UNIVERSITY OF LIVERPOOL

COMP 532
Machine Learning and BioInspired Optimization

Lecture 24: Swarm Intelligence

Dr. Shan Luo

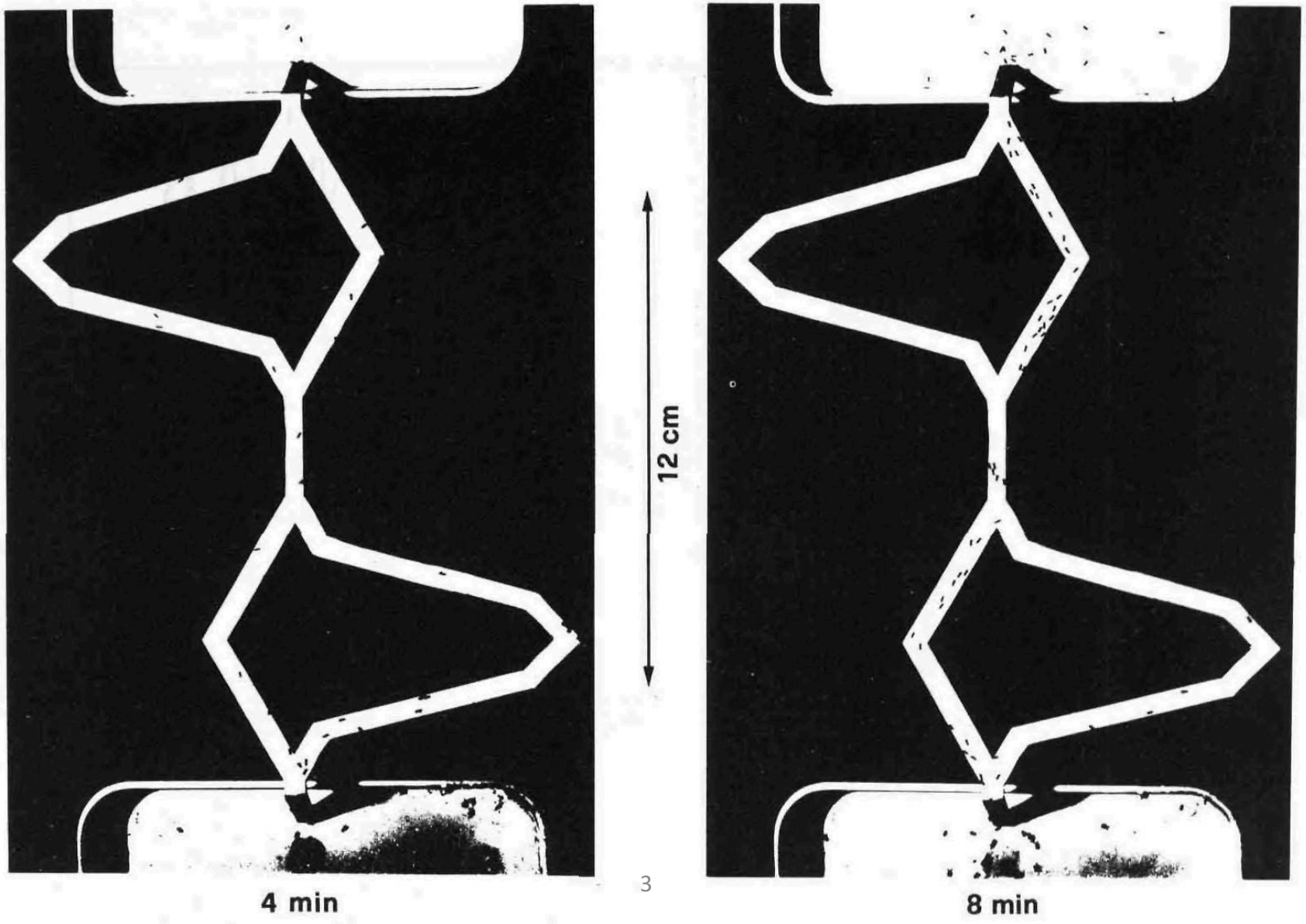
Department of Computer Science

shan.luo@liverpool.ac.uk

Overview

- What is Swarm Intelligence?
- Particle Swarm Optimization
- Ant Systems and Ant Colony Optimization
  - The basic Ant System algorithm
  - Example: Travelling Salesman Problem
  - The ACO metaheuristic
- Bee Colonies and Swarm Robotics

Recap: Deneubourg Bridge



Shortest Paths

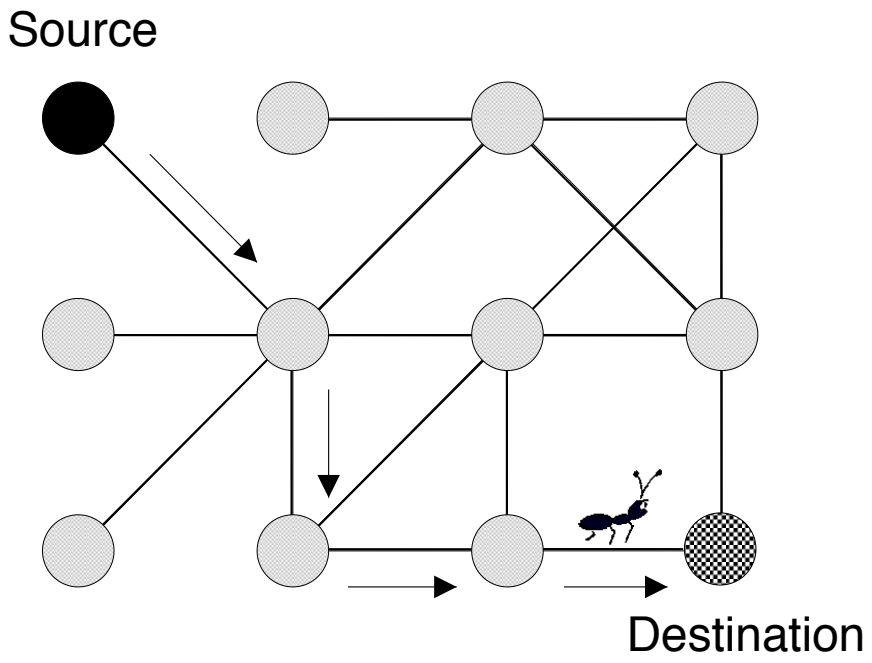
- Ants are pretty good at finding a shortest path
  - By depositing **pheromones** ants **reinforce** their trail
  - Shorter path = shorter travel time = more roundtrips = more pheromone
  - **Autocatalytic**: more pheromone leads to more ants, leads to more pheromone… etc
- We can mimic this behaviour to solve e.g. **logistics problems**!
- First formalised by **Marco Dorigo** and colleagues in the early 1990s

