ECAI 2014
T. Schaub et al. (Eds.)

© 2014 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-419-0-597

Privacy Preserving Landmark Detection

Shlomi Maliah[1] and **Guy Shani** and **Roni Stern**

Abstract. In many cases several entities, such as commercial companies, need to work together towards the achievement of joint goals, while hiding certain private information. Multi-agent STRIPS (MA-STRIPS) is a new and attractive model for describing collaborative multi-agent privacy preserving planning, which is appropriate for such problems. In single agent classical planning, landmarks are key to constructing strong heuristics for state space search. In this paper we propose a method for identifying landmarks in MA-STRIPS in a privacy preserving distributed setting. The agents collaborate to find sound landmarks without revealing their private actions or goals. In addition, we also propose a novel MA-STRIPS planner that uses these landmarks. We empirically show that our detected landmarks improve the performance of previous approaches, and that our new planner is faster than all existing planners for multi-agent problems.

1 Introduction

Many modern organizations outsource some of their tasks to outside companies. The organization must then work together with the outsourcing companies to achieve its goals, while disclosing as little as possible about its abilities. Consider for example the case of a military organization that has outsourced its food service to an outside company. The food service company must deliver food into logistics centers, from where it is picked by army trucks and distributed to the army bases. The military would not want, however, to disclose the whereabouts of these bases, the number of people in each base, or the number and location of army trucks. The two organizations must then collaborate while maintaining privacy about their actions, and communicating only over public actions, such as the interactions at the logistics centers.

More generally, consider the setting where a team of agents collaborate to achieve a set of goals while constrained to preserve each others privacy. The planning task in this setting is to generate a plan that achieves goals without breaking the privacy constraint. We call this planning task *collaborative privacy preserving planning* (CPPP).

A somewhat similar notion to CPPP was previously discussed in the distributed constraint optimization setting (DCOP), where an "agent" is responsible for the value of a single variable, and the overall task is to find an assignment of values to variables so as to maximize a joint reward. DCOP can be viewed as an instance of CPPP, where the agents' actions are assignments of values to variables. DCOP would then be a special case of CPPP where agents may perform a single (assign a value to a variable), while in the general CPPP each agent can perform a sequence of actions.

CPPP was discussed in the context of the multi-agent STRIPS (MA-STRIPS) model [1]. MA-STRIPS is an extension of STRIPS (or PDDL), where instead of a single agent performing one action at a time, there are multiple agents acting concurrently, each having a different set of capabilities. This model allows partitioning an agent's actions to private actions and public actions. Partitioning was initially intended to improve planning efficiency, but work on MA-STRIPS also identifies privacy as a desirable attribute on its own.

The best performing privacy preserving MA-STRIPS planner is MAFS [9, 7]. MAFS was shown to be privacy preserving in the sense that no private information is revealed to the other agents. This is achieved by a message passing algorithm in which each agent searches for a plan concurrently, publishing reachable states to other agents when public actions are performed. Privacy is preserved by hiding the private information in the published states.

MAFS relies on an efficient heuristic to guide the search. One of the best classical planning heuristics, which have also been used for MAFS, is based on identifying *landmarks*. A landmark is a literals/action that must be achieved/performed before achieving the goal. Discovering all landmarks is computationally hard [5], but there are polynomial methods that are able to identify some of the landmarks [5, 10]. In this paper we discuss the identification of landmarks without breaking the privacy constraints.

One way to do so is for each agent to apply existing landmark detection algorithms on a projection of the planning problem that ignores all private actions of the other agents. Such a projection-based approach has various problems. Landmarks for goals that are achieved by private actions of one agent would be unknown to all other agents. As private preconditions of an action are not known to all agents, then some agents would not be able to detect landmarks that are required to achieve these private preconditions. We propose here a distributed, privacy preserving algorithm for detecting landmarks, called PP-LM, that overcomes these shortcomings.

PP-LM can be used to improve MAFS, identifying more landmarks and as a results a better search heuristic. In addition to PP-LM, we also introduce a novel, highly efficient, MA-STRIPS planner that uses the landmarks found by PP-LM. This algorithm, which we call the Greedy Privacy Preserving Planner (GPPP), uses synchronized search for finding a sequence of public actions that would achieve the goal. GPPP uses the landmarks found by PP-LM to guide this search. Once a plan of public actions was jointly found, agents plan independently in their private spaces to achieve the preconditions of the agreed upon public actions.

