What does my knowing your plans tell me?

Yulin Zhang
Dept. of Comp. Sci. & Engr.
Texas A&M University
College Station, Texas, USA
yulinzhang@tamu.edu

Dylan A. Shell
Dept. of Comp. Sci. & Engr.
Texas A&M University
College Station, Texas, USA
dshell@tamu.edu

Jason M. O'Kane
Dept. of Comp. Sci. & Engr.
University of South Carolina
Columbia, South Carolina, USA
jokane@cse.sc.edu

***Abstract*—For robots acting in the presence of observers, we examine the information that is divulged if the observer is party to the robot's plan. Privacy constraints are specified as the stipulations on what can be inferred during plan execution. We imagine a case in which the robot's plan is divulged beforehand, so that the observer can use this *a priori* information along with the disclosed executions. The divulged plan, which can be represented by a procrustean graph, is shown to undermine privacy precisely to the extent that it can eliminate action-observation sequences that will never appear in the plan. Future work will consider how the divulged plan might be sought as the output of a planning procedure.**

I. Introduction

Autonomous robots are beginning to be part of our everyday lives. Robots may need to collect information to function properly, but this information can be sensitive if leaked. In the future, robots will not only need to ensure physical safety for humans in shared workspaces, but also to guarantee their information security. But information leakage can occur in a variety of ways, including through logged data, robot's status display, actions, or, as we examine, through provision of prior information about a robot's plan.

Established algorithmic approaches for the design and implementation of planners may succeed at selecting actions to accomplish goals, but they fail to consider what information is divulged along the way. While several models for privacy exist, they have tended to be either abstract definitions applicable to data rather than an agent operating autonomously in the world (such as encryption [1], data synthesis [2], anonymization [3], or opacity [4] mechanisms) or are focussed on a particular robotic scenario (such as robot division of labor [5] or tracking [6, 7]).

Figure 1 illustrates a scenario where the information divulged is subtle and important. It considers an autonomous wheelchair that helps a patient who has difficulty navigating by himself. The user controls the wheelchair by giving voice commands: once the user states a destination, the wheelchair navigates there autonomously. While moving through the house, the wheelchair should avoid entering any occupied bedrooms, making use of information from motion sensors installed inside each bedroom. We are interested in stipulating the information divulged during the plan execution:

- Positive disclosure of information: A therapist monitors the user, ensuring that he adheres to his daily regimen of activity, including getting some fresh air everyday (by visiting the front yard or back yard).
- Negative disclosure of information: However, if there is a guest in one of the bedrooms, the user does not want to disclose the guest's location.

Actions, observations, and other information (such as the robot's planned motion) may need to be divulged to satisfy the first (positive) stipulation. The challenge is to satisfy both stipulations simultaneously. Suppose the robot executes the plan shown in the right of Fig. 1, and that this plan is public knowledge. If, as it moves about, the robot's observations (or actions) are disclosed to an observer, then we know that the robot will attempt to see if M is occupied. Hence, on some executions, a third party, knowing there is a guest, would be able to infer that they're in the master bedroom.

This paper examines in detail how divulging the plan, as above, provides information that permits one to draw inferences. In particular, we are interested in how this plan information might cause privacy violations. As we will see, the divulged plan need not be the same as the plan being executed, but they must agree in a certain way. In our future work, we hope to answer the question of how to find pairs of plans (one be to executed and one to divulged), where there is some *gap* between the two, so that information stipulations are always satisfied.

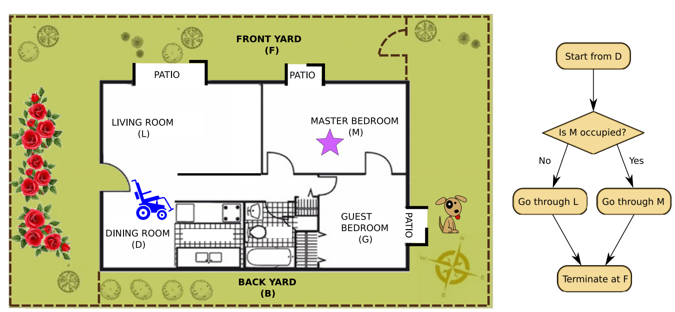

Fig. 1: An autonomous wheelchair navigates in a home. A plan, on the right, generates actions that depend on perception of the pink star (denoting that the bedroom is occupied).

