Visualising Ant Colony Optimisation

Chris Edwards

Major Project 2015

Overview

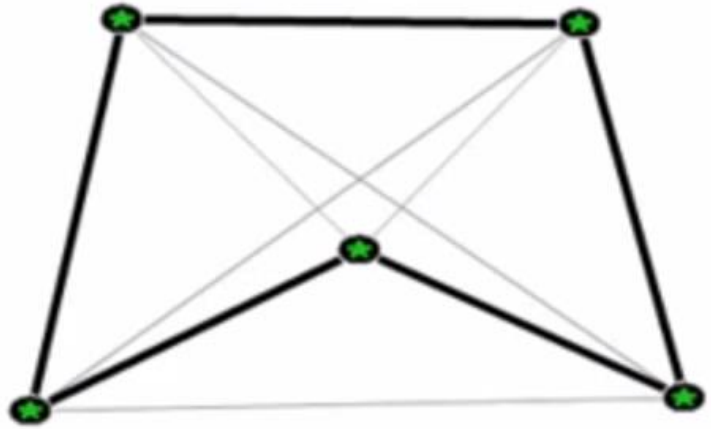
- Provide a visual representation of Ant Colony Methods
  - Nodes, Agents, Pheromone, Agent movement, Best route, …
- Enable user interaction
  - Modify parameters
  - Start/stop execution
  - Generate new problems
  - Load/save problem configurations
- Educational Objectives
  - Teaching tool
- Explore any potential extensions

Research

- ACO behaviours
  - Algorithm types
  - How do they operate?
- Existing solutions
  - What do they provide?

Example Solution

Click ‘Run’



- No intermittent steps
- Little user feedback
- Doesn’t really educate

Solution from: https://www.youtube.com/watch?v=QZnvRXRsHHg

Existing solutions

- Pros
  - Simplistic view and representation
- Cons
  - No agent visualisation
  - No indication of movement
  - ‘instant’ solving
  - No parameter modification
  - No pheromone visualisation
  - No choice of algorithm type or modifiers
- There are better/worse solution available this is just one example

Travelling Salesman Problem(TSP)

- ACO is commonly applied to the TSP
- This representation very customisable
  - User defined number of city locations
  - Modify the weighting of edges
  - Randomise city locations
  - Etc…
- However, this may be a difficult concept to grasp for new users…

Pheromone and distance data structures

- Both data structures are scalable and easily accessed
- Two-dimensional array
  - Size of each array depends on the # of cities
  - Easy to access elements
- Each element corresponds to an edge
  - E.g. Element [x][y]
    - Refers to the edge from city x to city y
    - DistanceMatrix[x][y] holds the length of this edge
    - Pheromone[x][y] holds the pheromone data for this edge

	0	1	2	3
0	0.0	15.65	22.45	16.32
1	15.65	0.0	13.98	19.21
2	22.45	13.98	0.0	12.78
3	16.32	19.21	12.78	0.0

Path 0 to 1 is now uphill, adjust the distance matrix…

	0	1	2	3
0	0.0	15.65	22.45	16.32
1	31.30	0.0	13.98	19.21
2	22.45	13.98	0.0	12.78
3	16.32	19.21	12.78	0.0

Concurrency

- Automated execution process is resource intensive
  - Caused GUI to ‘hang’ until completion
- Need to implement concurrent behaviours…
  - Java only allows UI updates on the Event Dispatch Thread
- SwingWorker
  - Correctly handles Swing UI communication

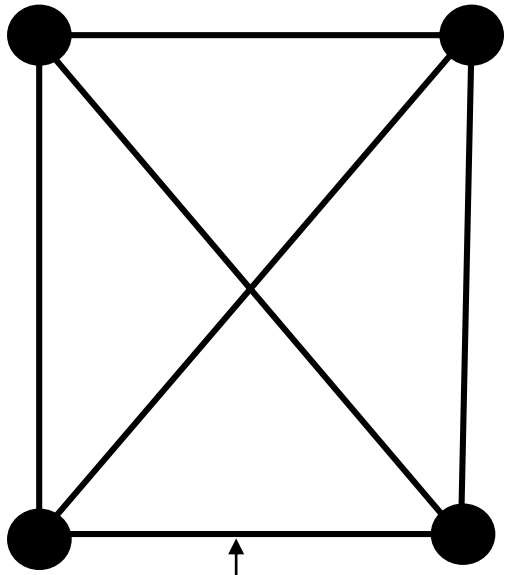
Complications - Rendering

- The rendering of the algorithms current state is far from trivial
- The application must display;
  - The city locations
  - The paths + pheromone concentrations between cities (edges)
  - Agent location
  - Agents movement
  - Current best route
- All must be done in ‘real time’
  - Every intermittent step will be shown
  - Existing solutions often just ‘skipped’ to the solution

