Predator Prey Report
Assignment 2

Hridoy Rahman #7340334

March 25, 2024

1 Summary

The Predator-Prey problem aims to simulate one predator pursuing multiple prey instances within a map. The objective is to eat as much prey as possible within a given number of moves, and the amount of prey eaten by the predator is less than the total amount spawned on the map. In other words, the predator is to evolve a successful pursuit with a sequence of moves that maximizes the number of prey consumed by the predator. The evolutionary process to evolve the predator to pursue prey employs genetic programming with the ECJ (A Java-based Evolutionary Computation Research System) framework to solve this predator-prey problem. The GP language used in the process of solving the predator-prey problems are Function sets: **IfPreyAhead, IfPreyBehind, IfPreyLeft, IfPreyRight, Progn2, Progn3** and **Terminal Sets: Left, Right, Move**. Prey randomly moves from one location to another when a predator looks left or right or moves in a position. Advance prey movements is used such as moving in 12 direction rather than moving in 4 direction, will discuss on comparative analysis. The simulation environment is a 2-dimensional array where the predator starts off at a starting position and starts hunting the preys. The fitness for the GP program is to eat as much prey as possible within the given amount of moves and as long as the number of prey eaten is less than the total amount of prey spawned on the map. The GP problem is performed over 10 runs with different parameter settings and statistics analysis performed (provided later in the section). The outcome of the GP program is to mark the current predator's location, the location where the prey got eaten, and the prey that survived the predator's pursuit. Overall, the program performs well with a 90% crossover and 10% mutation rate with 10 elites carried over to the next generation.

Objective	The objective is to eat as much prey as possible within a given number of moves, and $preyEaten < totalPreyAmount$
Terminal Set	Left, Right, Move
Function Set	IfPreyAhead, IfPreyBehind, IfPreyLeft, IfPreyRight, Progn2, Progn3
Fitness Cases	TotalAmountOfPrey - (minus) theNumberOfPreyEaten
Raw Fitness	The number of prey eaten.
Standardized Fitness	The total amount of prey - (minus) raw fitness
Hits	number of hits are the same as raw fitness.
Success Predicate	An S-expressions that eats the maximum amount of preys.

Table 1: Table of Predator-Prey

The table summarizes the Predator-prey problem's Objective, Fitness cases, GP language, Crossover and Mutation rate and results.

2 Experiment Details

2.1 Table of GP Parameters

Here are the GP parameters used for the Predator-prey problem.

Parameter	Value
Population Size	1024
Generation Size	50
Moves	600
max-x	30
max-y	30
preyAmount	30
Selection Method	Tournament Selection
Tournament Size	3
Runs (Jobs)	10
Elites	10
Crossover & Mutation Rate (Variation 1 without Elitism)	90% & 10%
Crossover & Mutation Rate (Variation 2 without Elitism)	100% & 0%
Crossover & Mutation Rate (Variation 3 without Elitism)	0% & 100%
Crossover & Mutation Rate (Variation 4 with Elitism)	90% & 10%
Crossover & Mutation Rate (Variation 5 with Elitism)	100% & 0%
Crossover & Mutation Rate (Variation 6 with Elitism)	0% & 100%

Table 2: Table of Parameters Used

The following parameters were used to run the predator-prey GP problem. There are 1024 individuals in one generation, running until 50 generations on one job or run. In other words, each evolutionary run consists of 50 generations, where 1 generation has 1024 individuals. Each simulation lasts **600 moves**, and the number of prey eaten by the predator is less than the total prey amount. In this simulation, 30 prey are present on the map. The grid size of the map is denoted by **max-x and max-y**, where the grid size is **30x30**. The selection method used in this GP problem is tournament selection with a tournament size of **3 individuals**. **10 runs** Independent evolutionary runs were performed, and statistics were gathered from them. 6 different parameters were used to check for the best performance, where **variations 1-3** are tested without elites carried over to the next generations. **Variations 4-6** are tested with **10 elites** carried over to the next generations. The Crossover and mutation rates primarily used are **90% crossover and 10% mutation, 100% crossover with 0% mutation, and 100% mutation with 0% crossover**. The best results were accumulated from **90% crossover and 10% mutation with 10 elites**, denoted as variation 4 in the table.

2.2 Fitness Function

Fitness Cases: the total amount of prey - (minus) number of prey eaten
Raw Fitness: The number of prey eaten.
Standardized Fitness: The total amount of prey - (minus) raw fitness.

```
KozaFitness f = ((KozaFitness)ind.fitness);
f.setStandardizedFitness(state,(preyAmount - preyEaten));
f.hits = preyEaten;
ind.evaluated = true
```

The fitness function takes the number of prey amounts denoted in the parameter file of **predator.params**, and after running the evaluation function prey eaten counts are returned for each individual. Then it calculates the fitness results of the problem by `subtracting the total prey amount - the number of prey eaten by the predator.`

2.3 GP Language

Terminal Set: Left, Right, Move
Function Set: IfPreyAhead, IfPreyBehind, IfPreyLeft, IfPreyRight, Progn2, Progn3

Function Set:

IfPreyAhead incorporates forward vision to look for preys. The predator looks for preys based on its orientation. For example, if the predator is looking up then it will look for prey on the upper cell, if it's looking down then it will look for prey on the bottom cell, and so on.

IfPreyBehind incorporates behind vision to look for preys, where the predator looks for preys on the cell behind from where it's looking. This is achieved by incorporating orientation of the predator.

IfPreyLeft incorporates left vision to look for preys, where the predator looks for preys on the cell that is left to it from where it's looking

IfPreyRight incorporates right vision to look for preys, where the predator looks for preys on the cell that is right to it from where it's looking.

Terminal Sets:

Move incorporates the movement of the predator. As well as updates preys location upon predator movement.

Left changes the orientation of the predator, by looking left from its current locaiton.

Right changes the orientation of the predator, by looking right from its current location.

