Social Insect Inspired AI Ant Algorithms

CSc355

Alan Dix
dixa@comp.lancs.ac.uk

Manolis Sifalakis
mjs@comp.lancs.ac.uk

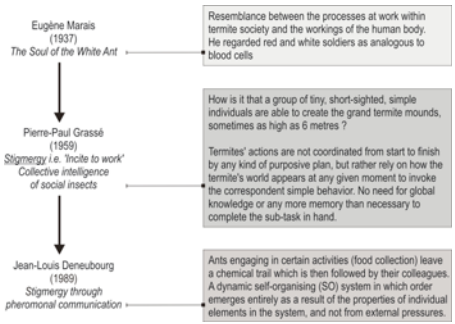
Lecture Overview

- What is Ant Algorithms?
- A brief history – timeline
- Self Organising systems
- Deneubourg's simple experiment
- Computational Model
  - Sample iteration of the algorithm
- Application: The Travelling Salesman Problem
- Other applications: Network Routing
- Reference List

What is Ant Algorithms

- Ant optimisation algorithms [Dorigo 1996] are multi-agent systems, which consist of agents with the collective behavior (stigmergy) of ants for finding shortest paths
  - Alternative to applying complex algorithms to static datasets
  - A set of artificial ants implement a simple algorithm collectively to solve a combinatorial problem by a cooperative effort
  - Originated from Alife research

A Brief History - Timeline



Eugène Marais
(1937)
The Soul of the White Ant

Resemblance between the processes at work within termite society and the workings of the human body. He regarded red and white soldiers as analogous to blood cells

Pierre-Paul Grassé
(1959)
Stigmergy i.e. 'Incite to work'
Collective intelligence
of social insects

How is it that a group of tiny, short-sighted, simple individuals are able to create the grand termite mounds, sometimes as high as 6 metres ?

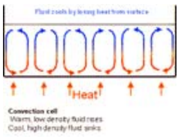
Termites' actions are not coordinated from start to finish by any kind of purposive plan, but rather rely on how the termite's world appears at any given moment to invoke the correspondent simple behavior. No need for global knowledge or any more memory than necessary to complete the sub-task in hand.

Jean-Louis Deneubourg
(1989)
Stigmergy through
pheromonal communication

Ants engaging in certain activities (food collection) leave a chemical trail which is then followed by their colleagues. A dynamic self-organising (SO) system in which order emerges entirely as a result of the properties of individual elements in the system, and not from external pressures.

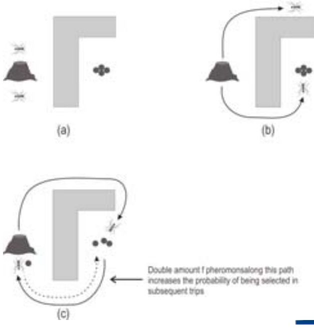
Self Organising systems: *Bénard cellular convection*

- Layer of fluid heated from below
- In right conditions, perfect temperature gradient is formed vertically across the fluid
- System becomes 'top heavy', warmer (light) molecules at the bottom rise to the top where they cool (heavy) and uniformly flow to the bottom again
- The liquid does not simply bubble away without a pattern or organisation
  ... but instead ...
  an ordered system is formed with millions of molecules self-organising in a hexagonal pattern
- Most efficient convection for energy-dissipation.



Convection cell
Warm, low density fluid rises.
Cool, high density fluid sinks.

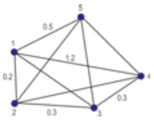
Deneubourg's Simple Experiment



Double amount f pheromonsalong this path increases the probability of being selected in subsequent trips

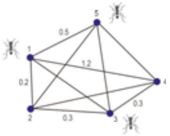
Terminology Index

- Node: a vertice
- Edge: a line connecting 2 vertices
- Graph: Diagram connecting nodes with edges
- Weighted graph: Edges have weights
- Path: Sequence of nodes-edges from between two nodes
- Hamiltonian path: A path without any node revisited
- Sigma: a sum of terms
- Delta: a difference between terms



Computational Model

- The environment
  - Weighted graph representing distances between nodes
- The ants
  - Init population randomly distributed
  - Travel across the graph
  - Ordered tabu list of visited nodes
  - Follow a Hamiltonian path



$$P = \frac{\tau(r,u)^{\alpha} \cdot \eta(r,u)^{\beta}}{\sum_{k} \tau(r,u)^{\alpha} \cdot \eta(r,u)^{\beta}}$$

