Multi-Agent Programming Contest 2012 - The Python-DTU Team

Villadsen, Jørgen; Jensen, Andreas Schmidt; Ettienne, Mikko Berggren; Vester, Steen; Balsiger Andersen, Kenneth; Frøsig, Andreas

Publication date:
2012

Document Version
Early version, also known as pre-print

Multi-Agent Programming Contest 2012

—

The Python-DTU Team

Jørgen Villadsen, Andreas Schmidt Jensen, Mikko Berggren Ettienne,
Steen Vester, Kenneth Balsiger Andersen, and Andreas Frøsig

Department of Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads, Building 321, DK-2800 Kongens Lyngby, Denmark

Abstract. We provide a brief description of the Python-DTU system, including the overall design, the tools and the algorithms that we plan to use in the agent contest.

Updated 1 October 2012: Appendix with comments on the contest added.

Introduction

1. The name of our team is Python-DTU. We participated in the contest in 2009 and 2010 as the Jason-DTU team [2,3], where we used the Jason platform [1], but this year we use just the programming language Python as we did in 2011 [4].
2. The members of the team are as follows:
   - Jørgen Villadsen, PhD
   - Andreas Schmidt Jensen, PhD student
   - Mikko Berggren Ettienne, MSc student
   - Steen Vester, MSc student
   - Kenneth Balsiger Andersen, BSc student
   - Andreas Frøsig, BSc student

   We are affiliated with DTU Informatics (short for Department of Informatics and Mathematical Modelling, Technical University of Denmark, and located in the greater Copenhagen area).
3. The main contact is associate professor Jørgen Villadsen, DTU Informatics, email: `jv@imm.dtu.dk`
4. We expect that we will have invested approximately 200 man hours when the tournament starts.

System Analysis and Design

1. The competition is built on the Java MASSim platform and the Java EISMASSim framework is distributed with the competition files. This framework is based on EIS and abstracts the communication between the server and the agents to simple Java method calls and callbacks. We decided to skip EISMASSim to instead follow a much cleaner Python-only implementation. Even though some work was needed to implement the protocol specific parts which EISMASSim handled, this left us with a more flexible implementation of which we have complete knowledge and control of every part of the implementation.

2. We do not use any existing multi-agent system methodology.

3. We do not plan to distribute the agents on several machines mainly for two reasons. Firstly, we had no need to, as we have enough computation power on a single machine to reason and send the action messages before the deadlines. Secondly, the shared data structure in our implementation would have to be replaced by a message server and a simple protocol. Due to limited time we have to prioritize differently.

4. We do not plan a solution with a centralization of coordination/information on a specific agent. Rather we plan a decentralized solution where agents share percepts through through shared data structures and coordinated actions using distributed algorithms.

5. Our communication strategy is to share all new percepts to keep the agents internal world models identical. Furthermore our agreement based auction algorithm heavily relies on communication and is part of how agents decide on goals.

6. We hope to achieve the following properties when designing an algorithm to assign goals to agents:
   (a) The total benefit of the assigned goals should be as high as possible. Preferably optimal or close to it.
   (b) The running time of the algorithm should be fast, since we need to assign goals to agents at every time step in the competition and still have time left for other things such as environment perception, information sharing, etc.
   (c) The algorithm should be distributed between the agents resembling a true multi-agent system.
   (d) It should not be necessary for the agents to have the same beliefs about the state of the world in order to agree on an assignment.
   (e) The algorithm should be robust. If it is possible, our agents should be able to agree on an assignment even if some agents break down or some communication channels are broken.

7. Each agent acts on its own behalf based on its local view of the world which is updated through percepts and is thus autonomous and reactive. This is implemented as an agent-control-loop in which the agents decide which actions to execute based on their current view of the world. When a repairer and a disabled agent moves towards each other the repairer decides and announces who should take the last step so they won't miss each other. This proactiveness is implemented by considering the current energy and the paths of the agents.

Software Architecture

1. We do not use any multi-agent programming language. We implement the multi-agent system using just the programming language Python.

   We choose Python as our programming language, as we think it has some advantages over for example Java, mainly in development speed/programmer effectiveness. Some of the reasons being that Python in contrast to Java:
   - is dynamically typed
   - is concise
   - is compact
   - supports multiple programming paradigms (object-oriented, imperative, functional)
   - is popular for scripting
   - does not need to be compiled before execution
2. We use Python 3.0 on Linux and Mac OS X as the development platforms and GEdit, Eclipse and TextMate as code editors/IDEs.
3. As the runtime platform for the competition we plan to use a suitable Linux system with Python 3.0.
4. Our implementation has mainly relied on custom best-first searches and a distributed auction-based agreement algorithm and a custom pathfinding algorithm tweaked for this domain.

References

1. Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. Programming Multi-Agent Systems in AgentSpeak Using *Jason*. John Wiley & Sons, 2007.
2. Niklas Skamriis Boss, Andreas Schmidt Jensen, and Jørgen Villadsen. Building Multi-Agent Systems Using *Jason*. Annals of Mathematics and Artificial Intelligence, 59: 373-388, Springer 2010.
3. Steen Vester, Niklas Skamriis Boss, Andreas Schmidt Jensen, and Jørgen Villadsen. Improving Multi-Agent Systems Using *Jason*. Annals of Mathematics and Artificial Intelligence, 61: 297-307, Springer 2011.

4. Mikko Berggren Ettienne, Steen Vester, and Jørgen Villadsen. Implementing a Multi-Agent System in Python with an Auction-Based Agreement Approach. In Louise A. Dennis, Olivier Boissier, and Rafael H. Bordini (Eds.): ProMAS 2011, LNCS 7217, 185-196, Springer 2012.

Acknowledgements

Thanks to Per Friis for IT support.

More information about the Python-DTU team is available here:

`http://www.imm.dtu.dk/~jv/MAS`

Appendix

The aim of the annual agent contest is to stimulate research in the area of multi-agent systems, to identify key problems and to collect suitable benchmarks.

The 2012 contest was organized by Tristan Behrens, Jürgen Dix, Michael Köster and Federico Schlesinger at the Clausthal University of Technology, Germany. The scenario and schedule were announced 20 February 2012 and the tournament took place 10-12 September 2012.

The 2012 winner was the Jason-UFSC team led by Jomi Fred Hübner, Federal University of Santa Catarina, Brazil. Like in 2011 we came in second. Both teams won all matches against the 5 other teams but we lost 1-2 against the winner.

The 5 other teams came from Brazil, China, Germany, Iran and USA. In addition 2 teams from Germany and Ireland did not make it in the qualification phase where the stability of the teams had to be proved.

Further details are available here: `http://multiagentcontest.org`