Second Chance Mating In Genetic Algorithms

(Applied to Pathfinding)

Gregory L Whitman
Mentor: Dr. Armstrong

Problem Description

Implement a genetic algorithm by applying it to pathfinding and test the effectiveness of second chance mating.

Background: What is a Genetic Algorithm?

- A method for solving optimization problems by jumping around searching through a possible solution space.

- This done by repeatedly modifying a population of individual solutions by mating and mutation. Which will evolve the population toward more optimal solutions. This is based off of Darwin's theory of evolution.

Background: How a Genetic Algorithm Works?

- There are 5 basic steps to a genetic algorithm.
    - Step 1: Initialize population
    - Step 2: Fitness Function
    - Step 3: Selection
    - Step 4: Crossover/Mutation
    - Step 5: Generate Next Generation/population
    - Repeat Steps 2-5

Background: Initialization

Create a population with solutions that are completely random in the solution space.

Ex: The solution set is {a, b, c, d} and we want 5 individuals.

- Each individual is a random assortment of a, b, c, or d.
  - aabcdb
  - baaaac
  - cbabab
  - abcada
  - bacdda

Background: Fitness Function

- This is one of the most important parts of a genetic algorithm. If this is wrong then the genetic algorithm will fail.
- The fitness function goes through the population of individual solution and scores.
  - Higher being most fit (better solution)
  - Lower being least fit (worse solution)
- Each scoring algorithm will be different for each problem that is trying to be solved with the genetic algorithm.

Background: Fitness Function continued...

Ex: We want to get the string ‘aaaaaa’ we will give 10 points for every correct character in the correct location. Using our individuals from before we will score them.

Individuals/Genome:	Calculation:	Fitness score:
aabcdb	20/60	.33333
baaaac	40/60	.66667
cbabab	20/60	.33333
abcada	30/60	.50
bacdda	20/60	.33333

Background: Selection

- Selection is randomly selecting 2 parents to participate in Crossover/Mating.
- But each individual solutions chance of being selected may not be equal.
- This is because their chance of being selected is based on their fitness score. The higher it is the more likely they will be selected.

Background: Selection continued...

Continuing with our example:

Individuals:	Fitness Scores:	Chance of being selected:	
aabcdb	.33333	33.33 %	
baaaac	.66667	66.67 %	
caaaaa	.83333	83.33 %	←----------
abcada	.50	50.00 %	
bacddd	.16667	16.67 %	

As you can see the individual with the highest fitness will have the greatest chance of being selected to pass on his solution/DNA

Background: Types of Selection

Accept/Reject:

- This involves normalizing the fitness values between 0-1. This is their cumulative probability.

- Generate two random number r1 and r2,

  r1 being a random individuals fitness.

  r2 will be a random number between 0-1.

- If r1 >= r2 then that individual will be selected.
  If not then both numbers will be regenerated and compared again.

Background: Crossover/Mutation

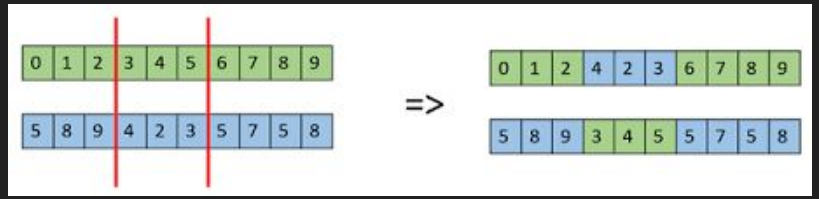
Take the two parents we previously selected and mate them.

This involves splitting their DNA/Genome up and then recombing them to create two new children which will be added the next generation this is known as Crossover.

Background: Crossover continued...

Double Point Crossover:

Parent A:



Child One:

Parent B:

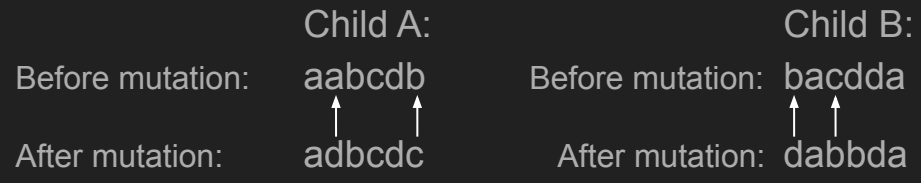
Child Two:

Background: Mutation

Now that we have created our two children from the previous step Crossover. It is time to introduce some added genetic variation (solution variation).

This is done by looping through the individuals genome and changing that specific value at that point to another option from the solution space.

- These are our two children
  - aabcdb
  - bacdda



Background: Filling Next Generation/Population

Now we take the two children that have been mutated and we begin to populate the next population. We fill the entire new population we children generated from the previous generation.

To do so we will repeat the selection and Crossover/Mutation steps until the population is full.

This generation will be more fit and contain better solutions.

Background: Repeat!

- With the new population now we can repeat these steps.
    - Step 1: Fitness Function
    - Step 2: Selection
    - Step 3: Crossover/Mutation
    - Step 5: Generate Next Generation/population
    - Repeat Steps 1-5
- After each repetition of these steps the population will have increasingly better solutions.

Background: Second Chance Mating

Now it’s time to talk about a variation of the genetic algorithm called second chance mating.

In second chance mating we will save spots in the new population and fill those saved spots with the fittest parents from the Old population.

Second Chance Mating: Continued...

Ex:

- New Population of size 10:
  - {Parent 1, Parent 2, Parent 3, Child 1, Child 2, Child 3, Child 4, Child 5, Child 6, Child 7}

The genetic algorithm will work exactly the same except for saving spots for some fit parents. And not creating a population full of children.

Second Chance Mating: Continued...

The reason for doing this is so that way a good solution/genome won’t be completely lost and gives the individual a second chance at passing on its genome.

