Artificial Intelligence 2, 2007: 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 `lastname_firstname.zip`, which, when opened, generates one folder for each problem (e.g. `Problem_1`, `Problem_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. The *complete* 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 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 `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!

NOTE: Please 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!

Deadline: 20080107

Problem 1, 5p, Basic GA program

a) Write a standard GA as defined on p. 22-23 in Chapter 2 in *Basics of evolutionary algorithms*. You are welcome to use the code from Chapter 3, 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

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

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: $g(-1, 1) = 87100$.

Note: the important thing is the *actual* minimum, not just the output from your program!

Problem 2, 5p, The traveling salesman problem

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 (Hint: use the command `randperm` in Matlab to generate such chromosomes). 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 $4 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 4$. The fitness should be taken as the inverse of the route length (calculated using the ordinary cartesian distance measure, i.e. *not* the city-block distance measure). The program should always generate syntactically correct routes, i.e. routes in which each city is visited once and only once until (NOTE!), 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 in problem 5.3b. Your program *must* contain a separate function `order_crossover` with the following interface

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

where `c1`, `c2`, `c1new`, and `c2new` are chromosomes (paths). For mutations use the swap mutations defined in problem 5.3a. 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?

(b) Use your program to search for the shortest possible path between the cities whose coordinates are given in the file

`www.me.chalmers.se/~mwahde/courses/isd/ai2/2007/TSPcities2.m`

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 or a Matlab file (.mat), 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, 5p, The n-parity problem using backpropagation

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. Write a backpropagation program and use it to find a network that can represent

a) The 2–parity (XOR) function,
b) The 3–parity function,
c) The 4–parity function.

For each $n = 2, 3, 4$, use an $n - n - 1$ network. 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). You may start from the file `BP.m`, available at

`www.me.chalmers.se/~mwahde/courses/isd/ai2/2007/BP.m`

The maximum error ϵ_{tot} (computed efter *complete* training epochs, as indicated in the file `BP.m`) should not exceed 0.01. ϵ_{tot} is defined as

$$\epsilon_{\text{tot}} = \sqrt{\frac{1}{2^n} \sum_{m=1}^{2^n} \epsilon_m^2}, \tag{3}$$

where $\epsilon_m^2 = (o - y_1^o)^2$ denotes the error for input-output pair m (o is the desired output and y_1^o the actual output).

You should provide the best networks in electronic format (as a text file or a .mat-file), as well as (NOTE!) a test program where the user can enter two (a), three (b), or four (c) input elements and obtain the output from your best networks. Encode the best networks (for $n = 2, 3, 4$) in the test program so that it can run directly without having to load any input files. Also, you should indicate the value of the error (over the whole training set), for each network.

Problem 4, 5p, Particle swarm optimization

Particle swarm optimization (PSO) is a stochastic optimization method based on the properties of swarms, such as bird flocks, fish schools etc. Implement a basic PSO (as described in the handout) in Matlab. Remember to place separate Matlab functions in separate files.

a) 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 $[-10, 10]$.

b) Modify your PSO so that it handles integer programming (see the handout) and use the modified program to find a global minimum (there are two) of the function

$$f = -(15 \;\; 27 \;\; 36 \;\; 18 \;\; 12)\mathbf{x} + \mathbf{x}^T \begin{pmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{pmatrix} \mathbf{x}, \tag{5}$$

where $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)^{\mathrm{T}}$, and $x_i \in \{-30, -29, \ldots, 29, 30\} \in \mathbf{Z}$. Note that the function is positive for many values of $\mathbf{x}$, but that the global minimum value $f^* < 0$.

Problem 5, 5p, 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

$$f(k) = k(n - k), \tag{6}$$

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

$$p_1(k) = 2^{-n}\binom{n}{k}, \tag{7}$$

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. $\bar{f}_2 = a + bn + \ldots$, where a, b, ... are constants (i.e. independent of n).

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