Massachusetts Institute of Technology

# 16.413: Principles of Autonomy and Decision Making

# Final Project: Gnu Robo Challenge

## Objectives — Distributed Nov 8th

The objective of the 16.413 final project is for each of you to learn to create embedded autonomous agents based on the methods that you have learned through lecture and problem sets, and to communicate your designs effectively in writing (Phase I Report due November 19th and Phase II Report due December 3rd).

The specific project challenge is to design and implement an autonomous agent (simulated robot) that will compete head to head with another robot within the GNU Robots simulator. Your robot's objective is to accumulate the highest number of points from collecting prizes, without having the robot expire due to damage or lack of energy. We will conduct our class tournament from 9am-12 noon December 1st.

More specifically, the project objectives include the following:

- To design and implement a complete agent:
  - Select and implement two or more decision making or estimation methods suitable for achieving the robot challenge.
  - Develop suitable representations and problem encodings that will allow these methods to perform reactively.
  - Design, integrate and demonstrate a closed-loop agent based on these methods and encodings.
  - Document the agent design and architecture, highlighting design options considered, and unique innovations.
  - Evaluate and report your agent's overall performance analytically and empirically.
- Explore sophisticated versions of the basic algorithms taught in class:
  - Use the web and library to evaluate and select suitable advanced decision making and estimation algorithms.
  - Implement and demonstrate these algorithms within the agent.
  - Provide a design rationale and tutorial explanation of the methods employed.

## General

- You are welcome to work together in teams, just like in the real world, or individually, as you prefer. Teams will allow you to develop a more ambitious agent. Team size, however, is limited to a maximum of two.
- On the day of the competition you will give a brief, one minute – one slide presentation of your agent.
- The results of your project are to be captured in a written document of 10 to 20 pages for Phase II.
- Separate written reports are to be done by each student.

## Project Grading

1. A – represents mastery: the ability to extend and apply the methods of the course in a way that is novel and insightful; the ability to explain and motivate in a manner that is particularly intuitive.
2. B – represents competence: the ability to design, implement, explain and evaluate a working agent based on the methods described below.
3. C – represents partial competence of the above.

## Important Deadlines

- Nov 3rd Gnu Robot simulator documentation distributed.
- Nov 8th Final project guidelines distributed (this document).
- Nov 19th Project Phase I completed and Phase I report due.
- Dec 1st Class tournament and 1 minute project presentations, 9am-12 noon. Code posted to Athena by 9am.
- Dec 3rd Phase II reports due.

Reports due by 5pm of day assigned to Brian O' Conaill, near 33-330.

## Gnu Robots Simulator

Documentation for the Gnu Robots simulator was distributed five days ago during Wednesday's class, November 3rd. The documentation is posted on the course web site, on the Lecture page, under Wed Nov 3rd. By now each of you should have loaded the Gnu Robot simulator, have played with it extensively, and are beginning to formulate your strategy of how to effectively maximize your agent's score. For those who have not done so, its urgent that you do so immediately. Please send any questions that you have regarding the simulator and game rules to 16.410-instructors@mit.edu. We will maintain a list of answers to frequently asked questions.

## Designing Your Robot

Your objective is to write a scheme agent that guides your robot to maximize its objective. Typically, this will involve the agent performing some level of planning in order to acquire its reward quickly. In addition, the agent may need to perform some level of estimation, in order to model its world and locate itself within this world.

Each agent is required to leverage at **least two methods** (or their extensions) taught in class. Other than this requirement of leveraging at least two methods, there are no additional rules on how you implement your agent. We will provide suggestions and examples that will set you on the path to constructing a competent design. The most successful implementations, however, are likely to vary from our suggested implementations. Employing your own innovations will be *essential* for you to demonstrate your *mastery* of the course.

