Parameter Optimization in Genetic Algorithmns: Niching vs Non-Niching

Michael Encinas
Computer Engineering and Computer Science
Speed School of Engineering
University of Louisville, USA
MAENCI01@louisville.edu

***Abstract*— This project is going to investigate the impact that parameters have in tuning a genetic algorithm under which there is niching or no niching. This will entice digging deeper into different benchmark fitness functions as well as a variety of different mutation and crossover rates to determine algorithm performance. I will test and analyze each algorithms fitness and effectiveness in solving its optimization problems.**

Keywords—Genetic Algorithm, Niching, Sharing, Crowding, Fitness Functions, Mutation, Crossover

I. Introduction

One of the biggest challenges into today's world is being able to solve complex optimization problems. A modern heuristic algorithm that has been used to solve complex problems is the algorithm known as Genetic Algorithm (GA) which derives from the idea of an evolutional process. GA are well known for being able to explore large and complex spaces by improving solutions thought operations such as mutations, crossovers, and selection. In my study I will create genetic algorithms with unique parameters and measure their fitness over generations as well as initial and final population over their landscape. I will implement different fitness benchmarks and bring in niching vs no niching to see if I am able to effectively optimize my problems.

II. Literature Review

A. Genetic Algorithm

Genetic Algorithms (GA) represent a powerful optimization technique that is used in various fields. GA is rooted in evolutionary principles in which the process starts by evolving an initial population near-optimal configuration. Another way to understand GA is the statement published by Katoch, Chauhan, and Kumar which states GA mimics the Darwinian theory of survival of the fittest.[5] Every individual in that population is a potential solution to the optimization problem at hand and their fitness is calculated by a fitness calculation that will show how well the problem objective was achieved through was calculations.

The core parameters and operators of GA include selection, crossover, and mutations. During the initial selection phase, individuals are chosen based on their fitness values. Crossovers involve combining genetic information from two parent individuals to create offspring while mutations introduce small random changes to individual solutions which creates diversity within the population.

As the GA progresses through multiple generations, the population will continue to evolve, and the quality of the solutions tends to keep improving. In a study by Liu, Gao, and Yang Liu they conclude that function is determined according to the optimization goal of the problem and that higher fitness correlates with better outcomes. [6]

Termination criterion is defined usually by several generations or by meeting certain fitness thresholds. GA has demonstrated success in solving complex optimization problems across various fields and domains.

Figure 2: Below is a graph illustrating the GA process [7]

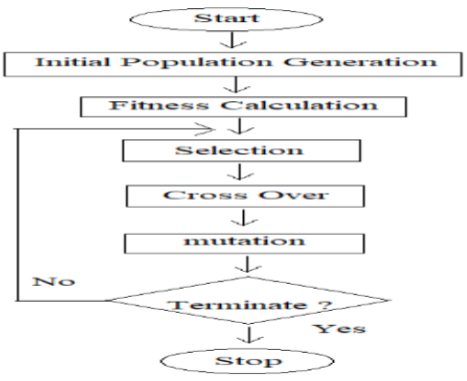
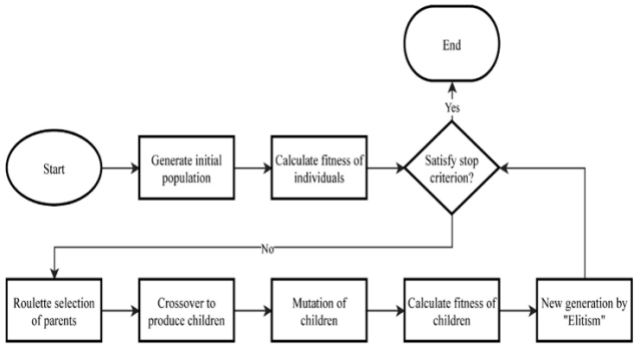


Figure 3: Table illustrating parameters of GA.[6]

Name	*variable name and value*
Population	POP_SIZE
Cross (Infection) probability	CROSS_RATE(INFECT_RATE)
Mutation probability	MUTATE_RATE
Epochs	N_GENERATION
All population	$[POP_1, POP_2 \ldots, POP_{POP\_SIZE}]$ $POP_i = [POP_i^1, POP_i^1, \ldots, POP_i^{34}]$

From the illustration below the authors show the process of genetic algorithm. The authors of that illustration further stated how GA is extremely dependent on crossover and mutation strategies in reference to the final outcome of any optimization goals. [8]

Figure 4



B. Niching and benchmarks function

Genetic Algorithm even if effective still sometimes don't always optimize solutions if they prematurely converge. Over the past years, researchers have developed algorithms known as Niche Genetic Algorithm (NGA). NGA have been able to get rid of genetic drifts in their evolutionally computation which happens to due populations converging on a higher fitness quickly and uniformity which causes premature converge. [9].

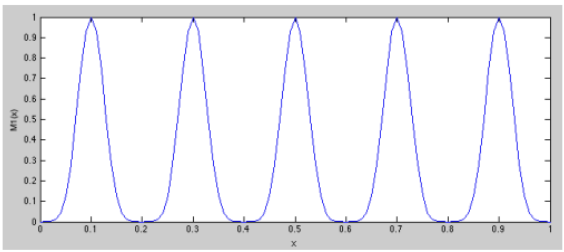
Two niching techniques are sharing and combined. Fitness sharing is the technique that reduces the fitness of individuals if the individuals are too close to each other in their population. The reason why this can be important is because it helps reduce clustering in our algorithm and helps the algorithm spread out which might get out of local optimums as well as add diversity.

