13 Ant Colony Optimisation

Ant Colony Optimisation

- introduced by Dorigo in 1992
- inspired by the foraging behaviour of ants
- because ants are interesting!
  - they solve complex tasks by simple local means
  - ant productivity is better than the sum of their single activities
  - ants are 'grand masters' in search and exploitation
- for example, they are able to find the shortest path from food to nest
  - but how do they do it?

Ant Colony Optimisation

信息素

- to find the shortest path they use pheromone
- ants deposit pheromone as they move along a path
- this is used by other ants to help them decide which route to take
  - who consequently lay even more pheromone along the same path
  - a positive feedback mechanism
- so they alter the local environment in order to indirectly communicate with each other
  - a process known as stigmergy - '*stimulation by work*'
- and their combined actions lead towards optimal solutions
- this leads to robustness and adaptability

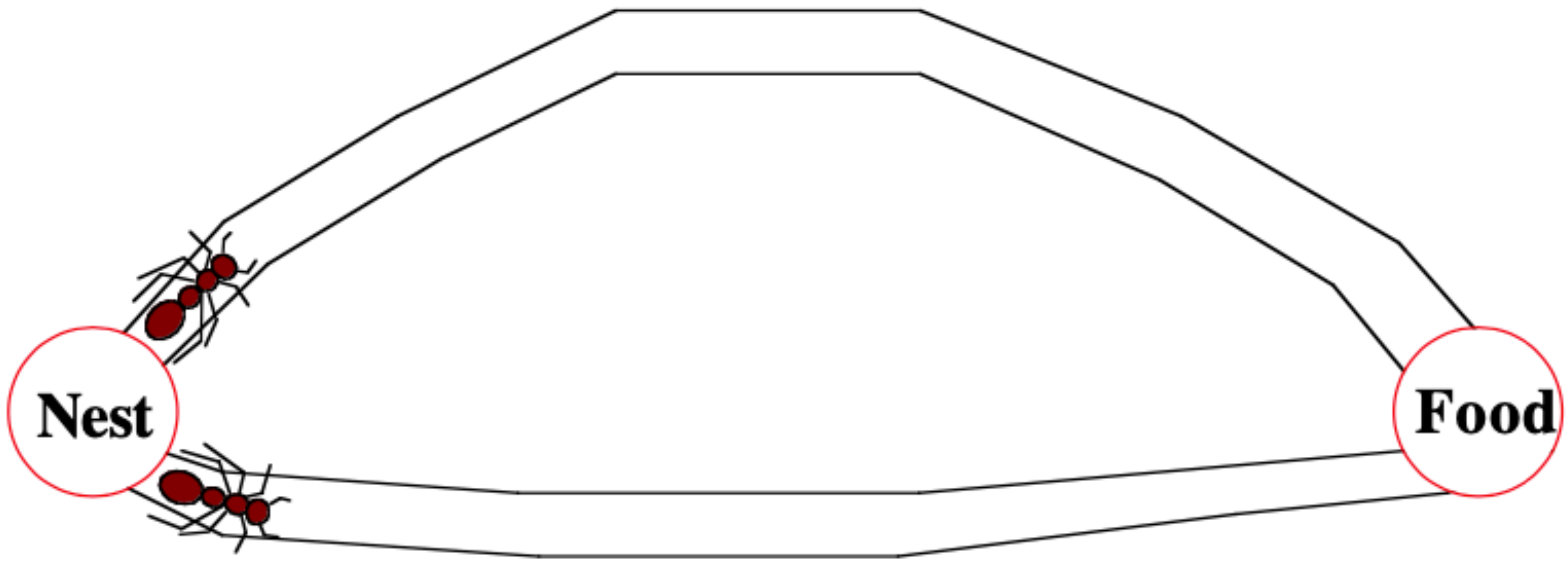
Stigmergy（斯蒂格默基）是一种社会学和生物学概念，用于描述在没有任何集中式控制或通信的情况下，一组个体如何通过间接相互作用来协同工作。在中文中，通常将其翻译为"诱导性协作"、"诱导协同"或"诱导合作"。

Ant Foraging Example

- suppose there are two paths from nest to food
- initially each has no pheromone laid on it
  - so each is equally likely to be chosen
- two ants set off from the nest, one down each path…

