Assignment Part B

Trond Zachariassen

February 18, 2024

Task 1 : Theory

1. **What are the categories of optimization problems? Provide suitable examples for each category.**

   Continuous optimization: deals with optimizing functions over continuous domains. Examples include finding the minimum of a mathematical function using techniques like gradient descent or Newton's method.

   Discrete optimization: involves optimizing functions over discrete domains. Examples include the traveling salesman problem, where the goal is to find the shortest possible route that visits each city exactly once.

   Combinatorial optimization: focuses on optimizing functions over finite sets of discrete items. Examples include the knapsack problem, where the goal is to maximize the value of items selected for a knapsack subject to a weight constraint.

2. **What are the types of crossover methods used in genetic algorithms?**

   Single-point crossover: In this method, a single crossover point is randomly selected along the length of the parent chromosomes. Offspring are created by swapping the genetic material between the parents at this crossover point.

   Two-point crossover: Similar to single-point crossover, but two crossover points are selected instead of one. Genetic material between these two points is swapped between parents to create offspring.

   Uniform crossover: In this method, each gene in the offspring is randomly selected from one of the parents with equal probability. This allows for greater exploration of the search space compared to single-point or two-point crossover.

   Multi-point crossover: This method involves selecting multiple crossover points along the length of the parent chromosomes. Genetic material between these points is exchanged to create offspring.

3. **Write the methods for parent selection for crossover?**

   Roulette wheel selection: Individuals are selected with probabilities proportional to their fitness values.

   Tournament selection: A subset of individuals is randomly chosen from the population, and the fittest individual among them is selected as a parent.

   Rank-based selection: Individuals are ranked based on their fitness values, and selection probabilities are assigned based on their ranks.

   Random selection: Parents are chosen randomly from the population with uniform probability.

4. **What are the factors affecting the convergence of genetic algorithm?**

   Population size: Larger populations can increase exploration but also computational costs.

   Selection pressure: Balancing exploration and exploitation by adjusting selection pressure can impact convergence.

   Crossover and mutation rates: Properly balancing these rates affects the algorithm's ability to explore and exploit the search space.

   Termination criteria: Determining when to stop the algorithm can influence convergence behavior.

5. **What will be the impact of "never" (0%), "rarely" ($\approx$ 1%), and "too often" ($\approx$ 50%) mutation?**

   Never (0%) mutation: The algorithm may become stuck in local optima, as it lacks the ability to introduce new genetic diversity.

   Rarely ($\approx$ 1%) mutation: The algorithm may still be able to explore the search space effectively while maintaining stability. However, it may take longer to converge to the optimal solution.

   Too often ($\approx$ 50%) mutation: Excessive mutation can disrupt stable genetic structures, leading to a loss of convergence and inefficiency in exploration and exploitation.

6. **What will be the impact of keeping "too many" and "too few" chromosomes in GA?**

   Too many chromosomes: Increases diversity and exploration but also increases computational complexity and can lead to premature convergence.

   Too few chromosomes: Limits diversity and exploration, making it more likely to converge prematurely to suboptimal solutions.

7. **What will be the impact of crossover to perform "all the time" (100%) and "never" (0%) in GA?**

   Crossover all the time (100%): Promotes exploration by combining genetic material from different individuals, leading to a wide exploration of the search space. However, it may reduce exploitation by not allowing individuals to retain beneficial genetic material.

   Crossover never (0%): Eliminates the exchange of genetic material between individuals, which may lead to premature convergence and a lack of genetic diversity. It limits exploration and may hinder the algorithm's ability to find optimal solutions.