Crowding can focus more on trying to replace similar individuals in its population to try to add diversity. For example, if you have an individual added to a population it will fight with similar individuals in the population and replace it which will help form optimal subpopulation groups.

Another way to measure these techniques are using benchmark functions. I will first be using x ^ 2 as my original function but then I will be using these two functions below as M1 and M4.

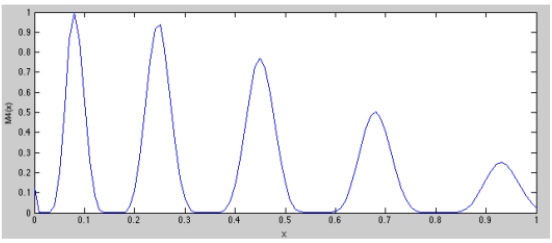
Figure 5 and Figure 6 illustrate the fitness calculation and the image its displayed in landscape.

Figure 5



$$M1(x) = \sin^6(5\pi x)$$

Figure 6



$$M4(x) = e^{-2(\ln 2)(\frac{x-0.08}{0.854})^2} \sin^6(5\pi[x^{0.75} - .05])$$

The functions above in M1 and M4 will help me obtain my fitness results. M1 creates multiple peaks (5) and can help avoid premature convergence to explore more search space.

M4 is a hybrid function that creates a sharp peak in the beginning, then over time starts to gradually go down which creates an oscillation effect. This can balance exploration and exploitation.

C. Mutations and Crossover

Mutations and Crossover are two important parameters that I will be varying for my genetic algorithm.

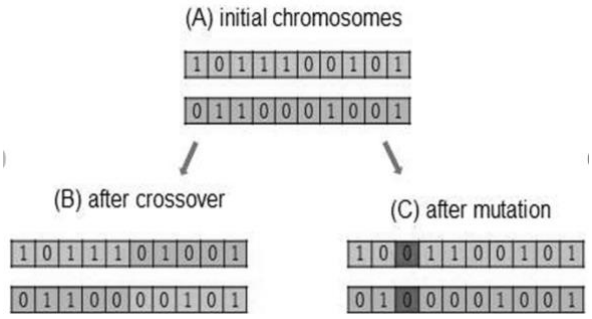
In my example, I am using 10-bit strings to represent individuals, which will act as chromosomes in the genetic algorithm. I will be applying single-point crossover**,** where two parent chromosomes exchange half of their bits to produce offspring.

For instance, if the crossover probability is set to 100%, the parents will always exchange bits, however, if it is set to 20%, there is only a 20% chance of crossover occurring. Figure 7 illustrates this process.

For mutation, it involves introducing random changes to individual bits in the chromosome, which represents a mutated gene. Figure 7 also demonstrates how mutation operates. For example, if the mutation probability is set to 10%, each bit in the chromosome has a 10% chance of mutating, which will add diversity to the population and helping the algorithm avoid premature convergence.

Figure 7 illustrate below showing single-point crossover and mutation.

Figure 7



III. Proposed Approach

When I set up my Genetic Algorithm, I have designed it to have three fitness functions to evaluate and test my performance.

1. $F(x) = X^2$
2. $M1(x) = (\text{Sin}^6(5\pi x))$
3. $M4(x)=e-2(\ln 2)(0.854x-0.08)2\cdot\sin 6(5\pi(x0.75-0.05))$

These fitness functions vary in terms of their fitness ranges. For F(x) it will have a range of 1-50 with its output being X^2. For M1 and M4 it will range from 0 to 1 for X.

I will use bits (10) to represent chromosomes for my individuals. The bits will be decoded to represent integer numbers for potential solutions.

A. Initial Parameters

I set my initial parameters below. However, DEFAULT_MUT_RATE and DEFAULT_CROSS_RATE are placeholders and will be updated in the experiments.

```
POP_SIZE = 100
NUM_BITS = 10
GENERATIONS = 100
DEFAULT_MUT_RATE = 0.1
DEFAULT_CROSS_RATE = 0.8
```

B. Experimental Setup

I will evaluate my algorithm by varying mutation rates, crossover rates, and niching methods which is described below in my code.

```
mutation_rates = [0.2, 0.5, 0.8]
crossover_rates = [0.2, 0.5, 0.8]
niching_methods = [None, "sharing", "crowding"]
```

Each combination of these parameters will be tested. For the selection process, I will use a 50% selection rate for choosing individuals for the next generation.

C. Data Collection

After setting up my Genetic Algorithm, I evaluated its performance by running it 10 times with varying combination of mutation rates, crossover rates, and niching methods. During each run I tracked the minimum fitness, average fitness, and maximum fitness values across all generations.

Each run I stored the fitness values for all individuals in the population as well as the min fitness, average fitness, and max fitness for each generation. I was able to visualize these results as well as visualize the landscape of just one run for his benchmark with and without niching for comparison later on.

D. Visualization

For my visualization and code, I used Python while running Jupyter notebook I used the libraries; NumPy, Random, Math, and Matplotlib.

I plotted the fitness levels example over generations (average minimum fitness, average fitness, and average maximum fitness). The average fitness levels are running ten times but averaged out once for each generation

I visualized the fitness landscape for my regular F(x) function as well as my M1(x) and M4(x) functions. These landscapes will visualize the search space as well as show plots for initial and final population plots.

Figure 8 which illustrates fitness levels over generations.

