Swarm intelligence

Part 2

Prof. Dr. Marco Dorigo

Examples

• Cemetery organization and brood sorting	➠ data clustering
• Birds flocking	➠ particle swarm optimization
• Foraging	➠ ant colony optimization
• Self-assembly and cooperative transport	➠ robotic implementations
• Division of labor	➠ adaptive task allocation

Examples

- Cemetery organization and brood sorting ➠ data clustering
- Birds flocking ➠ particle swarm optimization
- Foraging ➠ ant colony optimization
- Self-assembly and cooperative transport ➠ robotic implementations
- Division of labor ➠ adaptive task allocation

From scientific to engineering swarm intelligence

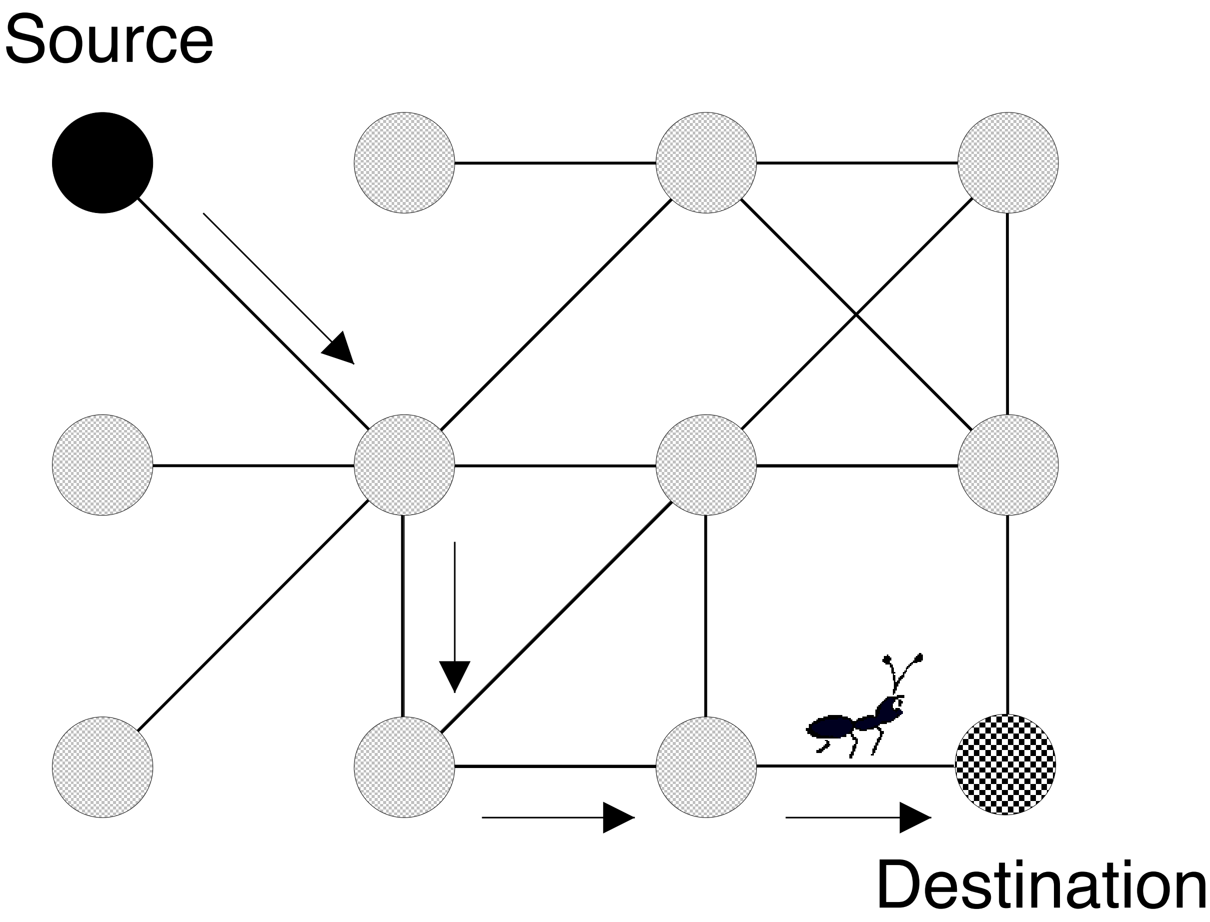
Examples

- Cemetery organization and brood sorting ➠ data clustering
- Birds flocking ➠ particle swarm optimization
- Foraging ➠ ACO: Shortest path / Network routing / Combinatorial optimization
- Self-assembly and cooperative transport ➠ robotic implementations
- Division of labor ➠ adaptive task allocation

Examples

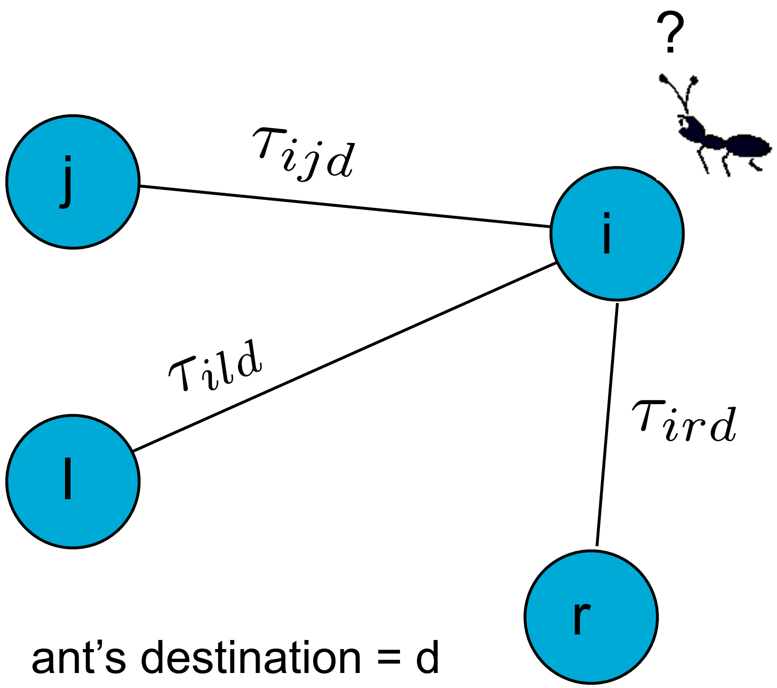
- Cemetery organization and brood sorting ➠ data clustering
- Birds flocking ➠ particle swarm optimization
- Foraging ➠ **ACO:** **Shortest path**, Network routing, Combinatorial optimization
- Self-assembly and cooperative transport ➠ robotic implementations
- Division of labor ➠ adaptive task allocation

