Automatic Design of Behavioral Arbitrators for Khepera IV Robots: a Comparison between Deep Reinforcement Learning and Particle Swarm Optimization for the Training of Artificial Neural Networks

Lucas Wälti

Professor : Alcherio Martinoli
Assistant(s) : Cyrill Baumann

1. Introduction

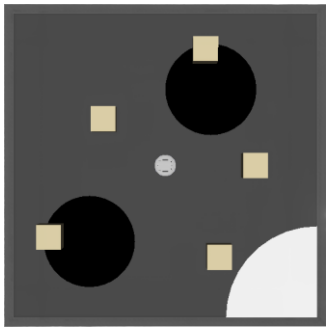
Basic behaviors for robots can be relatively easily implemented but in order to solve complex tasks, an adequate combination of behaviors is usually required. This however may be complex and tedious to program and it is therefore interesting to evaluate ways of learning a policy able to leverage a set of predefined behaviors. In this project, Artificial Neural Networks (ANN) were used as arbitrators for the robot. Three algorithms were evaluated to train the ANNs:

- Deep Q-Learning (DQN)
- Vanilla Policy Gradient (VPG)
- Particle Swarm Optimization (PSO)

Both DQN and VPG are Reinforcement Learning (RL) based methods, while PSO is a heuristic, population based algorithm.

2. Experiment

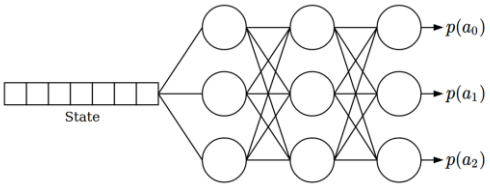
In order to evaluate the performances of the selected algorithms, a simple foraging task was adapted from a previous project and used to this end. The robot shall find some resource corresponding to the black areas and then reach the white area, as illustrated in the figure below:



Arena for the foraging task.

3. Training

Each algorithm was trained individually on a single robot in the arena. While the RL based methods require a large exploration (such as varying the robot's spawn location or randomly removing obstacles), it was observed that PSO needs as little difference as possible between the particles' evaluations, which helps make the fitness computations more reliable.



Fully connected ANN architecture used by VPG and PSO to implement the policy. The output is the probability of each action.

4. Results

Deriving a policy from an (Action-)Value Function can be hard. DQN typically did not yield satisfactory results since it learns such a function from which the policy is then derived. On the other hand, VPG, which directly learns the policy, was able to reliably learn the task in a reasonable amount of iterations. PSO was also able to learn a policy, as long as the fitness computation was reliable enough. In a sense, it completely surpassed VPG as a particle was able to solve the task after only one or two iterations. However, it should be noted that PSO initializes numerous particles. Since the problem here is quite small, it is comparable to a brute force approach where a good candidate is found by randomly generating enough particles. Furthermore, PSO does not make it possible to fine tune the weights of the network to make the action selection more robust. While picking an action using an "argmax" approach is enough, sampling an action from the network's outputted probabilities when trained with PSO is not reliable, as opposed to sampling the output of a network trained with VPG.

5. Conclusion

Gradient based RL approaches should provide the most robust policies when training ANNs. However PSO may be used to quickly identify a suitable candidate which can be then refined by a RL algorithm.

Miscellaneous

This project was entirely implemented in C++ in order to facilitate a later integration on the real robots. All ANN related operations were implemented with the PyTorch framework and its C++ API. Noise resistant PSO had to be implemented from scratch.