HEURISTIC IN ARTIFICIAL INTELLIGENCE

Hello!

I am Jahanzeb Naeem

1

What is a Heuristic?

Definition of a Heuristic

- Enabling a person to discover or learn something for themselves.
- Two important terminologies:
  - Heuristic Search
    - May not find the optimal solution to the problem
    - Will give a good solution in reasonable time
  - Heuristic Function
    - Rank all the possible alternatives, based on the available information
    - Helps the algorithm to select the best route

Types of Heuristic Algorithms

- A* (Done Previously)
- Hill Climb
- ACO (Ant Colony Optimization)

2

Hill Climbing

Hill Climbing

- Hill Climbing Algorithm can be categorized as an informed search.
- We can implement any node-based search or problems like the n-queens problem using it.

Features of Hill Climbing

- Variant of generate and test algorithm
  - Algorithm
    - Step 1: Generate possible solutions
    - Step 2: Test to see if this is the expected solution
    - Step 3: If the solution has been found quit else go to step 1
  - It takes the feedback from the test procedure
  - Feedback is utilized by the generator in deciding the next move
- Uses the Greedy approach

Types of Hill Climbing

- Simple Hill climbing
  - It examines the neighboring nodes one by one and selects the first neighboring node which optimizes the current cost as next node.
  - Algorithm
    - ***Step 1 :*** *Evaluate the initial state. If it is a goal state then stop and return success. Otherwise, make initial state as current state.*
    - ***Step 2 :*** *Loop until the solution state is found or there are no new operators present which can be applied to the current state.*
    - *a) Select a state that has not been yet applied to the current state and apply it to produce a new state.*
    - *b) Perform these to evaluate new state*
      - *i. If the current state is a goal state, then stop and return success.*
      - *ii. If it is better than the current state, then make it current state and proceed further.*
      - *iii. If it is not better than the current state, then continue in the loop until a solution is found.*
    - ***Step 3 :*** *Exit.*

Types of Hill Climbing (Cont.)

- Steepest-Ascent Hill climbing
  - It first examines all the neighboring nodes and then selects the node closest to the solution state as of next node.
  - Algorithm
    - ***Step 1 :*** *Evaluate the initial state. If it is goal state then exit else make the current state as initial state*
    - ***Step 2 :*** *Repeat these steps until a solution is found or current state does not change*

      i. Let 'target' be a state such that any successor of the current state will be better than it;

      ii. for each operator that applies to the current state

      a. apply the new operator and create a new state

      b. evaluate the new state

      c. if this state is goal state then quit else compare with 'target'

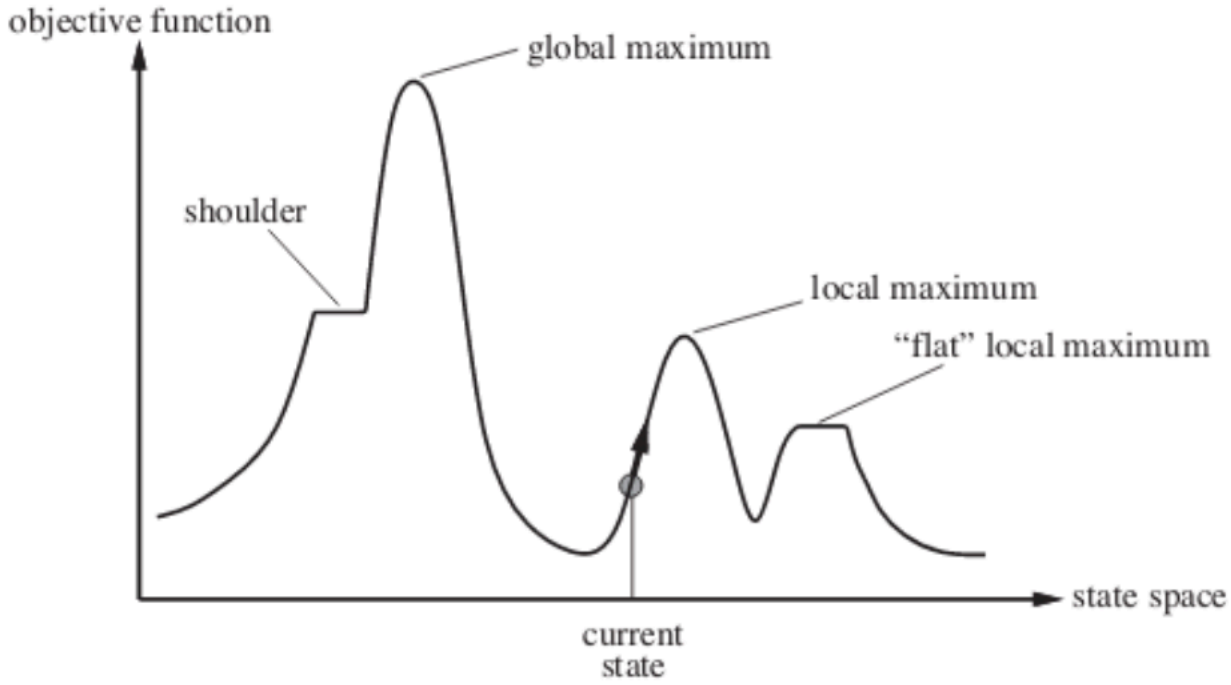
      d. if this state is better than 'target', set this state as 'target'

