

## **Problem Set 4**

### **Due December 12, 2006, 11:59pm PST**

### **Capture Those Flags, Gofer!**

Your goal is to design an electronic “gofer” (an assistant who performs chores by running around fetching things). Your boss is a high-powered Hollywood executive who doesn’t have time to run around fetching coffee, paperwork, and the like. Therefore, he has selected you to come up with a robot that can do this for him. Unfortunately, you aren’t the only one so “chosen.” Your agent must compete against other agents for the right to be named as this executive’s official robotic assistant.

---

#### RULES

The service area is divided into a rectangular grid. Each cell in the grid is either empty or contains an impenetrable wall. Empty cells may contain robots (and wall cells cannot). Each robot occupies one grid cell, has an orientation (one of the four cardinal directions), and is controlled by one player.

Each robot is informed of its starting location and direction as well as those of its opponent. The layout of the rectangular grid is also known.

Unfortunately (or perhaps fortunately, depending on how you view things), the robot’s sensors are limited. While they know precisely the location of all current requests (flags), they can only sense the four nearest walls, and hear the sound of the opponent robot (robots are not quiet). And these senses are limited too. There is a fixed probability of misreading the surrounding walls. The opponent robot does not count as a wall.

Furthermore, due to echos, robots may misinterpret their auditory sensors. The noise heard is directly related to the distance between the robot and its opponent. With a 50% chance, the robot predicts the correct distance. However, with a 25% chance, the robot estimates a distance that is 1 square too close. And, with a 25% chance, the robot estimates a distance that is 1 square too far. Distance is the length of shortest path through the floor plan that does not go through a wall.

Players alternate turns. Each turn a player’s robot observes its immediate surroundings (via the two sensors described above). Each turn the player must decide which action to take. There are four actions available. If the player does not decide within 2 seconds, its robot takes no action and the turn passes to the next player. The four actions are nothing, forward, turn left, and turn right.

The first causes the robot to remain in the same square with the same orientation.

The second advances the robot one square in its current direction. If there is a wall or another robot in the way, the robot does not move. There is also a chance (known to the robot) that this move fails, leaving the robot in its original location. The last two actions turn the robot  $90^\circ$  to the left or right (respectively). There is a chance (known to the robots, but different than above) that this rotation fails, leaving the robot's orientation unchanged.

At the beginning of the game, a set of service locations (known as flags) are specified by the boss. The first agent to arrive at any of these locations is awarded a point for having performed the necessary fetching service. Throughout the game, the boss may, from time-to-time, remember new tasks and add them to the list of flags. An agent only receives credit for a task if it is the first one to the location after the flag is revealed by the boss. The locations of all flags and whether or not they have been collected (and if so, by whom) is known to both agents.

When the competition starts, each player has 10 seconds to do any necessary initializations based on the preliminary map. The competition runs for a fixed (and known) number of rounds. A round consists of (in order): the placing of new flags, the turn for the first agent, and the turn for the second agent. At the end, the agent with the most number of claimed flags is the winner.

---

## Competition Rules

All entries must be submitted by 11:59pm on December 12, 2006. Students may only submit one entry.

All games will be run in pairs. Each game of a pair will pit the same two opponents against each other on the same game board with the same parameters, but with different random number seeds. The players swap places (who is player 1 and who is player 2) between games of a pair.

The entries will first be all played against each other in a "round robin" fashion: each competitor playing a pair of games against each other competitor. The results of this pool play will seed the competitors in a single elimination tournament. The top teams will get "byes" (automatic victories) in the first round of the tournament if the number of entries is not a power of two. Ties will be broken by continuing to play pairs of games until the number of games won by each opponent is no longer even.

The pool play games will all be on the same board. It will not be revealed before the submission deadline. The knock-out tournament will also all be run on the same board. It too will not be revealed until after the submission deadline.

