Swarm Intelligence – W3: Multi-Agent Systems based on Ant Trail Laying/Following Mechanisms: Algorithms and Applications
Outline

• The Traveling Salesman Problem (TSP)
• An Ant System (AS) for the TSP
• Moving beyond the original AS
  – Ant Colony System (ACS)
  – ACS with local search for TSP: ACS-3-Opt
• Trail laying and following mechanisms applied to network routing
  – ABC
  – AntNet
AS and ACS and their application to TSP
The Traveling Salesman Problem
Graph \((N,E)\)

\(N\) : set of cities (nodes)
\(E\) : set of connecting roads (links)
\(d_{ij}\) : distance between city \(i\) and \(j\)

**Problem:** Find the shortest path which allow the salesman to visit once and only once each city in the graph

**Difficulty:** NP-hard problem; time for computing the shortest route grows in a nonpolinomial way with the number of cities in the network -> metaheuristics/machine-learning class (e.g., ACO, GA) provide near-optimal solutions!
How Hard are NP-Hard Problems?

TSP – Brute force

- A 30 city tour would have to measure the total distance of be $2.65 \times 10^{32}$ different tours. Assuming a trillion additions per second, this would take $252,333,390,232,297$ years.
- Adding one more city would cause the time to increase by a factor of 31.

QAP – exact algorithms (e.g. Bixius & Anstreicher 2001)

- around 30+ max instances
- ex. 36 nodes (wiring application): 180h CPU on a 800 MHz Pentium III PC
- Same problem with ACO: 10 s on the same machine
Artificial Ants and the Shortest Path Problem

Pheromone trail depositing

Source

Destination

Probabilistic rule to choose the path

?
Problem!

The extension of the real ant behavior (forward/backward trail deposit and slow pheromone decay rate) to artificial ants moving on a graph doesn’t work:

**problem of self-reinforcing loops**

Probabilistic solution generation plus pheromone update

-> self-reinforcing loops

Example of possible self-reinforcing loop
Solution!

Probabilistic rule to choose the path

Pheromone trail depositing

Source

Destination

Memory

Probabilistic rule to choose the path

?
The First ACO Algorithm: The Ant System (AS)
Design Choices for AS
(Dorigo, Colorni, Maniezzo, 1991)

- Ants are given a memory of visited nodes
- Ants build solutions probabilistically without updating pheromone trails (forwards ants)
- Ants deterministically backward retrace the forward path to update pheromone (backwards ants)
- Ants deposit a quantity of pheromone function of the quality of the solution they generated
- Pheromones evaporates much more quickly than in nature
$b_i(t), \ (i = 1 \ldots n) : \text{number of ants at the node } i \text{ at the iteration } t$

$m = \sum_{i=1}^{n} b_i(t) = \text{constant: total number of ants}$
AS for TSP- Individual Ant Behavior

Memory of ant k: list of visited nodes $J_i^k$

The inverted value of the distance $\eta_{ij} = 1/d_{ij}$ between nodes i and j is called visibility; this information (heuristic desirability) is static, i.e. not changed during the problem solution.
AS for TSP- Individual Ant Behavior

\[ \tau_{ij}, \text{quantity of virtual pheromone deposited on the link between the node } i \text{ and } j \]
Assumptions on TSP

• **Usual assumption**: fully connected graph (i.e. there is a direct route with a given distance from any city in the problem to any other); city list work ok

• **Real problem**: not fully connected; problem with city list

• **Possible solutions**:
  – Assume virtual routes so that fully connected; give very bad scores to ants choosing virtual routes (e.g., high but not infinite virtual distance; Dorigo suggestion)
  – Alternative: break not valid tours asap and either relaunch a new ant or consider less ants for updating pheromones at the next iteration (Martinoli suggestion); computationally more efficient but risk to lose constructive aspect of trail laying/following; does not work for dead end edges with end criterion being at the start city
  – Graph connectivity: full – dense – sparse; probably different solutions work better as a function of the connectivity degree; interesting problem
AS for TSP - Algorithm

**Loop** \(t = 1\)


different ant on each node \(t\) there are \(n = |N|\) nodes \(t\)

**For** \(k := 1\) to \(m\) \(t\) each ant builds a tour, in this case \(m = n\) \(t\)

**For** \(step := 1\) to \(n\) \(t\) each ant adds a node to its path \(t\)

Choose the next node to move by applying a probabilistic *state transition rule*

**End-for**

**End-for**

**Update pheromone trails**

**Until** End_condition \(\text{e.g., } t = t_{\text{max}}\)
During a tour $T$, an ant $k$ at the node $i$ decided to move towards the node $j$ with the following probability (idea: roulette wheel):

$$
p_{ij}^k(t) = \begin{cases} 0 & \text{if node } j \text{ have been visited by ant } k \text{ already because of tabu list} \\ \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in J_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} & \text{if the node have not been visited yet} \end{cases}
$$