From real to artificial ants



Ant colony optimization

Building a solution



ant's destination = d

$$P_{ijd}(t) = f(\tau_{ijd}(t))$$

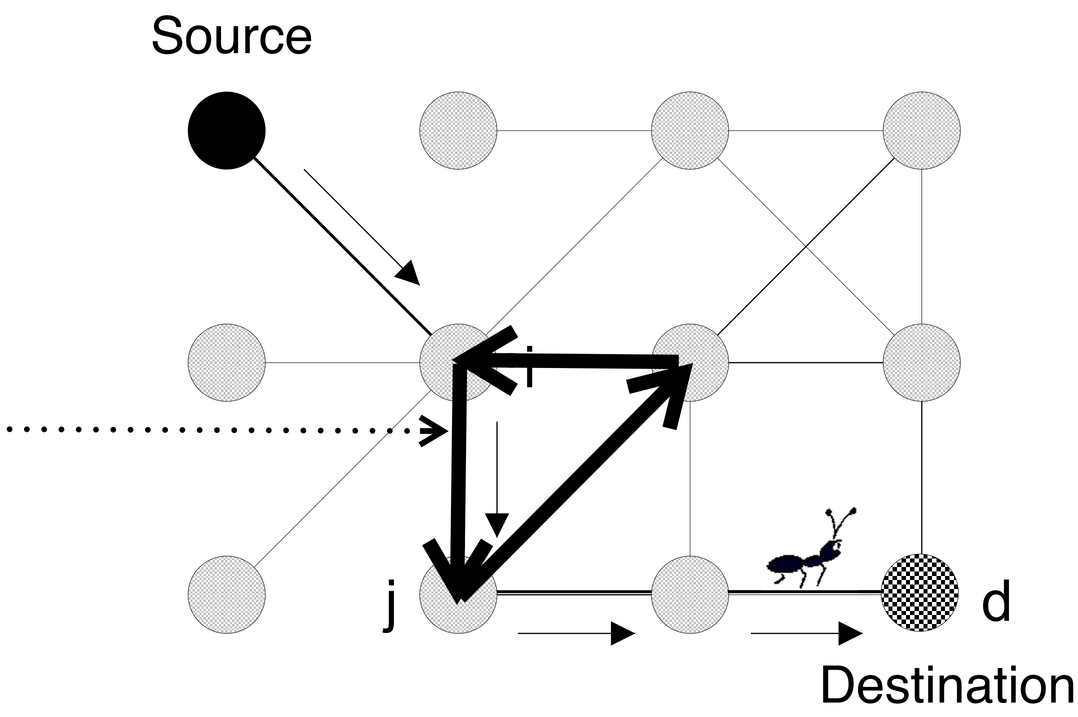
Problem!

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**

Probabilistic solution generation plus pheromone update

-> self-reinforcing loops

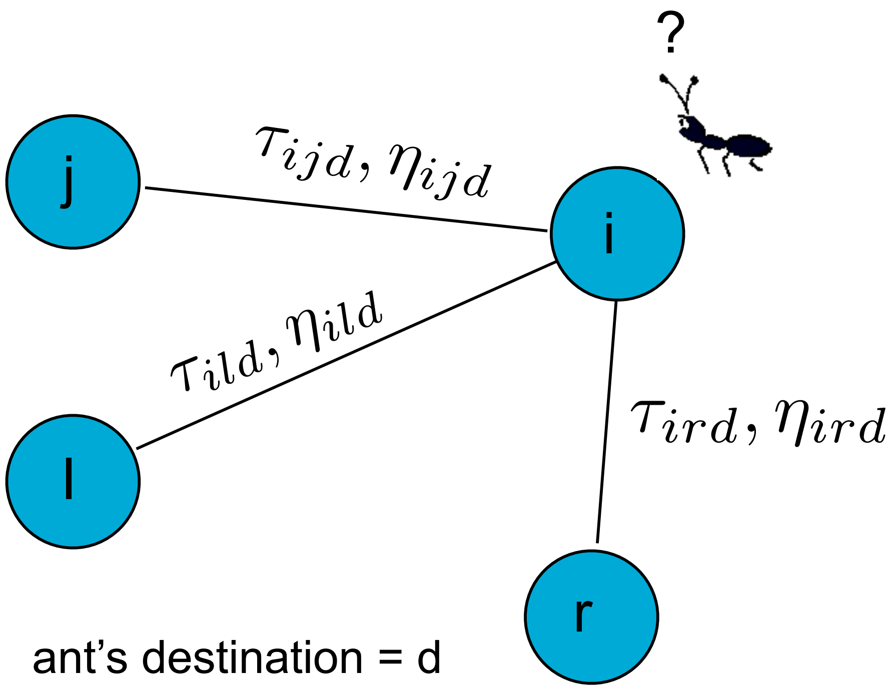
Example of possible self-reinforcing loop



