Prioritized Shaping of Models for Solving DEC-POMDPs*

(Extended Abstract)

Pradeep Varakantham†, William Yeoh†, Prasanna Velagapudi‡, Katia Sycara‡, Paul Scerri‡
†School of Information Systems, Singapore Management University, Singapore 178902
‡Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15217, USA
†{pradeepv, williamyeoh}@smu.edu.sg ‡{pkv, katia, pscerri}@cs.cmu.edu

ABSTRACT

An interesting class of multi-agent POMDP planning problems can be solved by having agents iteratively solve individual POMDPs, find interactions with other individual plans, shape their transition and reward functions to encourage good interactions and discourage bad ones and then recompute a new plan. D-TREMOR showed that this approach can allow distributed planning for hundreds of agents. However, the quality and speed of the planning process depends on the prioritization scheme used. Lower priority agents shape their models with respect to the models of higher priority agents. In this paper, we introduce a new prioritization scheme that is guaranteed to converge and is empirically better, in terms of solution quality and planning time, than the existing prioritization scheme for some problems.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI

General Terms

Algorithms, Experimentation

Keywords

DEC-POMDP, Uncertainty, Multi-Agent Systems

1. INTRODUCTION

Cooperative multi-agent and multi-robot teams in domains such as sensor networks and disaster rescue [1, 2] require that agents plan courses of action that achieve their joint objectives. In complex domains, where agents are faced with many options, uncertainty and risk, finding good plans can be computationally extremely difficult. An interesting class of multi-agent POMDP planning problems can be solved by having agents iteratively solve individual POMDPs, find interactions with other individual plans, shape their transition and reward functions to encourage good interactions and discourage bad ones then recompute a new plan. One such algorithm, Distributed-Team's Reshaping of Models for Rapid Execution (D-TREMOR), has been shown to efficiently compute POMDP plans for hundreds of agents, an order of magnitude scale up over most centralized, joint POMDP planners [2].

However, the speed and quality of the D-TREMOR planning process depends on how the models are shaped with respect to possible interactions. In this paper, we look at the priority ordering of agents when they shape their models to improve interactions. The intuitive idea is to give order to the agents and make lower priority agents plan around the plans of the higher priority agents. In decentralized prioritized planning, the agents can plan simultaneously with conflicts in the plans resolved in favor of the higher priority agents. One prioritization scheme that is shown to work well is prioritize agents that are more valuable to the team. We have applied this same concept to D-TREMOR, creating an algorithm called PD-TREMOR. Specifically, priorities are dynamically set based on the expected utility the agent computes for its local plan. Although these values change at each iteration, at least one additional agent's priority is fixed to ensure convergence.

2. BACKGROUND

We employ the DPCL model [2] to represent the problems of interest in this paper. DPCL is similar to the DEC-POMDP model in that they are both represented by the tuple of $\langle \mathbb{S}, \mathbb{A}, \mathbb{P}, \mathbb{R}, \Omega, \mathbb{O} \rangle$, where $\mathbb{S}, \mathbb{A}, \Omega$ are the joint states, actions and observations, respectively, and $\mathbb{P}, \mathbb{R}, \mathbb{O}$ are the joint transition, reward and observation functions, respectively. The primary difference between DPCLs and DEC-POMDPs is that the interactions between agents in DPCL are limited to *coordination locales* (CLs). CLs represent situations where the actions of one agent affect the local transition and reward functions of other agents. There are two kinds of CLs: positive and negative CLs. Intuitively, *positive CLs* are CLs where the effects result in a positive gain in joint rewards. Conversely, *negative CLs* are CLs where the effects result in a negative gain in joint rewards. Formally, a CL is defined as the tuple of $\langle t, \{(s_i, a_i)\}_1^n \rangle$, where t is the decision epoch, s_i is the local state of agent i and a_i is the action taken by agent i. The set of CLs is computed from the joint transition and reward functions.

D-TREMOR [3] is a distributed DPCL algorithm, where each agent iteratively solves its individual POMDP, broadcasts its individual plan to all other agents, computes its own CLs, shapes its own POMDP model taking into account its *active* CLs, that is, CLs with a high probability of occur-

*This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

Algorithm 1 PD-TREMOR(Agent i)

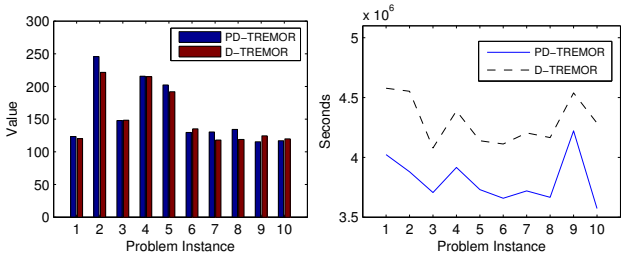
1: $\pi_i \leftarrow$ SolveIndividualPOMDP($\mathcal{M}_i$)
2: $iter \leftarrow 0$
3: **for all** $cl \in allCLs$ **do**
4: $R^0_{i,cl} \leftarrow$ SetInitialPriority($\mathcal{M}_i, cl, \pi_i$)
5: **while** $iter < MaxIterations$ **do**
6: $\alpha CLs \leftarrow$ ComputeActiveCLs($\mathcal{M}_i, allCLs, \pi_i$)
7: **for all** $cl \in \alpha CLs$ **do**
8: $val_{i,cl} \leftarrow$ EvaluateCL($\mathcal{M}_i, cl, \pi_i$)
9: CommunicateCL($i, cl, pr_{i,cl}, val_{i,cl}, R^{iter}_{i,cl}$)
10: $recCLs \leftarrow$ ReceiveCLs()
11: $\mathcal{M}_i \leftarrow$ ShapeModel($\mathcal{M}_i, recCLs, \{R^{iter}_{i,cl}\}$)
12: $\{\pi_i, val_i\} \leftarrow$ SolveIndividualPOMDP($\mathcal{M}_i$)
13: $iter \leftarrow iter + 1$
14: **for all** $cl \in allCLs$ **do**
15: $R^{iter}_{i,cl} \leftarrow$ UpdatePriority($val_i, val_{i,cl}$)

