The MADP Toolbox: An Open-Source Library for Planning and Learning in (Multi-)Agent Systems

Frans A. Oliehoek
University of Liverpool,
University of Amsterdam
fao@liverpool.ac.uk

Matthijs T. J. Spaan
Delft University of Technology
Delft, The Netherlands
m.t.j.spaan@tudelft.nl

Philipp Robbel
Media Lab, MIT
Cambridge, MA, USA
robbel@mit.edu

João Messias
University of Amsterdam
Amsterdam, The Netherlands
jmessias@uva.nl

Abstract

This article describes the MultiAgent Decision Process (MADP) toolbox, a software library to support planning and learning for intelligent agents and multiagent systems in uncertain environments. Some of its key features are that it supports partially observable environments and stochastic transition models; has unified support for single- and multiagent systems; provides a large number of models for decision-theoretic decision making, including one-shot decision making (e.g., Bayesian games) and sequential decision making under various assumptions of observability and cooperation, such as Dec-POMDPs and POSGs; provides tools and parsers to quickly prototype new problems; provides an extensive range of planning and learning algorithms for single- and multiagent systems; and is written in C++ and designed to be extensible via the object-oriented paradigm.

Introduction

Decision making is a core topic of research in the fields of artificial intelligence and machine learning. Especially research on planning and learning in stochastic, partially observable and/or multiagent settings (MASs) has received much attention in the last decade, since these settings hold great promise to deal with challenging decision making problems encountered in the real world.

For instance, *Markov decision processes (MDPs)* can be used for aircraft collision avoidance (Kochenderfer and Chryssanthacopoulos 2011), *partially observable MDPs (POMDPs)* may enable active perception and certain robotics applications (Kaelbling, Littman, and Cassandra 1998; Spaan 2012; Hsu, Lee, and Rong 2008; Spaan, Veiga, and Lima 2014), *decentralized POMDPs (Dec-POMDPs)* (Bernstein, Zilberstein, and Immerman 2000; Oliehoek 2012) enable reasoning about the behavior of a team of robots or other decision makers acting on local information (Amato et al. 2015), and *partially observable stochastic games (POSGs)* allow representing situations with self-interested agents (Hansen, Bernstein, and Zilberstein 2004).

All these models are closely related and instances of what we refer to as *multiagent decision processes (MADPs)*. While many software libraries are available for planning or learning in specific sub-classes of MADPs—e.g., there are many toolboxes focusing on single-agent, fully observable reinforcement learning—no comprehensive libraries are available that support the more complex partially observable multiagent settings. The MADP Toolbox aims to fill this void by providing the building blocks for developing planning and learning algorithms for existing and novel instances of MADPs.

Some important features of the MADP Toolbox are:

Toolkit and Library. It can be used in two ways: as a toolbox of a number of planning and learning algorithms and as a software library to develop one's own models and algorithms.

Modular design. It is object-oriented, making it easy to extend to new problem classes, and to implement new planning and learning algorithms.

Flexible. It was inherently designed for multiagent problems, providing support for mixing different types of agents. It is also very suitable for single-agent problems, which are just a special case.

Feature rich. It includes many code-intensive features. For instance, it has support for factored models (including variable elimination and max-sum optimization), it supports basic inference in partially observable settings, and provides support for simulations of different communication scenarios. Finally, MADP comes equipped with a large number of state-of-the-art methods particularly for POMDPs and Dec-POMDPs.

Connected. It reads a number of problem formats: ProbModel XML (Arias et al. 2012), `.pomdp` file format (Cassandra 1998), `.dpomdp` file format, and exports SPUDD (Hoey et al. 1999) format. It also includes (parts of) the pomdp-solve library[1].

Documented. It comes with fairly extensive user and developer documentation. This is complemented by an API reference, as well as a user email list.

[1] `http://pomdp.org`

```
agents: 2
discount: 1
values: reward
states: tiger-left tiger-right
start:
uniform

actions:
listen open-left open-right
listen open-left open-right

observations:
hear-left hear-right
hear-left hear-right

T: * :
uniform
T: listen listen :
identity

O: * :
uniform
O: listen listen : tiger-left : hear-left hear-left : 0.7225
O: listen listen : tiger-left : hear-left hear-right : 0.1275
.
.
.
O: listen listen : tiger-right : hear-left hear-left : 0.0225

R: listen listen: * : * : * : -2
R: open-left open-left : tiger-left : * : * : -50
.
.
.
R: open-left listen: tiger-right : * : * : 9
```

Figure 1: The Dec-Tiger problem as a `.dpomdp` file (partial listing).