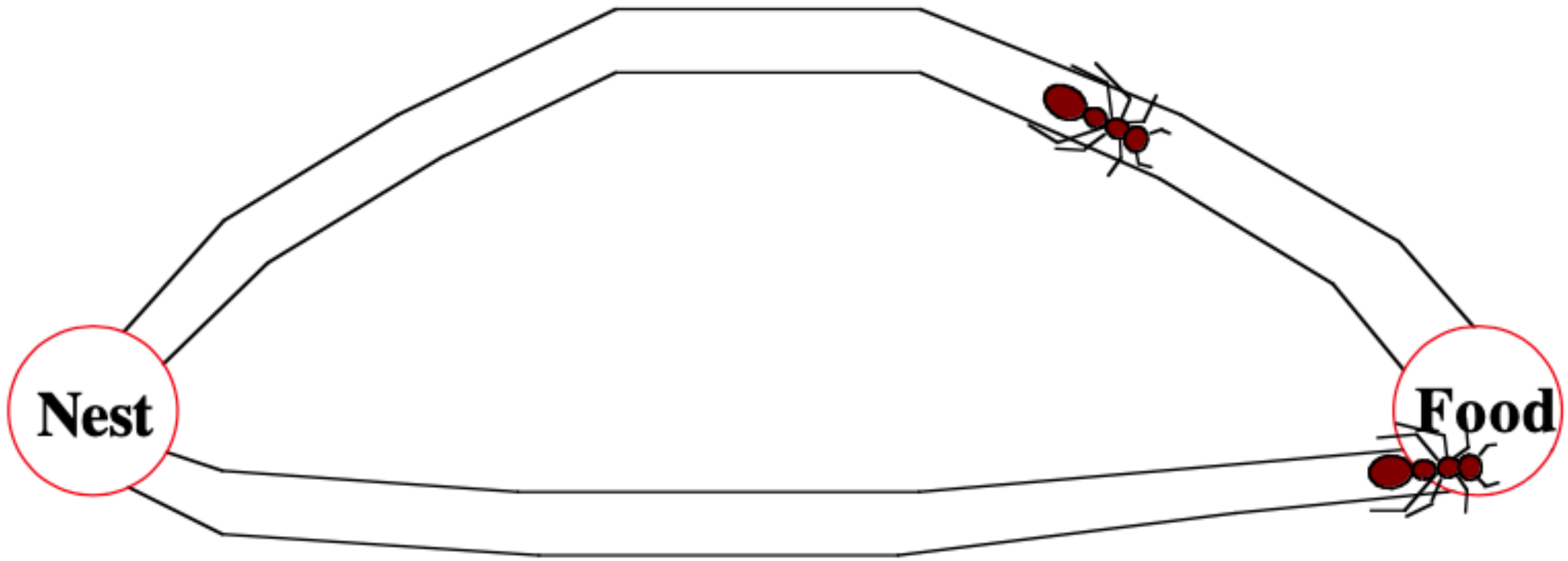
Ant Foraging Example

two ants set off from the nest, one down each path…



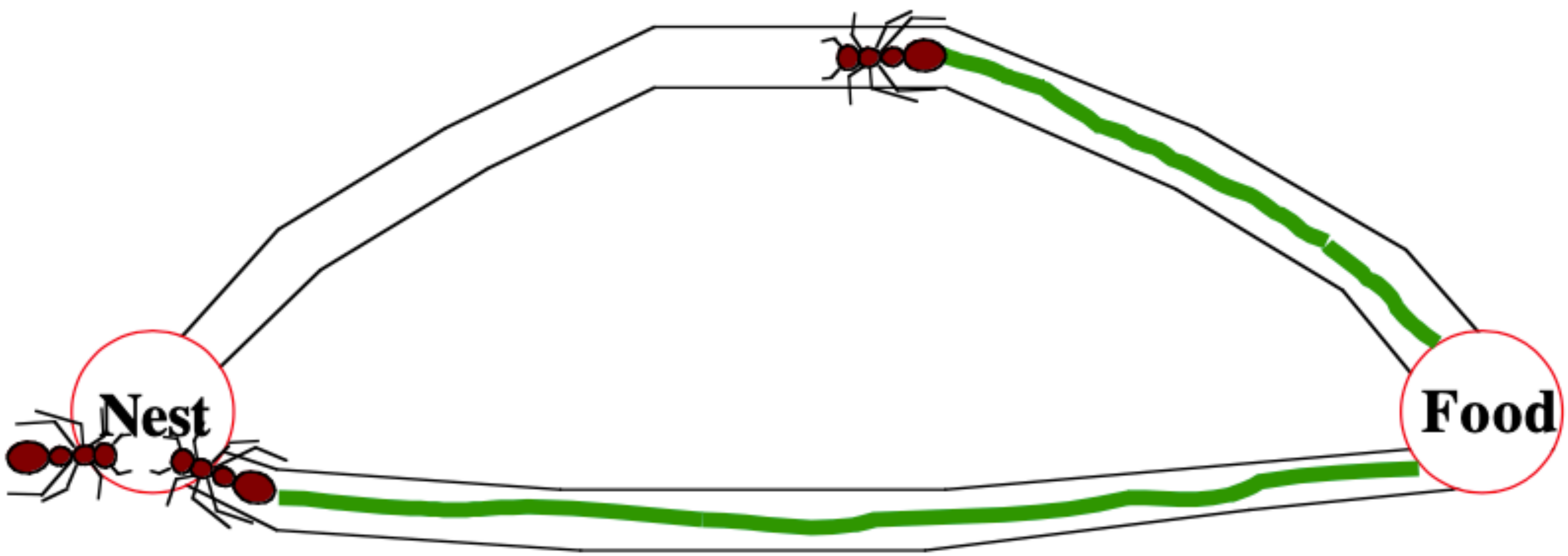
Ant Foraging Example

…the ant on the shorter path can make more journeys…



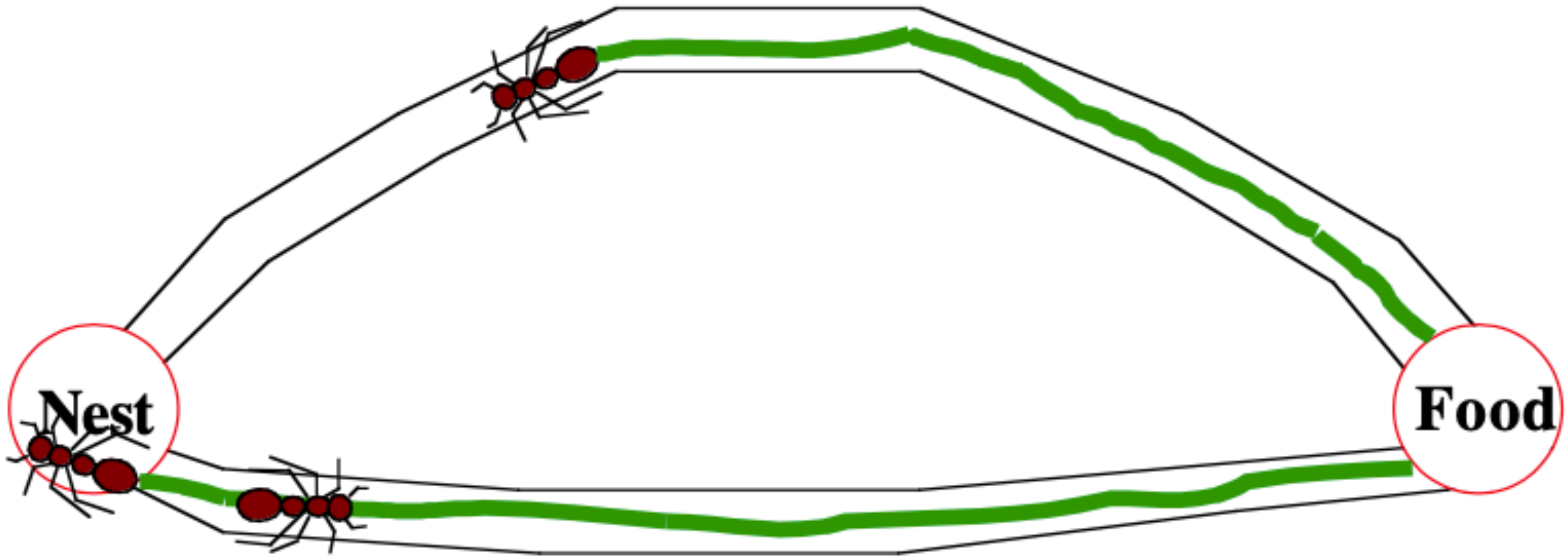
Ant Foraging Example

…so over time it lays more pheromone on its path than the other ant does…