$\alpha$: parameter controlling the influence of the virtual pheromone
$\beta$: parameter controlling the influence of the local heuristic (visibility)

$J_i^k$: list of nodes still to be visited for ant $k$ when it is at node $i$; starting from an exhaustive list of all the cities in the problem, nodes get scratched during a tour $T$; at the beginning the list contains all nodes but $i$; also called **tabu list**

$T$: tour, it last $n = |N|$ steps ($N =$ number of nodes in the problem) in which the probabilistic transition rule below is applied

$t$: iteration index: number of times the whole algorithm is run; $1 \leq t \leq t_{\text{max}}$
AS for TSP – Virtual Pheromone Update

At the end of each tour $T$, each ant $k$ deposits a quantity of virtual pheromone $\Delta \tau_{ij}^k$ on the link $(i,j)$; pheromones sum up

$$\Delta \tau_{ij}^k = 0,$$  when $(i,j)$ has not been used during the tour $T$

$$\Delta \tau_{ij}^k = \frac{Q}{L^k(t)},$$  when $(i,j)$ has been used during the tour $T$

$L^k(t) =$ length of the tour $T$ done by ant $k$ at iteration $t$

$Q =$ parameter (adjusted by heuristic, not sensitive)

Note: the longer the tour, the lower is the quality of the solution, the smaller the quantity of pheromone dropped
AS for TSP – Virtual Pheromone Update

\[
\tau_{ij}(t + 1) \leftarrow (1 - \rho)\tau_{ij}(t) + \Delta\tau_{ij}(t)
\]

with \( \Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \)

\( \rho = \) evaporation coefficient

At iteration \( t = 0 \) each link is initialized with a small homogenous pheromone quantity \( \tau_{0} \)
AS for TSP – Virtual Pheromone Update with Elitism (EAS)

\[
\tau_{ij}(t + 1) \leftarrow (1 - \rho)\tau_{ij}(t) + \Delta \tau_{ij}(t) + e\Delta \tau_{ij}^e(t)
\]

with \(\Delta \tau_{ij}(t) = \sum_{k=1}^{m} \Delta \tau_{ij}^k\)

\[
\Delta \tau_{ij}^e(t) = \frac{Q}{L^+} \quad \text{if (i,j) belongs to the best tour } T^+
\]

\[
\Delta \tau_{ij}^e(t) = 0 \quad \text{otherwise}
\]

\(e = \text{parameter (adjusted by heuristic, not sensitive)}\)

**Note:** idea, best tours get extra reinforcement
AS for TSP – Evolution of the Best Tour Length

Example: 30 nodes problem
## AS for TSP – Performance as a Function of the Problem Dimension

<table>
<thead>
<tr>
<th>Network</th>
<th>( n ) (dimension)</th>
<th>best solution</th>
<th>Mean number of iterations for the near-optimal solution</th>
<th>Simulation time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 X 4</td>
<td>16</td>
<td>160</td>
<td>5,6</td>
<td>8</td>
</tr>
<tr>
<td>5 X 5</td>
<td>25</td>
<td>254,1</td>
<td>13,6</td>
<td>75</td>
</tr>
<tr>
<td>6 X 6</td>
<td>36</td>
<td>360</td>
<td>60</td>
<td>1020</td>
</tr>
<tr>
<td>7 X 7</td>
<td>49</td>
<td>494,1</td>
<td>320</td>
<td>13440</td>
</tr>
<tr>
<td>8 X 8</td>
<td>64</td>
<td>640</td>
<td>970</td>
<td>97000</td>
</tr>
</tbody>
</table>
Summary of AS

- **Ants** are launched at each iteration from each node to explore the network.
- **Ants** build their paths probabilistically with a probability function of:
  1. *artificial pheromone values*, and
  2. *heuristic values* (in TSP: city visibility)
- **Ants** memorize visited nodes.
- Once reached their destination nodes (in TSP the last city on their list) **ants** retrace their paths backwards, and update the pheromone trails.
Artificial vs. Natural Ant Systems

Individual capabilities

- **Memory**
  - **A-ants**: node list
  - **N-ants**: no explicit memory mechanism in the model but reactive recognition of nest/source differences

