Appendix to
Helpful Information Sharing for Partially Informed Planning Agents

Submission 978

1 Theoretical Analysis

In this section, we repeat the lemmas and theorems given in the paper and provide proofs of those.

Lemma 1. *Given an HIS problem $M = \langle \mathcal{P}_0, \mathbf{\Delta} \rangle$, and a solution π' to $T_{ka}(M)$, we let b_i and b'_i represent the actor's belief directly before applying action a_i in $\Pi_{exe}(\pi')$ in M and $T_{ka}(M)$, respectively. If the actor's belief tracking is sound and complete, then for any i, $b_i \subseteq b'_i$.*

Proof. We note that ramifications, assumptions and knowledge acquisition actions reduce the belief of the actor since they limit the number of possible world states (and reduce uncertainty to the agent's belief).

We will use a proof by induction. For the base case, the same initial information, including the information that is shared by the helper, is given in M and $T_{ka}(M)$, so $b_0 = b'_0$. As our induction hypothesis, we assume that $b_i \subseteq b'_i$ for some i. Since belief tracking is sound and complete, then after the application of $a_i \in \Pi_{exe}(\pi')$, all ramifications and sensors for which the preconditions are satisfied are fired in M. In contrast, the translation chooses which applicable ramifications and sensors to apply. Therefore, since we only allow the actor to make assumptions that are known to be true by the helper, the sensors and ramifications that are prescribed by π' are a subset of those that are fired in M, and therefore $b_{i+1} \subseteq b'_{i+1}$. □

Theorem 1. *Given an HIS problem $M = \langle \mathcal{P}_0, \mathbf{\Delta} \rangle$, under the assumption that belief tracking is sound and complete, there exists a solution to $T_{ka}(M)$ if and only if there is a solution to M.*

Proof. First, we show that if there is a solution to M, there exists a solution to $T_{ka}(M)$. A solution to M is the application of a set of information sharing interventions $\Delta \subseteq \mathbf{\Delta}$, followed by a PO-executable plan $\pi = \langle a_0, a_1, \dots, a_n \rangle$. We denote by b_0^{Δ} the belief after applying (the possibly empty) Δ to the initial belief b_0. According to Definition 5, π is guaranteed to be executable in b_0^{Δ}, since it is PO-executable.

Based on Lemma 1, we know that for every i, $b_i \subseteq b'_i$. If belief tracking is sound and complete, after the execution of each action in M, all the applicable sensing actions are fired and all ramifications that can be performed based on the invariant information in initial state I, are performed as well. Since $T_{ka}(M)$ includes an equivalent action that represents each of these events, we can simply translate every event, in order, into its compiled-world action equivalent to guarantee that the next action (a_{i+1}) is executable in $T_{ka}(M)$. Since we can do this for every action in the sequence, we know that there exists a solution to $T_{ka}(M)$.

Second direction: If there is a solution to $T_{ka}(M)$, there is a solution to M. Assume to the contrary that there exists some solution π' to $T_{ka}(M)$, but there is no solution to M.

This means there exists some action $a_i \in \Pi_{exe}(\pi')$ that is applicable in the belief b'_i in $T_{ka}(M)$ but not in b_i in M. This in turn means that there is some precondition L (or KL) of a_i that holds in b'_i in $T_{ka}(M)$ but not in b_i. This is, however, in contradiction to Lemma 1 which guarantees that $b_i \subseteq b'_i$. □

Corollary 1. *Given a HIS problem $M = \langle \mathcal{P}_0, \mathbf{\Delta} \rangle$, for any solution π' of $T_{ka}(M)$, $\Pi_{exe}(\pi')$ is a PO-executable plan of M.*

Proof. The proof for Corollary 1 is direct from the proof of Theorem 1. □

Theorem 2. *For any HIS problem $M = \langle \mathcal{P}_0 \mathbf{\Delta} \rangle$ if*

$$\mathcal{C}'_{exe} \cdot |\mathcal{A}'_{exe}| + \mathcal{C}'_{as} \cdot |\mathcal{A}'_{as}| + \mathcal{C}'_{ram} \cdot |\mathcal{A}'_{ram}| < \mathcal{C}'_{ka}$$

then for any optimal solution π'^* to $T_{ka}(M)$, $\Pi_{ka}(\pi'^*)$ represents a minimal set of information sharing interventions that guarantee the actor can achieve its goal.