In the remainder of this document we provide a concise formalization of what we mean by a *multiagent decision process (MADP)*, and how the toolbox provides support for these, discuss technical details and related software.

A Brief Overview: Planning & Learning in (Multi-)Agent Systems

An MADP is a decision problem for one or more agents, considered for a particular number h of discrete time steps, or stages $t = 0, 1, 2, ..., h-2, h-1$. The basic idea is that at each stage the agents will take an action as a result of which the environment changes, and agents are provided with new feedback. MADP models are defined over (typically discrete and finite) sets of states, actions and observations. State transitions are governed by a Markovian transition model that depends on the current state and the action of each agent. In the canonical case, the environment is only partially observable and each agent receives its own individual observations, drawn from a probabilistic observation model. At each time step, an agent receives a scalar reward signal and its objective is to optimize the (discounted) sum of rewards it will receive. In the cooperative case, known as a Dec-POMDP, agents share the same reward function, while in the non-cooperative case, a POSG, each agent possesses its own reward function. POMDPs (Kaelbling, Littman, and Cassandra 1998) are the special case of having a single agent, while a multiagent MDP (Boutilier 1996) is the special case where the state is fully observable by all agents. The traditional MDP is the special case that combines these special cases. Finally, one-shot problems, such as strategic games

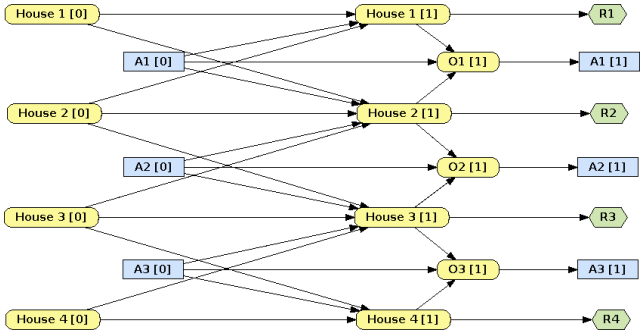

Figure 2: The Firefighting Graph problem in OpenMarkov.

or Bayesian games (Osborne and Rubinstein 1994), can be interpreted as special cases with a single stage.

These frameworks have in common the assumption that each agent will try and optimize its behavior in order to maximize the (sum of) reward(s) assigned to it. As such, the reward functions have the role of specifying the task and the agents' behavior results from a form of (potentially strategic) optimization.

Roughly speaking, MADPs can be approached in two manners. In the first, one assumes that the model is given and that the task is to find behavior that is optimal ('planning') or rational according to some definition ('solving a game'). In the second approach, one assumes that the model is not known in advance, but that agents *learn* to interact during simulations ('multiagent reinforcement learning'). This means that core functionality required for supporting MADP research is the ability to specify models for and performing simulations of MADPs.

As such, a key functionality that MADP provides is the ability to conveniently specify MADPs. Three main methods are supported currently: 1) using Anthony Cassandra's plain text `.pomdp` file format or a multiagent `.dpomdp` extension of that (see Figure 1 for an example), 2) in the graphical OpenMarkov[2] editor for ProbModelXML files (see Figure 2), or 3) by writing your own class that implements the appropriate interface.

Finally, the aim of the toolbox is not only to facilitate specification, but also to provide existing algorithms for planning and learning, as well as the building blocks to easily develop new such algorithms. To this end, various types of classes are provided by the toolbox:

- Classes that represent different models (i.e., different instances of MADPs). Currently interface classes are defined for all mentioned models above (POSGs, Dec-POMDPs, MMDPs, Bayesian Games etc.), and a number of benchmark problems (e.g., Dec-Tiger, factored firefighting, etc.) are available.
- A collection of support classes that represent histories, beliefs, value functions, and policies, etc., as well as some algorithms to perform belief updating and approximate inference.