Rendering - Paths

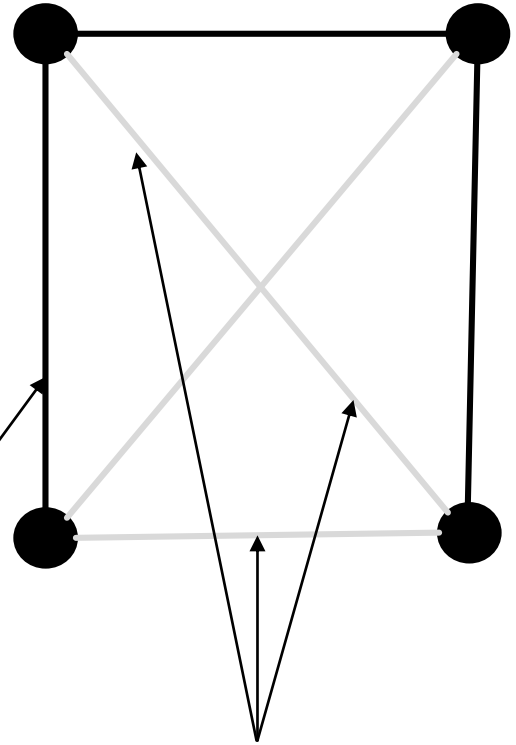
- A path represents an edge connecting two nodes
- Given the location of these nodes (cities) is known;
  - A path from each city can be drawn to every other city
- These edges must also model pheromone
  - The best way to model this is to modify line opacity
  - Pheromone levels for an edge are scaled by 2000
    - Line opacity will be: edgePhero * 2000 or 255 which ever is smaller

This is the general case, it may vary depending on parameter values



Initially all edges have the same pheromone

Eventually…

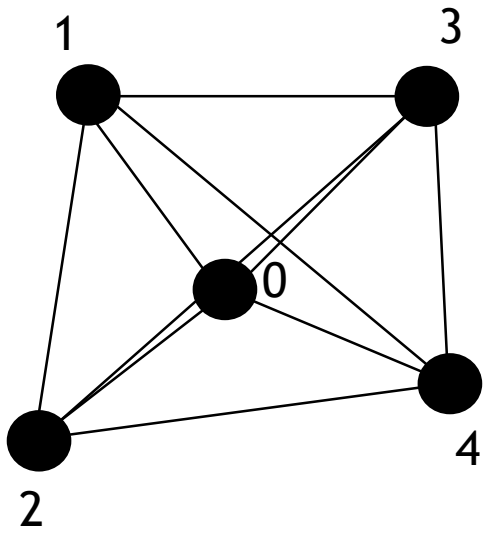


‘shortest’ paths have more pheromone: darker lines

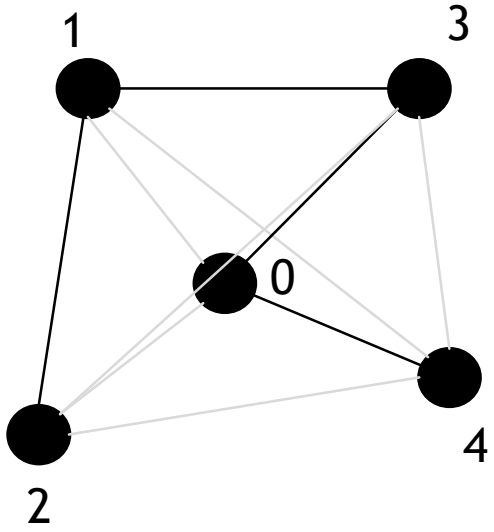
‘longest’ paths have less pheromone: lighter lines