Proof. Assume to the contrary that there is a plan π' such that $\Pi_{exe}(\pi')$ is PO-executable in M, but that $\Pi_{ka}(\pi') < \Pi_{ka}(\pi'^*)$ (the cost of information for π' is smaller than for π'^*). However, since a single information sharing action costs more than all other actions, this means that π' will have a lower total cost than π'^*, which contradicts the assumption that π'^* is an optimal solution. □

2 Lazy BFS

To increase efficiency over the exhaustive approach, we suggest *lazy breadth first search* (Lazy-BFS), which avoids the need to compute the value of nodes that are guaranteed not to represent optimal solutions. The structure of the search tree is similar to the one described above for BFS. The novelty here is that instead of fully evaluating each node, we use a lazy approach to node evaluation; when a node is expanded during the search, we map it to a *relaxed* intervention set, which is a intervention set that is guaranteed to overestimate the value of the node. We compute the exact value of the node only if the value for its relaxed set achieves the objective. Computational savings are achieved by storing the values of the relaxed sets and reusing them for future nodes that are mapped to the same relaxed set. To achieve a relaxation of the examined intervention set, we exploit the parameterized representation of our information sharing interventions and use *paramaterized padding* [Keren *et al.*, 2019], which associates each intervention set to a superset that includes all interventions that share the same value of one or more of its parameters.

Lazy-BFS is described in Algorithm 1. In addition to a queue that holds expanded nodes (Line 2), a table of relaxed sets is generated in Line 3. The search iteratively extracts a node Δ_{cur} from the queue and finds its corresponding padded set $\underline{\Delta}$ (Line 6). If the value of the padded set has not yet been computed, it is computed and cached in the table in Line 8. If the value of the padded set is $True$, then the actual value of the examined set Δ_{cur} is computed in Line 11. Otherwise, the value is set to $False$ in Line 13. If a solution is found, it is returned in Line 16. The successors of the current node which have not yet been expanded are inserted into the queue in lines 18-23. For each possible information sharing intervention, $\delta \in \mathbf{\Delta}$, $\Delta_{cur} \cup \{\delta\}$ represents the sensor extension set that results from adding δ to Δ_{cur}. The algorithm fails in Line 24 if all the available intervention sets were explored, but none of them guaranteed the is goal reachable for the actor.

Algorithm 1 Lazy Breadth First Search (Lazy-BFS)

```
1: Input: $M = \langle \mathcal{P}_0, D \rangle$
2: Create a queue $Q$ initialized to $\Delta_\emptyset$ (empty modificaiton set)
3: Create a table of padded set estimations $T$
4: while $Q$ is not empty: do
5:     $\Delta_{cur} \leftarrow Q$.dequeue()
6:     $\underline{\Delta} \leftarrow ExtractPaddedSet(\Delta_{cur})$
7:     if $\underline{\Delta} \notin T.keys$ then
8:         $T[\underline{\Delta}] = V(\underline{\Delta})$ (if the current padded set is not in the table, compute its value and store in table)
9:     end if
10:    if $T[\underline{\Delta}]$ is $True$ then
11:        $V_{cur} = V(\Delta_{cur})$ (only if $B^{PO}(\mathcal{P}_0^{T[\underline{\Delta}]})$ is true, compute the actual value of $\Delta_{cur}$)
12:    else
13:        $V_{cur} = False$
14:    end if
15:    if $V_{cur}$ is $True$ then
16:        return $V_{cur}$ (solution found)
17:    end if
18:    for all $\delta \in \mathcal{M}$ do
19:        if $\Delta_{cur} \cup \{\delta\}$ has not been expanded then
20:            enqueue $\Delta_{cur} \cdot \delta$ onto $Q$
21:        end if
22:    end for
23: end while
24: return Fail
```

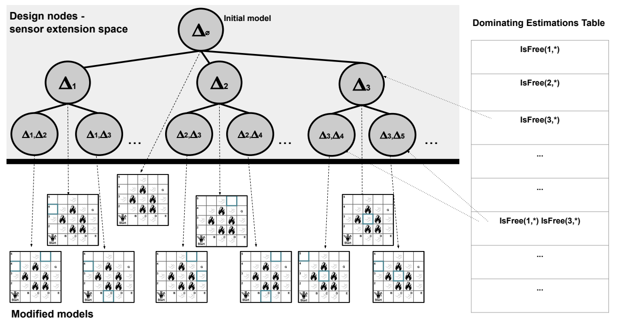

Figure 1: The search for an optimal information sharing solution using cached values of padded intervention sets.

