Experimental Results

We now provide some proof-of-concept experimental results showing that our algorithm can solve considerable size QDec-POMDP problems. We experiment with a variant of the box pushing problem (**?**) where a set of boxes are spread in a grid, and the agents must push each box to a designated location at the edge of the grid (the end of the column it appears in). Each box may be either in a pre-specified location, or at its goal location to begin with, and the agent must be in the same location as the box in order to observe where it is. Agents may move in the 4 primary directions, and can push boxes in these 4 primary directions, if they occupy the same location as the box. Some boxes are heavy and must be pushed by a few agents jointly (in our example, heavy boxes are pushed by 2 agents). Agents can also only observe the location of other agents when they are in the same location. All transitions and observations in these problems are deterministic. We experimented with four box pushing domains. The smallest example that we tried was a 2×2 grid, with 2 boxes and 2 agents and the largest had a 3×3 grid with 3 boxes. Each A_i has 11 possible actions (4 move actions, 4 push actions, observing the other agent, and observing each box), and hence there are 121 joint actions. We ran two Dec-POMDP solvers on this fully deterministic Dec-POMDP problem — the GMAA-ICE algorithm with the Q_{MDP} search heuristic (**?**) using the MADP package[1], and Incremental Policy Generation (IPG) (**?**). The results are presented in Table 1. Our compilation approach solves all the problems using the Fast Downward (FD) classical planner (**?**), while IPG solves only the smallest instance, and GMAA-ICE solves the smaller instances but not the larger one. Manually observing the trees, we saw that the planner computed the intuitive plan tree.

We acknowledge that this comparison is not entirely fair, because Dec-POMDP solvers try to optimize solution quality, whereas we only seek a satisfying solution. Thus, Dec-POMDP solvers may need to explore many more branches of the search graph, at a much greater computational cost. Furthermore, many Dec-POMDP solvers are naturally anytime, and can possibly produce a good policy even when stopped before termination. It may well be that solvers may reach a satisfying policy, which is the goal in a QDec-POMDP, well before they terminate their execution. That being said, our experiments demonstrate that our approach can provide solutions to decentralized problems and may be competitive with current Dec-POMDP solvers. Our experiments investigate scaling up in terms of states and the horizon, yet another source of complexity in Dec-POMDP problems is the number of agents. It would be interesting to examine in future work how our approach scales with the number of agents.

An interesting aspect of our approach is the ability to compactly represent large problems. For example, the 3×3 box pushing example that we describe above, required a model size of over 1GB (specifying only non-zero probabilities) in the traditional Cassandra format for Dec-POMDPs, while our factored representation required less than 15KB.

[1]`staff.science.uva.nl/~faolieho/madp`

Conclusion

We presented a new model for multi-agent planning problems, called QDec-POMDP, which emphasizes valid, rather than optimal solutions, that achieve a given goal, in the spirit of classical and contingent planning. We analyzed the complexity of the new model, concluding that it is as hard as the standard Dec-POMDP model for a given horizon. Then, we presented a factored version of this model, motivated by similar representations used in classical and contingent planning. Our representation is compact and can describe models with tens of thousands of states and about 150 joint actions using file sizes of less than 15KB. We intend to investigate even more compact methods for specifying the effects of joint actions. Next, we described a solution method for deterministic QDec-POMDPs, based on a compilation approach to classical planning. Our method creates a classical planning problem whose solution is a linearized joint plan tree. We demonstrated the advantage of this compilation method over Dec-POMDP solvers using a number of examples. Our approach solves small problems much faster and scales to larger problems compared to existing Dec-POMDP solvers.

In this paper, our focus was on providing an exposition of the model, its properties, and potential. Of course, this is only the first step towards developing more scalable solvers for QDec-POMDP domains. In particular, we know well from contingent planning that it is much harder to scale up offline solution methods. Hence, we intend to explore online planning in QDec-POMDPs. This raises some non-trivial challenges as we will need some mechanism that will allow different agents with different belief states to jointly plan (**?**), unlike the offline case in which a global plan is generated for a group of agents that share an initial belief state. The advantage, however, is that agents can focus on the relevant part of the state space at each planning phase, requiring smaller encodings and smaller plans. In addition, online methods are likely to better deal with non-deterministic effects. A second possible direction for scaling up would allow agents to plan independently, enforcing certain constraints on the joint solution.

Finally, it would be interesting to study variants of the QDec-POMDP model in more detail to identify the sources of its complexity, and, in particular, variants that have lower complexity. For example, we suspect that solving QDec-POMDPs with deterministic transitions might belong to a lower complexity class. Additional insights concerning belief state representation may also help yield more efficient algorithms.

Acknowledgments

Support for this work was provided in part by the National Science Foundation under grants IIS-0915071 and IIS-1116917, the Paul Ivanier Center for Robotics Research and Production Management and the Lynn and William Frankel Center for CS Research.

Voluptate pariatur accusamus sit similique impedit non re-
sumenda soluta ut nam est, eos a rerum nobis impedit?