      e. if target is better than current state set current state to Target
    - ***Step 3 :*** *Exit.*

Types of Hill Climbing (Cont.)

- Stochastic hill climbing
  - It does not examine all the neighboring nodes before deciding which node to select.
  - It just selects a neighboring node at random and decides (based on the amount of improvement in that neighbor) whether to move to that neighbor or to examine another.

State Space diagram for Hill Climbing



State Space diagram for Hill Climbing (Cont.)

- Local maximum
  - It is a state which is better than its neighboring state however there exists a state which is better than it (global maximum).
  - This state is better because here the value of the objective function is higher than its neighbors.
- Global maximum
  - It is the best possible state in the state space diagram.
  - This because at this state, objective function has highest value.

State Space diagram for Hill Climbing (Cont.)

- Plateua/flat local maximum
  - It is a flat region of state space where neighboring states have the same value.
- Ridge
  - It is region which is higher than its neighbors but itself has a slope.
  - It is a special kind of local maximum.
- Current state
  - The region of state space diagram where we are currently present during the search.

State Space diagram for Hill Climbing (Cont.)

- Shoulder
  - It is a plateau that has an uphill edge.

Problems in different regions in Hill climbing

- Hill climbing cannot reach the optimal/best state (global maximum) if it enters any of the following regions:
  - Local maximum
    - At a local maximum all neighboring states have a values which is worse than the current state.
    - Since hill-climbing uses a greedy approach, it will not move to the worse state and terminate itself.
    - The process will end even though a better solution may exist.
    - Solution
      - Utilize backtracking technique.

Problems in different regions in Hill climbing

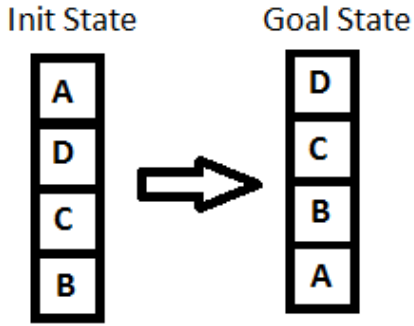
- Plateau
  - On plateau all neighbors have same value.
  - Hence, it is not possible to select the best direction.
  - Solution
    - Make a big jump.
    - Randomly select a state far away from the current state.
    - Chances are that we will land at a non-plateau region

Problems in different regions in Hill climbing

- Ridge
  - Any point on a ridge can look like peak because movement in all possible directions is downward.
  - Hence the algorithm stops when it reaches this state.
  - Solution
    - In this kind of obstacle, use two or more rules before testing.
    - It implies moving in several directions at once.

