Tartarus 2

Experiment 1: Running with default values:

80 steps average across 10 runs: 8.756

160 steps average across 10 runs: 8.53

Increasing the number of steps did not significantly improve fitness, and we think that this is because the dozer ends up repeating the same move until the end. So any fitness that was not 10 was like due to the dozer simply going forward the whole time.

Experiment 2: Adding PROG2 #####<not really sure why increasing number of moves doesn’t help>####

80 steps average across 10 runs: 4.96

160 steps average across 10 runs: 5.29

The overall fitness has been greatly improved by the addition of PROG2. This can be attributed to the fact that the dozer now has the capability of executing certain logic one after another. This could be beneficial since there may be a certain move that is optimal after the execution of another move. We observed that increasing the number of steps did not increase fitness, and we think that this may be because the complexity of executing one move after another was not quite enough.

Experiment 3: adding PROG3

80 steps average across 10 runs: 4.175

160 steps average across 10 runs: 3.28

Adding this node improved the average fitness values again, but this time, it seems that increasing the number of steps does improve the fitnesses. We are not quite sure why this is the case since PROG3 can be mimicked by a PROG2 that has another PROG2 as a child. Perhaps the option to execute two in a row is significantly more complex than two.

Experiment 4: Evaluating with different tree complexity settings. ####

2, 4, 25: average 5.185

4, 8, 50: average 2.705

8, 16, 100: average 3.28

16, 32, 200: average 2.925

It seems that in general, increasing tree complexity improves the fitness, but there is an anomaly in the second case.

Giving the dozer more sensory information \*\*\*

When the dozer was given the ability to look two blocks forward, the results’ fitnesses did not improve much at all at 3.34 average across 10 runs of 160 steps.

Adding our own modifications

Modification 1: dozer can see if there is a box in front at any point.

We hypothesized that since we are giving the dozer up to 5 additional spaces of information, the dozer’s ability to decide when to go forward will be improved.

To implement, the modification made in the previous part was undone, and the new node was given an arity of 2. It evaluated its left child if there was a box in its line of sight, and if not, it evaluated its right child.

After 10 runs of 160 steps, the average fitness was 2.5.

This is certainly better than our results from after adding PROG3, but while observing the visualizations, we found that it would waste a noticeable number of moves. On our best individual, which was able to achieve a fitness of 1 on several grids, would go towards a block if it detected it in its line of sight. Since it was not able to tell if it was against a wall until it was directly adjacent, it would waste up to 4 moves.

Modification 2: sequential execution conditioned on front detection

This new node would evaluate both children if there was a box in front of the dozer, and if there wasn’t it would evaluate just one. We hypothesized that this would give the dozer the opportunity to execute a certain sequence of actions only when it had moved a box so that it would be immediately ready for the next.

For 10 runs of 160 steps, the average fitness was 2.925

This is a slight improvement on top of what we saw after implementing PROG3. We think that the intricacies of this node was lost in the fairly random nature of genetic programming. It is possible that this case is generally covered by PROG2 and PROG3.