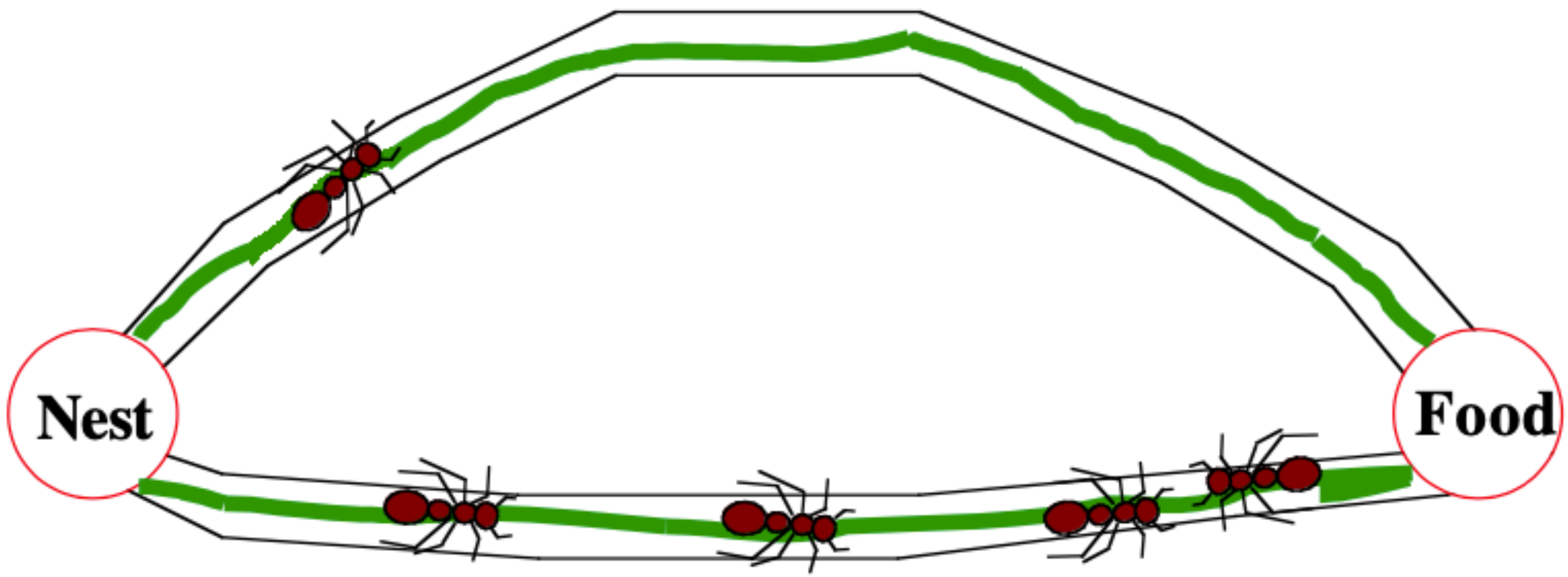
Ant Foraging Example

…so the next ant out of the nest is more likely to follow the shorter route…



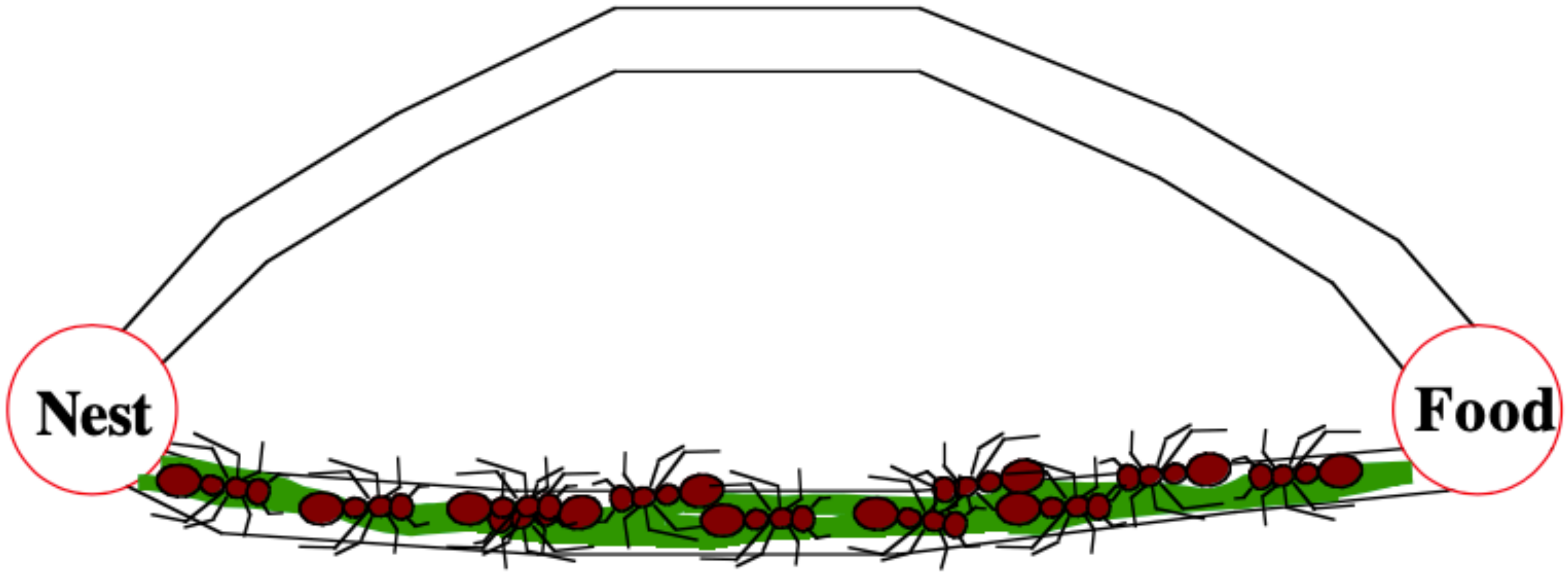
Ant Foraging Example

…and as more ants leave the nest, more are likely to use the path with the higher pheromone level, further reinforcing it….



Ant Foraging Example

…until the shorter path is almost always used.