- **Navigation**
  - **A-ants**: next node distance available (a priori map or long range sensor assumed)
  - **N-ants**: next node distance only measured a posteriori, when reached the next node (not introduced in the model); only short range sensors

- **Embodiment**
  - **A-ants**: no (physical interference not possible)
  - **N-ants**: yes (physical interference possible)
Artificial vs. Natural Ant Systems

Coordination capabilities

• Trail laying/following mechanisms (pheromone-based)
  – A-ants: centralized, synchronous (at the end of each tour/iteration of the algorithm), serial (ant 1 to k) coordination
  – N-ants: distributed, asynchronous, fully parallel system; no additional coordination for pheromones laying/following/evaporation

• Other information sharing mechanisms
  – A-ants: centralized, synchronous, ID-based (index k of an ant) information sharing among ants (e.g., elitist ants, best tour)
  – N-ants: distributed, asynchronous, and anonymous; role of peer-to-peer communication not well understood; nest might play the role of primitive shared blackboard for chemical signals (e.g., overall level of pheromone)
Extending the Ant System Algorithm
Constructive Heuristic and Local Search

Current wisdom says that a very good strategy for the approximate solution of NP-hard combinatorial optimization problems is the coupling of:

– **a constructive heuristic** (i.e. generate solutions from scratch by iteratively adding solution components)

– **local search** (i.e., start from some initial solution and repeatedly tries to improve by local changes)

This two methods are highly complementary. The problem is to find good couplings: ACO appear (as shown by experimental evidence) to provide such a good coupling.
2 Extensions of AS

• **Ant Colony System (ACS)** — improved constructive heuristic
  
  (Gambardella & Dorigo, 1996; Dorigo & Gambardella, 1997)
  
  – Different transition rule
  
  – Different pheromone trail updating rules: global and local
  
  – Use of a candidate list for the choice of the next city

• **ACS-3-opt** — constructive heuristic + local search
  
  (Gambardella & Dorigo, 1996; Dorigo & Gambardella, 1997)
  
  – Standard ACS + local search
  
  – In case of TSP problems, 2-opt (2 edges exchanged), 3-opt (3 edges exchanged), and Lin-Kernighan (variable number of edges exchanged) are used as local search algorithms
Ant Colony System

Loop

Place one ant on each node /*there are \( n \) nodes */

For \( k := 1 \) to \( m \) /* each ant builds a solution, in this case \( m=n \)*/

For \( \text{step} := 1 \) to \( n \) /* each ant adds a node to its path */

Choose the next city to move by applying a probabilistic solution construction rule

End-for

End-for

Update pheromone trails

Until End_condition
Different Transition Rule

An ant $k$ on city $i$ chooses the city $j$ to move according to the following rule:

$$j = \begin{cases} \arg \max_{u \in J_i^k} \{[\tau_{iu}(t)][\eta_{iu}]^\beta\} & \text{if} \hspace{1em} q \leq q_0 \\ J & \text{if} \hspace{1em} q > q_0 \end{cases}$$

With $J \in J_i^k$ being a city that is randomly selected according to:

$$p_{iJ}^k(t) = \frac{[\tau_{iJ}(t)][\eta_{iJ}]^\beta}{\sum_{l \in J_i^k}[\tau_{il}(t)][\eta_{il}]^\beta}$$

- $q$: uniform distributed random variable $[0,1]$
- $q_0$: parameter between 0 and 1, controls exploration/exploitation ($q_0 = 0$ as AS)
- $q \leq q_0$: deterministic rule, exploitation of the current knowledge of the problem (heuristic knowledge + learned knowledge)
- $q > q_0$: probabilistic rule, more exploration, roulette wheel like in the original AS
Virtual Pheromone: Global Update with Elitism

AS: all ants can update pheromones trails in the same way
EAS: all ants update pheromones trails; extra amount for the best tour
ACS: the global update is performed exclusively by the ant that generated the best tour from the beginning of the trial (length $L^+$); it updates only the edges of the best tour ($T^+$) since the beginning of the trial (best-so-far, saving of computing time, no major difference with best-of-iteration)

Update rule for $(i,j)$ edges belonging to $T^+$:

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

with $\Delta \tau_{ij}(t) = 1 / L^+$

Note 1: the result is a more directed exploration, ants are encouraged to search for path in the vicinity of the best tour found so far.

