HERIOT WATT UNIVERSITY

Any Colony Optimisation (ACO)

Dr. Michael Lones
Room EM.G31
M.Lones@hw.ac.uk

What is ACO?

- Ant colony optimisation is a well known example of a **swarm optimisation algorithm**
  - Invented by Marco Dorigo in 1992
  - It is the oldest, and one of the most successful, swarm optimisation algorithms
  - It is used to find the optimal path through a weighted graph, and is mostly used in combinatorial optimisation
  - It is quite different to an evolutionary algorithm

Combinatorial Optimisation

- In combinatorial optimisation, the aim is to find an optimal ordering of components or actions
  - Where components/actions are selected from a finite set of possible components/actions
  - You've already seen at least one of these: the **travelling salesman problem**, where solutions are a particular ordering of a finite set of destinations
  - Other well known examples include bin packing, vehicle routing, job shop scheduling and nurse rostering

Travelling Salesman Problem

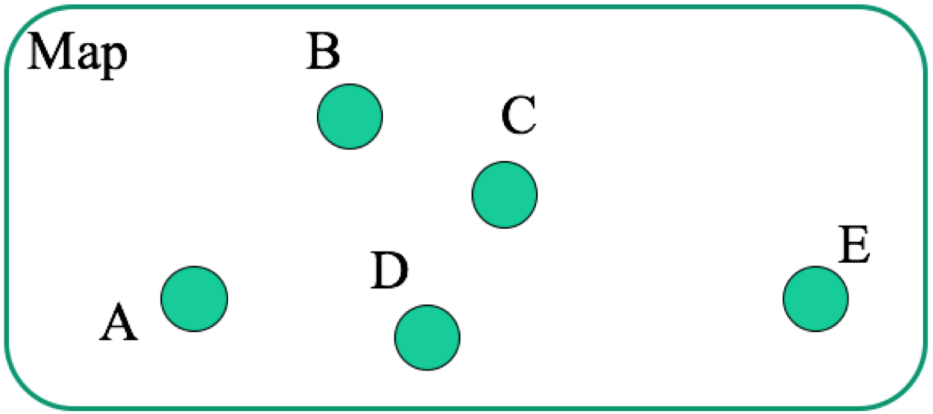
From Introduction to EAs lecture

The TSP

The Travelling Salesperson Problem

You have *N* 'cities', and an associated distance matrix. Find the shortest **tour** through the cities. I.e., starting from one of the cities ('A', for example), what is the quickest way to visit all of the other cities, and end up back at 'A' ?

Many many real problems are based on this (vehicle routing probs, bus or train scheduling), or exactly this (drilling thousands of holes in a metal plate, or … an actual traveling Salesperson problem!).



	A	B	C	D	E
A		5	7	4	15
B	5		3	4	10
C	7	3		2	7
D	4	4	2		9
E	15	10	7	9	

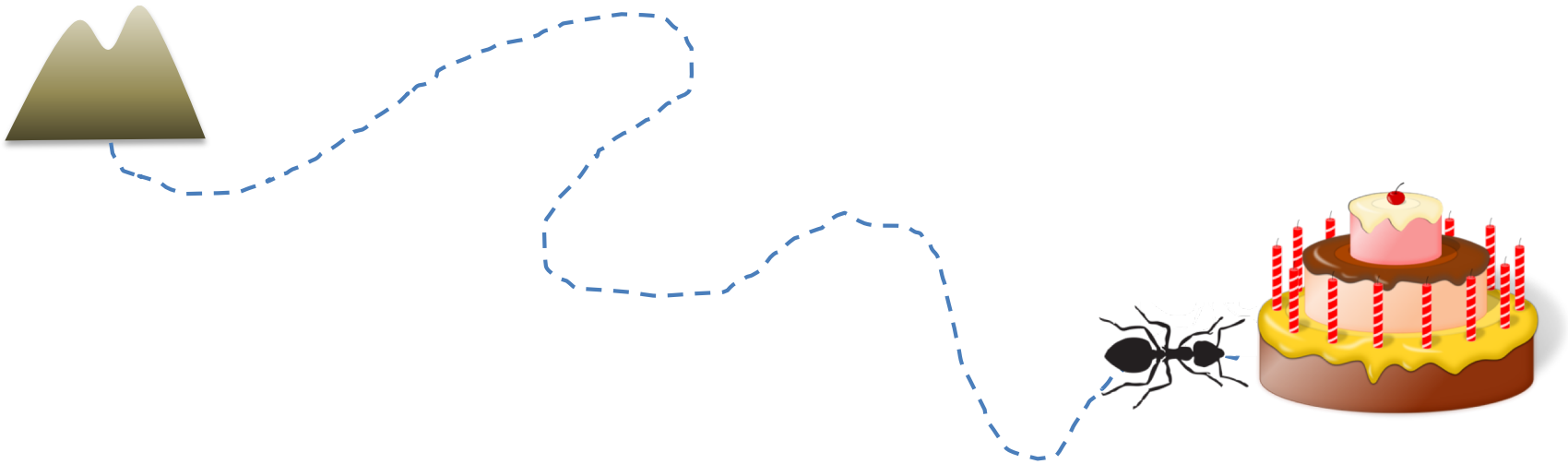
What is ACO?

HERIOT WATT UNIVERSITY

- Ant colony optimisation was inspired by swarm behaviour within ant colonies
  - Particularly by the manner in which they **collectively forage** for food in their environment
  - And the way in which they use **chemical messengers** to share knowledge about the location of food

