Introduction to Artificial Intelligence (236501)

Spring 2011

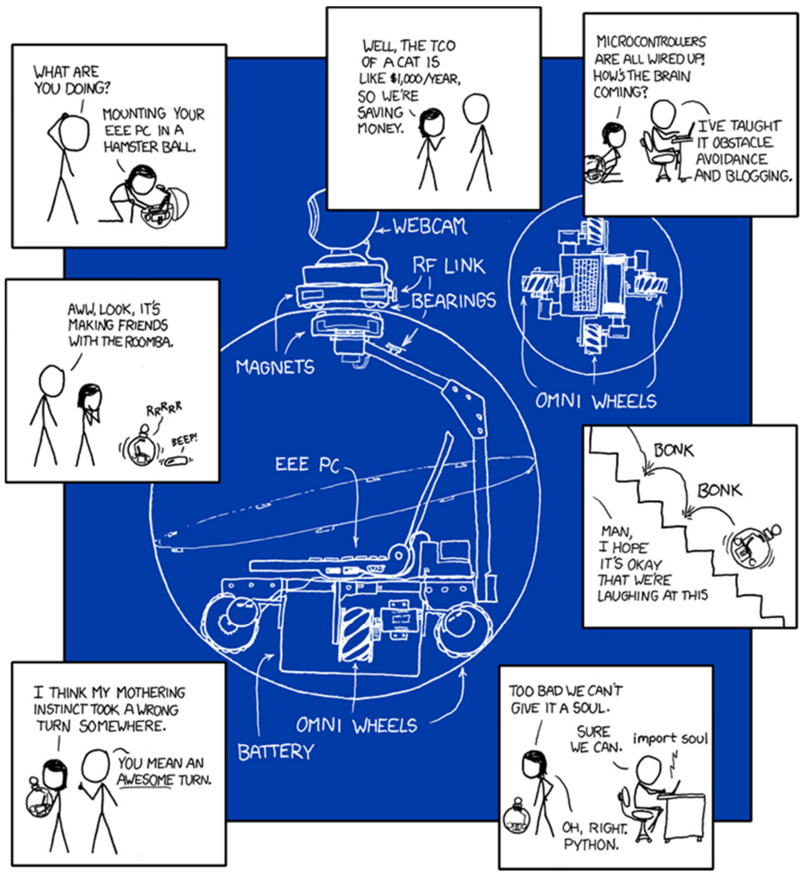
Homework Assignment 1: The Multiple-Robot Problem

Assignment Goals:

- To understand the benefits of representing problems as state-spaces.
- To review the pros and cons of heuristic search algorithms.
- To learn how to design, implement, and analyze experiments.

Notes:

- Due date: 26/4/2011.
- This assignment is for submission in pairs or singles. No trios will be allowed.
- **Copying of any kind will not be tolerated, and will result in expulsion from the course.**
- Feel free to ask questions via email: omerlevy@cs.technion.ac.il


WHAT ARE YOU DOING?
MOUNTING YOUR EEE PC IN A HAMSTER BALL.
WELL, THE TCO OF A CAT IS LIKE $1,000/YEAR, SO WE'RE SAVING MONEY.
MICROCONTROLLERS ARE ALL WIRED UP! HOW'S THE BRAIN COMING?
I'VE TAUGHT IT OBSTACLE AVOIDANCE AND BLOGGING.
WEBCAM
RF LINK
BEARINGS
MAGNETS
OMNI WHEELS
AWW, LOOK, IT'S MAKING FRIENDS WITH THE ROOMBA.
RRRRR
BEEP!
EEE PC
BONK
BONK
MAN, I HOPE IT'S OKAY THAT WE'RE LAUGHING AT THIS
I THINK MY MOTHERING INSTINCT TOOK A WRONG TURN SOMEWHERE.
YOU MEAN AN AWESOME TURN.
OMNI WHEELS
BATTERY
TOO BAD WE CAN'T GIVE IT A SOUL.
SURE WE CAN.
import soul
OH, RIGHT. PYTHON.

<u>Problem Setting:</u>

The multiple-robot problem is modeled after the popular iRobot® Roomba®. Here is a video demonstration of the robot in action: http://www.youtube.com/watch?v=ewdbilSWjaM

We will introduce a few additional features to our dust-cleaning robot. First, the robot perceives the room as a two-dimensional grid. It also has a complete map of the room, including the locations of dirt piles and obstacles. This problem was presented in the tutorials, and carries much resemblance to the traveling-salesman problem. In order to make things more interesting, we added multiple robots, which operate in parallel.