Example

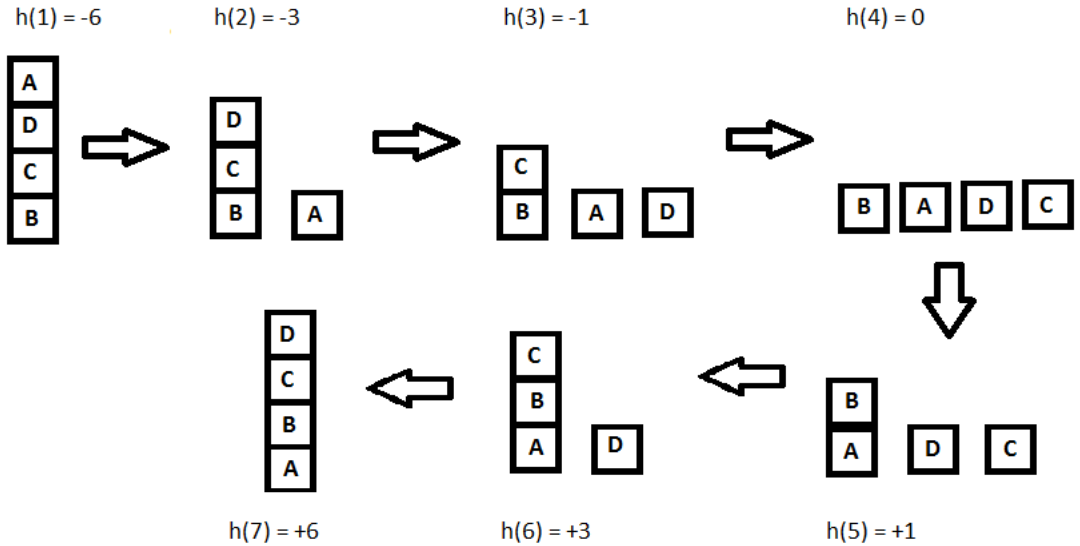
- Key Point
  - While solving any hill-climbing problem is to choose an appropriate heuristic function.
  - Function h(x)
    - +1 for all the blocks in the support structure if the block is correctly positioned
    - -1 for all the blocks in the support structure.



- We will call any block correctly positioned if it has the same support structure as the goal state.

Example (Cont.)

- Let's look at all the iterations and their heuristics to reach the target state:



3

Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO)

- In the real world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep traveling at random, but instead follow the trail laid by earlier ants, returning and reinforcing it if they eventually find food
- Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate.

Ant Colony Optimization (ACO)

- A short path, by comparison, gets marched over faster, and thus the pheromone density remains high.
- Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

Ant Colony Optimization (ACO)

- Thus, when one ant finds a good (short) path from the colony to a food source, other ants are more likely to follow that path, and such positive feedback eventually leaves all the ants following a single path.
- The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the search space representing the problem to be solved.
- Ant colony optimization algorithms have been used to produce near-optimal solutions to the traveling salesman problem.

Ant Colony Optimization (ACO)

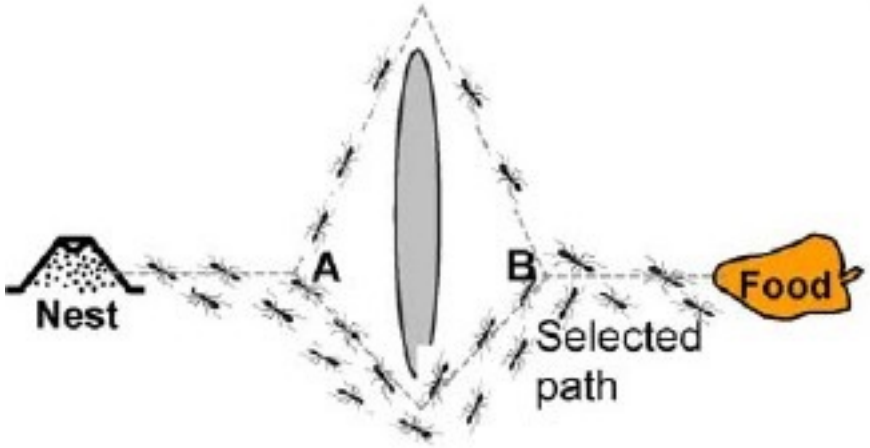
- They have an advantage over simulated annealing and genetic algorithm approaches when the graph may change dynamically. The ant colony algorithm can be run continuously and can adapt to changes in real time.
- This is of interest in network routing and urban transportation systems.