[2] `http://www.openmarkov.org`

Name	Type	MAS	PO	Language	URL
APPL	planning	–	√	C++	http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/
BURLAP	planning/RL	√	–	Java	http://burlap.cs.brown.edu
AI-Toolbox	planning/RL	–	√	C++	https://github.com/Svalorzen/AI-Toolbox
RLlib	RL	–	–	C++	https://github.com/HerveFrezza-Buet/RLlib
RLPy	RL	–	–	Python	https://bitbucket.org/rlpy/rlpy
MARL Toolbox	RL	√	–	Matlab	http://busoniu.net/repository.php
PyBrain	ML/RL	–	–	Python	http://pybrain.org
markov decision making	robot planning	√	√	ROS	http://wiki.ros.org/markov_decision_making

Table 1: Comparison to closely related software and toolboxes.

- Classes that represent different planning algorithms for different models. These specify interfaces, such as an interface to support planning for Dec-POMDPs, as well as complete planning methods, including: value iteration, Perseus (Spaan and Vlassis 2005), Monahan's algorithm (Monahan 1982), Incremental Pruning (Cassandra, Littman, and Zhang 1997), JESP (Nair et al. 2003), DP-LPC (Boularias and Chaib-draa 2008), (Generalized) Multiagent A* (GMAA) and variants (Szer, Charpillet, and Zilberstein 2005; Oliehoek, Whiteson, and Spaan 2009; Spaan, Oliehoek, and Amato 2011; Oliehoek et al. 2013), GMAA-ELSI for factored Dec-POMDPs (Oliehoek et al. 2008).
- Classes that represent types of agents that can be employed (individually or in a team) in a simulation to facilitate learning algorithms. Again, these include interfaces, such as an interface for agents with full observability or agents that share observations via communication, as well as fully specified agent behaviors, such as a Q-learner agent, or an agent performing on-line MDP planning.

For more detailed information, we refer to the included developer documentation.

Technical Details

The MADP Toolbox can be used in two ways: as a toolbox and as a software library. The fact that MADP can be used as a library also means that it can support the deployment and real-time execution of decision-theoretic control policies for autonomous agents (such as the markov_decision_making ROS package mentioned below). Since MADP has been implemented in C++ and is organized according to object-oriented paradigm, it is easy to extend. Since MADP aims to support the core planning tasks, C++ provides the additional advantage that it is very fast for such computationally expensive routines.

The toolbox can be downloaded at `https://github.com/MADPToolbox/MADP` and is developed under Debian GNU/Linux and has been tested under various recent Linux distributions. It also has (experimental) Mac OSX support. MADP 0.3.1 adheres to the C++98 standard and is relatively self-contained: required Boost header files and required libraries and the like are included. MADP comes with extensive documentation both in form of a user/developer manual, as well as API documentation (also available online).

Related Software

Many toolboxes exists for decision making, but few deal with MASs or partially observable settings. The most closely related software and toolboxes for planning, reinforcement learning (RL) and machine learning (ML) applications are shown in Table 1. Marked in the MAS and PO columns are whether the respective toolbox includes algorithms for multiagent or partially-observable settings. Closest to MADP's scope is BURLAP, which currently does not support partially-observable multiagent settings.

The MADP software distribution uses functionality from, and therefore includes, versions of POMDP-solve[3] and libDAI (Mooij 2008). The MADP Toolbox is the back-end a number of solution methods on the Thinc lab solver page[4]. It is also used in the markov_decision_making ROS package[5] in order to parse model files and query agent policies in real time.

Acknowledgments

We are grateful to Abdeslam Boularias for the basis of his DP-LPC code, Anthony Cassandra for allowing us to include POMDP-solve, Joris Mooij for allowing to include libDAI, and many others that made contributions to the MADP Toolbox.

The MADP Toolbox has been developed over the course of many years, and has thus been supported by various funding agencies including: The Dutch Ministry of Economic Affairs, AFOSR, NWO, FP7 Marie Curie, FCT. F.O. is currently supported by NWO Innovational Research Incentives Scheme Veni #639.021.336.