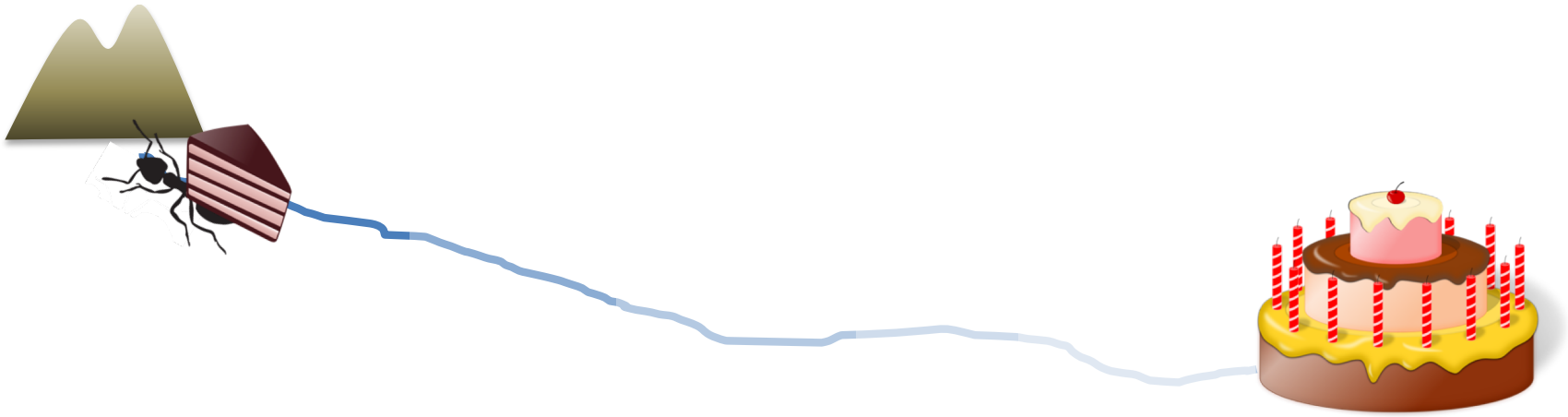
Ant Foraging

- An ant wanders around its environment looking for food



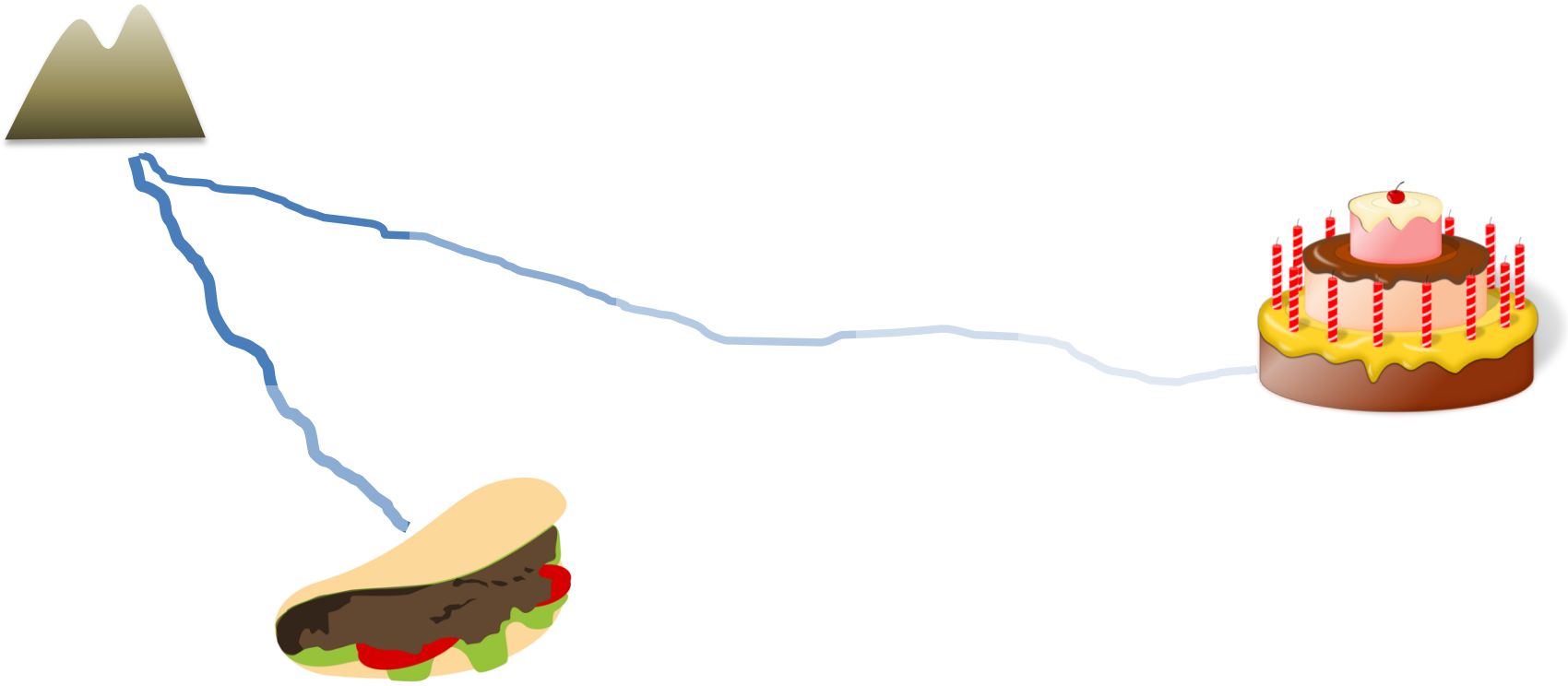
Ant Foraging

- When it heads home, it leaves a pheromone trail
  - Which gradually evaporates over the course of time



