Evolutionary Computation
Practical Assignment 1

1 Genetic algorithm

In this practical assignment you need to implement a genetic algorithm (GA) and test it on three functions. To goal is to get insight into the convergence behavior of GAs so you need to track and trace some population measures during the run.

The GA we use is based on family competition: this means that 2 parent solutions generate 2 offspring solutions (using crossover) and the best 2 of 4 these solutions become members of the next generation. More specifically, one generation t of this GA looks like:

1. Randomly shuffle the population $P(t)$.
2. Pair solution 1 with solution 2, solution 3 with solution 4, etc. ...
3. Each parent pair creates 2 offspring solutions using crossover.
4. Family competition: the best 2 solutions of each family of 4 are copied to the next population $P(t+1)$.

When a parent and a child have the same (best) fitness, the child is copied to the next generation. Recombination is done with either uniform crossover (UX) or 2-point crossover (2X). Note that we do not use mutation in these experiments.

2 Fitness functions

The functions are defined over the binary domain. The goal is to find the binary vector that maximizes the function value; in each case the optimal solution is the string of all ones. We assume a stringlength $\ell = 100$.

1. Counting Ones Function:

$$x_i \in \{0,1\} : CO(x_1 \ldots x_\ell) = \sum_{i=1}^{\ell} x_i$$

2. Trap Functions (tightly linked):

$$TF(x_1 \dots x_\ell) = \sum_{j=0}^{\frac{\ell}{k}-1} B(x_{jk+1} \dots x_{jk+k})$$

with:

$$B(x_1 \dots x_k) = \begin{cases} k & \text{if} \quad CO(x_1 \dots x_k) = k \\ k - d - \frac{k-d}{k-1} CO(x) & \text{if} \quad CO(x_1 \dots x_k) < k \end{cases}$$

(a) Deceptive Trap Function: $k = 4, \;\; d = 1$

Number of Ones	4	3	2	1	0
Fitness Value	4	0	1	2	3

(b) Non-deceptive Trap Function: $k = 4, \;\; d = 2.5$

Number of Ones	4	3	2	1	0
Fitness Value	4	0	0.5	1.0	1.5

Both Trap functions consist of 25 concatenated subfunctions of length $k = 4$, each having an optimum at 1111 and at 0000. The overall function has therefore $2^{25} - 1$ local optima and 1 global optimum (= the string of all ones). The difference between the two trap functions is the difference d between the values of the two optima at each substring. As a result of this fitness difference the first trap function is so-called, fully deceptive, while the second is not. Deceptive functions are functions where the lower-order schema fitness averages that contain the local optimum 0000 have a higher value than the lower-order schema fitness averages that contain the global optimum 1111 in each subfunction.

(a) Deceptive Trap Function:

$$\begin{cases} F(111*) &=& \frac{4+0}{2} &= 2 \\ F(000*) &=& \frac{3+2}{2} &= 2.5 \end{cases}$$

$$\begin{cases} F(11**) &=& \frac{4+0+1}{4} &= 1.25 \\ F(00**) &=& \frac{3+2.2+1}{4} &= 2 \end{cases}$$

$$\begin{cases} F(1***) &=& \frac{4+0+3.1+2}{8} &= 1.125 \\ F(0***) &=& \frac{3+3.2+3.1+0}{8} &= 1.5 \end{cases}$$

(b) Non-deceptive Trap Function:

$$\begin{cases} F(111*) &=& \frac{4+0}{2} &= 2 \\ F(000*) &=& \frac{1.5+1}{2} &= 1.25 \end{cases}$$

$$\begin{cases} F(11**) &= \frac{4+0+0.5}{4} &= 1.125 \\ F(00**) &= \frac{1.5+2(1)+0.5}{4} &= 1 \end{cases}$$

$$\begin{cases} F(1***) &= \frac{4+0+3(0.5)+1}{8} &= 0.813 \\ F(0***) &= \frac{1.5+3(1.0)+3(0.5)+0}{8} &= 0.75 \end{cases}$$

3. Trap Functions (not linked):

   In the above tightly linked Trap function the 25 subfunctions are placed adjacent to each other on the bit string.

   To investigate the impact of linkage we also look at the Trap function where the subfunctions are maximally spread out over the string.