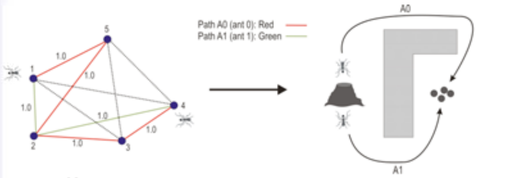
Computational Model

- Ant Tour
  - Hamiltonian path across all nodes in graph
  - Path length L computed based on distances η
  - Pheromone left on each node in the path
    - Short tour → High pheromone
    - Long Tour → Low Pheromone

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L^{k}(t)}$$

  - Pheromone increase at the end of tour $\tau_{ij}(t) = \tau_{ij}(t) + (\Delta\tau_{ij}^{k}(t) \cdot \rho)$
- Evaporation
  - Remove edges of poor paths $\tau_{ij}(t) = \tau_{ij}(t) \cdot (1 - \rho)$

Sample Iteration



$\rho = 0.6$
$\alpha = 3.0$
$\beta = 1.0$
$Q = 2.0$
$\eta = 1/\text{dist}$

	A0	A1
Dist.(steps)	2	4
Q/Tour	1.0	0.5
η	0.50	0.25

$\tau_{incr\,A1} = 0.1 + (0.5 \cdot 0.6) = 0.4$
$\tau_{evap\,A1} = 0.4 \cdot (1.0 - 0.6) = 0.16$

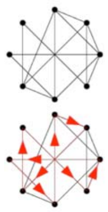
$$P_{A1} = \frac{(0.28)^{3} \cdot (0.5)^{1}}{(0.16)^{3} \cdot (0.25)^{1} + (0.28)^{3} \cdot (0.5)^{1}} = 0.915 = 91.5\%$$

$\tau_{incr\,A0} = 0.1 + (1.0 \cdot 0.6) = 0.7$
$\tau_{evap\,A0} = 0.7 \cdot (1.0 - 0.6) = 0.28$

$$P_{A0} = \frac{(0.16)^{3} \cdot (0.25)^{1}}{(0.16)^{3} \cdot (0.25)^{1} + (0.28)^{3} \cdot (0.5)^{1}} = 0.085 = 8.5\%$$

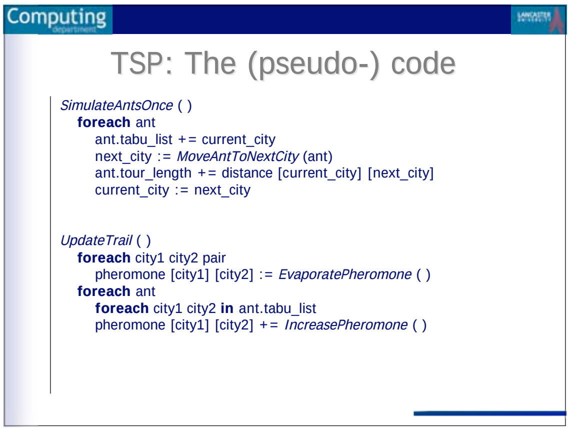
Application: Travelling Salesman Problem

- A set of cities. A salesman needs to travel through all the cities following an optimal route (Hamiltonian path) that minimises the distance travelled.
  - Fist studied in 1930s
  - NP-hard: Not been found an algorithm that solves the general problem (for any number of cities at any arrangement), in polynomial time.
  - (Sub-)optimal solutions are possible for specific instances of the problem



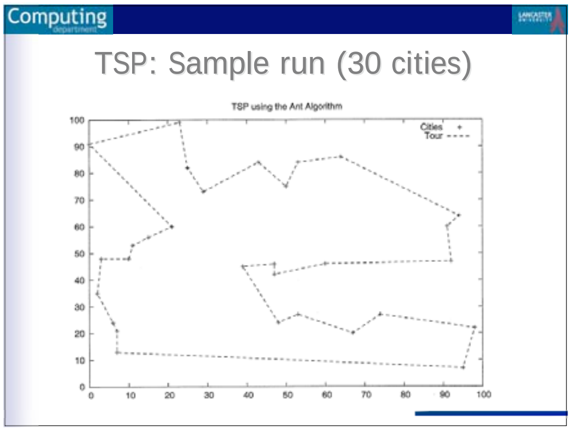
TSP: The (pseudo-) code

```
Main ( )
    Initialise ( )
    while time < MAX_TIME
        SimulateAntsOnce ( )
        UpdateTrail ( )
        time++

Initialise ( )
    while cities < MAX_CITIES
        new_city := CreateCity ( )
        RandomlyPlaceCity (new_city)
        foreach city
            distance [new_city] [city] := 0
            pheromone [new_city] [city] := 0
    foreach city1 city2 pair
        distance [city1] [city2] := CalculateDistance (city_pair)
    while ant < MAX_ANTS
        PlaceInCity (ant)
```