Evaporation

- so, in the real world, ants initially wander randomly, and upon finding food return to their colony
  - all the time laying down pheromone trails
- other ants are likely to follow an existing trail, returning and reinforcing it if they eventually find food
- over time, however, the pheromone trail starts to evaporate
  - which reduces 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

Evaporation Helps Avoid Local Optima

- a short path gets marched over faster than a long path
- so the pheromone density remains high
  - it's laid at least as fast as it can evaporate
- pheromone evaporation is crucial for avoiding convergence to a locally optimal solution
- without evaporation the paths chosen by the first ants would tend to be excessively attractive to the following ones
- so exploration of the solution space would be constrained

ACO Algorithm

- a population based metaheuristic
- used to find approximate solutions to difficult optimisation problems
- the optimisation problem is encoded as the environment
    - usually a graph, with weighted edges
- a population of ants is created, usually divided into generations

ACO Algorithm

- each generation of ants traverse the graph and builds their own solutions
    - find their own paths
- each ant lays pheromone on edges proportionate to the quality of its solution
- successive generations converge towards optimal paths
    - helping to build an optimal solution

ACO Algorithm

- the basic algorithm begins with initialisation:
  - set up nodes
  - create edges between nodes, and for each edge:
    - set edge weighting
    - set pheromone level
- these three procedures are then performed each generation:
  - Generate Ant Solutions
  - Perform Daemon Actions (optional)
  - Update Pheromone Levels

ACO: Generate Ant Solutions

- this manages the population (colony) of ants
- each generation a set of ants are created and placed at nodes
    - perhaps randomly
- each ant travels through the graph until its task is complete
- an ant needs to perform edge selection to decide which node to visit next
    - edge selection is stochastic and based on:
        - pheromone levels
        - heuristic information (such as edge-weighting)
- each ant's solution is evaluated

ACO: Edge Selection Formula

- selecting which edge to traverse next is probabilistic
- the best known rule is ant system (AS), created by Dorigo et al in 1996:

$$\text{prob(choose edge i)} = (\tau_i^\alpha . \eta_i^\beta) \,/\, \Sigma_n(\tau_i^\alpha . \eta_i^\beta)$$

where:

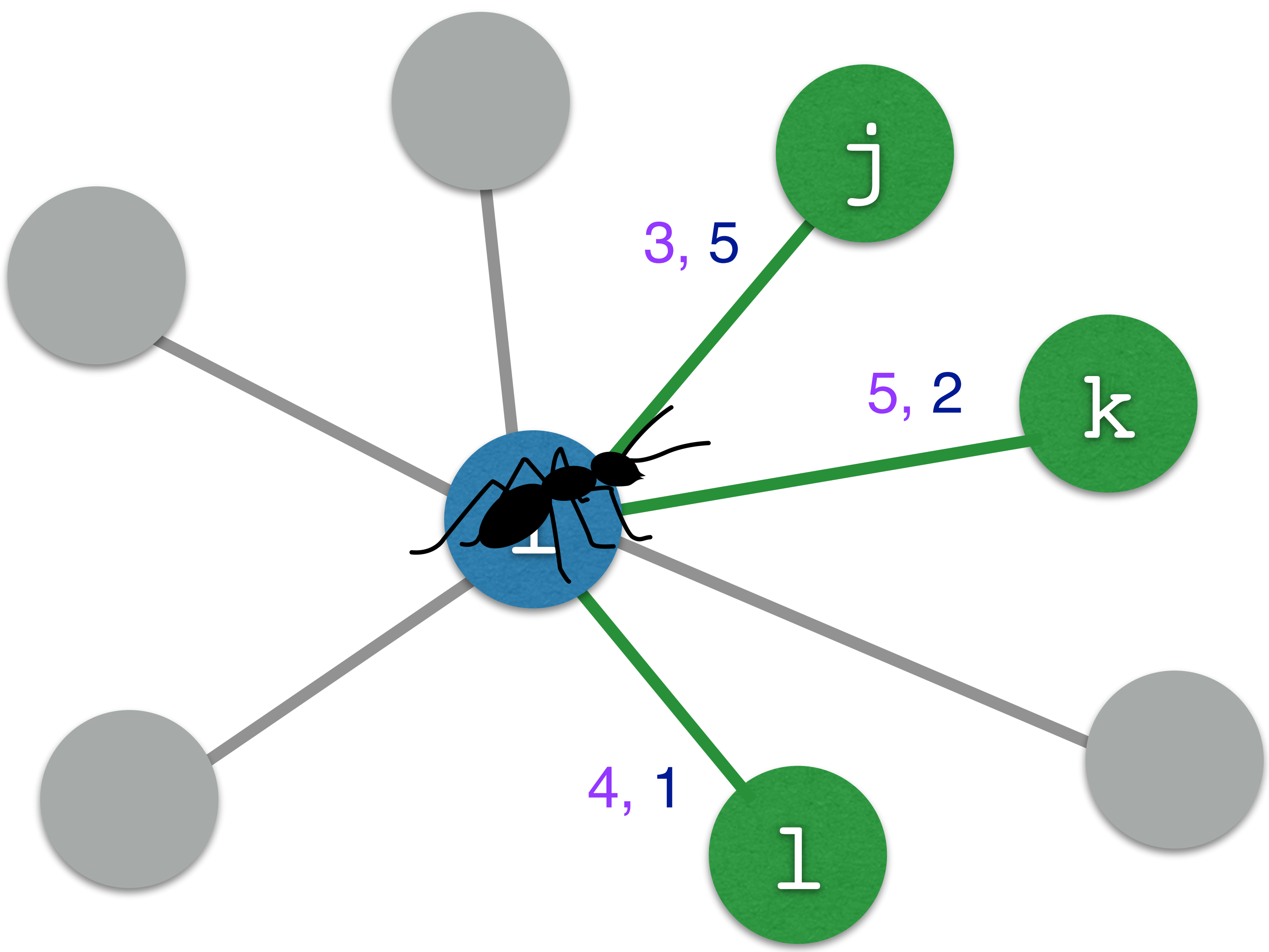
n = number of edges

τ_i = level of pheromone on edge i 'trail level'

η_i = 1/(length of edge i) 'attractiveness'

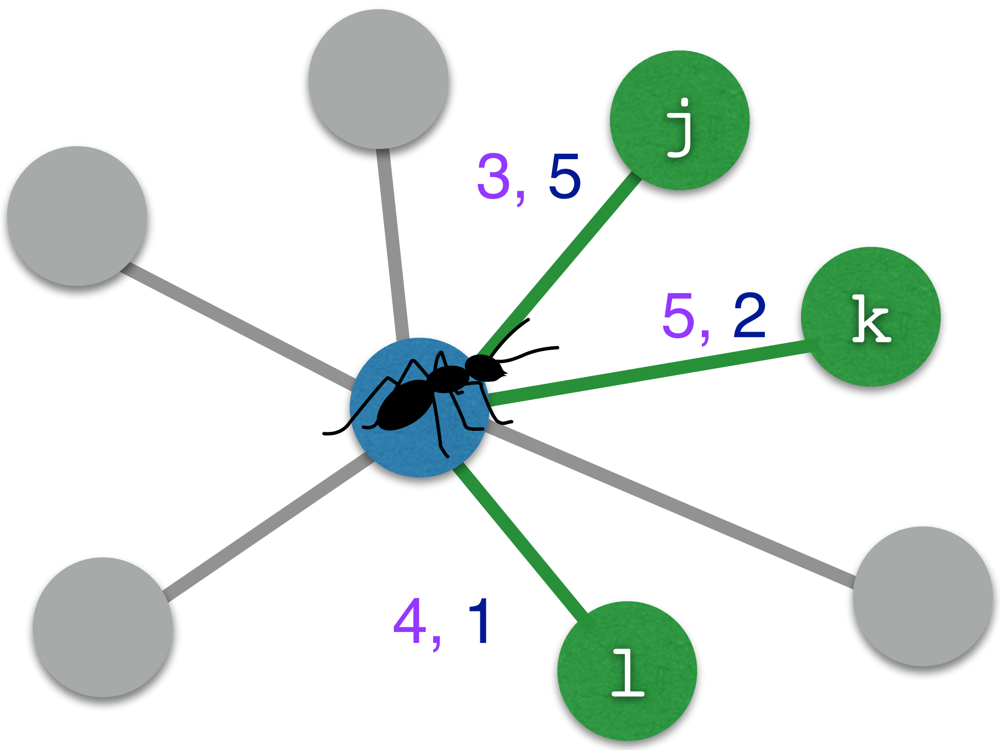
α,β are positive real parameters that determine the relative importance of pheromone versus heuristic information (edge length)