Dorigo, M., et al., 1996. Ant system: optimization by a colony of cooperating agents. *T-SMC (Cybernetics)*.

From Real Ants to Artificial Ants

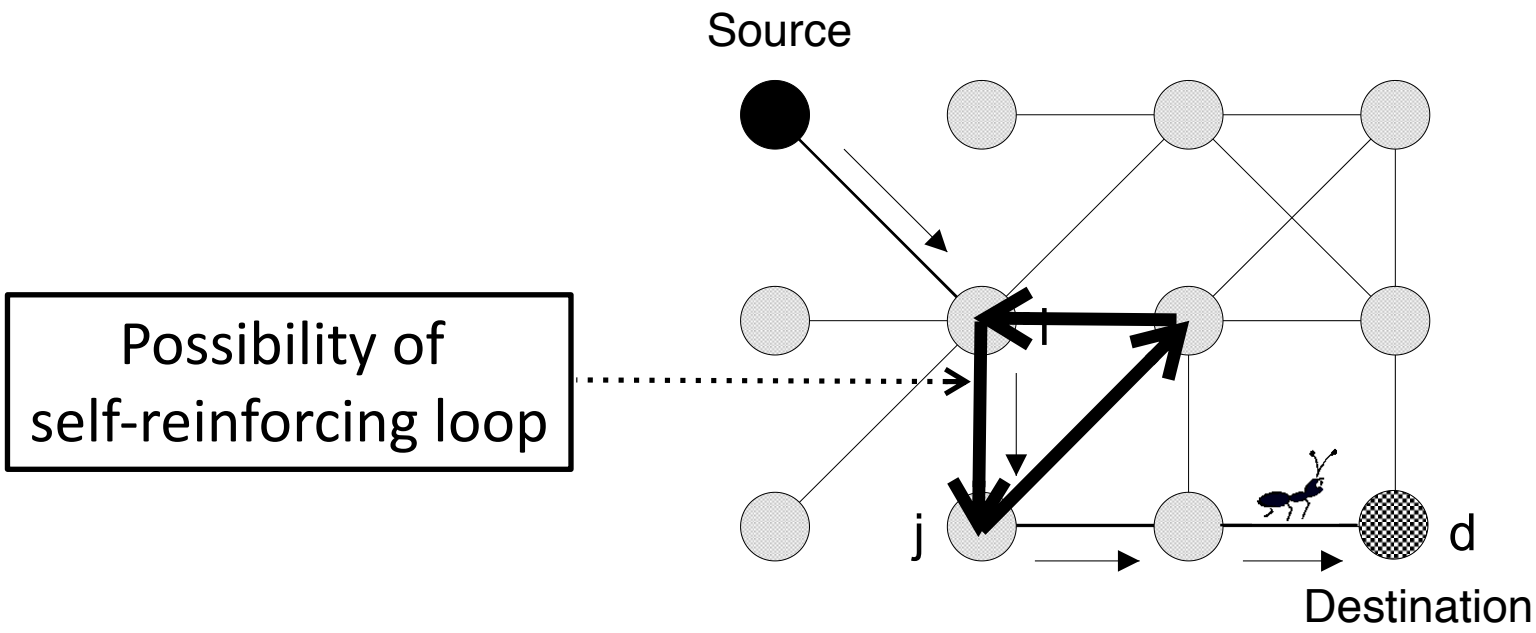
Basic Idea:

- Ants placed on a graph
- Pheromones are deposited on edges
- Ants choose next edge with probability proportional to pheromone level



From Real Ants to Artificial Ants

The direct extension of the real ant behavior (forward/backward trail deposit)
to artificial ants moving on a graph doesn't work:
problem of **self-reinforcing loops**



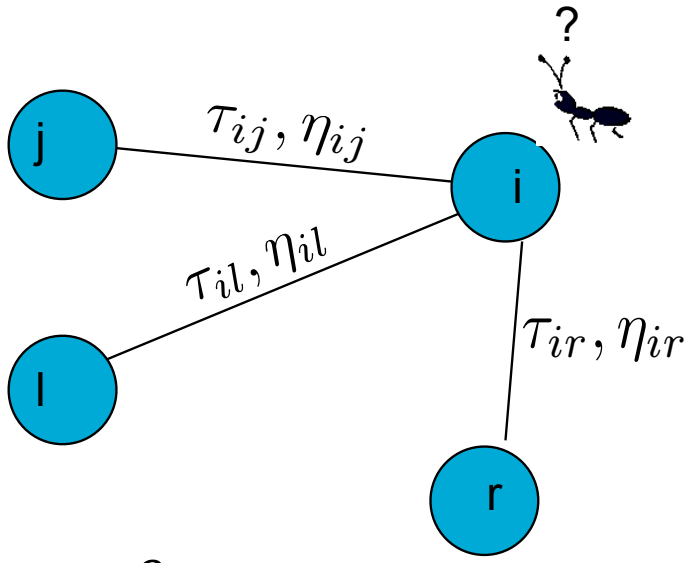
Artificial Ants

- Ants are launched at regular instants from each node to randomly chosen destinations
- Ants build solutions **probabilistically** (without updating pheromone trails), based on
  - artificial pheromone values
  - heuristic values
- Ants **memorize** visited nodes and costs incurred (path length)
- Once reached their destination, ants **retrace** their paths backwards, and update the pheromones
  - quantity of pheromone is a function of the quality of the solution (path length)

Finding a Path

Which path to take?

