Plans that Remain Private, Even in Hindsight

Yulin Zhang and **Dylan A. Shell**
Department of Computer Science and Engineering
Texas A&M University, College Station, TX 77843
yulinzhang | dshell@tamu.edu

Abstract

We study a model in which a robot executes a plan whilst being observed. The observer is interested in learning about the robot and its environment, and processes a sequence of symbols as evidence of the robot's actual execution. The observer uses this sequence to draw inferences about the robot's behavior, including the use of later parts of the evolution to clarify uncertainty present in earlier parts. We say an estimator that performs this sort of smoothing uses 'hindsight.' We give an algorithm which examines whether a given plan solves the planning problem (i.e., is guaranteed to attain a goal state) while the information learned by the observer satisfies a given privacy stipulation. Also, we provide an efficient incremental algorithm that permits search for such private plans. This enables our previous results in planning subject to a filtering observer to be extended to a smoothing one too, for an important class of privacy stipulations.

Introduction

As autonomous robots become part of our daily lives, the information they collect, including what is needed for them to function correctly, can be both sensitive and valuable. This information may be leaked in a variety of ways, such as robot's status display, logged data, direct observations of actions executed, etc. This paper examines planning subject to constraints on information that may be leaked.

Previously, we examined what information could be learned from observation of plan execution by a filtering adversary, namely, one who uses its current estimate and the latest observation to construct a new estimate for the current time [11, 12]. That work gave algorithms to search for plans which satisfy formal stipulations on information disclosed to adversaries, under different assumptions about the adversary's prior knowledge and how robot's actions and observations are disclosed. This paper considers a different kind of adversary: one who may construct estimates for any past time by playing back all its received observations. Unlike the filtering adversary who is interested in the current state and hence only looks forward, the playback adversary may wish to learn more about its past with the power of hindsight, operating like a smoother.

Figure 1 gives a small scenario that shows how the playback adversary is capable of violating a user's privacy, whereas a filtering adversary, being less powerful, can not. Consider a robot collecting an item from the mailbox outside, and the robot's plan (essentially its navigation route) is described in the caption. Further, the robot simply writes a symbol 'a' to the log if it takes an action, and writes a symbol 'o' when an observation is received. Suppose that the robot is taken to the maintenance department because it needs an upgrade. The maintenance staff have full access to the robot's log and plan, but the owner desires that the presence or absence of a house guest is never divulged.

The filtering adversary described in [11] never learns whether there was a guest in the house (because it only has a stream of alternating a's and o's). But the playback adversary knows that the robot was in the master bedroom because there was a guest, if the robot came out from the bedroom and did not terminate—which is learned later by considering the total length of the completed sequence. This paper formalizes and examines how to verify whether a stipulation is satisfied on the information leaked to such playback adversaries from the plan, and provides an incremental algorithm to search for a plan that will respect privacy stipulations.

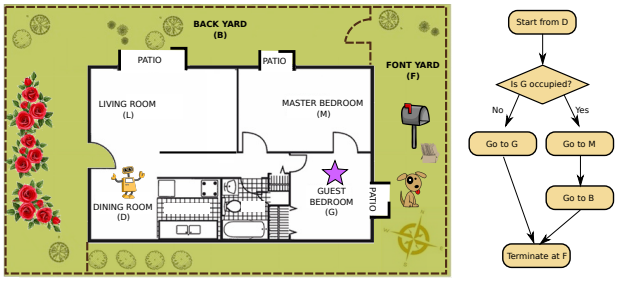

Figure 1: To pick up a newly delivered item from the mailbox, a robot navigates from the living room to the front yard. The robot can either take a shortcut to enter through the guest bedroom, or go through the master bedroom to the front yard via the back yard. But the robot should be obeisant and refrain from entering the guest bedroom when someone is inside, which is signified by the star.

