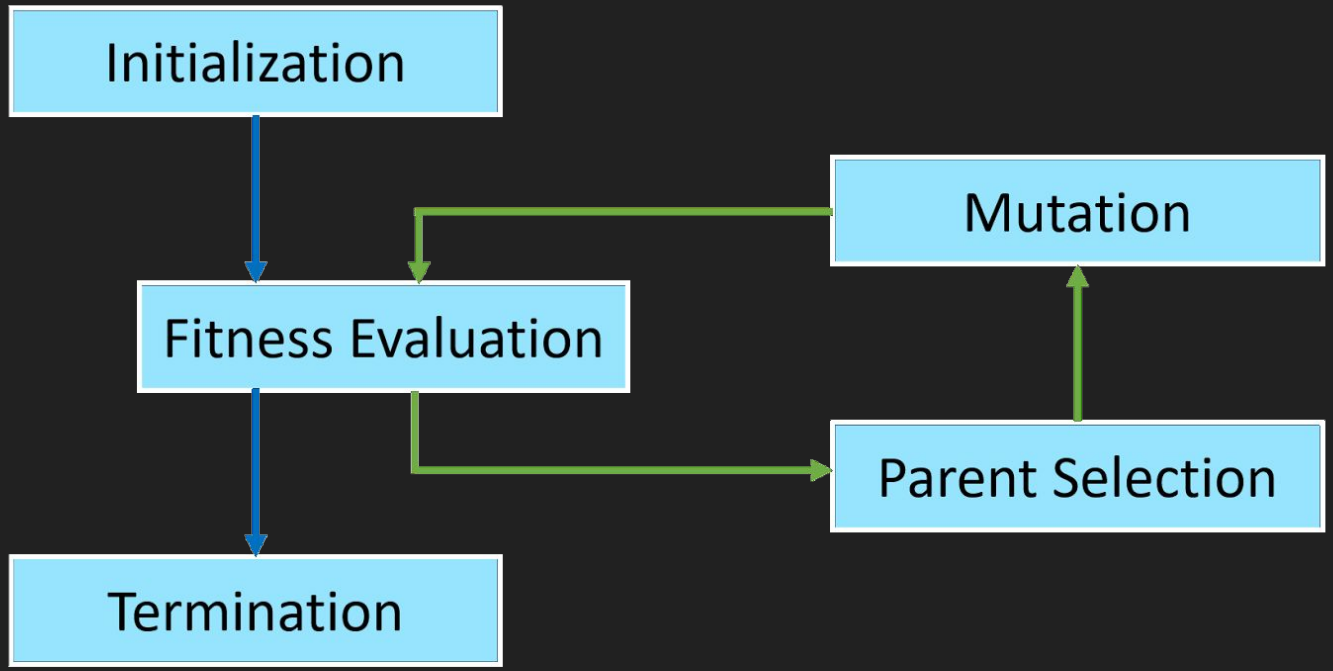
Evolutionary Algorithms:

Inducing Decision Trees

Alexander Arnold, Brian Lee, Zen Ly, Ammar Abu Shamleh

Brief Overview of Genetic Algorithms

The most common Evolutionary Algorithm



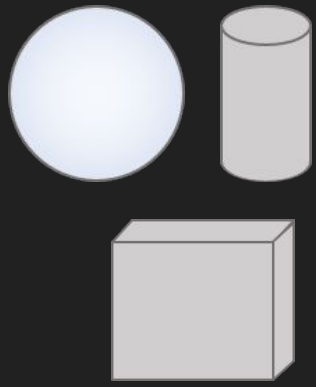
Intro to the problem

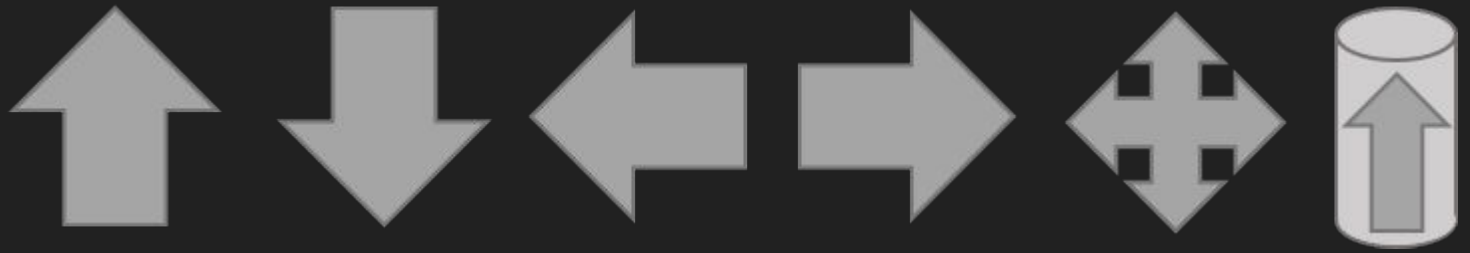
Robot Cleaner

2D grid of squares with 50% chance to have a can in each

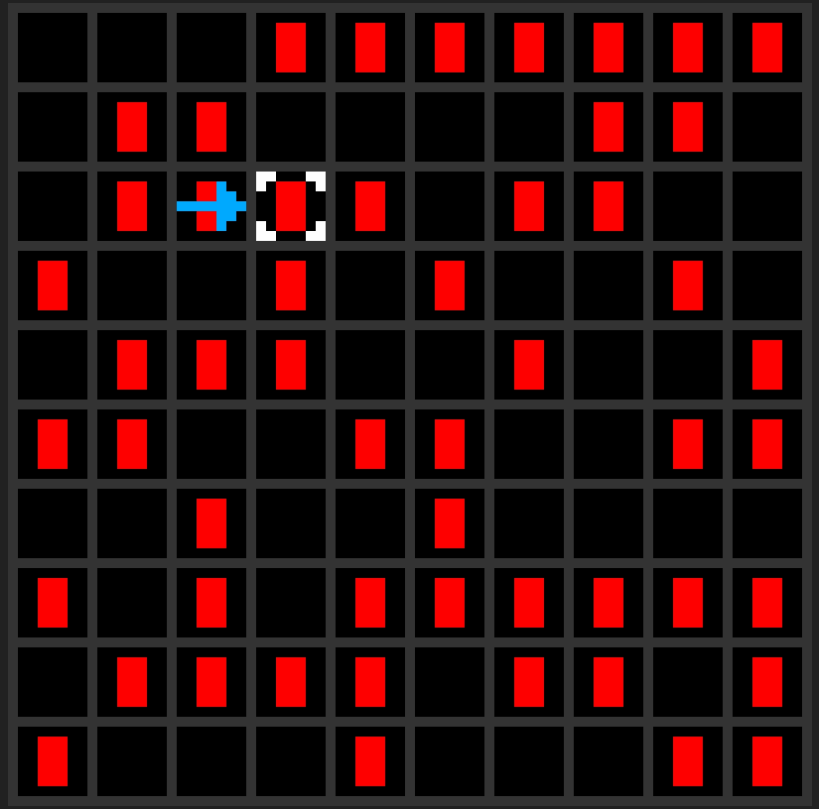
Inputs: Four Adjacent Squares + Current Square (Empty, Can, Wall)

Actions: Up, Down, Left, Right, Random Movement, Pick Up Can





Example Layout



Testing Solutions

10 points for a can pick up

-1 for trying to pick up a can on an empty square

-5 for trying to move into a wall

Amount of moves given is 2 * the number of squares in the room

The average points over 15 runs is the fitness

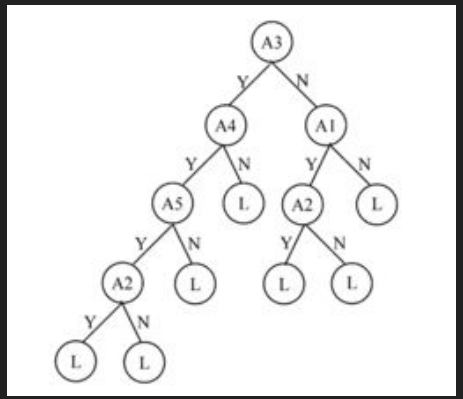
Decision Trees

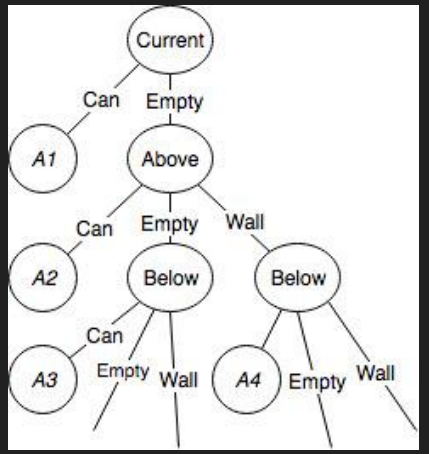
Decision tree is an easy way of representing decisions

Nodes represent conditions, and edges represent possible evaluations/outcomes

Leaf nodes represent actions (or classifications, depending on context)