We empirically demonstrate that PP-LM finds more landmarks than the projection-based approach, especially in domains where agents must collaborate in order to achieve goals. The performance of MAFS was also improved when using PP-LM. The overall performance was then greatly improved by using GPPP. With GPPP, problems were solved substantially faster than with MAFS, often exhibiting a speedup of over an order of magnitude.

[1] ISE Department, Ben-Gurion University, E-mail:{shlomima@post.bgu.ac.il,shanigu@bgu.ac.il,roni.stern@gmail.com

2 Preserving Privacy in MA-STRIPS

As a preliminary, we define MA-STRIPS and the privacy preserving property we wish to achieve.

Definition 1 (MA-STRIPS [1])**.** An MA-STRIPS problem is represented by a tuple $\langle P, \{A_i\}_{i=1}^{k}, I, G\rangle$ where:

- P is the set of possible literals (facts about the world)
- $I \subseteq P$ is the set of literals true at the start state
- k is the number of agents
- A_i is the set of actions that agent i can perform.
- $G \subseteq P$ is the set of literals that needs to be true after executing the generated plan (i.e., the goal state)

Each action in A_i has the standard STRIPS syntax and semantic, that is $a = \langle$ pre(a), add(a), del(a) $\rangle$. The sets A_i are disjoint, that is, no public action can be executed by two different agents.

To efficiently solve an MA-STRIPS planning problem, Domshlak and Brafman partition the actions of each agent to *private* and *public* actions [1]. An action is regarded as a *private action* of an agent if none of its preconditions or effects are preconditions or effects of an action of another agent. The literals in the preconditions and effects of a private action are regarded as *private literals*.[2]

This partitioning to private and public actions and literals serves two purposes. First, it suggests a decomposition of the planning task; Public actions are decided together while private actions can be planned individually, reducing complexity [1]. Second, it allows preserving some form of privacy while planning a joint activity.

Privacy is commonly defined with respect to the capabilities of an adversary agent that attempts to reveal some private information. We assume that the adversary agent knows about the existence of the private actions and literals. The adversary agent does not:

- Know about the private literals and actions of other agents.
- Know which private literals are true in the initial state.
- Know the private literals and effects (i.e., which private literals are literals and/or effects of which actions).
- Observe the execution of private actions during plan execution.

The information an adversary may seek is the private actions in the generated plan and which private literals hold or do not hold during plan execution. Following Nissim and Brafman [8], we say that a planning algorithm is *privacy preserving* if none of the private information, i.e., the private states and actions, are passed between the agents during planning. A deeper privacy analysis, including theoretical results about the information an adversary can infer indirectly, are beyond the scope of this work.

2.1 Privacy Preserving Solvers

The best performing MA-STRIPS solver is the MAFS algorithm [9]. MAFS is a distributed, privacy preserving algorithm, where each agent searches using its private and public actions. MAD-A* and Selfish-MAD-A* [8] are variants of MAFS for finding optimal solutions and solutions for self-interested agents, respectively.

In MAFS, each agent runs a best-first search towards the goal, guided by a heuristic function. The search performed by each agent is conducted in a state space restricted to the public literals and the private literals of that agent. We call this state space the *individual state space* of the agent. During the search, when the currently expanded state is generated using a public action, it is published (broadcast) to all other agents, masking the private literals in that state. The published state is inserted into the other agents' search lists, allowing it to be expanded by them with their own private actions. Thus, these other agents learn about the new public literals that were achieved and can use them to continue advancing toward the goals. When an agent achieves the last goal, it broadcasts that all goals have been achieved. Then, each agent reconstructs its private plan and a complete plan is found.

Key to the success of MAFS is the heuristic used by the agents when searching their individual state space. As mentioned earlier, some of the most successful heuristics in planning are based on identifying landmarks [6, 10, 5]. Next, we provide required background about landmark heuristics, and discuss their applicability to CPPP.

2.2 Landmarks in Classical Planning

A landmark Φ for a classical planning task $\Pi = \langle P, A, I, G\rangle$ is a logical formula over the facts P, which must be satisfied at some state along every solution of Π [5]. As in most literature dealing with landmarks, we will restrict our attention to landmarks that are facts or disjunctions over facts. Each planning task has some trivial landmark consisting of all the goal literals.

Although it is PSPACE-hard even to check whether a given literal is a landmark or not, there are several efficient algorithms that return a set of landmarks and orderings [5, 10]. Recent landmark detection algorithm work as follows. First, every literal that is true in the goal and not in the initial state is identified as a landmark. Then, for every landmark p, if all actions that achieve p (denoted as "achievers" of p) have some literal $q \in P$ as a precondition, then q is also a *fact landmark*. If there are no common preconditions, a set of literals that is a hitting set over the preconditions of all achievers of p can be used as a *disjunctive landmark*, denoting that at least one of the literals in the disjunctive landmark must be achieved in every solution. This landmark detection process continues until all landmarks are resolved.

