Partial Disclosure of Private Dependencies in Privacy Preserving Planning*

Extended Abstract

Rotem Lev Lehman
Information Systems Engineering
Ben Gurion University, Israel
levlerot@post.bgu.ac.il

Guy Shani
Information Systems Engineering
Ben Gurion University, Israel
shanigu@bgu.ac.il

Roni Stern
Palo Alto Research Center, USA
Ben Gurion University, Israel
sternron@post.bgu.ac.il,rstern@parc.com

KEYWORDS

Privacy; Multi-agent; Planning

ACM Reference Format:
Rotem Lev Lehman, Guy Shani, and Roni Stern. 2021. Partial Disclosure of Private Dependencies in Privacy Preserving Planning: Extended Abstract. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 3 pages.

1 INTRODUCTION

Designing autonomous agents that act collaboratively to achieve a common set of goals has been a major goal for Artificial Intelligence research for many years. *Collaborative Privacy-Preserving Planning* (CPPP) is a multi-agent planning task in which agents need to achieve a common set of goals without revealing certain private information [2]. In particular, in CPPP an individual agent may have a set of private facts and actions that it does not share with the other agents. CPPP has important motivating examples, such as planning for organizations that outsource some of their tasks, and has recently received considerable attention from the academic community [1, 4, 7, 8, 10, 12].

When planning, agents either explicitly or indirectly publish *dependencies* between the public actions [6]. For example, in a multiagent Mars Rover scenario, the need to take a camera from a base station before sending an image, incurs a *private dependency* between these public actions. In both single search and a two-level approach some private dependencies are revealed during planning.

In many cases, however, the agents can construct a plan requiring only a small portion of the private dependencies. It may be preferable to reveal only a part of the dependencies, intuitively reducing the amount of disclosed private information. This raises the challenge of how to choose which private dependencies to share and which not to. We suggest 4 different heuristic methods for deciding which dependencies should be published first, and compare them experimentally on standard benchmarks from the CPPP literature. Our results show that in many domains using our heuristics allows computing a plan while publishing only a small portion of the private dependencies. We also analyze the makespan cost of the plans, showing that publishing fewer dependencies may not increase the plan cost significantly.

*A complete version of this paper can be found in [5].

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

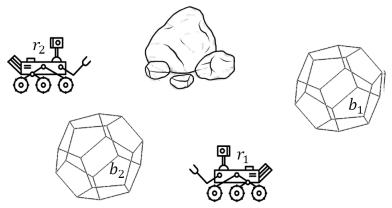

Figure 1: A rovers domain, with 2 base stations b_1 and b_2, and 2 rovers, r_1 and r_2, taking measurements of a rock.

2 PRIVACY PRESERVING PLANNING

Let $public(P)$ be the set of public facts in P, and let $private_i(P)$ be the set of variables that are private to agent i. Similarly, let $private_i(P)$ and $private_i(A_i)$ be the set of variables and actions, respectively, that are private to agent i. When a public action is executed, all agents are aware of its execution, and view the public effects of the action. The goals can be public, but can also be private to a single agent. An agent is aware only of its *local view* of the problem, that is, its private actions and facts, its public actions, the public facts, and the public projection of the actions of all other agents. That is, for public actions of other agents, the agent's local view contains only the public preconditions and effects of these actions.

Figure 1 illustrates a CPPP problem from the Rovers domain with 2 Mars Rovers. The Rovers need to perform sensor measurements on rocks, and, due to limited carriage capacity, can only carry 2 sensors at a time. Unused sensors are stored in base stations, and can be taken and returned to the base stations as needed. The public facts in this problem are the sensors located in bases, and the current condition of the target rock. The public actions are taking and returning sensors to the base stations, putting collected rock samples at the base stations, and performing various examination actions on rocks, such as taking an image, mining a mineral, or collecting a sample. The sensors held by a rover and its position are private, and the private actions are movement actions.

