SE36010 Assignment 2
Development of a Practical Application
(Spring 2013)

Thomas Jansen and Jun He

1 Introduction

This is the second assignment for SE36010 "Engineering Knowledge Based Systems" and comprises 50% of the total marks for the module. It gives information about how and when to hand in, specifies the tasks you have to perform and how you will format your assignment, and highlights the criteria from which the assignment will be marked. Please follow these instructions carefully.

2 Hand-In

For this assignment you have to implement an ant colony optimiser and write a report about it. Please hand-in both parts, the code and the report, electronically through Blackboard. The deadline is **on the 3rd of May, 2013**.

By submitting your work to Blackboard you are acknowledging that it is your own work and that you are aware of both the University's and the Department's views on plagiarism.

3 Task

Agents that collaborate to solve a common task can use different means of communication. Stigmergy is an indirect communication mechanism that works by leaving traces in the environment. One example for stigmergy is the communication of ants that leave pheromone trails. It is known that this enables ants to find shortest paths to food sources [1]. This observation served as inspiration to implement systems of simulated ants to solve complex optimisation problems, so called ant colony optimisation [5]. They have been applied successfully to many different difficult complex combinatorial optimisation problems, among them the computation of degree-restricted minimum spanning trees. While computing minimum spanning tree is not a hard problem [3] this changes when a degree restriction is introduced [6]. Finding efficient heuristics (among them ant colony optimisation) is an active research topic [8].

In this assignment you will implement a simple ant colony optimiser (ACO) for the minimum spanning tree problem (without any degree restriction). You can use any programming language you like provided that the code can be compiled (if compilation is required) and executed on the publicly available machines in the Department. The ACO will be similar to the 1-Ant system with the Kruskal-based construction procedure due to Neumann and Witt [7]. It will, however, incorporate a slightly more flexible pheromone update inspired by Doerr, Neumann, Sudholt, and Witt [4]. You will use this system to experiment with different parameters. You have flexibility with respect to additional features you wish to implement. In order to familiarise yourself with the task you should read the paper by Neumann and Witt [7] carefully, concentrating on the algorithms.

(Feel free to skip all proofs.) All other references provided may be interesting but are not strictly necessary for this assignment.

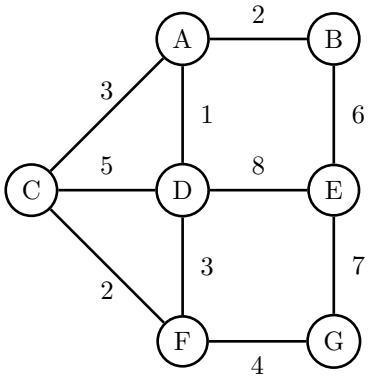
The input for your system is an undirected graph and the parameters of your ACO. You are free to chose the input format and the form of input. You can decide to read the input from a file, enter it using the keyboard or by means of a graphical user interface (or a combination of this). From this input your system computes the construction graph as outlined in Section 4, page 2411 [7].

Your ACO implements the 1-Ant as defined in Algorithm 1 on page 2408 [7] with the construction as defined in Algorithm 3 on page 2411 [7]. The pheromone update differs from the one described in [7] and is closer (but not identical) to the one described in [4]. Consider an input graph with m edges and n nodes, and an edge with current pheromone value v. If the edge is part of the solution that triggered the update the new value is $\min\{(1-\rho)\cdot v+\rho, 1\}$. Otherwise the new value is $\max\{(1-\rho)\cdot v, 1/((m-n+1)\log_2(n))\}$.

The parameters of your ACO are at least α and β (as needed in the construction; Algorithm 3 on page 2411 [7]), and ρ (as needed for the pheromone update). We have $\alpha, \beta \in \mathbb{R}_0^+$ and $\rho \in [0,1]$. You can decide to accept additional parameters. Examples are a target value for the solution that is sought or the number of runs that should be executed automatically with the same set of parameters.

If you decide to have a target value for the solution as additional input parameter you can use this as stopping criterion for the ACO. Otherwise your implementation should compute the value of a minimal spanning tree for the input and use this value as stopping criterion for the ACO.

Test your implementation using the graph below and different values for α, β, ρ, among them at least the settings $(\alpha=1, \beta=0, \rho=1)$, $(\alpha=1, \beta=0, \rho=0.1)$, $(\alpha=0, \beta=150, \rho=1)$, and $(\alpha=0, \beta=150, \rho=0.1)$.



4 Requirements

The assignment asks you to produce two things, a piece of code and a report.

Your code must run on one of the publicly available machines in the Department. You need to include full instructions for compiling and running your code. If your instructions are unclear, or if your code does not run on a machine in the labs you will receive zero marks for this part. Please take this warning seriously.