ACO: Edge Selection Example



- ant at node `i`
- 3 edges lead to nodes not yet visited
- pheromone level shown in purple
- edge length shown in blue

ACO: Edge Selection Example



using weightings:

$\alpha = 1$

$\beta = 2$

edge	τ	τ^α	η = 1÷length	η^β	$\tau^\alpha.\eta^\beta$	prob = $(\tau^\alpha.\eta^\beta) \div \Sigma$
i to j	3	3	0.2	0.04	0.12	0.12 ÷ 5.37 = **0.022**
i to k	5	5	0.5	0.25	1.25	0.25 ÷ 5.37 = **0.233**
i to l	4	4	1	1	4	4 ÷ 5.37 = **0.745**
					Σ=5.37	1.00

ACO: Perform Daemon Actions

- these optionally occur after individual ant solutions have been constructed, and before pheromone levels are updated
- they can perform actions that are problem specific or centralised
    - and so cannot be performed by individual ants
- for example, applying local search of the solutions to select which solutions are allowed to update pheromone levels

ACO: Update Pheromone Levels

- once all the ants in a generation have completed their tour, the pheromone levels on the edges are updated
- each edge's pheromone level is initially decreased by a certain percentage
    - evaporation
- each edge then receives an amount of additional pheromone
- the amount laid is proportional to the quality of the solutions to which it belongs
- this procedure is then repeated for each generation, until a termination criterion is satisfied

ACO: Update Pheromone Formula

- the pheromone update formula looks like this:

$$\tau_i \leftarrow (1-\rho).\tau_i + \rho.\Sigma(\Delta\tau_i \text{ for ant } k)$$

where:

τ_i = pheromone level on edge i

ρ = pheromone evaporation coefficient

$\Delta\tau_i$ = amount of pheromone deposited by an ant in this generation:

Q/L if the ant uses the edge in its tour

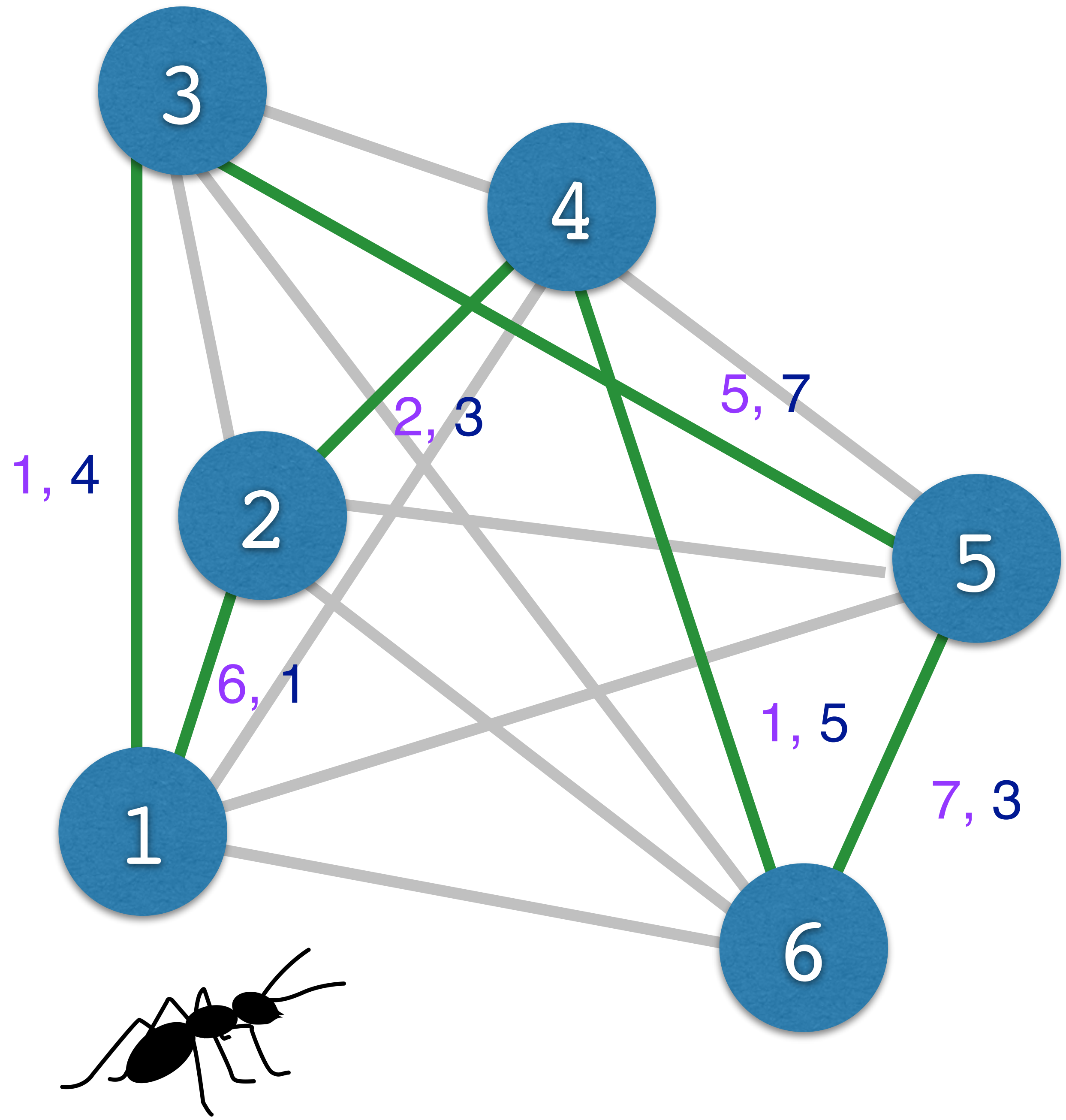
0 otherwise

where L is the tour length

Q is a constant

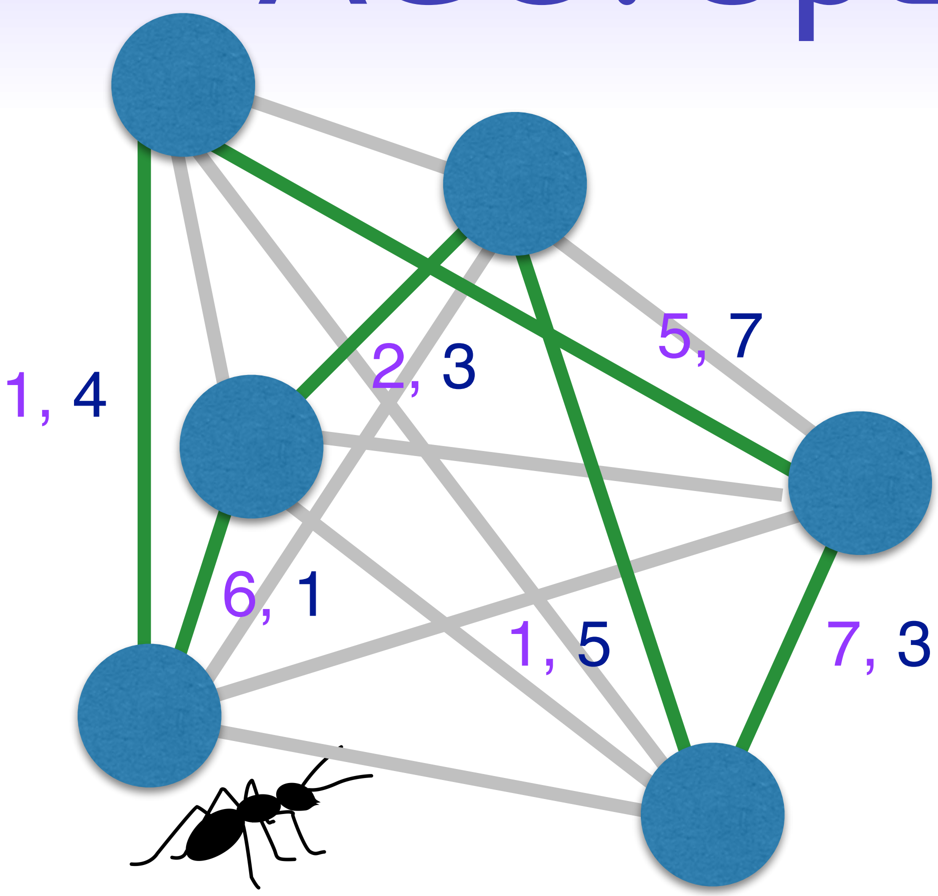
- in words:
  - perform evaporation on the current pheromone level
  - add the new pheromone laid by each ant

ACO: Update Pheromone Levels



- ant has completed its tour
- for the edges travelled:
- pheromone level shown in purple
- edge length shown in blue

ACO: Update Pheromone Levels



using :
evaporation coefficient ρ = 0.1
pheromone per ant Q = 10

first evaporate pheromone on <u>all</u> edges in the graph

edge	initial τ	after evap τ
1 to 2	6 − 0.6	5.4
2 to 4	2 -0.2	1.8
4 to 6	1 −0.1	0.9
6 to 5	7 −0.7	6.3
5 to 3	5 −0.5	4.5
3 to l	1 −0.1	0.9

…and for all other edges (9 more)

then add pheromone for <u>each</u> ant's tour

length	Δτ=Q/L	updated τ
1	0.43	**5.83**
3	0.43	**2.23**
5	0.43	**1.33**
3	0.43	**6.73**
7	0.43	**4.93**
4	0.43	**1.33**

10/23 L = 23 *(repeat for each ant)*

ACO Example: Travelling Salesman Problem

- in the traveling salesman problem (TSP) a set of locations (cities) and the distances between them are given
- the problem consists of finding a closed tour of minimal length that visits each city once and only once
- to apply ACO to the TSP we represent:
  - cities as nodes
  - distances between cities as the weightings (lengths) of edges between those nodes
- this graph is known as the construction graph

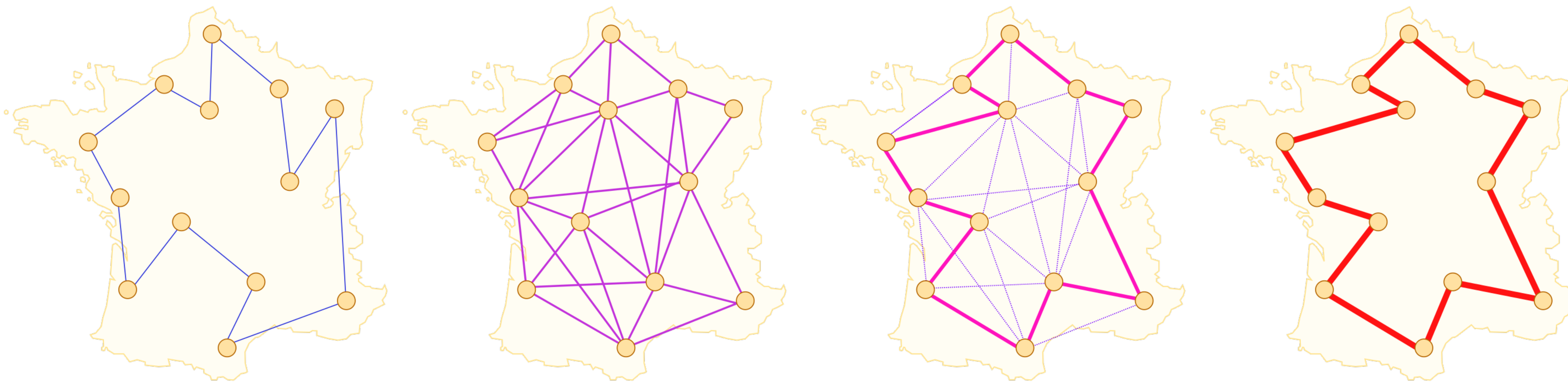
ACO for TSP

- each generation, a set of ants can be placed on random nodes in the graph
- each ent then tours the graph, visiting each city once, and recording its tour
- it selects edges by looking at:
  - whether or not the edge leads to a node already visited (valid or invalid)
  - for valid edges it probabilistically chooses which edge to traverse based on:
    - the amount of pheromone on each edge
    - the length of each edge (longer edges are less likely to be chosen)