II. Problem Description

In this problem, there are three entities: a *world*, a *robot*, and an *observer*. As shown in Fig. 2, the robot interacts with the world by taking observations from the world as input, and

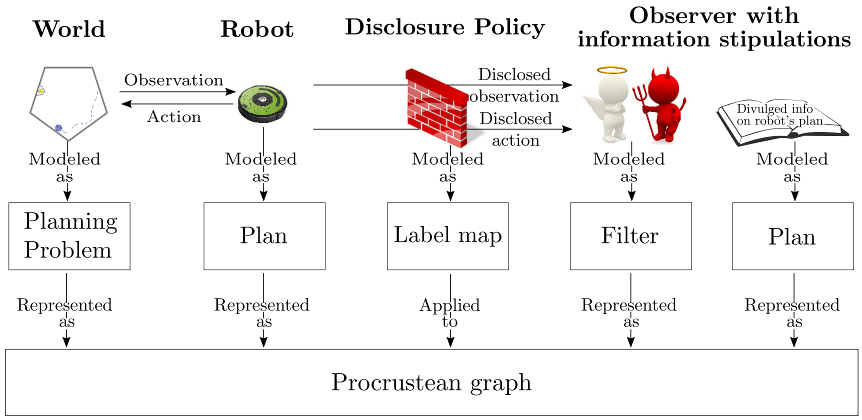

Fig. 2: An overview of the setting: the robot is modeled abstractly as realizing a plan to achieve some goal in the world and the third party observer as a filter with divulged plan as its prior knowledge. All four, the world, the plan, the filter, and the divulged plan have concrete representations as p-graphs.

outputting an action to influence the world state. This interaction generates a stream of actions and observations, which may be perceived by the observer, though potentially only in partial or diminished form. We model the stream as passing through a function which, via conflation, turns the stream generated by the world–robot interaction into one perceived by the observer, the disclosed action-observation stream. As a consequence of real-world imperfections (possible omission, corruption, or degradation) or due to explicit design, the observer, thus, may receive less information. For this reason, the function is viewed as a sort of barrier, and we term it an *information disclosure policy*.

The observer is assumed to be unable to take actions to interact with the world directly—a model that is plausible if the observer is remote, say a person or service on the other side of a camera or other Internet of Things device. Given its perception of the interaction, the observer estimates the plausible action-observation streams, consistent with the disclosed action-observation stream. This estimate can be made 'tighter' by leveraging prior knowledge about the robot's plan. The observer's estimate is in terms of world states, so the notion of tightness is just a subset relation. In this paper, we will introduce stipulations on these estimated world states and our main contribution will be in examining how the divulged plan could affect the satisfaction of these stipulations.

A. Representation

To formalize such problem, we represent these elements with p-graph formalism and label map [8]. The world is formalized as a planning problem (W, V_{goal}), where W is a p-graph in state-determined form (see definition of state-determined in [8, Def. 3.7]) and V_{goal} is the set of goal states. The robot is modeled as a plan (P, V_{term}), where P is a p-graph and V_{term} specifies the set of plan states where the plan could terminate. The plan solves the planning problem when the plan can always safely terminate at the goal region in finite number of steps (see definition of solves in [8, Def. 6.3]). The information disclosure policy is represented by a label map h, which maps from the actions and observations from W and D to an image space X. The observer is modeled as a tuple (I, D), where I is a filter represented by a p-graph with edge labels from X, D is the p-graph representing the divulged plan with actions and observations labeled in the domain of h. The plan in D might be less specific than the actual plan P, representing 'diluted' knowledge of the plan; to capture this, we require that all possible action-observation sequences (called executions for short) in D should be a superset of those in P, denoted as $\mathcal{L}(D) \supseteq \mathcal{L}(P)$ (the set of executions is called the language, see [8, Def. 3.5], hence the symbol $\mathcal{L}(\cdot)$).

