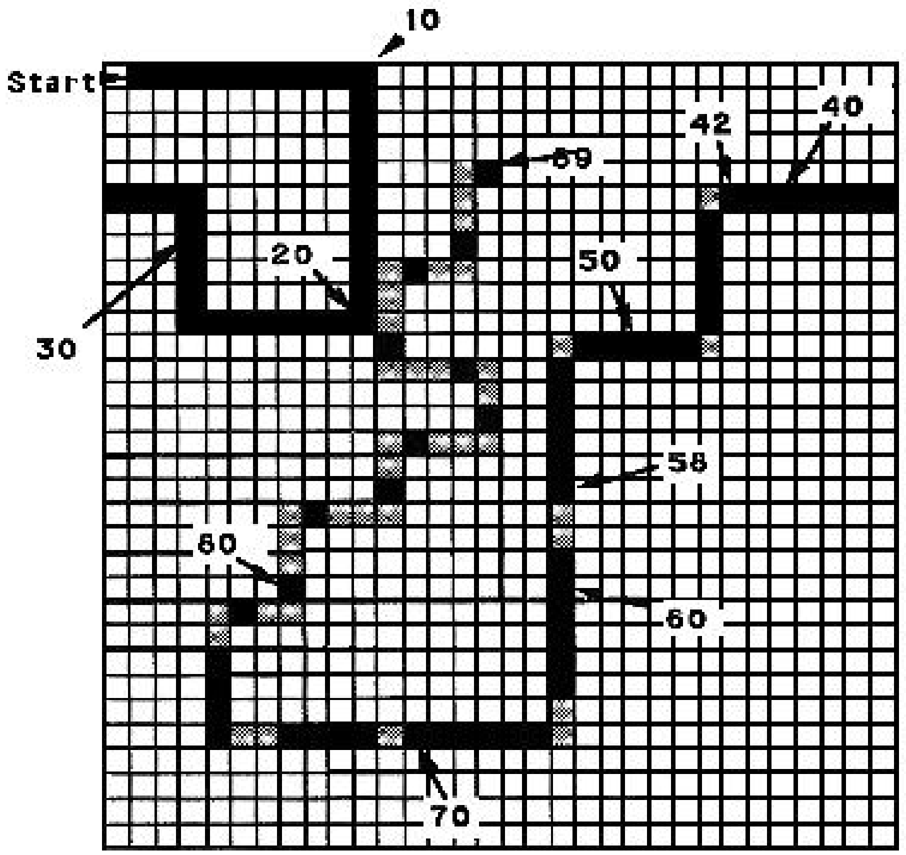
6.836 Embodied Intelligence–Research Assignment 4

Massachusetts Institute of Technology

Due: Wednesday, April 13th, 2001

Please turn in EACH problem on a SEPARATE page. Please hand in any code you wrote for the problem set (it doesn't have to be pretty).

During lecture we looked at evolutionary systems that evolved an 'ant' to prowl over a 2D grid of cells with toroidal boundary conditions (wrap-around), following trails of food. One trail that has become something of a *de facto* standard is the "John Muir" trail illustrated in the following figure:



Cells are either empty or full (the light gray cells in the figure illustrate the optimal path, but are as empty as blank ones). The ant starts in the cell marked *Start*, facing right ("east"). Thereafter it is always in a cell and facing in one of four possible directions. At the beginning of each time-step it gets one bit of sensory information: whether there is food in the cell in front of the cell it currently occupies (i.e., the cell it would move to if it moved forward). At each time-step it has one of four possible actions. It can (**F**) move forward one cell; (**R**) turn right ninety degrees without changing cells; (**L**) turn left ninety degrees without changing cells; or (**N**) do nothing. If an ant moves onto a food-cell, it consumes the food and the food disappears; when the ant leaves that cell, the cell is empty. The fitness of the ant is rated by counting how many food elements it consumes in 200 time-steps. There are 89 food cells in the John Muir trail.

For the following exercises, files containing a representation of the John Muir trail, and the alternate Santa Fe trail, can be found on the course web pages (`http://www.ai.mit.edu/courses/6.836/handouts/handouts.html`) under the info for Problem Set 3.

1. The sensory-motor coordination for the ant can be implemented using a finite state automaton (FSA). Design a 'genetic encoding' representation, specified by a fixed-length bit string, that encodes ant sensory-motor FSA transition tables. Then, by hand, try to design the best possible ant for the John Muir trail. Show the encoding and the corresponding FSA state-transition table and diagram. What fitness does your ant controller score?

Note: A blind ant with an 89 state FSA could traverse that particular trail perfectly — but would probably get very lost on another trail. Limit yourself to, at most, a 16-state FSA.

2. Does your ant use the **N** operation? Under what circumstances might the **N** operation be useful?

3. How many possible different individuals are possible in your representation?

4. How does your ant fare on the "Santa Fe" trail?

5. Generalize your representation to allow 2^n states. Express the number of bits in your representation, and the number of possible individuals, as functions of n.

6. Write an evolution program that can evolve individuals (for the John Muir trail) using your representation. Include multi-point crossover and mutation. Decide on a fixed number of states with which to run the program. (You do not have to limit yourself to 16 this time, but think twice before making it too large.) Experiment with different population sizes, and with different parameters for what proportion of the fittest individuals are retained, what proportion are used for reproduction, and what levels of mutation are used. Run your system and plot how fitness increases by generation.

7. Repeat the above experiments, but now measure fitness by testing each individual on both the John Muir and Santa Fe trails.

8. Show the best individuals your system found from each of Problems 6 and 7 (i.e. their FSA state-transition tables and diagrams),, and analyze their behavior. Does the Problem 6 ant demonstrate any particular specializations? How do both compare to your hand-coded solution from Problem 1?

9. Assume that you only know that your ant will be given a fixed size (NxN) input grid with food and a starting position in the upper left corner. Wit an unlimited number of time steps to gather food, and O(N) states, describe an ant that is optimal.

(Bonus, just for fun). Suppose you had evolved your hand-designed ant from Problem 1. Is there a series of one-bit mutations from an all-zero string to your best hand-designed machine such that the fitness never decreases at any step? Is there a series of one-bit mutations from an all-zero string to your best hand-designed machine such that the fitness always increases at every step? How many possible one-bit mutation evolutionary paths are there, from the all-zero species to your hand-designed machine?