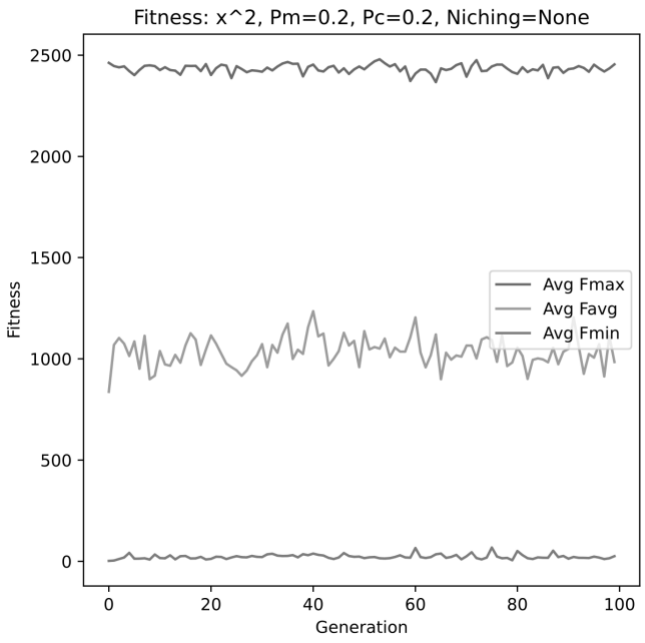
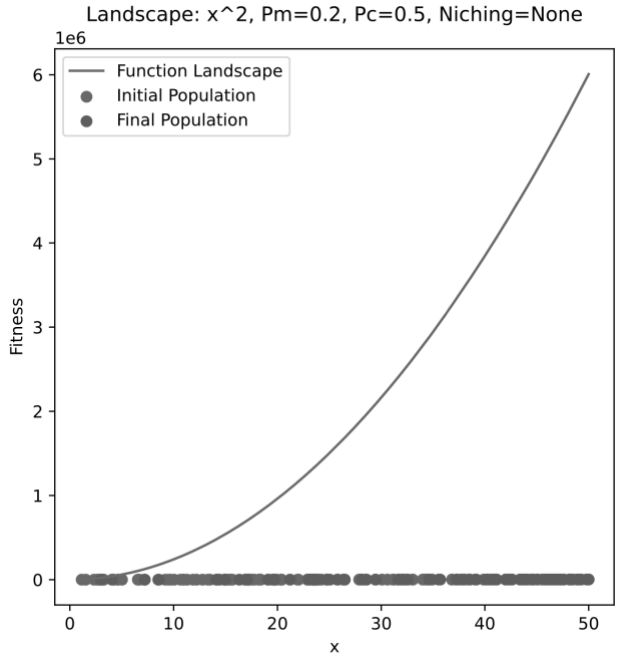


Figure 9 which illustrates landscape over functions



E.. Results and Analysis

I first be checking fitness levels with no niche before I address landscapes. For Fitness X^2 with no niche, when I have Mutation at .2 and crossover at .2, .5 or .8 my fitness levels stay high for maximum as well as for average fitness and lower fitness my fitness appears to do a lot better compared to making mutation go to .5. When mutation rate is at .5 or .8, I get way lower fitness scores compared to mutation rate at .2. I also observe that when mutation is low, and crossover rate is .2 to .8 then I get to maximum my fitness at high levels as well as get higher fitness for averages compared to when mutations are higher.

My landscape for Fitness function X^2 with no niche tends to show that final population is usually towards the middle to late part of my upper curve since I assume once you reach getting past the smaller numbers the fitness levels will rise drastically. Initial populations tend to stay in the beginning or middle.

For niching with sharing on the same fitness function above I observe that for the best parameters which is mutation rate at .8 with crossover rate at .2. It appears upon observation even if fitness levels are not as high for max which ranges from 400 – 1400 there seems to be a range of optimal spikes. This appears to add more diversity to my population.

Mutation plays a bigger factor in niching sharing than no niching because when mutation is set to .2 or .5, I do get lower fitness levels compared to mutation rate set higher at .8. I also notice that my initial population and final populations plots for my landscape is more spread out equally which demonstrates the ability to get out of local optimums to add diversity.

When looking at crowding fitness results, for my mutation rates and crossover rates they now achieve back to high fitness results at 2500. However, there is a lot more spikes compared to just no niching. When crossover rate is lower at .5 and .2, I get a lot more spikes and at .8 rate it starts to spike a lot less and look more like no niching. Crossover rates play a bigger factor in crowding compared to the other niche.

I will select crowding when crossover rate is lower because the spikes can help get out of local optimums and show more subpopulations being created. This is a good balance I feel towards having high fitness and being able to go up and down in spikes to get out of local optimums. For landscape I get a lot of initial population points in the beginning, but I do have a little more of final population points in the beginning and middle compared to no niching.

I will now be looking at M1 Benchmark. For M1 benchmark with no niching when crossover rate is higher with .5 or .8 rates then I get a lot more final population peaks at the top of the five high peaks in my landscape. When it comes to fitness rates with no niching, I get a lot more spikes up and down for average fitness average and I stay consistent at high and low. It reminds me of crowding on fitness levels.

For the peaks I get initial population and final population both getting to upper peaks equally on all five peaks which shows that M1 can get high fitness and have diverse population out of local optimal.