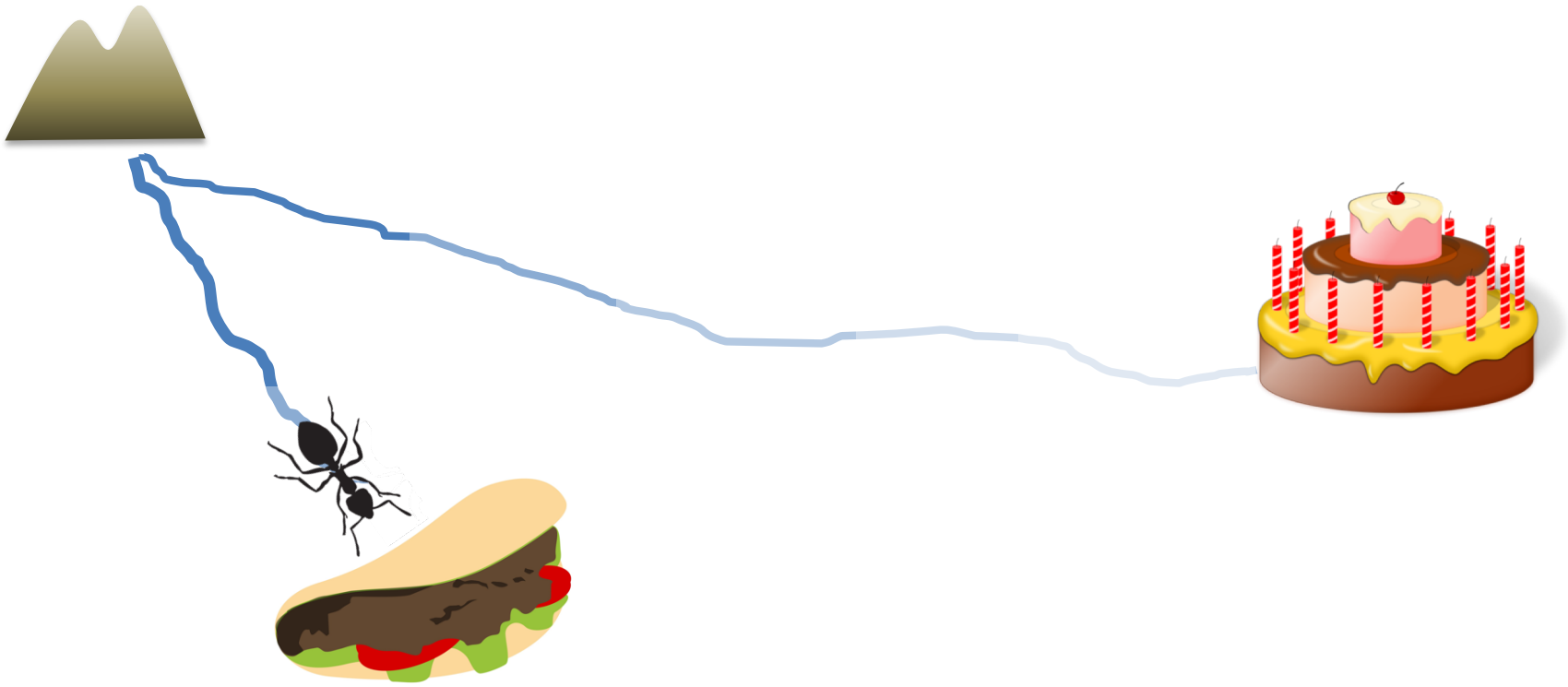
Ant Foraging

- Meanwhile, another ant might have found closer food
  - Because it's closer, the pheromone trail has evaporated less



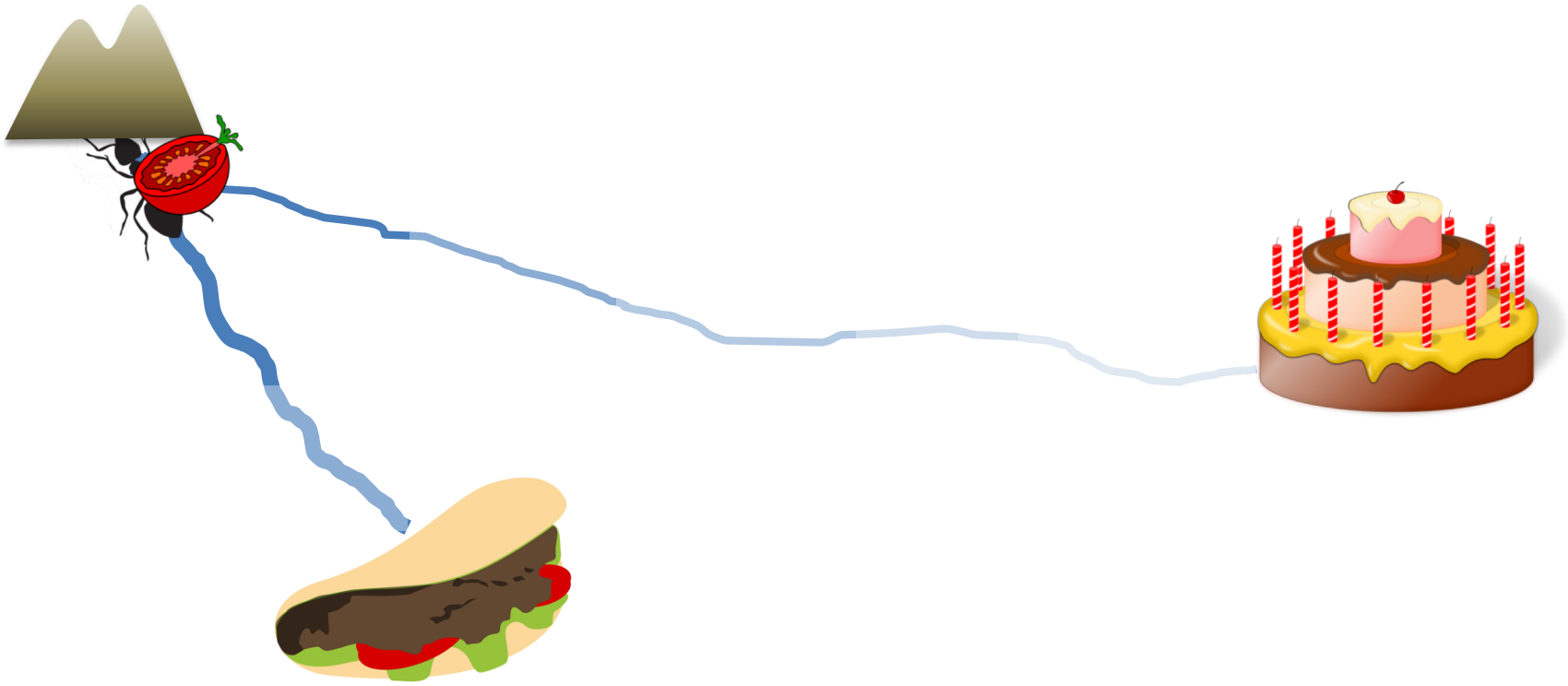
Ant Foraging

- Another ant feels peckish and starts to forage for food
  - It's more likely to find food if it follows the stronger trail