The initial stages of the tournament will be run on December 13th and 14th. The results will be posted online in real time as the tournament proceeds. The semi-finals and finals (all games after the field has been paired down to 4 agents) will be displayed live in

EBU II, room 205 at 2:30pm on Friday, December 15th. All are heavily encouraged to attend (it should take about an hour... maybe less if the agents are speedy) and watch the competition. Food, drink, and the like will be provided.

At the beginning of each class period until the competition, the instructor will ask for any rule questions (concerning either the game or the tournament). All clarifications will be given at the beginning of class to all present.

### Assignment

The assignment is to write an agent for the aforementioned competition. The address <http://www.cs.ucr.edu/~cshelton/courses/cs205-f2006/> has the code which provides the game server (the program that runs the game) and an example agent (a player). Your job is to write a program that communicates with the server and plays the game. All of the communication code is already written. The code contains a README detailing how to use it.

You are welcome (and highly encouraged) to use code that you wrote for previous assignments. Additionally, any of the published solution code is okay for you to use. However, other than the solutions I wrote, all of the code must be your own.

This game incorporates many elements from the class thus far. I do not know of a single best solution. However, I am looking for four things:

1. **Your solution should be principled.** It should not be a collection of random rules. Rather it should be (more-or-less) an elegant solution to the problem. We did not solve the museum problem by slapping together a bunch of special-case rules, but rather reduced it to an HMM problem and solved an exact problem. While there is no exact solution to this game, that doesn't mean that you can't approach the problem with the same kind of general formalism. Of course, there will be special cases and hacks that might enter into the code. They should be few and far between.
2. **Your solution should perform well.** While your grade isn't based on how well you do in the competition, it is based on how well your agent performs in an absolute sense. Two sample agents are given. You should *certainly* be able to easily best them. Furthermore you should be able to beat slightly more principled opponents.
3. **Your solution should use AI methods.** The code you write should solve the problem using AI techniques. We have discussed many in class. You have even authored a few. There are still more listed in the book. And there are even more listed in AI papers (although you are certainly not required to use them). All of them are useful for this problem in one way or another. Feel free to use any AI method.
4. **Your write-up should be clear.** I want you to turn in a write-up with the code (as a plain text file). I'm imagining something like four to eight paragraphs. You

should explain how you approached the problem, which algorithms or methods you used (and why), and any modeling assumptions you made. I would also like to see some analysis of its behavior (why it works or doesn't and how you would improve it).

As such, your grade will depend on how well you meet those criteria. Here is a rough grading rubric.

- **A:** An "A" assignment would compile smoothly and run without errors. It would do one of the following.
  - Incorporate at least three algorithms from past assignments, each in a useful way critical to the success of the agent.
  - Employ at least two algorithms from past assignments, but with *major* modifications to deal with the complexities of the game. (By major, I mean that the basic underlying assumptions of the algorithm were changed and the algorithm had to be re-derived based on the new assumptions — this one is tricky.)
  - Employ at least one algorithm from past assignments and one other algorithm (one discussed in class or the book, but which did not make it to a problem set) in an appropriate and utile way.
  - Employ a new algorithm not covered in the book or class (but never-the-less clearly in AI) correctly and effectively.
- **B:** A "B" assignment would consist of one of the following.
  - Any of the "A" criteria, but with moderate (but not major) compiling or run-time difficulties.
  - An agent that used only two algorithms from past assignments, but does so to *great* effect and without compile or run-time bugs.
- **C:** A "C" assignment would consist of one of the following.
  - An incomplete attempt that does not come close to compiling, but otherwise meets the criteria for "A."
  - A buggy use of a single past algorithm, or the use of a single past algorithm to little effect.

This assignment will test your ability to apply your knowledge to a new problem. Therefore, I will not tell you how to approach it. However, we have covered (or will cover) a number of topics, all which could be useful: search, game-playing, optimization, logical reasoning, planning, probabilistic inference, hidden Markov models, Markov decision processes, supervised learning, and parameter estimation.

**Good Luck!**

May the best gofer win.