Originally, landmarks were used as subgoals [5], guiding a base planner inside a control loop. More recently, the number of landmarks which are yet to be achieved plus the number of landmarks that should be achieved again, was used as a heuristic function for very successful state space planners [10]. Note that this is an inadmissible heuristic estimate, because an action might achieve more than one landmark. There are also admissible heuristics that are based on landmarks [6], which are useful for optimal planning.

To estimate which landmarks should be achieved again, landmark detection algorithms also identify various types of *ordering* between landmarks, and there are several ways to use these orderings. In our experiments, we considered two types of orderings: *greedy-necessary* and *reasonable*. Briefly, a landmark L_1 is ordered before L_2 in a greedy-necessary order if L_1 needs to be true to achieve L_2. A landmark L_1 is ordered after L_2 in a reasonable order if L_1 is needed at the same time or after L_2 is achieved, and achieving L_2 deletes L_1. The heuristic function we used in our experiments counts a landmark L_1 as a landmark that should be achieved again if there is another landmark (or goal predicate) L_2 that has not been achieved yet and either: (1) L_1 is also are not true in the current state and there exists a *greedy-necessary* ordering $L_1 \rightarrow L_2$ (as L_1 is needed to achieve L_2), or (2) there exists a *reasonable* ordering $L_2 \rightarrow L_1$ (since when L_2 is achieved it will delete L_1, which is needed afterwards).

[2] This partition to private and public actions that we considered is based on the domain description. Accepting a partition as input is also possible.

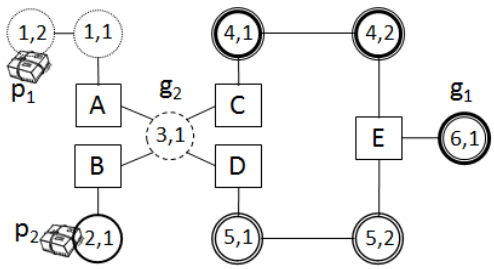

Figure 1. A logistics example, with 6 agents. Squares mark public cities, while circles mark private cities. Private cities are labeled i, j where i is the agent, and j is the city index for that agent. Different circle types represent different agents. All actions move packages between nearby cities. There are two packages p_1, p_2 that must be delivered to g_1, g_2 respectively.

The exact definitions of the different types of orders can be found in previous work [10], and the algorithms presented in this paper are robust to other landmark-based heuristic functions. We reported above the details of the heuristic function used in our experiment for completeness and to allow reproducibility of our results

2.3 Projection-Based Landmark Heuristic

In MAFS, each agents searches in its individual state space. Thus, the heuristics used cannot directly consider private actions and literals of other agents. The individual state space is, in fact, a projection of the full multi-agent problem onto the agent's private and public literals and actions. Therefore, it can be viewed as a single agent planning problem, allowing the agent to use landmark heuristics, and in general any existing heuristic from the planning literature.

An agent may identify some landmarks by running an existing landmark detection algorithm on its individual state space. There are, however, a few substantial problems with detecting and using landmarks in this projection of the full, multi-agent state space.

Projection-based landmark detection algorithm are limited, as every agent only identifies landmarks related to goals it can achieve. A special case of this problem is when a goal is only achievable by a private action of some agent a_1. As a result, the other agents are not aware of any action that achieves the goal, and therefore cannot find any landmark for that goal. Moreover, if this is the only goal, then all other agents would simply search blindly towards achieving it.

As an example, consider the logistic setting in Figure 1 and assume that the only goal is to move package p_1 to location g_1. The only action that can put a package in g_1 is a private action of agent a_6. Thus, all the agents would not be able to infer any landmark and would search blindly.

Projection-based landmarks can also produce low quality landmarks, i.e., disjunctive landmarks. For example, assume that a goal can be achieved by agent a_i by several actions, each having also some set of preconditions. Now, assume that only one of these set of preconditions are achievable from the start state. If this information depends on other agents and a_i is not aware of it, then it would infer a disjunctive landmark, while a more effective landmark would only consider the preconditions of the achievable action. For example, assume that the landmark being developed is the goal at(p2; [3; 1]) from Figure 1. A projection-based algorithm would detect $\{at(p2; A) \vee at(p2; B) \vee at(p2; C) \vee at(p2; D)\}$, while in fact $at(p2; B)$ could have been inferred. The next section provides a possible remedy to these shortcomings, finding more and better (more refined) landmarks while still preserving privacy.