For sharing niche, I had a hard time picking the best optimization parameters since my fitness levels were all low and even if they were higher than the others on the landscape at peak they were unable to get a good diverse spread of initial and final population plots. I found that when Mutation rate is raised to .5 and .8 I get lower fitness rates but my peaks on landscape are very diverse on initial and final populations.

For crowding niche with M1 benchmark, when mutation rate is .2 and crossover rate is .2, I get stuck in local optimums. However, with fitness I stay higher than sharing niche, in comparison the results were as no niching for M1 which shares with x^2 results.

Crowing when crossover rate is high, I tend to get more optimization on my landscape reports. With initial and final populations tended to be very nicely spread all around and at all 5 peaks. M1 with crowding showed best results when mutation rates were high at .8 and crossover rate was .8. I was able to get a good balance of average fitness and minimum fitness in spiking up and down and for my landscape optimization visualization I was able to get a good spread of plots for my optimization becoming more global.

For my M4 benchmark I will continue to repeat the process that I have done for M1. For M4 without niching, crossover rates are a big factor for stabilizing fitness levels as well as being able to reach all peaks for landscape for final and initial population. This algorithm seems to have the right balance of high and low as well as capture fitness consistently. When mutation rate is high, I do not seem to have good plots for my landscape visualization and my spikes tend to be more unstable in my fitness ranges.

For M4 with sharing niching, I get optimal results when mutation rate is low at .2 and crossover rate Is high at .8. Even if my fitness results are lower than one and crowding which I will discuss later, the peaks from high to low are all reached at different levels. This adds greats plots around all subpopulations of the landscape of my optimization problem.

For M4 with Crowding Niche I was again get good results with mutation rate low at .2 and crossover rate at .8. I can get spikes up and down all fitness ranges which shows that I have a wide range of diversity in each generation. For my landscape I have a wide range of initial and final populations in all my peaks almost equally for M4.

IV. Conclusion

Based on my results, mutation and crossover really matter depending on the function. If you have X^2 for example mutation appears to matter more but on M1 it appears to balance it out with crossover, however, on M4 crossover tends to matter more than mutation. Niching played a big role, depending on your goals, sharing might prove to be effective, however, if you want a balance of fitness and having a diverse population such as sharing than crowding would be able to achieve this plus the addition of having a high fitness level.

I also observe that my normal function I created (X^2) really peaks high in the beginning unlike my M1 and M4 which are way more balanced. The niches can add more diversity to my data so even if my functions are not performing as well in diversity then my niches can add diversity to my population which can improve my generations from not getting stuck in local optimums.

Based on my results, I would like to add that functions play a huge role and so do niches. Parameters such as crossover and mutation really depend on what you are trying to achieve. I did not change my population size, generation size, or number of bits. I also kept my range 1 – 50 on my regular function which can also play a big role in fitness for that function.

If I was to increase population size and generations then maybe my results could improve or not improve. Future studies will most likely need to be conducted on this if it has not already.

REFERENCES

[1] J. Müller-Trede, S. Choshen-Hillel, M. Barneron, and I. Yaniv, "The Wisdom of Crowds in Matters of Taste," *Management Science*, vol. 64, no. 4, pp. 1779-1803doi: 10.1287/mnsc.2016.2660.

[2] Z. Da and X. Huang, "Harnessing the Wisdom of Crowds," *Management Science*, vol. 66, no. 5, pp. 1847-1867doi: 10.1287/mnsc.2019.3294.

[3] N. R. Burns, M. D. Lee, and D. Vickers, "Are individual differences in performance on perceptual and cognitive optimization problems determined by general intelligence?," *The Journal of Problem Solving*, vol. 1, no. 1, p. 3, 2006.

[4] S. K. M. Yi, M. Steyvers, M. D. Lee, and M. J. Dry, "The Wisdom of the Crowd in Combinatorial Problems," *Cognitive Science*, vol. 36, no. 3, pp. 452-470, 2012, doi: https://doi.org/10.1111/j.1551-6709.2011.01223.x.

[5] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091-8126, 2021, doi: 10.1007/s11042-020-10139-6.

[6] Y. Liu, Y. Gao, Y. Liu, R. Ieee 22nd International Conference on Software Quality, and C. D. D. Security Companion Guangzhou, "IMGA: Improved Microbial Genetic Algorithm," in *2022 IEEE 22nd International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*: IEEE, 2022, pp. 189-192.

[7] S. Sharma and V. Jain, "A Novel Approach for Solving TSP Problem Using Genetic Algorithm Problem," *IOP Conference Series: Materials Science and Engineering*, vol. 1116, no. 1, p. 012194, 2021, doi: 10.1088/1757-899x/1116/1/012194.

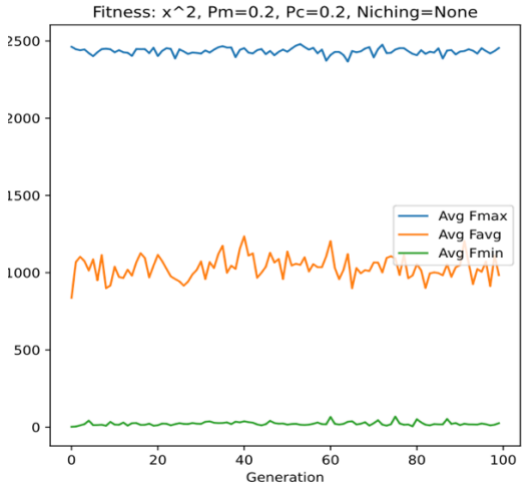
[8] D. Yang, Z. Yu, H. Yuan, and Y. Cui, "An improved genetic algorithm and its application in neural network adversarial attack," *PLOS ONE*, vol. 17, no. 5, p. e0267970, 2022, doi: 10.1371/journal.pone.0267970.