- Based on:
  - Pheromone level τ_{ij}
  - Heuristic info η_{ij}
- Probabilistically:

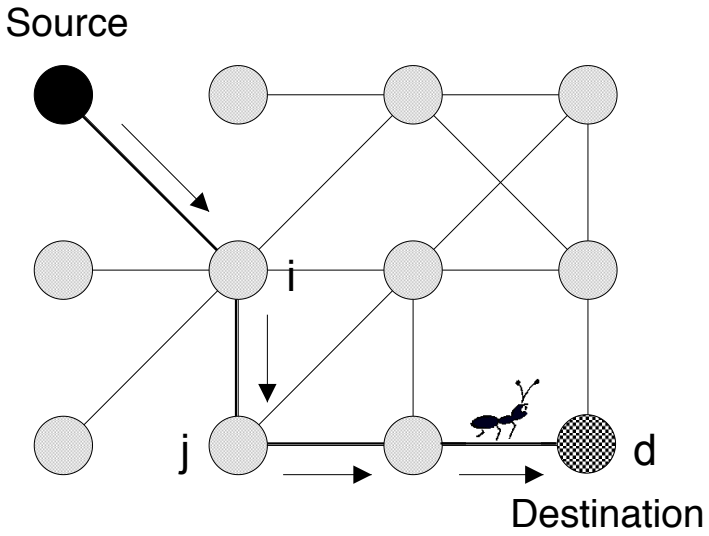


$$p_{ij} = \frac{(\tau_{ij})^{\alpha} \cdot (\eta_{ij})^{\beta}}{\sum_{k \in N_i} (\tau_{ik})^{\alpha} \cdot (\eta_{ik})^{\beta}}$$

where N_i are nodes reachable from i
and α and β are parameters

Depositing Pheromone

- Ant computes the total cost of the current path from start to destination
- Quality of the path is inversely proportional to the cost



- Pheromones along the path are updated as

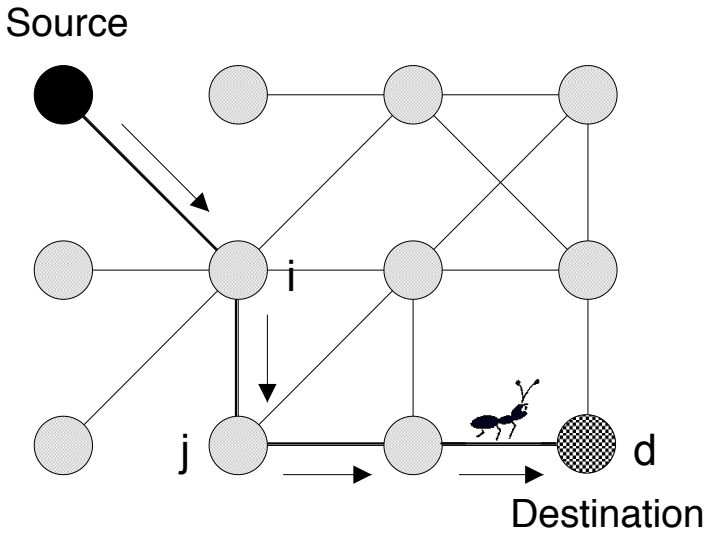
$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}$$

pheromone evaporation with rate ρ

To avoid unlimited increase of pheromone trails and allow forgetting of earlier choices

Depositing Pheromone

- Ant computes the total cost of the current path from start to destination
- Quality of the path is inversely proportional to the cost



- Pheromones along the path are updated as

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}$$

pheromone evaporation with rate ρ

pheromone deposited proportional to quality

Example: Traveling Salesman Problem

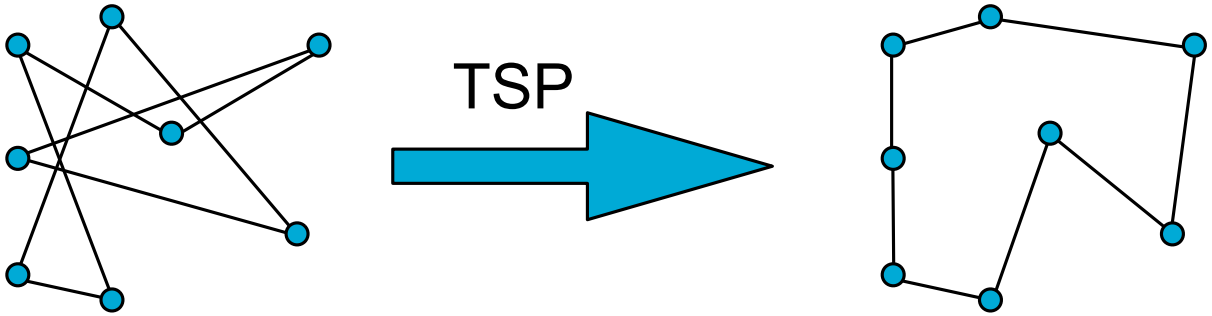
Famous Combinatorial Optimization problem (NP hard)

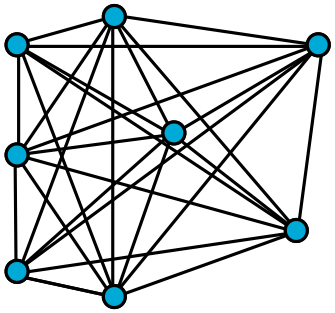
- N – set of nodes (cities), $|N| = n$
- A – set of arcs, fully connecting N
- Weighted graph $G = (N, A)$
- Each arc has a weight d_{ij} – distance between i and j
- Problem:

Find minimum length Hamiltonian circuit

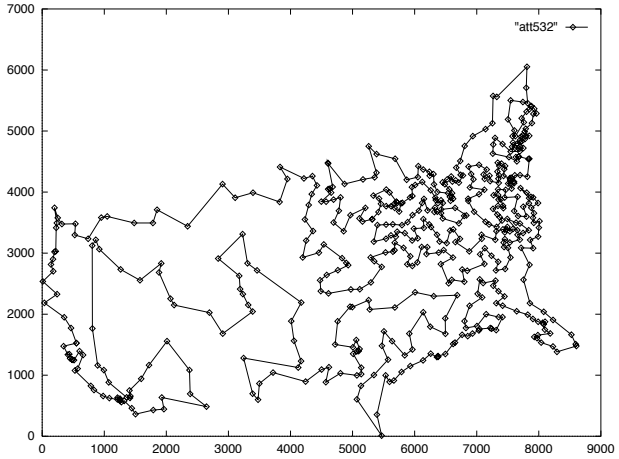
(minimum length path passing through each city exactly once)

Traveling Salesman Problem





Construction graph used by artificial ants



The TSP

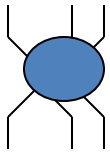
Traveling SalesAnt?

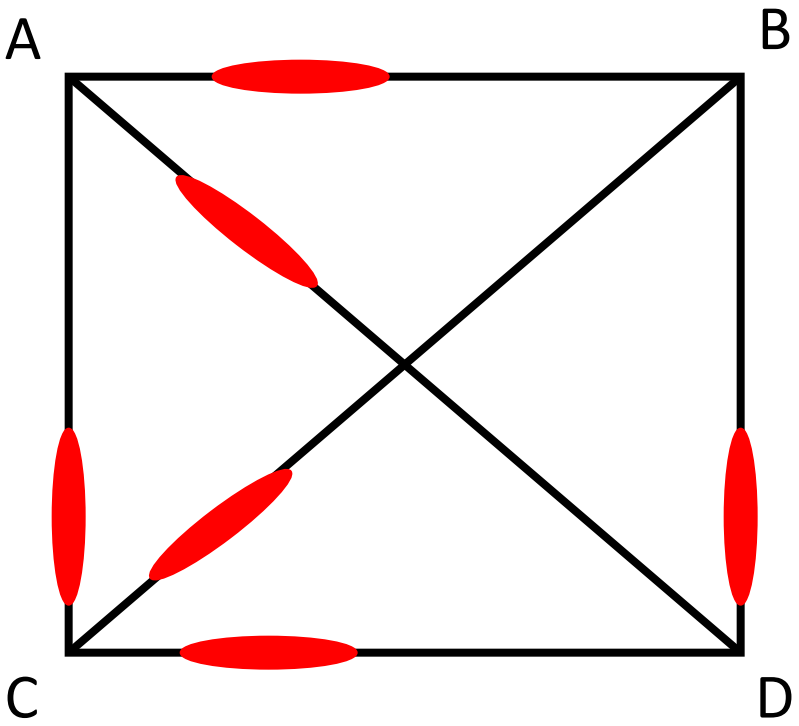
Each ant is a simple agent with the following characteristics:

- It chooses the next city to go to with a probability that is a function of the distance and of the amount of pheromone on the connecting edge;
- To force the ant to make legal tours, transitions to already visited cities are disallowed until a tour is complete;
- When it completes a tour, it deposits artificial pheromones on each edge (i, j) in the tour.
- Amount of pheromone depends on the total distance of all edges in the tour.

