MAPR and CMAP

Daniel Borrajo and **Susana Fernández**
Universidad Carlos III de Madrid
Av. Universidad, 30
28911 Leganés, Spain
dborrajo@ia.uc3m.es;sfarregu@inf.uc3m.es

Abstract

We have developed a Multi-Agent Planning (MAP) framework to solve classical planning tasks for multiple cooperative agents with private information. It includes two new fully configurable MAP algorithms. In particular, we present to the CoDMAP competition three different configurations whose objectives are: minimize planning time; maximize quality of plans; and maximize the privacy level among agents. The main motivation for these systems is to be able to use any current state-of-the-art planner as the baseline problem solver; that is, our approach is planner-independent. Therefore, we can automatically benefit from advances in state-of-the-art planning techniques. They also avoid the extensive communication effort of other approaches.

Introduction

We have developed a MAP framework that can be configured in many different ways. First, we can choose between two MAP algorithms: MAPR and CMAP. Second, they can use several methods for assigning public goals to agents. Finally, we can select the underlying planner to be used. In this paper, we describe briefly all these options and at the end we report the actual submitted planners. This work has been partially reported in (Borrajo 2013b; 2013a).

The paper is structured as follows. Section describes MAPR and CMAP, and Section details the configuration that participated in the competition.

Multi-Agent Planning in our Framework

We have devised two alternative MAP algorithms: MAPR and CMAP. MAPR stands for Multi-Agent Planning by plan Reuse, while CMAP stands for Centralized MAP.

MAP Algorithms

The main steps of the algorithms are (for further details, we refer the reader to (Borrajo 2013b)):

1. Compilation of MA-PDDL into PDDL. It translates the input MA-PDDL format of CoDMAP into a PDDL file. As a side effect it extracts the agents' types and private predicates
2. Generation of m obfuscated PDDL domain and problem files (one for each agent ϕ_i)
3. Assignment of public goals to agents, while each agent might also have an additional set of private goals. As a side effect, a subset of agents are selected to plan for.
4. Planning using a state-of-the-art planner.
   - MAPR
     - It calls the first agent to provide a solution (plan) that takes into account its private and public goals.
     - Then, it iteratively calls each agent with the solutions provided by previous agents, augmented with domain and problem components needed to reproduce the solution (goals, literals from the state and actions). Each agent receives its own goals plus the goals of the previous agents. Thus, each agent solves its own problem, but taking into account the previous agents' augmented solutions.
     - Since previous solutions might consider private information, all private information from an agent is obfuscated for the next ones.
   - CMAP
     - It merges all the domains of the selected agents in step 3 into a single domain file and all the problem files into a single problem file. In both files, private information of each agent has been obfuscated.
     - It calls any single agent standard planner to solve the new problem

Goal Assignment

For each goal in $g \in G$ and agent in $\phi_i \in \Phi$, MAPR computes a relaxed plan from the initial state of the agent, I_i, following the relaxed plan heuristic of FF (Hoffman & Nebel 2001). If the relaxed plan heuristic detects a dead-end, then the cost of ϕ_i achieving g is $c(g, \phi) = \infty$. Otherwise, $c(g, \phi) = c(RP)$ where $c(RP)$ is the number of actions in the relaxed plan. All these costs define a cost matrix, $c(G, \Phi)$. We have devised eight goals assignment strategies: `all-achievable`, `rest-achievable`, `best-cost`, `load-balance`, `contract-net`, `all`, `subset` and `subgoals`. Next, we only describe the ones used in the configurations participating in the competition.

- `rest-achievable`: MAPR first assigns to the first agent ϕ_1 all goals that it can reach (cost less than ∞). Then, it removes those goals from the goals set, and assigns to the second agent all goals that it can reach from the remaining set of goals. It continues until the goals set is empty.
- `subset`: for each public goal $g_i \in G$, it computes the relaxed plan. However, instead of just computing the cost, it computes the subset of agents that appear in the relaxed plan for that goal, $\Phi_i \subseteq \Phi$. Those are agents that could potentially be needed to achieve the goal g_i. It is computed as the subset of agents that appear as arguments of any action in the relaxed plan. Then, it assigns g_i to all agents in Φ_i.
- `subgoals`: it is defined for MAPR only. The other assignment strategies assume that for each goal, there exists at least one agent that can achieve the goal by itself. This is not true in domains as logistics when agents are considered to be trucks and airplanes. In that case if an object has to move from the post-office of one city to the post-office of another city, none of the assignment strategies would allow MAPR to solve the goal individually for a given agent. Thus, we have devised a method that is inspired in (Maliah, Shani, & Stern 2014; Crosby, Rovatsos, & Petrick 2013). During goal assignment, a goal policy is computed. The goal policy is an ordered list of agent and subgoals that it needs to achieve at a given time step. At planning time, MAPR considers one policy step at a time, forcing the corresponding agent to achieve the goals specified at that policy step.