Problem Description

We examine the three-way relationships shown in Figure 2: a robot follows a *plan* to interact with the *world*, mediated via a stream of observations and actions, so as to achieve some goal. Both the plan and action–observation stream may be partially disclosed to a third party, termed the *observer*. The observer is able to play back the stream, and combine it with its knowledge about the robot's plan, so as to infer properties of the interaction. We are interested in examining whether the stipulations, especially privacy stipulations, are satisfied on what can be learned by the observer. The problem will be formalized in terms of procrustean graphs and label maps.

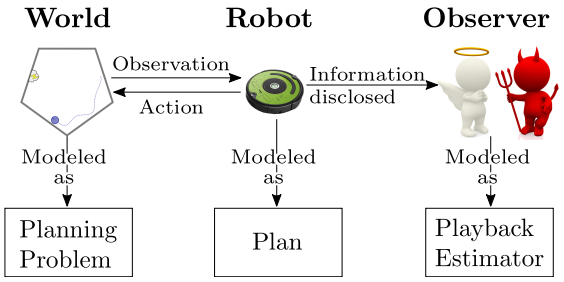

Figure 2: The robot is modeled as a plan to achieve some goal in the world, and the observer is modeled as an estimator which can play back the received action–observation stream.

Procrustean graphs, planning problem and plans

We model the world with a procrustean graph (p-graph), which is an edge-labeled bipartite directed graph capturing the robot's interactions with its environment [8]. A robot receives observations through sensors from its environment, and then takes actions in the world. A stream of alternating actions and observations, also called a *string* or an *execution*, causes transitions in the p-graph. Hence, the states of the p-graph are partitioned into action states and observation states. For action (observation) states, the set of outgoing edges are labeled with actions (observations). We denote the set of vertices in a p-graph G by $V(G)$, the set of all actions and observations be $U(G)$ and $Y(G)$, respectively. Each p-graph has a set of initial states, denoted as $V_0(G) \subseteq V(G)$. The set of strings that can be successfully traced in G is called the language of G, denoted $\mathcal{L}(G)$. For any string $s \in \mathcal{L}(G)$ of length k, it gives a sequence of $k+1$ states when traced in G. We call this sequence of states a *state trajectory* consistent with s, and denote all state trajectories consistent with s in G as $\textsc{Traj}_s^G$.

A *planning problem* is a p-graph W with a set of goal states V_{goal}. A plan is a p-graph P with a set of termination states V_{term}. When the plan (P, V_{term}) is executed on the planning problem (W, V_{goal}), the plan P receives observations from the world W and then chooses actions to execute on W. We say that a plan (P, V_{term}) *solves* a planning problem (W, V_{goal}) if (i) the plan P can handle every observation from the world W, (ii) every action that the plan takes is available in the world W and, (iii) when the plan terminates, the current state of the world is within V_{goal}. For example, a plan that guides the robot from its initial position to the mailbox is shown in Figure 3.

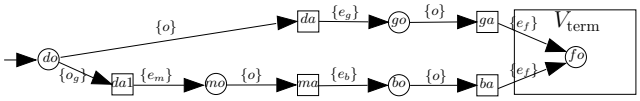

Figure 3: The plan that solves the planning problem shown in Figure 1. The first letter 'd', 'm', 'b', 'g', 'f' indicates that the robot is in the dinning room, master bedroom, back yard, guest bedroom, or front yard. Action e_m represents entering master bedroom. The observation o_g represents that the guest bedroom is occupied. Otherwise, it gives a generic observation o.

Information disclosure and observer

The robot's interaction with the world generates a stream of actions and observations. Such a stream may be disclosed to the observer only imperfectly, either by design or as a consequence of real-world fallibility: multiple strings in the world may look the same to the observer. We use a label map h to model *information disclosure*. The label map h is a function mapping an action or an observation to a new symbol, this new symbol being disclosed to the observer.

