ECM3412 Nature-Inspired Computation

Assignment One: Bin-Packing with Ant Colony Optimisation

Hand-out date:	**10th November 2021**
Hand-in date:	**10th December 2021**
Feedback:	**Jan 2022**

This CA is worth **40%** of the overall module mark

This is an **individual assessment** and you are reminded of the University's Regulations on Collaboration and Plagiarism, details of which are available on the College web page https://www.exeter.ac.uk/students/administration/rulesandregulations/ug-pgt/academicmisconduct/

Task

What you will do in this assignment is to implement ant colony optimisation (ACO) to the problem of bin packing. You will need to carry out a variety of experiments to help find out what parameters for the algorithm are best for this problem. The 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 description of the experiments you should carry out. The final section indicates what should 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. 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:

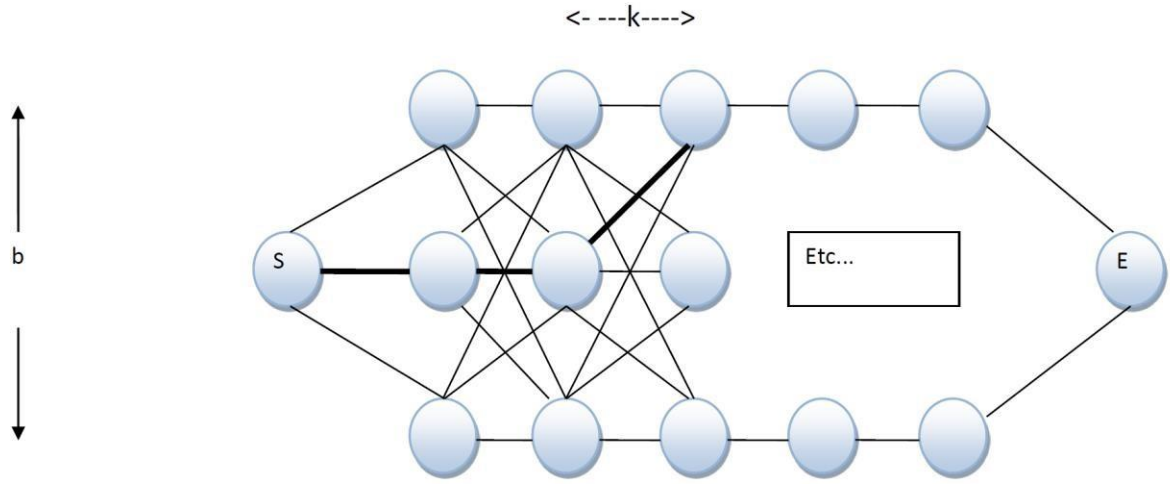
(bin1: 1, 3) (bin2: 4) (bin3: 2, 5, 6)

then bin1 has weight 36kg (17+19), bin2 has weight 6kg, and bin3 has weight 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. 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)

where *d* = 1.

You will need to create a construction graph to represent the problem. This should be a structure which contains the pheromone values for every possible decision made by the ants. For instance, a problem with *k* items and *b* bins will look like this:



Therefore, an ant travelling from the start (S) to the end (E) will have to place each of the k items into one of the b bins. The route taken by an ant marked in bold shows that item 1 will be placed in bin 2, item 2 will also be placed in bin 2, and item 3 will be placed in bin 1.

The fitness of a route is worked out by calculating the difference *d* between the heaviest and lightest bins, where the goal is to minimize this difference. There are two specific bin-packing problems you should address. In each, there are 500 items to be packed into a number of (*b*) bins (either 10 or 50). In problem BPP1, *b* = 10 and the weight of item *i* (where *i* starts at 1 and finishes at 500) will be ***i***. In problem BPP2, b = 50 and the weight of item *i* will be $\boldsymbol{i^2}$.

The Ant Colony Optimisation Algorithm

Implement the ACO like this:

1. Randomly distribute small amounts of pheromone (between 0 and 1) on the construction graph.

2. Generate a set of *p* ant paths from S to E (where *p* is a variable and specified below).

3. Update the pheromone in your pheromone table for each ant's path according to its fitness.

4. Evaporate the pheromone for all links in the graph.

5. If a termination criterion has been reached, then stop. Otherwise return to step 2.

Termination Criterion: When the algorithm has reached a maximum number of fitness evaluations (generated paths). The result is then the fitness of the best ant in the population at the end.

Generating Ant Paths: An ant will traverse your construction graph by making a decision at each new item it comes to (i.e. an ant at S can choose to go to bin 1, 2 or 3 in the illustration above). This selection is made at random, but biased by the amount of pheromone on the choices ahead (e.g. if an ant is placed at position S and bin 1 has a pheromone value of 0.5, bin 2 has a pheromone value of 0.8 and bin 3 has a pheromone value of 0.1, the ant should have a 5/14 chance of selecting bin 1, an 8/14 chance of selecting bin 2, and a 1/14 chance of selecting bin 3). This should be repeated for all *k* variables and *b* bins. There is no local heuristic for this implementation.

Pheromone Update: Once the fitness has been computed, the pheromone must be updated accordingly. With the bin packing problem, we want to reward paths that lead to bin packs with smaller differences. Therefore the pheromone update for the path will be *100/fitness*.

Pheromone Evaporation: Finally, the pheromone on all paths must be evaporated. This is achieved simply by multiplying all paths within the construction graph by the evaporation rate *e* (specified below).

Implementation and Experimentation

Implement the described ACO 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 ACO with p = 100 and e = 0.90
Experiment 2: Run five trials of the ACO with p = 100 and e = 0.50
Experiment 3: Run five trials of the ACO with p = 10, and e = 0.90
Experiment 4: Run five trials of the ACO with p = 10, and e = 0.50

Do all of the above, first on BPP1 and then on BPP2 (as defined in the Bin Packing Problem section).

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*

Question1: Which combination of parameters produces the best results?

Question 2: What do you think is the reason for your findings in Question 1?

Question 3: How do each of the parameter settings influence the performance of the algorithm?

Question 4: Do you think that one of the algorithms in your literature review might have provided better results? Explain your answer.

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 ACO.

Submission

Submit **both your report and code** using E-BART (https://bart.exeter.ac.uk/) by 12noon on the hand-in date shown above. The report should have a **maximum of 4 pages (4 sides of A4, references do not count towards the limit)** 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. **Your code should be clearly commented and submitted as a zip file**.

Marking Scheme

Correct and efficient implementation of the algorithm	15%
Documentation of code	5%
Correct results from the ACO 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%