2.4 Data Format

- MAP - 2 dimensional
- Predator's current position - marked with "#" (Hash symbol)
- Location where preys are eaten - marked with "X" (Alphabet X)
- Location of surviving prey - marked with "p" (Alphabet small-p)
- Empty cells - marked with "." (Dot).
- Orientation of Predator's (Left, Right)

The GP problem is performed in a grid environment using a 2-dimensional array. Upon initializing the 2-D array, the map is filled with 0s. Then, with the given amount of prey, the prey is randomly allocated to the grid environment (map). Predator starts at position **0,0** and starts hunting prey throughout the map. The predator's trail is marked by #, where prey eaten by the predator is marked by **X** and the location of the survived prey is marked by **small-p**, and the empty cells are marked with **".(dot)"**.

2.5 Variations of Experiments

Crossover and Mutation Rate	Values
Crossover & Mutation Rate (Variation 1 without Elitism)	90% & 10%
Crossover & Mutation Rate (Variation 2 without Elitism)	100% & 0%
Crossover & Mutation Rate (Variation 3 without Elitism)	0% & 100%
Crossover & Mutation Rate (Variation 4 with Elitism)	90% & 10% with 10 elites
Crossover & Mutation Rate (Variation 5 with Elitism)	100% & 0% with 10 elites
Crossover & Mutation Rate (Variation 6 with Elitism)	0% & 100% with 10 elites

Table 3: Table of Parameters Used

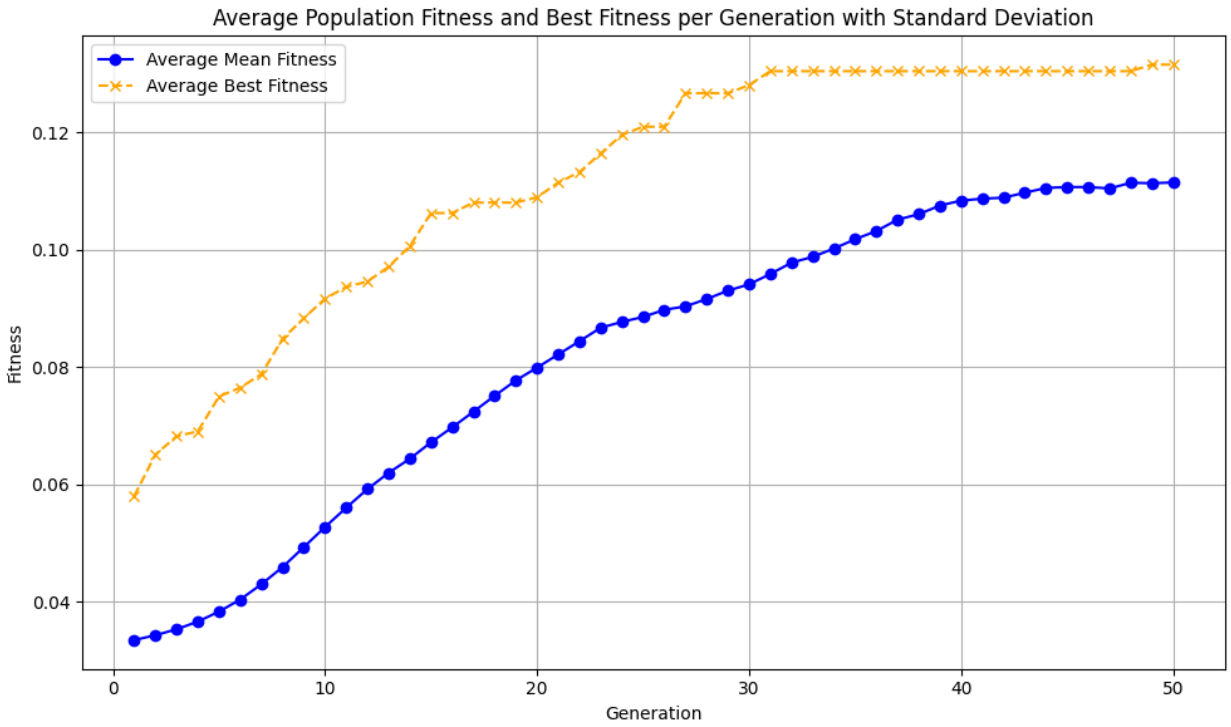
The following Table 3 depicts the parameters used to run this Predator-prey GP problem. Where the variation 4 is found as the best parameter runs across all the GP problem runs compared with other variations of parameters and each parameter variation ran 10 simulations with different seed values.

3 Results

3.1 Tables of Result

The following data table consists of parameter values used to run the GP problem (Predator-Prey). The variation that outperformed all the other variations is variation 4 with 90% crossover rate, and 10% mutation rate with 10 elites carried to the next generation.

The graph depicts the Average Mean Fitness and Average Best Fitness plotted across 10 runs. The blue line shows the average mean fitness of all individuals in the population, and the yellow line shows the average fitness of the best individual in the population across all



(a) (Variation 4) 90% & 10% with 10 Elites

generations. With a 90% crossover rate, a 10% mutation rate, and 10 elites, the predator is able to eat 27 prey out of 30 prey given in the grid environment. The standard deviation is not depicted in the graph, but the standard deviation of mean fitness and best fitness are shown in the **(Table 4)**. Table 4 shows the values of Average Mean fitness, Standard Deviation Mean Fitness, Average Best fitness, and Standard Deviation Best Fitness across all 10 runs for 50 generations for the best-performed parameters with crossover 90%, 10% mutation rates, and 10 elites. The rest of the tables are provided in the appendix.