Ant Colony Optimization (ACO)

- Definition
  - A heuristic optimization method for shortest path and other optimization problems which borrows ideas from biological ants.
  - Based on the fact that ants are able to find shortest route between their nest and source of food.

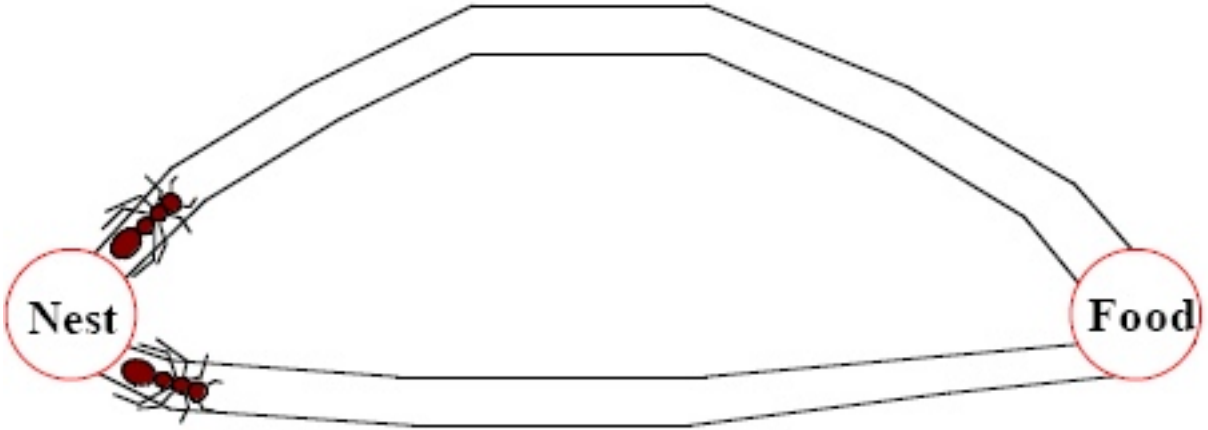
Ant Colony Optimization (ACO)

- Shortest Route
  - Shortest route is found using pheromone trails which ants deposit whenever they travel, as a form of indirect communication.



Ant Colony Optimization (ACO)

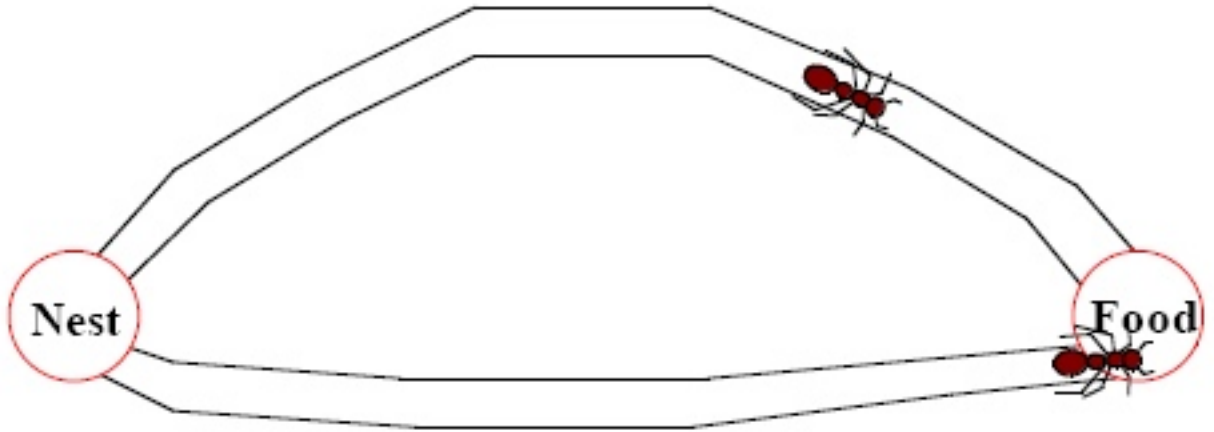
- Ant foraging – Co-operative search by pheromone trails.



When ants leave their nest to search for a food source, they randomly rotate around an obstacle.