Your code must be able to accept undirected weighted graphs as input (without the need to recompile the code), compute the corresponding construction graph, and run the ACO on it. It must accept at least α, β, and ρ as input parameters. It should output at least the best solution found together with the number of ants needed.

The report must include sections "Introductions and Background", "Overview of Features", "Discussion of Implementation", "Experimental Results", "Conclusion", and "References". There are no strict lower or upper bounds on the word count. Give a detailed and accessible description and discussion of all aspects without being unnecessarily lengthy.

5 Assignment Marks

This assignment will be assessed under departmental assessment criteria in appendix AA (Development) of the Student Handbook (`http://www.aber.ac.uk/~dcswww/Dept/Teaching/Handbook/AppendixAA.pdf`). The marks breakdown for this component is as follows:

1. Marks for the software project:
    - 10 marks for a working ACO implementation. If you use code from the internet to help with this you must reference it fully, and make it clear which parts of the code are your own.
    - 10 marks for a system which allows experimentation with parameters in a convenient way. Remember that your ACO is a randomised algorithms, results may vary from run to run even if neither the input nor the parameter changed, and that one is most interested in its average performance.
    - 10 marks for code quality (comments, structure, installation instructions and so on).
    - 10 marks for a convenient user interface (possibly but not necessarily including a graphical user interface).
    - 10 marks for an accessible output format (possibly not but necessarily including a visualisation of the spanning tree found).
2. Marks for the report:
    - 10 marks for introduction and background: What decisions have you made with respect to input and output format, additional parameters.
    - 10 marks for an overview of the features of your implementation: What is your system capable of, how can it be used, which part of the implementation are particularly clever or interesting.
    - 10 marks for a discussion of the implementation including the challenges you have faced.
    - 10 marks for the presentation and discussion of your experiments. What influence do the different parameters have? How stable is the behaviour of the ACO?
    - 5 marks for conclusion including a discussion of any shortcomings and things you would now do differently.
    - 5 marks for references.

The indicative mark breakdown for both sections combined sums to 100. This coursework is worth 50% of the overall mark for SE36010, and so you can, if you prefer, think of each part of this assessment as counting for a quarter of the module.

6 Plagiarism

Please follow the guidelines from the Student Handbook (`http://www.aber.ac.uk/~dcswww/Dept/Teaching/Handbook/handbook.pdf`) to help you avoid straying from legitimate and desirable cooperation into the area of plagiarism.

- Append a bibliography to your work listing all the sources you have used, including electronic ones.
- Surround all direct quotations with inverted commas, and cite the precise source (including page numbers, or the URL and the date you accessed it if the source is on the Web) either in a footnote or in parentheses directly after the quotation.
- Use direct quotations sparingly and make sure that the bulk of the work is in your own words.
- Even if you don't use direct quotations, important ideas still need to be credited.
- Remember that it is your own input that gives a piece of work merit. Whatever sources you have used, the structure and presentation of the argument should be your own. If you are using electronic sources, dont cut and paste sections into your work. If you are using books or papers, put them aside when you actually sit down to write. In this way you won't be tempted to copy in material that you don't understand, or be at risk of unintentionally copying in more material than a brief quotation, or of accidentally leaving quotations unmarked. Including someone else's work in your own is readily detectable because the style will be different.

References

[1] R. Beckers, J. L. Deneubourg, S. Goss, and J. M. Pasteels (1990): Collective decision making through food recruitment. *Insectes Sociaux* 37(3):258–267. `http://dx.doi.org/10.1007/BF02224053`

[2] T. N. Bui and C. M. Zrncic (2006): An ant-based algorithm for finding degree-constrained minimum spanning tree. In M. Cattolico (Ed.): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)*, ACM Press, pages 11–18. `http://dx.doi.org/10.1145/1143997.1144000`

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein (2001): *Introduction to Algorithms*. 2nd edition. MIT Press.

[4] B. Doerr, F. Neumann, D. Sudholt, and C. Witt (2011): Runtime analysis of the 1-ANT ant colony optimizer. *Theoretical Computer Science* 412(17):1629–1644. `http://dx.doi.org/10.1016/j.tcs.2010.12.030`

[5] M. Dorigo and T. Stützle (2004): *Ant Colony Optimization*. MIT Press.

[6] M. R. Garey and D. S. Johnson (1979): *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann.

[7] F. Neumann and C. Witt (2010): Ant colony optimization and the minimum spanning tree problem. *Theoretical Computer Science* 411:2406–2413. `http://dx.doi.org/10.1016/j.tcs.2010.02.012`

[8] S. Sundar, A. Singh, and A. Rossi (2012): New heuristics for two bounded-degree spanning tree problems. *Information Sciences* 195:226–240. `http://dx.doi.org/10.1016/j.ins.2012.01.037`