Design choices for artificial ants

- Ants are given a memory of visited nodes
- Ants build solutions probabilistically without updating pheromone trails
- Ants deterministically backward retrace the forward path to update pheromone
- Ants deposit a quantity of pheromone function of the quality of the solution they generated
- Ants can use problem specific heuristic information

Building a solution



ant's destination = d

$$P_{ijd}(t) = f(\tau_{ijd}(t), \eta_{ijd}(t))$$

Building a solution

$$P_{ijd}(t) = f(\tau_{ijd}(t), \eta_{ijd}(t))$$

- τ_{ijd} is the amount of pheromone trail on edge (i,j,d) and is stored in a pheromone table
- η_{ijd} is an heuristic evaluation of link (i,j,d) which introduces problem specific information

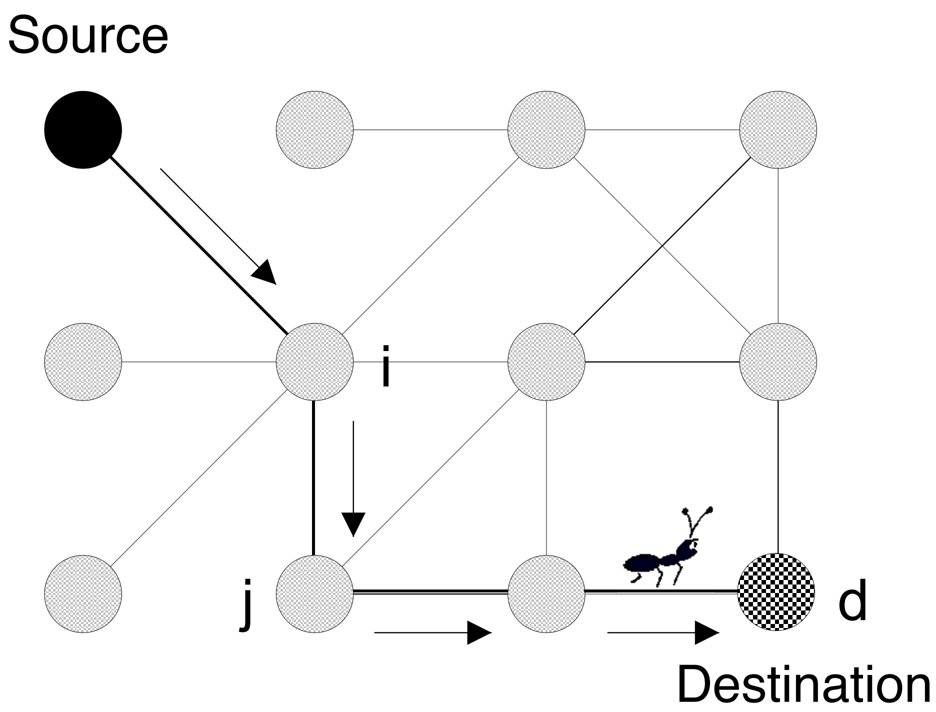
Building a solution

$$P_{ijd}(t) = \frac{[\tau_{ijd}]^{\alpha} \cdot [\eta_{ijd}]^{\beta}}{\sum_{r \in J_i} [\tau_{ird}]^{\alpha} \cdot [\eta_{ird}]^{\beta}}$$

- τ_{ijd} is the amount of pheromone trail on edge (i,j,d) and is stored in a pheromone table
- η_{ijd} is an heuristic evaluation of link (i,j,d) which introduces problem specific information
- J_i is the set of feasible nodes ant k positioned on node i can move to
- α and β are parameters

Ant colony optimization

Updating pheromones



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

Ant colony optimization

Updating pheromones

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

where the *i and j* are the nodes visited by ant *k*, and

$$\Delta\tau_{ijd}(t) = quality^k$$

where $quality^k$ is set proportional to the inverse of the cost (time, length, etc.) paid by ant *k* to build the path from *i* to *d* via *j*

The algorithm

- Ants are launched at regular instants from each node to randomly chosen destinations
- (Forward) Ants build their paths probabilistically with a probability function of:
  - artificial pheromone values, and
  - heuristic values
- Ants memorize visited nodes and costs incurred
- Once reached their destination nodes, (backward) ants retrace their paths backwards, and update the pheromones

Why does it work?

Three important components:

- **TIME**: a shorter path receives pheromone quicker (as in real ants): **differential length effect**
- **QUALITY**: a shorter path receives more pheromone (as in some ant species)
- **COMBINATORICS**: a shorter path receives pheromone more frequently because it is likely to have a lower number of decision points

How Does it Work?

It works very well on

- shortest path problems with dynamic costs (e.g., routing in telecommunications networks)
- constrained shortest path problems (e.g., NP-hard problems)

Artificial vs Real Ants: Main Similarities

- **Colony of individuals**
- **Exploitation of stigmergy & pheromone trail**
  - Stigmergic, indirect communication
  - Pheromone evaporation
  - Local access to information
- **Shortest path & local moves (no jumps)**
- **Stochastic and myopic state transition**

Artificial vs Real Ants: Main Differences

Artificial ants:

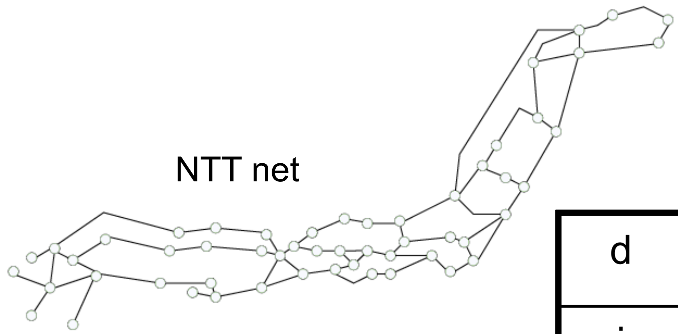
- Live in a discrete world
- Deposit pheromone in a problem dependent way
- Can have extra capabilities
  - Local search, lookahead, backtracking
- Exploit an internal state (memory)
- Deposit an amount of pheromone function of the solution quality
- Can use local heuristic information