Our project looked at using a GA to find values for leaf nodes

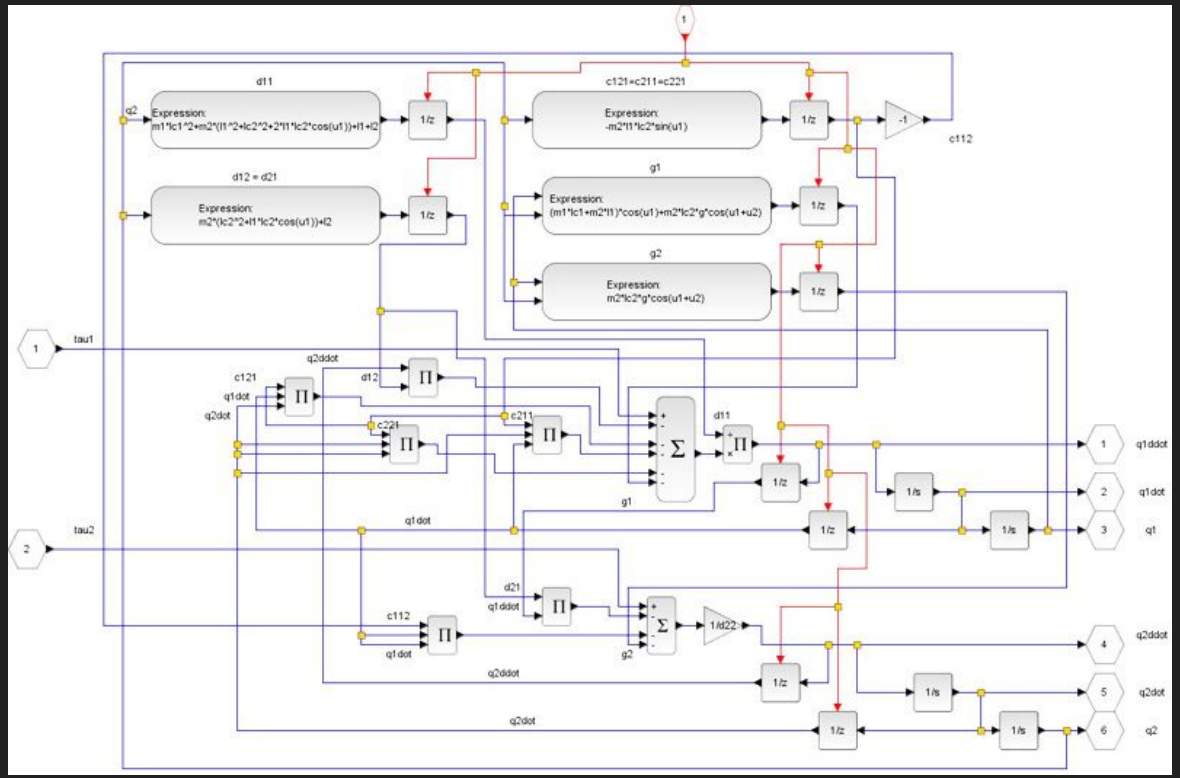




Algorithm Design Overview

1. Solution encoding
2. Random solutions are generated
3. Fitness evaluated
4. K-way tournament parent selection
5. Crossover
6. Mutation
7. New population generation
8. Repeat steps 3-7 for fixed number of generations

How Can We Represent a Control Scheme?



Encoding

[3,5,3,3,5,3,2,5,2,2,5,2,2,5,2,2,5,2,3,5,3,3,5,3,1,5,1,1,5,1,1,5,1,1,5,1,2,5,2,2,5,2,2,
5,2,1,5,1,1,5,1,1,5,1,3,5,3,3,5,3,2,5,2,2,5,2,2,5,2,2,5,2,3,5,3,3,5,3,0,5,0,0,5,0,0,5,0
,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,
5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,0,5,0,3,5,3,3,5,3,2,5,2,2,5,2
,2,5,2,2,5,2,3,5,3,3,5,3,1,5,1,1,5,1,1,5,1,1,5,1,1,5,1,1,5,1,1,5,1,1,5,1,1,5,1,1,5,1,3,
5,3,3,5,3,2,5,2,2,5,2,2,5,2,2,5,2,3,5,3,3,5,3,5,5,5]

Make sense?

Encoding

Solutions are represented as a 243 character string

The robot can always see 5 squares. With 3 possibilities per square, there are 243 combinations

Each element in the string corresponds to one of the situations, and specifies the action taken for that situation

Fitness Function

15 simulations for each solution. Scores are averaged

The average score is the solutions fitness

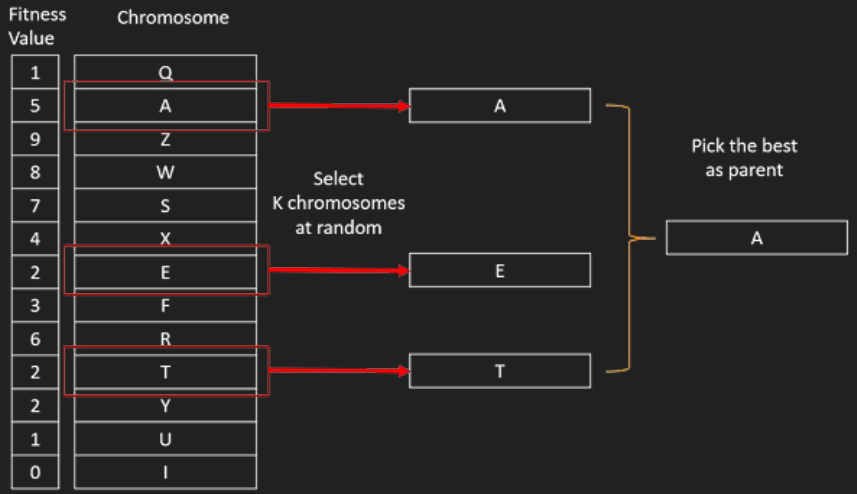
Scoring system is as described earlier

Parent Selection

As fitnesses can be negative, Roulette Wheel Selection was unsuitable

We chose to use k-way tournament selection

Also gives us more control (can vary k)