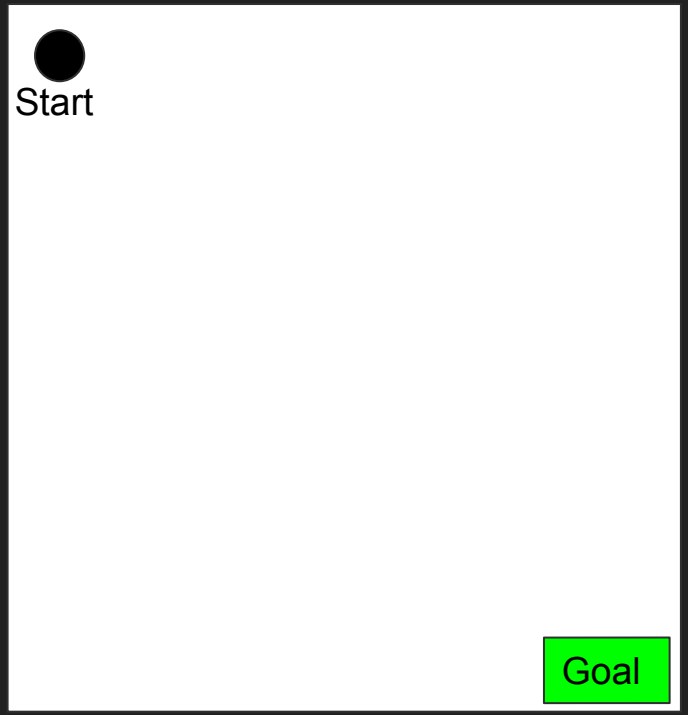
Primary Objective

The primary objective is to analyze the impact of the second chance mating approach in genetic algorithms.

Governing Propositions:

- The task will be a simple pathfinding problem.
- Limitation: 120 person-hours over 10 weeks

Simplified Pathfinding Example



We want our genetic algorithm to find a path from the start to the goal.

Initialization: Defining Our individuals

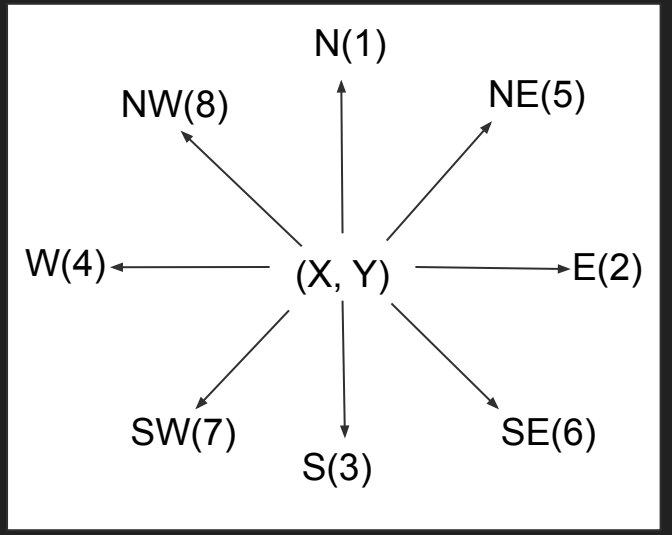
We must give our individuals 3 things to start with.

1. Fitness (how good solution is)
2. X and Y coordinates starting at the Start (position on field)
3. Genome (list of directions to travel)

Initialization: Encoding Our Individuals/Solutions

First we must figure out how we will encode our genome and what is means.

1 = North
2 = East
3 = South
4 = West
5 = North East
6 = South East
7 = South West
8 = North West



Initialization: Creating Our Genome

This is how we will know which direction to move.

1: North = Y+1

2: East = X+1

3: South = Y-1

4: West = X-1

5: North East = X+1, Y+ 1

6: South East = X+1, Y-1

7: South West = X-1, Y-1

8: North West = X-1, Y+1

Initialization: Defining The Length Of Our Genome

Obviously the directions 1, 2, 3, 4, ...etc wouldn't move us very far. So in order to fix this we make our genome length longer for instance a genome length of 170 would be 170 random integers between 1-8. (our directions)

Ex:

Genome: 623671237612376123456712376821672836481735417...etc

We can store this in an array

Initialization: Creating Our Population

Now that we have our individuals figured out we must make a population of them

This is simple we just repeat creating the individuals a set amount of times.

Ex: A population of 5 individuals with a genome length of 170 would be:

Individual 1: 125673471523476517465765127364172476821752 13 etc...
Individual 2: 348561783458134578135417845357813567135423 46 etc...
Individual 3: 151754871257451827547158476567541764517515 41 etc...
Individual 4: 567562764175347616687656766637636675763377 3 etc...
Individual 5: 125635677852357863256342234336785643687854 56 etc...

Fitness Scores: Testing Our population

- Now that we have our population it's time to test them and give each individual a fitness score.

- In order to do this we must loop through the population and at each individual loop through their genome (set of directions)

Fitness Scores: Testing Our population continued...

At the end of each individual's genome (meaning they used all their moves). We will calculate the distance between them and the goal.

We can do this using the distance equation:

Distance = |x1-x2| + |y1-y2|

Where the Red x1 and y1 are the Individuals coordinates.

And the Blue x2 and y2 are the goals coordinates.

Calculating the fitness:

- We will have to invert the distance because the shorter the distance to the goal the higher their fitness score should be and vice versa.

- We will take the previously calculated distance and take them
  Fitness = 1 / ((distance*distance) + 1))

- If they reach the goal they will receive a much higher fitness based off how how many moves it took them to get there
  Fitness = 100000/((moves*moves))

Selection: Normalization

Now that we have our individuals fitness scores we must use them to select parents to partake in Crossover eventually.

In order to do this we must first normalize all of the individuals scores to the populations cumulative probability. Which I will demonstrate.

Selection: Normalizing Fitness Scores continued...

First we must loop through our population summing their fitness scores. We will call this fitnessSum.

Then we will loop back through the population and correct the fitness scores by taking fitnessScore / fitnessSum.

This will force all the fitnessScores to be between 0-1 and the higher the Original fitness score the higher probability that individual will be selected.

Selection: Picking our Parents

We need two parents for Crossover.

We will pick them by using a method called Accept / Reject.

Selection: Accept or Reject

To begin we will randomly select an individual from the population and get their fitness. And then will will also generate a random number (r) between 0-1. Which is why we normalized everything.

If (r < individualsFitness)
Return parent
Else
Repeat

If (r) is less than the individuals fitness score then they are selected to mate.

We do this twice for two parents

Parent A, and Parent B

Crossover:

Now that we have our two parents it's time to mate them using Crossover.