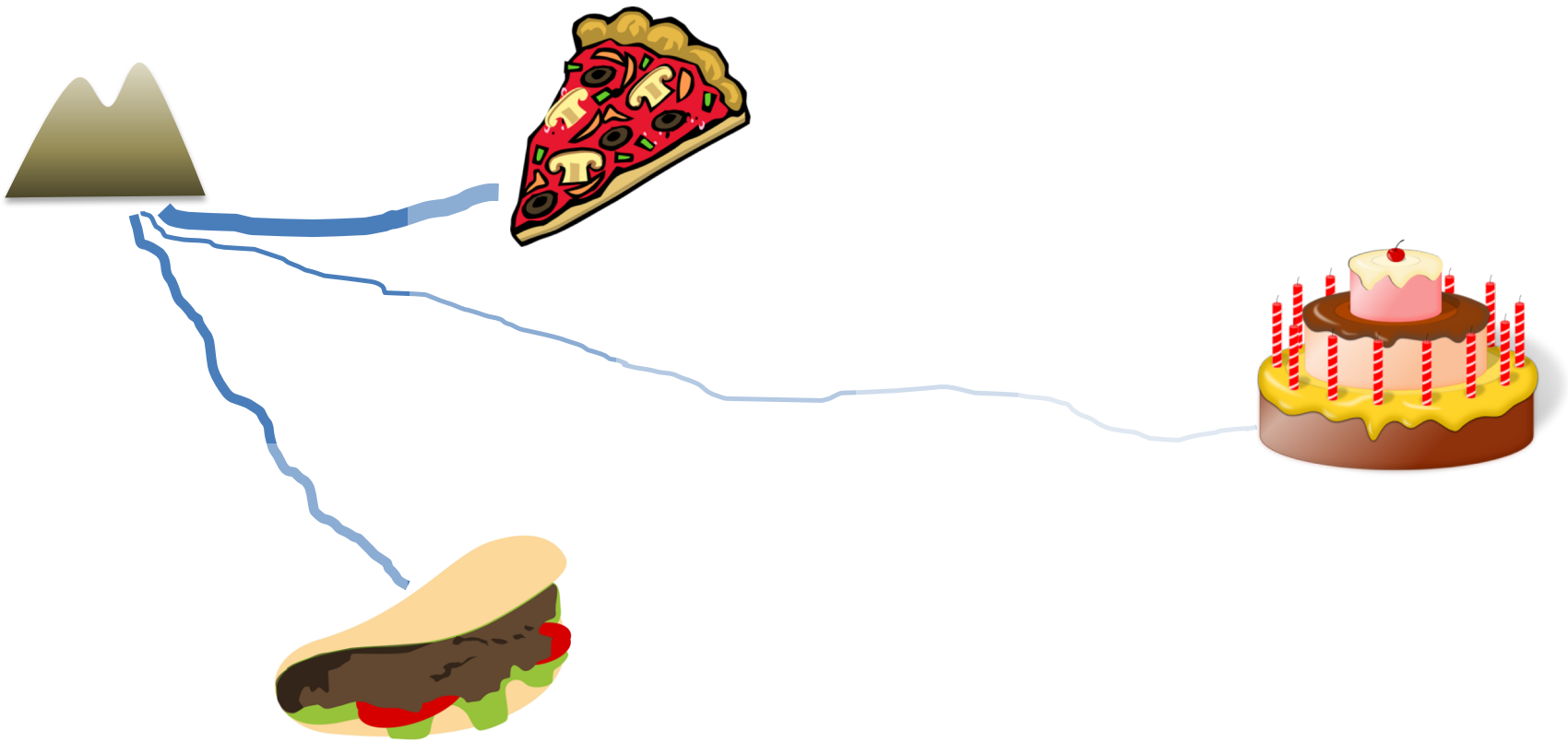
Ant Foraging

- When it returns, it will further strengthen the pheromone trail
  - Which means more ants will follow it, making it even stronger



Ant Foraging

- Over time, the pheromone trails are likely to give a good indication of the best paths to food sources



Stigmergy

- **Stigmergy** is the indirect sharing of information through changes made to an environment
  - Stigmergy underlies the ability of organisms to reach consensus decisions via indirect communication
  - The laying of pheremones by social insects is a prominent example
  - Humans also engage in stigmergy, e.g. signposting, collaborative online authoring (e.g. Wikipedia)

Stigmergy

- Using stigmergy, **a problem is solved bit by bit**
  - Social insects leave markers or messages: these don't solve the problem in themselves, but they affect other individuals in a way that helps them solve the problem
  - This is how ants find shortest paths: bit by bit through the efforts of a series of multiple individuals