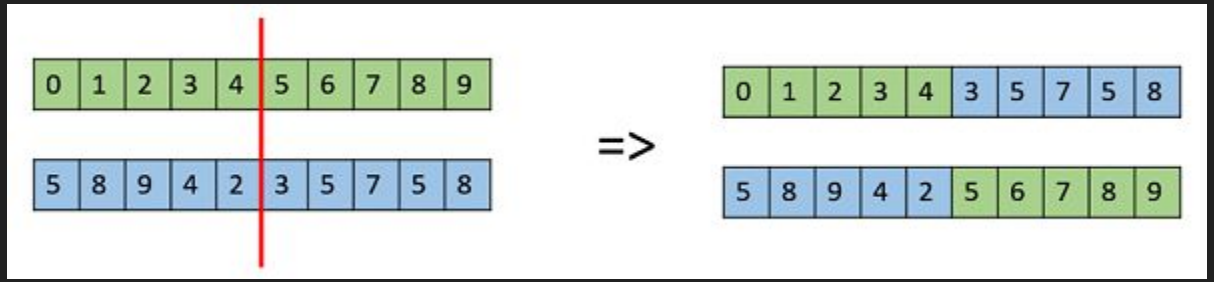


Crossover

Simply pick a random point between 0 and 243, and split the two parents

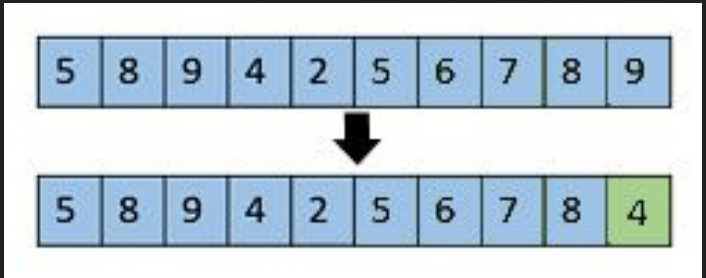
Form two new children

Never produces any invalid solutions



Mutation

Simply pick a random point in the string, and randomize it between 0 and 6 (inclusive)



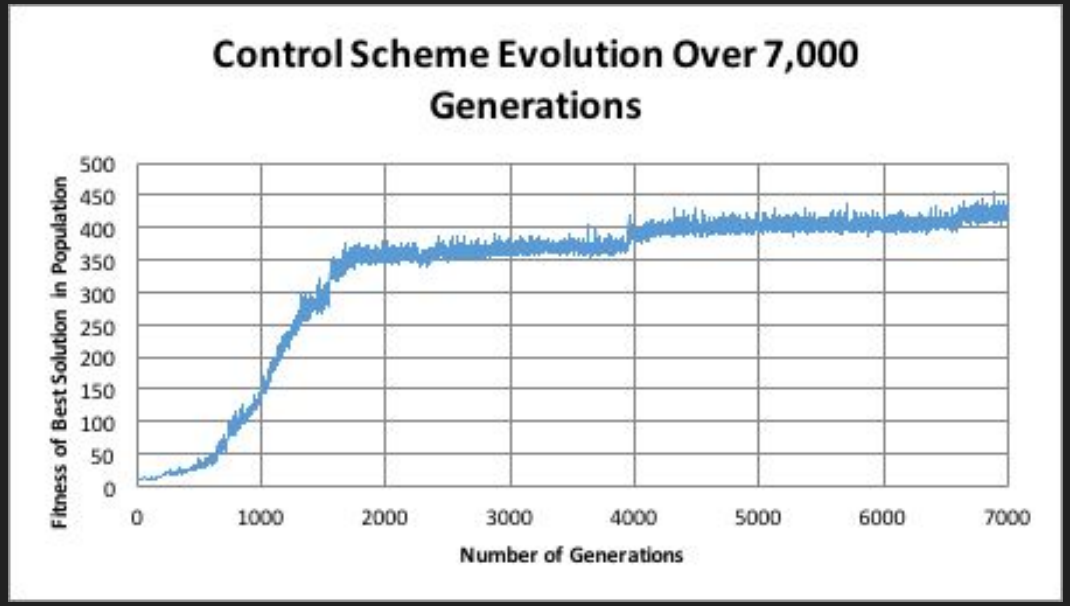
After mutation, the solution is added to the new population

An entirely new population of solutions is generated, and becomes the next generation (fixed population size of 200)

Hand Coded Solution

- Tried two hand coded solutions
- One suggested by Lyndon in the introductory presentation
  - Move randomly and pick up a can if you on top of one
  - Picks up on average 19 cans and hits a lot of walls.
  - Average of 90 points
- Our own hand coded solution
  - Tries to avoid walls
  - If the robot can see a can, move in that direction
  - Picks up on average 7 cans
  - Average of 76 points
- Both run poorly
  - Never picks up more than 20 cans
  - Never gets more than 100 points

Initial Results



- Average fitness - 358
- Average number of cans in grid- 50 cans (max points is 500)
- Completely surpasses hand-designed solution even in initial run
- BUT can we improve upon this?