Some of the issues you should think about include:

- How should your agent be structured? Do you have a single, monolithic planner, or a hierarchy of planners operating at different levels of abstraction (e.g., MDP at one level, path planner at the next)?
- How should you cope with computational complexity? How expressive of a problem representation do you need? What encodings should you use? Do you use an incomplete algorithm?
- Do you precompute a policy for all possible situations, or do you interleave planning and execution?
- How should you cope with hidden and uncertain information? Do you represent uncertainty using a CSP framework, in terms of consistent states of the world, or do you use a probabilistic representation and Bayesian inference?
- How should you cope with opponents? We have not discussed game theory in this class, but many of the algorithms we have discussed in class can be easily extended to competitive game scenarios. Games are covered in Chapter 6 of AIMA (Adversarial Search).

In order to ensure that you stay on schedule, we have broken the project down into two phases, described below. Note that each Phase involves an implementation and a writing assignment.

## Phase I – Single Robot In A Known Map

**Reports due November 19th**

In Phase I, you must implement an agent for the robot that operates under the following conditions:

1. There is only one robot in the world.
2. You can query whether or not map positions are walls or not (robot-is-wall X Y).
3. You can query where the prizes are (robot-get-prizes).

One possible implementation strategy is the following:

1. Use robot-is-wall to determine the location of all the walls.
2. Query robot-get-prizes to determine the location of the prizes.
3. Use A* search to compute the shortest path between every pair of prizes. The result is a roadmap.
4. Encode the roadmap, with distances and rewards as a Markov decision process.
5. Use value or policy iteration to construct a policy, which specifies the order to visit the prizes to maximize reward.
6. Execute the high-level policy to decide which reward to go after next, while using the shortest-path plans computed by A*, to move towards each reward.
7. Replan whenever a baddie is discovered.

You can use the above strategy to demonstrate your *competence* of the course material; however, a demonstration of course *mastery* will require *substantial innovation* beyond this recipe.

On November 19th each student will turn in a report of no more than 3 ½ pages:

1. A description of your implementation for Phase I (2 pages).
2. The output of your controller on the file maze.map (1/2 page).
3. A description of your planned implementation for Phase II (1 page).

For item 2, we do not want the complete sequence of events, just the output consisting of your score and final energy level. The output of your controller should be no more than a few lines.

## Phase II – Two Robots Competing In An Unknown Map

**Competition and Implementation Due December 1st**
**Report Due December 3rd**

In Phase II, you must implement an agent under the following conditions:

1. There is an opponent robot in the world, competing for rewards.
2. You can no longer query where walls of the maze are.
3. You can no longer query where the rewards are.

A simple extension to your Phase I agent would have it build local maps as it makes observations, incrementally improve the map after each action, and have it watch for opponent robots. The agent would then plan its actions as in Phase I.

However, a better strategy might involve planning to herd the opponent robot into a corner, predicting where the opponent is going and trying to beat it to food or rewards, or performing adversarial search to account for the actions of the opponent.

You will likely want to design your Phase II strategy at the beginning of Phase I, so that your Phase I efforts can be leveraged in Phase II; however, your approach may also change considerably after some experience.

From 9am to noon on December 1st, during the 16.413 and .410 class periods, we will have a head-to-head competition between students in class. We will have a preliminary seeding round, involving round-robin play. If there are more than 16 robots, then the top 16 will advance to the playoffs.

The rules of play are as follows (and are subject to revision):

- Each player must make their code publicly available via Athena by 9am, December 1st.
- There is no limit on the number of files required to implement a player's agent.

- Each player (team) will be given 1 minute and at most one slide to explain how their implementation works.
- Each game will last either 4 minutes, until the simulator prizes are gone, or until one of the players' robot expires (whichever comes first).
- The score for each player is the number of prizes, plus a bonus of 100 if the robot is still alive at the end of the game.
- If a player's agent takes longer than 30 seconds to generate any one move, the player forfeits the game.

On December 3rd each class member will turn in a 10-20 page final report describing their design, implementation and algorithms employed. This report will demonstrate each of the specific project objectives in the bulleted list at the beginning of this document. **Please review each bullet in the objective carefully, as you outline your final report**.