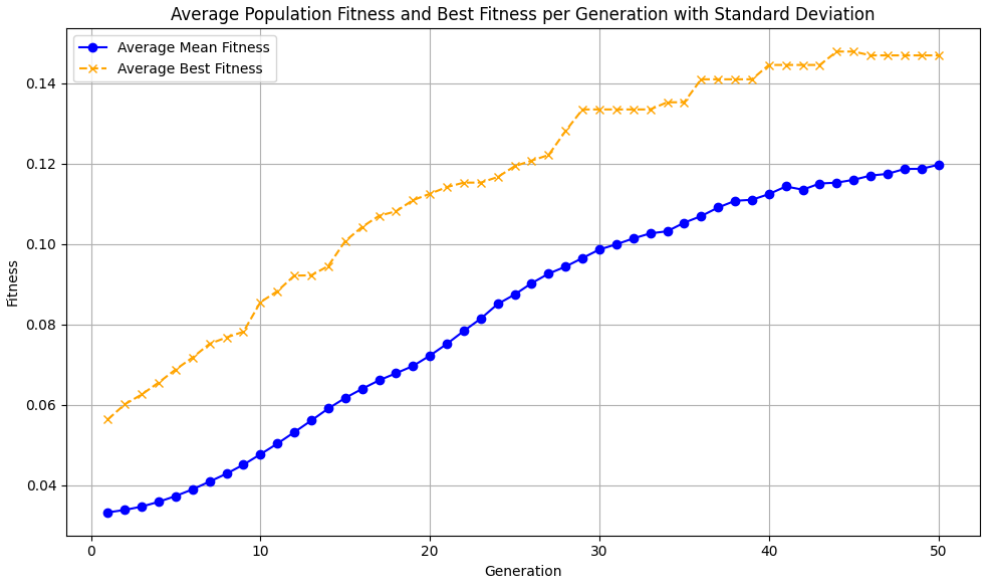
The table below depicts the statistics collected from test run of variation 4.

Table 4: Fitness Statistics

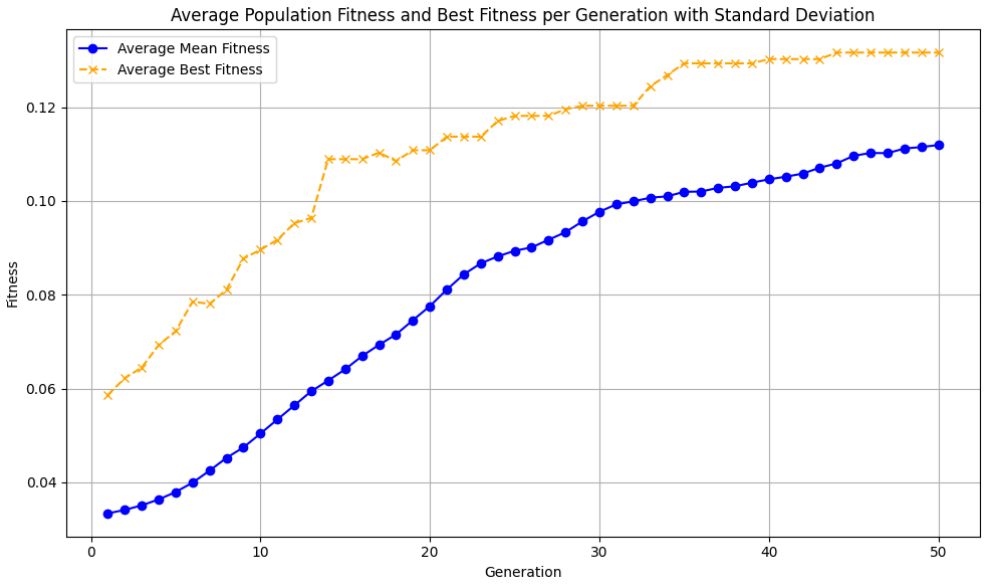
Generation	Average Mean Fitness	Std. Dev. Mean Fitness	Average Best Fitness	Std. Dev. Best Fitness
1	0.033412	0.000446	0.057967	0.006821
2	0.034245	0.000471	0.065049	0.006809
3	0.035256	0.000572	0.068168	0.007091
4	0.036566	0.000711	0.069002	0.006548
5	0.038268	0.000908	0.074952	0.008755
6	0.040323	0.001131	0.076454	0.007791
7	0.042940	0.001459	0.078761	0.007392
8	0.045859	0.002252	0.084800	0.008514
9	0.049264	0.003780	0.088298	0.007618
10	0.052694	0.004883	0.091632	0.008405
11	0.055986	0.005682	0.093652	0.010402
12	0.059162	0.006900	0.094561	0.010532
13	0.061941	0.007129	0.097071	0.009899
14	0.064334	0.006537	0.100606	0.008264
15	0.067143	0.007039	0.106237	0.010638
16	0.069691	0.007498	0.106237	0.010638
17	0.072359	0.008159	0.108023	0.014816
18	0.075078	0.008079	0.108023	0.014816
19	0.077691	0.008173	0.108023	0.014816
20	0.079882	0.008625	0.108932	0.013900
21	0.082128	0.007758	0.111432	0.014355
22	0.084382	0.007204	0.113218	0.017082
23	0.086679	0.006998	0.116392	0.019435
24	0.087675	0.005956	0.119567	0.021006
25	0.088520	0.006092	0.120956	0.020843
26	0.089739	0.006522	0.120956	0.020843
27	0.090325	0.006096	0.126670	0.032235
28	0.091579	0.006823	0.126670	0.032235
29	0.093007	0.007040	0.126670	0.032235
30	0.094085	0.007856	0.128059	0.031786
31	0.095846	0.009488	0.130440	0.033843
32	0.097774	0.010993	0.130440	0.033843
33	0.098772	0.011713	0.130440	0.033843
33	0.098772	0.011713	0.130440	0.033843
34	0.100183	0.013707	0.130440	0.033843
35	0.101774	0.015528	0.130440	0.033843
36	0.103170	0.016826	0.130440	0.033843
37	0.105108	0.019018	0.130440	0.033843
38	0.106041	0.020324	0.130440	0.033843
39	0.107521	0.021916	0.130440	0.033843
40	0.108364	0.022412	0.130440	0.033843
41	0.108691	0.022938	0.130440	0.033843
42	0.108861	0.023486	0.130440	0.033843
43	0.109705	0.023263	0.130440	0.033843
44	0.110524	0.024116	0.130440	0.033843
45	0.110670	0.024489	0.130440	0.033843
45	0.110670	0.024489	0.130440	0.033843
46	0.110674	0.025239	0.130440	0.033843
47	0.110433	0.023983	0.130440	0.033843
48	0.111407	0.025473	0.130440	0.033843
49	0.111327	0.024438	0.131551	0.032902
50	0.111453	0.024606	0.131551	0.032902

3.2 Performance Plots

These are the plots of each parameter settings, includes six parameter variations, each with 10 runs. The best hits are generated by the parameters 90% and 10% crossover, and mutation rate, with 10 elitism, provides 27 hits out of 30 hits of prey's amount, where the preys movement are limited to random cardinal movement such as up, down, left, and right. In other words, 27 prey are eaten by the predator. Variation 1 and variation 2 perform closely without any elites carried over to the next generation. Variation 3 & 6 converges early after a few generations of runs. With elitism, variations 4 and 5 closely perform with a small amount of differences on average best fitness, but variation 4 outperforms every other variation of parameter tested on the GP problem. Note that: the variations are described in detail in Table 3.