3 Privacy Preserving Landmark Detection

In this section we present a landmark detection algorithm called PP-LM (Privacy Preserving LandMark detection algorithm) that improves upon the projection-based approach described above by allowing agents to communicate the public landmarks that they discover. This collaborative distributed landmark identification algorithm is intended to be performed jointly by the agents prior to a distributed search (e.g., before running MAFS). A prominent benefit of PP-LM over the projection-based approach is that PP-LM enables an agent to find a landmark that is not achievable by it, as well as finding more refined landmarks.

The algorithm progresses by agreeing on a landmark to explore. All agents collaborate on the development of that landmark, until it is resolved as satisfied by the initial state, or by another set of landmarks. This process proceeds until all landmarks are eventually satisfied by the initial state. We now explain this process in detail.

3.1 Initialization

We say that a landmark is a *public landmark* if it only contains public literals, while we say that a landmark is a *private landmark* if it contains at least one private literal. For example, $at(p_1, [1, 2])$ is a private landmark, while $at(p_1, E)$ is a public landmark. Each agent maintains a list of private and public landmarks, initialized by the set of public and private goals it can achieve.

3.2 Identifying a Landmark to Develop

First, the agents agree on a single landmark to develop. We take a two step approach. First, prior to each selection the agents must agree on one leading agent (say, by turn). Then, that agent picks a landmark to develop, either public or private from its relevant landmark list. If a public landmark is chosen, then the leading agent informs everybody about it. When a private landmark is chosen, the leading agent only informs other agents that it is working on some private landmark. We do not enforce any particular ordering on the landmarks to be developed. This can be done by any tie breaking method.

3.3 Identifying Achievable Literals

Once all agents agree on developing landmark l, we begin the process of identifying which literals are achievable by the entire team of agents before the landmark is achieved. To do so, each agents maintains a set of public literals it knows are achievable by the team as well as the set of private literal it can achieve (including those it can achieve with the help of other team members actions). We call this set the agent's set of *achievable literals*. Initially this set contains only the literals in the initial state (the public ones and the private ones known to that agent). The agent then apply all private and public actions it can perform given its set of achievable literals, ignoring the delete effects of these actions. Whenever a public literal is achieved, the agent publishes it to all other agents. The other agents add this literal to their set of achievable literals, potentially allowing more actions to be performed, and more literals to be achieved. This process repeats until no more public literals can be achieved by any agent. To avoid circular reasoning, such that the achievement of l requires l to be achieved, all agents ignore all actions that achieve l (or any literal in l in the case of disjunctive landmarks).

When this phase is concluded, all agents have the same set of public literals in their set of achievable literals. These are the public

literals that can be achieved without achieving l first (given delete-relaxation). Note that these sets of achievable literals are an approximation of the real set of literals that the team can achieve, as we use a "delete relaxation" approach to construct these sets. Thus, not all literals in this set are indeed achievable. However, all the literals not in this set are guaranteed to be unachievable without achieving first the landmark l. These are used later to detect additional landmarks.

As an example consider the logistics example in Figure 1. Assume that the landmark being developed is the goal $at(p_2, [3, 1])$. Now, agent 3 must ignore all actions that move p_2 to $[3, 1]$. Restricting our attention to literals involving p_2, the set of agent 3's achievable literals contains only $at(p_2, B)$. Note that if we did not ignore actions that achieve $at(p_2, [3, 1])$, then agent 3's set of achievable literals we have p_2 at any public city. Note that this set of achievable public literals is recreated for every landmark we develop, that is, we do not reuse achievable sets that were computed for other landmarks.

3.4 Identifying Satisfying Agents

The next phase identifies which agents can satisfy the current landmark l. These are the agents that have an action that achieves l, and the preconditions of this actions are in their set of achievable literals. Every agent that can satisfy l declares so to the entire team.

If only one agent can satisfy l (e.g., when l is a private landmark) then that agent now identifies new landmarks, public and/or private, by considering the literals in the preconditions of the applicable actions that achieve l that are not true in the initial state or an existing landmark. [3] The agent then insert these new landmarks to its list of landmarks. For example, if $at(p_2, [3, 1])$ is the landmark being developed, and the only public achievable literal is $at(p_2, B)$, then only agent 3 can satisfy this landmark, using the action $move(p_2, B, [3, 1])$. Then, a new public landmark $at(p_2, B)$ is identified and added to agent 3s landmark list.

