they publish by applying some cryptographic tool (**?**; **?**; **?**).

As noted by Brafman (**?**), this form of privacy preserving is weak, as there is no constraint on what other agents can *infer* from the information sent during planning and during execution. For example, if the public plan consists of an agent a_i picking up a package and the pickup action requires a truck to be present at the location of the package, then all agents now know that a_i controls at least one truck. Recent work by Brafman (**?**) considered a stronger and well-defined form of privacy preserving.

Definition 7 (Strongly Private)**.** *A variable or a specific value of a variable is* strongly private *if other agents cannot deduce its existence from the information available to them.*

The information available to an agent is assumed to be (1) its local view, (2) the messages passed between the agents during planning, and (3) the sequence of public actions (of all agents) in the resulting plan. A multi-agent planning algorithm is said to be *strongly privacy preserving* if all private information remains strongly private (**?**).

While appealing, achieving such a strong form of privacy may be difficult. In fact, the only algorithm proven so far to have this strict form of privacy – a secure version of MAFS – is only guaranteed to preserve this privacy in a short list of specific domains (logistics, satellites, rovers) and under very restricted conditions (unit action cost and heuristic functions that ignore private actions).

Object-Cardinality Privacy In the creation of the DP-projection we proposed as well as in the corresponding algorithm DPP, some information about the dependencies between public actions is published. Thus, any planner using it cannot by strongly privacy preserving. On the other hand, the DP-projection does not share explicitly any private information, and thus it does preserve the more common weaker form of preserving privacy claimed by most privacy preserving planners. Next, we define stronger privacy preserving property in which the *cardinality of private objects* is strongly private (Definition 7), and prove that generating the DP projection preserves this property.

Definition 8 (Cardinality of Private Objects)**.** *Overloading previous notation, denote by $private_t(a_i)$ the set of objects of type t that are private for agent a_i. The cardinality of private objects of type t for agent a_i is $|private_t(a_i)|$.*

The notion of *objects* and *types* has been native to the planning domain description language (PDDL) for a while now, and is used in the recent multi-agent extension of PDDL (**?**) and in the official domain descriptions used in the recent CoDMAP (**?**). Briefly, objects and types are used to conveniently parameterize actions and facts, such that an action or a fact can be defined with respect to an object of a given type. For example, in a logistics domain, the trucks able to pick up packages are all objects of the type *truck*. Then, the pick up action and location fact can be defined once to all trucks, parameterized by an object of type *truck*. For example, the action *load* is parameterized by an object t of type truck and an object p of type package, and its preconditions include the fact that t is at the same location as p. Concretely, the privacy extension of multi-agent PDDL (**?**) supports defining for every agent a_i a list of private objects of any type t, which is exactly $private_t(a_i)$.

Hiding the cardinality of private objects is motivated by real-world scenarios. Consider, for example, the logistic problem above. It is realistic to assume that the agents that collaborate in the planning task, i.e., the various delivery companies, know that packages are delivered using trucks between logistic centers. On the other hand, it is likely that each company would like to hide its logistics capabilities, such as the number of trucks that it controls, or the number of private logistic centers it maintains.

Formally, let us denote by M_i the messages sent by agent i to other agents during planning, and denote by T the set of all object types.

Definition 9 (Preserving Cardinality of a Type)**.** *The cardinality of type t for agent i is preserved during a planning process if no other agent a_j can infer the value of $|private_t(a_i)|$ from the set of messages M_i.*

There are cases where the cardinality of private objects is revealed by the local view of other agents. Consider for example an agent a_i controlling two trucks. If other agents are aware of two separate actions for loading the same package at the same logistic center by a_i, then the other agents can easily infer that a_i controls at least two trucks, even if the names of the objects or the action are obfuscated. Hence, the cardinality of objects that participate in preconditions of public actions may be compromised by the domain description, even before planning commences. Thus, in our logistics example, agents can only hide the number of private locations that they control.

Definition 10 (Preserving object cardinality privacy)**.** *A planning algorithm preserves object cardinality privacy if for any agent and type $t \in T$ the cardinality of private objects of type t can only be revealed during the planning process if it could have already been inferred from the local view prior to planning.*

Clearly, generating the DP projection (Algorithm 2) does not violate the weak form privacy, in the sense that no private fact is shared between the agents. Theorem 1 shows that this process also preserves object cardinality privacy.

Theorem 1 (DP Projection Preserves object cardinality privacy)**.** *The process of generating the DP projection preserves object cardinality privacy, and sharing the DP projection in a planning algorithm does not break object cardinality privacy.*

Proof outline: The DP projection relies on the dependency tree for each public action a_p of agent i. Given a regression tree, one could add private objects of any type t and private actions that are parametrized by them such that they will not modify the number of *true* leaves, or the sequence of public actions in each *true* branch (see Example 0.2). Moreover, we could add any number of private objects of type t and private actions that would modify the number of leaves, yet each leaf in the new tree would contain the same set of public actions as a leaf in the original tree. In these cases, although the number of private objects was modified, the projection does not change. □

Domain	GPPP	MAPR	PMR	MAPlan/ FF+DTG	PSM-VRD	DPP
blocksworld	12	**20**	**20**	**20**	**20**	**20**
depot	0	0	0	13	17	**19**
driverlog	14	**20**	19	17	**20**	**20**
elevators	**20**	19	19	11	12	**20**
logistics	**20**	19	0	18	18	**20**
rovers	19	19	**20**	**20**	12	**20**
satellites	14	**20**	12	14	18	**20**
sokoban	9	0	6	**18**	**18**	17
taxi	20	20	20	20	20	20
wireless	2	3	2	7	4	0	**9**
woodworking	15	0	0	0	12	13	**16**
zenotravel	**20**	**20**	18	**20**	13	**20**
sum	184	159	147	157	167	**224**

Table 1: Coverage results for a timeout of 30 minutes over the CoDMAP instances.

