Assignment One: Bin-Packing with an Evolutionary Algorithm

Hand-out date: **13th November 2020**
Hand-in date: **11th December 2020**
Feedback by: **January 2021**
This CA is worth **30%** of the overall module mark

This is an **individual assessment** and you are reminded of the University's Regulations on Collaboration and Plagiarism.

Task

What you will do in this assignment is implement an evolutionary algorithm and apply it to a bin-packing problem. You will do a variety of experiments to help find out what parameters for the algorithm are best in this case. Implementation can be in the programming language of your choice. In the following sections, 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 handed in.

The Bin-Packing Problem

The bin-packing problem (BPP) involves *n* items, each weighing a different amount, that 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. E.g,. if there are 6 items with weights as follows: 1=17kg, 2=12kg, 3=19kg, 4=6kg, 5=4kg and 6=28kg and they are placed into 3 bins as follows:

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

then bin1 has a weight of 36kg (17+19), bin2 has weight of 6kg, and bin3 has weight of 44kg (12+4+28). The difference between these bins is large so it is a poor solution. The solution quality can be measured by taking the difference *d* between the heaviest and lightest bins – in this case *d* = 38. The task for the evolutionary algorithm is to minimise this difference. A far better solution in this case is:

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

where *d* = 1.

There are a number of representations that could be used for this problem, but for this assignment you should use the following k-ary representation. 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 (where item 1 is in bin 1, item 2 is in bin 3, item 3 is in bin 1 etc.) and 1 1 2 2 2 3. In the general case, if the *i*th number in the chromosome is *b*, then item number *i* is in bin *b*.

The fitness of a solution is worked out by calculating the difference *d* between the heaviest and lightest bins. There are two specific bin-packing problems you should 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 ***2i***. In problem BPP2, *b* = 100 and the weight of item *i* will be ***$2i^2$***.

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

The evolutionary algorithm should be implemented as follows:

1. Generate an initial population of *p* randomly generated solutions, and evaluate the fitness of every individual in the population.
2. Use 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, which is 10,000 (see implementation and experimentation section below).

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 to create two new children.

Multi-Gene Mutation: You should use the following parameterised mutation operator for this, M*k*, where *k* is an integer. The M*k* operator simply repeats the following *k* times: choose a gene (locus) at random, and change it to a random new value within the permissible range. Hence M1 changes just one gene, while M15 might change 15 genes (maybe fewer than 15, 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.

Implementation and Experimentation

Implement the described EA in such a way that you can address the BPP problems, and then run the following experiments and answer the subsequent questions. 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**.

Experiment 1: Run five trials of the EA with crossover & operator M1 and population size 10.

Experiment 2: Run five trials of the EA with crossover & operator M1 and population size 100.

Experiment 3: Run five trials of the EA with crossover & operator M5 and population size 10.

Experiment 4: Run five trials of the EA with crossover & operator M5 and population size 100.

Experiment 5: Run five trials of the EA with only operator M5 and population size 10.

Experiment 6: 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.

Analysis

Record the best fitness reached at the end of each trial and any other variables during the run that you think will be important in answering the following questions.

Hint: *You should think carefully about how best to present your results to show the behaviour of the algorithm during your trials*

Question 1: Which combination of parameters produces the best results for BPP1 and BPP2?
Question 2: Why do you think this is the case?
Question 3: What was the effect when you removed mutation? What about crossover?
Question 4: Which other nature-inspired algorithms might be effective in optimising the BPP problems? Explain your choice(s).

In your answers, describe your observations of the results any explanations or conclusions you can draw from your results. In addition, you should describe any further experiments you conducted to better answer questions 1-3 above and to demonstrate your understanding of the parameters used in the EA.

Submission

Report: Submit your report using EBART by 12noon on the hand-in date. The report should have a maximum of 4 pages (4 sides of A4) which should include a description of your experiments where tables and/or graphs of results should take up no more than 2-3 pages, and your answers to the questions/descriptions in the remaining space.

Code: Submit your **clearly commented code** as a zip file using the CEMPS electronic 'CA Submit' system (http://empslocal.ex.ac.uk/submit/) by 12noon on the hand-in date. If you are unsure how to use this system, please ask me.

Marking Scheme

Correct and efficient implementation of the algorithm	15%
Documentation of code	5%
Correct results from the EA runs	20%
Quality (e.g. readability & usefulness) of tables and graphs	20%
Answers to Questions	20%
Conclusions & Further Experiments	20%
Overlength submissions (per page)	-10%