This involves taking both the parents genome and combining them to produce two new children

Crossover: Double point mutation

Ex: Parent A :
141 | 2442 | 343

Parent B:
124 | 2342 | 341

Child 1:
141 | 2342 | 343

Child 2:
124 | 2442 | 341

At fixed points within the set genome copy over chunks of each Parent into the children in the order above.

Mutation:

Now that we have our children we must mutate them to introduce variation in the solutions.

Mutation is performed by looping through each of the children's genome. And at each step check to see if a randomly generated number mRate is <= the mutation Rate. And if it is then randomly chance that step to something else in the solution space.

Mutation: Continued...

Ex: Before Mutation:

Child 1:

1 4 1 2 3 4 2 3 4 3

Child 2:

1 2 4 2 4 4 2 3 4 1

Ex: Before Mutation:

Child 1:

1 2 1 2 3 4 3 3 4 3

Child 2:

1 2 4 2 1 4 2 3 4 1

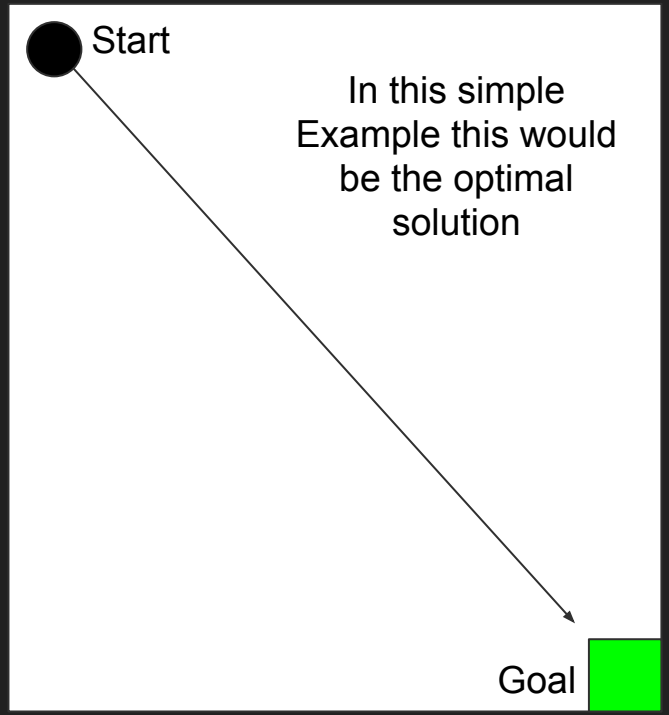
As you can see where mutation took place the direction at that current step changed

Filling the next Generation:

Now that we created and mutated the children it is time to repeat selection and Crossover/Mutation until the children that were created are enough to fill a new population.

And then repeat! After each generation the populations individual solution should get better. Closing in on an optimal solution.

What the optimal path will look like...



Which the genome if perfect would look like this…

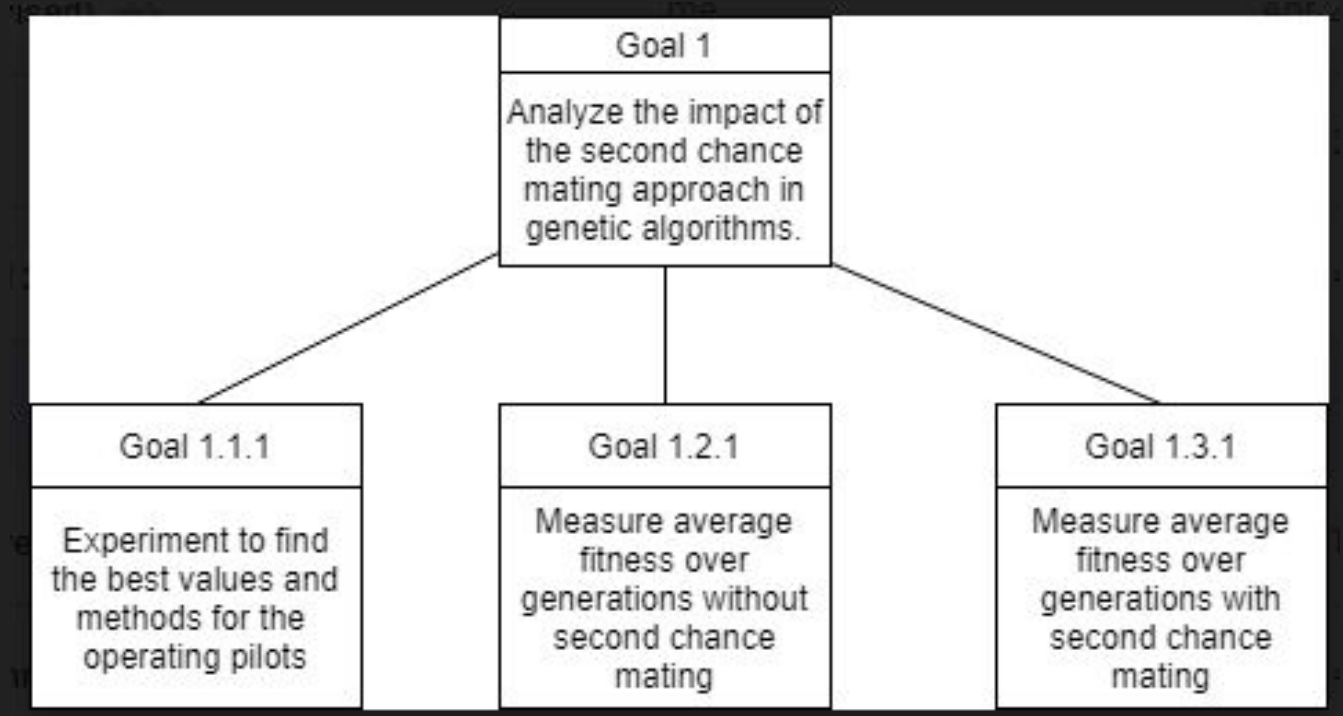
6666666666666666666...etc

Hypothesis

Second chance mating will do better than without for pathfinding.

Why: The more fit individuals DNA from the previous generation is not just thrown away. They have a second chance to pass on their DNA.