Experimental Results

We experimented with benchmarks from the 2015 CoDMAP competition (**?**). We run DPP on a 2.66 GHz machine with 4 cores and 8 GB of memory. Note tah while the computer had multiple cores, we implemented DPP to run on a single core. DPP was implemented using the FastDownward (FD) planner (**?**) for the high-level planning and the Fast-Forward (FF) planner (**?**) for extending high-level plans. FD was configured to use preferred operators, deferred heuristic evaluation, and two heuristics: FF and a landmark-based heuristic. This is a common configuration used also by the LAMA planner (**?**).

We compare DPP with the best performing and most relevant privacy-preserving planners: GPPP (**?**), MAPlan/FF+DTG (an extension of the MAFS algorithm (**?**) using the FF and DTG heuristics together), MAPR-p, PMR (**?**), and PSM-VRD (**?**; **?**). Details on these planners can be found in the CoDMAP website.

the domains in the competition. Our projection approach – DPP – solves more problems than all other approaches. In 7 of the 12 domains, DPP solved all instances, and in 2 more it solved 19 out of 20 problems. In all domains DPP is either the best performing algorithm (in terms of coverage) or within 3 problems of the best performing algorithm. This is in contrast to its closest rivals – GPPP and MAPlan – where GPPP is non-competitive in 4 domains and MAPlan in 2 domains. Thus, DPP shows robustness across all domains.

wireless. This is because the wireless domain has many dead-ends that are difficult to escape. Both FF and FD do not have a sufficiently strong mechanism to detect and avoid dead-ends. Still, in this domain too, DPP is the best-preforming planner.

DPP and all other planners is that once the projection was constructed we use an off-the-shelf classical planner, rather than a coordinated joint search mechanism. Even the construction of the DP projection can be easily distributed, having each agent add its projected actions and dependency facts. Our results point to the strengths of the mature classical planners over the rather new joint search mechanisms, such as GPPP (**?**) and MAFS (**?**), which is used by MAPlan.

We further analyze the actual size of the DP projection and runtime of creating it (using Algorithm 2). We chose a challenging problem on each domain, and compute various metrics during the projection construction. Table 2 reports the size of the original problem, in terms of the number of public and private actions of all agents, and the number of actions in the projection. We further report the maximal depth of a regression tree in our experiments, as well as the time it took to compute the projection in seconds.

Obviously, in domains where there are no private actions, our method is extremely fast. Satellite instances took much time, mainly due to the large number of private actions. Still,

[2] http://agents.fel.cvut.cz/codmap/results/

	Action count				
Domain	Public	Private	Projection	Depth	Time
blocks	2448	0	23256	2	0.057
depot	6134	0	74174	3	0.142
driverlog	56448	1248	1110720	2	2.66
elevators	768	112	447747	11	498.2
logistics	264	52	480	3	0.019
rover	1196	1642	1196	3	1.53
satellite	3139	42099	3139	2	19.9
sokoban	1856	0	4172	2	0.665
taxi	153	0	153	2	0.001
wireless	5814	0	113414	3	7.98
woodworking	8226	0	8226	2	0.168
zeno-travel	288	1098	1008	3	0.266

Table 2: Projection computation metrics over a large problem from each domain.

there is minor dependencies between the actions, resulting in shallow regression trees with a large branching factor.

more difficult instances of this domain contain private floors, that only a single elevator can reach. As boarding and exiting various quantities of passengers in floors requires different actions, there are many possible combinations for achieving the preconditions of actions, as reflected by the maximal depth of the regression trees in this domain. Hence, this is the only domain in the set of benchmarks that poses a true challenge to the regression tree computation.

ignore the identity of passengers, and focus only on their number, we can learn more general regression trees faster. Another method for scaling up would be to use an approximate regression mechanism, by, e.g., regressing only one precondition at a time.

Conclusion

process in which some information about private dependencies is shared while privacy is preserved. We showed that the DP projection also preserves a strong form of privacy in which the cardinality of private objects of a given type cannot be inferred by any adversary agent. The benefit of our DP projection is demonstrated in a planner that effectively uses the DP projection to solve more benchmark problems than any other state-of-the-art privacy preserving planner.

the DP projection as a heuristic for guiding the search of a complete solver, such as MAFS.

Acknowledgments: We thank the reviewers for their useful comments. This work was supported by ISF Grant 933/13, and by the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive Robotics Center of Ben-Gurion University of the Negev.