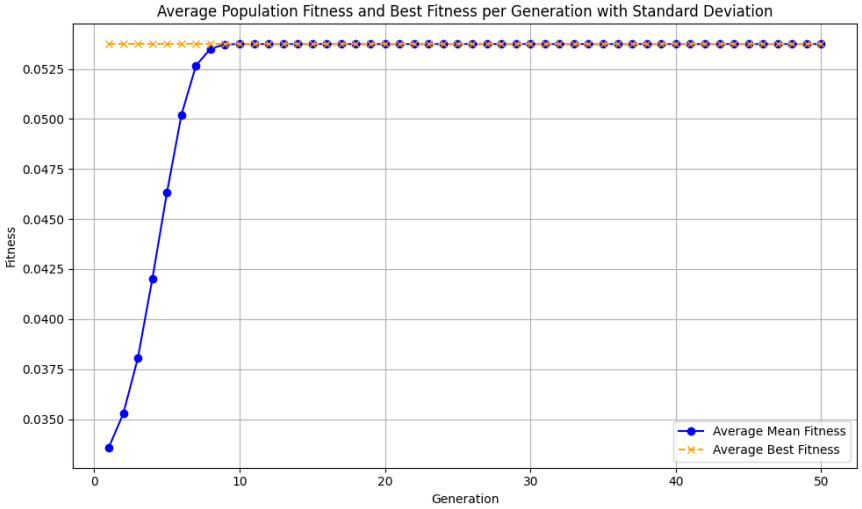


(a) (Variation 1) 90% & 10% with no Elitism



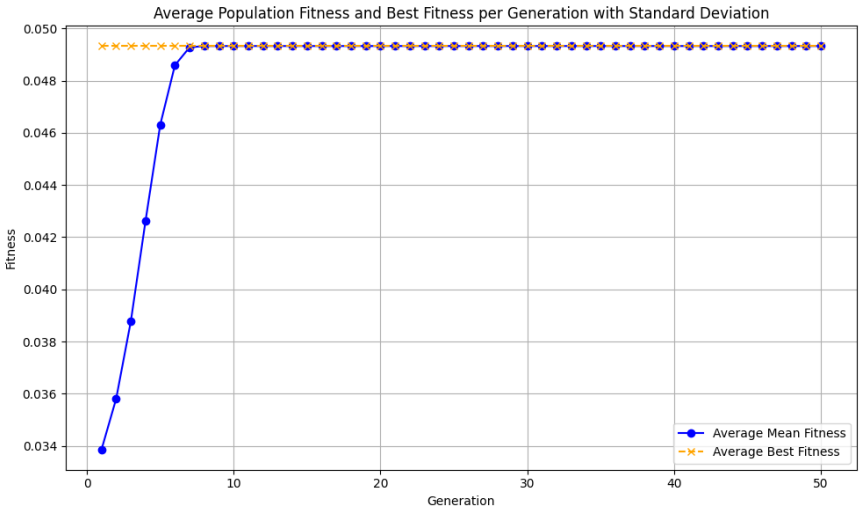
(b) (Variation 2) 100% & 0% with no Elitism

The worst-performed plots are grouped together on the next page. (Variation 3 and Variation 6 from Table 3).



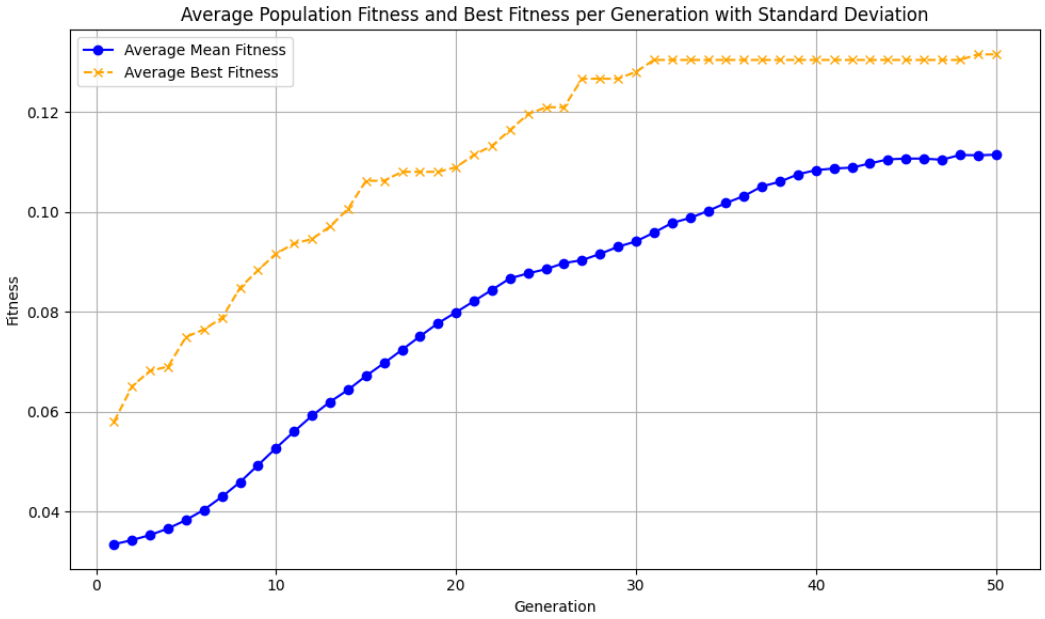
(a) (Variation 3) 0% & 100% with no Elitism

Figure 3: Average of Mean and best Fitness plotted in a graph across 10 runs with no elitism