Below is a formal representation of the problem:

- Constants:
  - W, H – The room's width (x-dimension) and height (y-dimension).
  - R – The number of robots.
- S – Each state contains the locations of each robot, the dirt piles, and the obstacles.
- $O = \{up, right, down, left, nop\}^R$ – Operators that cause robots to run into other robots, obstacles, or room borders are not applicable in those states. In other words, each operator is an array of R directions $(d_0, d_1, \dots, d_{R-1})$ with respect to each robot.
- $G = \{s \in S \mid s\ has\ no\ dirt\}$
- $\forall o \in O:\ cost(o) = 1$ – Each operation costs a single time unit.

As with any Polish household, we need to clean our room before the cleaner arrives. In other words, we need to come up with the shortest plan in terms of time (not in terms of energy). Theoretically, we will measure the quality of each solution against the optimal solution's cost:

$$quality = \frac{cost(optimal)}{cost(plan)}$$

Not only do we want to save time in cleaning, we are also limited in computation time. Therefore, we must come up with an **anytime algorithm** for this problem; one that generates better solutions as more computation time is given.

Assignment Instructions:

1. Choose **two informed algorithms**, and modify each one to have **anytime** behavior. Explain why these algorithms are suitable for the multiple-robot problem.
2. Think of **two non-trivial heuristics** for the multiple-robot problem, which can be used in conjunction with the algorithms you chose. Explain why they are suitable.
3. Create **at least three sets of problem instances** with varying difficulty, and explain why each set of problems is more/less difficult than the other. There is no need to exaggerate in difficulty; choose instances that can be solved in reasonable time.
4. Compare different combinations of algorithms and heuristics across computation time and across difficulties, and present your results in a graphical manner. Measure computation time using two metrics: a parameter specific to the algorithm, and the actual computation time (use `time.clock()`).
5. What conclusions can you draw from these results? Do your findings reflect inherent properties of the algorithms or those of the multiple-robot problem?

Provided Code:

The following code is provided for your convenience:

- problem.py – An interface for problem spaces.
- problem_agent.py – An interface for problem solving agents.
- multi_robot_problem.py – An implementation of the multiple-robot problem.
- search package – Contains a toolbox of basic search algorithms.

Competition:

In accordance with the conclusions from your research report, you will implement a problem-solving agent for the multiple-robot problem. This agent will compete against agents designed by your peers in a variety of problems and time limits. Each track (problem instance and time limit combination) will reward the agent with a score proportional to the quality of its solution. The overall score is the weighted sum of all track scores. Winners of the competition and specific tracks will be awarded bonus points to their final grade.

<u>Submission Instructions:</u>

Dry: Research Report

- This is the most important part; it will determine your grade.
- Limit your report to **eight pages**.
- The report should contain the following sections:
    - Solution Proposal – Where you will present your algorithms and heuristics.
    - Experiment Design – Where you will present the difficulty types, and explain how you conducted your experiments **in a manner that is reproducible**.
    - Results – Where you will present the experiments' results. Use graphs or tables to present your results, and add textual explanations where necessary.
    - Conclusions – Where you will analyze your results and try to draw deep and profound conclusions from them.
- Refrain from writing code in the report.
- Submit the printed report to the course's inbox next to the coffee cart.
- Submit an electronic version of your report and all **experimentation code** to the "Research Report" assignment in the course website.

Wet: Competition

- You will write your agent in the Python programming language. The agent must inherit from the `ProblemAgent` class in the provided code.
- No preprocessing or preprocessed data is allowed. No use of network resources is allowed. Any attempt of sabotage or cheating will be dealt with harshly.
- You will submit (electronically) a zip file containing:
    - submissions.txt – Name, ID, email (comma delimited, each student in a separate row)
    - agent.py – Contains the implemented agent. Must contain exactly one class that inherits from `ProblemAgent`.
    - Other Python files, in the same directory (no sub-directories).
- Make sure that your code assumes a flat hierarchy; i.e. that the code provided by the staff is in the same directory as yours.
- The submission should **not** contain any of the provided code.
- Submit the **competition code** the "Competition" assignment in the course website.

Good luck!