ECM3412 Nature-Inspired Computation
Assignment One: Bin-Packing with an Evolutionary Algorithm

This is an INDIVIDUAL assignment and is worth 20% of the module
Handout: w/b 20th October 2008
Handin: 12.00noon Thursday 20th November – On Paper, To the Education Office
Back to you: w/b 1st December

You are reminded of the University's Regulations on Collaboration and Plagiarism, details of which are available on the School web page

What you will do in this assignment is implement a simple evolutionary algorithm. You will apply it to a bin-packing problem. You will do a variety of experiments to help find out what parameters for the EA are best in this case. Implementation can be in the programming language of your choice. In the following sections, basic details of the problem are provided, then basic details of the algorithm, and finally a step by step guide of what you are expected to do. The final section indicates what should be in your report to be handed in.

The Bin-Packing Problem

The bin-packing problem (BPP) involves *n* items, each with its own weight. Each item must be placed in one of *b* bins. The task is to find a way of placing the items into the bins in such a way as to make the total weight in each bin as nearly equal as possible. Eg, if there are 6 items with weights as follows: (17, 12, 19, 6, 4, 28) and we place them into 3 bins as follows:

(bin1: 1, 3) (bin2: 4) (bin3: 2, 5, 6)

then bin1 has weight 36 (17+19), bin2 has weight 6, and bin3 has weight 44 (12+4+28). These are quite different, so it is a poor solution. We can measure the solution quality by taking the difference *d* between the heaviest and lightest bins – in this case *d* = 38. What we want to do here is to minimise this difference. A far better solution in this case is:

(bin1: 1, 2) (bin2: 3, 4, 5) (bin3: 6)

with *d* = 1. We choose a representation as follows (there are many other possible ways to design the representation, but I expect you to use this one.) A chromosome (candidate solution to the problem) is represented as a list of *n* numbers, where each number can be anything in the range from 1 to *b*. For example, the two candidate solutions above would be represented as 1 3 1 2 3 3 and 1 1 2 2 2 3. Basically, if the *k*th number in the chromosome is *i*, then this means item number *k* is in bin *i*.

The fitness of a chromosome is worked out by calculating the difference *d* between the heaviest and lightest bins. There are two specific bin-packing problems I want you to address. In each, there will be 500 items to be packed into a number of (*b*) bins (either 10 or 100). In problem BPP1, *b* = 10 and the weight of item *i* (where *i* starts at 1 and finishes at 500) will be $\mathbf{2i}$. In problem BPP2, *b* = 100 and the weight of item *i* will be $\mathbf{i^2}$.

Hence, in the case of BPP1, a chromosome which starts:

3236… etc…

indicates that item 1 (weight 2) is in bin 3, item 2 (weight 4) is in bin 2, item 3 (weight 6) joins item 1 in bin 3, item 4 (weight 8) is in bin 6, and so on.

The Evolutionary Algorithm

Implement the EA like this:

1. Generate an initial population of *p* randomly generated solutions, and evaluate the fitness of everything in the population.
2. Use the binary tournament selection (with replacement) **twice** to select two parents *a* and *b*.
3. Run single-point crossover on these parents to give 2 children, *c* and *d*.
4. Run mutation on *c* and *d* to give two new solutions *e* and *f*. Evaluate the fitness of *e* and *f*.
5. Run weakest replacement, firstly for *e*, then *f*.
6. If a termination criterion has been reached, then stop. Otherwise return to step 2.

Termination Criterion: Will simply be having reached a maximum number of fitness evaluations. The result is then the fitness of the best chromosome in the population at the end.

Binary Tournament Selection: Randomly choose a chromosome from the population; call it *a*. Randomly choose another chromosome from the population; call this *b*. The fittest of these two (breaking ties randomly) becomes the selected parent.

Single-Point Crossover: Randomly select a ‘crossover point’ which should be smaller than the total length of the chromosome. Take the two parents, and swap the gene values between them ONLY for those genes which appear AFTER the crossover point.

Mutation: You should use the following parameterised mutation operator for this, which I will call M*k*, where *k* is a number. The M*k* operator simply repeats the following *k* times: choose a gene at random, and change it to a random new value. Hence M1 changes just one gene, while M3 might change 3 genes (maybe fewer than 3, because there is a chance that the same gene might be chosen more than once).

Weakest Replacement: If the new solution is fitter than the worst in the population, then overwrite the worst (breaking ties randomly) with the new solution.

WHAT YOU HAVE TO DO

Implement the described EA in such a way that you can address the BPP, and then run experiments which address the following questions. But first note that, in all of the below, a single trial means that you run the algorithm once and stop it when 10,000 fitness evaluations have been reached. Different trials of the same algorithm should be seeded with different random number seeds.

Run five trials of the EA with crossover & operator M1 and population size 10.
Run five trials of the EA with crossover & operator M1 and population size 100.
Run five trials of the EA with crossover & operator M3 and population size 10.
Run five trials of the EA with crossover & operator M3 and population size 100.
Run five trials of the EA with only operator M3 and population size 10.
Run five trials of the EA with only crossover and population size 10.

Do all of the above, first on BPP1 and then on BPP2. In each trial, record the best fitness reached at the end of it.
So: which operator/popsize combination do you think is best when using the EA on BPP1? Which combination is best on BPP2? How sure are you about your conclusions? What other population sizes or operators might be better?

What to hand in

Hand in a report of a maximum of 4 pages (4 sides of A4). Tables and/or graphs of results should take up no more than 2 pages. In the remaining space, I want to see the fitness evaluation function (clearly commented) from your source code, and the answers to the following three questions.

Question 1: Which operator/popsize combination is best for BPP1?
Question 2: Which operator/popsize combination is best for BPP2?

Question 3: What happens when you remove the crossover or mutation?

In your answers, describe your observations of the results, and describe any tentative explanations or conclusions you feel like making, and describe any further experiments you felt it interesting or useful to do.

Marking Scheme

Correct and efficient implementation of fitness function	20%
Documentation of fitness function	5%
Correct results from the EA runs	20%
Quality (e.g. readability & usefulness) of tables and graphs	20%
Answers to Questions	15%
Tentative Conclusions & Further Experiments	20%