From scientific to engineering swarm intelligence

Examples

- Cemetery organization and brood sorting ➠ data clustering
- Birds flocking ➠ particle swarm optimization
- Foraging ➠ Shortest path / **ACO: Network routing** / Combinatorial optimization
- Self-assembly and cooperative transport ➠ robotic implementations
- Division of labor ➠ adaptive task allocation

The routing problem



Routing table of node k

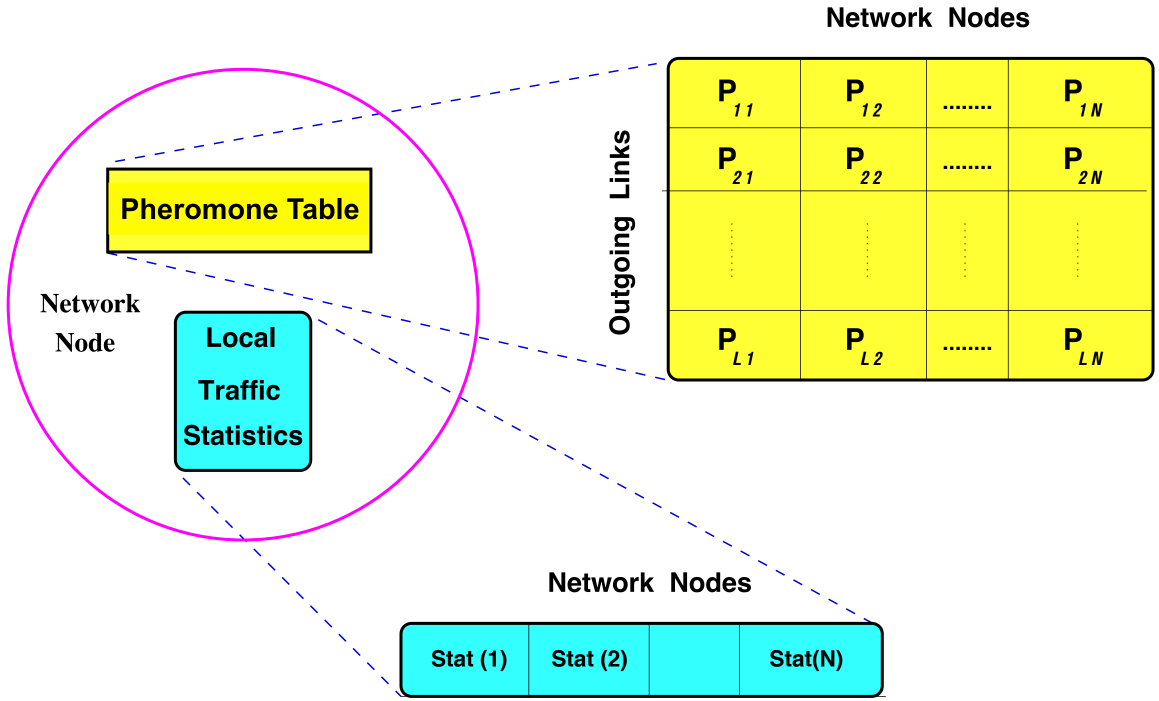
d	1	...	j	...	k-1	k+1	...	N
j	i_1	...	i_j	...	i_{k-1}	i_{k+1}	...	i_N

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

Ant colony optimization: application to routing

The AntNet algorithm

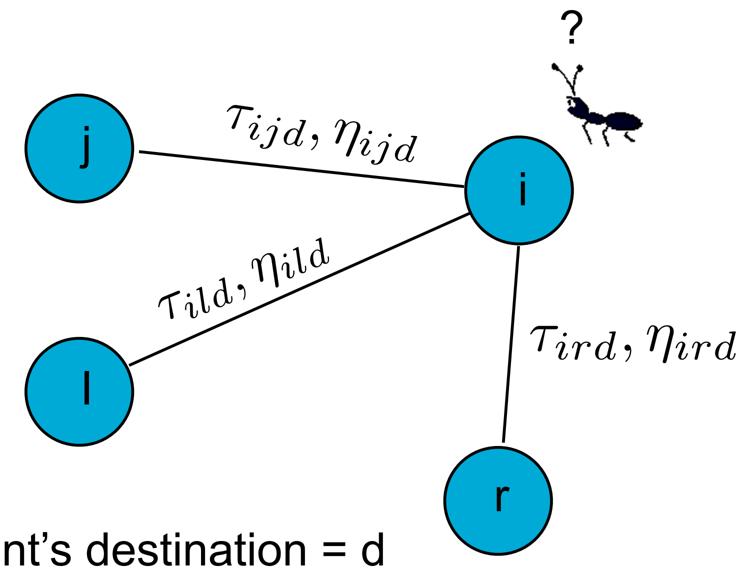
- Ants are launched at regular instants from each node to randomly chosen destinations
- Ants are routed probabilistically with a probability function of:
  - artificial pheromone values, and
  - heuristic values, maintained on the nodes