The observer is assumed to know the planning problem and have some knowledge about the robot's plan, which is obtained from a side-channel. The *disclosed plan* is modeled as a p-graph, which may encode knowledge that is weaker than knowing the robot's exact plan (see [12]). When receiving any disclosed symbols through the label map, the observer will (1) infer all potential interactions that could happen in the world according to its prior knowledge, (2) play back these interactions to identify all consistent state trajectories in the world and, (3) obtain the estimated world states at each time step. Formally, we have:

Definition 1 (observer's estimate)**.** Given planning problem (W, V_{goal}), robot's disclosed plan D, and information disclosure policy h, when the observer receives a string $\boldsymbol{x} = x_1 x_2 \dots x_k$,

1) the set of potential strings inferred by the observer is $\mathcal{S}_{\boldsymbol{x}} = h^{-1}(\boldsymbol{x}) \cap \mathcal{L}(W) \cap \mathcal{L}(D)$,
2) the set of trajectories consistent with those potential strings is $\mathcal{T}_{\boldsymbol{x}} = \cup_{s' \in \mathcal{S}_{\boldsymbol{x}}} \textsc{Traj}_{s'}^W$.
3) the set of estimated world states $\mathcal{W}_{\boldsymbol{x}}^m$ at time step m is extracted from these state trajectories, i.e., $\mathcal{W}_{\boldsymbol{x}}^m = \cup_{\tau \in \mathcal{T}_{\boldsymbol{x}}} \tau[m]$, where $\tau[m]$ is the m^{th} world state on trajectory τ.

In the above, the set of potential strings $\mathcal{S}_{\boldsymbol{x}}$ is the set of strings (i) each of which shares image $\boldsymbol{x}$, (ii) that is a legal execution from the world W and, (iii) may be executed by the robot's plan. The estimated trajectories are the ones that are consistent with some string in $\mathcal{S}_{\boldsymbol{x}}$. To compute these consistent trajectories, the observer has to be able to play back each string in $\mathcal{S}_{\boldsymbol{x}}$, tracing it in the world graph W. Finally, the estimated world states at a particular time are the corresponding ones from all trajectories.

To constrain what the observer learns, we write propositional formula Φ on the observer's estimate $\mathcal{W}_{\boldsymbol{x}}^t$. An atomic symbol v is created for each world state v to denote the fact that the world state is v; it is evaluated to True if and only if v included in $\mathcal{W}_{\boldsymbol{x}}^t$. The negation $\neg v$ represents $v \notin \mathcal{W}_{\boldsymbol{x}}^t$. With each symbol grounded in this way, we can write the

propositional formulas with these symbols connected by logic operators NOT, AND, OR, and evaluate them on observer's estimate.

Now, we have the following satisfaction problem.

Problem: CHECKPLAN $\big((W, V_{\text{goal}}), (P, V_{\text{term}}), h, D, \mathbf{\Phi}\big)$

Input: A planning problem (W, V_{goal}), a plan (P, V_{term}), an information disclosure h, a disclosed plan D, and a stipulation $\mathbf{\Phi}$.

Output: True if (P, V_{term}) solves the planning problem (W, V_{goal}) and, $\forall s \in \mathcal{L}(W) \cap \mathcal{L}(P)$, $\mathbf{\Phi}$ is always evaluated as True on $\mathcal{W}^m_{h^{-1} \circ h(s)}$ for all integer $0 \leq m \leq k$; False otherwise.

An algorithm to check a plan with hindsight

To solve CHECKPLAN, the key is to trace all trajectories that are consistent for the images of the strings in the plan. Unhappily there can be many strings such strings. Instead of computing the beliefs for each string from scratch, we propose an graph-based algorithm to produce the set of all beliefs for any string and its extensions. The stipulations are violated once it is violated on some string in the plan.

A p-graph representing observer's prior knowledge

First, we construct a p-graph to integrate observer's prior knowledge about the planning problem and the disclosed plan, and then compute the estimated trajectories in this new graph.