Note 2: notice the weighted sum of old and new pheromones, different from AS.
Virtual Pheromone: Local Update

All ants can perform a local update. When an ant $k$ in city $i$ selects city $j \in J^k_i$ the pheromone concentration on edge $(i, j)$ is updated as follows:

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

$\xi$: parameter; $\xi = 0.1$ from experimental finding, course book $\xi = \rho$

$\tau_0$: parameter, also representing the initial pheromone quantity on all edges (like in AS). From experimental finding: $\tau_0 = (nL_{nn})^{-1}$; $n =$ number of cities, $L_{nn} =$ length of the tour produced by the nearest neighbor heuristic only

**Note:** application of the local update rule make the pheromone level decreases each time that an edge is visited $\rightarrow$ indirectly favor exploration of not yet visited edges $\rightarrow$ avoid stagnation and convergence to a common path
Candidate List

• A candidate list is a list of cities of length $cl$ ($cl =$ algorithmic parameter) to be visited from a given city; cities in the candidate list are ranked according to the inverse of their distance, the list is scanned sequentially.

• An ant first restrict the choice of the next city to those in the candidate list; it uses standard ACS transition rule to select a city.

• Once all the $cl$ closest cities in the candidate list for a given city $i$ have been visited, the next city $j$ is selected from the closest of the yet unvisited cities.
ACS for TSP – Comparison with Other Algorithms

ACS ran for 1250 iterations (end criterion) using 20 ants (25’000 tours generated); average over 15 runs for all implementations; see course book for more results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Shortest tour length</th>
<th># of tours before best tour found</th>
<th>Shortest tour length</th>
<th># of tours before best tour found</th>
<th>Shortest tour length</th>
<th># of tours before best tour found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eil50 (50 cities)</td>
<td>425</td>
<td>1830</td>
<td>428</td>
<td>25000</td>
<td>443</td>
<td>68512</td>
</tr>
<tr>
<td>Eil75 (75 cities)</td>
<td>535</td>
<td>3480</td>
<td>545</td>
<td>80000</td>
<td>580</td>
<td>173250</td>
</tr>
<tr>
<td>KroA100 (100 cities)</td>
<td>21282</td>
<td>4820</td>
<td>21761</td>
<td>103000</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
GA for TSP

- A number of pubs with local search (on top of standard GA operators):
  - Merz and Freisleben, IEEE CEC, 1997
  - Bouhmala N., *Hybrid Metaheuristics* 2004
AS for TSP – Results 50 cities

Example of solution found on Eil50 problem
ACS for TSP – Results on ATT532 Problem
2 Extensions of AS

• Ant Colony System (ACS)
  (Gambardella & Dorigo, 1996; Dorigo & Gambardella, 1997)
  – Different transition rule
  – Different pheromone trail updating rule
  – Use of local update of pheromone trail
  – Use of a candidate list for the choice of the next city

• ACS-3-opt
  (Gambardella & Dorigo, 1996; Dorigo & Gambardella, 1997)
  – Standard ACS + local search
  – In case of TSP problems, 2-opt (2 edges exchanged), 3-opt (3 edges exchanged), and Lin-Kernighan (variable number of edges exchanged) are used as local search algorithms
ACS + Local Search

Loop /* t=0; t:=t+1 */

Place one ant on each node /* there are \( n \) nodes */

For k := 1 to \( m \) /* each ant builds a solution, in this case \( m=n \) */
    For step := 1 to \( n \) /* each ant adds a node to its path */
        Choose the next city to move by applying a probabilistic solution construction rule
    End-for
End-for

Apply local search

Update pheromone trails

Until End_condition /* e.g., \( t=t_{\text{max}} \) */
Example of Local Search: 2-Opt

- 2-opt swapping: (b,f) and (a,c) replaced by (a,f) and (b,c)
- Tour 2 shorter than tour 1
- Pheromones trail of iteration $t-1$ are proper to the edges and follow the swap; afterwards pheromone update of iteration $t$ – evaporation, local and global update
For TSP Problems: Local Search using 3-Opt

- For each ant $k$, at each iteration of ACS, three edges at the time are exchanged iteratively until a local optimum is reached while all other sub-tour orientations are maintained unchanged; full 3-opt, $2^3=8$ combinations;
- in ACS-3-opt restricted permutation: only moves that do not revert the order in which the cities are visited, e.g: $(k,l), (p,q), (r,s) \rightarrow (k,q), (p,s), (r,l)$
- Computational speed up obtained by using nearest neighbor list, etc. See for instance [Bentley 1992].
Advantages and Drawbacks of Local Search

