Visualising Ant Colony Optimisation

Major Project

Christopher Edwards

Aims of the Project

- Provide a visual representation of an Ant Colony Optimisation algorithm

- To be used in an educational environment
    - Teaching resource

- The algorithm’s parameters can be changed by the user
    - Number of cities
    - Number of ants
    - Alpha value
    - Beta value
    - Decay rate
    - etc

Ant Colony Optimisation

- Family of Swarm Intelligence methods
  - SI typically involves a population of simple agents
- Probabilistic path finding technique
  - Return the optimal path in a graph
- Based on real-world ant behaviors
  - Ants always find the shortest path between nest + food
  - Work together based on pheromone trails
- Currently this project focuses on the TSP
  - Travelling salesman problem

Probabilistic property

- An agents next location is calculated using the following probability:

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}$$

  - α: Meta heuristic value relating to the favoring of pheromone
  - β: Meta heuristic value relating to the favoring of pheromone
  - η: 1/distance from current to node xy

Pheromone deposit

- The pheromone trails are updated as the agents traverse the graph
- The equation to model this behavior is:

$$p_{xy}^{k} = (1 - \rho)\tau_{xy}^{k} + \Delta\tau_{xy}^{k}$$

$$\Delta\tau_{xy}^{k} = \begin{cases} Q/L_k & \text{if Agent } k \text{ uses curve } xy \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

  - This correctly factors in decay and allows for any ant moving through node xy to add to the pheromone concentration at node xy

Technical Implementation

- Implemented in Java and its Swing packages
  - Cross-platform
  - JUnit tests
- Visualising the algorithm is difficult
  - Easy to implement, difficult to visualise
    - Showing pheromone levels to the user
    - Showing the agents moving
    - Show the best route
- Do all this in a reasonable time

Technical Challenges

- Use of SwingWorker to execute the algorithm in its own thread
  - Allows updates of the view without it ‘freezing up’

- Simulate ants moving on the ‘paths’
  - Get the location the ant started and its destination
  - Use linear interpolation
    - Given the x /y co-ordinate of the start and finish
    - get the value of any x/y co-ordinate on the path between these two points
    - value1*(1-µ)+value2* µ);
      - value1/2 is the X or Y value (e.g value1 = start x, value2 = destination x)
      - µ is how far on that path you want (e.g µ 0.5 = half way between the points)

Current algorithm

- To update the view to reflect the current state
- For every ant(agent)
    - Draw the ant at their current location
    - If they aren’t finished, move them
        - During this move, get their current X and the destination
            - Linearly interpolate these two locations to show the ant moving
            - Update pheromone
        - If the ant has the current best path, set its path as best
            - Loop through the cities visited in order and draw a line between them
            - paint the updated pheromone
    - Once all ants have finished
        - Remove all pheromone trails and only show the best path
- This is the general form there are other intricacies

Demonstration

- Current features:
  - load configurations from a file
  - Save configurations to a file
  - Set the speed
  - Modify algorithm parameters
  - Error checking on all fields
  - Display the best route and distance
  - See the ants move
  - Visualise pheromone

Future work

- Display how many ants are at each city
    - Currently shows 1 ‘ant’ regardless
- Experiment with correct limits to add to text fields
- Potentially add variations of the algorithm
    - Elitist ants, Min-Max system etc.
- Test the application thoroughly
- Potentially set up an nest-food style of problem
- Potentially extend the application to visualise BCO as well