Effectiveness Of Second Chance Mating In Genetic Algorithms (Applied to Pathfinding)

Gregory Whitman, Dr. Armstrong
Gwhitman55@gmail.com, Piffle@cs.ship.edu

Abstract:

This paper contains information about Second Chance Mating in genetic algorithms applied to pathfinding and its effectiveness. The experiment found that Second Chance Mating was statistically significantly slower than a genetic algorithm without Second Chance Mating implemented.

1. Introduction:

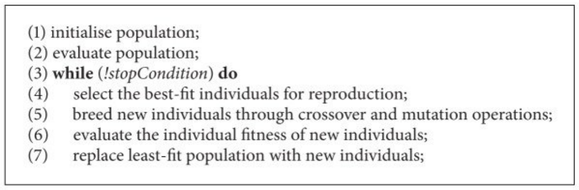
Genetic algorithms are used for special types of problems. They are used when a problem becomes computationally expensive and an approximate solution can be used. A good example of a problem that genetic algorithms could be applied to is the Traveling Salesman Problem. Genetic algorithms were first introduced back in the 1960s by John Holland and were based on Darwin's Theory of Evolution.

2. Literature Background

Below is information about genetic algorithms and how they were implemented for pathfinding in this experiment. Also there is information about Second Chance Mating.

2.1 What A Genetic Algorithm Is And How They Are Implemented:

Genetic algorithms are search algorithms that jump around through the possible solution set of the problem and test each solution and give it a “fitness” which is a rating of how “good” the solution is. The higher the fitness is the better the solution is at solving the problem [6]. Before the genetic algorithm can be made there must be a way of encoding the genome. Which will be discussed later.



```
(1) initialise population;
(2) evaluate population;
(3) while (!stopCondition) do
(4)     select the best-fit individuals for reproduction;
(5)     breed new individuals through crossover and mutation operations;
(6)     evaluate the individual fitness of new individuals;
(7)     replace least-fit population with new individuals;
```

Figure 1: This is an example pseudocode for a genetic algorithm.

2.3 Initialize Population:

The first step of a genetic algorithm is to initialize the first population. A population is a group of individuals which contain a random range of possible solutions from the solution set [8]. A population's size often depends on the type of problem being solved. Sometimes the size of the initial population has an effect on the accuracy and the speed in which the genetic algorithm finds a solution.

2.4 Evaluate Fitness/Fitness Function:

The next step of the genetic algorithm is to evaluate the population. This is done using a fitness function. Each individual in the population is tested in the fitness function and assigned a fitness score. The higher a fitness score is the better that individual was solving the problem [4]. A fitness function is one of the most important parts of the genetic algorithm. If the fitness function is wrong then the genetic algorithm will have a hard time succeeding or will flat out fail. The fitness function is also one of the hardest parts of a genetic algorithm because for every problem the fitness function will be different.

2.5 Selection:

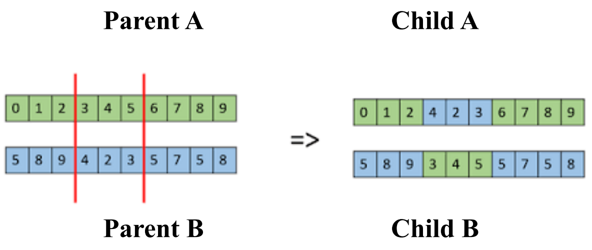
After each individual in the population has been tested and given a fitness score. Next is selecting individuals for mating from the population. This may seem like a random process but it is not entirely. The higher an individual's fitness score is, the more likely they are to be chosen for mating. This ensures that the better, more fit individuals pass on their better solutions. This is modeled after Darwin's Theory in which the fittest survive [6].

The method used for selection involves normalizing the fitness scores of each of the individuals. Normalizing means taking the fitness scores and cramming them between two set values like 0-1. Then a random individual in the population is selected to look at their fitness score. Next a random number between 0-1 is generated. If the individual's fitness score from earlier is greater than or equal to the random number then that individual is selected for crossover. If the Individual's fitness score is less than the randomly generated number then the process will be repeated until an individual gets selected. This process will repeat two times in order to get parent A and parent B for crossover. Also the parents can not be the same.