ACO for TSP

- at the end of a generation each ant lays pheromone on the edges it traversed during its tour
- the amount of pheromone laid per edge is inversely proportional to the length of the tour
  - so shorter tours lay more pheromone per edge
  - while longer tours lay less pheromone per edge
- this information encourages ants in future generations to select edges that formed part of shorter tours
- and hence the ants collectively contribute towards finding an optimal solution: the shortest tour of the graph

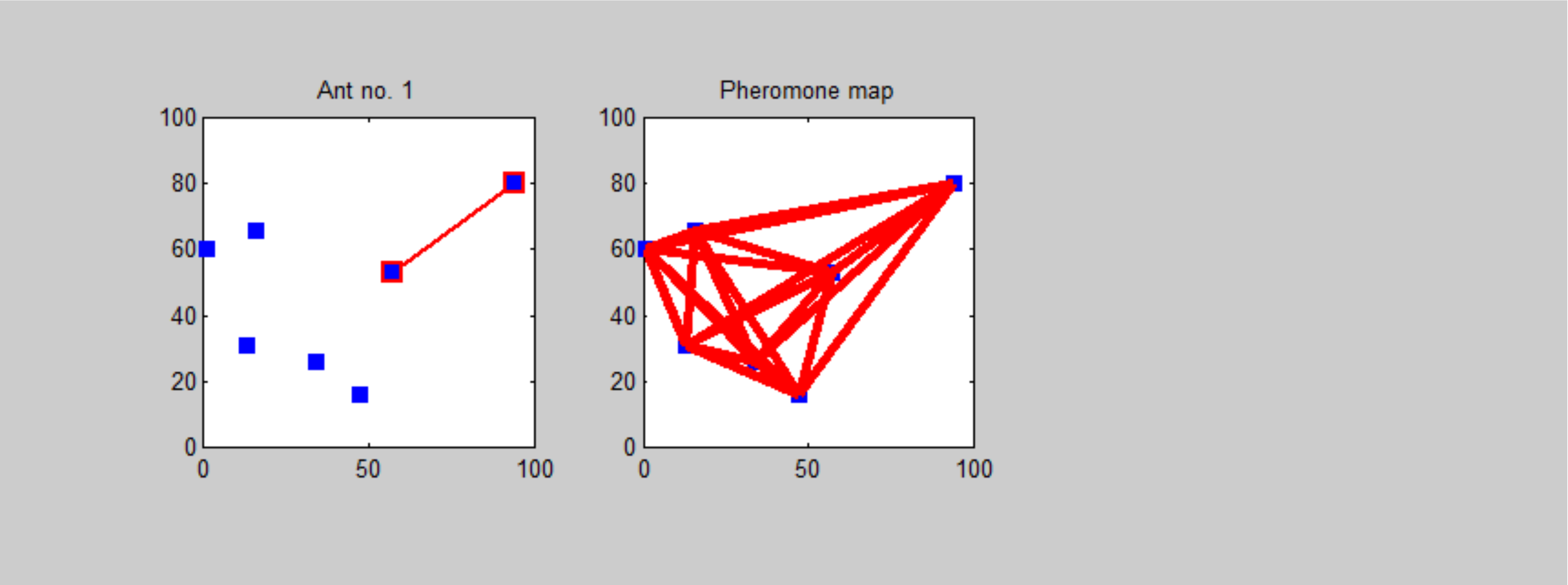
ACO for TSP: Tour de France



1 **2** **3** **4**

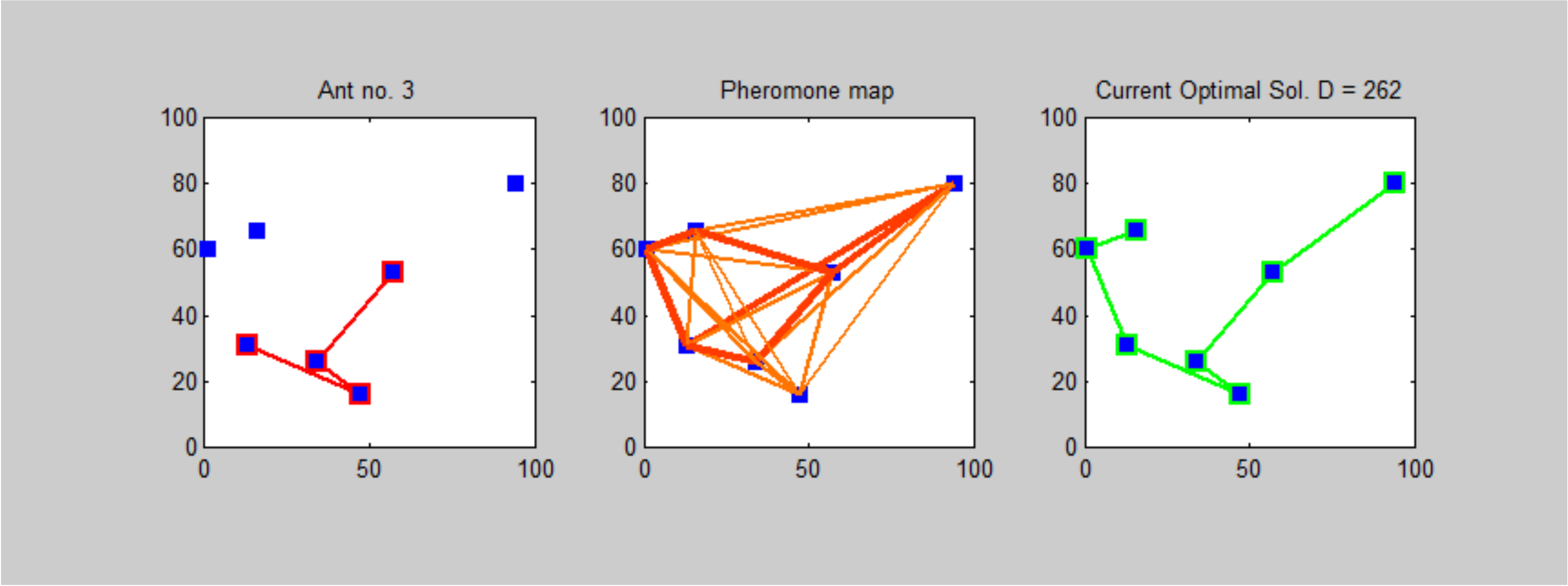
Example Run: ACO for TSP

Example Run: ACO for TSP



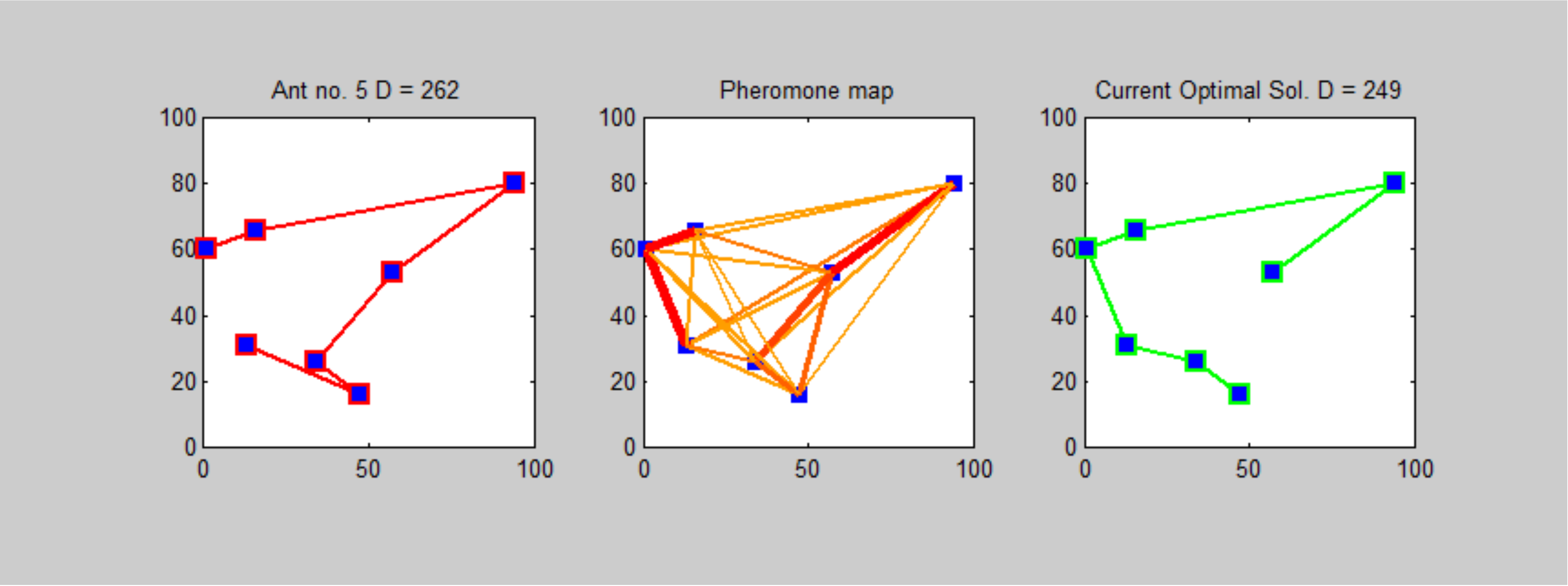
start of run

Example Run: ACO for TSP



middle of run

Example Run: ACO for TSP



end of run

Main Types of ACO

- the three most common types of ACO are:
- Ant System (AS)
  - introduced by Dorigo et al, 1996
- Ant Colony System (ACS)
  - Dorigo et al, 1997
- MAX-MIN ant system (MMAS)
  - Stützle & Hoos, 2000

Ant System (AS)

- the original ACO algorithm
- main characteristic:
  - pheromone values are updated by *all* ants that have completed the tour
- edge selection and pheromone update formulae are those seen in the previous slides

Ant Colony System (ACS)

- a major improvement over Ant System
- ants in ACS use a different edge selection rule:
- the pseudorandom proportional rule:
- the probability for an ant to move from city x to city y depends on:
  - a random variable q, uniformly distributed over $[0,1]$
  - a parameter q_0:
    - if $q \leq q_0$ then the edge that maximises $\tau_{xy} . \eta_{xy}^{\beta}$ is chosen
    - otherwise the same formula as AS is used
- this rule favours exploitation of the pheromone information

ACS: Local Pheromone Update

- the exploitation provided by the edge selection rule is counterbalanced by the local pheromone update:
- *during a tour*, after each edge is traversed an ant updates that edge's pheromone level:

$$\tau_i \leftarrow (1-\rho).\tau_i + \rho.\tau_{init}$$

- where:
  - $\rho \in (0,1]$ is the pheromone decay coefficient
  - τ_{init} is the initial value of the pheromone

ACS: Local Pheromone Update

- the main aim of the local update is to diversify the search performed by subsequent ants during one iteration
