A Benchmark for Cooperative Learning Agents

Jason M. Black, Dean F. Hougen

Robotics, Evolution, Adaptation, and Learning Laboratory (REAL Lab)
School of Computer Science
University of Oklahoma
Norman, OK 73019
donblas@donblas.org, hougen@ou.edu

Introduction

Cooperative multi-agent systems are of current interest due to their relevance to both robotics and networking. Researchers often create machine learning environments to explore these domains; however, the lack of reuse of previous environments prevents comparisons between the works of research groups. Further, while a growing understanding of the relationships between problem domains and machine learning techniques has emerged over time, there have been few attempts to systematically measure the performance of machine learning techniques as domain characteristics vary.

Our primary goal is to create TASER, the Team Agent Simulator for Efficient Research—a non-trivial simulation system aimed at the efficient comparison of machine learning algorithms given a wide variety of conditions, concentrating on cooperative tasks. Our secondary goal is the adoption of TASER by other researchers.

Background

Available intelligent agent software ranges from extensive architectures such as SOAR (Laird et al. 1987) and development libraries such as MASON (Luke et al. 2005) to the many small environments used for individual experiments. While SOAR and MASON provide substantial resources to users, both tools have equally substantial learning curves. Also, neither tool provides a ready-made simulation environment to the user, leaving this as a task for the adopting researcher. On the other end of the spectrum, small-scale multi-agent machine learning simulators are designed for limited use and are not useful as flexible benchmarks.

Of all of the multi-agent simulators available, perhaps the most famous, and most widely used, is the RoboCup Soccer Simulator. More specifically, the subproblem of keepaway soccer has become popular in the testing of machine learning. Stone et al. (2006) discuss recent efforts at building this subproblem into a domain useful for comparing and benchmarking machine learning techniques. While this is a great move forward in the development of simulators geared towards rapid prototyping and machine learning algorithm comparison on a large scale, there are three aspects of cooperation and learning on which we wish to focus. These are local perspective, robotic realism, and heterogeneity. Before explaining these three foci in greater detail, we review our domain of application.

Pursuit Domain

A well known multi-agent domain is the pursuit domain, also known as the predator/prey problem (Stone and Veloso 2003). The original problem, introduced by Benda et al. (1986), takes place on a toroidal grid world containing five agents: four predators and one prey. The goal is for the predators to surround the prey within a given time limit. This domain is appealing because it is simple in design, non-trivial to solve in an implementation without global information, and focuses on cooperation as the means by which the problem is solved.

The pursuit problem can be broken down into seven key design considerations which determine how a given implementation functions. *World Attributes* are its size and shape, and whether its edges are solid or toroidal. Square grid worlds with toroidal edges were used by Benda and in most other instances. *Agent Attributes* are characteristics such as movement, speed, or stamina. *Agent Actions* are the moves that agents can make, traditionally restricted to moving in the four cardinal directions. *Agent Sensors* and *Agent Communication* vary between implementations. *Agent Learning* is non-existent if a static policy is used, but agents may also use machine learning methods such as reinforcement learning or genetic algorithms. Finally, the *Win Conditions* are the predators' goal(s). In most variations, including the original, the win condition occurs when the predators occupy the four squares adjacent to the prey.

Using the pursuit domain as the basis for a reusable machine learning simulator is not a new concept, as the Pursuit Domain Package, described by Kok and Vlassia (2003), is such a software package used as a teaching aid at the University of Amsterdam. While this package did not contain all of the features we were looking for, it directly inspired the creation of TASER.

TASER Implementation

In implementing TASER, we emphasized the following three aspects of cooperation:

1. A *local perspective*, not global information, should always be used. It is unrealistic to assume perfect information in the real world for teams of robots, or even in the software world where miscommunication and failures can occur. Agents, like humans, naturally observe their surroundings from their own point of view.
2. In the same spirit as local perspective, we also want to encourage realistic sensors and actions in our agents. We call this *robotic realism*.
3. Not all agents may be equal, and it may be possible for a team of agents with different abilities to solve a problem better than a group of homogeneous agents. This is why we emphasize the ability to allow *heterogeneity* among a team of agents in all aspects of their design.