[3] `http://www.pomdp.org/code/`

[4] `http://lhotse.cs.uga.edu/pomdp/`

[5] `http://wiki.ros.org/markov_decision_making`

References

Amato, C.; Konidaris, G. D.; Cruz, G.; Maynor, C. A.; How, J. P.; and Kaelbling, L. P. 2015. Planning for decentralized control of multiple robots under uncertainty. In *Proceedings of the International Conference on Robotics and Automation*.

Arias, M.; Díez, F. J.; Palacios-Alonso, M. A.; Yebra, M.; and Fernández, J. 2012. POMDPs in OpenMarkov and ProbModelXML. In *Multiagent Sequential Decision-Making Under Uncertainty*. Workshop at AAMAS12.

Bernstein, D. S.; Zilberstein, S.; and Immerman, N. 2000. The complexity of decentralized control of Markov decision processes. In *Proc. of Uncertainty in Artificial Intelligence*, 32–37.

Boularias, A., and Chaib-draa, B. 2008. Exact dynamic programming for decentralized POMDPs with lossless policy compression. In *Proc. of the International Conference on Automated Planning and Scheduling*.

Boutilier, C. 1996. Planning, learning and coordination in multiagent decision processes. In *Proc. of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, 195–210.

Cassandra, A.; Littman, M. L.; and Zhang, N. L. 1997. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proc. of Uncertainty in Artificial Intelligence*, 54–61. Morgan Kaufmann.

Cassandra, A. R. 1998. *Exact and approximate algorithms for partially observable Markov decision processes*. Ph.D. Dissertation, Brown University.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *Proc. of the National Conference on Artificial Intelligence*, 709–715.

Hoey, J.; St-Aubin, R.; Hu, A. J.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proc. of Uncertainty in Artificial Intelligence*, 279–288.

Hsu, D.; Lee, W. S.; and Rong, N. 2008. A point-based POMDP planner for target tracking. In *Proceedings of the International Conference on Robotics and Automation*, 2644–2650.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134.

Kochenderfer, M. J., and Chryssanthacopoulos, J. P. 2011. Collision avoidance using partially controlled markov decision processes. In *Proc. of the International Conference on Agents and Artificial Intelligence*, 86–100.

Monahan, G. E. 1982. A survey of partially observable Markov decision processes: theory, models and algorithms. *Management Science* 28(1).

Mooij, J. M. 2008. libDAI: library for discrete approximate inference.

Nair, R.; Tambe, M.; Yokoo, M.; Pynadath, D. V.; and Marsella, S. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proc. of the International Joint Conference on Artificial Intelligence*, 705–711.

Oliehoek, F. A.; Spaan, M. T. J.; Whiteson, S.; and Vlassis, N. 2008. Exploiting locality of interaction in factored Dec-POMDPs. In *Proceedings of the Seventh Joint International Conference on Autonomous Agents and Multiagent Systems*, 517–524.

Oliehoek, F. A.; Spaan, M. T. J.; Amato, C.; and Whiteson, S. 2013. Incremental clustering and expansion for faster optimal planning in decentralized POMDPs. *Journal of Artificial Intelligence Research* 46:449–509.

Oliehoek, F. A.; Whiteson, S.; and Spaan, M. T. J. 2009. Lossless clustering of histories in decentralized POMDPs. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, 577–584.

Oliehoek, F. A. 2012. Decentralized POMDPs. In Wiering, M., and van Otterlo, M., eds., *Reinforcement Learning: State of the Art*, volume 12 of *Adaptation, Learning, and Optimization*. Berlin, Germany: Springer Berlin Heidelberg. 471–503.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. The MIT Press.

Spaan, M. T. J., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of AI Research* 24:195–220.

Spaan, M. T. J.; Oliehoek, F. A.; and Amato, C. 2011. Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2027–2032.

Spaan, M. T. J.; Veiga, T. S.; and Lima, P. U. 2014. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Journal of Autonomous Agents and Multi-Agent Systems*.

Spaan, M. T. J. 2012. Partially observable Markov decision processes. In Wiering, M., and van Otterlo, M., eds., *Reinforcement Learning: State of the Art*. Springer Verlag. 387–414.

Szer, D.; Charpillet, F.; and Zilberstein, S. 2005. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proc. of Uncertainty in Artificial Intelligence*, 576–583.