ECMM409 Nature-Inspired Computation

Assignment One: Problem Solving with Ant Colony Optimisation

Hand-out date:	**14th October 2019**
Hand-in date:	**13th November 2019**
Feedback:	**5th December 2019**

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, 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 write a 'research paper' on the application of ant colony optimisation (ACO) to the problem described below. You will need to research the various nature-inspired algorithms that have been applied to this problem, implement the ACO approach and then carry out a variety of experiments to help find out what parameters for the algorithm are best for this problem. The implementation of the algorithm can be in the programming language of your choice. In the remainder of this document, the following is provided: details of the problem, the basic details of the algorithm, the requirements for the literature review and a description of the experiments you should carry out. The final section indicates what should be in your paper to be handed in.

The Problem

Working for a bank, you have been asked to develop an ant colony optimization system which will find the largest amount of money that can be packed into a security van. The money is separated into 100 bags of different denominations and the weight and value of the money of each bag is shown on the outside of the bag.

e.g.

Bag 1 Value = £94, Weight = 5.7Kg
Bag 2 Value = £74, Weight = 9.4Kg
.
.
.
Bag N Value = £x, Weight = iKg

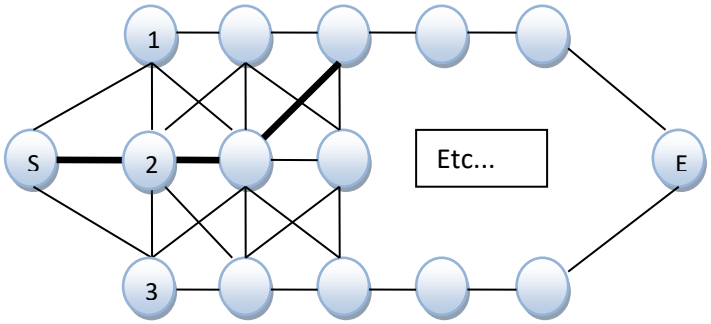
The security van has a weight limit to what it can carry, so your system must try and decide which bags to put on the van, and which ones to leave behind. The best solution will be the one which packs the most money (in terms of value) into the van without overloading it.

- Your system will have to read in the 100 bag values from a file which is posted on the ELE system named BankProblem.txt.

- The file contains the weight limit for the security van on the first line and subsequent lines contain the bag number, the values and weights for each bag of money in turn. Weights are all in kilos and the values are all in pounds sterling.

- You must decide how to represent this problem to the ant colony optimisation algorithm, you must also decide what the fitness function should be.

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. An example construction graph is shown below and has been discussed in lectures. You should choose the most appropriate construction graph and fitness function for your problem.



Once computed, your fitness function values will need to be used to update the amount of pheromone left by each ant on your graph.

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 population of ants and compute a path from the start to the end of the graph for each ant.
3. Update the pheromone in your pheromone table for each ant's path according to its fitness.
4. Evaporate the pheromone on all edges 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 10,000 fitness evaluations. The result is then the fitness of the best ant in the population.

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 has three possible path choices ahead in the example above). This selection is made at random, but biased by the amount of pheromone on the choices ahead (e.g. if the ant is at position S in the above diagram and the edge to choice 1 has a pheromone value of 0.5, the edge to choice 2 has a pheromone value of 0.8 and the edge to choice 3 has a pheromone value of 0.1, the ant should have a 5/14 chance of selecting choice 1, an 8/14 chance of selecting choice 2, and a 1/14 chance of selecting choice 3). This should be repeated for all variables for the problem represented in your construction graph.

Pheromone Update: Once the fitness has been computed, the pheromone must be updated accordingly. HINT: You will need to think carefully about how your fitness is converted into pheromone to ensure it optimizes correctly.

Pheromone Evaporation: Finally, the pheromone on all paths must be evaporated. This is achieved simply by multiplying all edges within the construction graph by an evaporation rate, usually between 0.5 and 0.95.

Literature Review

In this section, you should review a minimum of 3-4 other nature-inspired approaches that could also have been used to solve this problem. This can include variants of the ant-colony approach described here, but you should take care to properly consider the range of potential approaches and briefly provide a critical analysis of these.

Implementation and Experimentation

Implement the described ACO in such a way that you can address the above problem and then run the following experiments and answer the subsequent questions. Note that, in the experiments below it is recommended that you run the algorithm for at least 10,000 fitness evaluations. Different trials of the same algorithm should be seeded with different random number seeds.

You should devise your own set of experiments to determine the effect (if any) that the following parameters have on the performance of the optimisation:

1. Population size *(p)*
2. The amount of pheromone that is deposited according to fitness *(m)*
3. The evaporation rate *(e)*

Your experiments should assess the performance of the algorithm over a number of trials for each setting of *p, m* and *e,* to provide scientifically robust results.

'Conference Paper' Report

Your submission should be a PDF in the style of a paper using one of the style files (Latex style file or Word document template) posted on ELE. **Please do not include your name in the 'author' section – simply replace the name with your candidate number.**

The paper should have a **maximum of 5 pages** (not including references), which should include a short literature review on nature-inspired approaches and a description of your experiments where tables and/or graphs of results should take up no more than 2 pages. You should also include a description of your chosen representation in the 'Method' section, but there is no need to explain the basic algorithm (i.e. as described above). In the remaining space, write a Discussion and Further Work section including your answers to the following questions.

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, and describe any tentative explanations or conclusions you feel like making, and describe any further experiments you felt it interesting or useful to do.

Your paper should therefore minimally include the following sections: 1) Literature Review, 2) Method (i.e. your algorithm design), 3) Description of Results, 4) Discussion and Further Work, 5) References.

Submission

Paper: Submit your report (i) on paper to the Student Services Office and (ii) electronically uploading the PDF using the Turnitin link on the ELE by 12noon on the deadline shown.

Code: Submit your **clearly commented code** as a zip file using the EMPS coursework submit system (http://empslocal.ex.ac.uk/submit) to ECMM409 – "CA1 – Programming Task and Report".

Marking Scheme

Quality of Literature Review	15%
Correct and efficient implementation of the algorithm	20%
Quality of code documentation	5%
Correct results from the ACO runs	15%
Quality (e.g. readability & usefulness) of tables and graphs	15%
Answers to Questions 1-4	15%
Tentative Conclusions & Further Experiments	15%