8. **What are the different possible termination criteria to stop the evolution loop in GA?**

   Generations limit: Maximum number of generations reached

   Convergence criteria: no improvement in the best fitness for a certain number of generations

   Solution quality threshold: reaching a predefined fitness score

   Computational resources exhausted: time limit or memory constraints

Task 2 : Methods

Table 1: Performance Analysis of GA with Different Parameters

Test	Pop. Size	Selection Method	Crossover Method	Mutation Rate	Generations	Best Fitness
1	10	Tournament (5)	Single-point	0.01	1756	1
2	10	Tournament (5)	Single-point	0.03	853	1
3	100	Tournament (5)	Single-point	0.01	116	1
4	100	Tournament (5)	Single-point	0.01	232	1
5	100	Tournament (5)	Single-point	0.05	91	1
6	500	Tournament (5)	Single-point	0.05	29	1
7	2000	Tournament (5)	Single-point	0.01	17	1

I have run multiple tests and observed how changes to the parameters impact the results.

Population Size:
Increasing population size will decrease the amount of generations it will take to find the answer, but this will use more computating power.

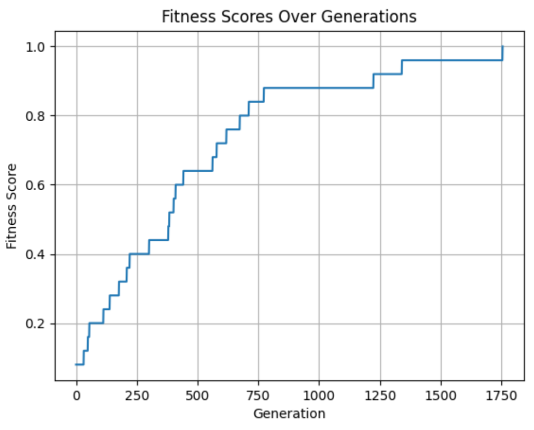
Mutation Rate:
A lower mutation rate is more stable but can cause it to use a longer time to find the right answer, a bit of mutation is beneficial to be able to find the solution.

We can also observe from the results of tests 1 and 2 that changing the mutation rate drastically decreased the number of generations. This is because the mutation allows for more than one letter to change at a time compared to the previous test. But if the mutation rate is too high, it could lead to instability. This will cause the algorithm to get trapped in a loop where it changes too many letters, hindering its ability to converge effectively.

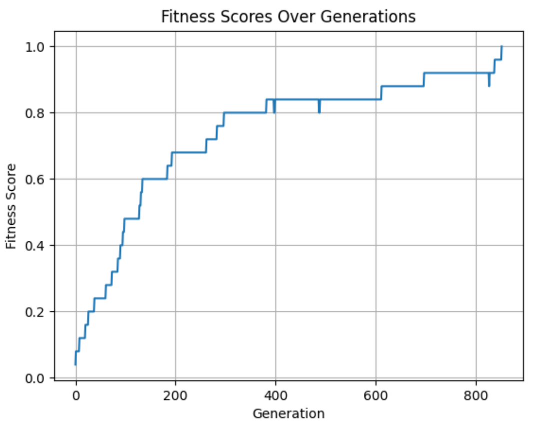
We can also observe the difference between the results of test 3 and 4, where the parameters are the same but the number of generations differs significantly. This discrepancy can be attributed to randomness and chance, as the algorithm utilizes random letters when generating new strings.