Example 1 (continues=example)**.** *Lazy-BFS applied to Example 1 is depicted in Figure 1. The left part of the image represents the search in the space of information sharing interventions. On the top left, we have the nodes of the search tree, where each node represents an applicable set of interventions. Each node corresponds to a modified environment, at the bottom of the image, where the values of some cells are revealed to the actor. Padding is applied to estimate the value of a intervention set, and the computed value is stored in a table (on the right). In this example, each information sharing intervention $IsFree(x, y)$ reveals whether the cell (x, y) is free. Paramaterized Padding is applied by ignoring the second parameter (the row) of the intervention examined, and the padded set includes all interventions of the entire row.*

For example, the value of node $IsFree(1, 4)$ is overestimated by the padded set $IsFree(1, 1)$, $IsFree(1, 2)$, $IsFree(1, 3)$, $IsFree(1, 4)$ and $IsFree(1, 5)$. When expanding a node that is mapped to the same padded set, such as $IsFree(1, 5)$, the precomputed value is reused. Similarly, evaluating the value of the pair $\langle IsFree(1, 4), IsFree(4, 5)\rangle$ is done by calculating the value of applying $IsFree(1, 1)$, $IsFree(1, 2)$, $IsFree(1, 3)$, $IsFree(1, 4)$, $IsFree(1, 5)$, $IsFree(4, 1)$, $IsFree(4, 4)$ and $IsFree(4, 5)$. This value is reused for all intervention sets that consider one extension in column 1 and one in column 4.

Paramaterized padding was first suggested by Keren et al. 2019 to produce heuristic node estimations in a best first search for a design solution that minimizes the expected cost to goal. The heuristic values induce an order by which nodes are expanded but do not remove the need to compute the exact value of all nodes expanded in the search. Here we use padding to avoid redundant and expensive computations of the goal test.

Our lazy approach to node evaluation is useful in HIS domains where a large portion of the possible interventions have no effect on the actor's ability to achieve the goal and where it is costly to compute the effect and potential benefit of each intervention. In such settings, it may be potentially beneficial to reuse precomputed values in order to decide whether it is worthwhile to compute the actual value of a node. We devote the next section to characterizing domains for which

Lazy-BFS with paramterized padding is guaranteed to return an optimal solution.

Theoretical Analysis of Lazy-BFS

Lazy-BFS only computes the exact values of interventions sets for which a padded set achieves the objective. Accordingly, Lazy-BFS is guaranteed to produce optimal solutions in settings where padded sets of information sharing interventions are guaranteed to provide *over-estimations* of the original set, i.e., if the agent cannot reach its goal when the padded set is applied, then it cannot reach it if the original set is applied. This is a *monotonicity* property, and we will characterize a family of HIS models that are monotonic in this sense.

Definition 1 (Monotonic Model). *A HIS model M is* monotonic *if for every two information sharing intervention sets $\Delta, \Delta' \subseteq \mathbf{\Delta}$ s.t. $\Delta \subseteq \Delta'$, if $\Pi_{poe}(\mathcal{P}_0^{\Delta}) \neq \emptyset$ then $\Pi_{poe}(\mathcal{P}_0^{\Delta'}) \neq \emptyset$.*

Monotonicity does not necessarily hold in general for HIS. For example, a model is non-monotonic if one intervention may cancel the effect of another intervention (e.g., by contradicting it), or if changing an actor's knowledge may mislead an agent to fail in a new way because it becomes trapped in the environment. We focus our attention on models where all actions are *reversible*, i.e., for which it is always possible to return to a state that was previously visited. We specify conditions under which models that include only reversible actions are monotonic.

