Intelligent Systems (IS Fall 2013)

Assignment 5: Evolutionary Computation

Student: Fang–Lin He

My answers to questions

- Question 1

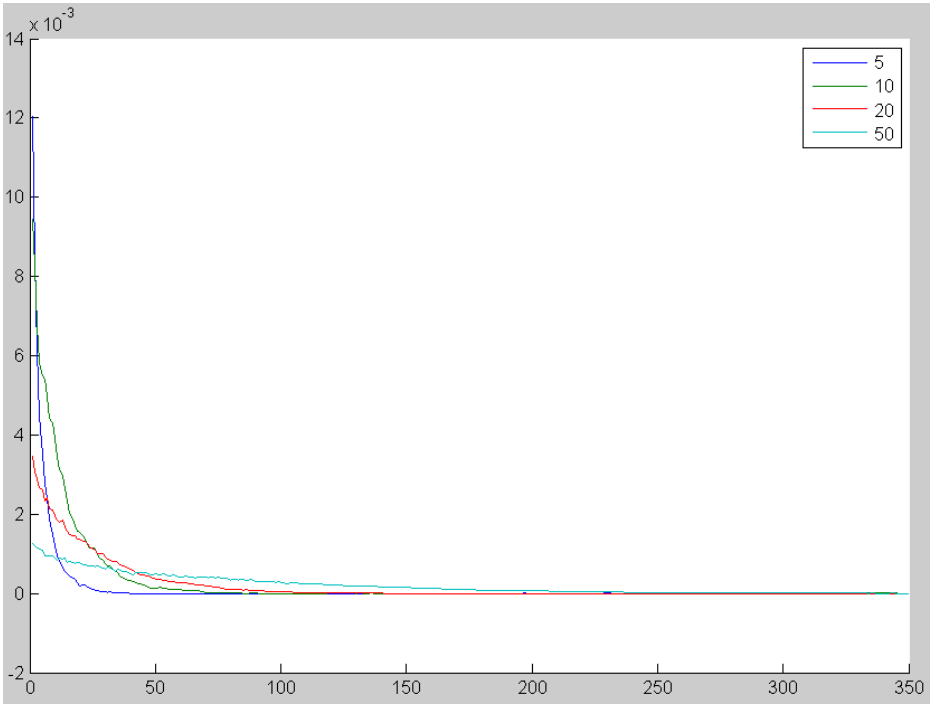
  Implement a binary Genetic Algorithm that uses fitness proportional selection, 1-point crossover, and bit-flip mutation to solve the problem in which the fitness is the number of 1s in the chromosome, i.e. the optimal solution is the chromosome where all genes are set to 1.

  A. (40 points) Run the algorithm 10 times for each of the four following versions of the problem: l = 5; 10; 20; 50, where l is the length of the chromosomes. Vary the population size and mutation rate to obtain good results (fast solution).

  - See Q1.m which calls the function *BinaryGA*.
  - Parameters:
    - chromosome_size: the length of the chromosomes (i.e., l)
    - population_size: number of populations
    - mutation_rate: mutation rate
    - iteration_time: the maximum number of generations
    - good_solution_percent: when some percentage of all the populations have reached optimal solution (all 1s for each gene), BinaryGA will return the second output as the number of generation when it reaches this criteria.
    - repeat_time: how many times to run for each chromosome_size
  - BinaryGA runs the genetic algorithm, collects the best fitness in each generation, and returns the best fitness and the generation time when it first fulfill the criteria (that is 75% of populations have got all 1's).
  - The mutation rule in my program is: for each gene, it has mutation_rate (in my program, 0. 00001) probability to mutate. In the beginning, I set mutation_rate as 0.001 or 0.005, it works well with the chromosome_size 5, 10, it doesn't work so well with chromosome_size 20, and it doesn't work at all with chromosome_size 50. So after, I check the fitness and found that, the higher mutation_rate, the slower it converges and the more unstable. Finally I decrease the mutation_rate and found the best one is 0.00001. I did not expect it would be so small!

B. (10 points) Plot the best fitness in each generation (averaged over the 10 runs), for each of the four problems. There should be one graph with four curves, the x-axis being the generations, and the y-axis the average (best) fitness.

- My plot: x-axis is the generations; y-axis is the average best fitness.



- Each fitness curve is subtracted by its minimum fitness so that the four curves align at the end.
- From the observation of this plot, I found that the smaller chromosome size, the faster GA converges.
- For smaller chromosome size, the fitness is higher in the beginning, which means the variation of each population is higher. In this case, to pick up 'good solution' is easier, because for good chromosomes, they have higher probabilities to be selected as parents, so it can converge faster.

- Question 2

Implement (1+ λ)-ES to optimize the following function:

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2]$$

known as the Rosenbrock function where N is the number of dimensions, and -5 <= x_i <= 10, i = 1, 2, ..., N.