Ant colony optimization: application to routing

The AntNet algorithm

- Ants are launched at regular instants from each node to randomly chosen destinations
- Ants are routed probabilistically with a probability function of:
  - artificial pheromone values, and
  - heuristic values, maintained on the nodes



ant's destination = d

$$P_{ijd}(t) = f(\tau_{ijd}(t), \eta_{ijd}(t))$$

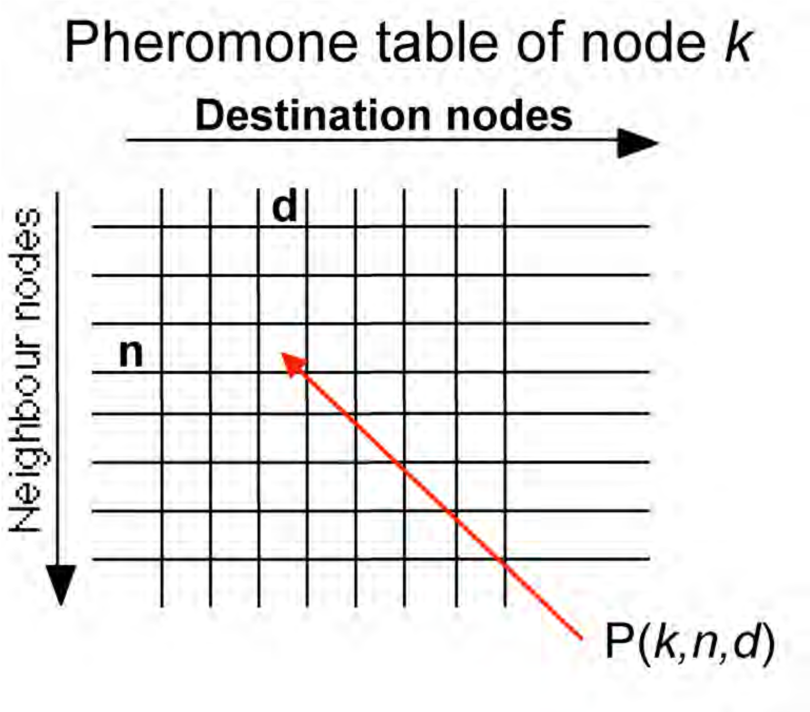
- τ_{ijd} is the pheromone trail
- η_{ijd} is an heuristic evaluation of link (i,j)which introduces problem specific information

AntNet: Data Structures

- **Pheromone table**:

  Pheromone values are normalized to 1

  -



AntNet's Decision Rule

$$P_{ijd}(t) = f(\tau_{ijd}(t), \eta_{ijd}(t))$$

Ant colony optimization: application to routing

AntNet's Decision Rule

$$P(i,j,d) = \frac{\tau(i,j,d) + \alpha \cdot l_j}{1 + \alpha\left(|N_i| - 1\right)}$$

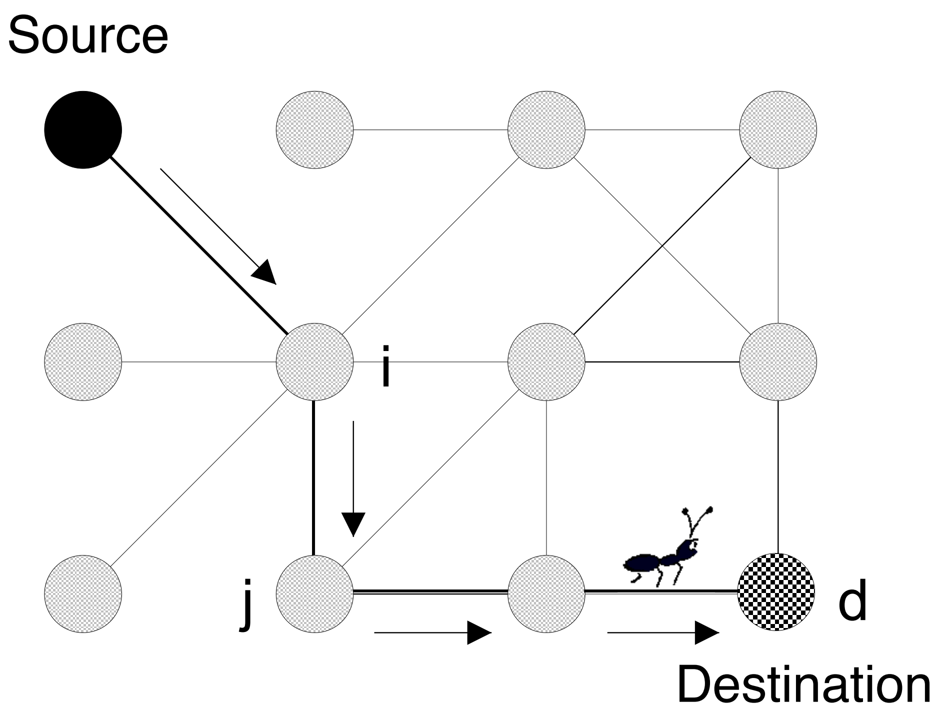
where l_j is a heuristic correction normalized in [0,1] and proportional to the length q_j (in bits waiting to be sent) of the queue of the link connecting node *i* with its neighbor *j*:

$$l_j = 1 - \frac{q_j}{\sum_{k=1}^{|N_i|} q_k}$$

The AntNet algorithm

- Ants are launched at regular instants from each node to randomly chosen destinations
- Ants are routed probabilistically with a probability function of:
  - artificial pheromone values, and
  - heuristic values, maintained on the nodes
- Ants memorize visited nodes and elapsed times
- Once reached their destination nodes, ants retrace their paths backwards and update the pheromone tables

Updating pheromones



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

Ants' Pheromone trail depositing

$$\tau_{ijd}(t+1) = (1-\rho)\cdot\tau_{ijd}(t) + \Delta\tau_{ijd}(t)$$

where the (*i*,*j*)'s are the links visited by ant *k*, and

$$\Delta\tau_{ijd}(t) = f(\text{quality\_of\_solution})$$

where *quality* is set proportional to the inverse of the time it took the ant to build the path from *i* to *d* via *j*