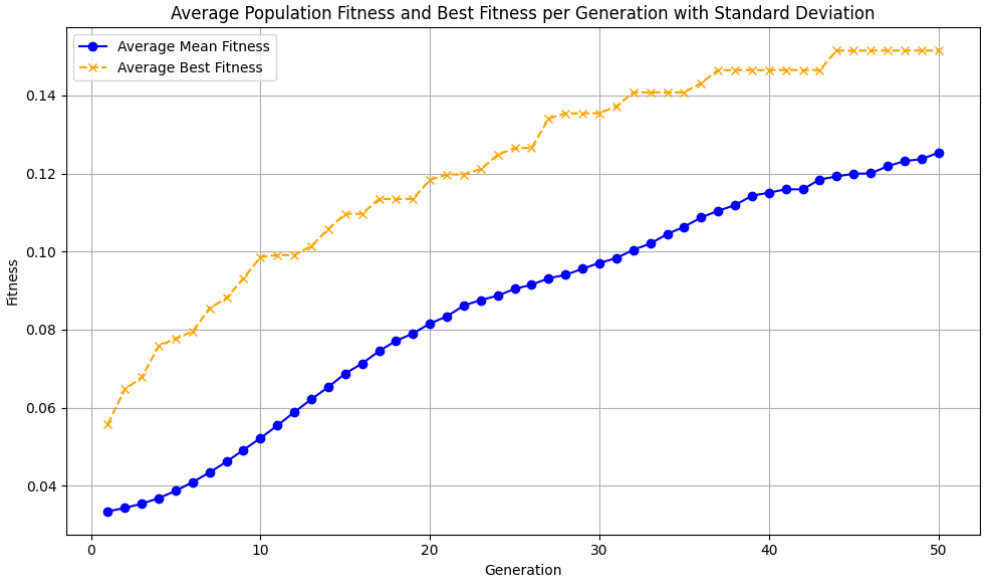


(a) (Variation 6) 0% & 100% with 10 Elite

Figure 4: Average of Mean and best Fitness plotted in a graph across 10 runs



(a) (Variation 4) 90% & 10% with 10 Elite



(b) (Variation 5) 100% & 0% with 10 Elite

Figure 5: Average of Mean and best Fitness plotted in a graph across 10 runs

3.3 Best GP Expressions

The following GP expression is the best one obtained from all the parameters tested in the GP problem.

```
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=3.0 Adjusted=0.25 Hits=27
Tree 0:
 (progn3 (progn3 (if-prey-behind (progn2 move
     (progn3 (progn3 (if-prey-behind left move)
         (progn3 move move move) (if-prey-ahead (if-prey-ahead
         left left) (if-prey-behind left move))) (if-prey-left
         (if-prey-left move (progn3 left move move))
         (if-prey-ahead left left)) (progn3 (if-prey-behind
         left move) (if-prey-behind (progn3 (if-prey-left
         move move) (progn3 (if-prey-behind left move)
         (progn3 move move move) (if-prey-ahead (if-prey-ahead
         left left) (if-prey-behind left move))) (if-prey-ahead
         (if-prey-left (if-prey-left (if-prey-behind
             (if-prey-ahead right (progn3 move right move))
             move) left) (progn2 move move)) (if-prey-ahead
         (if-prey-left move left) (if-prey-behind
         left move)))) move) (if-prey-ahead (if-prey-behind
         move right) (if-prey-behind left move)))))
     move) (progn3 move move move) (if-prey-ahead
     (if-prey-ahead left left) move)) (if-prey-left
     (if-prey-left move (progn3 (progn3 (if-prey-behind
         left move) (if-prey-behind (if-prey-left
         right right) (if-prey-ahead (if-prey-ahead
         left left) (if-prey-behind left move))) (if-prey-ahead
         (if-prey-behind move right) (progn2 move
         move))) (if-prey-ahead (if-prey-ahead (if-prey-behind
         move move) (if-prey-ahead right left)) (if-prey-behind
         (if-prey-left move move) (if-prey-left (if-prey-ahead
         left (if-prey-ahead (progn2 move move) move))
         (if-prey-behind right right)))) (progn3 (if-prey-left
         move move) (progn2 (if-prey-behind (if-prey-ahead
         (progn2 (progn2 move right) (progn2 (if-prey-ahead
             move (progn3 move move move)) (progn3 move
             right right))) (if-prey-ahead (if-prey-left
         move left) (progn2 (if-prey-left (if-prey-behind
         (if-prey-left right right) (if-prey-ahead
         left move)) (if-prey-ahead left move)) (if-prey-left
         move (if-prey-behind left move))))) (if-prey-behind
```

```
    move move)) (progn3 move move left)) (if-prey-ahead
    (if-prey-left move left) (progn3 move move
    left))))) (if-prey-ahead right (progn3 move
right move))) (progn3 (if-prey-left move
move) (progn2 (progn2 (if-prey-behind (progn3
(if-prey-behind move (if-prey-left move move))
(if-prey-left (if-prey-left right right)
    (if-prey-ahead right (progn3 move right (if-prey-behind
        move right)))) (progn3 (if-prey-left move
move) (progn2 (progn2 (if-prey-behind (if-prey-behind
right right) (if-prey-left (progn3 (if-prey-behind
move move) (progn3 move right move) (if-prey-ahead
move left)) (if-prey-behind right right)))
(progn3 move right right)) (if-prey-left
move left)) (if-prey-ahead (if-prey-left
(if-prey-left left left) (progn2 move move))
(progn3 move move move)))) (if-prey-behind
move move)) move) (if-prey-left move left))
(if-prey-ahead (if-prey-behind move right)
    (progn3 (if-prey-ahead (if-prey-ahead left
        left) (if-prey-behind left move)) move move))))
```

```
*   MAP - 2 dimensional [30x30]
*   Predator's current position - marked with "\#" (Hash symbol)
*   Location where preys are eaten - marked with "X" (Alphabet X)
*   Location of surviving prey - marked with "p" (Alphabet small-p)
*   Empty cells - marked with "." ( Dot ).

Best Individual's Map
=====================
Simulation Map:
##########..#..###############.
###.........#..#.......#######
#..##########..#..###X#######.
#..#........#####.#....#....#.
#..#..####X######.#..##X######
#..#..#....###X#X######X....#.
#####.#..####X#####..#..######
....X.#..#.#....#....#..#...#.
#######..#.#########X#..#..###
###.#....####...#.......X..###
..#.#.......#..##########.p#..
..#.#.......#..##.........#..
..X#####X####..##############..
..###.........#..#.#.........
..###.##########..#.#.........
..######X.........X.#.........
####..#.###########.#...###X##
..#...#.##.....p....#...#.....
##X####.#X.........#..p#..###
......#.##.........#...#..#..
#X########.....##########..X..
#.....#.#......X....#####..#..
#..########....X..####X#####..
#..#..#.#......#..#.....#.....
#..#..##########..#..#########
...#..#.....#.....#..#..X.....
####..#..#X########..#..####X#
..##X######.#........#..#.....
#######..#..###X######..#..###
..#......#..#..........#..#..
```

