The MADP Toolbox

An Open Source Library for Planning and Learning in (Multi-)Agent Systems

Oliehoek, Frans A.; Spaan, Matthijs T. J.; Terwijn, Bas; Robbel, Philipp; Messias, João V.

Publication date
2017

Document Version
Final published version

Published in
Journal of Machine Learning Research

Citation (APA)
Oliehoek, F. A., Spaan, M. T. J., Terwijn, B., Robbel, P., & Messias, J. V. (2017). The MADP Toolbox: An Open Source Library for Planning and Learning in (Multi-)Agent Systems. *Journal of Machine Learning Research* , *18*(89), 1-5. http://jmlr.org/papers/volume18/17-156/17-156.pdf

Journal of Machine Learning Research 18 (2017) 1-5 Submitted 3/17; Revised 5/17; Published 8/17

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

Frans A. Oliehoek FAO@LIVERPOOL.AC.UK
Department of Computer Science, University of Liverpool
Liverpool, United Kingdom

Matthijs T. J. Spaan M.T.J.SPAAN@TUDELFT.NL
Delft University of Technology
Delft, The Netherlands

Bas Terwijn B.TERWIJN@UVA.NL
Informatics Institute, University of Amsterdam
Amsterdam, The Netherlands

Philipp Robbel ROBBEL@ALUM.MIT.EDU
Media Lab, Massachusetts Institute of Technology
Cambridge, MA, USA

João V. Messias JMESSIAS@UVA.NL
Informatics Institute, University of Amsterdam
Amsterdam, The Netherlands

Editor: Alexandre Gramfort

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. 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 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; is released under the GNU GPL v3 license; and is written in C++ and designed to be extensible via the object-oriented paradigm.
Keywords: software, decision-theoretic planning, reinforcement learning, multiagent systems

1. 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 systems (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 or certain robotics applications (Kaelbling et al., 1998; Spaan, 2012; Hsu et al., 2008), *decentralized POMDPs (Dec-POMDPs)* (Bernstein et al., 2000; Oliehoek and Amato, 2016) 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 for representing situations with self-interested agents (Hansen et al., 2004).

All these models are closely related and instances of what we refer to as *multiagent decision processes (MADPs)* (Oliehoek and Amato, 2016). 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 toolbox are:

Toolkit and Library. It can be used in two ways: as a toolbox containing 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 implement new planning and learning algorithms.

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

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

Connected. It reads a number of problem formats: ProbModelXML (Arias et al., 2012), `.pomdp` file format, `.dpomdp` file format, and exports SPUDD (Hoey et al., 1999) format.

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

2. Supported Problem Classes and Algorithms

Here we give a concise overview of the different MADP frameworks that are currently supported by the toolbox along with implemented algorithms.[1] The MADP Toolbox is designed to support both *planning*, i.e., settings where the model is given in advance, and reinforcement learning, i.e., settings where no such model is available, but the agents *learn* to interact during simulations.

Decentralized POMDPs. One of the most general models that the toolbox represents is the Dec-POMDP. Formally, a Dec-POMDP is a tuple $\langle \mathcal{D}, \mathcal{S}, \boldsymbol{\mathcal{A}}, T, R, \boldsymbol{\mathcal{O}}, O, h, b^0 \rangle$, where $\mathcal{D} = \{1, \ldots, n\}$ is the set of n agents, $\mathcal{S}$ is a set of states s, $\boldsymbol{\mathcal{A}}$ is the set of joint actions $\boldsymbol{a} = \langle a_1, \ldots a_n \rangle$, T is the transition function that specifies $\Pr(s^{t+1}|s^t, \boldsymbol{a}^t)$, $R(s, \boldsymbol{a})$ is the immediate reward function, $\boldsymbol{\mathcal{O}}$ is the set of joint observations $\boldsymbol{o} = \langle o_1, \ldots, o_n \rangle$, O is the observation function: $\Pr(\boldsymbol{o}^{t+1}|\boldsymbol{a}^t, s^{t+1})$, h is the horizon of the problem, $b^0 \in \triangle(\mathcal{S})$ is the initial state distribution at time $t = 0$. The typical goal of the agents is to optimize the expected (possibly discounted) cumulative reward. The toolbox supports a range of solution methods for Dec-POMDPs, including JESP (Nair et al., 2003), DP-LPC (Boularias and Chaib-draa, 2008), (Generalized) Multiagent A* (GMAA) and variants (Szer et al., 2005; Oliehoek et al., 2013), and direct CE (DICE) policy search (Oliehoek et al., 2008a). In general, GMAA-ICE is the fastest optimal solution method included, while DICE is the most scalable approximate method for Dec-POMDPs in the toolbox. The toolbox also includes methods to deal with 1-step delayed communication (Oliehoek et al., 2008b).