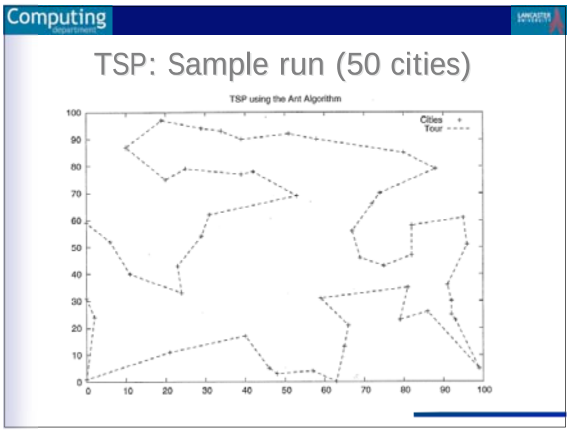


TSP: The (pseudo-) code

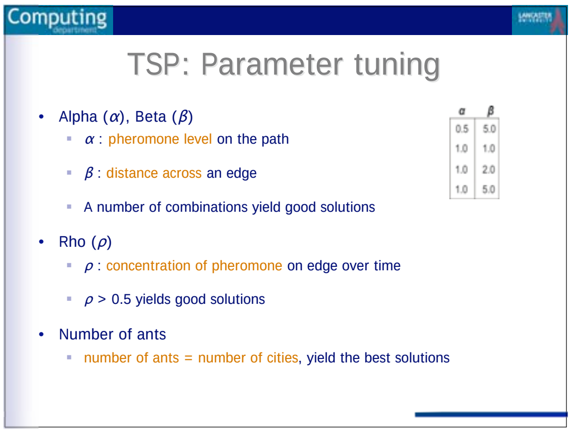
```
*SimulateAntsOnce* ( )
    **foreach** ant
        ant.tabu_list += current_city
        next_city := *MoveAntToNextCity* (ant)
        ant.tour_length += distance [current_city] [next_city]
        current_city := next_city

*UpdateTrail* ( )
    **foreach** city1 city2 pair
        pheromone [city1] [city2] := *EvaporatePheromone* ( )
    **foreach** ant
        **foreach** city1 city2 **in** ant.tabu_list
        pheromone [city1] [city2] += *IncreasePheromone* ( )
```



TSP: Sample run (30 cities)



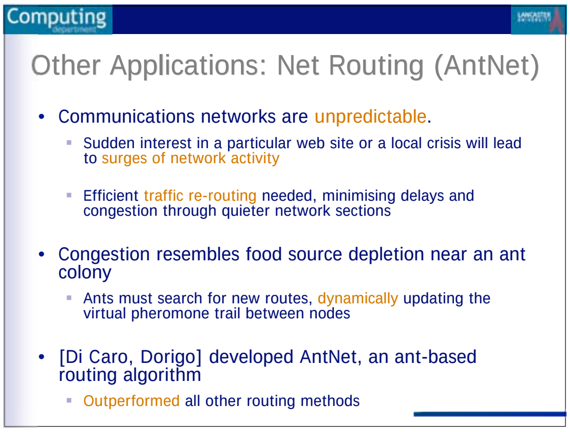
TSP: Sample run (50 cities)



TSP: Parameter tuning

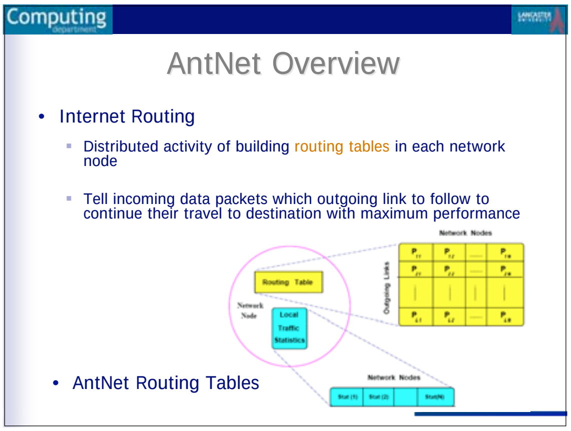
α	β
0.5	5.0
1.0	1.0
1.0	2.0
1.0	5.0

- Alpha (α), Beta (β)
  - α : pheromone level on the path
  - β : distance across an edge
  - A number of combinations yield good solutions
- Rho (ρ)
  - ρ : concentration of pheromone on edge over time
  - ρ > 0.5 yields good solutions
- Number of ants
  - number of ants = number of cities, yield the best solutions