AntNet: Data Structures

- **Trips vector**:
  contains statistics about
  ants' trip times from current node *i*
  to each destination node *d*
  (means and variances)

Trips vector of node *i*

$\mu(i,1)$ $\sigma^2(i,1)$	…	$\mu(i,d)$ $\sigma^2(i,d)$	…	$\mu(i,N)$ $\sigma^2(i,N)$

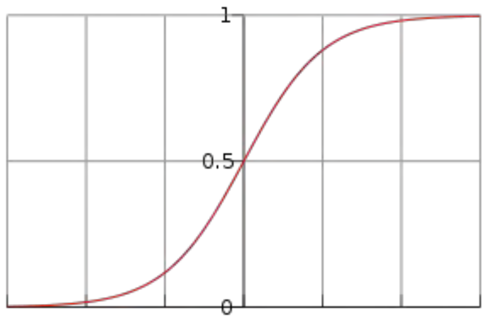
AntNet: Pheromone reinforcement computation

$\Delta\tau_{ijd}(t)$ is a normalized value in]0,1] function of:

- T : time experienced by the artificial ant
- μ : avg time for the same destination memorized in the Trips table
- σ : std. dev. for the same destination memorized in the Trips table

$$\Delta\tau_{ijd}(t) = 1 - f\left(\frac{T}{\mu + \sigma}\right)$$

where *f* is a sigmoid between 0 and 1



AntNet: The Algorithm

- **Ants** (**F-ants**) are launched at regular instants from each node to randomly chosen destinations
- **Ants** are routed probabilistically with a probability function of:
  (i) **artificial pheromone values**, and
  (ii) **heuristic values**,
  maintained on the nodes
- **Ants** memorize visited nodes and elapsed times
- Once reached their destination nodes, **ants** retrace their paths backwards (**B-ants**), and update the pheromone tables

AntNet is distributed and not synchronized

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 (remember the differential length effect)
  - explicit information, by storing experienced trip times
- **F-ants** share queues with data packet
- **B-ants** fast backpropagate info collected by **F-ants** to visited nodes
- **B-ants** use higher priority queues

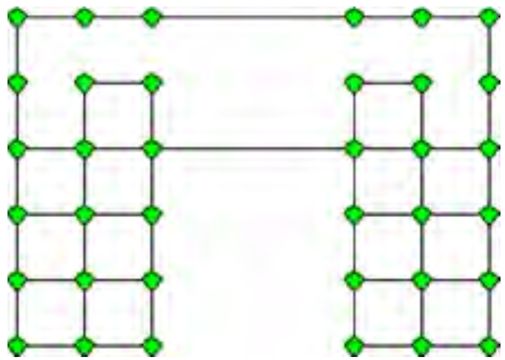
Experimental setup

- Realistic simulator
- Many topologies
- Many traffic patterns
- Comparison with many state-of-the-art algorithms
- Performance measures:
  - throughput (bit/sec) measures the quantity of service
  - average packet delay (sec) measures the quality of service

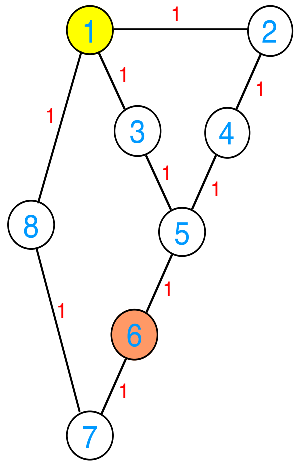
Ant colony optimization: application to routing

Experimental setup

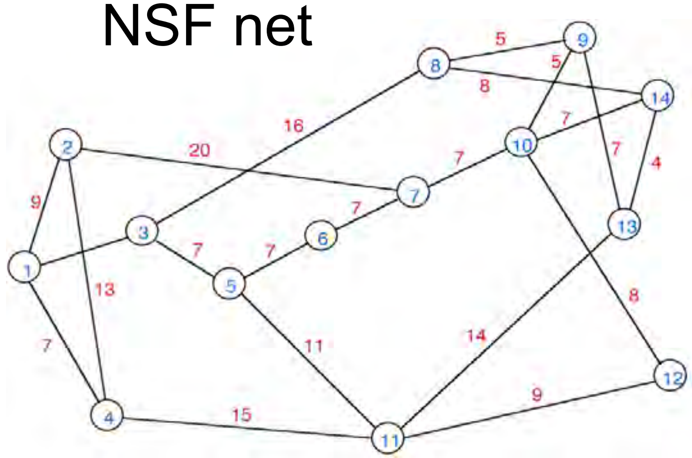
6x6 grid net
(36 nodes)



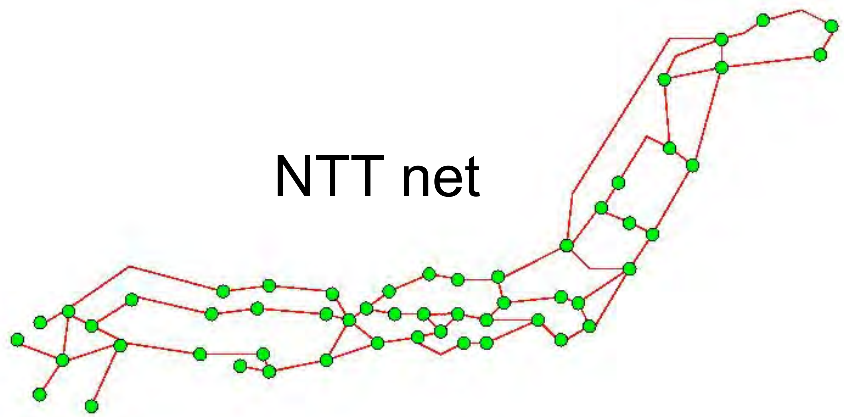
SIMPLEnet
(8 nodes)