Ant Colony Optimization (ACO)

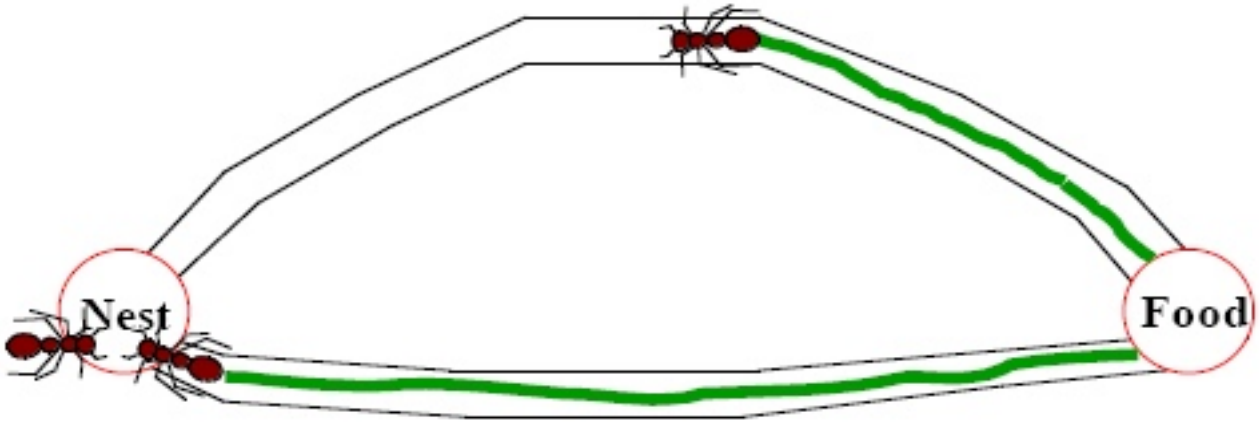
- Ant foraging – Co-operative search by pheromone trails.



Initially the pheromone deposits will be the same for the right and left directions

Ant Colony Optimization (ACO)

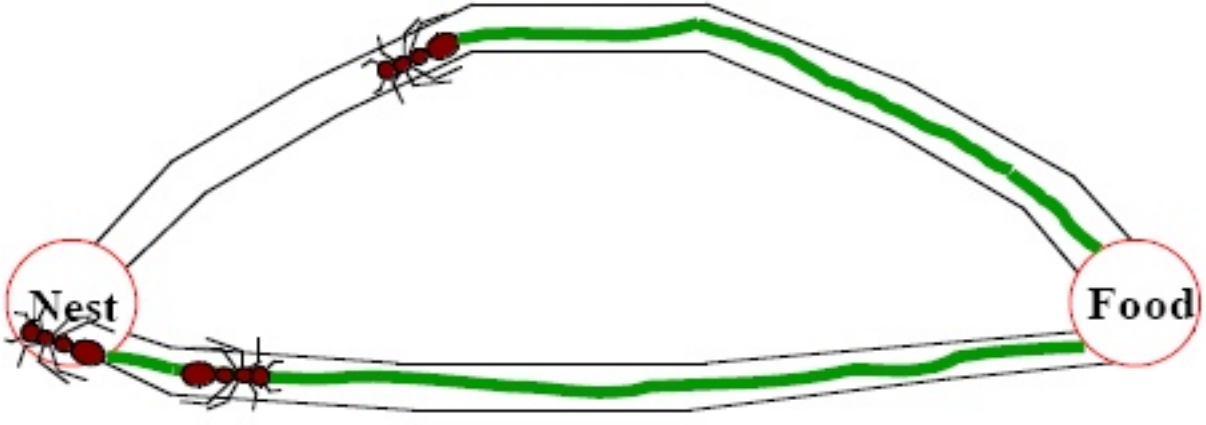
- Ant foraging – Co-operative search by pheromone trails.



When the ants in the shorter direction find a food source, they carry the food and start returning back, following their pheromone trails, and still depositing more pheromone.