There are two main approaches for CPPP: *single search* and *two-level search*. Single search algorithms, such as MAFS [8] run a distributed forward search in which each agent searches for a plan using its own action space. Whenever an agent expands a state that was generated by applying a public action, it also broadcasts this state to all other agents, revealing a private dependency. Two-level search planners compute a public plan, known as a coordination scheme [3, 8, 11], and then each agent independently extends the public plan into a complete plan by adding private actions. In the

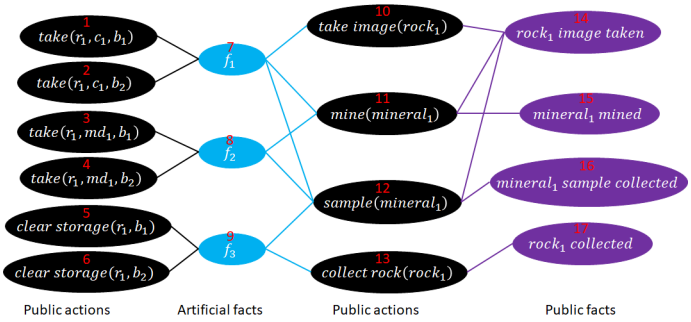

Figure 2: Local perspective of agent 1 private dependencies in the Rovers domain. Each node n is marked with a unique identifier colored red. $r_1 = rover_1, c_1 = camera_1, b_i = base_i, md_1 = mineral\_detector_1$

Dependency Projection Planner (DPP) [7], the agents compute together a single agent projection of the CPPP problem that captures the dependencies between public actions.

3 PARTIAL DISCLOSURE OF DEPENDENCIES

We now present the main contribution of this paper — reducing the amount of disclosed private dependencies and hence, the amount of disclosed information.

Figure 2 shows the local perspective of the private dependencies of agent 1 in the Rovers domain. Black nodes in the first and third columns represent public actions, purple nodes in the fourth column represent public facts, and blue nodes in the second column denote artificial facts that capture private dependencies between public actions. The public actions on the first column generate the artificial facts while the public actions on the third column require them as preconditions, and generate the public facts. Our methods always publish one of the black edges in the graph, between a public action and an artificial fact that it can generate. The blue and purple edges (artificial preconditions and public effects) are always known.

We suggest 4 different methods for selecting which dependencies to publish first. We take an iterative approach — all agents publish one artificial effect of one public action at each iteration. Our methods assign a heuristic score to each dependency. The dependency with the highest score is published at each iteration. If the public projection cannot be solved, all agents recompute the heuristic scores, and publish the next dependency.

Our first method, m_1, publishes dependencies that are used as preconditions in as many public actions as possible. The score of a dependency is set to the out-degree of the artificial fact (blue node). The second method, m_2, publishes an effect that enables the achievement of as many public facts as possible, denoted by outgoing purple edges in Figure 2. The third method, m_3, maximizes the amount of public actions that can be executed. That is, instead of publishing the dependency that provides a precondition for several actions, we publish the dependency that enables as many public actions as possible. The last method, m_4, is similar to m_3, but focuses on the public effects (purple nodes), not on the public actions. To avoid choosing to publish the same effect at the next iteration, the heuristic always measures the amount of new artifacts that will be disclosed at the next iteration.

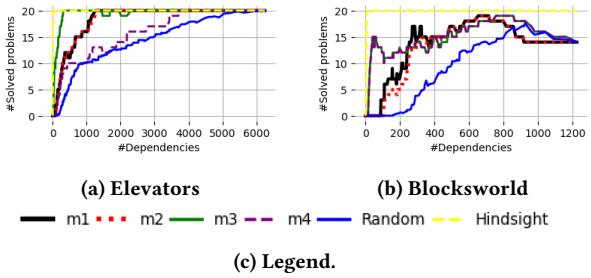

Figure 3: Number of solved problems for each amount of published dependencies in the Joint projection method.

4 RESULTS AND DISCUSSION

We evaluate our methods using standard benchmarks [9]. For each problem in each domain, we ran the projection-based solver with a growing number of revealed dependencies. Figure 3 shows the number of problems that were solved on two domains given the number of revealed dependencies by each agent. We also compute a "Hindsight value", the number of private dependencies in the final plan, and a random ranking (averaged over 10 executions).

m_3, which prioritizes enabling additional public actions, performs the best in all domains. The rest of the methods vary in their performance. On Blocksworld, m_4, that prioritizes achieving additional public facts, is the best together with m_3, but on Elevators m_4 performs the worst. On Elevators, arguably the domain with the highest amount of required collaboration, differences between methods are most pronounced. On some domains, not shown here, the heuristic methods performed only slightly better than random, but in most of the domains, they outperformed random selection.

