is randomly chosen. This structure ties all agents together since the reward model cannot be decomposed among subgroups of agents. In an effort to provide insight on the degree of interaction among all agents, we distinguish between four classes \( \{c_0, c_1, c_2, c_3\} \), each of which depends on the number of interaction events \( e \). For each class \( c_k \), we randomly choose \( e \) such that \( e \in \left[ \frac{k}{4} e_{\max}, \frac{k}{4} (1 + e_{\max}) \right] \), where \( e_{\max} \) denotes the number of joint state and action pairs.

As depicted in Figure 2, the constraint formulation allows us to deal with larger numbers of agents. We calculated optimal value functions for 100 instances of each class, and reported the average computational time. The COP variant was able to scale up to 6 agents at horizon 10 in about 5,000 seconds. We could also produce results for up to 10 agents in about 40,000 seconds using a more powerful machine. It can also be seen that increasing the number of interaction events on each problem does not substantially increase the amount of time required to solve these problems. This shows that even for dense reward matrices, our approach will continue to perform well. Despite this high running time, MPS is the first generic algorithm that scales to teams of more than two agents without taking advantage of the locality of interaction. For example the BLP algorithm as it currently stands can only solve two-agent problems. Moreover, ND-POMDP techniques exploit the small number of local interactions among agents to scale to multiple agents, but in this problem all agents interact.

As depicted in Figure 2, the constraint formulation allows us to deal with larger numbers of agents. We calculated optimal value functions for 100 instances of each class, and reported the average computational time. The COP variant was able to scale up to 6 agents at horizon 10 in about 5,000 seconds. We could also produce results for up to 10 agents in about 40,000 seconds using a more powerful machine. It can also be seen that increasing the number of interaction events on each problem does not substantially increase the amount of time required to solve these problems. This shows that even for dense reward matrices, our approach will continue to perform well. Despite this high running time, MPS is the first generic algorithm that scales to teams of more than two agents without taking advantage of the locality of interaction. For example the BLP algorithm as it currently stands can only solve two-agent problems. Moreover, ND-POMDP techniques exploit the small number of local interactions among agents to scale to multiple agents, but in this problem all agents interact.

In the future, we plan to explore extending the MPS algorithm to other classes of problems and larger teams of agents. For instance, we may be able to produce an optimal solution to more general classes of Dec-MDPs or provide approximate results for Dec-POMDPs by extending the idea of an occupancy distribution to those problems. Furthermore, the scalability of our approach to larger numbers of agents is encouraging and we will pursue methods to increase this even further. In particular, we think our approach could help increase the number of agents that interact in conjunction with other structure in the model such as locality of interaction (as in ND-POMDPs) or sparse joint reward matrices (as in bilinear programming approaches).

8 Acknowledgements

We would like to thank Frans Oliehoek and the anonymous reviewers for their helpful comments about initial versions of the paper as well as Matthijs Spaan and Marek Petrik for providing algorithmic results and code respectively. Research supported in part by AFOSR MURI project #FA9550-091-0538.

References