   (a) The first subfunction has its four defining bits at positions 1, 26, 51, and 76 in the bit string.

   (b) The second subfunction has its four defining bits at positions 2, 27, 52, and 77 in the bit string.

   (c) The third subfunction has its four defining bits at positions 3, 28, 53, and 78 in the bit string.

   (d) ...

   (e) The 25^{th} subfunction has its four defining bits at positions 25, 50, 75 and 100 in the bit string.

3 Experiments

1. Experiment 1: Counting Ones Function.
   Run 2 experiments each with a different crossover operator: 2X and UX.

2. Experiment 2: Deceptive Trap Function (tightly linked).
   Run 2 experiments each with a different crossover operator: 2X and UX.

3. Experiment 3: Non-deceptive Trap Function (tightly linked).
   Run 2 experiments each with a different crossover operator: 2X and UX.

4. Experiment 4: Deceptive Trap Function (not linked).
   Run 2 experiments each with a different crossover operator: 2X and UX.

5. Experiment 5: Non-deceptive Trap Function (not linked).
   Run 2 experiments each with a different crossover operator: 2X and UX.

The population size N is an important parameter for genetic algorithms. We only consider multiples of 10 and use bisection search to find the minimal required population size. Start with $N = 10$ and if successful - this is, the global optimum has been found - you are done. If not successful, double the population size ($N = 20$) and try again. Keep doubling until the optimal solution is found or until the population has reached a maximum of $N = 1280$. When a population size is found that is successful, bisection search is applied to find the smallest population necessary. For instance, say $N = 250$ is the minimal population size needed to solve a problem reliably, then the bisection search would try the following sequence of population sizes: $N = 10, 20, 40, 80, 160, 320, 240, 280, 260, 250$.

GAs are stochastic algorithms so we perform multiple runs to average out the stochastic effects. We consider a problem to be solved reliably when 24 out of 25 independent runs find the optimal solution. The GA is stopped when one of the new offspring solutions is the global optimum, or when no new offspring solution with higher fitness than its parents has been created during a complete generation.

4 Deliverables

You need to email **two separate** files:

1. A report in .pdf form
2. A zip archive file of the source code

The report should describe:

- what did you do (e.g.: which programming language, CPU used)
- what did you observe
- what conclusions can you draw from your results

The report should contain **5 tables** (one for each test function) reporting the minimal population size for which 24 out of the 25 runs are successful, the average number of generations for this minimal population size, the average number of fitness function evaluations for this minimal population size, and finally the average CPU time (seconds) required for running the GA (25 times) with this minimal population size. Averages are taken over the 25 runs with the minimal required population size found by the the bisection method. Also show - between parentheses - the corresponding standard deviation for each average number over the 25 runs. If the GA did not find the global optimum with the maximum population size $N = 1280$, simply report FAIL in the table.

In addition, you need to trace and plot the following measures of the population when optimizing the Counting Ones function with a population size $N = 250$ (no need to average, a single run is sufficient):

1. Plot the proportion $prop(t)$ of bits-1 in the entire population as a function of the generation t. If the GA converges to the global optimum this value should change from $prop(t = 0) = 0.5$ to $prop(t = t_{converged}) = 1.0$.

2. Plot the number of selection errors $Err(t)$ and the number of correct selection decisions $Correct(t)$ that are made as a function of the generation t. A selection error occurs whenever two parents have a different bit (1 or 0) at a position i and the winners of the family competition have both a bit-0 at that particular position i. Similarly, a correct selection decision occurs whenever two parents have a different bit (1 or 0) at a position i and the winners of the family competition have both a bit-1 at that position i. So in each generation you need to check $\ell \times \frac{N}{2}$ times for a possible selection error or correct selection decision ($\ell = 100$, ie. the string length, $N =$ population size).

3. Consider the two competing schemata $1******...*****$ versus $0******...*****$. Plot - as a function of the generation t - the number of solutions in the population that are member of the respective schemata (their sum should always equal the population size N). Plot - again as a function of the generation t - the schema fitness and the schema standard deviation, in other words, compute the average fitness and the standard deviation of the solutions that are a member of the $1******...*****$ (resp. $0******...*****$) schemata.