- Local search is **complementary** to ant pheromone mechanisms, so probability it achieves a major impact on a given problem is high
- The **quality of the achieved solution** is in particular improved; the computational cost is increased
- Local search lacks of good starting solutions on which it can perform combinatorial optimization; these solutions are provided by artificial ants using pheromone mechanisms
- Depending on the type of problem, wished solution quality and desired computational cost, an appropriate balance between local search and trail-based algorithms has to be chosen (**no free-lunch theorem!**)  
- **Clear correspondence with GA**: crossover (pheromone) and mutation (local search)
Trail Laying/Following Mechanisms applied to Communication Networks: Ant-Based Routing Algorithms
The Routing Problem

- The practical goal of routing algorithms is to build routing tables

Routing table of node $k$ (N-nodes net)

<table>
<thead>
<tr>
<th>Destination node</th>
<th>1</th>
<th>...</th>
<th>$j$</th>
<th>...</th>
<th>$k-1$</th>
<th>$k+1$</th>
<th>...</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next node</td>
<td>$i_1$</td>
<td>...</td>
<td>$i_j$</td>
<td>...</td>
<td>$i_{k-1}$</td>
<td>$i_{k+1}$</td>
<td>...</td>
<td>$i_N$</td>
</tr>
</tbody>
</table>

- Routing is difficult because costs are dynamic
- Adaptive routing is difficult because changes in the control policy determine changes in the costs and vice versa
Two Main Algorithms up to Date

• **ABC (Ant-Based Control),** [Schoonderwoerd, Holland, et al., 1996]
  – Target: telephone network (symmetric level of congestion on one given source-destination pair)
  – Test: UK telephone network

• **AntNet [DiCaro and Dorigo, 1998]**
  – Target: packet-switching network style Internet
  – Tests: more exhaustive on several networks
Ant-Based Control (ABC) Algorithm
A Simple Network Example with 5 Nodes

Routing table of node 4

<table>
<thead>
<tr>
<th>Neighboring Nodes</th>
<th>Destination Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8 0.2 0.1 0.1</td>
</tr>
<tr>
<td>3</td>
<td>0.1 0.3 0.8 0.1</td>
</tr>
<tr>
<td>5</td>
<td>0.1 0.5 0.1 0.8</td>
</tr>
</tbody>
</table>

Original table

Updated table
ABC Algorithm - Definitions

- \( d = \) destination; \( s = \) source, \( n = \) neighbor node
- Assumption: same level of traffic congestion \( s-d \) and \( d-s \) (ok for telephone networks)
- \( N \) nodes total, \( k_i = \) neighboring nodes to node \( i \)
- Routing table node \( i \) (time-variant matrix with \( k_i \) rows and \( N-1 \) columns):

\[
R_i = \left[ r_{n,d}^i (t) \right]_{k_i,N-1}
\]

- \( r_{n,d}^i (t) : \) For ants: probability that an ant with destination \( d \) will be routed from \( i \) to neighbor \( n \)
- For calls: deterministic path (pick up the higher value for choosing the route)

\[
\sum_n r_{n,d}^i (t) = 1
\]

Sum of all possible routes to neighbors at a given node = 1
ABC Algorithm – Definitions

• Node $i$ has (maximal) capacity $C_i$ (max number of connections) and spare capacity $S_i$ (capacity available for new connections)
• Once call set-up between d and s, each node in the route is decreased in its spare capacity by a given amount
• If no spare capacity left for at least one of the node in the route under construction, the call is rejected
• When a call terminates (hanging up or rejection), the corresponding reserved capacity for each of the nodes in the route is made available again for other nodes
ABC – Updating Rules

- Ants launched from any node (exist an optimal rate) continuously; travel from $s \to d$
- Ants die when they reach $d$
- For routing table updating: $s$ is viewed as $d$ (ant has only information about the traffic at visited nodes; information used by future ants and calls)
- Each visited node’s routing table updated according to:

\[
\begin{align*}
    r^{i}_{i-1,s}(t+1) &= \frac{r^{i}_{i-1,s}(t) + \delta r}{1 + \delta r} & \text{Reinforce} \\
    r^{i}_{n,s}(t) &= \frac{r^{i}_{n,s}(t)}{1 + \delta r} & \text{for } n \neq i-1 & \text{ Decay (normalization to 1)}
\end{align*}
\]

$\delta r$ = reinforcement parameter
ABC – Reinforcement

\[ \delta r = \frac{a}{T} + b \]

\( T = \text{absolute time spent in the network} \)
\( a, b = \text{parameters} \)

Idea: ants have an age, the older they are and the less influence they have on the routing table; ants age faster if they pick up congested routes
ABC – Enforced Delay & Noise