When more than one agent can satisfy l, each such agent identifies landmarks required for achieving l using the same collaborative process. Then, each of these agents broadcasts the set of public landmarks it requires to achieve l (the private landmarks are not published to preserve privacy). All of these set of public landmarks are joined together to form a public disjunctive landmark that is added to the list of landmarks of all the relevant agents.

For example, we may developing $at(p_1, E)$, the public achievable literal set will contain $at(p_1, C)$ and $at(p_1, D)$. Two agents, 4 and 5, would now be able to satisfy the landmark. Developing their landmarks, we would eventually learn that $at(p_1, C) \vee at(p_1, D)$ is a public disjunctive landmark. Once a landmark has been satisfied as explained above (i.e., either by generating more landmarks or by the initial state), the agents decide on the next landmark. The process then repeats until no more landmarks are identified.

Table 1 demonstrates the differences between the landmarks detected by the projection approach and our PP-LM method. For example, the projection-based landmark detection algorithm would not identify $at(p_1, [1, 1])$ as a landmark, because it assumes that actions moving p_1 to city E, being handled by other agents, have no preconditions. The projection-based approach, which doesn't run the reachability analysis for landmark detection that we propose (i.e., collaboratively generating the set of public literals achievable by the team), would identify the landmark $at(p_2, A) \vee at(p_2, B) \vee at(p_2, A) \vee at(p_2, B)$, while we identify the more refined landmark $at(p_2, B)$.

[3] Literals existing in all preconditions are considered as *fact landmarks*. If no such literals exist, then a *disjunctive landmark* is formed containing the set of literals that occur in all of these actions preconditions.

Method	Private landmarks	Public landmarks
Projection landmarks	$\{at(p_2, [3, 1])\}$ $\{at(p_1, [6, 1])\}$	$\{at(p_2, A) \vee at(p_2, B) \vee$ $at(p_2, C) \vee at(p_2, D)\}$ $\{at(p_1, E)\}$
PP-LM	$\{at(p_2, [3, 1])\}$ $\{at(p_1, [6, 1])\}$ $\{at(p_1, [3, 1])\}$ $\{at(p_1, [1, 1])\}$	$\{at(p_2, B)\}$ $\{at(p_1, A)\}$ $\{at(p_1, C) \vee at(p_1, D)\}$ $\{at(p_1, E)\}$

Table 1. Differences in landmark detection between the projection method of MAFS and PP-LM, over the example in Figure 1

4 Greedy Privacy Preserving Planner

Next, we propose a new algorithm for CPPP called the Greedy Privacy Preserving Planner (GPPP), which is based on the landmarks identified by PP-LM. In GPPP, the agents search jointly in the space of public actions, while each agent searches in the space of its private actions. We use a synchronized approach: one of the agents is designated as the *leader*, performing the search in the public action space and contacting the other agents to receive information it needs. Importantly, even the leader does not know about the private actions and literals of other agents.

First we define some supporting terms. A *private state* of an agent is a set of that agent's private literals. We assume that a private state of an agent has a *private state identifier* (PSI), which is an index to a specific private state. A key attribute of a PSI is that the only agent that can map a PSI to a private state is the corresponding agent of that private state. Thus, sharing a PSI with other agents does not break the privacy constraints.A *public joint state* (PJS) is a set of PSIs, one per agent, and a set of public literals. Formally, we define a PJS as a tuple $\langle \{PSI_i\}_1^k, pub \rangle$, where PSI_i is a PSI for a private state of agent i, and pub is a set of public literals.

In GPPP, the leader searches in the space of PJSs. Initially, each agent generates a single private state according to the initial state. Then, each agent runs a "private action delete relaxation" process, which means applying that agent's private actions and ignoring their delete effects. This continues until no more private literals are added. Each agent then sends the PSI of the resulting private state to the leader along with a landmark-based heuristic computed using the landmarks known to that agent that it can satisfy. The leader then constructs a PJS from the PSIs of the agents and the public literals in the initial state. Each PJS has a heuristic value, computed as the sum of the heuristics of its constituent private states.

An agent a_i applying an action A generates another PJS. In the generated PJS, the public literals are updated according to *eff(A)*. A new private state for a_i is generated by first updating the current private state with the private effects of A. Then, a_i performs again the "private action delete relaxation" process to update its private state with all achievable private literals, except those that are mutexes of preconditions or effects of A. The resulting PJS contains the PSI of this updated private state, and is returned to the leader, along with its heuristic value. In addition, a_i notifies the leader if in a generated PJS all the goals known to it have been achieved. If all agents report that in a given PJS all their known goals were achieved, then that PJS is identified as a potential goal state.