To combine the observer's prior knowledge about the world and robot's plan, we construct a product graph $J = W \otimes D$ as the tensor product graph of world W and disclosed plan D with initial states $V_0(W) \times V_0(D)$. The language of this joint graph is the set of executions that could happen in the world and could potentially be taken by the robot's plan, i.e., $\mathcal{L}(J) = \mathcal{L}(W) \cap \mathcal{L}(D)$. In addition, if we trace any string $s \in \mathcal{L}(J)$ in J, take the first part (world state) of each joint state in the trajectory TRAJ^J_s, and denote this new sequence as $\text{TRAJ}^{J,W}_s$, then we will obtain exactly the same trajectory as tracing s in the world, i.e., $\text{TRAJ}^{J,W}_s = \text{TRAJ}^W_s$.

Now, instead of computing the trajectories in the world graph, we can compute it in the joint graph J, essentially pretending that it is new 'world'. The observer's belief is a set of states in the joint graph.

A graph-based algorithm for CHECKPLAN

To solve CHECKPLAN, we need to check both the solutions for the planning problem and the stipulations on the disclosed information. A tensor product graph is constructed to examine whether the plan always terminates at a goal state. To examine the disclosed information, we give an algorithm that incrementally constructs all beliefs which are learned by the playback observer. In estimation with hindsight, the observer is able to playback the observations, refining previous beliefs by eliminating the states from which subsequent events crash. A illustration of the motivation example is shown in Figure 4: when the observer knows that the robot is currently in the backyard (B) after recieving '`oaoao`', it also knows that the robot was in the master bedroom (M) instead of the guest bedroom (G) at the previous time step, since the string '`ao`' crashes on G. Instead of playing back each string in the plan, we conduct a breadth-first search (BFS) on the belief graph to efficiently simulate the observer's estimation process as shown in Algorithm 1.

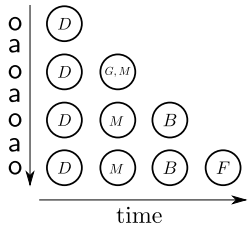

Figure 4: A snapshot of the belief history for the playback observer in Figure 1 when processing the stream '`oaoaoao`'. The vertical axis shows the observer's beliefs when part of the stream is received. The horizontal axis shows the belief at a particular time in the history.

Firstly, we construct a product graph $W \otimes P$ to check whether every termination state in P is paired with some goal state in W in the product graph (line 1–4). If not, then the plan does not solve the planning problem. Otherwise, it does.

The observer is only able to see the strings in $h\langle J \rangle$, which is obtained by replacing the labels on the edges in J with their images after label map h. In $h\langle J \rangle$, one string may reach two states non-deterministically. We construct a deterministic form J' for $h\langle J \rangle$, following Algorithm 2 in [8] (line 5–6). During this state-determined transformation, states in $h\langle J \rangle$ are merged into a single state v' in J' if they are non-deterministically reached by some string. We say that these states are the corresponding states for v'. Each state in J' is a belief state, and its corresponding states are included in the belief. Not all belief states in J' will be active and perceived by the observer, since the plan may not produce those beliefs. We construct a product graph $J' \otimes h\langle P \rangle$ to mark each belief state in J' as active, if it is paired with some plan state in the product graph (line 7–8). Stipulations will be evaluated on these active belief states once they are generated. The plan fails to satisfy the stipulations, if the stipulations are violated on any active belief.

