

**CENG 580—Multiagent Systems  
Homework Assignment III**

**Due to** May 27 , 2007

U.Erdogdu

**Rock Sample Problem**

In this homework you are asked to design a controller for a robot that is sent to a distant planet for sampling rocks.

The robot has an energy level. The robot is sent with a launcher, which can provide energy to the robot. However, when the robot leaves the launcher, he's on his own.

Below are the characteristics of the problem.

- The environment is  $n \times n$  grid world.  $n$  is always odd and the launcher (start and goal points for the robot) is assumed to be at the center of the environment. Also assume no obstacle or rock exists at the center of the environment.
- The environment has obstacles and their positions are unknown to the robot.
- The environment has  $k$  rocks, where  $n = 2k + 1$ . Rocks are stationary. Their positions are unknown too. Obstacles and rocks do not share same cell.
- The robot has an initial energy level of 100. When robot is out of energy, it is lost, episode stops.
- Robot can navigate at the environment with four possible actions *up*, *down*, *left* and *right*. Bumping obstacles has no effect. If robot moves out of the boundary, it is lost, episode stops. Each action has energy cost of 1.

- Robot can take rock samples when at the same cell with the rock, by *sample* action. Each sampling has energy cost of 5.
- When the robot returns to launcher, episode stops.
- The proposed reinforcement signal for the robot is structured as:
  - when robot is lost, he will receive -10.
  - when robot reaches launcher, he will receive a reward of  $(e/10) + (k \cdot 10)$ , where  $e$  is the energy level and  $k$  is the number of rocks sampled.
- You can introduce your own reinforcement signal structure too.

Design and implement the learning process of hunters using Q-learning. Your program should have training mode and acting mode. Training mode should load environment configuration ( $n$ , places of obstacles) and execute training runs for the given environment configuration and should save a policy for acting at the end. Acting mode should load a saved policy appropriate and the environment configuration, then use the policy to act at the given environment.

Training mode should only output the policy, no other information is required. Acting mode should output each step of execution in detail, in any way you find appropriate.

You have to submit your source code and a small documentation that contains *only* the following information:

- How to compile/run your program and whether your program has any dependency for compiling/running (Example: If your C++ code requires VS, you should note it here)
- How to save and load policy information.
- Your RL setting which includes
  - Your state space structure
  - Your Reinforcement Signal structure

**Case to be concentrated on:**

grid size ( $11 \times 11$ ) ( $n = 11, k = 5$ )

percentage of the cells to have obstacle (%30)