• **Delay imposed** on ant reaching a given node $i$:

\[ D_i = ce^{-dS_i} \]

$c,d$ parameters

Idea: less congested nodes delay less ants

• **Tunable noise** parameter $g$
  
  • $g =$ probability to chose route at random
  
  • $1-g =$ probability to choose route according to routing tables

Idea: increase exploration
ABC - Sample Results

Call failure percentage with different algorithms – static call probabilities

• 30-nodes BT network
• 10 runs
• 15’000 time steps total

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mean</th>
<th>Standard dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without load balancing (fixed, shortest routes)</td>
<td>12.57%</td>
<td>2.16%</td>
</tr>
<tr>
<td>Original mobile agents</td>
<td>9.19%</td>
<td>0.78%</td>
</tr>
<tr>
<td>Improved mobile agents</td>
<td>4.22%</td>
<td>0.77%</td>
</tr>
<tr>
<td>Ants (0% noise)</td>
<td>1.79%</td>
<td>0.54%</td>
</tr>
<tr>
<td>Ants (5% noise)</td>
<td>1.99%</td>
<td>0.54%</td>
</tr>
</tbody>
</table>
ABC - Sample Results

Call failure percentage with different algorithms – dynamic call probabilities

- 30-nodes BT network
- 10 runs
- 15’000 time steps total
- after 7’500 steps different set of call probabilities

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mean</th>
<th>Standard dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without load balancing (fixed, shortest routes)</td>
<td>12.53%</td>
<td>2.04%</td>
</tr>
<tr>
<td>Original mobile agents</td>
<td>9.24%</td>
<td>0.80%</td>
</tr>
<tr>
<td>Improved mobile agents</td>
<td>4.41%</td>
<td>0.85%</td>
</tr>
<tr>
<td>Ants (0% noise)</td>
<td>2.72%</td>
<td>1.24%</td>
</tr>
<tr>
<td>Ants (5% noise)</td>
<td>2.56%</td>
<td>1.05%</td>
</tr>
</tbody>
</table>
AntNet Algorithm
Two Main Algorithms up to Date

• ABC (Ant-Based Control),
  [Schoonderwoerd, Holland, et al., 1996]
  – Target: telephone network;
  – Test: UK telephone network

• AntNet [DiCaro and Dorigo, 1998]
  – Target: packet-switching network style Internet
  – Tests: more exhaustive on several networks
Using Pheromones and Heuristic to Choose the Next Node

\[ p_{ijd}^k(t) = f(\tau_{ijd}(t), \eta_{ij}(t)) \]

- \( \tau_{ijd} \) is the pheromone trail (multiple pheromone trail for the same link i,j!)
- \( \eta_{ij} \) is an heuristic evaluation of link \((i,j)\) which introduces problem specific information (e.g., in AntNet \( \eta_{ij} \) is \( \propto \) to the inverse of link \((i,j)\) queue length)
where the \((i,j)\)'s are the links visited by ant \(k\), and

\[
\Delta \tau_{ijd}^k(t) = \text{quality}^k
\]

where \(\text{quality}^k\) is set proportional to the inverse of the time it took ant \(k\) to build the path from \(i\) to \(d\) via \(j\).
AntNet: Data Structures

- **Routing table \( R_i \):**
  Memorizes probabilities of choosing each neighbor nodes for each possible final destination

- **Trips vector \( \Gamma_i \):**
  contains statistics about ants’ trip times from current node \( i \) to each destination node \( d \) (means and variances);
  used for calculating pheromone reinforcement

<table>
<thead>
<tr>
<th>Trips vector of node ( i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu(i,1) )</td>
</tr>
<tr>
<td>( \sigma^2(i,1) )</td>
</tr>
</tbody>
</table>
AntNet: the Role of F-ants and of B-ants

- **F-ants** collect implicit and explicit information on available paths and traffic load
  - implicit information, through the arrival rate at their destinations
  - explicit information, by storing experienced trip times
- **F-ants** share queues with data packet
- **B-ants** are F-ants which reached their destination; they fast backpropagate info collected by F-ants to visited nodes and update routing tables $\Gamma$ & trip vectors $\Gamma$
- **B-ants** use higher priority queues (usually available on real network for control packages)
**AntNet: The Algorithm**

