Ma, K., Wilson, M., & Aha, D.W. (2016). *Integrating reinforcement learning algorithms with underwater vehicles* (Technical Note AIC-16-221). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research on Artificial Intelligence.

Integrating Reinforcement Learning Algorithms with Underwater Vehicles

Kevin Ma, Mark Wilson, and David W. Aha

[1]Navy Center for Applied Research in Artificial Intelligence (NCARAI);
Naval Research Laboratory (Code 5514); Washington, DC
kevinma.sd@berkeley.edu | {*first.last*}@nrl.navy.mil

29 July 2016

1. Introduction

There is a need for underwater vehicles to be able to find sources of material (i.e., plumes) in aquatic situations. As part of the Integrated Microfluidic Sensor-AUV Platforms for ISR and Autonomy project for NRL's Intelligent Decision Aids Group, we introduce WandaLearning, a Java program that allows Wanda, a UUV, to make autonomous decisions based on its environment sensors to find the source of a plume in a bounded area. It utilizes MOOS-IvP, an autonomy framework for underwater vehicles, and BURLAP, the Brown-UMBC Reinforcement Learning and Planning code library.

RL focuses on finding some mapping or function of actions to rewards that allows the agent to succeed at a user-defined task. It primarily does so by directly interacting in its environment to determine the benefit or harm of performing a particular action in a state. It utilizes this information to update its mapping, which balances between new information and past information. Unlike supervised learning, RL does not need training data to indicate what is right or wrong; all the information needed can be collected through exploration in the environment, whether by random choice or predicting by using current approximations.

MOOS-IvP is composed of two sets of C++ modules, combined to provide greater assistance to the UUV community. Developed by Paul Newman at MIT (now at Oxford), MOOS was created as middleware, allowing an interface for programs to communicate to robots. It utilizes a central database to handle communication among processes, requiring programs to subscribe to and post messages. Created by Mike Benjamin at the Naval Undersea Warfare Center (and now MIT), IvP Helm is a module that utilizes interval programming to reconcile environment data and autonomy demands, finding the best action to take at the current state. MOOS-IvP also refers to many other assisting programs bundled with the two core modules. uSimMarine is a module that simulates the actions of a vehicle. pMarineViewer is a module that can display the trajectory and history of uSimMarine or other programs using logs. NCARAI has created an additional module, pPlumeSimulator, for the purpose of imitating plumes in a designated search area.

The BURLAP code library is a Java framework for the development of RL algorithms and environments. It has been utilized for its flexibility in defining environments, though we have chosen to modify its source code for serialization purposes.

2. Method

Given our decision to not discretize our environment so as to better represent reality, we employed continuous-domain RL algorithms. We chose to integrate the two applicable methods built-in to the

BURLAP code library: Least-Squares Policy Iteration (LSPI) and Gradient-Descent Sarsa(λ) (i.e., Sarsa). Both can be modified such as by changing discount factors or Bases specificity, though this can impact accuracy, memory usage, and time required to process. Based on a small sampling of tests, Sarsa so far appears to be more stable and accurate, though it usually requires much greater amounts of time due to being an on-policy method.

To imitate reality, WandaLearning can add diversity to the environment. An argument to the program allows users to set whether the starting position of the agent and the plume's characteristics (i.e. direction and position) should be static. To prevent situations where the agent could not find a plume (because it did not wander into it in the large search-area), plume randomization creates a relationship between direction and location by splitting the navigable area into quadrants and determining "good" directions to increase chance of contact.

Currently, WandaLearning has three main modes. One can create policies from scratch using either LSPI or Sarsa. The environment can be made diverse, as stated earlier, and one can enter maximum steps and sampling amount as arguments. The second mode is continuing from previous efforts. This is utilized if previous attempts crashed, usually from communication errors between WandaLearning and MOOS-IvP, or if previous attempts were not accurate enough. The last mode is running policies, a way to test policies post-completion. This was not integrated into the other two modes as the long running time may cause required log files to become quite large.