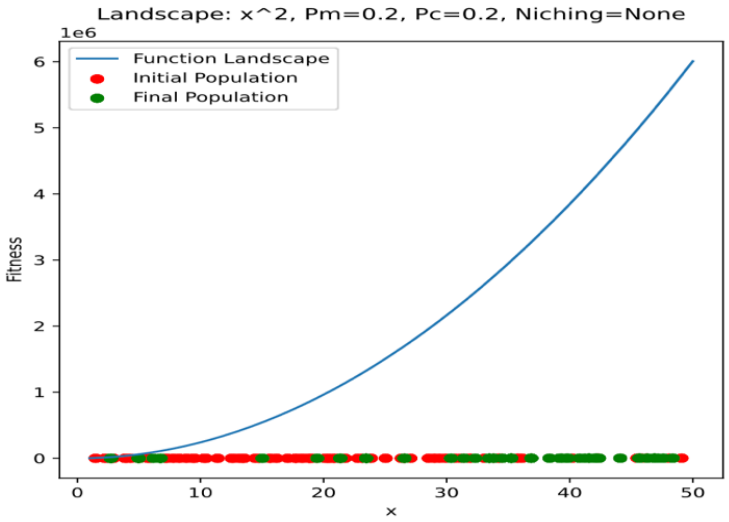
[9] L. Yuan, M. Liand J. Li, “Research on Diversity Measure of Niche Genetic Algorithm”, 2008. doi: 10.1109/wgec.2008.66.

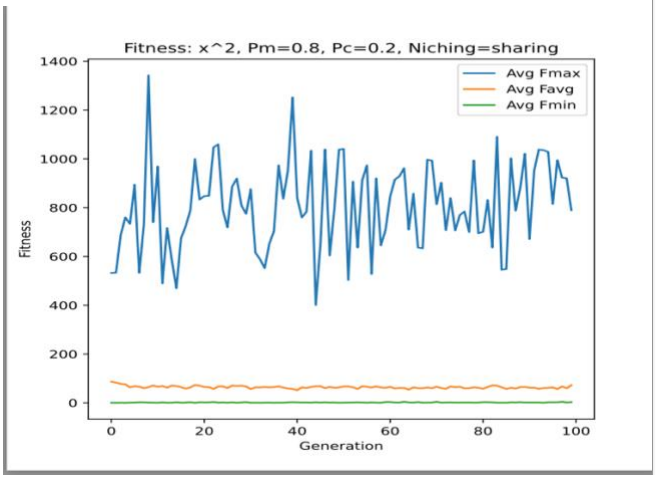
[10] B. Wutzl, K. Leibnitz, F. Rattay, M. Kronbichler, M. Murata, and S. Golaszewski, "Genetic algorithms for feature selection when classifying severe chronic disorders of consciousness," *PLOS ONE*, vol. 14, p. e0219683, 07/11 2019, doi: 10.1371/journal.pone.0219683

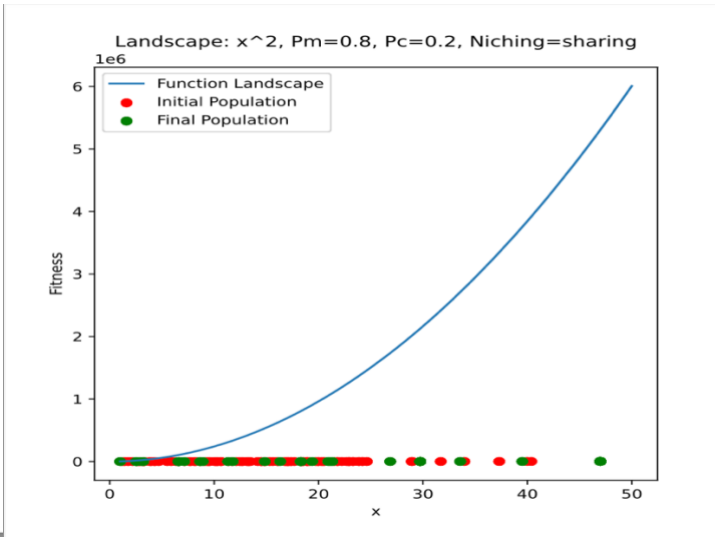
Best for specific functions with and without niching