Goal Tree


Goal 1
Analyze the impact of the second chance mating approach in genetic algorithms.
Goal 1.1.1
Experiment to find the best values and methods for the operating pilots
Goal 1.2.1
Measure average fitness over generations without second chance mating
Goal 1.3.1
Measure average fitness over generations with second chance mating

Solution Description

Tools: Java, Eclipse IDE, Processing IDE, Personal Computer

Implement a genetic algorithm using second chance mating and evaluate its effectiveness when applied to pathfinding.

To do so:

1. Implement second chance mating
2. Remove second chance mating
3. Measure the average fitness of both the algorithms and compare them.

Experiment Design

- Setup two different genetic algorithms. One with second chance mating and the second without second chance mating.

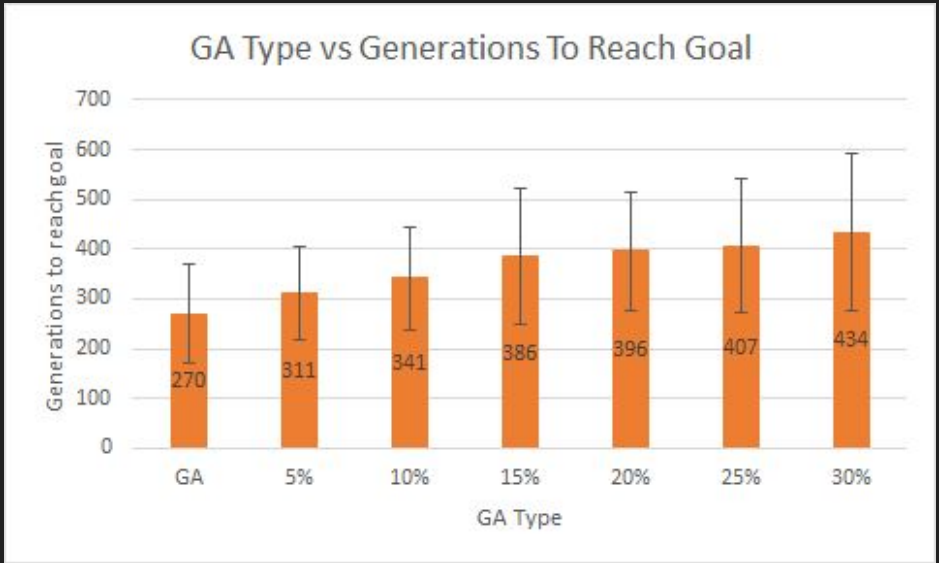
- Get a baseline using the genetic algorithm without second chance mating.

- Then run multiple tests on the second chance mating genetic algorithm each test will have a greater amount of individuals carried over from previous generation Ex: .05%, 10%, 15%...30%

Block Design

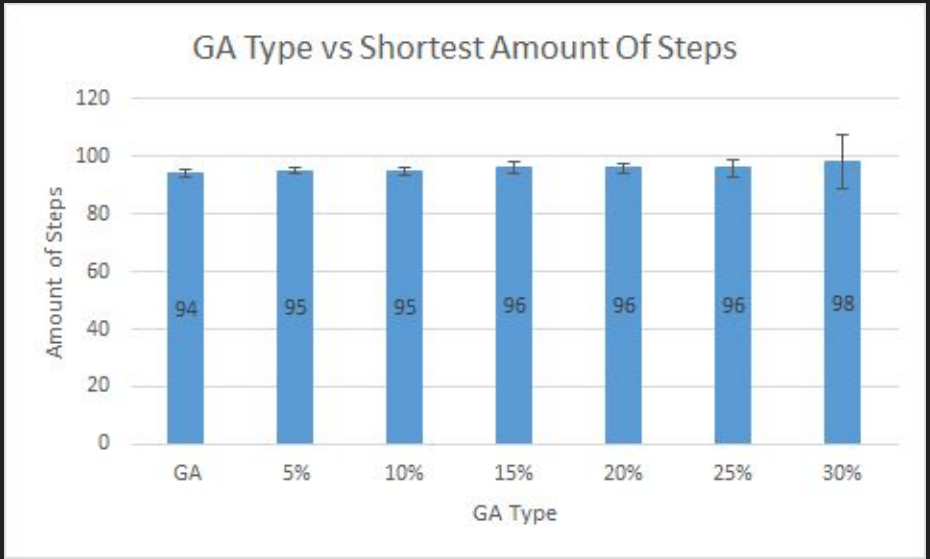
Algorithm:	Genetic Algorithm	SCGA 5%	SCGA 10%	SCGA 15%	SCGA 20%	SCGA 25%	SCGA 30%
Trials:	30	30	30	30	30	30	30
Genome Length:	170	170	170	170	170	170	170
Mutation Rate:	2.5%	2.5%	2.5%	2.5%	2.5%	2.5%	2.5%
Generations:	1000	1000	1000	1000	1000	1000	1000

Results



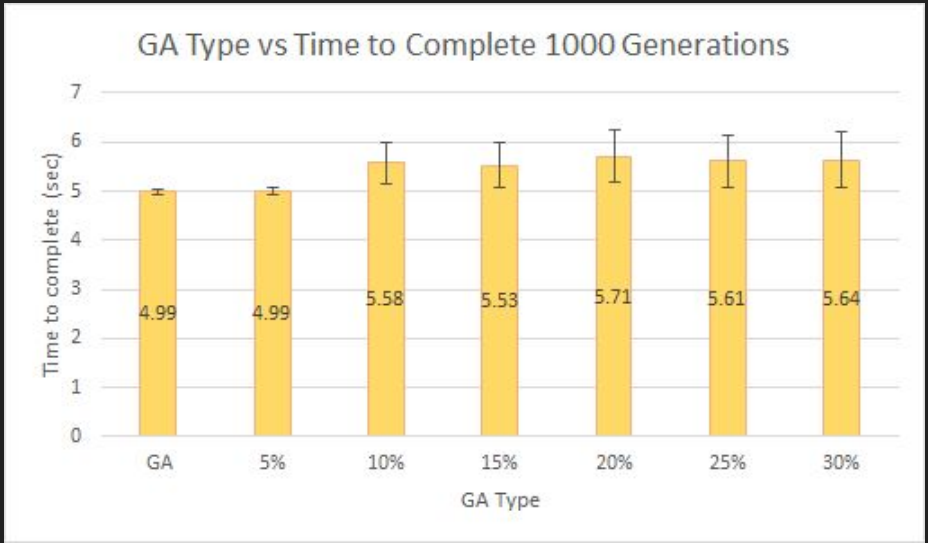
TYPE	Generations	Standard Deviation
GA	270	99
5%	311	95
10%	341	103
15%	386	137
20%	396	118
25%	407	136
30%	434	157

Results continued...