The active belief states in J' only contribute to part of the beliefs generated by the observer. They will be refined when the observer can play back its observations. Some states in the beliefs of past times will be eliminated when there is no string as an extension of these states to the states in the frontier. We say that these states are not alive. A BFS search on J' is conducted to simulate this playback estimation (line 9–33). Starting from each active belief state V_k in the frontier of the search, we mark each corresponding state in belief V_k as alive. Then we propagate liveness backward to find the states in the past beliefs that are not alive. For each transition $V_{k-1} \xrightarrow{x} V_k$, we mark each state in the belief V_{k-1} as alive if

Algorithm 1 $CheckPlan((W, V_{\text{goal}}), (P, V_{\text{term}}), h, D, \mathbf{\Phi})$

1: $K = W \otimes P$
2: **for** $(w, r) \in K.vertices$ **do**
3: **if** $r \in V_{\text{term}}$ and $w \notin V_{\text{goal}}$ **then**
4: **return** False
5: $J \leftarrow W \otimes D$
6: $J' \leftarrow STD(h\langle J \rangle)$
7: $Q \leftarrow J' \otimes h\langle P \rangle$
8: $active\_v = \pi_{J'}(Q.vertices)$
9: $q \leftarrow [J'.initVertex]$
10: $visited = []$
11: **while** q is not empty **do**
12: $m \leftarrow q.pop()$
13: add m to $visited$
14: **if** $m \notin active\_v$ **then**
15: **continue**
16: **if** m.correspState violates stipulation $\mathbf{\Phi}$ **then**
17: **return** False
18: $p \leftarrow []$
19: **for** $(n, x) \in J'.incoming(m)$ **do**
20: add (n, x, m) to p
21: $p \leftarrow [(n, x, m)]$
22: **while** p is not empty **do**
23: $(v', x, v) \leftarrow p.pop()$
24: $b_v \leftarrow v.correspStates$
25: $b_{v'} \leftarrow refine(J, v'.correspStates, x, b_v)$
26: **if** $b_{v'}$ violates stipulation $\mathbf{\Phi}$ **then**
27: **return** False
28: **if** $b_v! = b_{v'}$ and $v' \notin J.initStates$ **then**
29: **for** $(u, x) \in J'.incoming(v')$ **do**
30: add (u, x, v') to p
31: **for** $w \in m.children()$ **do**
32: **if** $w \notin visited$ **then**
33: add w to q
34: **return** True

the state transitions to some state in V_k under x in $h\langle J \rangle$. Otherwise, we mark that it is not alive. We refine belief V_{k-1} by removing all states that are not alive, and construct a new belief V'_{k-1} (line 23–25). When none of the states in belief V_k are eliminated, i.e., $V'_{k-1} = V_{k-1}$, then we may stop propagating the liveness, since no new beliefs will be generated. If V'_{k-1} is finer than V_{k-1}, then one must keep propagating the liveness in V'_{k-1} backward (line 28–30). Stipulations must be evaluated on the refined belief states when they are generated (line 26–27). The evaluation can stop early when one of these beliefs violates the stipulations.

An incremental algorithm to search for a plan

We are interested in seeking plans that never disclose information to playback observers that violate given stipulations:

Problem: SEARCHPLAN $\big((W, V_{\text{goal}}), D, h, ?, \mathbf{\Phi}\big)$
Input: A planning problem (W, V_{goal}), a disclosed plan D, an information disclosure h, a stipulation $\mathbf{\Phi}$.
Output: A plan (P, V_{term}), such that CHECKPLAN$\big((W, V_{\text{goal}}), D, h, P, \mathbf{\Phi}\big) = \text{True}$.

As mentioned in the satisfaction problem, J' captures the observer's beliefs. We are interested in searching for a plan which reaches the goal in W and always generates beliefs in J' that satisfy the stipulations. To do this, we construct a product graph of $h^{-1}\langle J' \rangle$ and W, denoted as $T = h^{-1}\langle J' \rangle \otimes W$. The joint state in T consists of two parts: states in $h^{-1}\langle J' \rangle$ to examine stipulations and states in W to examine the goal condition. Next, we conduct a AND-OR search on T to find a subgraph such that (i) each action state has a single outgoing edge bearing one action, (ii) each observation state has outgoing edges bearing all observations in the world, (iii) all beliefs generated from J' in the subgraph must satisfy stipulations, and (iv) the states eventually terminating in the subgraph must give world states all of which are in the goal region. When constructing the AND-OR search, we are able to incrementally search for an action for each action state in the subgraph, and obtain a partial plan. By calling the CHECKPLAN procedure, we are able to examine whether stipulations are satisfied on all generated beliefs of the partial plan. If the partial plan fails to satisfy the stipulations, then one must backtrack the action choice just made and choose a different action. This process is repeated until a solution is found.