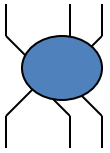
E.g. A 4-city TSP

Initially, random levels of pheromone are scattered on the edges





Pheromone

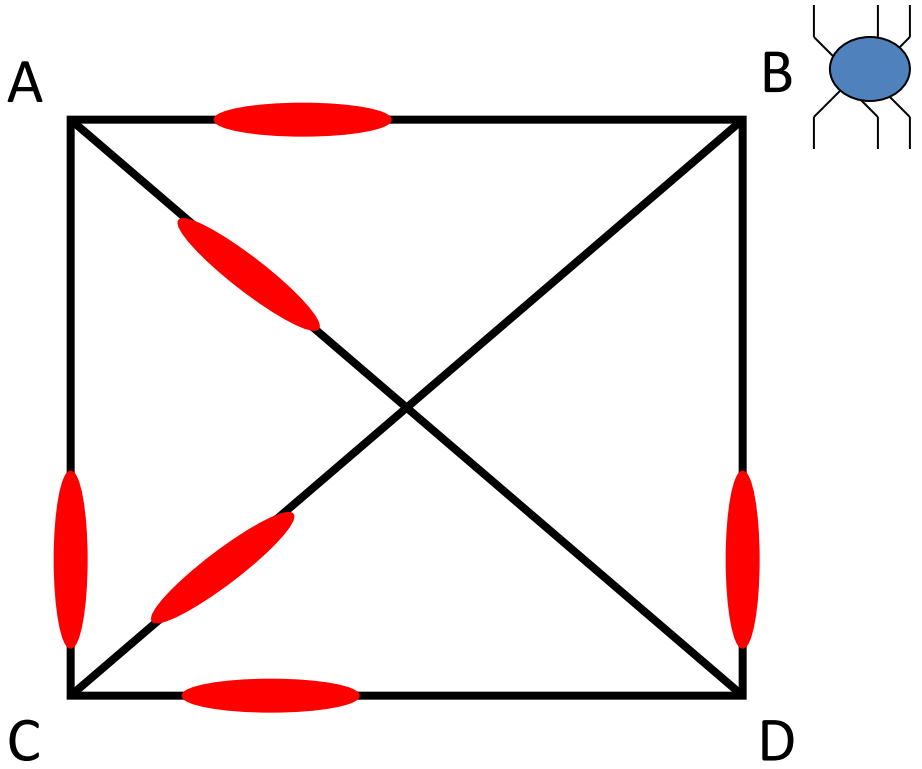


Ant

AB: 20, AC: 30, AD: 10, BC: 10, BD: 30, CD: 20

E.g. A 4-city TSP

An ant is placed at a random node



AB: 20, AC: 30, AD: 10, BC: 10, BD: 30, CD: 20

E.g. A 4-city TSP

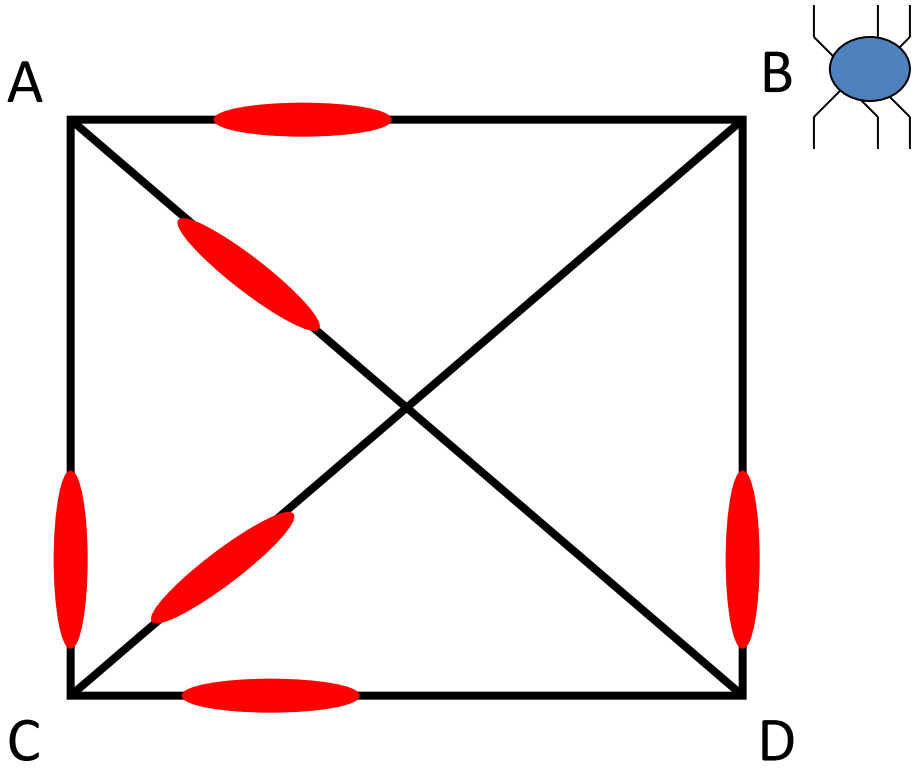
The ant decides where to go from that node,

based on probabilities

calculated from:

- pheromone strengths,
- next-hop distances.

Suppose this one chooses BC



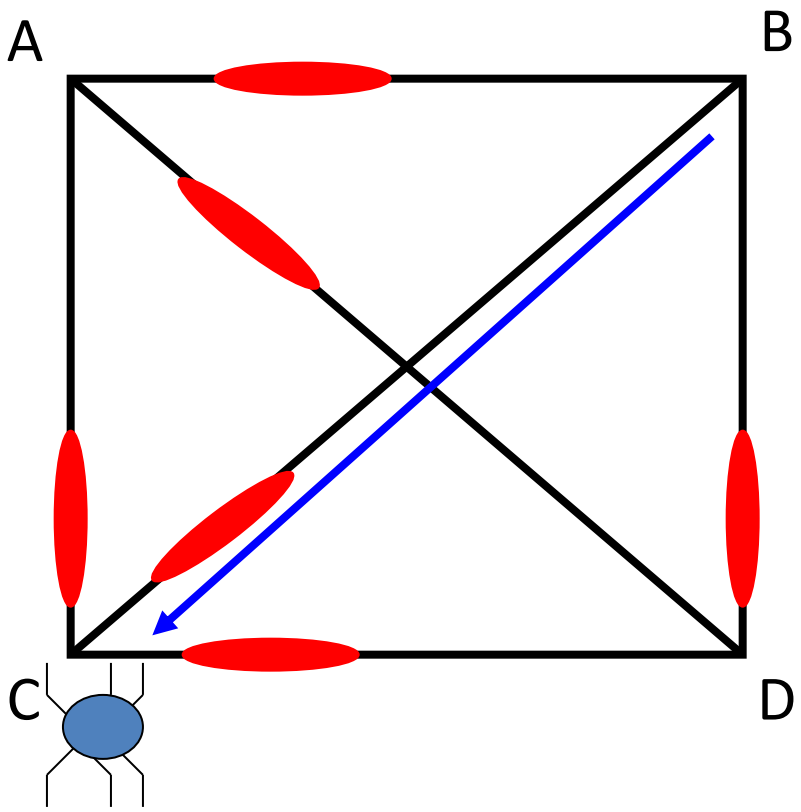
AB: 20, AC: 30, AD: 10, BC: 10, BD: 30, CD: 20

E.g. A 4-city TSP

The ant is now at C, and has a ‘tour memory’ = {B, C} – so he cannot visit B or C again.