3.4 Variations of Experiments

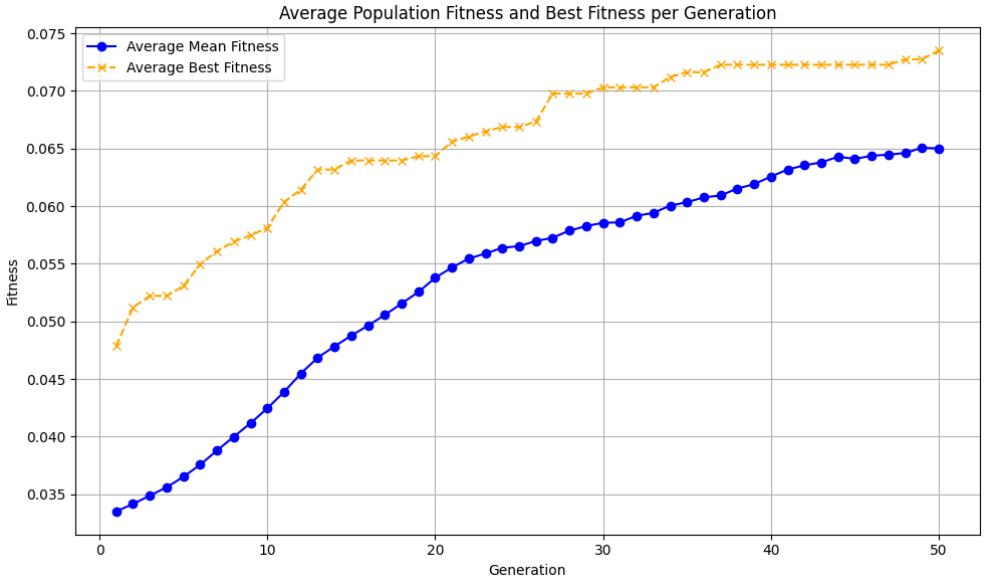
The experiment is done with 6 different variations of parameters. The variation 1 to 3 parameter sets are from ***Table3*** experimented without any elitism, and variation 4 to 6 parameter sets used elitism where the number of elite is 10.

3.5 Comparative Analysis

There were multiple comparisons done with simpler modification to advance, and one advance comparison will be shown where advanced prey movement is used.

The first instance of the Predator-prey application used GP language such, IfPreyAhead, move, left, right, progn2, progn3 without updating preys location. In this GP problem run, all the preys gets eaten.

The second instance of the Predator-prey application used the same GP language as the first instance but added preys random movement, where prey randomly moves, up, down, left, and right. This problem instance used only 300 moves to eat preys where the original problem mentioned in the experiment design uses 600 moves. The best parameter for this comparative instance of the Predator-prey GP problem is variation 4 (90% Crossover, 10% mutation rate with 10 elites). Within a grid of 30x30 with 30 preys inside of it, the predator was able to eat 22 preys out of 30 preys. Here's a graph of how the second instance of the GP problem performed.



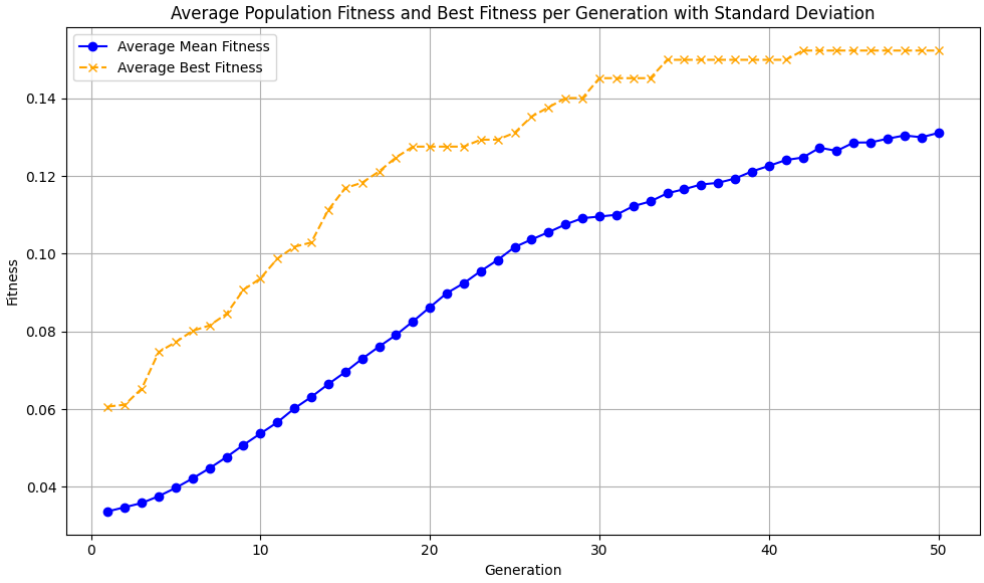
(a) Predator-prey Simple Version: (Variation 4) 90% & 10% with 10 elites

Figure 6: Average of Mean and best Fitness plotted in a graph across 10 runs with 10 elites

[Appendix A, 5.4] shows the results of 300 moves with similar settings but GP languages used are IfPreyAhead, move, left, right, progn2, progn3. And the preys had random movements such as up, down, left, and right.

The Final Instance, where advanced prey movements are incorporated. This instance is being compared with the GP problem that is defined in this paper. The preys movement is changed using 12 cases where first 4 cases used random movement up, down, left, right.