Theorem 3. *For any HIS model $M = \langle \mathcal{P}_0, \mathbf{\Delta} \rangle$, if all actions are reversible and the actor is using the K-planner with an optimal solver, then M is monotonic.*

Proof. Assume to the contrary that there exist some HIS model M that complies with the requirements, but that is not monotonic. This means, according to Definition 1, that there exist a pair of (possible empty) intervention sets $\Delta, \Delta' \subseteq \mathbf{\Delta}$ such that the goal is reachable in $\mathcal{P}_0^{\Delta}$ but not in $\mathcal{P}_0^{\Delta \cup \Delta'}$. Since the goal is reachable in $\mathcal{P}_0^{\Delta}$ we know there is some path $\pi = a_0, a_1, \ldots, a_n$ that is PO-executable in $\mathcal{P}_0^{\Delta}$ and for which the corresponding history $h = b_0, a_0, b_1, a_1, \ldots, b_n, a_n, b_{n+1}$ in $\mathcal{P}_0^{\Delta}$ reaches a goal belief but there is no such history (and corresponding path) in $\mathcal{P}_0^{\Delta \cup \Delta'}$. We consider two cases: one where the actor follows π in $\mathcal{P}_0^{\Delta \cup \Delta'}$ and one where it follows a different path.

For the first case, we let $h' = b'_0, a_0, b'_1, a_1, \ldots, b'_n, a_n, b'_{n+1}$ represent the history for following π in $\mathcal{P}_0^{\Delta \cup \Delta'}$. According to Definition 6, information sharing interventions only change the information available to the actor, leaving the action sets and sensor models unchanged. Accordingly, the initial belief in the modified model subsumes the belief in the original model, i.e., $b_0 \subseteq b'_0$ and since a_0 is executable in b_0, it is executable in b'_0. Moreover, since the sensor model is unchanged, any observation that can be made in b_0 can be made in b'_0. This guarantees that for any i $0 \leq i \leq n$, $b_i \subseteq b'_i$ and therefore π is PO-executable in $\mathcal{P}_0^{\Delta \cup \Delta'}$ and if G holds in b_{n+1}, it holds in b'_{n+1}, thus contradicting our assumption.

In the second case, the actor follows some arbitrary path π' in $\mathcal{P}_0^{\Delta \cup \Delta'}$. If π' achieves the goal, we reach a contradiction. If the path does not achieve the goal and since we assume that all actions are reversible, we know that the actor can always return to a previously encountered state. Bonet and Geffner 2011 show that any fact that is known in some belief state is known in any belief state that is reachable from it. They also show that the T_k (and $K_{prudent}(\mathcal{P})$) transformation is sound and complete for simple problems with a connected space, which are the only problems we consider here. This means that an actor that uses an optimal solver can always return to the initial state from any state it reaches. If it does, its belief b'_0 subsumes the initial belief b_0 and since π is PO-executable in b_0, it is PO-executable in b'_i and reaches a goal belief. This refutes our assumption and concludes our proof. □

Corollary 2. *For any HIS model $M = \langle \mathcal{P}_0, \mathbf{\Delta} \rangle$, if all actions are reversible and the actor is using the K-planner with an optimal solver, then Lazy-BFS is guaranteed to return an optimal solution.*

Proof. Lazy-BFS iteratively explores sensor extension sequences of increasing size, thus guaranteeing that an optimal solution will be encountered in the search before any solution with greater length. Since all permutations of a sequence will yield the same result, the algorithm only needs to consider distinct sets of sensor extensions. Further, according to Theorem 3we have monotonicity, which guarantees that if an optimal solution is encountered its value will be computed (in Line 11), and will be returned by Lazy-BFS. □

In a nutshell, the mononicity property relies on the actor's ability to backtrack from any state, which guarantees that information sharing interventions only add paths to its set of PO-executable paths.

To characterize models in which padding can be automatically generated, we focus on *lifted* information sharing interventions for which each grounded instantiation specifies a single intervention. Each lifted intervention $\delta(p_1, \ldots, p_n)$ is characterized by a set of parameters, $p_1, \ldots, p_n$, and a set of valid values, $dom(p_i)$, for each parameter p_i. A grounded intervention $\delta(v_1, \ldots, v_n)$ is a valid assignment $v_i \in dom(p_i)$ to all parameters.