POMDPs. A Dec-POMDP with just a single agent or with instantaneous communication reduces to a POMDP (referred to as a *multiagent POMDP* in the latter case). The MADP Toolbox includes a number of solution methods for these: Perseus (Spaan and Vlassis, 2005), Monahan's algorithm (Monahan, 1982), incremental pruning (Cassandra et al., 1997) with accelerated vector pruning (Walraven and Spaan, 2017).

1. For more details on these frameworks, please see, e.g., Oliehoek and Amato (2016).

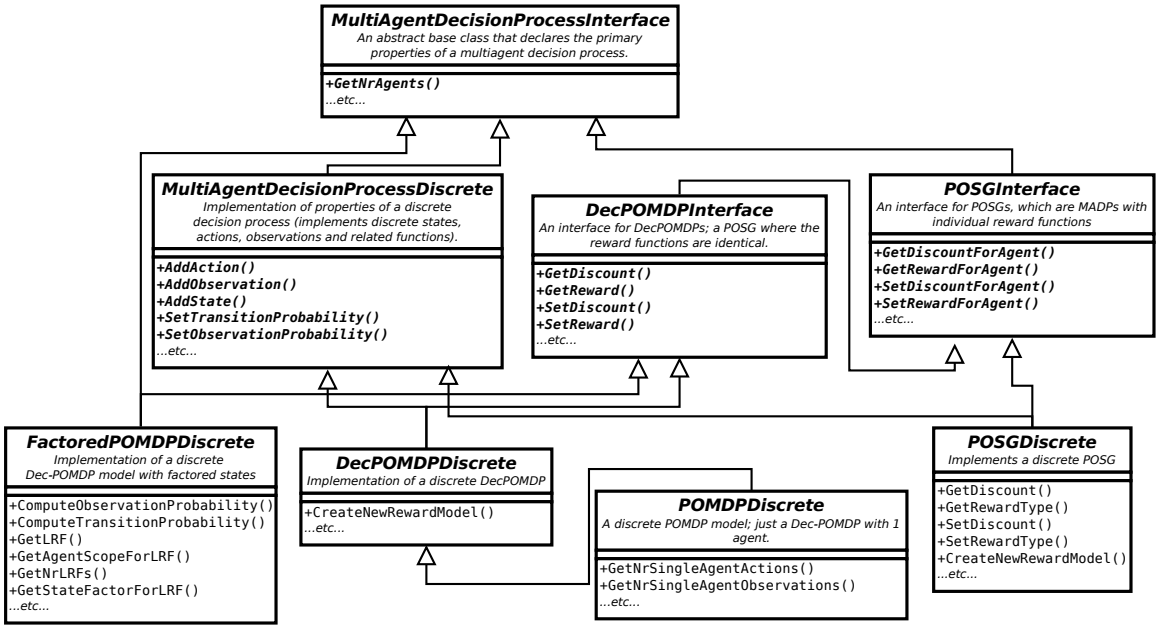

Figure 1: A simplified view of the object-oriented approach to representing problems. Illustrated is how both DecPOMDPDiscrete and POSGDiscrete inherit the implementation of states, actions, observations, etc. from MultiagentDecisionProcessDiscrete. In contrast, FactoredDecPOMDPDiscrete does not inherit this implementation.

MDPs. When the problem additionally is fully observable, the model simplifies further to the standard MDP. For these settings, the toolbox provides algorithms such as value iteration and policy iteration, as well as a GPU accelerated version of the latter. Additionally, there is a Q-learner agent and an agent performing on-line MDP planning.

Factored Models. Since the aforementioned models can be quite large for real problems, compact specification is an important issue. The toolbox includes support for compact *factored* representations (e.g., Oliehoek et al., 2008c, and references therein). For instance, it includes the GMAA-ELSI factored Dec-POMDP solver (Oliehoek et al., 2008c) and exports to SPUDD (Hoey et al., 1999).

Inheritance and Extensibility. The toolbox is designed to be extensible via the object-oriented paradigm. To illustrate this Figure 1 gives a (slightly simplified) illustration of how the discussed problem classes inherit from one another. The figure also shows that the toolbox support POSGs, which generalize Dec-POMDPs to include individual payoff functions.