Here are the graphs showing the fitness scores converging through generations

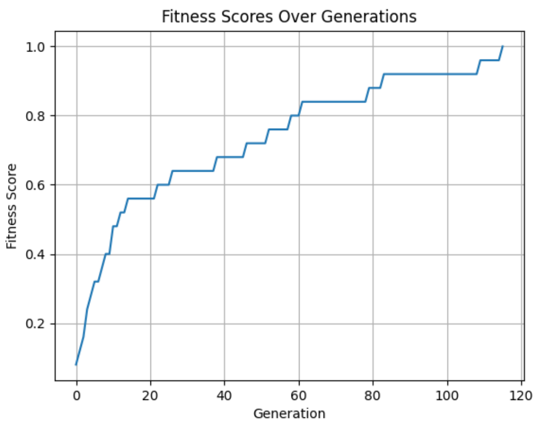
Test: 1



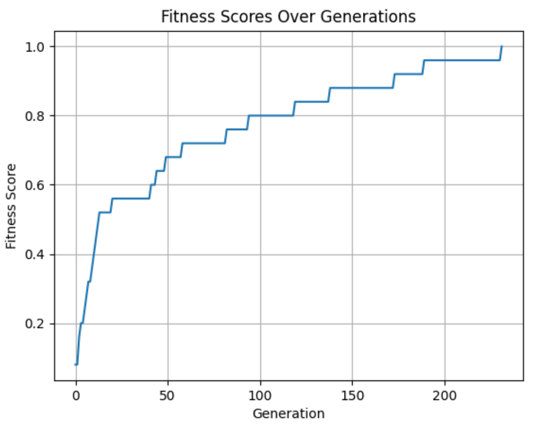
Test: 2



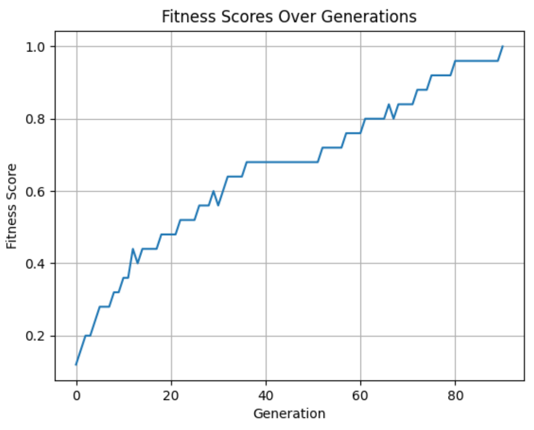
Test: 3



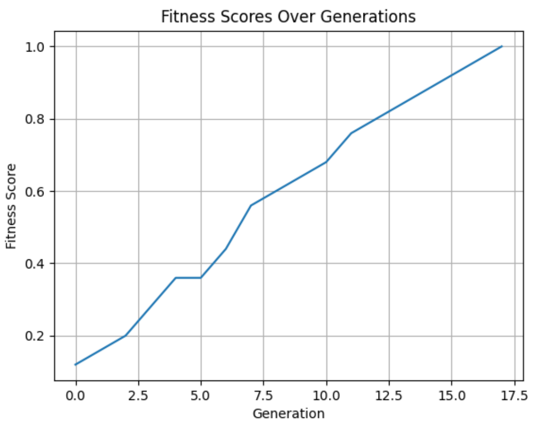
Test: 4



Test: 5



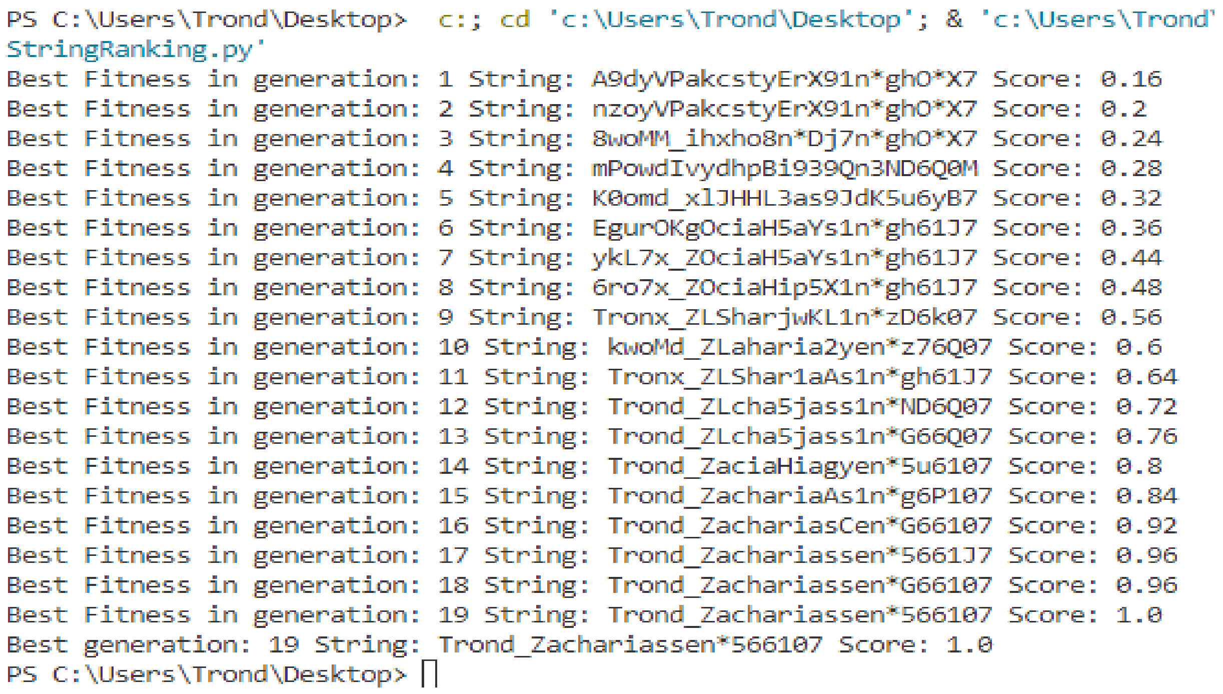
Test: 6



Task 3 : Coding

Parameter used in testing of code.

Pop. Size	Selection Method	Crossover Method	Mutation Rate	Generations	Best Fitness
2000	Tournament (5)	Single-point	0.01	19	1



```
PS C:\Users\Trond\Desktop>  c:; cd 'c:\Users\Trond\Desktop'; & 'c:\Users\Trond'
StringRanking.py'
Best Fitness in generation: 1 String: A9dyVPakcstyErX91n*ghO*X7 Score: 0.16
Best Fitness in generation: 2 String: nzoyVPakcstyErX91n*ghO*X7 Score: 0.2
Best Fitness in generation: 3 String: 8woMM_ihxho8n*Dj7n*ghO*X7 Score: 0.24
Best Fitness in generation: 4 String: mPowdIvydhpBi939Qn3ND6Q0M Score: 0.28
Best Fitness in generation: 5 String: K0omd_xlJHHL3as9JdK5u6yB7 Score: 0.32
Best Fitness in generation: 6 String: EgurOKgOciaH5aYs1n*gh61J7 Score: 0.36
Best Fitness in generation: 7 String: ykL7x_ZOciaH5aYs1n*gh61J7 Score: 0.44
Best Fitness in generation: 8 String: 6ro7x_ZOciaHip5X1n*gh61J7 Score: 0.48
Best Fitness in generation: 9 String: Tronx_ZLSharjwKL1n*zD6k07 Score: 0.56