Again, he decides next hop (from those allowed) based on pheromone strength and distance;

Suppose he chooses CD

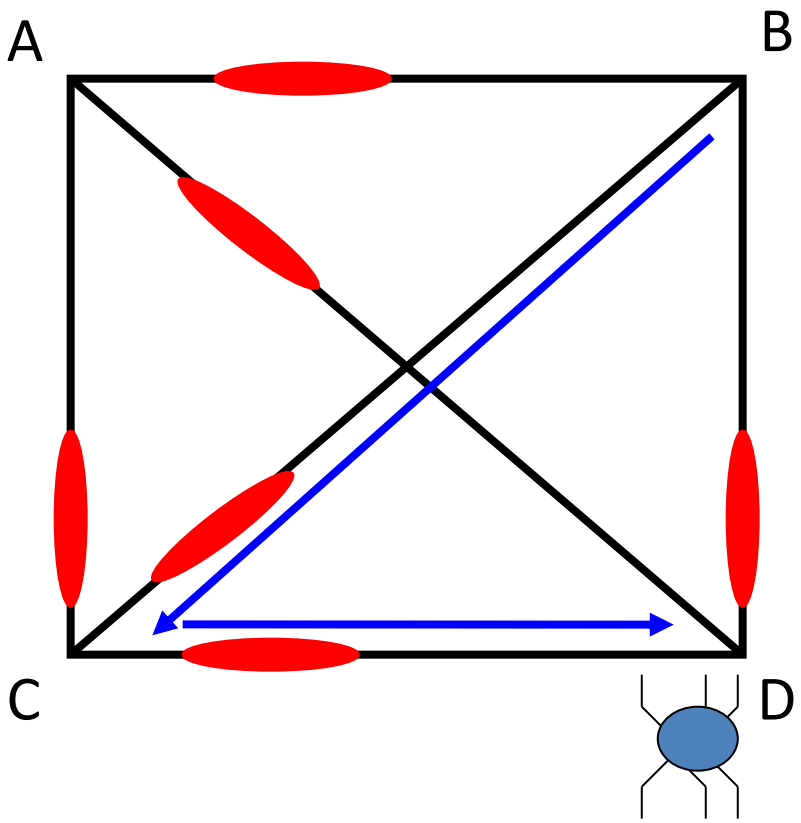


AB: 20, AC: 30, AD: 10, BC: 10, BD: 30, CD: 20

E.g. A 4-city TSP

The ant is now at D, and has a ‘tour memory’ = {B, C, D}

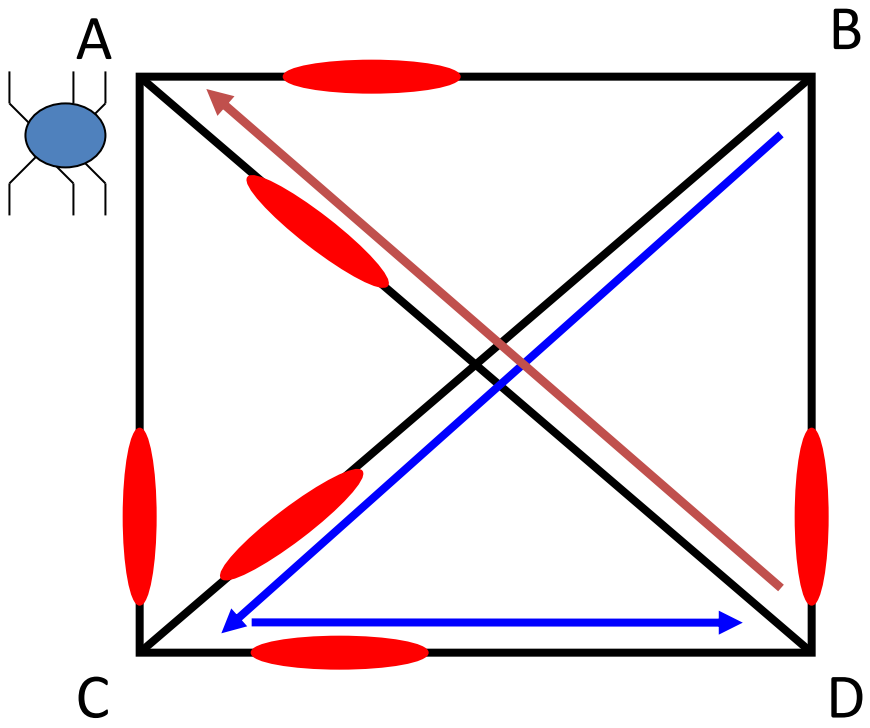
There is only one place he can go now:



AB: 20, AC: 30, AD: 10, BC: 10, BD: 30, CD: 20

E.g. A 4-city TSP

So, he has nearly finished his tour, having gone over the links: BC, CD, and DA.

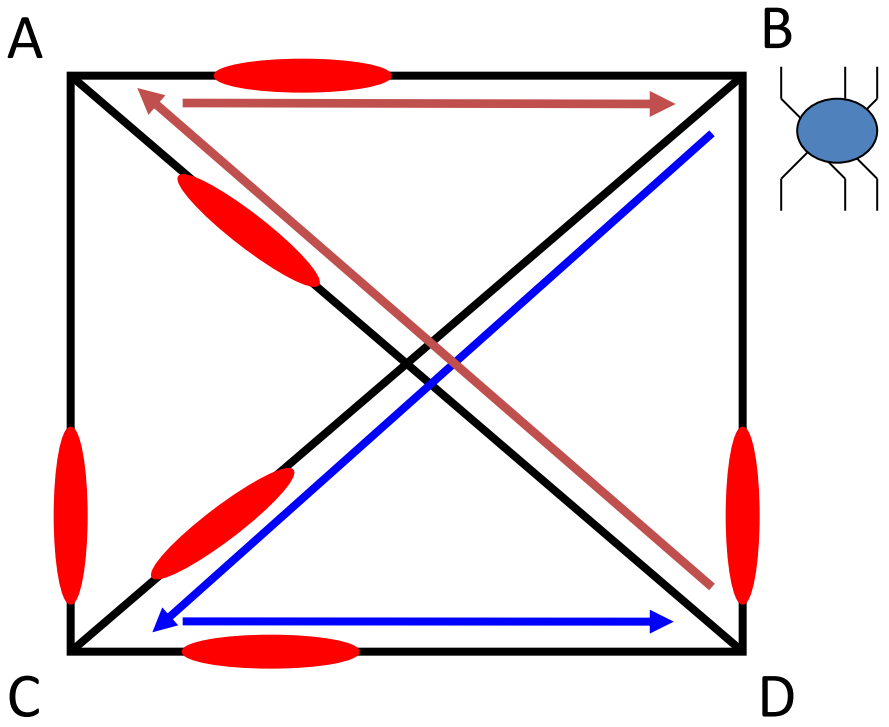


AB: 20, AC: 30, AD: 10, BC: 10, BD: 30, CD: 20

E.g. A 4-city TSP

So, he has nearly finished his tour, having gone over the links: BC, CD, and DA. AB is added to complete the round trip.

