M057 and GI06
Homework #1
(Due 16 Nov 2012)

Coursework Summary

Abstract: This coursework is designed to familiarise you with two basic evolutionary computation techniques, genetic algorithms and genetic programming. You are required to implement basic `GA` and `GP` algorithms. However, as your goal is to get as *optimal* solutions as possible, you may modify the algorithms as you see fit or and additionally try other combinatorial optimisation algorithms. You will turn in a final report consisting of separate reports for each problem.
Programming: These are to be coded from "scratch" so no evolutionary computation library or code that is not your own is acceptable. You may, however, use a library or public domain code for tree manipulations if desired. The choice of programming language is yours.
Collaboration: If desired you may collaborate with one other student and give a single joint report.

Problem A [50pts]

Aim: The aim of this exercise is to use a genetic algorithm to find low cost solutions to the traveling salesman problem (TSP).
Exercise: Implement a genetic algorithm for solving the TSP problem. You may choose use any programming language and at a minimum to use the *ox* crossover operator (see handout) and to use the reciprocal exchange/*swap* operator for mutation. You are also encouraged experiment with your own variations of these algorithms.

When generating your results i) give the best results that you can generate using only the *ox* and *swap* operators, ii) give the best results from any additional techniques you design or those already in the literature.

Requirements:

1. Run a GA algorithm for TSP with *ox* and *swap* operators.
2. Run your algorithm on the four datasets `tspadata1.txt, tspadata2.txt, tspadata3.txt, tspadata4.txt` (the datasets are assymetric adjacency matrices a[i,j] is the cost to travel from city i to city j).

   `http://www.cs.ucl.ac.uk/staff/M.Herbster/4C57/data/tspadata1.txt (17 cities)`
   `http://www.cs.ucl.ac.uk/staff/M.Herbster/4C57/data/tspadata2.txt (48 cities)`
   `http://www.cs.ucl.ac.uk/staff/M.Herbster/4C57/data/tspadata3.txt (100 cities)`
   `http://www.cs.ucl.ac.uk/staff/M.Herbster/4C57/data/tspadata4.txt (999 cities)`

   Note: The optimal tour for `tspadata1.txt` *has a cost of 39.*
3. Design, implement and discuss two experiments. Here you must design two separate experiments designed to elucidate methods that may be applied to finding solutions to the TSP problem with genetic algorithm. A few ideas are listed below.
   (a) Compare roulette wheel to tournament selection
   (b) Compare a pure hill-climbing approach to a GA
   (c) Compare random search to a GA
   (d) Compare the PMX operator to the OX operator
   (e) Your own TSP solver versus a GA

Hand in [Problem A]:

1. Please print only the 5 "most significant sheets" the remaining source should be e-mailed.
2. The *output of your algorithm* which must include the vertex list which defines the tour and the cost of that tour. (Only include your grid vertex list in e-mail)
3. A *report* which explains the algorithms and techniques used and the experiments performed to obtain the best results.
4. For the grid problem please plot your Hamiltonian cycle on a grid.
5. The particular parameters and results of your *best runs* on each dataset. Also e-mail the vertex list and cost of your best runs in the following format in the *body* (no attachments or html e-mail) of your e-mail (**Subject: gatsp BASIC results**)

   ```
   firstname lastname tspadata1.txt v1 v2 ...  v17 cost
   firstname lastname tspadata2.txt v1 v2 ...  v48 cost
   firstname lastname tspadata3.txt v1 v2 ... v100 cost
   ```

   remember that cost $= \sum_{i=1}^{n-1} a[v_i, v_{i+1}] + a[v_n, v_1]$.
   Then a second e-mail (**Subject: gatsp D999 results**)

   ```
   firstname lastname tspadata4.txt v1 v2 ... v999 cost
   ```

Grading:

1. Correctness : You must turn in code and results that generates a solution to the smallest TSP (17 cities) using only the *ox* and *swap* operators. *It is not necessary for the other problems to find the absolute minimal solutions, however you are expected to find* **"good"** *solutions.*

2. Quality : significantly shorter tours will receive more points. Thus do not arbitrarily choose parameters, experiment a bit to find a good parameter setting. To generate the best results you may need to try additional operators other than *ox* and *swap* operators
3. Clarity : clear well-documented source code may receive more points.
4. Report includes (**basic methods**): you should write a description of your methods this should be at least 1 page and include at a minimum which operators were used and what parameters and how these parameters were set as well as the the *selection* function. The idea here, is that your description should allow one to replicate your experiments.
5. Report includes (**experiments chosen**): you should write-up the experiment done for "requirement 3" above.
6. Marks are given based on the clarity, and scientific quality of the report.

Problem B [50 pts]

Aim: Implement a simple genetic programming system from scratch.

Multiplexers

Exercise: Mitchell: Computer Exercises (Chapter 2) #1 pp 81 (Part I).

> Implement a genetic programming algorithm and use it to solve the "6-multiplexer" problem (Koza 1992). In this problem there are six Boolean-valued terminals, $\{a0, a1, d0, d1, d2, d3\}$, and four functions, `AND, OR, NOT, IF`. The first three functions are the usual logical operators, taking two, two, and one argument respectively, and the `IF` function takes three arguments. `(IF X Y Z)` evaluates its first argument `X`. If `X` is true, the second argument `Y` is evaluated; otherwise the third argument `Z` is evaluated. The problem is to find a program that will return the value of the d terminal that is addressed by the two a terminals. E.g., if $a0 = 0$ and $a1 = 1$, the address is 01 and the answer is the value of $d1$. Likewise, if $a0 = 1$ and $a1 = 1$, the address is 11 and the answer is the value of $d3$. The fitness of a program should be the fraction of correct answers over all 64 possible fitness cases (i.e., values of the six terminals).

You may develop any elaboration of the "Koza" GP, the aim is to use a "fair" technique in order to the maximize your test set performance. Finally, when "optimizing" you may wish to use only partial "test" sets however, when reporting performance you should report the performance on the full test set.

Part I [30/50 pts]:
Use genetic programming to discover the solution to the 6-multiplexer problem. (see Mitchell #1 p81)

Part II [15/50 pts]:
Use genetic programming to discover a solution to the 11-multiplexer (3 address, 8 data lines) problem. You may find it computationally expensive to use the full test set on this problem (2048 cases), thus you may find it necessary to develop a technique which only use a fraction (changing) of the test cases each generation.

Part III [5/50 pts]:
Use genetic programming to discover a solution to the *16-middle-3* problem. The correct logic for *16-middle-3* given input $\boldsymbol{x} \in \{0,1\}^{16}$ then

$$\text{16-middle-3}(\boldsymbol{x}) := \begin{cases} 0 & \text{not}(7 \leq \sum_{i=1}^{16} x_i \leq 9) \\ 1 & 7 \leq \sum_{i=1}^{16} x_i \leq 9 \end{cases}$$

Thus e.g., 16-middle-3(1000100100111001) = 1 and 16-middle-3(11111111111110111) = 0. Speed issues as above but now are exacerbated with 65536 test cases.

Requirements:

1. Run a GP algorithm for the multiplexors and 16-middle-3 problem.

Hand in:

1. The source code of your algorithm. Please print only the 5 "most significant sheets" the remaining source should be e-mailed.
2. Sample runs of your algorithm (include timings).
3. Your best solutions clearly noting the fraction of test cases correctly classified.
4. A report (2+ pages) describing your implementation (discussion of data structures), pseudo-code for your algorithm, a description of the experiments (to determine parameter choices, etc) performed find a solution, and finally any interesting design choices.

Grading:

1. Marks are given based on the clarity, and scientific quality of the report.
2. Marks are given for the correctness of solutions.