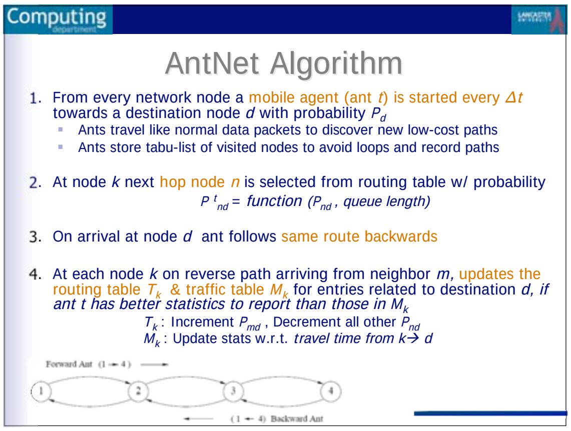
Other Applications: Net Routing (AntNet)

- Communications networks are unpredictable.
  - Sudden interest in a particular web site or a local crisis will lead to surges of network activity
  - Efficient traffic re-routing needed, minimising delays and congestion through quieter network sections
- Congestion resembles food source depletion near an ant colony
  - Ants must search for new routes, dynamically updating the virtual pheromone trail between nodes
- [Di Caro, Dorigo] developed AntNet, an ant-based routing algorithm
  - Outperformed all other routing methods



AntNet Overview

- Internet Routing
  - Distributed activity of building routing tables in each network node
  - Tell incoming data packets which outgoing link to follow to continue their travel to destination with maximum performance
- AntNet Routing Tables



AntNet Algorithm

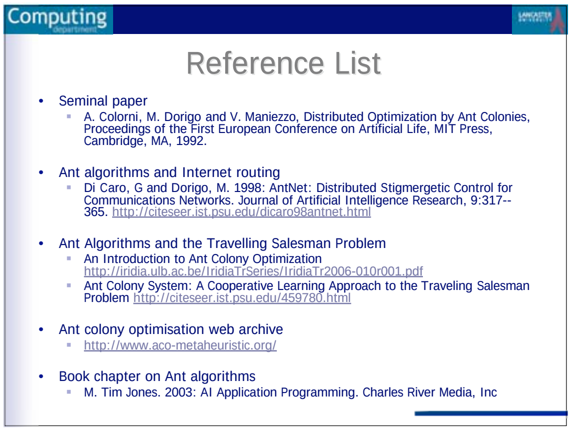
1. From every network node a mobile agent (ant *t*) is started every Δt towards a destination node *d* with probability P_d
   - Ants travel like normal data packets to discover new low-cost paths
   - Ants store tabu-list of visited nodes to avoid loops and record paths
2. At node *k* next hop node *n* is selected from routing table w/ probability

   $P^{t}_{nd} = \textit{function}\ (P_{nd}\ ,\ \textit{queue length})$
3. On arrival at node *d* ant follows same route backwards
4. At each node *k* on reverse path arriving from neighbor *m,* updates the routing table T_k & traffic table M_k for entries related to destination *d, if ant t has better statistics to report than those in* M_k

   T_k : Increment P_{md} , Decrement all other P_{nd}

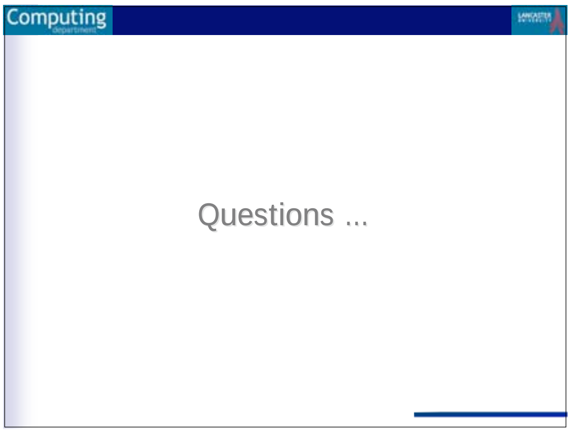
   M_k : Update stats w.r.t. *travel time from k→ d*

Forward Ant (1 → 4)
1
2
3
4
(1 ← 4) Backward Ant



Reference List

- Seminal paper
  - A. Colorni, M. Dorigo and V. Maniezzo, Distributed Optimization by Ant Colonies, Proceedings of the First European Conference on Artificial Life, MIT Press, Cambridge, MA, 1992.
- Ant algorithms and Internet routing
  - Di Caro, G and Dorigo, M. 1998: AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research, 9:317--365. http://citeseer.ist.psu.edu/dicaro98antnet.html
- Ant Algorithms and the Travelling Salesman Problem
  - An Introduction to Ant Colony Optimization http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2006-010r001.pdf
  - Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem http://citeseer.ist.psu.edu/459780.html
- Ant colony optimisation web archive
  - http://www.aco-metaheuristic.org/
- Book chapter on Ant algorithms
  - M. Tim Jones. 2003: AI Application Programming. Charles River Media, Inc



Questions ...