When a potential goal state has been identified, its corresponding sequence of public actions are extracted. Each agent then needs to verify that it can perform the public actions planned for it. This is done by planning to achieve the private preconditions of each public action assigned to it, starting from the first public action and pro-

ceeding sequentially. Note that an agent might discover that it cannot achieve the public action assigned to it. This may occur, since when the public actions were planned, each agent considered the list of achievable private literals by ignoring the delete effects of the private actions (the "private action delete relaxation" process). If an agent may not be able to achieve one of the public actions planned for it, it broadcasts that this action cannot be executed. If the leader receives this message, it prunes this PJS and the search for a sequence of public actions leading to the goal continues.

Finally, if a potential goal state is found to be a goal, i.e., all agents can locally satisfy the preconditions of the planned public actions and achieve all goals, then a solution has been found. Then, each agent maintains both the sequence of public actions in the plan as well as its private actions (used to satisfy public action preconditions and achieve its private goals). To ensure completeness the leader performs a best first iterative deepening search on the state space of PJSs, i.e., a best first search is performed while pruning all PJSs with heuristic value higher than a give threshold. If all PJSs are pruned, then the threshold is increased and the process restarts. In our experiments we set the initial bound to the number of found landmarks.

5 Experimental Results

We perform a set of experiments to evaluate the proposed privacy preserving landmark detection algorithm, and the proposed Greedy Privacy Preserving Planner (GPPP). The experiments were performed on MA-STRIPS translations of a subset of the domains from the international planning competition: logistics, elevators, zenotravel, rover, and satellite.

In addition, we introduce a new MA-STRIPS domain we call "MA-Blocks". Like the classical blocksworld domain, the task in MA-Blocks is to construct a specific blocks formation. In MA-Blocks, there multiple "arms" that can pick blocks and stack them, each corresponding to an agent. In addition, blocks can only be stacked at a finite set of locations, and each "arm" can reach only a subset of these locations. The reach of the different arms overlap, allowing agents to pass blocks to other agents.

Domain	Proj-LM	PP-LM	Full-LM
ProbLogistics	17.8	38.2	41.6
Elevators	13.9	16.8	22.7
MA-Blocksworld	26.8	72.0	76.8
Zenotravel	7.5	7.8	16.2
Rover	13.6	13.6	24.4
Satellite	15.6	15.8	28.1

Table 2. # landmarks detected with Proj-LM and PP-LM

We compare three landmark detection algorithms: (1) the projection-based landmark detection algorithm (used by Nissim et al.), denoted *Proj-LM*, (2) a landmark detection algorithm that ignores privacy constraints, denoted by *Full-LM*, and (3) the proposed privacy preserving landmark detection algorithm (PP-LM). Results for Full-LM provide an upper bound on the number of landmarks PP-LM may detect (as Full-LM ignores the privacy constraints).

The average number of landmarks found in each domain with each of the landmark detection algorithms is given in Table 2. In probLogistics and MA blocksworld domain, the number of landmarks found with PP-LM is substantially larger than Proj-LM (2.7 times more in the MA blocks domain). In fact, in these domains PP-LM was able to find almost all the landmarks found by Full-LM. In the elevators domain, the number of additional landmarks found by PP-LM was more modest. Furthermore, in some cases PP-LM produced more refined disjunctive landmarks, i.e., disjunctive landmarks that where subsets of disjunctive landmarks found by Proj-LM.

In the rover, zenotravel, and satellite domains, PP-LM did not find any additional landmarks over Proj-LM. A deeper look into these domains explains these results. In MA-Blocksworld, probLogistics, and elevators, agents need to collaborate to achieve goals. For example, in probLogistics the trucks and the planes need to collaborate to move package from one city to another. By contrast, in satellite, zenotravel, and rovers, every goal can be achieved by a single agent. For example, in the rover domain taking an image of a rock can be done by any single rover. As a result, the agents need to coordinate which agent will achieve which goal but not to collaborate. This eliminates the advantage of PP-LM over Proj-LM, as no public landmarks would be generated from private landmarks. Thus, using PP-LM in domains that require only coordination between agents is redundant.