Best Fitness in generation: 10 String: kwoMd_ZLaharia2yen*z76Q07 Score: 0.6
Best Fitness in generation: 11 String: Tronx_ZLShar1aAs1n*gh61J7 Score: 0.64
Best Fitness in generation: 12 String: Trond_ZLcha5jass1n*ND6Q07 Score: 0.72
Best Fitness in generation: 13 String: Trond_ZLcha5jass1n*G66Q07 Score: 0.76
Best Fitness in generation: 14 String: Trond_ZaciaHiagyen*5u6107 Score: 0.8
Best Fitness in generation: 15 String: Trond_ZachariaAs1n*g6P107 Score: 0.84
Best Fitness in generation: 16 String: Trond_ZachariasCen*G66107 Score: 0.92
Best Fitness in generation: 17 String: Trond_Zachariassen*5661J7 Score: 0.96
Best Fitness in generation: 18 String: Trond_Zachariassen*G66107 Score: 0.96
Best Fitness in generation: 19 String: Trond_Zachariassen*566107 Score: 1.0
Best generation: 19 String: Trond_Zachariassen*566107 Score: 1.0
PS C:\Users\Trond\Desktop> 
```

Stopping criteria: For the algorithm to stop, it needs to either successfully find the target string or exceed the maximum number of generations set in the parameters. Since I want the algorithm to find a very specific string, I want it to succeed. However, if it were finding a number or a combination to solve an equation, I would accept a value close to the target. For example, if the fitness score is 98%, then it can terminate and provide a result.

Link to code

Task 4 : Coding

Too be continued...

Link to code