Obfuscation

In MAPR, if the first agent solves the problem, then it cannot pass the private information directly to the rest of agents. So, it obfuscates the private parts and outputs an augmented obfuscated planning problem. We have implemented different obfuscation levels. First, a *simple obfuscation* consists of a random substitution σ for the names of all private predicates, actions and objects. This is done when the individual domain and problem files are generated at the beginning. Second, a *further obfuscation* removes arguments from literals and also removes static predicates. Third, MAPR can learn and share macro-operators, so that privacy-preserving is further augmented by not sharing the individual private actions (either between two public actions, or all). We have defined two alternative methods: only-one-macro that generates one macro-operator for the complete plan of each agent; and several-macros that generates several macro-operators.

Plans Parallelization

We have implemented an algorithm to transform a sequential plan into a parallel plan. First, a suboptimal algorithm generates a partially-ordered plan from a totally-ordered one by using a similar algorithm by Veloso et al. (Veloso, Pérez, & Carbonell 1990). Then, a parallel plan is extracted from the partially-ordered plan. The parallelization algorithm is planner independent. It receives two inputs: a planning task, Π, and a sequential plan, π, that solves the task. It outputs a parallel plan that is one of the potential parallelizations of the sequential plan. This helps improving makespan, since in MAP that is a useful criteria for optimizing.

Table 1: Summary of properties for MAPR and CMAP.

Planner	sound	complete	optimal	privacy
MAPR	√	X	X	strong[1]
CMAP	√	√[2]	√[3]	weak

Properties

Table 1 shows a summary of the properties of MAPR and CMAP. The notes mean further constraints are needed:

- 1: strong privacy is achieved if only-one-macro is used. Otherwise, the privacy level is medium if more than one macro is used.
- 2: if we use `all` goal assignment (which we did not use in the version of the CoDMAP)
- 3: if we use `all` goal assignment (which we did not use in the version of the CoDMAP) and an optimal planner (which we did not use in the version of the CoDMAP)

Participating Configurations in CoDMAP

As the base planner of both MAPR and CMAP, we have used Fast-Downward code (Helmert 2006) to build a simplified version of LAMA-2011. It only performs the first greedy best-first search with the FF and LM-COUNT heuristics and preferred operators. Regarding cost models, we have used unit cost (all actions have a cost of one); we have named it LAMA-UNIT-COST.

We have submitted three configurations of our system:

- *planner1*: CMAP algorithm with the `subset` goal-assignment strategy and LAMA-UNIT-COST as the base planner. It aims at optimizing planning time and coverage.
- *planner2*: CMAP algorithm with the `subset` goal-assignment strategy and LAMA-2011 as the base planner. It aims at optimizing plans' quality and coverage.
- *planner3*: MAPR algorithm with the `subgoals` goal-assignment strategy, the `min-goals` sort-agent scheme, LAMA-UNIT-COST as the base planner and learning only-one-macro. It aims at maximizing privacy among agents.

Acknowledgments

This work has been partially supported by the Spanish government through MICINN project TIN2011-27652-C03-02.

References

Borrajo, D. 2013a. Multi-agent planning by plan reuse. Extended abstract. In *Proceedings of the AAMAS'13*, 1141–1142.

Borrajo, D. 2013b. Plan sharing for multi-agent planning. In *Preprints of the ICAPS'13 DMAP Workshop on Distributed and Multi-Agent Planning*, 57–65.

Crosby, M.; Rovatsos, M.; and Petrick, R. P. A. 2013. Automated agent decomposition for classical planning. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of ICAPS'13*. AAAI.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hoffman, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Maliah, S.; Shani, G.; and Stern, R. 2014. Privacy preserving landmark detection. In *Proceedings of ECAI'14*, 597–602.

Veloso, M. M.; Pérez, M. A.; and Carbonell, J. G. 1990. Nonlinear planning with parallel resource allocation. In *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*, 207–212. San Diego, CA: Morgan Kaufmann.