## Base Case Experiment

The strategy we created essentially performs two spiral loops starting from one corner of the board, and afterwards moves forward 65% of the time, and turns left 35% of the time. Here's a table of average fitness results from one run with 1000 boards per strategy, where strategy 1 is equal chance of all moves, 2 is 60% chance to move forward and a 20% chance to move left or right, 3 is equal chance for all possibilities, except the dozer cannot turn in the same direction twice, and 4 is our strategy.

| | 80 moves | 160 moves |
|---|---|---|
| 1 | 8.993 | 8.246 |
| 2 | 8.585 | 7.446 |
| 3 | 8.762 | 7.842 |
| 4 | 6.708 | 4.971 |

## Tartarus 1

### Experiment 1

With default settings and allowing 80 moves per board, the average of the best fitnesses from 10 runs was 8.655. With 160 moves, the average was 8.57.

Increasing the number of moves taken cannot possibly decrease fitness, because once a block is against the wall, it cannot be pushed away. Therefore adding more turns can only result in a better or equal result. However, after watching the visualizations, within the first 60 turns or so the robot gets stuck moving forward against a wall or 2 blocks, or repeats the same path over and over again. In this manner, adding extra turns does not help significantly improve the fitness of the best individuals.

### Experiment 2

With the new terminal node that returns $0, 1$, or 2, using default settings, and allowing 80 moves per board, the average of the best fitnesses from 10 runs was 6.6. With 160 moves, the average was 4.935.

Including this new node resulted in individuals that were certainly better than random, and slightly better than our strategy. Based on the visualization, this is because the occasional random move from this new terminal causes the robot to get unstuck.

In the very best genomes, this new node appeared quite often. Only a few of the last functions did not contain this new terminal as at least one of its terminals. Across all the genomes, it almost always showed up at least a couple times.

### Experiment 3

We changed the following settings from default to get our best results:

- Population size: up to 200
- Crossover probability: down to 20.0
- Tournament size: up to 20
- Swap mutation probability: up to 50.0
- Shrink mutation probability: up to 50.0

Increasing mutation (both swap and shrink) improved our best average fitness. This helped because it diversified the individuals, which prevented early convergence and potentially explored more of the fitness landscape. Increasing shrink mutation also likely helped to prevent bloating. Increasing the population size helped in a similar manner, since we were able to begin with a wider range of individuals, therefore allowing us to explore more of the search space before converging. Increasing the tournament size to 20 was partly a result of increasing the population size, although in general having stronger selection was helpful in providing a good amount of evolutionary pressure while allowing mutation to create stronger genomes. Decreasing crossover probability can potentially help prevent bloating, but lowering crossover did not show significant improvements in fitness.

### Experiment 4

The average best fitnesses were as follows: 5.065 for $\{2, 4, 25\}$, 4.975 for $\{4, 8, 50\}$, and 4.555 for $\{16, 32, 200\}$.

Decreasing the max tree size values greatly reduced real clock running time for each generation. Similarly, increasing these values greatly increased the running time for each generation, especially so for generations $\approx 20$ and upwards. Decreasing the size decreased running time because there are fewer nodes to evaluate. Similarly, increasing the size increased running time because there are more nodes to evaluate.

Decreasing the max tree sizes seems to hinder the evolution of good strategies. This could be because this limits the creativity of the algorithm to a smaller set of possible solutions. Allowing the tree size to grow very large increased the diversity of solutions, making it perform better than the two smallest settings ( {2, 4, 45} and {4, 8, 50} ). However, this led to the solutions becoming too bloated, and so the largest settings did not perform better than the defaults {8, 16, 100} in either fitness or running time.

## Tartarus 2

### Experiment 1

With default settings and allowing 80 moves per board, the average of the best fitnesses from 10 runs was 8.53. With 160 moves, the average was 8.485.

Similarly to experiment 1 with Tartarus 1, the dozer often gets stuck, so adding moves does not significantly improve fitness. Based on the visualizations, this is the same reason that this dozer does not perform better than random, let alone our hand-made solution.