Additional Functionality. A key functionality that the toolbox provides is the ability to conveniently specify MADPs. Three methods currently exist: 1) using Anthony Cassandra's plain text `.pomdp` file format or a multiagent `.dpomdp` extension, 2) using the graphical OpenMarkov[2] editor for ProbModelXML files, or 3) writing a C++ class that implements the appropriate interface.

The MADP Toolbox also provides a collection of support classes that represent histories, beliefs, value functions, and policies, etc., as well as some algorithms to perform belief updates and approximate inference. Additionally, as part of GMAA, the toolbox also provides classes to represent and solve collaborative (Graphical) Bayesian games (Oliehoek et al., 2008b, 2012).

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

Table 1: Comparison to related software and toolboxes

3. Technical Details

The software can be used in two ways: as a toolbox and as a library. The fact that it can be used as a library provides flexibility to, for instance, 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 it has been implemented in C++ and is organized according to the object-oriented paradigm, it is easy to extend. Furthermore, as the toolbox aims to support core planning tasks, C++ provides the additional advantage that it is very fast for such computationally expensive routines.

The MADP Toolbox is released under the GNU GPL v3 license and can be downloaded from `https://github.com/MADPToolbox/MADP`. It is developed under Debian GNU/Linux, has been tested under various recent Linux distributions and also supports Mac OS X. In order to provide reliability, the toolbox includes a basic set of tests which cover most of the included functionality. MADP Toolbox 0.4.1 adheres to the C++98 standard and is relatively self-contained: required Boost header files and required libraries are included. It comes with extensive documentation both in form of a user/developer manual as well as API documentation (also available online).

4. Related Software

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

The MADP software distribution uses functionality from and therefore includes BOOST and libDAI (Mooij, 2010). The MADP Toolbox is the back-end of a number of solution methods on the THINC lab solver page.[3] It is also used in the MARKOV_DECISION_MAKING ROS package in order to parse model files and query agent policies in real time.[4]

Acknowledgments

We are grateful to Joris Mooij for allowing to include libDAI and to many others that made contributions. 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. was also affiliated with the University of Amsterdam, supported by NWO Veni grant #639.021.336, during the writing of this article.

3. http://lhotse.cs.uga.edu/pomdp/
4. http://wiki.ros.org/markov_decision_making

References

C. Amato, G. D. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling. Planning for decentralized control of multiple robots under uncertainty. In *IEEE Int. Conference on Robotics and Automation*, pages 1241–1248, 2015.

M. Arias, F. J. Díez, M. A. Palacios-Alonso, M. Yebra, and J. Fernández. POMDPs in OpenMarkov and ProbModelXML. In *Multi-Agent Sequential Decision Making in Uncertain Domains*, 2012.

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

A. Boularias and B. Chaib-draa. Exact dynamic programming for decentralized POMDPs with lossless policy compression. In *Int. Conference on Automated Planning and Scheduling*, pages 20–28, 2008.

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

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

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

D. Hsu, W. S. Lee, and N. Rong. A point-based POMDP planner for target tracking. In *IEEE Int. Conference on Robotics and Automation*, pages 2644–2650, 2008.

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

M. J. Kochenderfer and J. P. Chryssanthacopoulos. Collision avoidance using partially controlled Markov decision processes. In *Int. Conference on Agents and Artificial Intelligence*, pages 86–100, 2011.

G. E. Monahan. A survey of partially observable Markov decision processes: theory, models and algorithms. *Management Science*, 28(1):1–16, 1982.

J. M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, 2010.

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

F. A. Oliehoek and C. Amato. *A Concise Introduction to Decentralized POMDPs*. Springer Briefs in Intelligent Systems. Springer, 2016.

F. A. Oliehoek, J. F. Kooi, and N. Vlassis. The cross-entropy method for policy search in decentralized POMDPs. *Informatica*, 32:341–357, 2008a.

F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353, 2008b.

F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *Int. Conference on Autonomous Agents and Multiagent Systems*, pages 517–524, 2008c.

F. A. Oliehoek, S. Whiteson, and M. T. J. Spaan. Exploiting structure in cooperative Bayesian games. In *Uncertainty in Artificial Intelligence*, pages 654–664, 2012.

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

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

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

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

E. Walraven and M. T. J. Spaan. Accelerated vector pruning for optimal POMDP solvers. In *AAAI Conference on Artificial Intelligence*, pages 3672–3678, 2017.