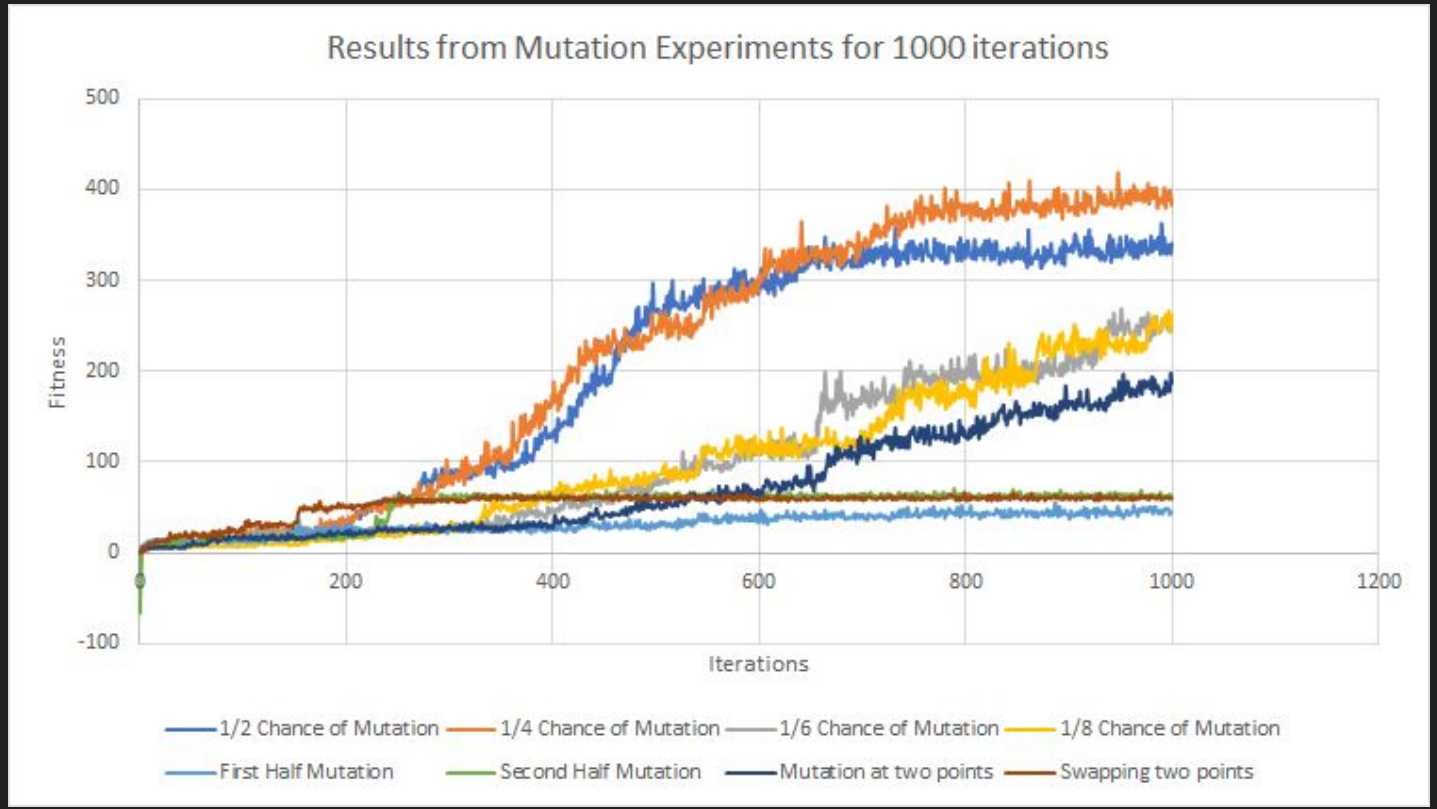
Finding Good Parameters

- Experiments to identify good parameters for the GA
- Parameters we experimented on:
  - How the individuals were mutated
  - How crossover was performed
  - How the parents were selected
- 5 trials of 1000 generations
- Keeping each of the other parameters the same

Experiments - Mutation

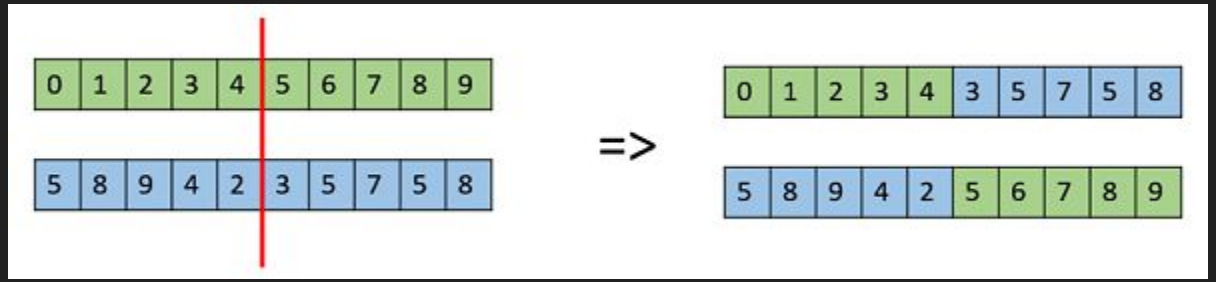
- In initial solution - 1/6 chance of mutation
- In our experiments - alter the chance of mutation
  - ½ chance of mutation
  - ¼ chance of mutation
  - ⅛ chance of mutation
- Mutating only in the first half or only in the second half of the genome
- Mutating the individual’s genome at two points
- Swapping two points in an individual with one another.