Domain	Proj-LM	PP-LM	Full-LM
ProbLogistics	0.02	0.12	0.03
Elevators	0.02	0.19	0.04
MA-Blocksworld	1.69	4.56	4.53
Zenotravel	0.07	2.27	0.14
Rover	0.03	0.03	0.03
Satellite	0.16	7.61	0.00

Table 3. Runtime (sec.) of the landmark detection algorithms

As the results above show, PP-LM is able to find more landmarks than Proj-LM while still preserving privacy. However, PP-LM requires more runtime and communications between the agents. Table 3 shows the average runtimes of running the different landmark detection algorithms. As expected, PP-LM is slower than Proj-LM. However, as demonstrated below, this runtime is often well spent. The overall runtime of PP-LM is small, and the planning time can be substantially reduced by finding even a few additional landmarks. Note that PP-LM is also slower than Full-LM even though Full-LM finds more landmarks. This is because Full-LM does not require a message passing mechanism to preserve privacy, as every agent had access to the complete problem definition.

	Proj-LM	PP-LM		Full-LM		
Domains	MAFS	MAFS	GPPP	MAFS	GPPP	FF
ProbLogistics (12)	**12**	**12**	**12**	**12**	**12**	**12**
Elevators (15)	**15**	**15**	**15**	**15**	**15**	**15**
MA-Blocks (15)	13	14	**15**	**15**	**15**	13
Zenotravel (13)	**13**	**13**	**13**	**13**	**13**	**13**
Rover (13)	**13**	**13**	**13**	**13**	**13**	**13**
Satellite (15)	11	11	**15**	12	**15**	14

Table 4. # instances solved under 30 minutes

Next, we compare the performance of GPPP and MAFS using the three landmark detection algorithms. Table 4 shows the number of instances solved by every algorithm under a 30 minutes timeout. For every domain, the total number of instances is given in brackets near the domain's name. We marked in bold the algorithm that solved the most instances in each domain.

The baseline in this comparison is MAFS with Proj-LM, described by Nissim et al. According to their experiments, MAFS outperforms previous MA-STRIPS solvers, and thus MAFS with Proj-LM is the best privacy preserving MA-STRIPS planner published so far. As can be seen, using PP-LM with MAFS already improves on this baseline, solving an additional instance in the MA-Blocks domain. When combined with GPPP, all instances in all domains were solved. Even if

privacy is ignored when searching for landmarks, GPPP still outperforms MAFS, solving 3 more instances in the satellite domain. Furthermore, GPPP with PP-LM is even able to solve more instances than FF [4], a planner that is neither privacy preserving nor distributed. This suggests that in domains with a natural decomposition to multiple agents, GPPP is a more suitable choice. However, our focus is not in developing fast, single agent planner, but in developing an efficient distributed and privacy preserving solver. FF is neither, and thus only serves as a baseline for comparison.[4]

	Proj-LM	PP-LM		Full-LM	
Domain	MAFS	MAFS	GPPP	MAFS	GPPP
ProbLogistics	5.74	4.61	**0.25**	4.49	0.11
Elevators	8.34	3.87	**0.30**	11.80	0.08
MA-Blocks	67.29	29.01	**2.72**	9.75	1.39
Zenotravel	44.68	46.71	**3.36**	5.33	1.03
Rover	9.18	9.18	**4.22**	1.18	0.88
Satellite	178.74	181.47	**7.68**	162.18	4.74

Table 5. Avg. runtime in sec. Best privacy preserving alg. is in bold

To obtain a finer grained comparison, we analyze the runtime of the different algorithms. To do so, we averaged the runtime of the instances solved by all algorithms in each domain. This follows the maximally conservative approach [2]. Table 5 shows these results, emphasizing in bold the best privacy preserving algorithm.

The results show two very clear trends. First, GPPP with PP-LM is the best performing privacy preserving algorithm, substantially outperforming the baseline MAFS with Proj-LM, as well as MAFS with PP-LM. Furthermore, GPPP substantially outperformed MAFS even if privacy is ignored and Full-LM is used. In fact, except in the rover domain GPPP with PP-LM outperforms MAFS even if MAFS ignores privacy and uses Full-LM. In general, GPPP is in most domains at least an order of magnitude faster than MAFS when using the same landmark detection algorithm.

The second trend that can be seen in Table 5 is that using a stronger landmark detection algorithm, i.e., an algorithm that detects more landmarks, results, in general, in better performance. GPPP with Full-LM performs better than GPPP with PP-LM. Not presented in this table, we also report that in preliminary experiments we also observed that GPPP with Proj-LM resulted in inferior results to GPPP with PP-LM. The same trend is shown in MAFS. With Proj-LM it performed worse than PP-LM in domains where PP-LM found more landmarks. Even in domains where PP-LM did not find more landmarks (xenotravel, rover, and satellite), MAFS with PP-LM performed the same or only slightly worse than MAFS with Proj-LM. By contrast, the gain of using PP-LM over Proj-LM when more landmarks are found, is substantial. MAFS with either Proj-LM and PP-LM performed worse than Full-LM, which detects more landmarks.