Type	Generations	Standard Deviation
GA	94	1.32
5%	95	1.10
10%	95	1.29
15%	96	2.12
20%	96	1.80
25%	96	2.85
30%	98	9.26

Results continued...



Type	Average Time (sec)	Standard Deviation
GA	4.99	0.06
5%	4.99	0.07
10%	5.58	0.42
15%	5.53	0.45
20%	5.71	0.53
25%	5.61	0.52
30%	5.64	0.58

Analysis

Generations to first reach goal:

T TEST						
	5%	10%	15%	20%	25%	30%
P VALUE	0.101555655	0.008352888	0.000373744	3.15245E-05	3.51982E-05	9.54403E-06
Alpha = .05	INSIGNIFICANT	SIGNIFICANT	SIGNIFICANT	SIGNIFICANT	SIGNIFICANT	SIGNIFICANT

Shortest path found by algorithm:

T TEST						
	5%	10%	15%	20%	25%	30%
P VALUE	0.171854047	0.016029934	0.001071056	0.001753096	0.00064482	0.028585255
Alpha = .0	INSIGNIFICANT	SIGNIFICANT	SIGNIFICANT	SIGNIFICANT	SIGNIFICANT	SIGNIFICANT

Time to complete 1000 generations:

T TEST						
	5%	10%	15%	20%	25%	30%
P VALUE	0.983975411	2.47992E-10	1.81539E-08	5.32208E-10	2.29598E-08	1.07962E-07
Alpha = .05	INSIGNIFICANT	SIGNIFICANT	SIGNIFICANT	SIGNIFICANT	SIGNIFICANT	SIGNIFICANT

Conclusion

Based on the results of the experiment and statistical analysis. Second chance mating performed worse in number of generations to first reach the goal, shortest path found by the algorithm, and time taken to complete 1000 generations.

- Second Chance mating did not perform better than the genetic algorithm without second chance mating.

Future Work

- Experiment with carryover values less than 5% in second chance mating genetic algorithm

Sources

1. Adams, Chad, Hirav Parekh, and Sushil J. Louis. “Procedural Level Design Using an Interactive Cellular Automata Genetic Algorithm.” In Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO ’17, 85–86. Berlin, Germany: ACM Press, 2017. https://doi.org/10.1145/3067695.3075614.

2. Alaguna, Camilo, and Jonatan Gomez. “Maze Benchmark for Testing Evolutionary Algorithms.” In Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO ’18, 1321–28. Kyoto, Japan: ACM Press, 2018. https://doi.org/10.1145/3205651.3208285.

3. Carr, Jenna. “An Introduction to Genetic Algorithms,” n.d., 40.

4. Castelli, Mauro, Luca Manzoni, and Leonardo Vanneschi. “The Effect of Selection from Old Populations in Genetic Algorithms.” In Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation - GECCO ’11, 161. Dublin, Ireland: ACM Press, 2011. https://doi.org/10.1145/2001858.2001948.

5. Giardini, Giovanni, and Tamás Kalmár-Nagy. “Performance Metrics and Evaluation of a Path Planner Based on Genetic Algorithms.” In Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems - PerMIS ’07, 84–90. Washington, D.C.: ACM Press, 2007. https://doi.org/10.1145/1660877.1660888.

6. Lu, Yuxin, Yongzhong Wu, and Yongwu Zhou. “Order Assignment and Routing for Online Food Delivery: Two Meta-Heuristic Methods.” In Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence - ISMSI ’17, 125–29. Hong Kong, Hong Kong: ACM Press, 2017. https://doi.org/10.1145/3059336.3059349.

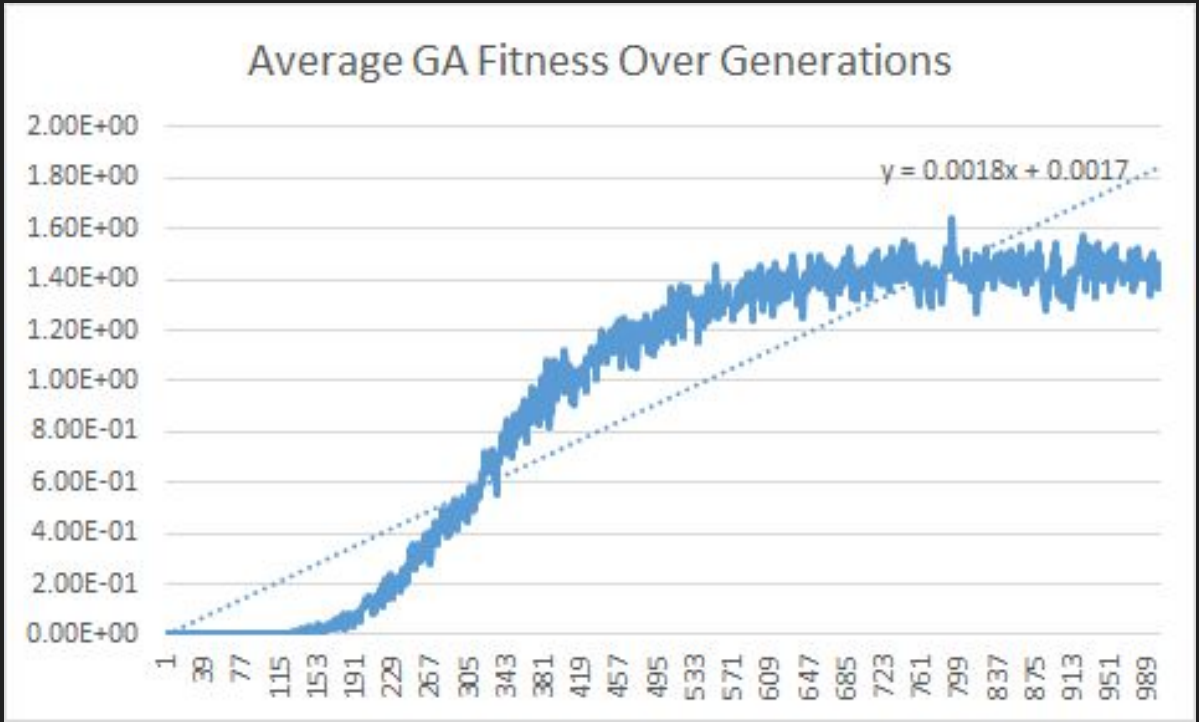
7. Rana, Prashant Singh, and Shivendra Pratap Singh. “Genetic Algorithm with Mixed Crossover Approach for Travelling Salesman Problem.” In Proceedings of the International Conference on Advances in Information Communication Technology & Computing - AICTC ’16, 1–4. Bikaner, India: ACM Press, 2016. https://doi.org/10.1145/2979779.2979824.