An interesting phenomenon occur in, e.g., BlocksWorld and Rovers, where some problems are solved when many dependencies are unavailable, but cannot be solved when more dependencies are published. This is because the projection method may produce a public plan that cannot be extended into a complete plan. Our methods often publish first dependencies that resulted in plans that could be extended. Later, additional dependencies confused the planner to choose plans that could not be extended.

These results show that choosing which dependencies to reveal have an impact on solvability. While our heuristic methods perform on many domains much better than random, there is clearly much room for improvement. It is also interesting to study, given the domain features, which method will be appropriate for that domain.

Finally, we consider the makespan cost of the generated public plan. Even when a plan could be obtained without revealing any dependency, better plans may be obtained by collaboration, revealing some dependencies. Thus, agents may trade off some privacy for increased efficiency. In our experiments, however, on average, the improvement between the cost of the first plan found to the best plan found is only about 20%. That is, our heuristics chose dependencies that not only allowed for a solution, but also generated non-optimal, but relatively low cost plans.

ACKNOWLEDGMENTS

This work is partially funded by BSF grant #2018684 and ISF grant # 210/17 to Roni Stern, and by ISF grant # 1210/18 to Guy Shani.

REFERENCES

[1] Ronen I. Brafman. 2015. A Privacy Preserving Algorithm for Multi-Agent Planning and Search. In *IJCAI*. 1530–1536.

[2] Ronen I Brafman and Carmel Domshlak. 2008. From One to Many: Planning for Loosely Coupled Multi-Agent Systems. In *ICAPS*. 28–35.

[3] Ronen I. Brafman and Carmel Domshlak. 2013. On the complexity of planning for agent teams and its implications for single agent planning. *Artificial Intelligence* 198 (2013), 52–71.

[4] Patrick Caspari, Robert Mattmuller, and Tim Schulte. 2020. A Framework to Prove Strong Privacy in Multi-Agent Planning. In *Distributed and Multi-Agent Planning (DMAP) workshop at ICAPS*.

[5] Rotem Lev Lehman, Guy Shani, and Roni Stern. 2021. Partial Disclosure of Private Dependencies in Privacy Preserving Planning. *arXiv e-prints*, Article arXiv:2102.07185 (Feb. 2021), arXiv:2102.07185 pages. arXiv:2102.07185 [cs.MA]

[6] Shlomi Maliah, Guy Shani, and Roni Stern. 2016. Stronger Privacy Preserving Projections for Multi-Agent Planning. In *the International Conference on Automated Planning and Scheduling (ICAPS)*. 221–229.

[7] Shlomi Maliah, Guy Shani, and Roni Stern. 2018. Action dependencies in privacy-preserving multi-agent planning. *Autonomous Agents and Multi-Agent Systems* 32, 6 (2018), 779–821.

[8] Raz Nissim and Ronen I. Brafman. 2014. Distributed Heuristic Forward Search for Multi-agent Planning. *JAIR* 51 (2014), 293–332.

[9] Michal Štolba, Antonın Komenda, and Daniel L Kovacs. 2015. Competition of Distributed and Multiagent Planners (CoDMAP). *The International Planning Competition (WIPC-15)* (2015), 24.

[10] Alejandro Torreño, Eva Onaindia, Antonín Komenda, and Michal Štolba. 2017. Cooperative multi-agent planning: A survey. *Comput. Surveys* 50, 6 (2017), 1–32.

[11] Alejandro Torreno, Eva Onaindia, and Oscar Sapena. 2014. FMAP: Distributed cooperative multi-agent planning. *Applied Intelligence* 41, 2 (2014), 606–626.

[12] Jan Tozicka, Michal Štolba, and Antonín Komenda. 2017. The Limits of Strong Privacy Preserving Multi-Agent Planning. In *ICAPS*.