

Evolutionary Computation 2007: Home problems, set 1

General instructions. READ CAREFULLY!

Problem set 1 consists of four parts. Problem 1.1 is mandatory, the others voluntary (but check the requirements for the various grades on the web page). After solving the problems, collect your answers and your programs in *one* zip file which, when opened, generates one folder for each problem (e.g. Problem 1.1, Problem 1.2 etc.) in the assignment. Make sure to keep copies of the files that you hand in!

You should provide a brief report in the form of a PDF, PS or text file. In the case of analytical problems, make sure to include all the steps of the calculation in your report, so that the calculations can be followed. Providing only the answer is *not* sufficient. Whenever possible, use symbolical calculations as far as possible, and introduce numerical values only when needed.

In *all* problems requiring programming, use Matlab (v.6 or v.7). The *complete* Matlab program for the problem in question (i.e. all source files) must be handed in, collected in the same folder. In addition, *clear* instructions concerning how to run the programs *should* be given in the report. It should *not* be necessary to edit the programs, move files etc. Programs that do not function or require editing to function will result in a deduction of points.

The maximum number of points for problem set 1 is 13. Incorrect problems will *not* be returned for correction, so please make sure to check your solutions and programs carefully before e-mailing them to `mattias.wahde@chalmers.se`.

You may, of course, discuss the problems with other students. However, each student *must* hand in his or her *own* solution. In obvious cases of plagiarism, points will be deducted from all students involved. Don't forget to write your name and civic registration number on the front page of the report!

(Strict) deadline: 20070926

Problem 1.1, 4p, Basic GA program

a) Write a standard GA as defined on p. 48 in Chapter 2. You are welcome to use the code from Chapter 2, but remember that a standard GA uses different operators to some extent. In addition to the main program, you should write Matlab functions (placed in *separate* M-files) for

1. initializing a population (`initpop.m`),
2. decoding a (binary) chromosome (`decode_chromosome.m`),
3. evaluating an individual (`evaluate_individual.m`),
4. ranking fitness values for the whole population (`rank_fitness.m`),
5. selecting individuals with roulette-wheel selection (`roulette_wheel_select.m`),
6. carrying out crossover (`crossover.m`)
7. carrying out mutations (`mutate.m`).

Hint: For fitness ranking, use the `sort` function in Matlab!

b) Next, as a test of your GA, find (and report) the *minimum* value of the function

$$g(x_1, x_2) = \left(1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right) \times \left(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right) \quad (1)$$

in the interval $x_1, x_2 \in [-5, 5]$ as well as the location (x_1^*, x_2^*) of the minimum. Define the fitness function f as $1/g(x_1, x_2)$. Select (and report) appropriate parameters for the GA (e.g. by trial-and-error). Use at least 20 genes (bits) per variable in the chromosomes.

Check carefully that you enter the function $g(x_1, x_2)$ correctly. Hint: $g(1, 1) = 1876$.

Note: the important thing is the *actual* minimum, not just the output from your program! In order to make sure that you have actually found the minimum, you may wish to carry out a few steps of gradient descent, starting from the best point found by the GA.

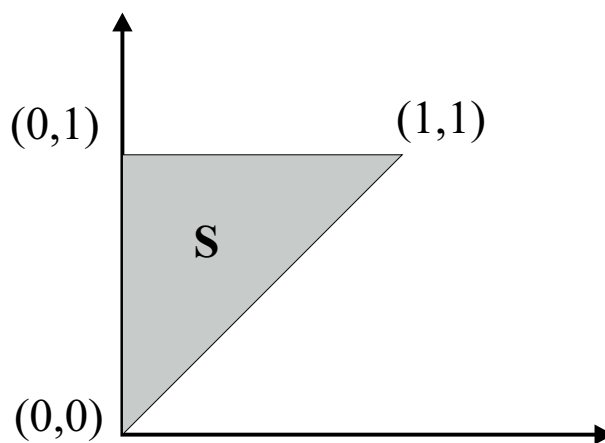


Figure 1: The set S used in Problem 1.1a.