8. Russell and Norvig. “Artificial Intelligence A Modern Approach.” by Peter Norvig and Stuart Russell, 116–19, 2nd ed. Pearson Education, 1195.

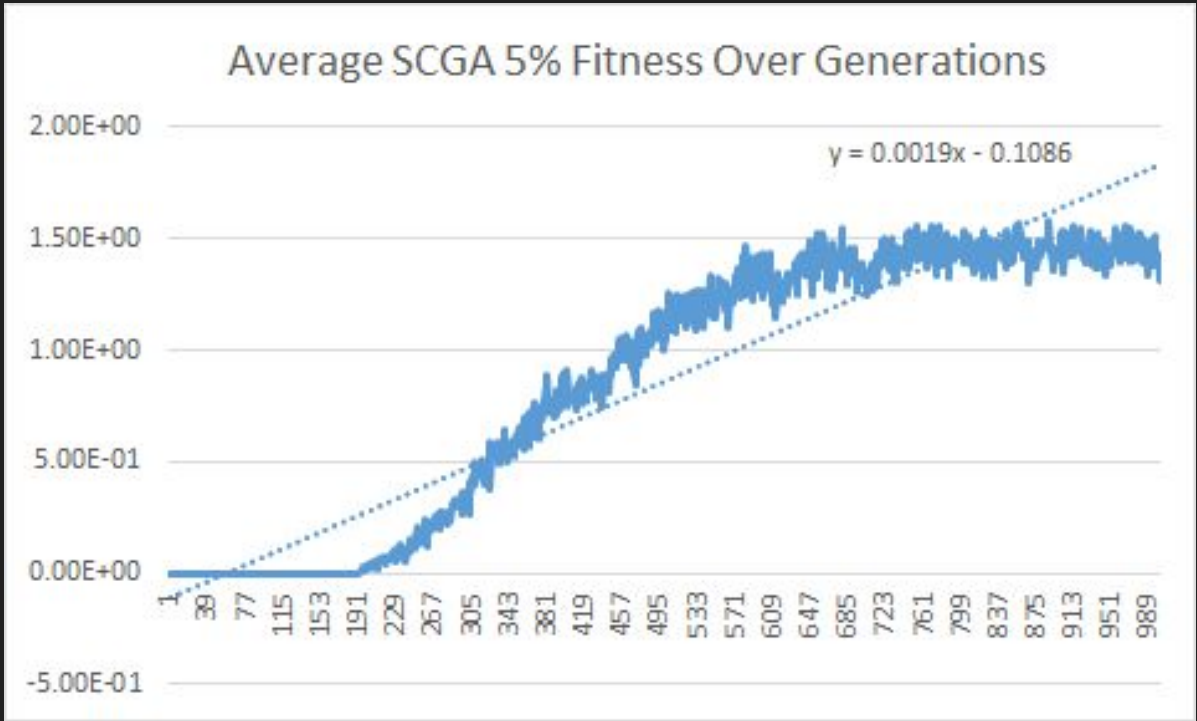
End

Any Questions?