- **Ants** are launched at regular instants, asynchronously from each node to randomly chosen destinations; modulation of ant rate as a function of traffic
- **Ants** build their paths probabilistically with a probability function of:
  1. *artificial pheromone values* (stored in the routing tables $R$), and
  2. *heuristic values* (length of queues, stored in the trip vectors $\Gamma$)
- **Ants** memorize visited nodes and elapsed times
- Once reached their destination nodes, **ants** retrace their paths backwards, and update the pheromone trails
AntNet: Experimental setup

- Many topologies
- **Realistic simulator** (though not industrial)
- Many traffic patterns
- **Comparison with many state-of-the-art algorithms**
- Performance measures
Experimental Setup:
Network Topologies

American NSF net

Japanese NTT net

6x6 grid net

Simple net
Experimental Setup: Traffic Characteristics

Traffic patterns are obtained by the combination of spatial and temporal distributions for sessions

- **Spatial distributions**
  - Uniform (U)
  - Random (R)
  - Hot Spots (HS)

- **Temporal distributions**
  - Poisson (P)
  - Fixed (F)
  - Temporary (TMP)
Experimental Setup:
Network Load

• **Heavy load near saturation**
  (1000 sec simulation)

• **Heavy load plus transient saturation**
  (1000 sec simulation)
Experimental Setup: Experiments Design

- **Experiment duration:**
  - Each experiment, lasting 1000 sec, is repeated 10 times
  - Before feeding data, routing tables are initialized by a 500 sec phase

- **Experiment typology:**
  - Study of algorithms behavior for increasing network load
  - Study of algorithms behavior for transient saturation
  - Evaluation of influence of control packet traffic on total traffic
Competing Algorithms

AntNet was compared with:

– OSPF (Open Shortest Path First, current official Internet routing algorithm)
– SPF (Shortest Path first)
– ABF (Adaptive Bellman-Ford)
– Q-routing (asynchronous on-line BF)
– PQ-R (Predictive Q-routing)
– Daemon: approximation of an ideal algorithm

It knows at each instant the status of all queues and applies shortest path at each packet hop
Measures of Performance

Standard measures of performance are

- Throughput (bits/sec): quantity of service
- Average packet delay (sec): quality of service

Good routing:

- Under high load: increase throughput for same average delay
- Under low load: decrease average delay per packet
How to Read Results

• Routing is a multi-objective problem (maximizing throughput and minimizing delay)
• Max throughput is the main criterion:
  non max throughput means
  – retransmissions,
  – error notification
  – augmented congestion
• Average packet delay has inherent very high variance
NSFNET & NTTnet (increasing UP traffic)

From Di Caro and Dorigo, 1998, *Journal of Artificial Intelligence Research*

Increasing Uniform-Poisson (UP) traffic

UP traffic increased by reducing the mean session inter arrival time
NSFNET & NTTnet (UP plus transient HS)

Throughput (b/s)

Packet Delay (sec)

Simulation Time (sec)

Data averaged over a 5 seconds sliding window

From Di Caro and Dorigo, 1998
# Routing Overhead

From Di Caro and Dorigo, 1998,
*Journal of Artificial Intelligence Research*

<table>
<thead>
<tr>
<th>Network Type</th>
<th>AntNet</th>
<th>OSPF</th>
<th>SPF</th>
<th>BF</th>
<th>Q-R</th>
<th>PQ-R</th>
<th>Daemon</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleNet</td>
<td>0.33</td>
<td>0.01</td>
<td>0.10</td>
<td>0.07</td>
<td>1.49</td>
<td>2.01</td>
<td>0.00</td>
</tr>
<tr>
<td>NSFNET-UP</td>
<td>2.39</td>
<td>0.15</td>
<td>0.86</td>
<td>1.17</td>
<td>6.96</td>
<td>9.93</td>
<td>0.00</td>
</tr>
<tr>
<td>NSFNET-RP</td>
<td>2.60</td>
<td>0.16</td>
<td>1.07</td>
<td>1.17</td>
<td>5.26</td>
<td>7.74</td>
<td>0.00</td>
</tr>
<tr>
<td>NSFNET-UP-HS</td>
<td>1.63</td>
<td>0.15</td>
<td>1.14</td>
<td>1.17</td>
<td>7.66</td>
<td>8.46</td>
<td>0.00</td>
</tr>
<tr>
<td>NTTnet-UP</td>
<td>2.85</td>
<td>0.14</td>
<td>3.68</td>
<td>1.39</td>
<td>3.72</td>
<td>6.77</td>
<td>0.00</td>
</tr>
<tr>
<td>NTTnet-UP-HS</td>
<td>3.81</td>
<td>0.15</td>
<td>4.56</td>
<td>1.39</td>
<td>3.09</td>
<td>4.81</td>
<td>0.00</td>
</tr>
</tbody>
</table>

*Ratio (10^-3) between bandwidth occupied by the routing packets and the total available network bandwidth*

From Di Caro and Dorigo, 1998
ACO Summary

• ACO metaheuristic
• Overall performance in the literature
• ACO theory
• Applications using ACO
Why Do Ant-Based Systems Work?