3. Discussion

Throughout the development process, various issues arose that led to fundamental changes. One of these changes regarded our state representation. In earlier implementations of WandaLearning**,** there were many "Objects" the state tracked. An oddity in the state representation was that the number of objects to track varied. This was to done to reflect discovering new information as the agent wandered in its environment. However, BURLAP was not designed for working with a variable amount of information, which led to issues with computation time, memory usage, and incompatibilities with matrix computations. Now, there is just one "Object": the agent. All the remaining attributes allow people to view how the agent acted using MOOS-IvP's uSimMarine.

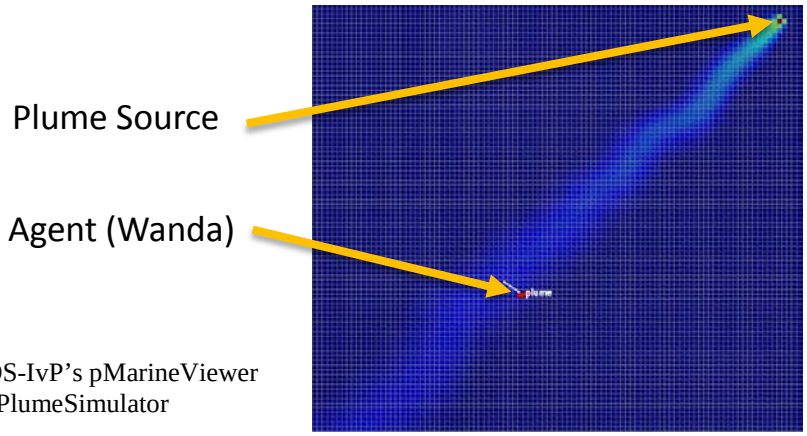
Original State	Current State

• (1) Agent
  o X Position
  o Y Position
  o Heading
  o Speed
  o Time
• (Variable) Location
  o X Position
  o Y Position
  o Concentration Value
	• (1) Agent
	  o X Position [-1, 100]
	  o Y Position [-1, 100]
	  o Heading [0, 360]
	  o Concentration [0, 10000]
	  o Current Direction [0, 360]

Due to the matrix library used by BURLAP being single-threaded, WandaLearning's time bottleneck is matrix computation. To provide redundancy, WandaLearning should have multiple iterations running. In doing so, one would also have to create multiple MOOSDB environments, due to

Ma, K., Wilson, M., & Aha, D.W. (2016). *Integrating reinforcement learning algorithms with underwater vehicles* (Technical Note AIC-16-221). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research on Artificial Intelligence.

MOOS' reliance on a centralized database. To help maintain multiple copies of WandaLearning, a script was created to replace commonly modified files. Also as a result of this specific module, memory usage can be quite large. Combined with the number of threads started by Java and MOOS-IvP, a desktop computer may run out of memory; this can be somewhat mitigated by reducing the number of samples.



Visual using MOOS-IvP's pMarineViewer with NCARAI's pPlumeSimulator

4. Next Steps and Conclusion

WandaLearning so far only has been tested on uSimMarine and pPlumeSimulator, not in an actual autonomous vehicle. Inherently, this can only be done once the simulations get more realistic and the program matured to increase stability and obtain better results. In addition, more algorithms relating to continuous domains can be implemented later, and alternatives to the current matrix library may make WandaLearning run faster and more efficiently than its current state.

Currently, WandaLearning's results are mixed. It works very well when it is very near the source of the plume, approximately 15 steps away. If too far away, it usually acts nearly randomly or sticks to the edges of the search area. This behavior is primarily due to how it learns and its small action set. The agent can only move straight, turn left, or turn right. It starts out moving randomly, though only leaving its current position one-third of the time. Added to the fact that for most of the search area plume-substance concentration is negligible, it would be hard for the agent to find the plume without extensive searching. To mitigate the issue, we are experimenting with the reward function to encourage the agent to stay away from the edges and increase exploration distance.

Acknowledgements

Kevin Ma is an undergraduate student who is attending the University of California, Berkeley. He thanks the Office of Naval Research (ONR) and the American Society for Engineering Education (ASEE) for running the Naval Research Enterprise Internship Program (NREIP) that funded his visit to NRL during the summer of 2016. Kevin also thanks Mark Wilson and David Aha for providing much assistance and direction to this project, which would never have progressed so far without them.