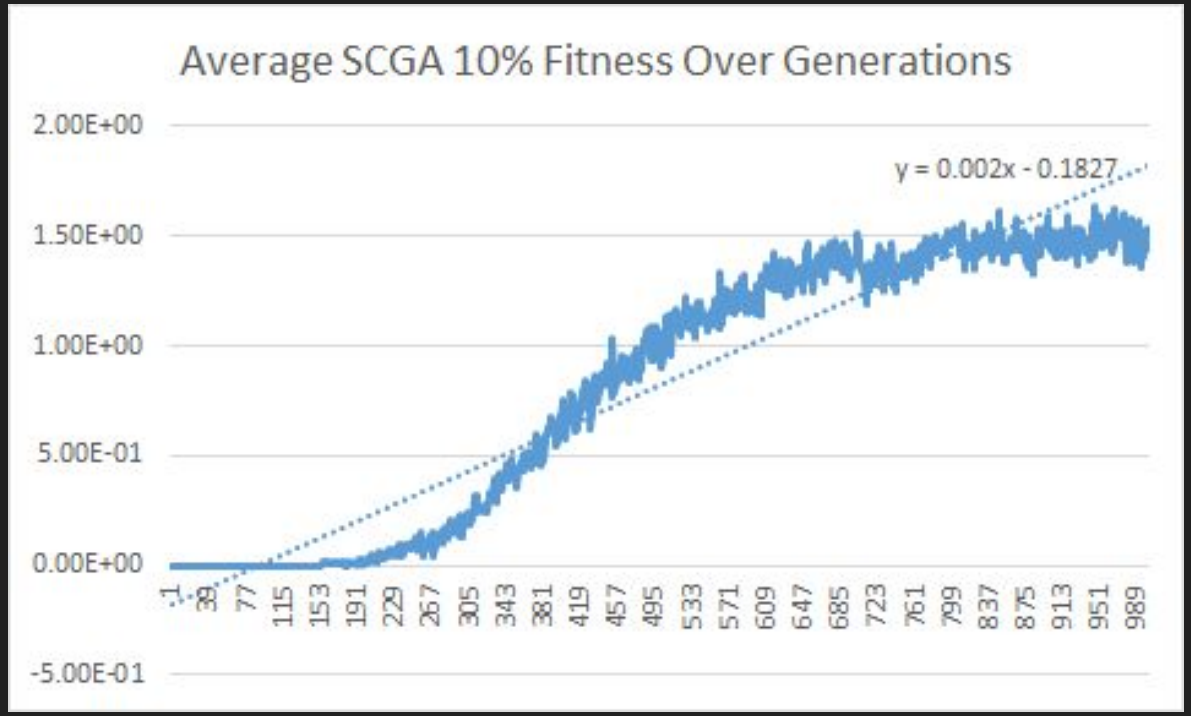
Extra


Average GA Fitness Over Generations
y = 0.0018x + 0.0017
2.00E+00
1.80E+00
1.60E+00
1.40E+00
1.20E+00
1.00E+00
8.00E-01
6.00E-01
4.00E-01
2.00E-01
0.00E+00
1
39
77
115
153
191
229
267
305
343
381
419
457
495
533
571
609
647
685
723
761
799
837
875
913
951
989

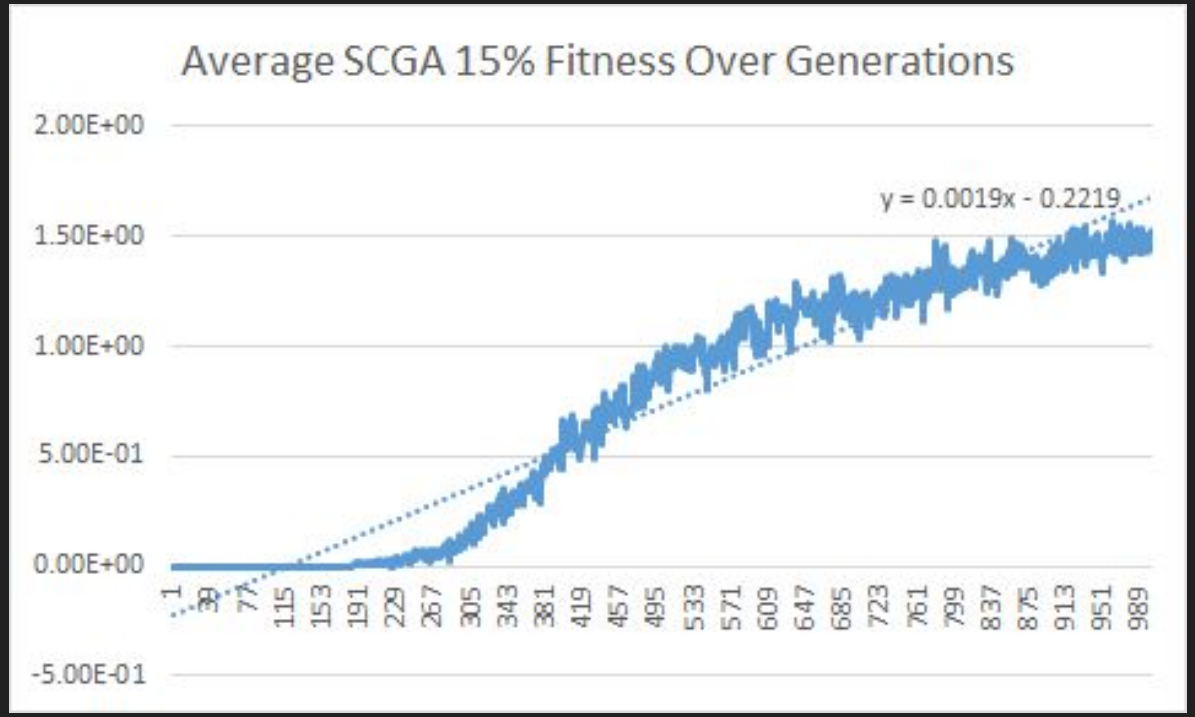
Extra Continued...



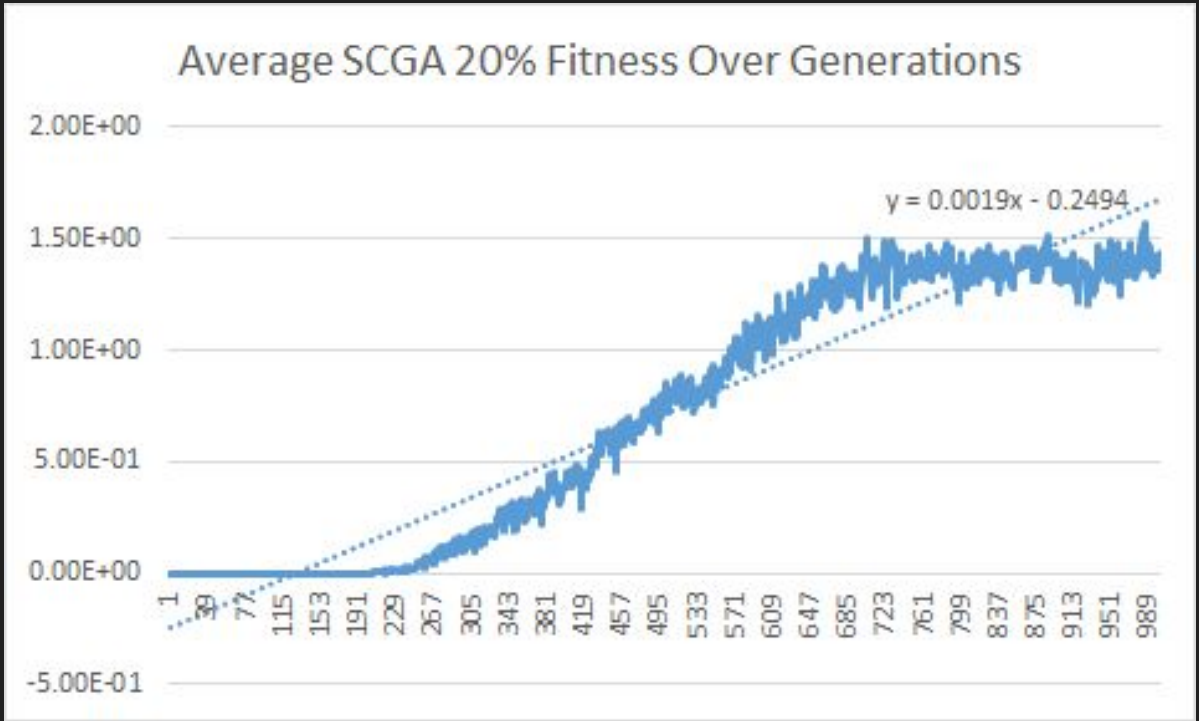
Extras Continued...