rence, and repeats the above steps until convergence or for a maximum number of iterations. The agents shape their POMDP models in two steps: (a) the individual transition and reward functions are modified in such a way that the joint plan evaluation is equal (or nearly equal) to the sum of individual plan evaluations; and (b) incentives or hindrances are introduced in the individual agent models based on whether a CL accrues extra reward or is a cost to the team members. This incentive/hindrance is the difference in the value of the plan for the team with the CL. To ensure convergence, D-TREMOR employs two mechanisms – probabilistic shaping of agent models to resolve positive CLs and a prioritization scheme that determines which agent model to shape to resolve negative CLs.

3. PD-TREMOR

Unfortunately, the prioritization scheme used by D-TREMOR is rather ad-hoc. It is based on agent IDs, which are arbitrary. As a result, one can construct simple examples where the scheme can lead to arbitrarily bad results. We thus introduce a prioritization scheme that associates the priority of an agent with the expected value of its individual plan. The larger the expected value of an agent, the higher the priority of that agent. The intuition is that agents with small expected rewards should shape their models so that they can find individual plans with higher expected rewards. This scheme is dynamic across iterations since the individual plans can change across iterations. However, this scheme ensures that the priority of at least one agent is fixed at each iteration to ensure convergence. Thus, it takes at most n iterations to fix the agent models of all agents.

We implement this scheme over D-TREMOR and refer to the new extension as *Prioritized D-TREMOR* (PD-TREMOR). Algorithm 1 shows the pseudocode. The overall algorithm has the same distributed structure as the D-TREMOR, with each "self" agent i running in parallel to other agents in the system. Each agent i now maintains a priority $R^{iter}_{i,cl}$ for each CL cl and iteration $iter$. Each agent starts by computing its individual plan π_i assuming no other agents exist in the environment (line 1) and sets its initial priorities with the SetInitialPriority() function (lines 3-4). It then computes its active CLs (line 6) and for each active CL, it evaluates the effect of that CL (line 8) and broadcasts that information together with its priorities to the other agents (line 9). Upon receiving the



(a) Solution Quality (b) Runtime

Figure 1: Experimental Results

CL and priority information of all other agents (line 10), each agent shapes its model according to those priorities with the ShapeModel() function (line 11). Intuitively, for each CL, low priority agents shape their models in favor of higher priority agents. Finally, each agent solves its individual POMDPs with its newly shaped model (line 12) and repeats these steps for a maximum number of iterations (line 5).

4. EXPERIMENTS

We run experiments using the disaster rescue problem described in [2]. Each problem instance was solved once by PD-TREMOR and 5 times with (non-prioritized) D-TREMOR. As D-TREMOR contained probabilistic shaping heuristics, these multiple runs were necessary to measure characteristic performance, which was unnecessary for PD-TREMOR's deterministic prioritization heuristics. However, we show results for problems with *only* negative interactions only due to space constraints. Our algorithm failed to perform statistically better for problems containing positive interactions.

Figure 1(a) shows the expected value of the solutions (joint plans) generated by D-TREMOR and PD-TREMOR. As D-TREMOR is stochastic, its performance data is displayed as a boxplot over each of the 10 problem instances. While overall value was highly specific to individual map instances, PD-TREMOR's plans consistently matched and exceeded the value of average D-TREMOR plans on most maps. This suggests that dynamic prioritization offers competitive performance when resolving negative interactions. An additional benefit of the dynamic prioritization can be seen in Figure 1(b), a plot of the total time taken for 10 iterations of each algorithm. Here, the solid line represents PD-TREMOR, and the dotted line represents the time taken by D-TREMOR. In every case, PD-TREMOR is able to complete the same number of iterations faster, implying that it is performing a more efficient search.

5. REFERENCES

[1] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked Distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of AAAI*, pages 133–139, 2005.

[2] P. Varakantham, J. Kwak, M. Taylor, J. Marecki, P. Scerri, and M. Tambe. Exploiting coordination locales in distributed POMDPs via social model shaping. In *Proceedings of ICAPS*, pages 313–320, 2009.

[3] P. Velagapudi, P. Varakantham, P. Scerri, and K. Sycara. Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In *Proceedings of AAMAS*, pages 955–962, 2011.