lab7

February 3, 2024

[5]:

```
# 7.  Write a program to implement simple genetic algorithm
import random
population_size = 10
chromosome_length = 6
mutation_rate = 0.1
target_chromosome = '110110'

# Generate initial population
population = [''.join(random.choice('01') for _ in range(chromosome_length))␣
 ↪for _ in range
             (population_size)]
#print(population)

# Evaluate fitness of a chromosome
def fitness(chromosome):
    return sum(bit == target_bit for bit, target_bit in zip(chromosome,␣
 ↪target_chromosome))

for generation in range(50):
    new_population = []
    for _ in range(population_size // 2):
        parents = random.choices(population, weights=[fitness(chromosome) for␣
 ↪chromosome in population],
                  k = 2)
        crossover_point = random.randint(0, chromosome_length-1)
        child1 = parents[0][:crossover_point] + parents[1][crossover_point:]
        child2 = parents[1][:crossover_point] + parents[1][crossover_point:]

        #mutation
        child1 = ''.join(bit if random.random() > mutation_rate else str (1 -␣
 ↪int(bit)) for bit in child1)
        child2 = ''.join(bit if random.random() > mutation_rate else str (1 -␣
 ↪int(bit)) for bit in child2)

        new_population.extend([child1, child2])
    population = new_population
    best_chromosome = max(population, key=fitness)
    print(f'Generation {generation+1}: Best chromosome - {best_chromosome},␣
↪fitness = {fitness  (best_chromosome)}')

     # Check for convergence
    if fitness(best_chromosome) == len(target_chromosome):
        print("Target chromosome reached!")
        break
```

```
Generation 1: Best chromosome - 010111, fitness = 4
Generation 2: Best chromosome - 110111, fitness = 5
Generation 3: Best chromosome - 110111, fitness = 5
Generation 4: Best chromosome - 110100, fitness = 5
Generation 5: Best chromosome - 110110, fitness = 6
Target chromosome reached!
```

[ ]:

```
```