Assignment 4

Evolutionary Computation

Deadline March 14, 2023

Handout for the *Natural Computing* lecture, March 2, 2023

Inge Wortel, Mihaela Mihaylova

In this assignment, you are going to test your understanding of evolutionary algorithms, implement some yourself, and investigate the influence of design choices like survivor selection and integration with local search.

Objectives of This Exercise

1. Explain how choices like fitness function and survivor selection affect selection pressure in a GA
2. Implement simple EAs and visualize their performance
3. Design a fair experiment comparing an expanded EA to a baseline

Exercises

For each exercise below, please provide answers (along with any requested figures etc) in a pdf file. Also provide your code for exercises 2, 3, 4 or a URL link to a Github repository containing this code. Along with your code, you should provide clear instructions on how to reproduce your results using that code.

Exercise 4.1 Fitness function and selection pressure (1 point)

Consider the following fitness functions:

(a) $f_1(x) = |x|$
(b) $f_2(x) = x^2$
(c) $f_3(x) = 2x^2$
(d) $f_4(x) = x^2 + 20$

1. Given each fitness function above, calculate the probability of selecting the individuals $x = 2$, $x = 3$, and $x = 4$ using fitness-proportional selection.
   - provide a table with fitness function value and selection probability for each individual and each function.
   - plot the pie chart ("wheel of fortune") with selection probability of the three individuals for each function.
2. What can you conclude about the effects of fitness scaling on selection pressure?

Exercise 4.2 Role of selection in GAs (2 points)

A simple (1+1)-GA for binary problems works as follows:

- Step 1: Randomly generate a bit sequence x
- Step 2: Create a copy of x and invert each bit with probability μ. Let x_m be the result.
- Step 3: If x_m is closer to the goal sequence than x, replace x with x_m.
- Step 4: Repeat steps 2 and 3 until the goal sequence is reached.

The Counting Ones problem amounts to fit a bit string whose sum of entries is maximum. Implement a simple (1+1)-GA for solving the Counting Ones problem.

1. Use bit strings of length $l = 100$ and a mutation rate $\mu = 1/l$. For a run of 1500 generations, plot the best fitness against the elapsed number of generations.
2. Repeat this process to obtain a total of 10 independent runs. In a single figure, plot the best fitness against elapsed number of generations in each run. How many times does the algorithm find the optimum?
3. Now replace Step 3 in the above algorithm with the following: "replace x with x_m". Again perform 10 runs of 1500 generations, and plot the runs together in one figure as before. Is there a difference in performance when using this modification? Justify your answer.

Exercise 4.3 Memetic algorithms vs simple EAs (3 points)

Recall the simple EA for the TSP described in our first lecture (see slides). Implement both this algorithm and a variation based on memetic algorithms (MAs). Use the 2-opt algorithm as a local search technique in the memetic algorithm. The 2-opt algorithm tries to swap all pairs of cities to see if this improves the length of the tour (see, e.g. `https://en.wikipedia.org/wiki/2-opt`).

You will investigate the effect of adding local search on performance. Please do so on two datasets:

- the TSP problem instance given in the file 'file-tsp'. This file contains a 50 x 2 matrix with coordinates (x_i, y_i) for each city $i = 1, ..., 50$
- a small instance of your choice selected from the "Symmetric Traveling Salesman Problem" benchmark instances available at `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`.

1. Compare your MA to the simple EA from the lectures by running both algorithms for for 1500 generations. Please repeat each experiment 10 times to account for stochasticity, and perform this comparison for both datasets, asking: does addition of local search improve performance?
2. Do you think the simple EA with the same number of generations is a fair baseline to answer this question? Why/why not? If your answer is no, then please explain how you would perform an experiment with a fair comparison.
3. In general, on the TSP problem, are memetic algorithms more effective than simple EAs? To answer this question, please rely on (recent) results from the literature, and include references to the papers used.

Exercise 4.4 Exploitation vs exploration (4 points)

Implement a string search GA like the one in the lecture, but now with:

- tournament selection with a tunable parameter K,
- your own target string of length L between 10 and 20 characters,
- an alphabet Σ containing all 26 lowercase letters and a space character,
- crossover with probability $p_c = 1$,
- a tunable mutation rate μ,
- a population size $N = 1000$, and
- a fitness as defined in the lecture

1. Using $K = 2$ and $\mu = 0.01$, run the algorithm 20 times to measure the time t_{finish} (in generations) needed to find the target. Visualize the distribution to see both the variation and average of t_{finish}.
2. Set $\mu = 0$ and repeat the experiment. Do you find the target? Explain your observations. (Hint: you may want to stop trying after some predefined number G_{max} of generations...)
3. Set $\mu = 0.1$ and repeat the experiment. Do you find the target? Explain your observations.
4. Starting from $\mu = 0$, gradually increase μ in small steps and make a plot of the relationship between μ and t_{finish} (visualizing both the average and some measure of the variation between runs in your figure). What optimal μ do you find? (Please provide both the figure and all relevant information, such as parameter values and the target string, in your assignment).
5. Now set $K = 5$ and repeat the above experiment. What do you find? Explain your observations.