	Proj-LM	PP-LM		Full-LM	
Domain	MAFS	MAFS	GPPP	MAFS	GPPP
ProbLogistics	65.8	**55.4**	58.5	55.4	58.7
Elevators	33.1	**32.1**	33.5	34.9	31.1
MA-Blocks	40.1	41.8	**38.0**	35.7	37.6
Zenotravel	32.9	33.1	**27.7**	24.2	27.5
Rover	**34.4**	**34.4**	43.9	37.6	44.4
Satellite	**34.7**	**34.7**	36.5	35.9	36.5

Table 6. Avg. solution length. Best privacy preserving alg. is in bold

[4] A comparison with centralized, privacy ignoring planners is not the focus of this paper, and requires a more comprehensive evaluation of recent state-of-the-art planners

As both MAFS and GPPP do not directly attempt to find short solutions, most of our analysis has focused on runtime. However, for completeness, we report in Table 6 the average solution length found by each of the algorithms. We marked in bold the algorithm that found the shortest solution. As can be seen, the found solution lengths are very similar, and there is no clear winner in this aspect. If guarantees on solution length are required, one may use PP-LM to detect landmarks and incorporate them in the optimal MAD-A* algorithm (the optimal variant of MAFS) [7].

6 Conclusion and Future Work

In this paper we proposed a new privacy preserving landmark detection algorithm called PP-LM. PP-LM is especially suited for the MA-STRIPS model, finding more landmarks than a projection-based landmark detection algorithm in domains where agents must collaborate in order to achieve goals. Experiments show that the identified landmarks of PP-LM can substantially improve the performance of the corresponding planner, while the additional overhead of finding them is usually small.

Using the landmarks found by PP-LM, we propose a novel, highly effective, privacy preserving MA-STRIPS planner named Greedy Privacy Preserving Planner (GPPP). GPPP uses the found landmark to first plan the public actions and then check if these public actions can be performed. In our experiments, GPPP was able to solve more instances, and more than an order of magnitude faster than MAFS, the best previously proposed privacy preserving planner.

We focused on the privacy preserving aspect of the MA-STRIPS model. As in previous work [1], we defined private and public actions and literals according to the domain description. In a more general setting, defining which information should be private would be done by the user. Furthermore, the definition of privacy preserving in this and previous MA-STRIPS works was binary: either private information is shared or not. Even if private information is not directly shared, an intelligent agent might infer some knowledge about it only by observing the publicly shared information. We are currently pursuing a more quantifiable privacy measure, possibly adapting existing privacy metric from the DCOP literature [3].

REFERENCES

[1] R. I. Brafman and C. Domshlak, 'On the complexity of planning for agent teams and its implications for single agent planning', *Artificial Intelligence*, **198**, 52–71, (2013).
[2] O. Etzioni and R. Etzioni, 'Statistical methods for analyzing speedup learning experiments', *Machine Learning*, **14**(3), 333–347, (1994).
[3] R. Greenstadt, B. J. Grosz, and M. D. Smith, 'SSDPOP: improving the privacy of DCOP with secret sharing', in *AAMAS*, (2007).
[4] J. Hoffmann, 'FF: The fast-forward planning system', *AI magazine*, **22**(3), 57, (2001).
[5] J. Hoffmann, J. Porteous, and L. Sebastia, 'Ordered landmarks in planning', *J. Artif. Intell. Res. (JAIR)*, **22**, 215–278, (2004).
[6] E. Karpas and C. Domshlak, 'Cost-optimal planning with landmarks', in *IJCAI*, pp. 1728–1733, (2009).
[7] R. Nissim and R. I. Brafman, 'Multi-agent A* for parallel and distributed systems', in *AAMAS*, pp. 1265–1266, (2012).
[8] R. Nissim and R. I. Brafman, 'Cost-optimal planning by self-interested agents', in *AAAI*, (2013).
[9] R. Nissim and R. I. Brafman, 'Distributed heuristic forward search for multi-agent systems', *CoRR*, (2013).
[10] Silvia Richter and Matthias Westphal, 'The LAMA planner: Guiding cost-based anytime planning with landmarks', *Journal of Artificial Intelligence Research*, **39**(1), 127–177, (2010).