The next four explored the adjacent cells, where preys are able to move to the adjacent cells not just basic up, down, left, and right. In the next step, the preys are moved two steps in each direction. Preys locations are updated once predator makes a move, looks left, or right. With these advancement, some small amount of preys are able to escape the Process. Below is a detailed analysis of GP problem provided in the experimental section vs advance prey movement.



(a) Predator-prey Comparative Version: (Variation 4) 90% & 10% with 10 elites

Figure 7: Average of Mean and best Fitness plotted in a graph across 10 runs with 10 elites

The following [Appendix A, 5.1] depicts the statistics gotten from the 10 runs of GP problem instance with advance prey movement.

The following [Appendix A, 5.2] depicts the best map result of the parameter variation 4 (Table 1). The following GP problem is able to eat maximum of 26 preys out of 30 preys. Where the original problem maximized the prey eaten with 27 preys eaten out of 30 preys.

The following [Appendix A, 5.3] depicts the s-expression or sequences of functions and terminals executed by the predator while hunting prey.

3.6 Analytical Discussion of Results

The predator-prey GP problem performs well with 90% crossover, 10% mutation with 10 elites in all problem instances of the Predator-Prey. The first instance ate all the preys where preys are static, later with dynamic implementation preys gets less eaten by the predator, in other words, preys are able to avoid predator's pursuit. The difference however, is slight in the sense of preys eaten with the advance prey location update function. With more moves added to termination conditions, the predator is able to hunt for more preys within the grid environment. With 300 moves and with parameter variation 4 the predator seem to be able to eat only 22 preys in the grid, where preys had basic random movement such up, left, right, down for avoiding the predator.

4 Conclusions

The predator-prey application correctly eats prey within a provided grid of 30x30 with the amount of 30 preys on the map. The specific problem uses various GP languages to define the scope of the problem. Parameter variation 4 performed well in all cases of results. Comparisons were done by using advnace prey movement by updating preys location in cardinal way, then adjacent exploration, lastly two steps ahead in each cardinal direction. In all, the GP problem satisfactorily performs well with parameter variations 4 (Table1). While parameter variation 3 and 6 converged early becuase of 0% crossover in it. Also, the variation 1, 2, 4, 5 performed closely one another with slight difference in output. However, the best output provided by running variation 4 parameter sweeping leads to 27 preys eaten out of 30 preys.

FutureWork : The terminal set can be increased. For example, looking up, and down. Checking within a radius for preys. Also preys could be evolved into sensing predators for example, evasion, area knowledge, grouping preys together.

5 Appendix A

5.1

Table 5: The following table depicts the statistics data collected from comparative test run with variation 4 parameter.

Generation	Avg. Mean Fitness	Std. Dev. Mean Fitness	Avg. Best Fitness	Std. Dev. Best Fitness
1	0.033691	0.000449	0.060668	0.007346
2	0.034771	0.000618	0.061144	0.007911
3	0.035842	0.000886	0.065301	0.009505
4	0.037611	0.001258	0.074700	0.009990
5	0.039740	0.001647	0.077259	0.008848
6	0.042177	0.002260	0.080160	0.011411
7	0.044803	0.003013	0.081545	0.012134
8	0.047671	0.003740	0.084519	0.010656
9	0.050791	0.004327	0.090723	0.007874
10	0.053723	0.004530	0.093652	0.010402
21	0.089880	0.012160	0.127579	0.011907
22	0.092432	0.013543	0.127579	0.011907
23	0.095547	0.012431	0.129365	0.012784
24	0.098411	0.012185	0.129365	0.012784
25	0.101713	0.011736	0.131151	0.013341
26	0.103680	0.011319	0.135317	0.017165
27	0.105554	0.010924	0.137698	0.019779
28	0.107546	0.010232	0.140079	0.021799
29	0.109160	0.010331	0.140079	0.021799
30	0.109572	0.010151	0.145198	0.027241
31	0.110030	0.010043	0.145198	0.027241
32	0.112254	0.010866	0.145198	0.027241
33	0.113460	0.011048	0.145198	0.027241
34	0.115560	0.012214	0.149921	0.030266
35	0.116616	0.014235	0.149921	0.030266
36	0.117833	0.014808	0.149921	0.030266
37	0.118257	0.015743	0.149921	0.030266
38	0.119361	0.015439	0.149921	0.030266
39	0.121197	0.015860	0.149921	0.030266
40	0.122616	0.016302	0.149921	0.030266
41	0.124124	0.017409	0.149921	0.030266
42	0.124775	0.018244	0.152302	0.030583
43	0.127293	0.021616	0.152302	0.030583
44	0.126479	0.019591	0.152302	0.030583
45	0.128620	0.021343	0.152302	0.030583
46	0.128657	0.021886	0.152302	0.030583
47	0.129619	0.021682	0.152302	0.030583
48	0.130451	0.020824	0.152302	0.030583
49	0.129987	0.021867	0.152302	0.030583
50	0.131106	0.021711	0.152302	0.030583

5.2

```
Best Individual's Map
=====================
Simulation Map:
#...X......######.#....p#.....
X...#.......#...X###....#.....
#..##......##....########....#
...#.......#.....#.....#######
#####......#.....#.....#.....#
...##X####.#.....#.....#.....#
...#...#X######..#.....#.....#
..##..#X####..#X####..X#....##
..##..X##.#.....#..#X####...#.
..#####.#.#.....#.....#.######
#####..p#.#.....#.....#.....##
..#.#######.p...X.....#.....#.
.##....##########....##....##.
.#.....#.#....####XX.#.....#..
.#.....#.#####.#...#X####..#..
.###...#.#X..######..#..######
####.###X#X....#..####X#...#.#
..p..###X###..##....#X.######.
####.#X....######...#.....#.##
...####.......#.######....#...
......X.......#.....#######...
X#....#.......#.....#.....####
.######......##....##....##...
.....####X##.#.....#.....#....
.....#.....######..#.....#....
.....#.......#..######...#....
.....#.......#.....#.######...
##..##......##....##....######
.###X##.....X.....#.....#.....
....#.#######.....#.....#.....
```