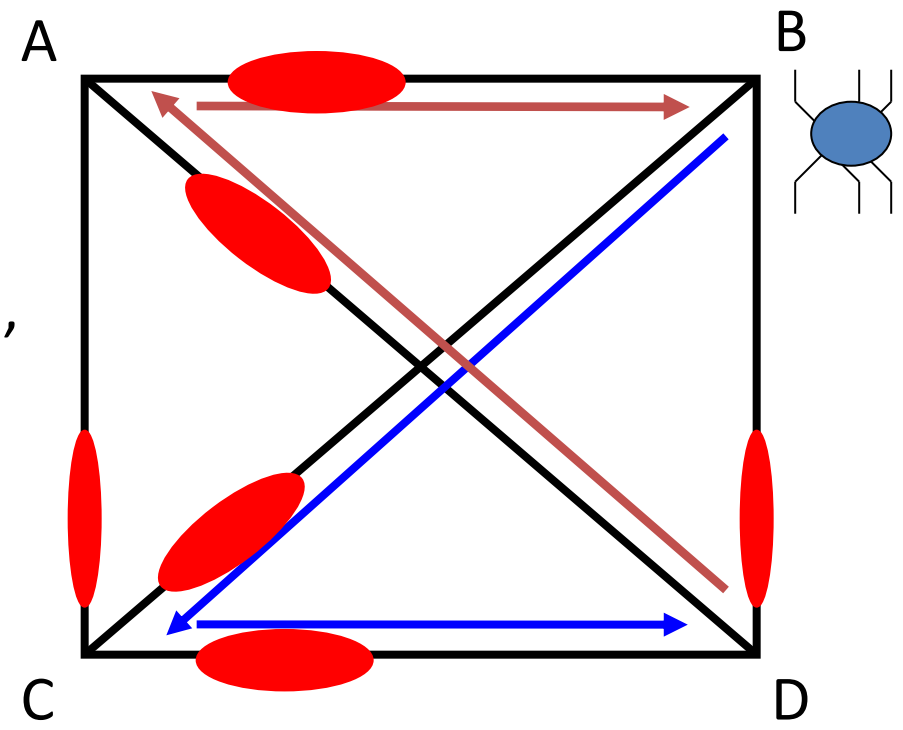
Now, pheromone on the tour is increased, in line with the fitness of that tour.



AB: 20, AC: 30, AD: 10, BC: 10, BD: 30, CD: 20

E.g. A 4-city TSP

Next, pheromone every-where is decreased a little, to model decay of trail strength over time



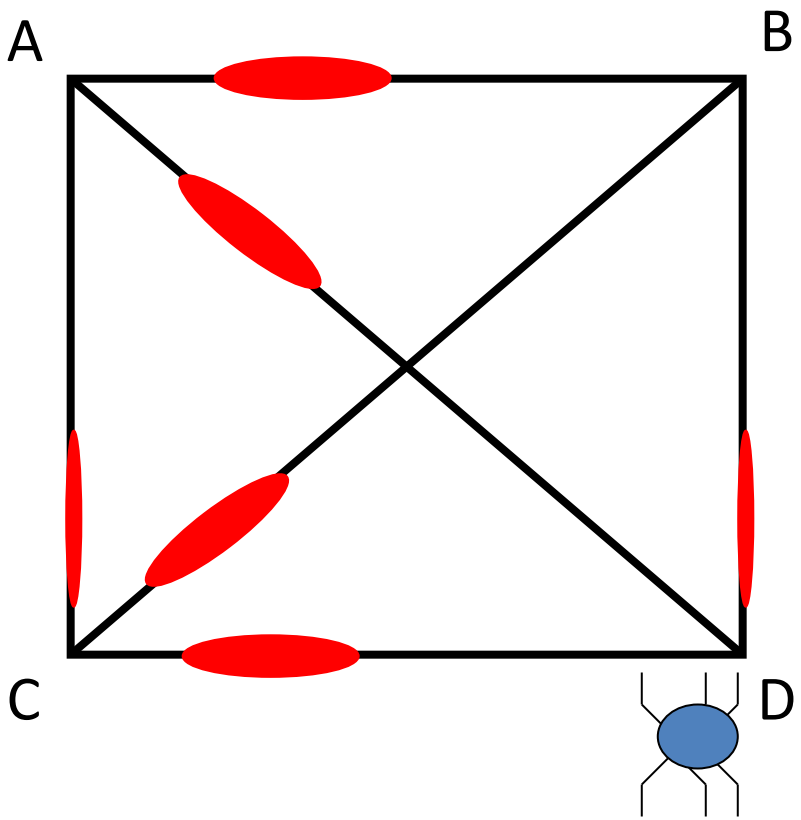
AB: 20, AC: 30, AD: 10, BC: 10, BD: 30, CD: 20

E.g. A 4-city TSP

We start again, with another ant in a random position.

Where will he go?

Next, the actual algorithm and variants.



