Artificial Intelligence 2, 2010: Home problems

General instructions. READ CAREFULLY!

The problem set consists of five parts. Problems 1 and 4 are 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 named `LastnameFirstname.zip`, which, when opened, generates one folder for each problem (e.g. `Problem1`, `Problem2` 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 or PS 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 easily. 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, you *must* use either Matlab or Microsoft Visual Studio .NET (or the 2008 (2010) Express Editions) for C-sharp or C/C++. The *complete* program for the problem in question (i.e. all source files, project files etc.) must be handed in, collected in the same folder. In addition, *clear* instructions concerning how to run the programs *must* be given in the report. It should *not* be necessary to edit the programs, move files etc, in order to run your programs. Programs that do not function or require editing in order to function will result in a deduction of points!

The maximum number of points for the problem set is 25. Incorrect problems will *not* be returned for correction, so please make sure to check your solutions and programs carefully before e-mailing them to `krister.wolff@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!

Deadline: 2010-12-10, no later than 24.00!

NOTE: Make sure to follow the instructions above, as well as the specific instructions for each problem below, as a failure to do so may result in a deduction of points! Please, note also that submissions after the deadline (i.e. after 24.00) will result in deduction of points.

Problem 1: Basic GA program (5p)

a) Write a standard genetic algorithm (GA) using (some of) the components described in Sect. 3.2.1 of the course book. It is strongly recommended to start from the program that you have written in connection with the Matlab introduction (available on the web page).

Your task is to implement functions for

1. initializing a population (`InitializePopulation`),
2. decoding a (binary) chromosome (`DecodeChromosome`),
3. evaluating an individual (`EvaluateIndividual`),
4. selecting individuals with tournament selection (`Tournamentselect`),
5. carrying out crossover (`Cross`)
6. carrying out mutations (`Mutate`).

Your program should employ elitism, i.e. in each generation, a single copy of the best individual should be copied (unchanged) to the next generation. Make sure that you understand how it works. Furthermore try to follow the coding standard (availible on the web page) as closely as possible. It will help you to reduce typical coding errors.

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

$$\begin{aligned} g(x_1, x_2) \;=\; & \big(1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\big) \times \\ & \big(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\big) \qquad (1) \end{aligned}$$

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 $1/g(x_1, x_2)$.

Since the goal of this problem is for you to familiarize yourselves with GAs, you should try several (at least five) different parameter settings, using the guidelines provided in Examples 3.5 - 3.7 in the course book. For each tested parameter set, provide (in your report) a table listing the parameter values as well as the average results (over at least 20 runs) obtained for the parameter set in question. Use at least 25 genes (bits) *per variable* in the chromosomes.

Check carefully that you enter the function $g(x_1, x_2)$ correctly. Hint: $g(2, 1) = 2275$.

Problem 2: The traveling salesman problem (5p)

The traveling salesman problem (TSP) has many applications in e.g. network routing and placement of components on circuit boards. In this problem, you will solve the TSP in a case where the cost function is simply taken as the length of the route. Write a GA that can search for the shortest route between N cities, using permutation coding for the path. Use an encoding method such that the chromosomes consist of lists of integers determining the indices of the cities. Examples of five-city paths starting in city 4 are e.g. (4,3,1,2,5), (4,1,5,2,3), (4,5,1,2,3) etc. The first chromosome thus encodes the path $4 \to 3 \to 1 \to 2 \to 5 \to 4$. The fitness should be taken as the inverse of the route length, calculated using the ordinary cartesian distance measure. The program should always generate syntactically correct routes, i.e. routes in which each city is visited once and only once until, in the final step, the tour ends by a return to the starting city.

Specialized operators for crossover and mutation are needed in order to ensure that the paths are syntactically correct. Use order crossover as described on p. 147 in the course book. Your program *must* contain a separate function `OrderCrossover` with the following interface

```
function [c1new, c2new] = OrderCrossover(c1,c2);
```

where `c1`, `c2`, `c1new`, and `c2new` are chromosomes (paths). For mutations use the swap mutations defined on p. 147 in the course book. Solve the following problems:

a) In the TSP, paths that start in different cities but run through the cities in the same order are equivalent (in the sense that the path length is the same). Furthermore, paths that go through the same cities in opposite order are also equivalent. Thus, for example, the paths $(1, 2, 3, 4, 5)$ and $(2, 3, 4, 5, 1)$ are equivalent, as are $(1, 2, 3, 4, 5)$ and $(5, 4, 3, 2, 1)$. Paths that are *not* equivalent are called *distinct paths*. How many distinct paths are there in the general case of N cities? Show *clearly* how you arrive at your answer!

b) Use your program to search for the shortest possible path between the cities whose coordinates are given in the file `loadcitylocations.txt` availible for download on the web page

This file contains a 50×2 matrix with the coordinates (x_i, y_i) for city i, $i = 1, \ldots, 50$. In addition to the full program, send also the shortest path you have found, in electronic format, i.e. in a text file, containing a vector with the city *indices* for the path in question. Note that the indices *should* be in the interval $[1, 50]$, *not* $[0, 49]$. For example, a path may be given as

`bestpath = [4 7 11 39 50 41 3 ...` etc.

The length of the path will be tested using the vector that you provide. Finally, in your report, draw the shortest path, and specify its length. For full points, it is required that the length of your shortest path should be less than 150 length units.

Problem 3: The 3-parity problem (5p)

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), with 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

$$\sigma(z) = \frac{1}{1+\mathrm{e}^{-cz}} \tag{2}$$

for some suitable value of c (around 1 to 3). 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

$$\Delta = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(d(i)-y(i)\right)^2} \tag{3}$$

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 the case the evolved networks fail to generate the correct output[1] 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. 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 program) you should also provide (in your report) a graph 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),
```

[1]Definition 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 consoderd correct.

where $x1, x2, x3$ 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.

Problem 4: Particle swarm optimization (5p)

Particle swarm optimization (PSO) is a stochastic optimization method based on the properties of swarms, such as bird flocks, fish schools etc. Implement a standard PSO algorithm, as described in Chapter 5, in the programming language of your choice (Matlab, C-sharp, C, or C++). Note: The standard PSO algorithm *should* include the inertia weight, see p. 128 and Eq. (5.20) in the course book! Remember to follow the coding standard.

Next consider the function

$$f(x,y) = 1 + (-13 + x - y^3 + 5y^2 - 2y)^2 + (-29 + x + y^3 + y^2 - 14y)^2 \tag{4}$$

Use your PSO to find the *minimum* of f. Let the variables x and y vary in the range $[-10, 10]$.

Problem 5: Analytical properties of GAs (5p)

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

$$f(k) = k(n-k) \tag{5}$$

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

$$p_1(k) = 2^{-n}\binom{n}{k} \tag{6}$$

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, i.e. it *should* be given in the form of a polynomial in n, i.e. $\bar{f}_2 = a + bn + \ldots$, where a, b, $\ldots$ are constants (i.e. independent of n).

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