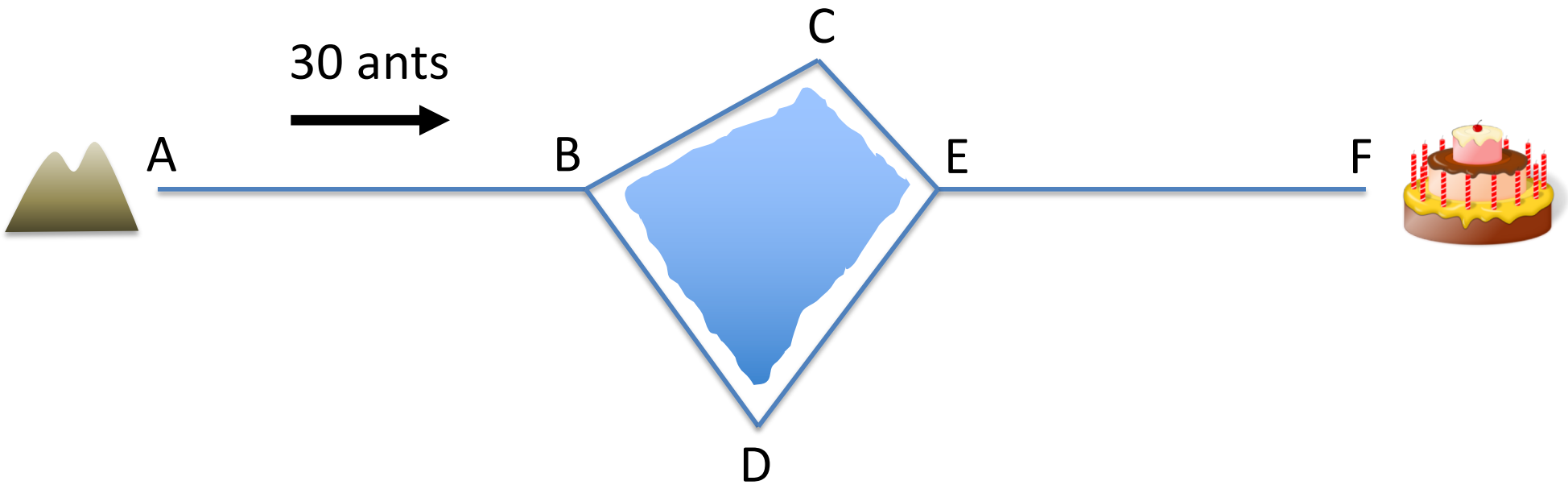
Stigmergy

- If you want to know more about stigmergy:
  - F. Heylighen, Stigmergy as a universal coordination mechanism, Cognitive Systems Research 38:4-13, 2016
  - http://www.sciencedirect.com/science/article/pii/S1389041715000327
  - "… stigmergy enables complex, coordinated activity without any need for planning, control, communication, simultaneous presence, or even mutual awareness."

Any Questions?

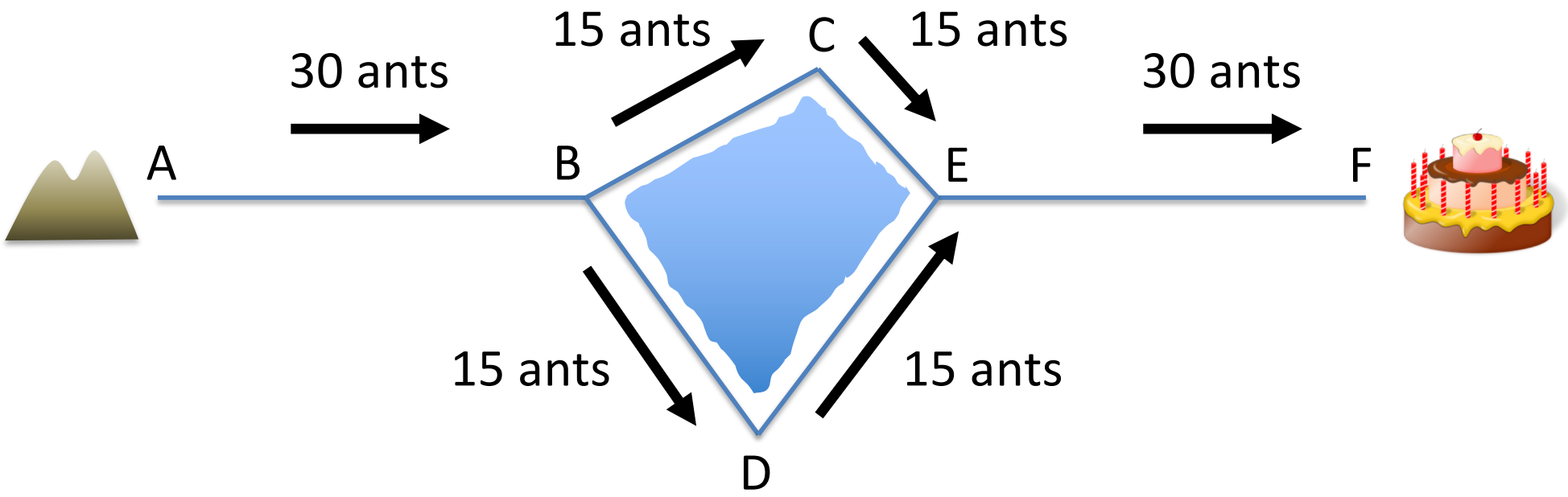
ACO: The basic idea

- A population of 30 artificial ants goes foraging through a graph-structured landscape



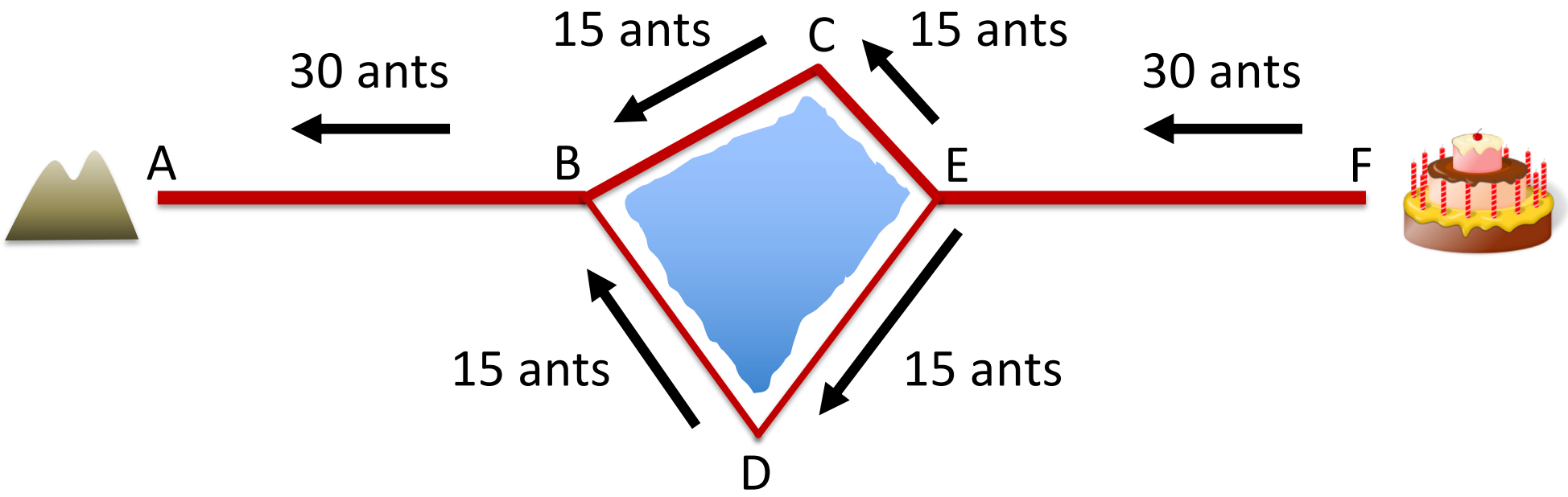
ACO: The basic idea

- They have no knowledge of the shortest route, so they split equally at the first junction