B. The observer's estimation of world states

Given any set of filter states B from filter I, the observer obtains an estimate of the executions that should've occurred to reach B, through a combination of the following sources of information [9, Def. 13]:

1) The observer can ask: What are all the possible executions, each of which has its image, reaching exactly B in the filter? The set of executions reaching exactly B is represented as $\mathbb{S}^I_B$. The preimages of $\mathbb{S}^I_B$, which we denote as $h^{-1}[\mathbb{S}^I_B]$, are the executions which are responsible for arriving at B in I.
2) The observer can narrow down the estimated executions to the ones that only appear in the divulged plan D. The set of all executions in D are represented by its language $\mathcal{L}(D)$.
3) Finally, the estimated executions can be further refined by considering those that appear in the world, i.e., $\mathcal{L}(W)$.

Hence, $h^{-1}[\mathbb{S}^I_B] \cap \mathcal{L}(W) \cap \mathcal{L}(D)$ represents a tight estimation of the executions that may happen. This allows us to find the estimated world states, defined as $\mathcal{W}^D_B$, by making a tensor product T of graph W, D and $h^{-1}\langle I \rangle$, where $h^{-1}\langle I \rangle$ is obtained by replacing each action or observation ℓ with its preimage $h^{-1}(\ell)$ on the edges of the p-graph I. For any vertex (w, d, i) from the product graph T, we have:

$$\mathcal{W}^D_B = \mathcal{W}^D_B \cup \{w\}, \text{if } i \in B.$$

C. Information stipulations on the estimated world states

Information stipulations are written as propositional formulas on estimated world states $\mathcal{W}_B^D$. Firstly, we will define a symbol $\boldsymbol{w}$ for each world state w in W. Then we can use connectives $\neg$, $\wedge$, $\vee$ to form composite expressions $\boldsymbol{\Phi}$ that stipulate the estimated world states involving these symbols. The propositional formulas can be evaluated based on the following definition:

$$\boldsymbol{w} = \text{True} \quad \text{if and only if} \quad w \in \mathcal{W}_B^D.$$

With all the elements defined above, we are able to check whether the stipulation $\boldsymbol{\Phi}$ is satisfied on every estimate $\mathcal{W}_B^D$, given the world graph W, information disclosure policy h, and the observer (I, D).

III. The observer's prior knowledge of the robot's plan

The divulged plan D is known by the observer prior to the robot's monitoring of the disclosed action-observation stream. Depending on how much the observer knows, there are four possibilities, from most-to least-informed:

I) The observer knows the exact plan P to be executed.

II) The plan to be executed can be hidden among a (non-empty) finite set of plans $\{P_1, P_2, \ldots, P_n\}$.

III) The observer may only know that the robot is executing *some* plan, that is, the robot is goal directed and aims to achieve some state in V_{goal}.

IV) The observer knows nothing about the robot's execution other than that it is on W.

A p-graph exists whose language expresses knowledge for each of these cases:

Case I. When $D = P$, the interpretation is straightforward: the observer tracks the states of the plan given the stream of observations (as best as possible, as the operation is under h).

Case II. If instead a set of plans $\{P_1, P_2, \ldots, P_n\}$ is given, we must construct a single p-graph, D, so that $\mathcal{L}(D) = \mathcal{L}(P_1) \cup \cdots \cup \mathcal{L}(P_n)$. This is achieved via the union of p-graphs $D = P_1 \uplus P_2 \uplus \cdots \uplus P_n$, cf. [8, Def. 3.6, pg. 18].