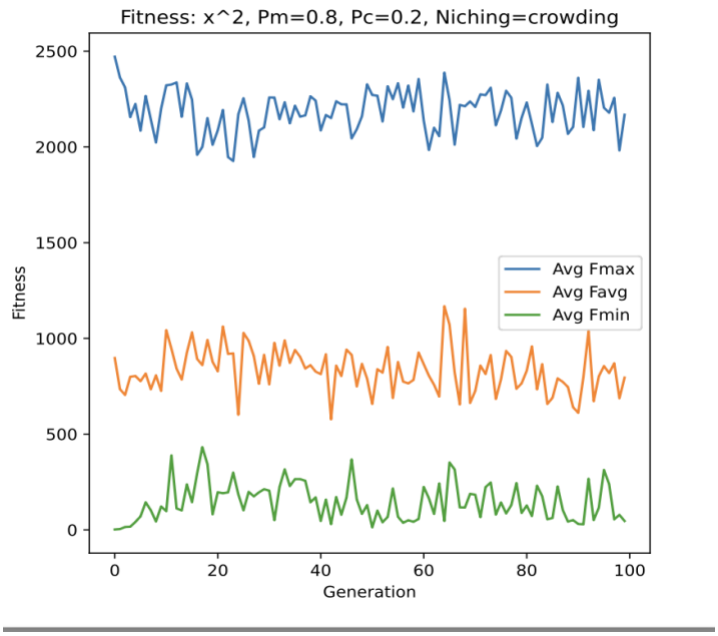
X^2

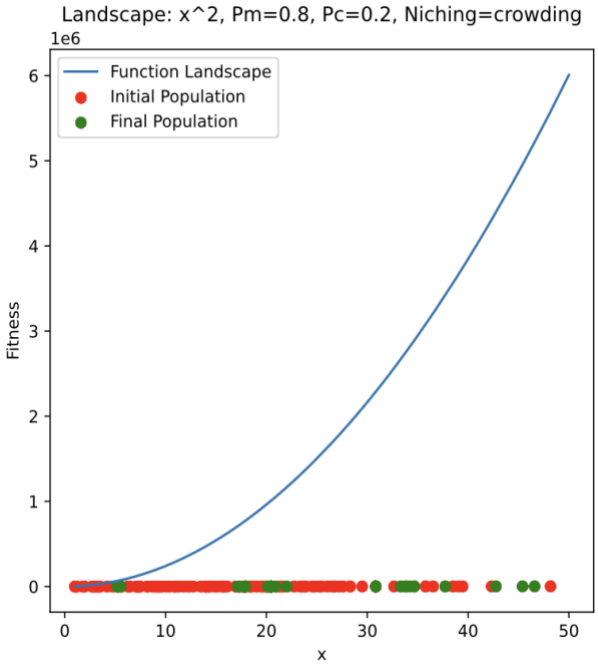




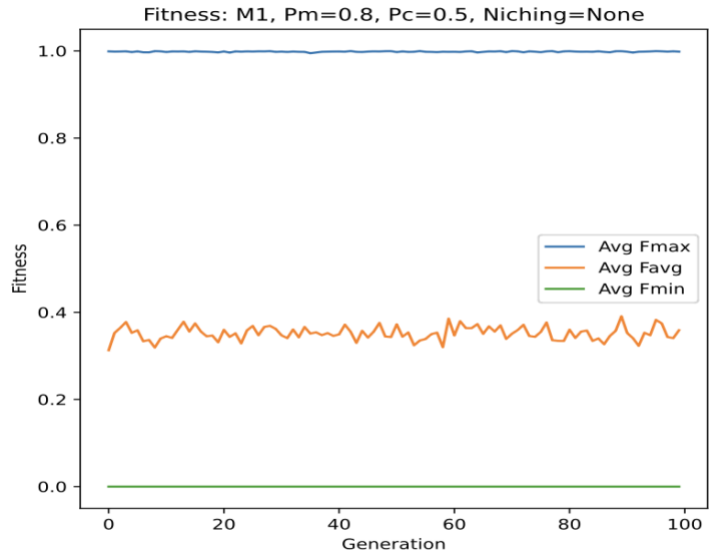


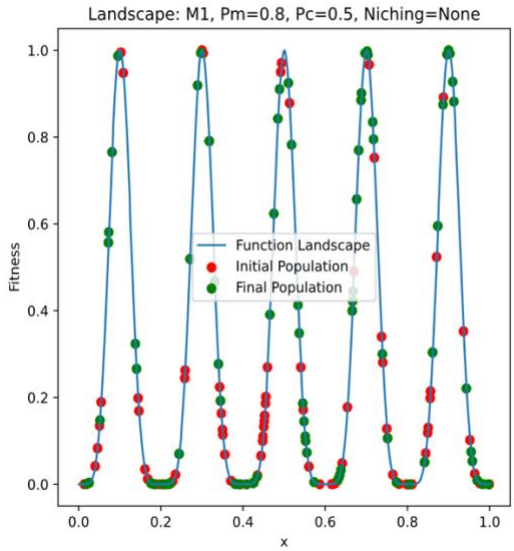


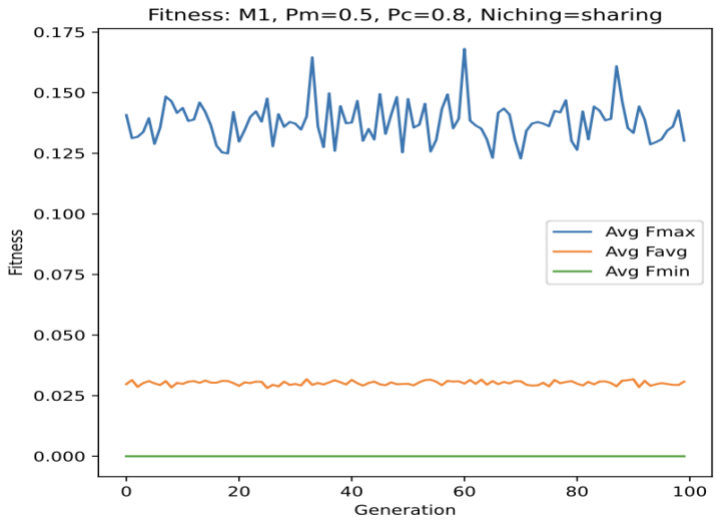


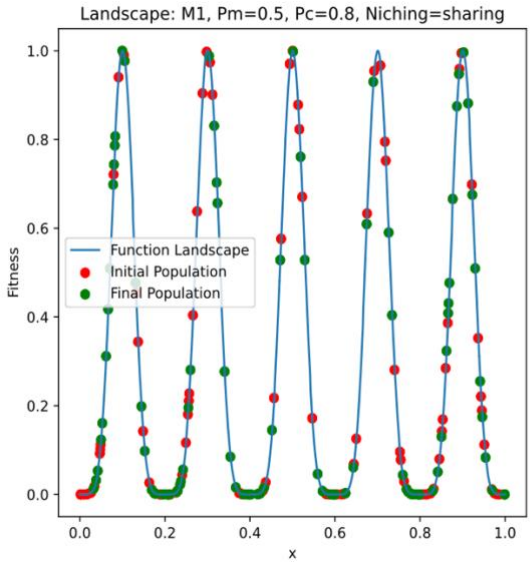


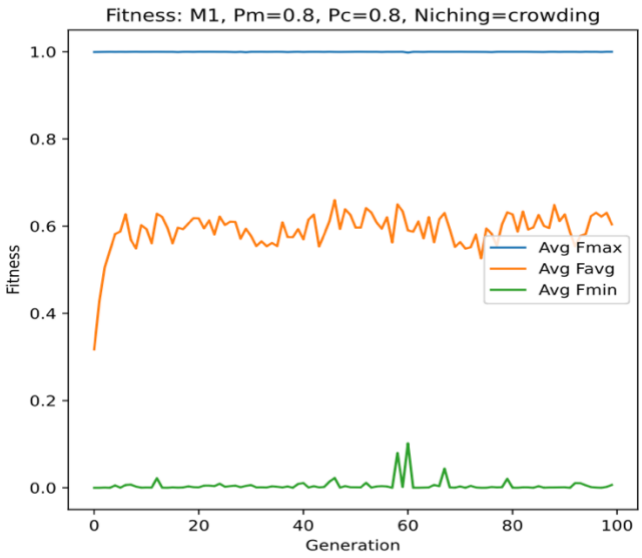
M1 Benchmark

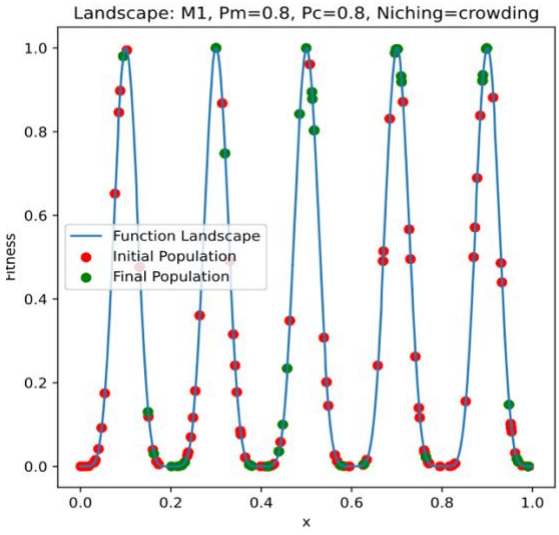




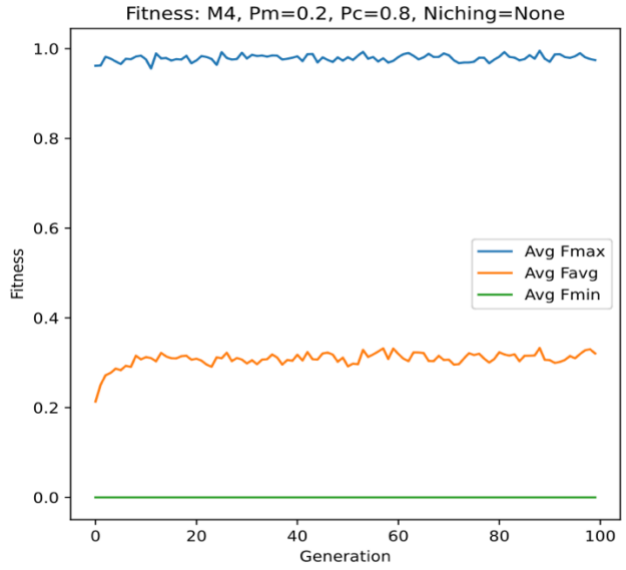


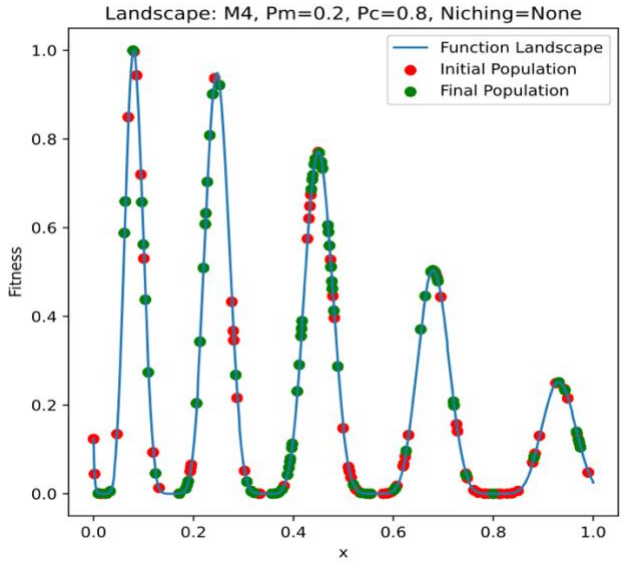