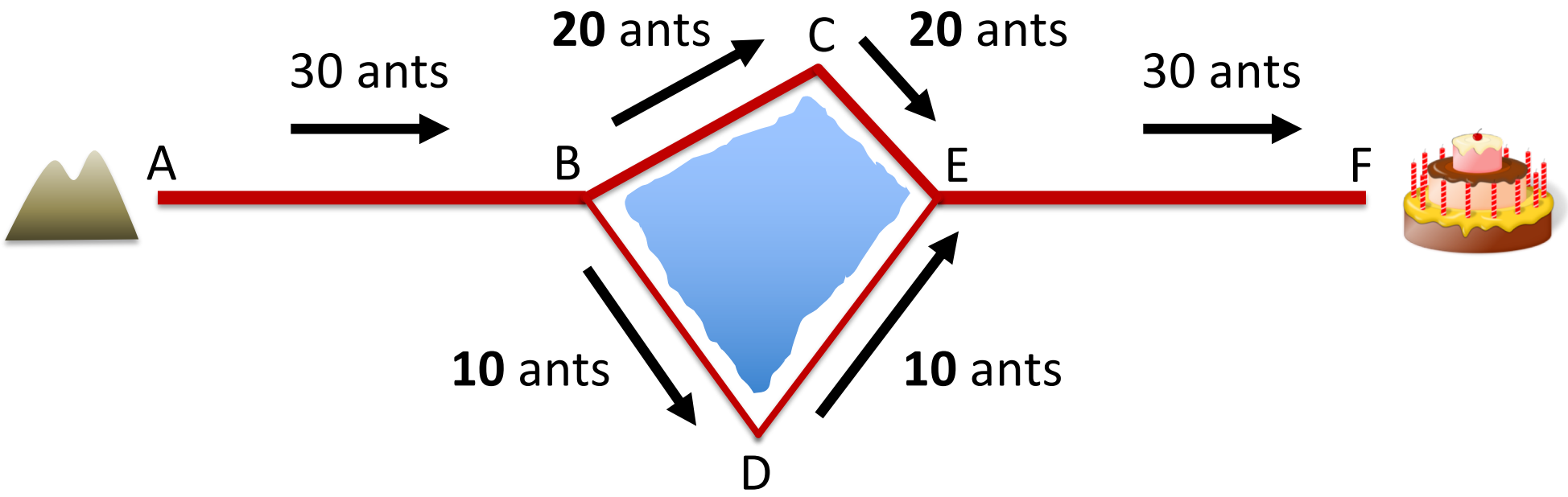
ACO: The basic idea

- On their way back, they deposit pheromone in inverse proportion to their journey length



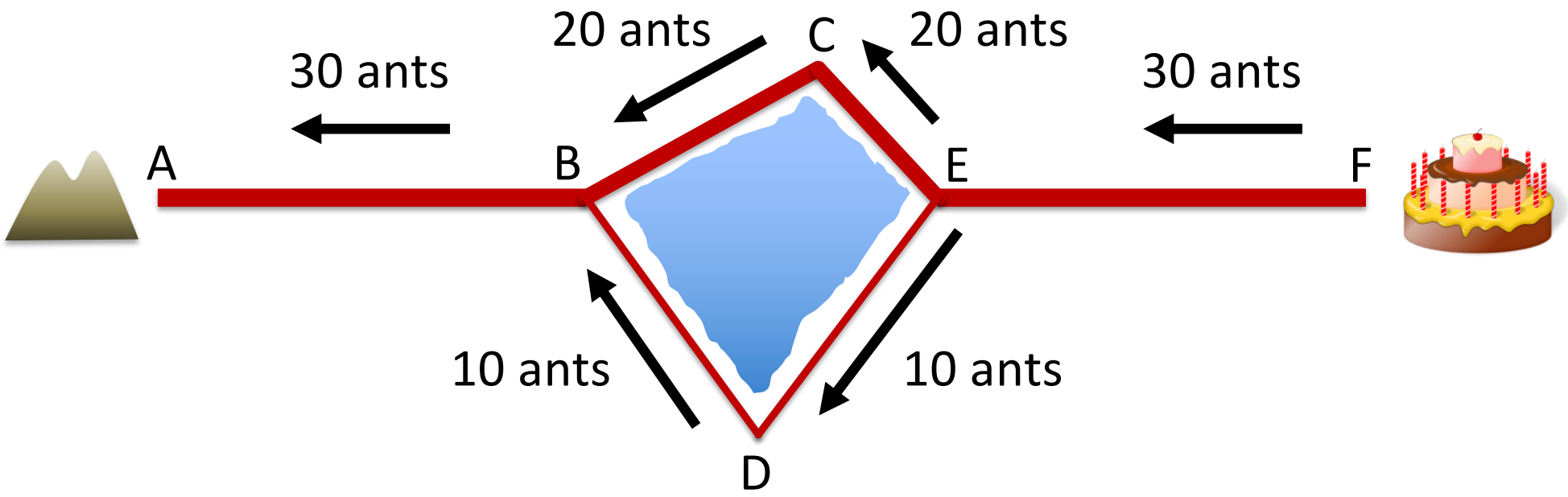
ACO: The basic idea

- On their next foraging trip, more ants follow the trails with higher pheromone concentrations