These aspects are well known in the robotics simulation community, but have not been guiding design considerations in previous pursuit domain implementations. We also implemented the seven components of the pursuit domain as flexibly as possible in order to promote the simulator's reusability. Thorough exploration of this flexibility, and comparison with related testbeds, remains as future work.

TASER Design

Our goal with TASER is to create a simulation environment that is as flexible without significantly increasing complexity for the user. We designed TASER so that the core simulation mechanics remain unchanged and all control is through the use of parameter scripts and plug-in modules. As can be seen in Figure 1, TASER is a self-contained simulation environment that accepts parameter scripts as input and which connects to modules for additional functionality. Parameter scripts define how the environment is to act for each experiment, and the number of times to run each experiment. Experiments are queued and executed in turn. At any point a single grid-world is active and currently interacting with its agents.

From Figure 1 we can see that there are three kinds of modules. The *Events* module contains user-defined code that determines if a win or partial win condition has been met. This module is associated with a grid-world. *Action* modules define available actions and how to choose those actions, while *Learning* modules determine how input to the agent from the environment is handled. Every agent is connected to an *Action* module and a *Learning* module, which may be shared with other agents. By doing this we are able to make agents as heterogeneous as desired.

Finally, using the parameter script, predefined logging options allow real-time storage of simulation details from TASER. For every instance of an experiment an output file is created upon completion which contains relevant information for gauging the success of each predator and the team as a whole in completing their goal(s).

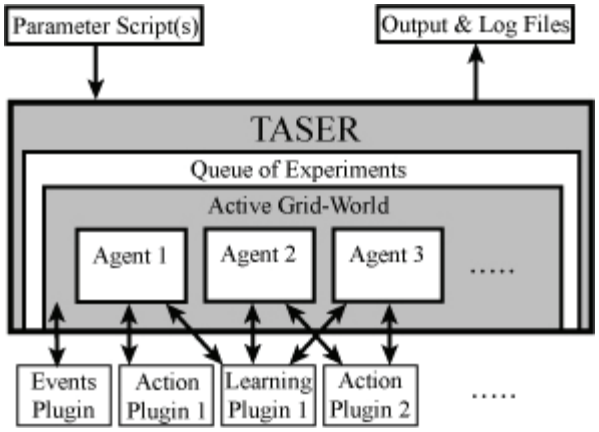

Figure 1: TASER's architecture.

Future Work

We will complete the implementation of TASER and document the environment. Benchmarking experiments and an initial algorithm comparison using reinforcement learning, genetic algorithms, and variations thereof will follow. TASER will then be released as open source software with full documentation and examples of its use for members of the community. Finally, we plan on examining formal methods of benchmark analysis to further refine the benefits that TASER can offer to the research community.

References

Benda, M.; Jagannathan, V.; and Dodhiawala, R. 1986. On optimal cooperation of knowledge sources – an empirical investigation, Technical Report, BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, Washington.

Kok, J. R., and Vlassia, N. 2003. The Pursuit Domain Package, Technical Report, IAS-UVA-03-03, Informatics Institute, University of Amsterdam.

Laird, J. E.; Newell, A.; and Rosenbloom, P. S. 1987. SOAR: An architecture for general intelligence. *Artificial Intelligence* 33(1):1-64.

Luke, S.; Cioffi-Revilla, C.; Panait, L.; Sullivan, K.; and Balan, G. 2005. MASON: A Multi-Agent Simulation Environment. *Simulation* 81(7):517-527.

Stone, P.; Kuhlmann, G.; Taylor, M. E.; and Liu, Y. 2006. Keepaway Soccer: From Machine Learning Testbed to Benchmark. In *RoboCup-2005: Robot Soccer World Cup IX*. Springer Verlag, Berlin.

Stone, P., and Veloso, M. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robotics* 8(3):345-383.