NSF net



NTT net



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

Competing Algorithms

AntNet was compared with:

- OSPF
- SPF
- Adaptive BF
- Q-routing (asynchronous on-line BF)
- PQ-R
- 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

Good routing:

- Under high load: increase throughput for same average delay
- Under low load: decrease avg delay per packet

Measures of performance are

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

NSFNET & NTTnet

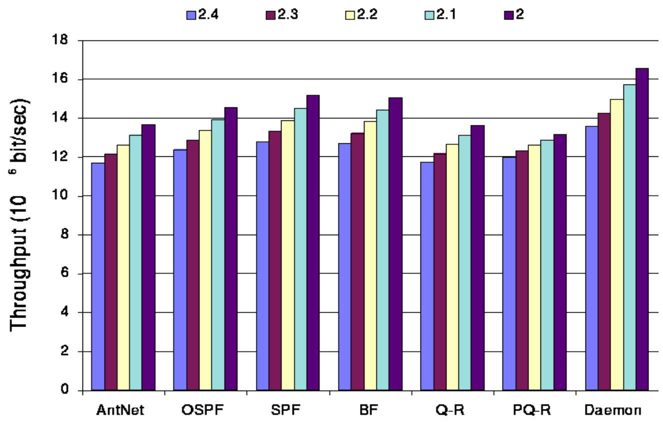
(increasing UP traffic)

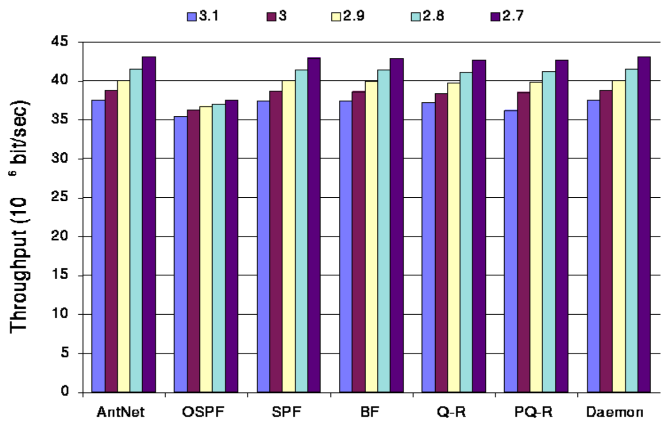
From Di Caro and Dorigo, 1998,
Journal of Artificial Intelligence Research

NSF net

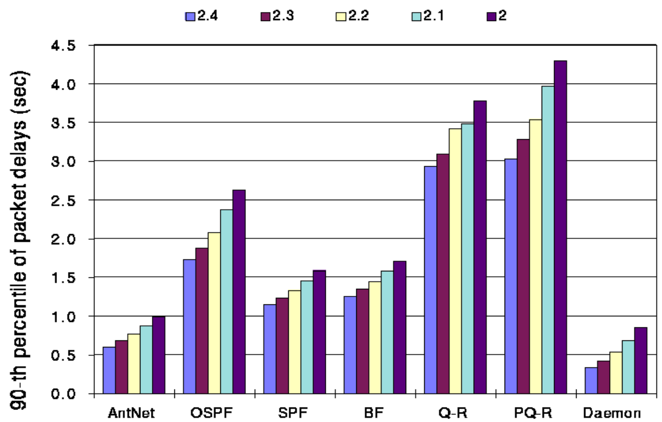
NTT net

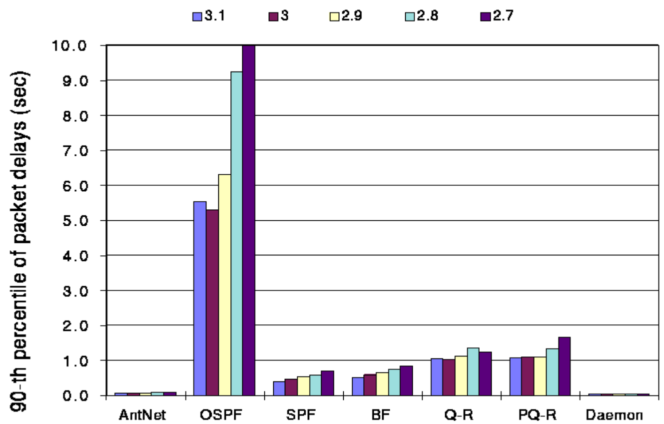
Throughput (b/s)





