distinct path, obfuscating part of the plan. Most of these works use goal-recognition/plan-recognition modules (Ramırez and Geffner 2009; 2010; E-Martin, R-Moreno, and Smith 2015; Sohrabi, Riabov, and Udrea 2016; Keren, Gal, and Karpas 2016a) to make inferences about the agent's goals and to achieve deception. However, using a goal recognition module will result in the system being prone to making assumptions about the observer's goal-recognition capabilities. On the other hand, we do not make any assumptions about the observer's goal recognition capability but instead provide obfuscation for the worst case adversary. We also explore the obfuscation versus cost trade-off in such scenarios. We do this by computing a cheapest solution to a goal obfuscation problem that is secure and follows certain standard cryptography assumptions.

6 Conclusion

In this work, we presented an approach to compute secure goal-obfuscated plans to conceal the true goal of an AI agent from an adversarial observer. Our approach provides a cost-efficient solution that satisfies four standard cryptographic assumptions to ensure that goal obfuscated plan is secure. Under these assumptions, the AI agent may incur an additional cost to reach its true goal, but the adversary will not be able to diagnose the true goal of the agent. Depending on the resource budget of the agent, it can modulate the amount of obfuscation it needs. A secure goal-obfuscated solution is achieved by obfuscating the true goal of the agent with v-1 other candidate goals. In order to choose these candidate goals, we introduce a goal similarity measure that aids in selecting goals that share common landmarks with the true goal. After selecting the set of decoy goals, we perform a state space search and followed by a bounded length belief space search to compute a secure goal-obfuscated solution. We present some theoretical guarantees on the goal obfuscation mechanism, and evaluate our approach using three IPC domains, `BlocksWorld`, `Logistics` and `Zenotravel`, to show the feasibility and usefulness of our approach.

This work opens two sets of research questions. Regarding the algorithms, the approach presented here is greedy in that the initial selection of v goals is not revisited. While exploring all possible sets of goals is computationally intractable, additional work should define methods with intelligent backtracking and improved heuristics. Furthermore, it assumes existence of equidistant states, relaxing this assumption while still identifying obfuscated plans is important for real world domains. Regarding a deployment setting, multi-agent domains consist of potential collaborators and adversaries all operating with different goals and time-horizons. Therefore, it is necessary for the agent itself to control the degree of obfuscation desired from each agent and how that changes over time. In this paper, we introduce the obfuscation utility trade off and present an algorithm to select obfuscated plans with guarantees and bounds.

References

Aha, D. W. 2018. Goal reasoning: Foundations, emerging applications, and prospects. *AI Magazine* 39(2).

Alkhazraji, Y.; Frorath, M.; Grtzner, M.; Liebetraut, T.; Ortlieb, M.; Seipp, J.; Springenberg, T.; Stahl, P.; and Wlfing, J. 2016. Pyperplan. https://bitbucket.org/malte/pyperplan.

Birnbaum, L.; Collins, G.; Freed, M.; and Krulwich, B. 1990. Model-based diagnosis of planning failures. In *AAAI*, volume 90, 318–323.

Bryce, D. 2014. Landmark-based plan distance measures for diverse planning. In *ICAPS*.

Cox, M. T. 2005. Field review: Metacognition in computation: A selected research review. *Artif. Intell.* 169(2):104–141.

E-Martin, Y.; R-Moreno, M. D.; and Smith, D. E. 2015. A fast goal recognition technique based on interaction estimates. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Keren, S.; Gal, A.; and Karpas, E. 2016a. Goal recognition design with non-observable actions. In *AAAI*, 3152–3158.

Keren, S.; Gal, A.; and Karpas, E. 2016b. Privacy preserving plans in partially observable environments. In *IJCAI*, 3170–3176.

Kuhn, L. D.; Price, B.; De Kleer, J.; Do, M. B.; and Zhou, R. 2008. Pervasive diagnosis: The integration of diagnostic goals into production plans. In *Aaai*, 1306–1312.

Kulkarni, A.; Srivastava, S.; and Kambhampati, S. 2018. A unified framework for planning in adversarial and cooperative environments. In *ICAPS Workshop on Planning and Robotics*.

Lindell, Y. 2005. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*. IGI Global. 1005–1009.

Masters, P., and Sardina, S. 2017. Deceptive path-planning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 4368–4375.

Molineaux, M.; Kuter, U.; and Klenk, M. 2012. Discoverhistory: Understanding the past in planning and execution. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 989–996. International Foundation for Autonomous Agents and Multiagent Systems.

Pozanco, A.; E-Martn, Y.; Fernndez, S.; and Borrajo, D. 2018. Counterplanning using goal recognition and landmarks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 4808–4814. International Joint Conferences on Artificial Intelligence Organization.

Ramırez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proceedings of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc*, 1778–1783.

Ramırez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI 2010)*.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2010. Diagnosis as planning revisited. In *KR*.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan recognition as planning revisited. In *IJCAI*, 3258–3264.

Štolba, M. 2017. *Reveal or Hide: Information Sharing in Multi-Agent Planning*. Ph.D. Dissertation, Czech Technical University.