

Nilsson's Official Homework 6 Solutions

Homework 6 Due Tuesday October 6, 2009 in class

I. Problems 4.1; 4.3; 4.4; 4.5 in Nilsson pp. 69-70, Problems 5.1, 5.2 in Nilsson

4.1 Specify fitness functions for use in evolving agents that

1. Control an elevator

2. Control stop lights on a city main street

1. There are several factors that might be considered in control of an elevator. These are average waiting time for a passenger outside the elevator before the elevator arrives, average waiting time for a passenger inside the elevator before it arrives at the passenger's desired floor, and maximal waiting times. During the early stages of evolving an elevator controller, it might be that these times are infinite because some passengers may never be serviced. Therefore, we might also include factors such as number of passengers that are never serviced. The fitness function itself can then be composed from these factors.

2. Here the factors might be average and maximal waiting times at stoplights for the through traffic and average and maximal waiting times for the side-street traffic. Sidestreet and through traffic are in conflict, so the fitness function will have to take into account the desired tradeoff. Other factors are volume of traffic along the main street under different traffic conditions (heavy, medium, slow).

4.3 How might the GP crossover process be changed to allow GP to relax the requirement that the execution of every subtree return a value?

One change might be that only those subtrees corresponding to value-returning programs would be selected for crossover. A disadvantage of this restriction is that candidate subtrees would actually have to be run to see if they returned values.

4.4 The crossover operation used in GP selects a random subtree in both parents. Comment on what you think the effects would be of biasing the random selection according to

1. Preferring those subtrees that were highly active during the fitness trials

2. Preferring larger subtrees to smaller ones, and vice versa

1. The only subtrees that are not active during a run are those corresponding to conditional branches not taken. Also, subtrees near the root of the tree are more likely to be active than those near the leaves. Thus, one possible effect of this bias would be to prefer larger subtrees associated with frequently met conditions to smaller subtrees associated with seldom met conditions.

2. Crossover of large trees (those nearer the root) makes more drastic changes in the offspring than does crossover of small trees (those nearer the leaves).

4.5 How could an evolutionary process, like GP, be used to evolve

1. Neural networks?

2. Production systems?

Describe in some detail. In particular, how would the crossover operation be implemented? Comment on whether or not you allow Lamarckian evolution in the evolution of neural networks.

1. A straightforward way to evolve neural networks would be to start with random ones. Crossover could be effected by trading hidden units among the parents. The number traded could be randomly selected, or it could be a parameter of the GP process. Mutation operators might be used to make random changes to the weights. At the same time, each neural network might be trained to do as well as it could for the given fitness function. (Such training might have to employ a process called reinforcement learning.) Lamarckian evolution would allow reproduction of the trained networks. That is, if hidden units were traded, they would be the trained hidden units. Ordinary Darwinian evolution would trade the untrained versions of the hidden units, although the parents selected to reproduce would be those whose training achieved high fitness. Another method of using GP to evolve neural networks has been explored by Gruau. (Gruau, F., "Genetic Synthesis of Modular Neural Networks," in Forrest, S.

(ed.), *Proc. of the Fifth International Conference on Genetic Algorithms*, San Francisco: Morgan Kaufmann, 1993.) He evolved programs (the genotype) using the standard GP process. Executing these programs produced neural networks (the phenotype). Those programs that produced the most fit neural networks were selected for reproduction.

2. A production system is a collection of rules. The GP process could start with a random population of production systems. Crossover could be effected by trading rules among the parents. (Several choices are possible here.) Mutation operators might be used to make random changes to the production rules themselves and/or to promote or demote production rules (changing their positions in the ordering).

5.1 A discrete elevator can sense the following information about its world:

1. What floor the elevator is stopped at.
2. What floors passengers in the elevator want to go to.
3. What floors passengers outside of the elevator want rides from and whether they want to go up or down.
4. The status of the elevator door (open or closed).

The elevator is capable of performing the following actions:

1. Go up exactly one floor (unless it is already at the top floor).
2. Go down exactly one floor (unless it is already at the bottom floor).
3. Open the elevator door.
4. Close the elevator door.
5. Wait Δ seconds (a fixed time sufficient for all in the elevator to get off and for all outside the elevator to get in).

Design a production system to control the elevator in an efficient manner. (It is not efficient, for example, to reverse the elevator direction from going up to going down either if there is someone still inside the elevator who wants to go to a higher floor or if there is someone outside the elevator who wants to get on from a higher floor.)

- Based on what can be sensed, we define the following Boolean variables:

out_above_up—There is someone outside the elevator and on a higher floor who wants to go up.

out_above_down—There is someone outside the elevator and on a higher floor who wants to go down.

$out_above \equiv out_above_up + out_above_down$

out_below_up—There is someone outside the elevator and on a lower floor who wants to go up.

out_below_down—There is someone outside the elevator and on a lower floor who wants to go down.

$out_below \equiv out_below_up + out_below_down$

out_at_down —There is someone outside the elevator at the current floor who wants to go down.

out_at_up —There is someone outside the elevator at the current floor who wants to go up.

in_up —There is someone inside the elevator who wants to go up.

in_down —There is someone inside the elevator who wants to go down.

in_at —There is someone inside the elevator who wants to get out at the current floor.

We also have the following state variable:

up —The intended meaning is that the elevator is in the process of an upward journey. \overline{up} then has the intended meaning that the elevator is in the process of a downward journey.

We define the following actions:

go_up —The elevator goes up exactly one floor.

go_down —The elevator goes down exactly one floor.

$door_seq$ —The elevator executes the sequence of opening the door, waiting Δ seconds, and then closing the door. We assume that during $door_seq$ all who want to get on or off the elevator do so and that the corresponding sensory features are reset accordingly.

Here is a production system for controlling the elevator:

$in_at + up \cdot out_at_up + \overline{up} \cdot out_at_down \rightarrow door_seq$

$up \cdot in_up \rightarrow go_up$

$up \cdot out_above \rightarrow go_up$

$up \rightarrow set\ up\ to\ 0$

$\overline{up} \cdot in_down \rightarrow go_down$

$\overline{up} \cdot out_below \rightarrow go_down$

$\overline{up} \rightarrow set\ up\ to\ 1$

- 5.2 An "artificial ant" lives in a two-dimensional grid world and is able to follow a continuous "pheromone trail" (one cell wide) of marked cells. The ant occupies a single cell and faces either up, left, down, or right. It is capable of five actions, namely, move one cell ahead (m), turn to the left while remaining in the same cell (l), turn to the right while remaining in the same cell (r), set a state bit "on" (on), and set the state bit "off" (off). The ant can sense whether or not there is a pheromone trace in the cell immediately ahead of it (in the direction it is facing), and whether or not the state bit is on. (Assume that the state bit is off initially.) Specify a production system for controlling the ant such that it follows the trail. Assume that it starts in a cell where it can sense a pheromone trace. (Recall that a production system consists of an ordered set of condition-action rules; the action executed is the one that corresponds to the first satisfied condition. A rule might have more than one action on its right-hand side.) Make sure your ant doesn't turn around and retrace its steps backward!

- We use the following notation:

x_1 is true iff there is a pheromone trace in the cell directly ahead of the ant.

x_2 is true iff the state bit is on.

Here is the production system:

$x_1 \rightarrow m, off$

$\overline{x_2} \rightarrow l, on$

$l \rightarrow r$