2.6 Crossover:

The next step is the actual mating of Parent A, and Parent B. The method that does this is called crossover. The crossover method just like in actual biology takes parts of both the parents' solutions and mixes them together in some form to create a new individual called a child [4], [5], [7]. There are many ways in which to cross over the two parents' solutions. This is an example of how it is done.

Two Point Crossover:



Two point mutation takes two random points in the parents solution and mixes them together to create the children. A good question is what to do with the children? Since there is a way to create children using the previous population it is time to create a new population, and this is done by repeating the selection and mating methods until there is a new full population of individuals with better, more fit solutions [4]. Ten fixed cut points were used in both genetic algorithms for the experiment.

2.7 Mutation:

During the crossover period there is something called mutation that takes place. Mutation just like in real life has a random chance of occuring [7]. In genetic algorithms, there is a small probability that at a single point in the solution it could change. This is a basic example of mutation.

```
Ex:     Child before mutation 1 1 1 1 1 1 1 1
                                ↑     ↑
                          Mutation occurs here

        Child after mutation  1 0 1 1 1 0 1 1
                                ↑     ↑
        The bits were flipped in these locations
```

2.8 Generating Next Population:

After having a brand new population also known as a new generation. This generation after crossover and mutation will have the best traits of the old populations solutions and variation from the parents due to mutations. This will hopefully lead to better and more fit individuals in this population [4].

The Fitness Function, Selection, and Crossover/Mutation are repeated creating new generations and eventually the genetic algorithm will be better and more optimal solutions until eventually little advancement is made. The fittest individuals in that final generation are close to the optimal solution and are considered complete.

2.9 Variables that affect a Genetic Algorithm:

Now it's time to talk about what factors go into affecting the speed of the algorithm and how valuable the solution it finds will be. The first variable is Population size. The size of the population affects the value of the final solution. If the population size is too big the solution at the end may not be as optimal as it could be. And if the population size is too small there may not be enough variation meaning the genetic algorithm will never make any progress. The next factor is the mutation rate. Too high of a mutation rate will cause the genetic algorithm to converge on a less optimal solution because it will be jumping around in the solution set too much. And too low of a mutation rate will cause the population to plateau and not make any improvements. Also the method of crossover will have an effect on the genetic algorithm [6], [7].

2.10 Genetic Algorithm applied to pathfinding:

A genetic algorithm will be used in order to find the optimal path in a pathfinding problem. The agents will have to travel across a 2D plane and reach a goal. At first the agents will move around randomly. But as generations pass the agents will start to get closer to the goal and find a better path due to the genetic algorithm.

2.11 Encoding of the Chromosomes:

To create the individuals for the genetic algorithm in pathfinding the chromosomes will look like this below.

Individual A

DNA: “16253”

Key:		
	1 = North	5 = Northeast
	2 = East	6 = Southeast
	3 = South	7 = Southwest
	4 = West	8 = Northwest

Translation for solution: North, Southeast, East, Northeast, and South

This will allow the agents to move around on the screen by updating their (x,y) coordinates according to the direction in the chromosome [1], [2], [3].

2.12 Fitness Function:

To calculate the fitness of the agents the genetic algorithm will use the Manhattan distance, which is the distance between two points.

Ex:

Individual (x_1, y_1)

Goal (x_2, y_2)

$$Distance = |x_1 - x_2| + |y_1 - y_2|$$

Before reaching the goal the agent's fitness score will be calculated as such.

$$Fitness = 1/((Distance * Distance) + 1)$$

If the agent reaches the goal the agent's fitness score will be modified to be calculated as such.

$$Fitness = 1000000/(Moves * Moves) + 1$$

This is done so that after the goal is found the agents will begin focusing on making the original path found more optimal. The fitness scores of the agents who reach the goal must be much higher than the agents who do not reach the goal. This is because the agents with a higher fitness will be more likely to get selected for crossover.

2.13 The playing field:

The agents will be moving around on a 2D field and if they touch the outer wall of the field they will stop moving and their fitness will be calculated from that point on.

2.14 Second Chance Mating:

Second chance mating is taking the most fit individuals from the previous generation and saving slots for them and directly inserting them into the next generation. There will also be children generated in the same method mentioned above.

3. Primary Objective:

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

Governing Propositions:

- The task will be a simple pathfinding problem.