Results of Mutation Experiments

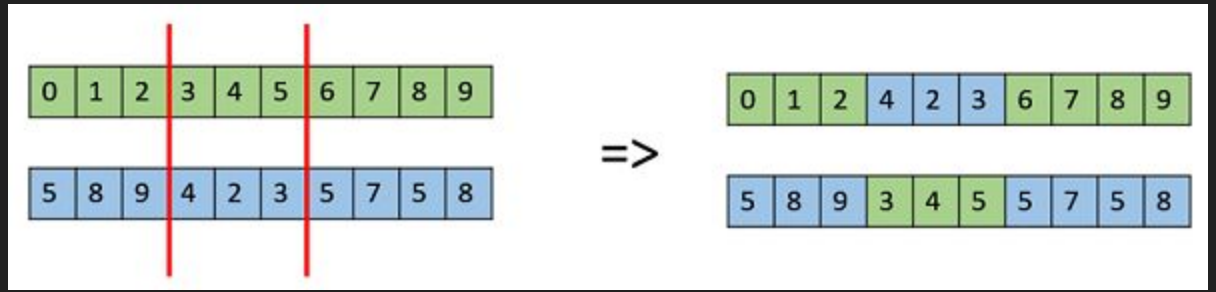


Experiments - Crossover

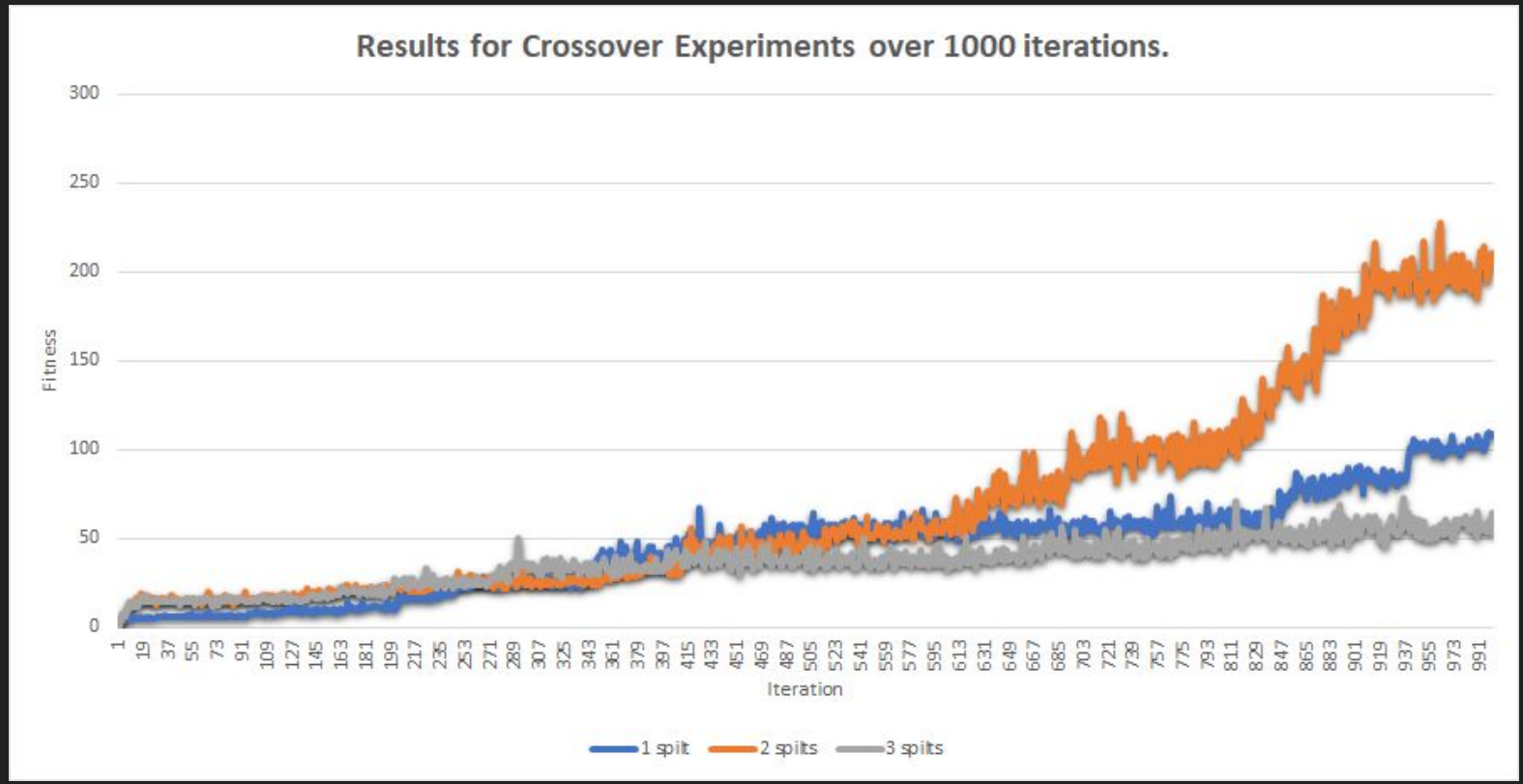
- In initial solution - one splice point for crossover