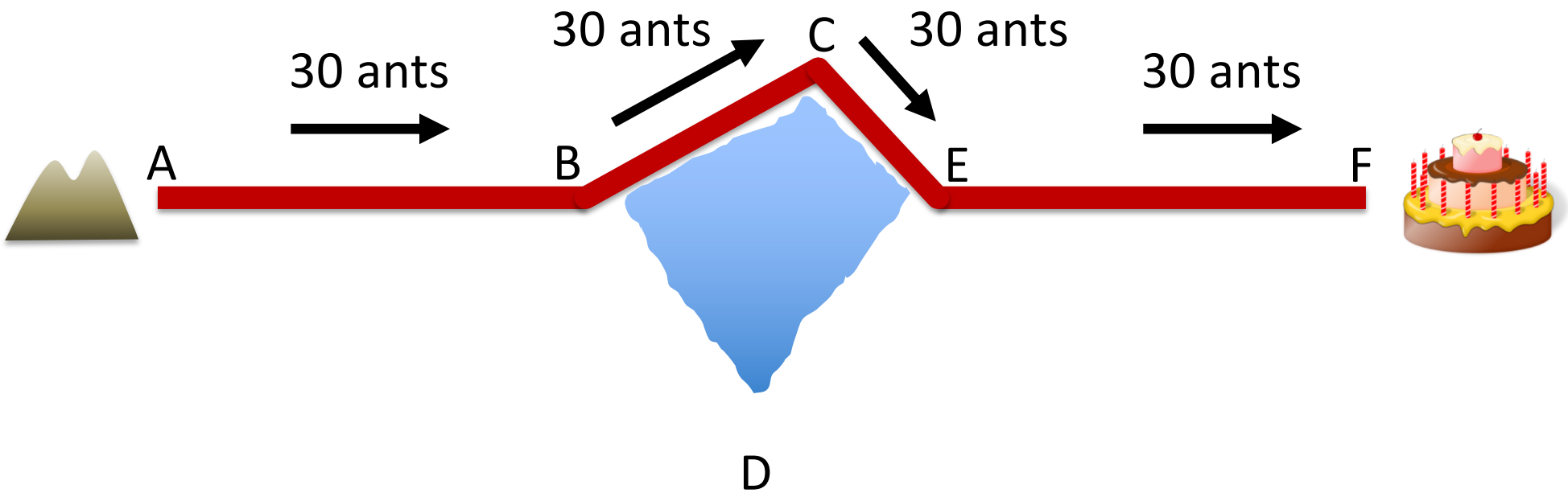
ACO: The basic idea

- On their way back, they increase the concentration of pheromone on the shortest path even more



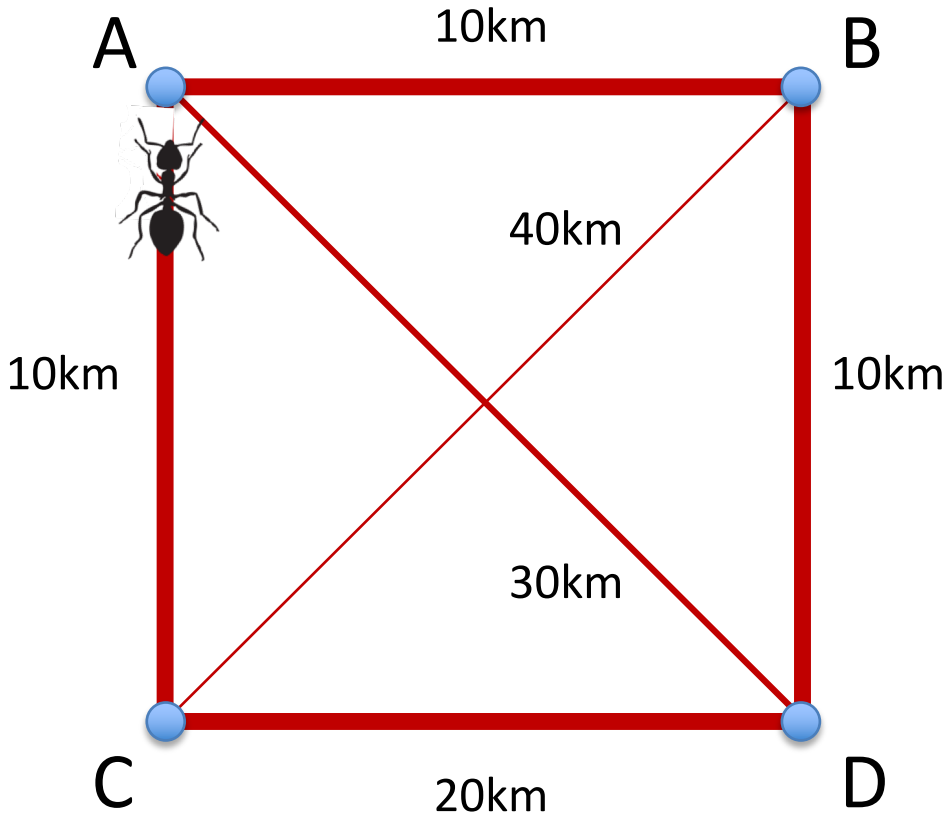
ACO: The basic idea

- Pheromone evaporates over time, so eventually only the shortest path will remain



Solving TSP with ACO

- ACO applies the same idea, with a succession of virtual ants laying virtual pheromones within the state space of a combinatorial optimisation problem:



*Not drawn to scale

Demo

- A demo of the ant colony system solving TSP: https://jtpio.github.io/aco-tsp/

ACO Algorithms

- There are a number of different ACO algorithms, but they all build upon this general idea:
  - A series of search processes (referred to as **ants**) move between nodes in a graph
  - They choose which direction to go in based on pheromone variables associated with each edge
  - Each ant's path (known as an **ant trail**) through the graph represents a candidate solution to a problem
  - When an ant has finished constructing a solution, the values of the pheromone variables on its path are updated according to the quality of the solution

ACO Algorithms

- Milestones in ACO's development:
  - **Ant System** was the original algorithm. It worked, but was not very competitive.
  - **Ant Colony System** added evaporation of pheromones to both diversify search (i.e. encourage new paths) and increase exploitation (i.e. follow the best paths so far). It is widely used in combinatorial optimisation.
  - **Max-Min Ant System** added mechanisms to further increase exploitation whilst avoiding premature convergence. It works well on problems like TSP, whose search spaces favour exploitation.