AB: 20, AC: 30, AD: 10, BC: 10, BD: 30, CD: 20

The Ant System Algorithm

LOOP

Place one ant on each city * #_cities = #_ants *

FOR step := 1 to #_ants * each ant builds a tour *

FOR k := 1 to #_cities * each ant adds a city to its path *

Choose the next city to move to applying a probabilistic state transition rule

END-FOR

END-FOR

Update pheromone trails

UNTIL End_condition

Dorigo *et al.* (1991)

The State Transition Rule

$$p_{ij} = \frac{(\tau_{ij})^{\alpha} \cdot (\eta_{ij})^{\beta}}{\sum_{k \in A_i} (\tau_{ik})^{\alpha} \cdot (\eta_{ik})^{\beta}}$$

A_i are cities reachable from i which have not yet been visited

α controls the importance of pheromones

β controls the importance of heuristic info

η_{ij} is the visibility, $1/d_{ij}$

Pheromone Update

After all ants have built a tour, pheromone trails are updated on **all** edges (i, j) as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} \quad \forall (i,j)$$ evaporation

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij} \quad \forall (i,j)$$ pheromone deposit

where $\Delta\tau_{ij} = \sum_k \Delta\tau_{ij}^k$

and $\tau_{ij}^k = {}^1\!/_{L^k}$

where L^k is the length of the solution found by ant k

Modified Versions of the Ant System

- **Elitist AS** (EAS) (Dorigo et al., 1991; 1996)
  - The iteration best solution adds more pheromone
- **Rank-Based AS** (ASrank) (Bullnheimer et al., 1997; 1999)
  - Only best ranked ants can add pheromone
  - Quantity of pheromone added is proportional to rank
- **Max-Min AS** (MMAS) (Stützle & Hoos, 1997)
  - Only iteration best or best-so-far ants can add pheromone
  - Pheromone trails have explicit upper and lower limits
  - Pheromone trail initialized to upper limit
  - Pheromone trail are re-initialized when stagnation
- **Ant Colony System** (ACS) (Gambardella & Dorigo, 1996)
  - Pheromones are updated also while building solutions
  - Only iteration best or best-so-far ants can add pheromone
  - Local search added

Ant Colony System

Three main ideas:

- Different **state transition** rule
- Different **global pheromone trail update** rule
- New **local pheromone trail update** rule

ACS' State Transition Rule

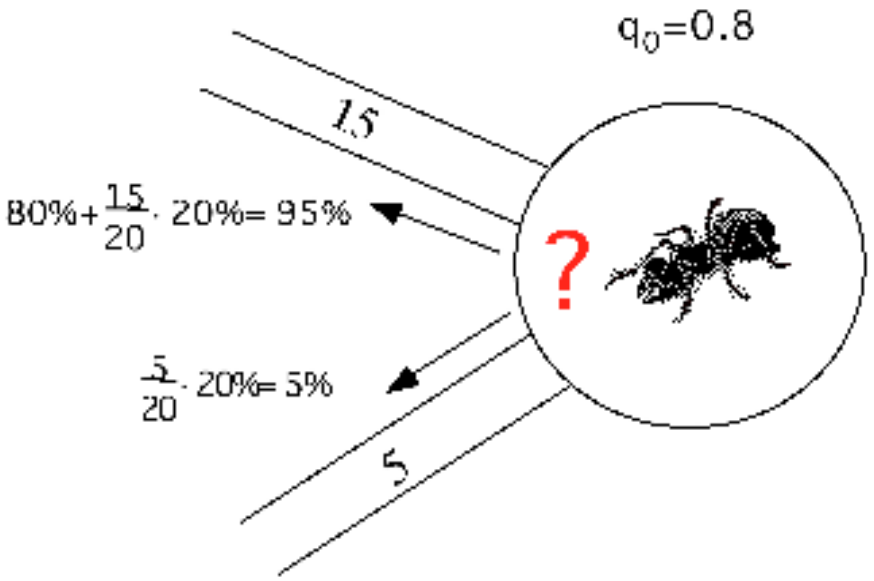
Next state:

with probability q_0

exploitation

with probability $(1 - q_0)$

biased exploration



ACS' State Transition Rule

Ants select next city j as:

$$j = \begin{cases} \arg\max_{j \in A_i} \left\{ (\tau_{ij}) \cdot (\eta_{ij})^{\beta} \right\} & \text{if } q < q_0 \\ J & \text{else} \end{cases}$$

where J is a probability distribution following

$$p_{ij} = \frac{(\tau_{ij}) \cdot (\eta_{ij})^{\beta}}{\sum_{k \in A_i} (\tau_{ik}) \cdot (\eta_{ik})^{\beta}}$$

This allows to tune **exploration** and **exploitation**

ACS' Pheromone Update Rule

Two pheromone update rules

- **Global**: instead of updating pheromones on all new tours, only update the **best tour found so far**

- **Local**: also update when constructing the tour

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\tau_0$$

this introduces **diversification**

The ACO Metaheuristic

What is a metaheuristic?

- A set of algorithmic concepts
- Can be used to define heuristic methods
- Applicable to a wide set of problems

Currently two major application classes:

- Routing in telecommunications networks
- *NP*-hard combinatorial optimization problems

The ACO Metaheuristic

```
procedure ACO-metaheuristic()
    while (not-termination-criterion)
        schedule subprocedures
            AntsConstructSolutions()
            PheromoneUpdates()
            Problem-specific-actions() /* Optional */
        end schedule subprocedures
    end while
end procedure
```

These are problem specific actions, such as local search

The ACO Metaheuristic

- Main **characteristics** of this class of algorithms
  - natural metaphor
  - stochastic nature
  - adaptivity
  - inherent parallelism
  - positive feedback

- **Applications**:
  - TSP-like combinatorial optimization problems
  - Routing in networks
  - Scheduling
  - And many more…

Wrap up

- Ant Systems and Ant Colony Optimization
  - The basic Ant System algorithm
  - Example: Travelling Salesman Problem
  - The ACO metaheuristic

- Next lecture: Bee Colonies and Swarm Robotics