Rendering – Best Route

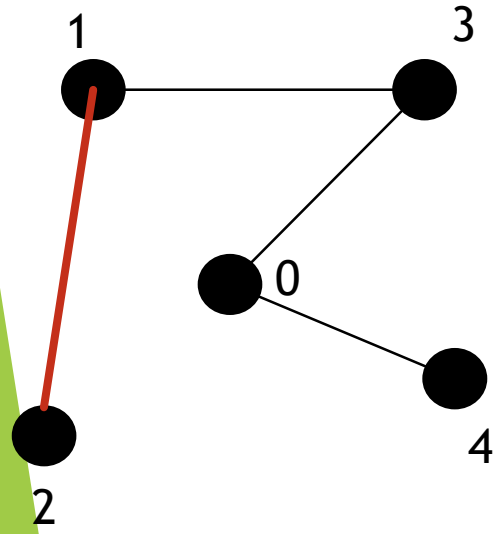
- Displaying the best route is slightly more complex than initially thought
- The best route is stored as a list of city indexes
  - E.g. [0,2,9,3,6,4,1,8,7,6]
- Must iterate through this list in pairs
- Drawn a line between each pair
  - The end of the previous line is the start of the next one
    - E.g 0 and 2, 2 and 9, 9 and 3 ….
- Each line will join to create the best path…



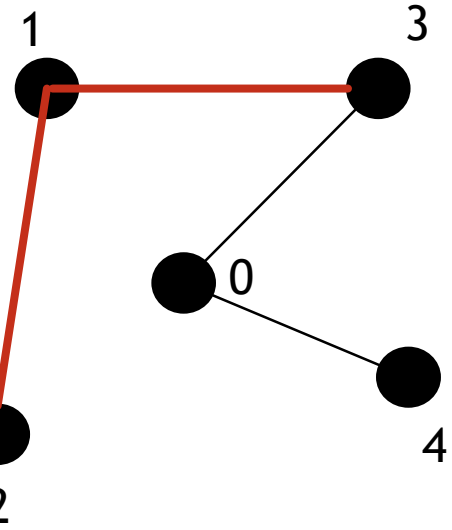
Converges to...



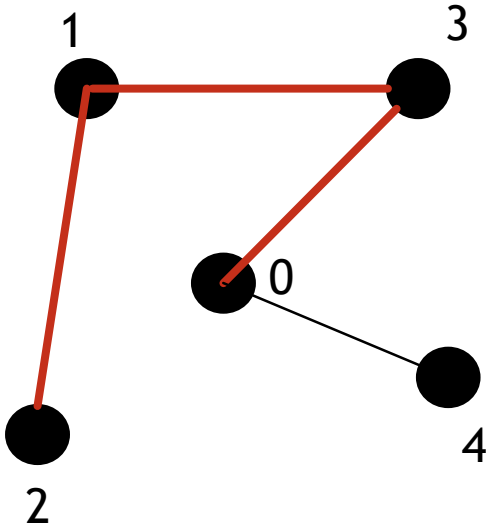
Best route:
[2,1,3,0,4]