- decreasing the pheromone concentration on the edges as they are traversed during a generation encourages subsequent ants to choose other edges
  - and hence to produce different solutions
  - so it's less likely that several ants produce identical solutions during a single generation

ACS: Offline Pheromone Update

- at the end of a generation a pheromone update is performed
- this greatly differs from Ant System
  - in AS all ants that traversed an edge update the pheromone on it
  - however in ACS this offline update is only performed by the best ant of this generation

MAX-MIN Ant System (MMAS)

- MAX-MIN ant system (MMAS) is another improvement, proposed by Stützle and Hoos (2000), over the original ant system idea
- MMAS differs from AS in that:
  - only the best ant adds pheromone trails
  - the minimum and maximum values of the pheromone are explicitly limited (τ_{max} and τ_{min})
    - these values are often experimentally chosen
- in AS and ACS the max and min values are limited implicitly:
  - the value of the limits is a result of the algorithm working
  - rather than a value set explicitly by the algorithm designer

Benefits of ACO

- ACO algorithms can produce near-optimal solutions to the traveling salesman problem
- they have an advantage over simulated annealing and genetic algorithm approaches when the graph can change dynamically
- because the ant colony algorithm can be run continuously and adapt to changes in real time
- this is particularly valuable to network routing and urban transportation systems

When is it worth doing ACO?

- when a solution is easy to find
- but a *good* solution is hard to find
    - (why?)
- when you potentially want to find several good solutions
- and note that often problems that can be represented well in ACO can be easily translated into EC problems, and vice-versa
- so performance comparisons can be made

Reading & References

- slides based on and adapted from:
  - *Ant Colony Optimization* (Riddle, University of Auckland)
  - *Swarm Intelligence* (Mohitz et al)
- recommended reading:
  - "*Swarm Intelligence*", Corne et al, Handbook of Natural Computing, pp 1599-1622, Springer
    - Swarm Computing module in Brightspace (TBD)
- go and play with Boids!
  - javascript (online sim): http://www.harmendeweerd.nl/boids/
  - java code (github): https://github.com/tofti/javafx-boids