Problem 1.2, 4p, Classical optimization

a) Use the analytical method described on p. 21 in Chapter 1 to determine the global minimum of the function

$$f(x_1, x_2) = 4x_1^2 - x_1x_2 + 4x_2^2 - 6x_2, \quad (2)$$

on the (closed) set S , shown in the figure. The corners of the triangle are located at $(0, 0)$, $(0, 1)$ and $(1, 1)$.

b) Use the penalty method (pp. 22-24) to find the minimum of the function

$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2, \quad (3)$$

subject to the equality constraint

$$x_1^2 - x_2 = 0. \quad (4)$$

Use gradient descent, starting from a suitable initial point $\mathbf{P} = (n_1, n_2)^T$, where n_1 and n_2 are *integers* (of your choice) to determine the minimum of the function $f_p(\mathbf{x}; \mu)$ for $\mu = 0.3, 1, 3, 10, 100$, and 1000 (and, if you wish, for larger values of μ as well). For each value of μ present (in a table) (1) the location of the minimum $\mathbf{x}^*(\mu)$, (2) the function value $f(\mathbf{x}^*(\mu))$ at the minimum, (3) the penalty term $p(\mathbf{x}^*(\mu); \mu)$.

Hint: It is a good idea to plot the function, and the constraint, to get a rough estimate of where the minimum might be located.

Problem 1.3, 3p, The 3-parity problem

Exclusive or (XOR) is a commonly used logical operator, which takes two inputs and for which the output y is equal to 1 if exactly one of the inputs x_1 or x_2 is equal to 1, and 0 otherwise. Thus, the **truth table** for XOR is given by

| x_1 | x_2 | y |
|-------|-------|-----|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

XOR can be generalized to n inputs, and the resulting Boolean function is called an n -**parity function**. These functions give the output 1 if an odd number of inputs are equal to 1, and 0 otherwise. Parity functions can be represented by feedforward neural networks (FFNNs), containing n inputs, n neurons in the hidden layer, and one output. For a given value of n , the number of parameters (weights and biases) in such a network is equal to $(n + 1)^2$.

Using a GA of your choice, evolve a discrete-time FFNN with 3 inputs, 3 hidden neurons, and one output, which generates the 3-parity function. The neurons in the FFNN should use the squashing function

$$\sigma(z) = \frac{1}{1 + e^{-cz}}, \quad (5)$$

for some suitable value of c ($\sim 1.0 - 3.0$). Let the fitness be the inverse of the mean-square deviation Δ between the desired output d and the actual output y , i.e. set $f = 1/\Delta$, where

$$\Delta = \sqrt{\frac{1}{m} \sum_{i=1}^m (d(i) - y(i))^2}, \quad (6)$$

where $m = 8$ for $n = 3$. Set the cutoff fitness f_c (for the EA runs) such that runs are terminated when Δ is equal to 0.01. (In case the evolved networks fail to generate the correct output¹ even when the fitness cutoff is reached, you may wish to use a different fitness measure, which is more focused on the *worst* output. For example, you may use the p^{th} root (with $p > 2$) of the average of the p^{th} powers of the differences $d(i) - y(i)$).

In your report, describe the encoding scheme (e.g. binary, real-number) that you have used and specify the operators that have been used for selection, crossover, and mutation, as well as the values of the various GA parameters (e.g. p_c , p_{mut} , population size etc.) and the value of c (see Eq. (5)). Note: for values of c around 1, a suitable range for the weights (and biases) is approximately $[-10, 10]$. In addition to your GA (i.e. the complete Matlab program), you should also provide (in your report) a plot over the average and maximum fitness as a function of the number of evaluated individuals. Furthermore, you should provide a test program where the user may enter three bits and then obtain the output from your *best* network. The interface to the test program should be *exactly* as follows

```
output = testnetwork(x1,x2,x3),
```

where x_1 , x_2 , x_3 are the three input bits. Encode the best network directly into the test program, so that the program runs directly, without the use of any additional input files etc.

¹Definition of correct output: if, for a given input signal, the desired output is equal to 0, an output of 0.02 or less will be considered correct. If instead the desired output is equal to 1, an output of 0.98 or more will be considered correct.

Problem 1.4, 2p, Analytical properties of GAs

Assume that a genetic algorithm with binary encoding is to be used in a case where the fitness function is given by

$$f(k) = k(n - k), \quad (7)$$

where k is the number of ones in the chromosome and n is the chromosome length, which is much larger than one, but finite. The population size is assumed to be infinite, and it is further assumed that the initialization is random so that, in the first generation, the probability distribution is

$$p_1(k) = 2^{-n} \binom{n}{k}, \quad (8)$$

Compute

1. The average fitness in the first generation,
2. The probability distribution $p_2(k)$ in the second generation, i.e. after evaluation and (roulette-wheel) selection.
3. The average fitness in the second generation. Simplify the answer as much as possible - it *should* be given in the form of a polynomial in n , i.e. $\bar{f}_2 = a + bn + \dots$, where a, b, \dots are constants (i.e. independent of n).

Note! Consider only selection - you may neglect crossover and mutation.