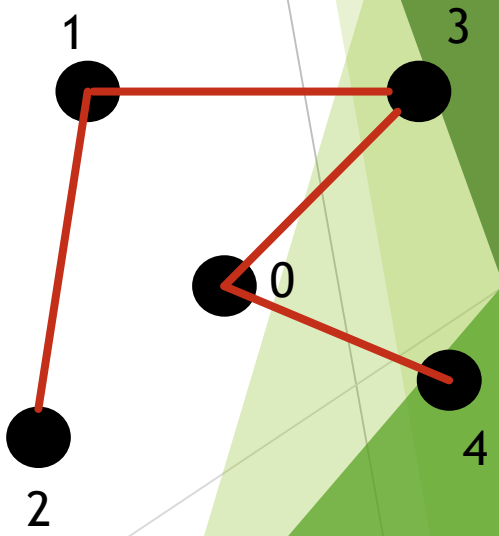
2 to 1



1 to 3



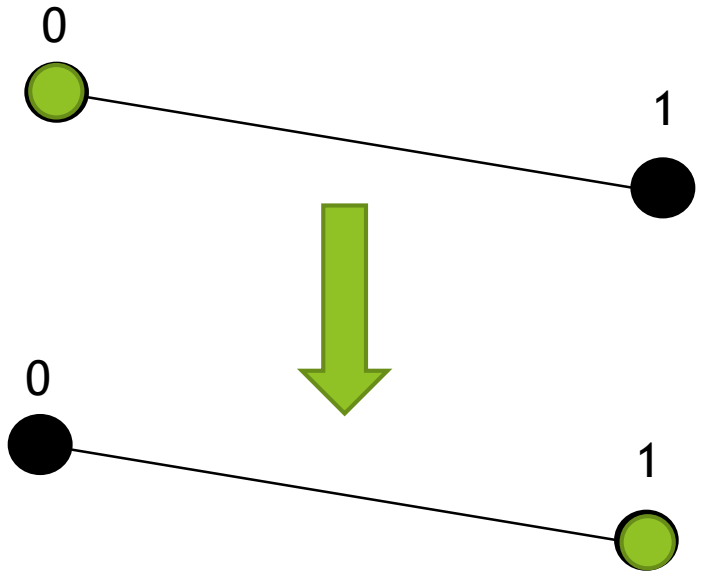
3 to 0



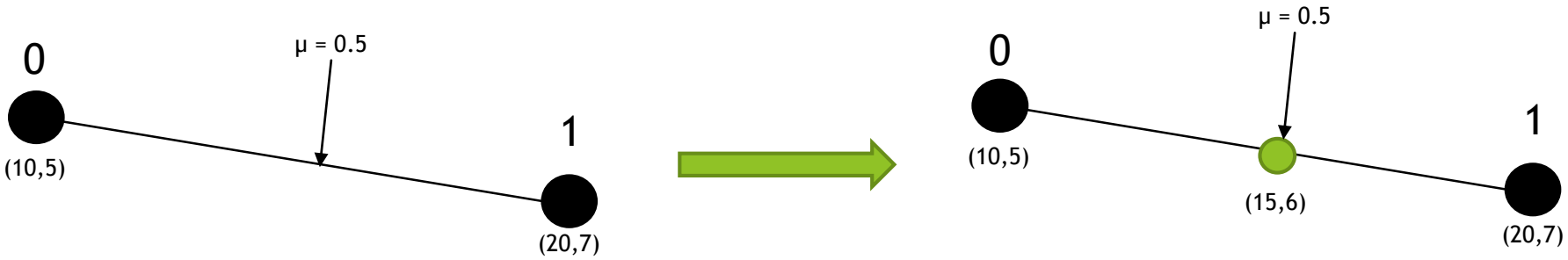
0 to 4

Rendering - Movement

- Most solutions offer no visualisation of the agents movement
- This is a difficult task
- Given the agents starting city index and its next probable location
  - The path it took can be extracted
- Once the path taken is known;
  - This can be highlighted to the user
    - Currently uses Linear Interpolation
    - Is this the best method
    - What is meant by highlighting…
      - Colours, images, …

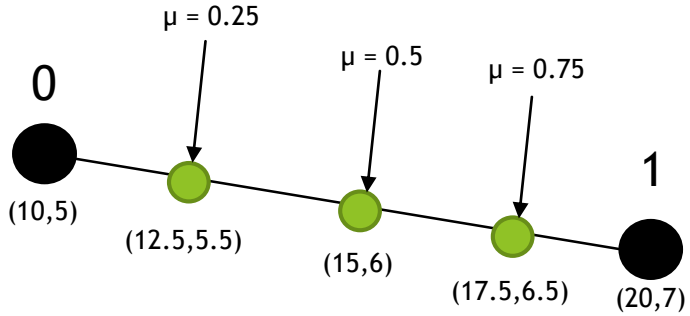


- No indication of movement
- Agent seems to ‘teleport’
- This is a trivial example, with more cities/agents its even more of a problem
- There must be a way to visualise this movement…



$Target_x = (0.x * (1 - \mu) + 1.x * \mu)$ $Target_x = (10 * (0.5) + 20 * 0.5) = 15$