For lifted extensions, intervention padding can be implemented using *parameterized padding*, by mapping a grounded intervention to a set of interventions with the same values on a set of lifted parameters. This is illustrated in the following example.

Example 2 (continues=example). *In our example, a intervention $IsFree(x, y)$ is a lifted representation, where x and y denote coordinates of a cell. The value of the grounded intervention $IsFree(1, 1)$ can be (over)estimated by considering the value of applying the set $IsFree(1, 1)$, $IsFree(1, 2)$, $IsFree(1, 3)$,* etc. *This value is cached, so when intervention $IsFree(1, 2)$ is examined, it is mapped to the same padded set, and the pre-computed value can be reused.*

In our empirical evaluation, we use PDDL [McDermott *et al.*, 1998] to represent lifted sensor extensions and implement parameterized padding.

3 Dataset

Our domains are adapted from the partially observable planning problems of [Bonet and Geffner, 2011] and [Albore *et al.*, 2009]. The adaptation to HIS involves limiting the actor's sensor model and specifying the information that can be revealed by the observer [1]. We also introduce a new domain called ESCAPE-ROOM.

- **WUMPUS**— the actor aims to achieve a goal location (a cell with gold) without falling into a pit or encountering a deadly wumpus, which both can be sensed from an adjacent location. The observer can reveal the location of a wumpus or a pit or whether a single cell is *safe*, i.e., without a wumpus or pit. The navigating robot domain described in Example 1 is similar to the WUMPUS domain, with dead-end cells represented by occupied cells that can be sensed from adjacent locations.
- **TRAIL**— the actor must follow a sequence of nodes with 'stones' in order to reach an unknown destination node. The actor cannot sense the stones. The observer can reveal whether a specific node has a stone, and whether a specific node is the goal.
- **COLOR-BALLS**— the actor navigates a grid to deliver balls of different colors to destination cells with the same color. A goal represents a subset of balls that need to be delivered. The actor can pick up the balls, but cannot sense them. The observer can reveal the location of a specific ball.
- **COLOR-BALLS-E**— same as COLOR-BALLS, but the actor cannot see the color of balls. The goal here is to deliver one of the balls of a certain color to each colored destination. The observer can reveal the location and color of a specific ball.
- **UNIX**— the actor needs to transfer files to their destination folders without knowing their location. The observer can reveal locations of specific files.
- **LOGISTICS**— the actor needs to transport packages to their destinations. Packages can be transported within cities via trucks, and between cities via airplanes. The location of each package is initially known to be one of a set of possible locations (e.g. package A is either in location 1, 2 or 3), but cannot be sensed before they are loaded. The observer can reveal locations of packages.
- **ESCAPE-ROOM**– the actor needs to exit a room through one of the doors of the room. The doors are initially locked, and can be unlocked only when a specific configuration of switches is set. The observer can reveal the type of a door (which configuration is needed for unlocking it) and the status of a given switch. In it, the actor needs to exit a room through one of the doors. The doors are initially locked and can be unlocked only when a specific configuration of switches is set. The observer can reveal the type of door (i.e. which configuration is needed for unlocking it) and the status of a given switch.

[1]The supplementary material includes the complete dataset and code base.

For each domain, we generated at least 100 benchmarks by randomly selecting the configuration of the initial setting, and the set of items that are known to the observer (with full knowledge as a special case).

References

[Albore *et al.*, 2009] Alexandre Albore, Héctor Palacios, and Héctor Geffner. A translation-based approach to contingent planning. In *IJCAI*, 2009.

[Bonet and Geffner, 2011] Blai Bonet and Hector Geffner. Planning under partial observability by classical replanning: Theory and experiments. In *IJCAI*, 2011.

[Keren *et al.*, 2019] Sarah Keren, Luis Pineda, Avigdor Gal, Erez Karpas, and Shlomo Zilberstein. Efficient heuristic search for optimal environment redesign. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2019.

[McDermott *et al.*, 1998] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl-the planning domain definition language. Technical report, Yale Center for Computational Vision and Control, 1998.