- In experiments - increasing the number of splices points - more combinations



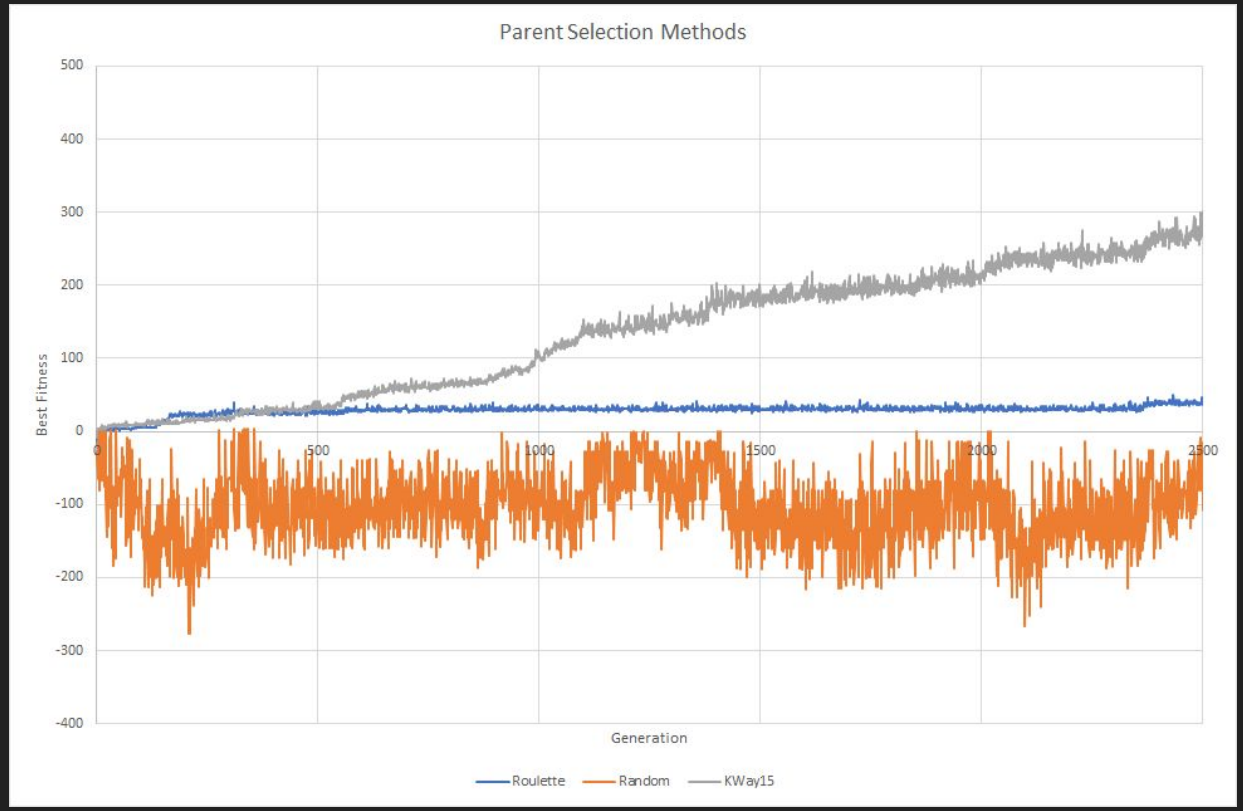
Results for Crossover Experiments



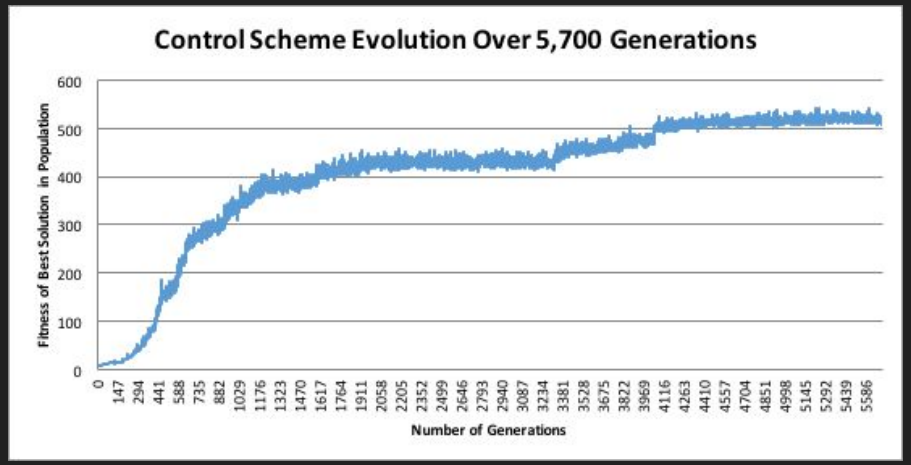
Experiments - Parent Selection

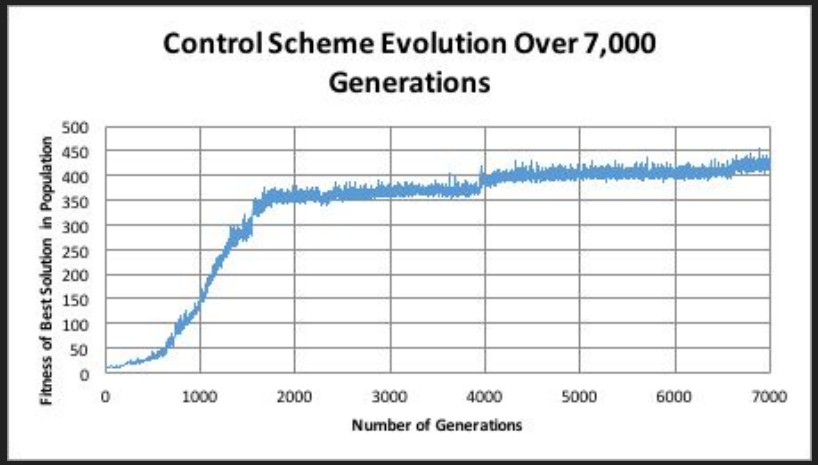
- In initial solution - K-way tournament selection with K = 15
- In experiments
  - Changing the value of K
  - Using different parent selections methods
    - Roulette wheel
    - Random selection

Different Parent Selection Methods



Results with optimised parameters

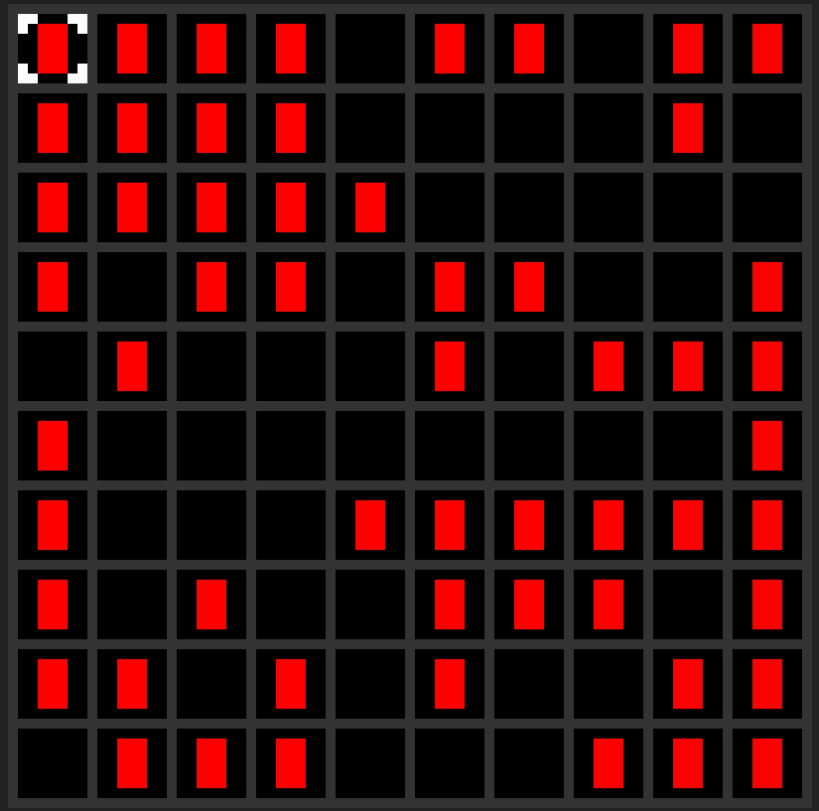




Brief analysis of the solution the GA found

[1,5,4,3,5,5,2,5,3,2,2,1,3,2,3,2,5,6,0,5,1,3,5,2,1,1,6,6,5,2,3,1,1,1,5,1,2,1,6,3,2,0,2,
5,4,3,5,0,3,5,1,0,5,5,2,5,5,2,2,3,1,2,6,2,5,4,2,3,0,2,5,4,0,5,6,3,2,1,0,2,1,0,5,4,0,5,2
,2,0,0,2,2,4,2,2,6,3,6,4,0,5,6,3,0,6,2,5,6,0,1,1,6,1,0,1,5,0,0,1,4,3,2,3,3,1,2,0,5,5,3,
5,5,2,3,6,0,5,0,2,3,5,0,5,4,0,5,5,3,3,5,2,5,6,0,5,3,1,2,3,1,3,3,1,5,4,3,5,4,2,5,4,6,2,4
,1,2,4,2,5,6,3,5,6,3,5,3,1,4,5,1,5,6,1,1,3,1,5,4,1,2,6,0,2,5,2,5,5,1,5,0,6,6,5,5,4,5,3,
4,6,1,5,6,0,6,4,2,2,6,5,0,1,3,0,6,6,0,4,4,5,4,5,6,1]

The Solution Found (a bit more understandable)



Future Work

- Train the GA on different grid layouts and grid sizes
  - We only trained it on 50% chance of a can in each square; when training GA, randomize that probability
  - Randomize grid size
  - Randomize starting location
- More testing of parameters (computation limitations)

References

- Melanie Mitchell, *Complexity: A Guided Tour*
- https://www.tutorialspoint.com/genetic_algorithms/