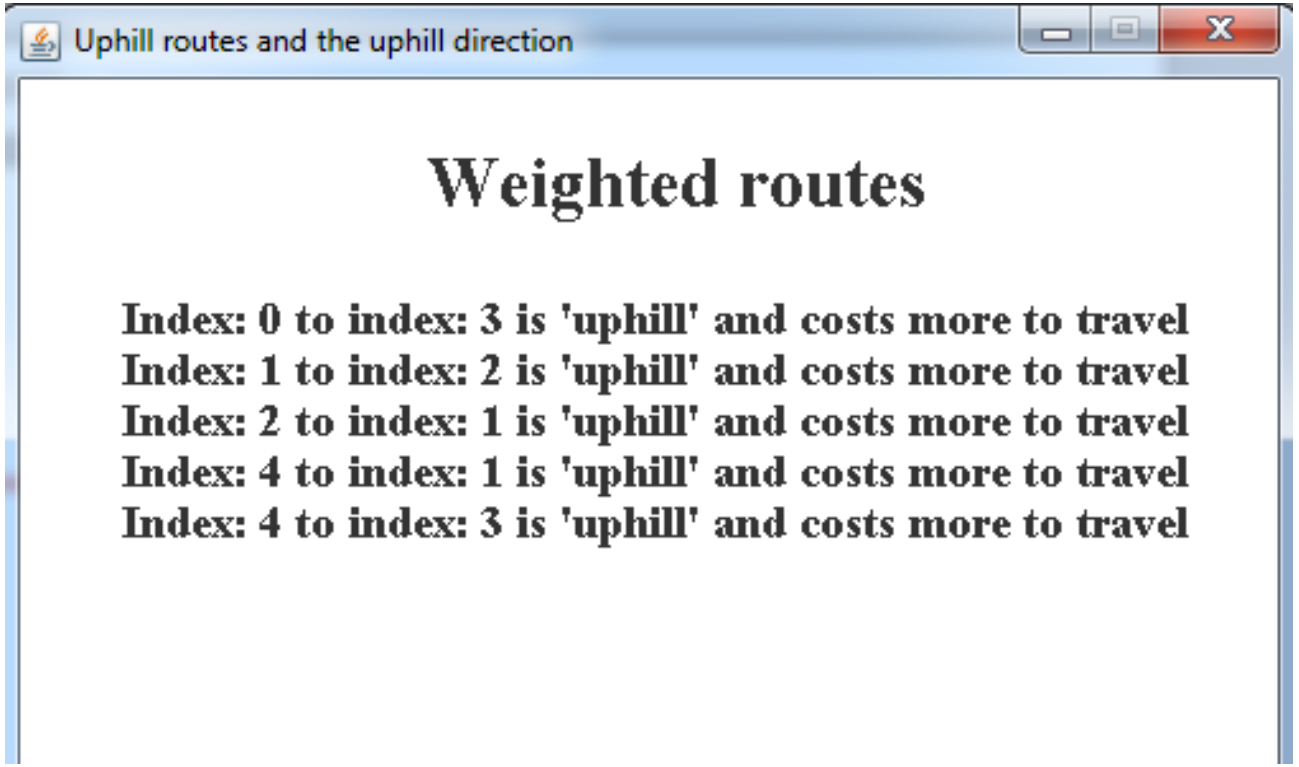
$Target_y = (0.y * (1 - \mu) + 1.y * \mu)$ $Target_y = (5 * (0.5) + 7 * 0.5) = 6$



- Shows agents movement
- Spacing between ‘ellipsis’ is relevant to path length
- Doesn’t show a direction
- Good starting point

Weighted Paths

- The user can enable or disable the generate of weighted (uphill) paths
- These will cost twice their normal cost
  - Double the corresponding distance matrix value
- Currently randomly generated
- Difficult to visualise without cluttering the view
- A basic separate view has been created…



- Very basic
- Easy to comprehend
- Doesn’t impact the main display

Step-based iteration

- Automated solving is not always what the user wants
- A way to solve the problem on a step-by-step basis was implemented
- The user can execute this at their own pace
- During step mode;
  - The user cant load or modify parameters once they have started execution
  - The algorithm will ‘step’ once and pause
    - Lets the user study what just happened
    - Predict next step

Elitist Ant System

- Added so users can compare algorithm execution
- One of the first proposed improvements to the Basic Ant System

- Retains knowledge of the current best across interactions
  - User defined number of ‘best’ routes retained
  - Portion of pheromone deposited along the retained best routes

- There needs to be a way to store this data sensibly…

Elitist Ant System - Storage

- As pheromone needs to be deposited on the best route, the best route must be stored
- The distance of each route is also stored
  - Used to compare if a new route Is better than the current x number stored
- This is the only data that is needed
  - Rather than storing the x number of Agents we can simple store only the route and distance belonging to the agent
    - Less overheads, more efficient
  - Can then iterate through the x stored best
    - Deposit pheromone
    - Update the current elite if we find better

Future Work

- Implement more algorithm variations
- Allow graph generation for incomplete graphs
- Support other algorithm types
  - Bee Colony Algorithms
  - A*, Dijkstra's algorithm, Depth-first, Breadth-first
- Improve the ant movement visualisation
  - Show direction
- Explore additional problem representations
  - Double bridge, nest/food, …

Evaluation

- The current application provides a solution to the problem
- Improves on features provided by competitors
- Simplistic to understand and use
- Would like to add additional algorithm types
- Room for improvement
- Framework can be reused to support other algorithm types