### Prog2

With the new prog2 node, using default settings, and allowing 80 moves per board, the average of the best fitnesses from 10 runs was 4.805. With 160 moves, the average was 4.0.

This means that Tartarus2 with prog2 performed better than all our other tests, including the Tartarus1 tests with better parameters. This is because the prog2 function allows the dozer to have information about past moves, which means it can make smarter decisions.

### Prog3

Adding prog3 significantly improves the fitnesses of the individuals, because the dozer has even more information about what it has done in the past. On top of improving fitnesses, prog3 also become much more prominent than prog2 within the genomes.

### Experiment 4

The average best fitnesses were as follows: 4.785 for $\{2, 4, 25\}$, 3.365 for $\{4, 8, 50\}$, 3.12 for $\{8, 16, 100\}$, and 3.065 for $\{16, 32, 200\}$.

The changes in max size did not have a noticeable affect on real clock time. Increasing max size always improved the average best fitnesses, but the amount by which it improved decreased as size went up. So, the fitnesses improved greatly going from $\{2, 4, 25\}$ to $\{4, 8, 50\}$, but not very much when going from $\{8, 16, 100\}$ to $\{16, 32, 200\}$. This could be because the negative side effects of bloat start to manifest as the maximum size increases. Simultaneously, though, bloat doesn't seem to be as detrimental to fitness as it was with Tartarus 1, since there still was a small improvement from $\{8, 16, 100\}$ to $\{16, 32, 200\}$. A larger tree means more prog nodes, and more prog nodes means more knowledge about past moves, which could partially explain why we saw these results.

## More Sensory Information

### Tartarus 1

With the new sensory node, default settings, and allowing 160 moves per board, the average best fitness was 4.11, which is an improvement from all other Tartarus 1 experiment results. When using the settings from Experiment 3, the average was 3.445.

The added information about the dozer's surroundings seemed to allow it to make smarter decisions about where to move, and the results reflect this. This aligns well with a more human method of solving the problem, where we base our move on the location of the boxes and the dozer in relation to one another rather than following some specified pattern.

### Tartarus 2

With the new sensory node, default settings, and allowing 160 moves per board, the average best fitness was 3.34.

Adding this new sensory node slightly worsened the fitnesses of the indiduals. Within the genomes, the new sensory node rarely showed up more than once. This means that this extra sensory data was not beneficial for the dozer, because it was mostly evolved away. Perhaps the benefits of other nodes outweigh the benefits of this new node, making it appear less prominently. This is in stark contrast to Tartarus 1, where this new sensory data was beneficial in comparison to the other available nodes.

## Choose Your Own Adventure

### Tartarus 1

Adding more sensory input to the dozer improved the individuals. Therefore, adding more input could potentially be helpful, since the dozer would know more about its surroundings. So, we decided to add three new sensing nodes to check the surroundings two spaces to the right, left, and behind the dozer. We hypothesized that the best fitnesses would increase as a result. However, the average best fitnesses across 2 tests of 10 runs using the default parameters were 4.245 and 4.24, which was slightly worse than sensing only 2 spaces forward (although it still was better than not sensing 2 spaces ahead at all). In the genomes, the sensing two spaces forward node gets used the most out of the four extra sensing nodes, but all of the nodes do get used. This indicates that sensing two spaces forward tends to be more beneficial than sensing in the other directions.

### Tartarus 2

Since prog3 improved upon prog2, we thought that adding higher progs would further improve the indviduals– perhaps as a result of the dozer having more information. Therefore we created a prog4 node in addition to prog2 and prog3, and ran this with the default parameters as before. We hypothesized that prog4 would show up often in the genome, improving the fitness as it did when we introduced prog3. We ran 3 separate tests, each with 10 runs. The fitness results coincide with this hypothesis, since the average best fitnesses for the tests were 2.525, 3.185 and 2.67, which averages to 2.79 over 30 runs. However, when looking at the genomes from these runs, prog4 didn't show up as often as we expected. When we introduced prog3, prog2 became much less prominent in the genome. But, with the addition of prog4, prog3 is still the most prominent prog node. The prog4 node was still being used and clearly contributed to an improved fitness, but it did so in a less prominent manner than we initially anticipated.