Implementing a simple ACO solver for the TSP

MSc Interactive Entertainment Technology
cs7032: AI & Agents
S. Luz (luzs@cs.tcd.ie)

October 28, 2013

1 Goals

In this practical you will implement an Ant Colony Optmisation (ACO) algorithm [DD99] for the Travelling Salesman Problem, and visualise its performance through Repast.

2 Some code

In the course web site you will find some Java code for this practical[1]. This code[2] should provide you with the basics for implementing an ACO simulation using Repast, namely: the basic model for the simulation (`TSAModel`[3]), a map (`TSAWorld`[4]) of the `TSACity`[5]s to be visited, backed by a `Object2DTorus`[6] and set by `TSAModel` to be displayed onto an `Object2DDisplay`[7]. The display includes methods for drawing the nodes (as green squares) the total amount of pheromone on each path/link (in white) and the best path (in red). The link drawing method is set so that the more pheromone a link contains the more clearly the link will be drawn.

You will also find a placeholder class for `TravellingSalesAnt`[8] which you will modify in order to implement the ACO meta-heuristic. The code contains some documentation explaining how the simulation is structured.

The libraries you will need in order to compile and run this simulation are distributed under the repast3 release and have been included in this lab's tar file in their .jar packaging for convenience. They include simulation-specific packages (such as `repastj.jar`), CERN's scientific computing libraries (`colt.jar`), plotting libraries (`plot.jar`) and the trove high performance collections library.

Emacs/JDEE users will find a `prj.el`[9] file which can be used for compilation and running of the simulation under JDEE with a minor modification to the setting of `jde-global-classpath`.

Bourne shell scripts and Windows "batch files" have been included for compilation (`compile.sh` and `compile.bat`) and stand-alone running of the simulation (`run.sh`, `run.bat`). They refer

[1] http://www.scss.tcd.ie/~luzs/t/cs7032/sw/tsa-0.7.tar.gz
[2] The package is named "sim.tsa", short for "Travelling Sales Ant", after a project by Armin Buch.
[3] sim.tsa.TSAModel
[4] sim.tsa.TSAWorld
[5] sim.tsa.TSACity
[6] uchicago.src.sim.space.Object2DTorus
[7] uchicago.src.sim.gui.Object2DDisplay
[8] sim.tsa.TravellingSalesAnt
[9] sim/tsa/prj.el

only to libraries included in the tar file and should be easily translated into equivalent .bat files (i.e. by replacing colons by semi-colons and forward slashes by backslashes etc).

Although the above described code template is provided, you are free to implement your own ACO algorithm for TSP from scratch. You may even use a different simulation platform, such as MASON[10], which offers similar functionality to Repast.

3 Exercises

From the list of exercises below, attempt to do exercise 1, plus **one** of the remaining exercises of your choice.

1. Implement the behaviour of a an active ant (see `newActiveAnt()` in the lecture notes) by modifying (implementing, in fact) methods step() and step2() in `TravellingSalesAnt.java`.

2. Modify the model so that the program allows you to save and load city maps (i.e. specific instances of TSP) so that you can compare the performance of the algorithm across several runs under different parameter settings. A database of symmetrical TSP (and related problems) and their best known tours can be found at the Univeity of Heidelberg's TSPLIB[11].

3. Experiment with different parameters settings (e.g. number of ants, distance weight, pheromone weight, amount of pheromone) and report briefly on how the algorithm performs.

4. `TSAModel` differs from the algorithm describe in class in that `TSAModel` currently updates pheromone only after all ants have finished their tour. Try modifying the code so that each ant leaves its pheromone trail immediately after it completes its tour. What effect does that have on the system?

5. Can you devise a heuristic for identifying unpromising solutions and dynamically tuning the ACO parameters in order to enable the algorithm to explore a wider solution space?

3.1 Delivering the assignment

This assignment is due on the first Tuesday after the reading week, and should be submitted (as usual) through blackboard. Any text should be submitted either as plain text or in PDF format. No MSWord files, please.

References

[DD99] Marco Dorigo and Gianni Di Caro. The ant colony optimization meta-heuristic. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London, 1999.

[10] http://cs.gmu.edu/~eclab/projects/mason/
[11] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/