Ant Colony System

Implementations vary, but this is fairly typical:

- Identify solution components $c_0 \ldots c_n$
- Initialise corresponding pheromones $p_0 \ldots p_n$
- Repeat until optimal solution found
  - for ***i*** in 0...***popsize*** times
    - Build trail t_i through the graph of ***c***s by following an edge at each turn with probability relative to its ***p***
    - Evaluate fitness of trail t_i
  - For the best trail in t_0 to t_i, increase each ***p*** in the trail in proportion to its fitness
  - Evaporate all ***p***s by a small amount
- End repeat

Ant Colony System

Implementations vary, but this is fairly typical:

- Identify solution components $c_0 \ldots c_n$

These are the connections between cities in TSP, e.g. AB, AC, AD, BC, …

- Initialise corresponding pheromones $p_0 \ldots p_n$
- Repeat until optimal solution found
  - for ***i*** in 0...***popsize*** times
    - Build trail t_i through the graph of ***c***s by following an edge at each turn with probability relative to its ***p***
    - Evaluate fitness of trail t_i
  - For the best trail in t_0 to t_i, increase each ***p*** in the trail in proportion to its fitness
  - Evaporate all ***p***s by a small amount
- End repeat

Ant Colony System

Implementations vary, but this is fairly typical:

- Identify solution components $c_0 \ldots c_n$

  These are the connections between cities in TSP, e.g. AB, AC, AD, BC, …

- Initialise corresponding pheromones $p_0 \ldots p_n$

  Either initialised at zero, or initialised randomly

- Repeat until optimal solution found
  - for ***i*** in 0...***popsize*** times
    - Build trail t_i through the graph of ***c***s by following an edge at each turn with probability relative to its ***p***
    - Evaluate fitness of trail t_i
  - For the best trail in t_0 to t_i, increase each ***p*** in the trail in proportion to its fitness
  - Evaporate all ***p***s by a small amount
- End repeat

Ant Colony System

Implementations vary, but this is fairly typical:

- Identify solution components $c_0 \ldots c_n$

  These are the connections between cities in TSP, e.g. AB, AC, AD, BC, …

- Initialise corresponding pheromones $p_0 \ldots p_n$

  Either initialised at zero, or initialised randomly

- Repeat until optimal solution found
  - for ***i*** in 0...***popsize*** times

    popsize is the number of "ants" in the colony
    - Build trail t_i through the graph of ***c***s by following an edge at each turn with probability relative to its ***p***
    - Evaluate fitness of trail t_i
  - For the best trail in t_0 to t_i, increase each ***p*** in the trail in proportion to its fitness
  - Evaporate all ***p***s by a small amount
- End repeat

Ant Colony System

Implementations vary, but this is fairly typical:

- Identify solution components $c_0 \ldots c_n$

  These are the connections between cities in TSP, e.g. AB, AC, AD, BC, …

- Initialise corresponding pheromones $p_0 \ldots p_n$

  Either initialised at zero, or initialised randomly

- Repeat until optimal solution found
  - for ***i*** in 0...***popsize*** times

    popsize is the number of "ants" in the colony

    - Build trail t_i through the graph of ***c***s by following an edge at each turn with probability relative to its ***p***
    - Evaluate fitness of trail t_i

    For TSP, trails should be complete and not contain any cycles

  - For the best trail in t_0 to t_i, increase each ***p*** in the trail in proportion to its fitness
  - Evaporate all ***p***s by a small amount
- End repeat

Ant Colony System

Implementations vary, but this is fairly typical:

- Identify solution components $c_0 \ldots c_n$

  These are the connections between cities in TSP, e.g. AB, AC, AD, BC, …

- Initialise corresponding pheromones $p_0 \ldots p_n$

  Either initialised at zero, or initialised randomly

- Repeat until optimal solution found
  - for ***i*** in 0...***popsize*** times

    popsize is the number of "ants" in the colony

    - Build trail t_i through the graph of ***c***s by following an edge at each turn with probability relative to its ***p***
    - Evaluate fitness of trail t_i

    For TSP, trails should be complete and not contain any cycles

  - For the best trail in t_0 to t_i, increase each ***p*** in the trail in proportion to its fitness

    Meaning that, in the next repeat, ants are more likely to follow the components of this trail

  - Evaporate all ***p***s by a small amount
- End repeat

Ant Colony System

Implementations vary, but this is fairly typical:

- Identify solution components $c_0 \ldots c_n$

  These are the connections between cities in TSP, e.g. AB, AC, AD, BC, …

- Initialise corresponding pheromones $p_0 \ldots p_n$

  Either initialised at zero, or initialised randomly

- Repeat until optimal solution found
  - for ***i*** in 0...***popsize*** times

    popsize is the number of "ants" in the colony

    - Build trail t_i through the graph of ***c***s by following an edge at each turn with probability relative to its ***p***
    - Evaluate fitness of trail t_i

    For TSP, trails should be complete and not contain any cycles

  - For the best trail in t_0 to t_i, increase each ***p*** in the trail in proportion to its fitness

    Meaning that, in the next repeat, ants are more likely to follow the components of this trail

  - Evaporate all ***p***s by a small amount

    Meaning that the algorithm will gradually forget about sub-optimal trails

- End repeat

Further Information

HERIOT WATT UNIVERSITY

- The "Essentials of Metaheuristics" book discusses ACO algorithms and gives pseudocode (see Section 8.3)
- Here's an overview by ACO's creator Marco Dorigo: http://www.scholarpedia.org/article/Ant_colony_optimization#Ant_colony_system
- The max-min ant system paper discusses at length the features needed to get good performance: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=88694D71F76DACF931C6817FB253E88D?doi=10.1.1.127.3897&rep=rep1&type=pdf

Suggested Reading

- Read the book chapter on "Swarm Intelligence" by David Corne, Alan Reynolds and Eric Bonabeau
  - Available from Vision
  - It's quite a long read (31 pages) but is a good overview of material covered today and tomorrow

Things you should know

HERIOT WATT UNIVERSITY

- About swarm intelligence generally
  - The properties of swarm systems
  - Awareness of application areas
  - A basic understanding of the rules of boids
- About ant colony optimisation
  - The general principles behind how it works
  - An idea of how you would apply it to a graph problem
  - *I don't expect you to know exact algorithmic details*

Tomorrow's Lecture

- Particle Swarm Optimisation
  - This is the best known swarm optimisation algorithm