1.5 person weeks over 1 semester

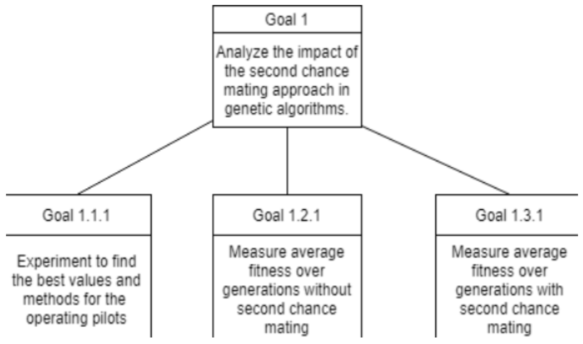
4. Hypothesis:

H_o: There will be no significant difference between second chance mating in a genetic algorithm than without.

H_a: There will be a significant difference in a genetic algorithm using second chance mating.

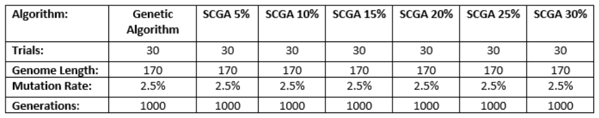
Why: The more fit individuals DNA from the previous generation is not just thrown away. They have a second chance to pass on their DNA.

5. 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

6. Experiment Design:

Data will be collected on the following experiments 5% , 10%...30% carry over rates for second chance mating. As well as data for the genetic algorithm without second chance mating implemented. Each experiment will be run 30 times in order to have enough samples for statistical analysis.



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

7. Solution Description:

Tools: Java, Eclipse IDE

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

8. Results:

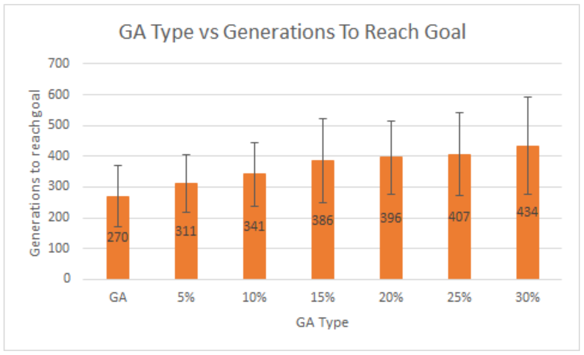


Figure 2: This is the data showing the average number of generations each experiment took to first reach the goal.

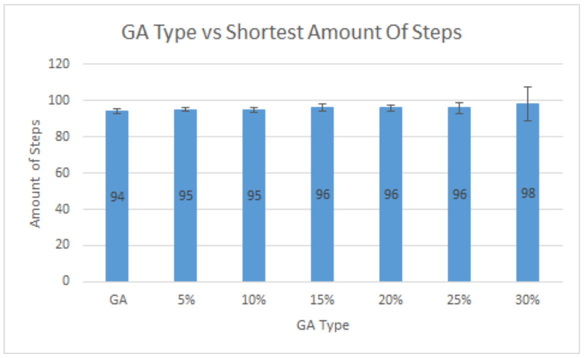


Figure 3: This is the data showing the average shortest path found in each experiment.

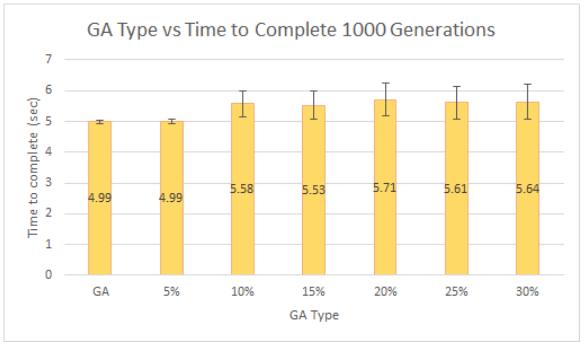


Figure 4: This is the data showing the average time in seconds each experiment took.

9. Conclusion:

Based on the results of the statistical analysis the genetic algorithm using second chance mating performed worse than the genetic algorithm that did not use second chance mating. Except for the case where the carry over rate was 5%. There it was found that there is no statistically significant difference.

10. Future Work:

Experiment with carry over rates less than 5% and look to see what those results will be.

References:

[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.