Case III. If the robot is known only to be executing some plan, we must consider the set of all plans, $P^\infty := \{P_1, P_2, P_3, \ldots, \}$. As the notation hints, there can be an infinite number of such plans, so the approach of unioning plans won't work. Fortunately, another structure, P^*, exists such that $\mathcal{L}(D) = \mathcal{L}(P^*) = \mathcal{L}(P^\infty)$, which will be proved afterwards. Here P^*, a finite p-graph, is called the *plan closure*.

Case IV. When taking $D = W$ the executions are, again, intersected with $\mathcal{L}(D)$ but as they already came from $\mathcal{L}(W)$, this shows why the observer is the least informed in the hierarchy.

Next, we will show the construction of the plan closure P^* and prove that $\mathcal{L}(P^*) = \mathcal{L}(P^\infty)$.

To start, we describe construction of P^*. The initial step is to convert W to its state-determined form $W' = \textsc{Sde}(W)$ (this is an operation described in [8, Algorithm 2, pg. 30]). Then, to decide whether a vertex in W' exists in some plan, we iteratively color each vertex green, red, or gray. Being colored green means that the vertex exists in some plan, red means that the vertex does not exist in any plan, and gray indicates that its status has yet to be decided. To start with, we initially color the goal vertices green, and non-goal leaf vertices (with no edges to other vertices) red. Using the iconography of [8], we show action vertices as squares and observation vertices as circles. Then gray vertices of each type change their color by iterating the following steps:

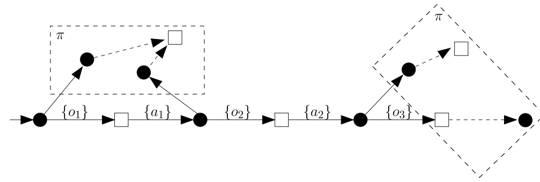

Fig. 3: The construction of a plan generating execution s using π, computed as part of Algorithm 1.

- □ → ■: $\exists$ some action a reaching ●, which is not an initial state.
- □ → ■: $\forall$ action a reaching ●.
- ○ → ●: $\forall$ observation o reaching ■, which is not an initial state.
- ○ → ●: $\exists$ some observation o reaching ■.

The iteration ends when no vertex changes its color. The subgraph that consisting of only green vertices and their corresponding edges is P^*. And P^* then contains only the vertices that exist in some plan leading to the goal states. For further detail of this algorithm for building P^*, we refer the reader to Algorithm 1.

Next, we prove that the P^* constructed from this procedure has the same language as P^∞. The proof shows that any green vertex is on some plan, by showing that we we can construct a plan π, that will lead to a goal state within a finite number of steps form any such vertex.

Lemma 1. *$\mathcal{L}(P^*)$=$\mathcal{L}(P^\infty)$.*

Proof. $\supseteq$: For any $s = s_0 s_1 s_2 \ldots s_k \in \mathcal{L}(P^\infty)$, according to the definition of P^∞, s is in the execution of some plan P'. Though s_k may not be a goal, using P', s can be extended: $\exists s' = s_0 s_1 \ldots s_k t_0 t_1 \ldots t_n \in \mathcal{L}(P')$, $k > 0, n \geq 0$ to reach an element of V_{goal}. Then $\mathcal{V}_{s'}^{P'}$ comprises vertices associated with those in W' marked green in V'_{goal}. And, tracing the execution s' on P' backwards on W', we find every vertex green back to a start vertex. But this means they are in P^*, and hence $s' \in \mathcal{L}(P^*)$, means $s \in \mathcal{L}(P^*)$ as well.

$\subseteq$: For any execution $s = s_0 s_1 s_2 \ldots s_k \in \mathcal{L}(P^*)$, s reaches V'_{goal}, or s is a prefix of some execution reaching V'_{goal} in W'. We show that there is a plan that can produce s. The execution s does not include enough information to describe a plan because: (1) it may not reach V'_{goal} itself, and (2) it gives an action after some observation that was revealed, but not every possible observation. To address this shortfall, we will capture some additional information during the construction of P^*, which we save in π. This provides an action that makes some progress, for states that can result from other observations. Now, using s as a skeleton, construct plan where once a transition outside of s occurs, either owing to an unaccounted-for observation or having reached the end of s, the plan reverts to using the actions that π prescribes. (See Fig. 3 for a visual example.) This is always possible because states arrived at in W' under s are green. This implies that all states in W are also assured to reach a goal states. The resulting plan can produce s, so some plan produces s, hence $s \in \mathcal{L}(P^\infty)$. □