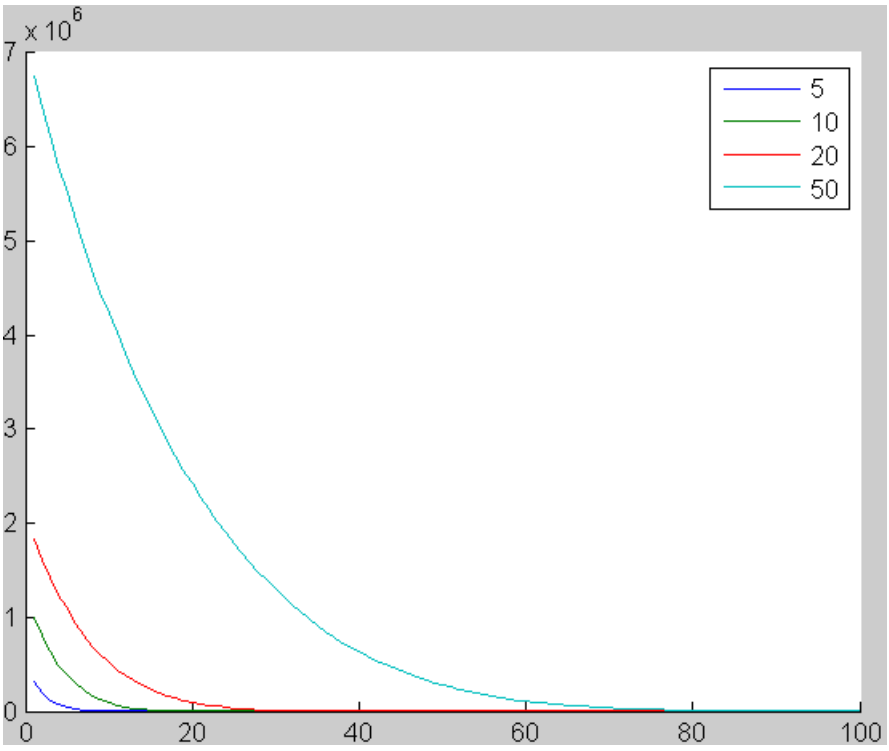
A. (40 points) Run the algorithm 10 times for each of the following four settings, N = 5, 10, 20, 50. Choose your own λ.

- See Q2.m which calls the function *EvolutionStrategy*.

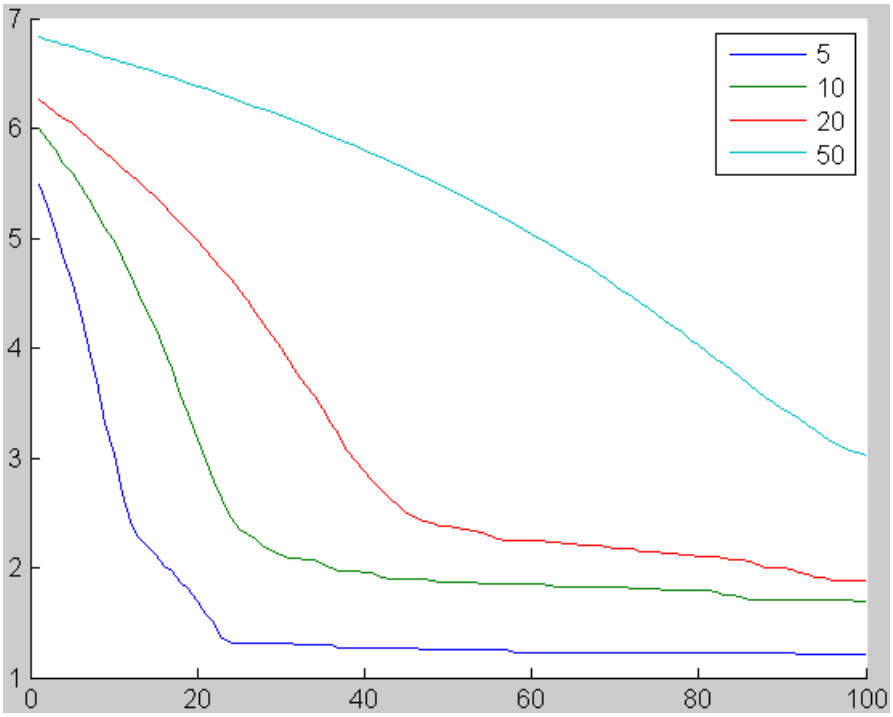
- Parameters:
  - chromosome_size: the length of the chromosomes (i.e., l)
  - num_offsprings_size: number of offsprings (λ)
  - learning_rate: learning rate (τ)
  - iteration_time: the maximum number of generations
  - repeat_time: how many times to run for each chromosome_size
- EvolutionStrategy runs the (1+λ)-ES, collects the fitness (fitness = Rosenbrock(chromosome)) in each generation, and returns the fitness at each generation time.
- The learning_rate, as suggested in the slides, is set as 1/sqrt(chromosome_size).
- During experiments, I found that the initial chromosome is very important. Sometimes the fitness is very high even if it has converged. Sometimes the fitness is very low after only a few iterations.

B. (10 points) Plot the average fitness (over the 10 runs) of the parent in each generation for each N. As with question 1B, there should be four curves here.

- My plot: x-axis is the generations; y-axis is the average fitness.



- Plot of log(y-axis):



- From these plots, I observed that, for bigger chromosome size, the fitness in the beginning is higher, and it converges slower.
- Sometimes the fitness is as low as 10, but sometimes the fitness is as high as 200. I tried longer generations, but it still seems the algorithm only finds the suboptimal solution. I adjusted the learning rate, but it still converges to suboptimal solution.