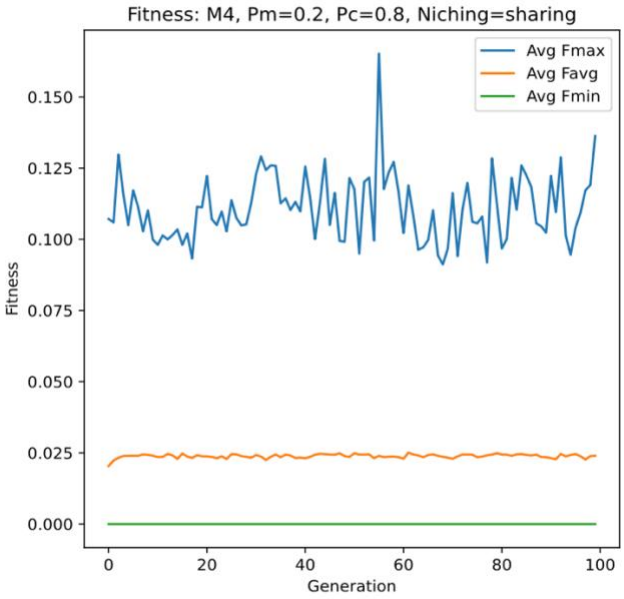


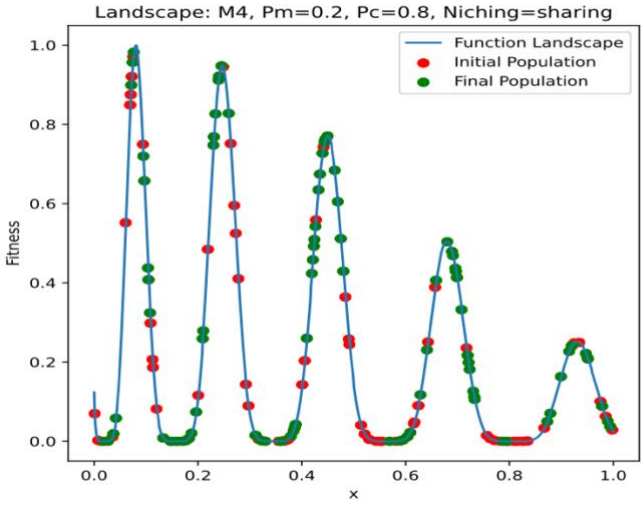


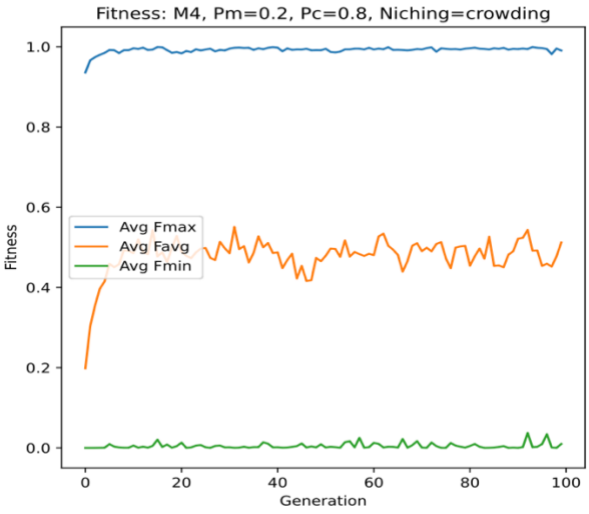
M4 Benchmark

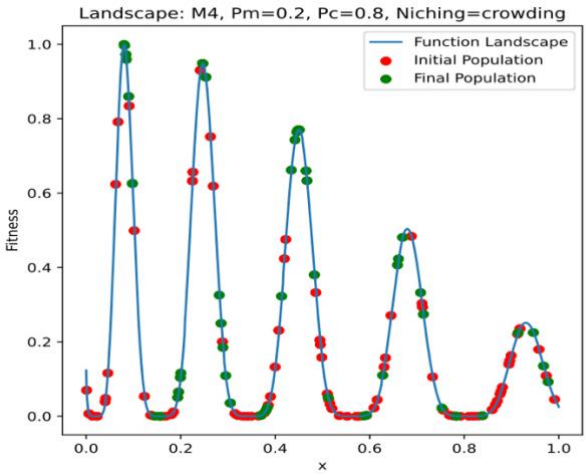

Fitness: M4, Pm=0.2, Pc=0.8, Niching=None
Avg Fmax
Avg Favg
Avg Fmin
Fitness
Generation


Landscape: M4, Pm=0.2, Pc=0.8, Niching=None
Function Landscape
Initial Population
Final Population
Fitness
x


Fitness: M4, Pm=0.2, Pc=0.8, Niching=sharing
Avg Fmax
Avg Favg
Avg Fmin
Fitness
Generation


Landscape: M4, Pm=0.2, Pc=0.8, Niching=sharing
Function Landscape
Initial Population
Final Population
Fitness
x


Fitness: M4, Pm=0.2, Pc=0.8, Niching=crowding
Avg Fmax
Avg Favg
Avg Fmin
Fitness
Generation
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


Landscape: M4, Pm=0.2, Pc=0.8, Niching=crowding
Function Landscape
Initial Population
Final Population
Fitness
x
0.0
0.2
0.4
0.6
0.8
1.0