Algorithm 1: P^*CONSTRUCTION(W, V_{goal})

Initialize queues red, green, gray as empty
$W' \leftarrow$ SDE(W), and initialize V'_{goal} as the associated vertices of V_{goal}
Initialize plan π as empty
for $v \in V(W')$ **do**
  if $v \in V'_{\text{goal}}$ **then**
    green.append(v)
  else if v has no edges to other vertices **then**
    red.append(v)
  else
    gray.append(v)
Q.extend(InNeighbor(red $\cup$ green)$\setminus$(red $\cup$ green))
while Q not empty **do**
  $v \leftarrow Q$.pop
  flag $\leftarrow$*True*
  if v is a ○ **then**
    if one of its outgoing neighbors is ■ **then**
      red.append(v)
    else if all of its outgoing neighbors are ■ **then**
      green.append(v)
    else
      flag $\leftarrow$*False*
  else if v is a □ **then**
    if one of its outgoing neighbors under label a is ● **then**
      green.append(v) and $\pi[v] = a$
    else if all of its outgoing neighbors are ● **then**
      red.append(v)
    else
      flag $\leftarrow$*False*
  if flag **then**
    Q.extend(InNeighbor(v)$\setminus\{$red $\cup$ green$\}$)
$P^* \leftarrow$ subgraph(W', green)
return P^* (and also π, if desired)

Thus, one may use $D = P^*$, for Case III.

IV. Experimental results

We implemented the algorithms with Python, and execute them on a OSX laptop with a 2.4 GHz Intel Core i5 processor. To experiment, we constructed a p-graph representing the world in Fig. 1 with 12 states, and the plan with 8 states. All the experiments are finished within 1 second. The information disclosure policy maps all actions to the same image, but observations to different images. As we anticipated, the stipulations are violated when the exact plan is divulged. But we can satisfy the stipulations by disclosing less information, such as $D = W$.

V. Summary and future work

We examine the planning problem and the information divulged within the framework of procrustean graphs. In particular, the divulged plan can be treated uniformly in this way, despite representing four distinct cases. The model was evaluated, showing that divulged plan information can prove to be a critical element in protecting the privacy of an individual. In the future, we aim to automate the search for plans: given P to be executed, find a D to be divulged, where $\mathcal{L}(D) \supsetneq \mathcal{L}(P)$, such that the privacy stipulations are always satisfied.

Acknowledgements

This work was supported by the NSF through awards IIS-1453652, IIS-1527436, and IIS-1526862. We thank the anonymous reviewers for their time and valuable comments.

References

[1] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. CRC Press, Inc., 1996.

[2] D. B. Rubin, "Discussion of Statistical Disclosure Limitation," *Journal of Offical Statistics*, vol. 9, no. 2, pp. 461–468, 1993.

[3] C. Dwork, "Differential privacy: A survey of results," in *Proceedings of International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[4] 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.

[5] A. Prorok and V. Kumar, "A macroscopic privacy model for heterogeneous robot swarms," in *Proceedings International Conference on Swarm Intelligence*. Springer, 2016, pp. 15–27.

[6] J. M. O'Kane, "On the value of ignorance: Balancing tracking and privacy using a two-bit sensor," in *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, 2008, pp. 235–249.

[7] Y. Zhang and D. A. Shell, "Complete characterization of a class of privacy-preserving tracking problems," *International Journal of Robotics Research—in WAFR'16 Special Issue*, 2018.

[8] F. Z. Saberifar, S. Ghasemlou, D. A. Shell, and J. M. O'Kane, "Toward a language-theoretic foundation for planning and filtering," *International Journal of Robotics Research—in WAFR'16 Special Issue*, 2018.

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