Three important components:

- **TIME**: a shorter path receives pheromone quicker (this is often called: “differential length effect”); on-line set-up (e.g. routing): real time; off-line set-up (e.g. TSP): over multiple iterations

- **QUALITY**: a shorter path receives more pheromone

- **COMBINATORICS**: in most real-world problems a shorter path receives pheromone more frequently because it is likely to have a lower number of decision points
What is a Metaheuristic?
(Definition by M. Dorigo)

• A **metaheuristic** is a set of algorithmic concepts that can be used to define or organize heuristic methods applicable to a wide set of different problems

• Examples of **metaheuristic** include
  – simulated annealing
  – tabu search
  – iterated local search
  – genetic algorithms
  – ant colony optimization
The ACO Metaheuristic

Dorigo, Di Caro & Gambardella, 1999

• Ant System and AntNet have been extended so that they can be applied to any shortest path problem on graphs

• The resulting extension is called Ant Colony Optimization metaheuristic
The ACO-Metaheuristics Procedure

procedure \textit{ACO-metaheuristics()}

\textbf{while} (not-termination-criterion)

\textbf{schedule sub-procedures}

\textit{generate-\&-manage-ants()}

\textit{update-pheromones()}

\textit{execute-daemon-actions()}

\{Optional\}

\textbf{end schedule sub-procedures}

\textbf{end while}

\textbf{end procedure}

These are problem specific actions, like local search
ACO: Quality of Results Obtained

SEQUENTIAL ORDERING PROBLEM (SOP)

*Best heuristic currently available*  Gambardella-Dorigo

QUADRATIC ASSIGNMENT PROBLEM (QAP)

*Among best heuristic currently available*  Gambardella-Dorigo-Taillard-Stützle

on “real-world” problems

ROUTING IN CONNECTION-LESS NETWORKS

*Among best heuristics currently available*  Di Caro-Dorigo

VEHICLE ROUTING PROBLEM (VRP)

*Among best heuristics currently available*  Gambardella et al.

for vehicle routing problems with time windows

SHORTEST COMMON SUPERSEQUENCE PROBLEM (SCS)

*Among best heuristics currently available*  Middendorf

TRAVELLING SALESMAN PROBLEM (TSP)

Good results, although not the best  Gambardella-Dorigo-Stützle

GRAPH COLOURING PROBLEM (GCP)

Good results, although not the best  Hertz

SCHEDULING PROBLEM

Promising preliminary results on the single machine weighted total tardiness problem  Dorigo-Stützle

MULTIPLE KNAPSACK PROBLEM (MKP)

Promising preliminary results  Michalewicz
ACO: Theoretical results

- Gutjahr (Future Generation Computer Systems, 2000; Information Processing Letters, 2002) and Stützle and Dorigo (IEEE Trans. on Evolutionary Computation, 2002) have proved **convergence with prob 1 to the optimal solution** of different versions of **ACO**

- Meuleau and Dorigo (Artificial Life Journal, 2002) have shown that there are strong relations between **ACO** and **stochastic gradient descent** in the space of pheromone trails, which converges to a local optima with prob 1

- Birattari et al. (TR, 2000) have shown the tight relationship between **ACO** and **reinforcement learning**

- Rubinstein (TR, 2000) has shown the tight relationship between **ACO** and **Monte Carlo simulation**
ACO: Real-World Applications

- Sequential ordering in a production line
  (Gambardella, under evaluation at MCM, Ferrari subcontractor, Italy)

- Routing of gasoline trucks in Canton Ticino
  (Gambardella, in use by Pina Petroli, Switzerland)

- Job-shop scheduling
  (Bonabeau, in use at Unilever, France)

- Project scheduling
  (Kouranos, in use at Intracom S.A, Greece)

- FaxFactory application
  (Rothkrantz, Delft Universitaet, in use at KPN, Netherlands)

- Water management problems
  (Mariano, Mexican Institute of Water Technology, Mexico)

- Vehicle routing with time windows
  (Gambardella, AntOptima, Migros Supermarkets, Switzerland; Number 1 Logistic Group, Italy)
Additional Literature – Week 3

**Book**

**Papers**