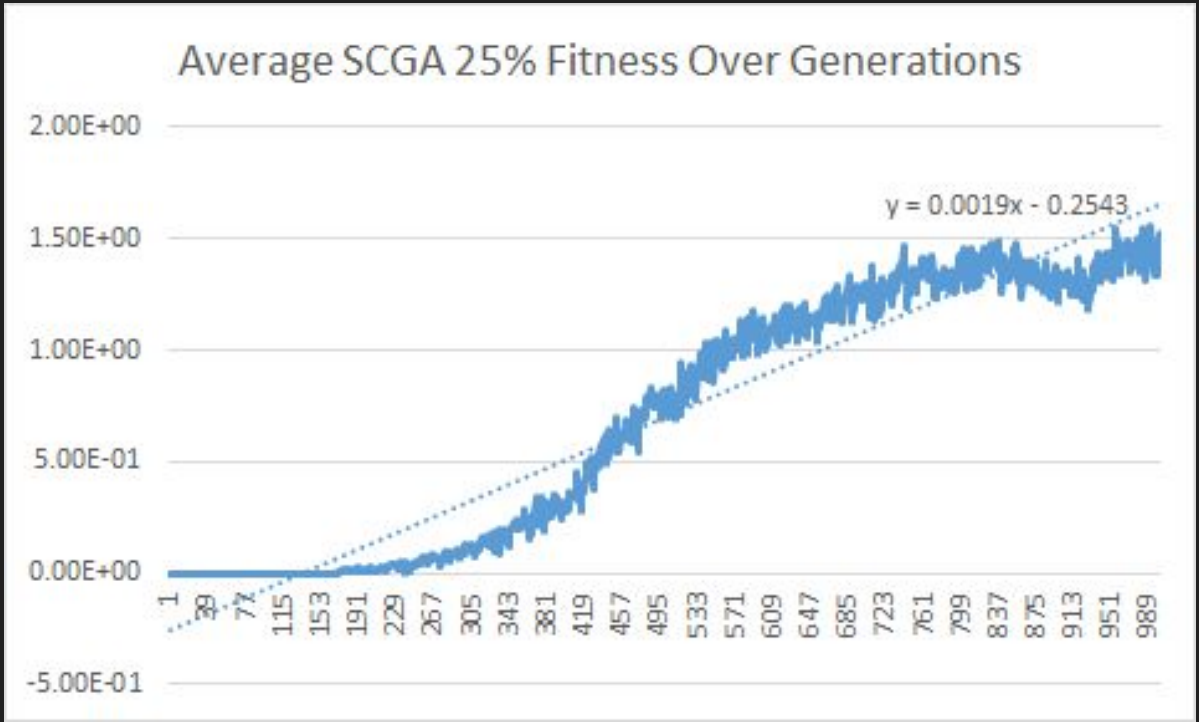
Extras Continued...



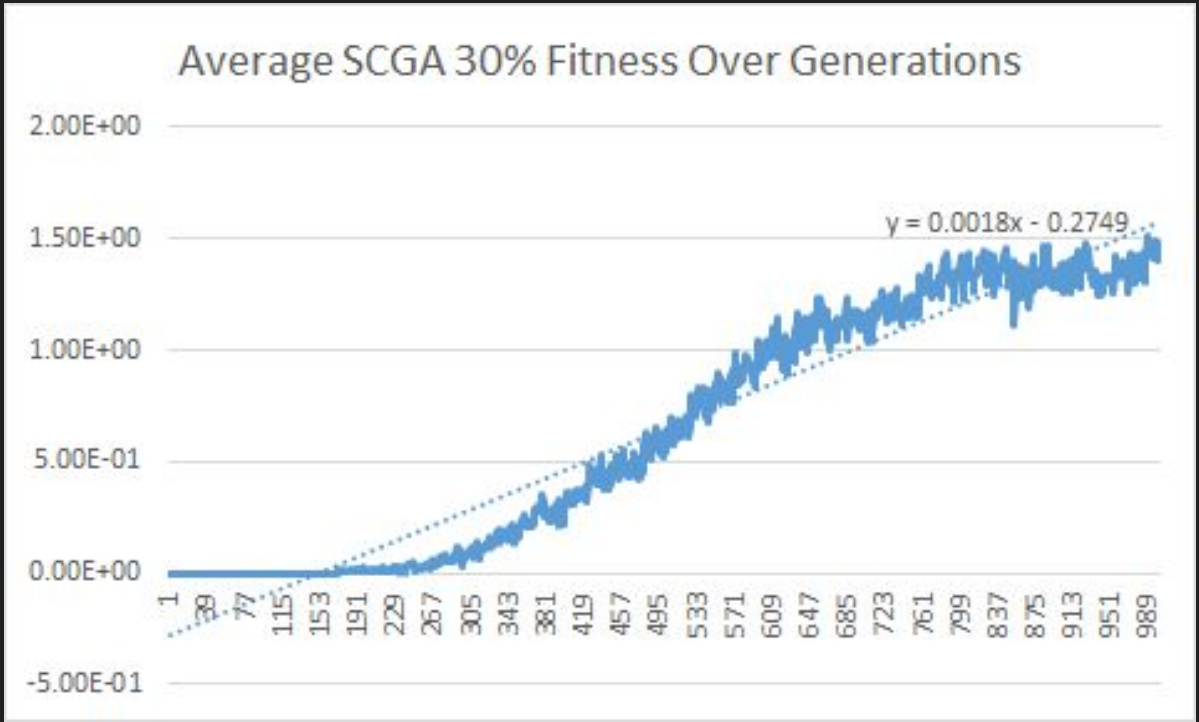
Extras Continued...



Extras Continued...



Extras Continued...



Extras Continued...

Showing the shortest path found...