Ant Colony Optimization (ACO)

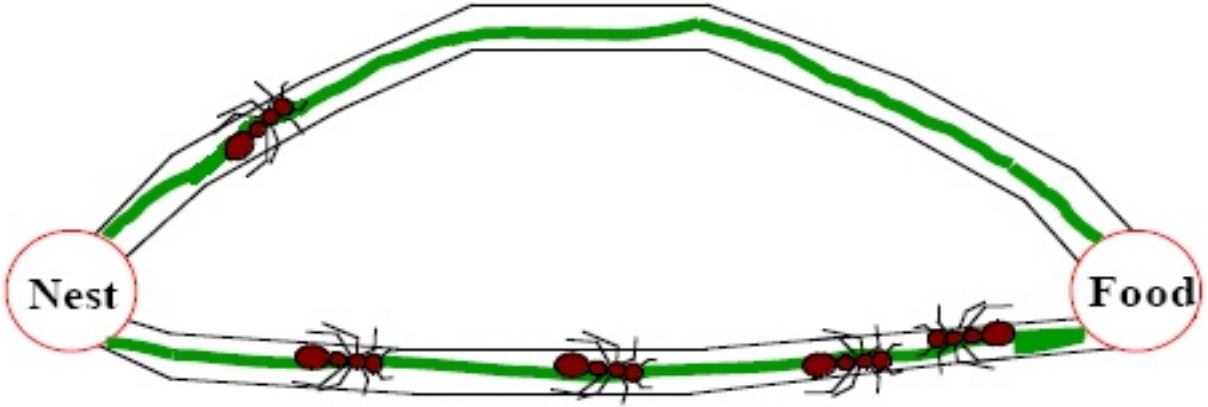
- Ant foraging – Co-operative search by pheromone trails.



An ant will most likely choose the shortest path when returning back to the nest with food as this path will have the most deposited pheromone

Ant Colony Optimization (ACO)

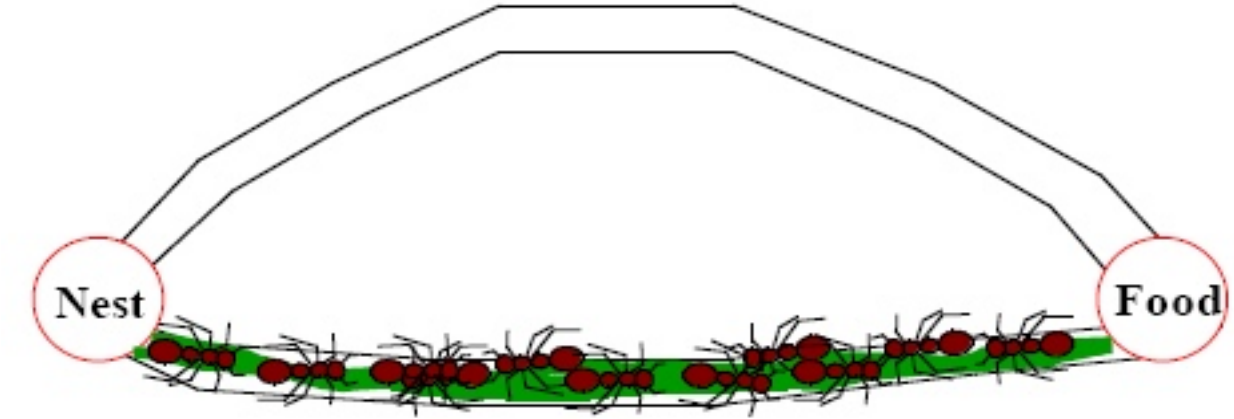
- Ant foraging – Co-operative search by pheromone trails.



For the same reason, new ants that later starts out from the nest to find food will also choose the shortest path.

Ant Colony Optimization (ACO)

- Ant foraging – Co-operative search by pheromone trails.



Over time, this positive feedback (autocatalytic) process prompts all ants to choose the shorter path

Ant Colony Optimization (ACO)

- Algorithm

```
Begin;
    Initialize the pheromone trails and parameters;
        Generate population of m solutions (ants);
        For each individual ant k∈m: calculate fitness (k);
        For each ant determine its best position;
        Determine the best global ant;
        Update the pheromone trail;
    Check if termination=true;
End;
```

Ant Colony Optimization (ACO)

- Characteristics
  - Exploit a positive feedback mechanism
  - Demonstrate a distributed computational architecture
  - Exploit a global data structure that changes dynamically as each ant transverses the route
  - Has an element of distributed computation to it involving the population of ants
  - Involves probabilistic transitions among states or rather between nodes