5.3

```
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=4.0 Adjusted=0.2 Hits=26
Tree 0:
 (if-prey-ahead (progn2 (if-prey-behind (progn2
     move (if-prey-left right right)) (progn2
     (if-prey-behind left right) (if-prey-left
```

```
(progn2 (if-prey-behind (progn2 (progn2 (if-prey-behind
    left right) (if-prey-left (progn2 right right)
    (if-prey-left (if-prey-left move left) (if-prey-left
        (if-prey-ahead (progn2 right move) (if-prey-behind
            left right)) (progn2 (if-prey-left left right)
        (if-prey-left move left)))))) (if-prey-left
    right right)) (progn2 (if-prey-left right
    left) (if-prey-left (if-prey-left (progn3
    right left move) move) (if-prey-left (if-prey-left
    move left) (progn2 move right))))) (progn3
    right (if-prey-behind (if-prey-behind right
    left) (if-prey-behind (progn3 (progn2 (if-prey-ahead
    right (if-prey-left (progn2 right right)
    (if-prey-left move left))) (if-prey-ahead
    left right)) (if-prey-left move left) (progn2
    left right)) (if-prey-left left right)))
    (progn3 (progn3 left left move) (if-prey-left
        right left) (if-prey-left (progn3 (if-prey-ahead
        left move) (progn2 left right) (progn2 left
        right)) move)))) (if-prey-ahead (progn2 right
move) (progn3 (if-prey-ahead (if-prey-left
left right) (progn3 left move right)) (if-prey-behind
move (if-prey-left move move)) (progn2 right
move))))))) (progn3 (if-prey-behind (if-prey-ahead
left right) (if-prey-behind left right))
(if-prey-behind (if-prey-behind right (if-prey-behind
    left (progn3 (if-prey-behind (if-prey-ahead
    left right) (if-prey-left move left)) (if-prey-behind
    (if-prey-behind right left) (if-prey-behind
    (progn3 (progn2 (if-prey-ahead right (if-prey-left
        (progn2 right right) (if-prey-left move left)))
        (if-prey-left (if-prey-ahead (progn2 right
            move) (progn3 (if-prey-ahead (if-prey-left
            left right) (progn3 left move right)) (if-prey-behind
            move (if-prey-left move move)) (progn2 right
            move))) (progn2 (if-prey-left left right)
            (if-prey-left move left)))) (if-prey-left
        move left) (progn2 left right)) (if-prey-left
    left right))) (progn3 (progn3 left left move)
    (if-prey-left right left) (if-prey-left (progn3
    (if-prey-ahead left move) (progn2 left right)
    (if-prey-left left right)) move))))) (if-prey-behind
    (if-prey-left right move) (if-prey-left left
    right))) (progn3 (progn3 left left move)
(if-prey-left right left) (if-prey-left (if-prey-left
```

```
    move left) (if-prey-left left move))))) (progn3
    (progn2 (if-prey-left move move) move) (if-prey-ahead
    (if-prey-behind (if-prey-left move left)
        (if-prey-left (if-prey-left (if-prey-left
            left left) (if-prey-left move right)) (if-prey-behind
            (if-prey-behind move left) (progn3 left move
            right)))) (progn3 (progn3 left move right)
    (if-prey-left move move) (if-prey-left move
    move))) (if-prey-left (if-prey-behind left
    move) (if-prey-left (if-prey-left move left)
    (if-prey-left left move)))))
```

5.4

```
Best Individual of Run:
Subpopulation 0:
Evaluated: true
Fitness: Standardized=8.0 Adjusted=0.1111111111111111 Hits=22
Tree 0:
 (progn2 (progn3 (if-prey-ahead move move)
    (if-prey-ahead right move) (if-prey-ahead
    move left)) (if-prey-ahead (progn2 (progn2
    (progn2 move move) (progn2 right move)) (if-prey-ahead
    (if-prey-ahead (progn3 (if-prey-ahead (progn2
        (if-prey-ahead (progn2 (progn3 (if-prey-ahead
            (progn3 right left left) (progn3 (progn3
            right left left) (if-prey-ahead left move)
            (progn3 move right move))) (progn2 right
            right) (if-prey-ahead (progn3 left right
            move) (if-prey-ahead move right))) (progn3
            (if-prey-ahead (if-prey-ahead (progn2 move
                (progn3 left move (if-prey-ahead right move)))
                (if-prey-ahead move left)) right) (progn3
            right move left) (progn3 move move right)))
            (if-prey-ahead (if-prey-ahead (if-prey-ahead
                right right) (progn2 left right)) (progn3
                (progn3 move right move) (if-prey-ahead move
                right) (progn2 move move)))) (progn2 right
        left)) (if-prey-ahead move left)) (if-prey-ahead
        (if-prey-ahead right move) (progn3 move right
        move)) (if-prey-ahead move left)) (progn2
        (if-prey-ahead left move) (if-prey-ahead
        move right))) (progn2 (if-prey-ahead left
    move) (if-prey-ahead move right)))) (progn2
    (if-prey-ahead left move) (if-prey-ahead
    move right))))
```

```
Best Individual's Map
=====================
Simulation Map:
#.............##.#..........
#.............#..##..........
#.............##.X#X#X##.....#
...........p..#..#..#..###...#
..............##X#..###..#####
...............#......###..###
##....##X##...##........######
.###..#...#...#...........X#..
...#.##...X..##..........#X#X.
##X#.#....#..#...........#..X#
.###X#...##.##...........#####
##.......X..#..............#X#
##.....#X#.##......p.........#
####...#...#.................#
...###.###X#.................#
.....###.....................##
.......###.p...p............#.
.........###................##.
.....p.....###.......###...#..
.............####X####.X..##..
...............###..####..#...
...p............#####.##.##...
.................X..#XX#..#....
.......p........####.##.##....
...................####..#.....
...................#.###X#.....
.................##.#.........
.................#.##.........
........p.......##.#..........
.................#.##..........
```

6 Bibliography

References

[1] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[2] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A Field Guide to Genetic Programming*, Lulu.com, 2008.

[3] Sean Luke et al., *ECJ: The Evolutionary Computation Toolkit User's Guide*, https://cs.gmu.edu/~eclab/projects/ecj/, Accessed on February 16, 2024.