Related work

In the AI and robotics communities, autonomous agents are modeled as transducers that receive observations from the environment and choose actions to influence the world. Information such as an agent's actions and observations could be leaked to adversaries, which will raise privacy concerns. Deceptive actions [2, 6] and goal obfuscated plans [3, 4] have been adopted to hide an agent's true goal for as long as possible. Sensor configurations [10], plans [11], and policies affecting how information is disclosed from the agents [9, 7] have also been exploited to meet both privacy and utility requirements. In multi-agent settings, privacy-preserving algorithms have also been developed to coordinate distributed plan search [1, 5].

Summary and future work

We examined the information divulged from plans to an observer with hindsight, and described how an observer may construct its estimates by combining its prior knowledge with its observations of the robot's plan execution. This algorithm enables us to incrementally search for plans that satisfy privacy stipulations, by extending our previous planning algorithm subject to a filtering observer. In the future, we aim to search for plans and information disclosure policies jointly so that stipulations are satisfied on a playback observer.

Acknowledgement

This work was supported by the NSF through awards IIS-1453652 and IIS-1527436.

References

[1] Brafman, R. I. 2015. A privacy preserving algorithm for multi-agent planning and search. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[2] Dragan, A. D.; Holladay, R.; and Srinivasa, S. S. 2015. Deceptive Robot Motion: Synthesis, Analysis and Experiments. *Autonomous Robots* 39(3):331–345.

[3] Kulkarni, A.; Klenk, M.; Rane, S.; and Soroush, H. 2018. Resource bounded secure goal obfuscation. In *AAAI Fall Symposium on Integrating Planning, Diagnosis and Causal Reasoning*.

[4] Kulkarni, A.; Srivastava, S.; and Kambhampati, S. 2019. A unified framework for planning in adversarial and cooperative environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[5] Maliah, S.; Shani, G.; and Stern, R. 2017. Collaborative privacy preserving multi-agent planning. In *Proceedings of International Conference on Autonomous Agents and Multi-agent Systems*, volume 31, 493–530.

[6] Masters, P., and Sardina, S. 2017. Deceptive Path-Planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 4368–4375.

[7] Mohajerani, S.; Ji, Y.; and Lafortune, S. 2018. Efficient synthesis of edit functions for opacity enforcement using bisimulation-based abstractions. 4849–4854.

[8] Saberifar, F. Z.; Ghasemlou, S.; Shell, D. A.; and O'Kane, J. M. 2019. Toward a language-theoretic foundation for planning and filtering. *International Journal of Robotics Research—in WAFR'16 special issue* 38(2-3):236–259.

[9] Wu, Y.-C.; Raman, V.; Lafortune, S.; and Seshia, S. A. 2016. Obfuscator synthesis for privacy and utility. In *NASA Formal Methods Symposium*, 133–149.

[10] Zhang, Y., and Shell, D. A. 2019. Complete characterization of a class of privacy-preserving tracking problems. *International Journal of Robotics Research—in WAFR'16 special issue* 38(2-3):299–315.

[11] Zhang, Y.; Shell, D. A.; and O'Kane, J. M. 2018a. Finding plans subject to stipulations on what information they divulge. In *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*.

[12] Zhang, Y.; Shell, D. A.; and O'Kane, J. M. 2018b. What does my knowing your plans tell me? In *IEEE/RSJ IROS Workshop—Towards Intelligent Social Robots: From Naive Robots to Robot Sapiens*.