Avg packet delay

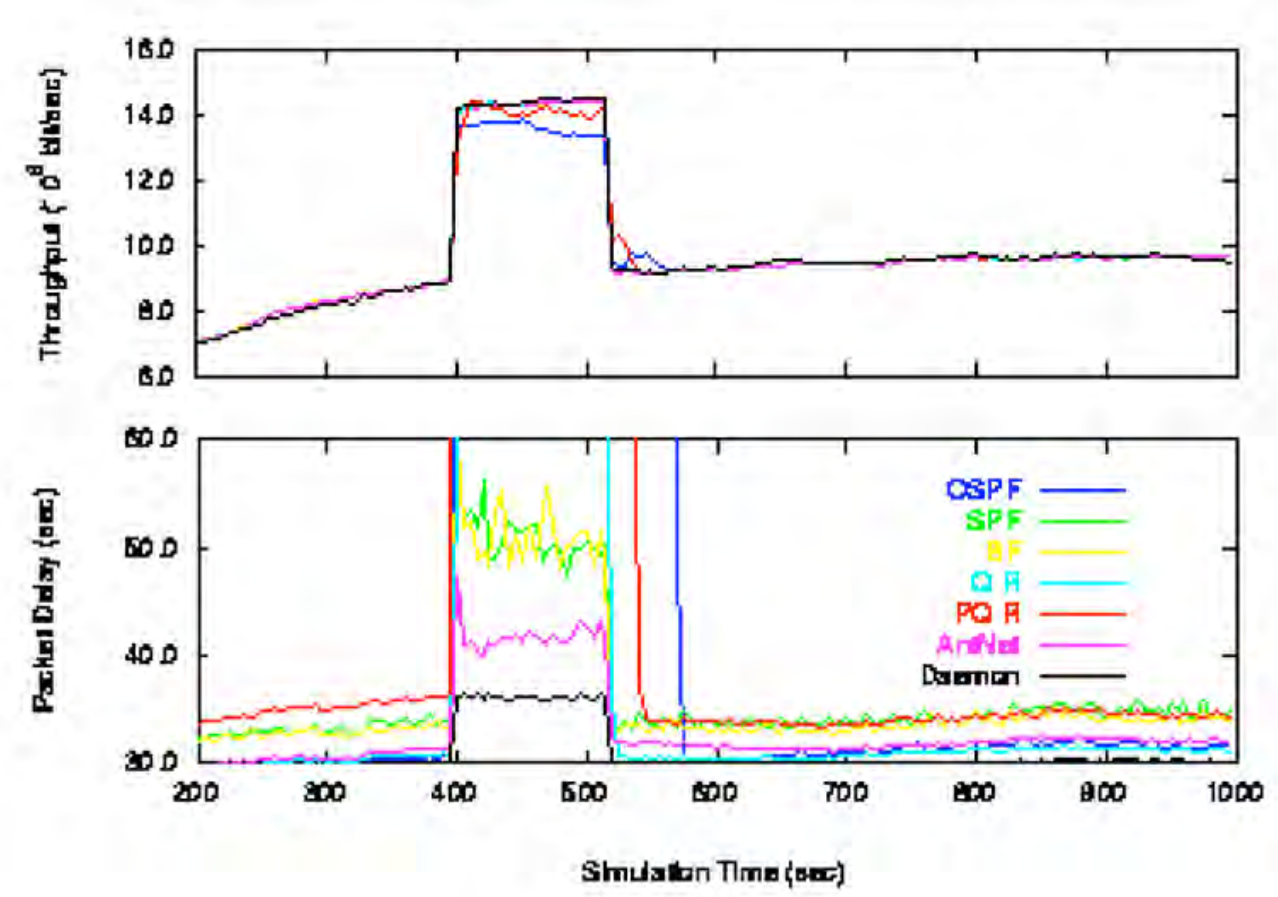




Increasing UP traffic
UP traffic increased by reducing the mean session inter arrival time

Ant colony optimization: application to routing

Results



NSF net – temporary saturation

Data averaged over a 5 seconds sliding window

AntNet Experiments: Summary

- Under low load all tested algorithms have similar performance
- Under high-load (near saturation) AntNet is the best algorithm
- Under a sudden variation in traffic load AntNet remains the best algorithm both in terms of throughput and of delay
- AntNet's overhead is negligible

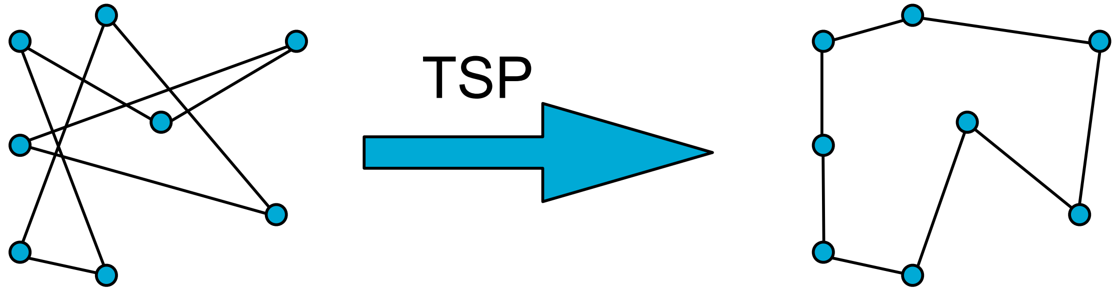
From scientific to engineering swarm intelligence

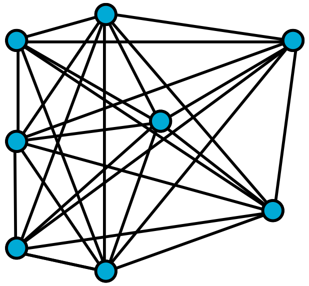
Examples

- Cemetery organization and brood sorting ➠ data clustering
- Birds flocking ➠ particle swarm optimization
- Foraging ➠ **ACO:** Shortest path, Network routing, **Combinatorial optimization**
- Self-assembly and cooperative transport ➠ robotic implementations
- Division of labor ➠ adaptive task allocation

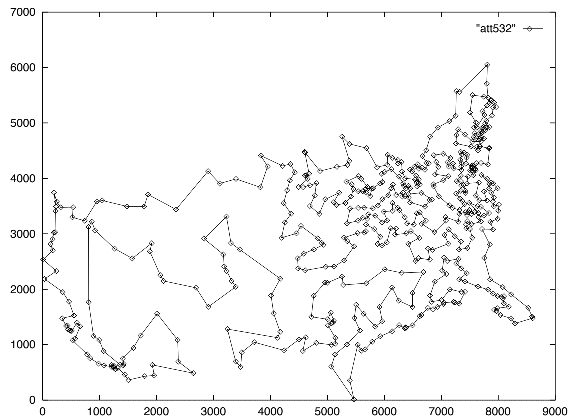
Ant colony optimization

Combinatorial optimization problems



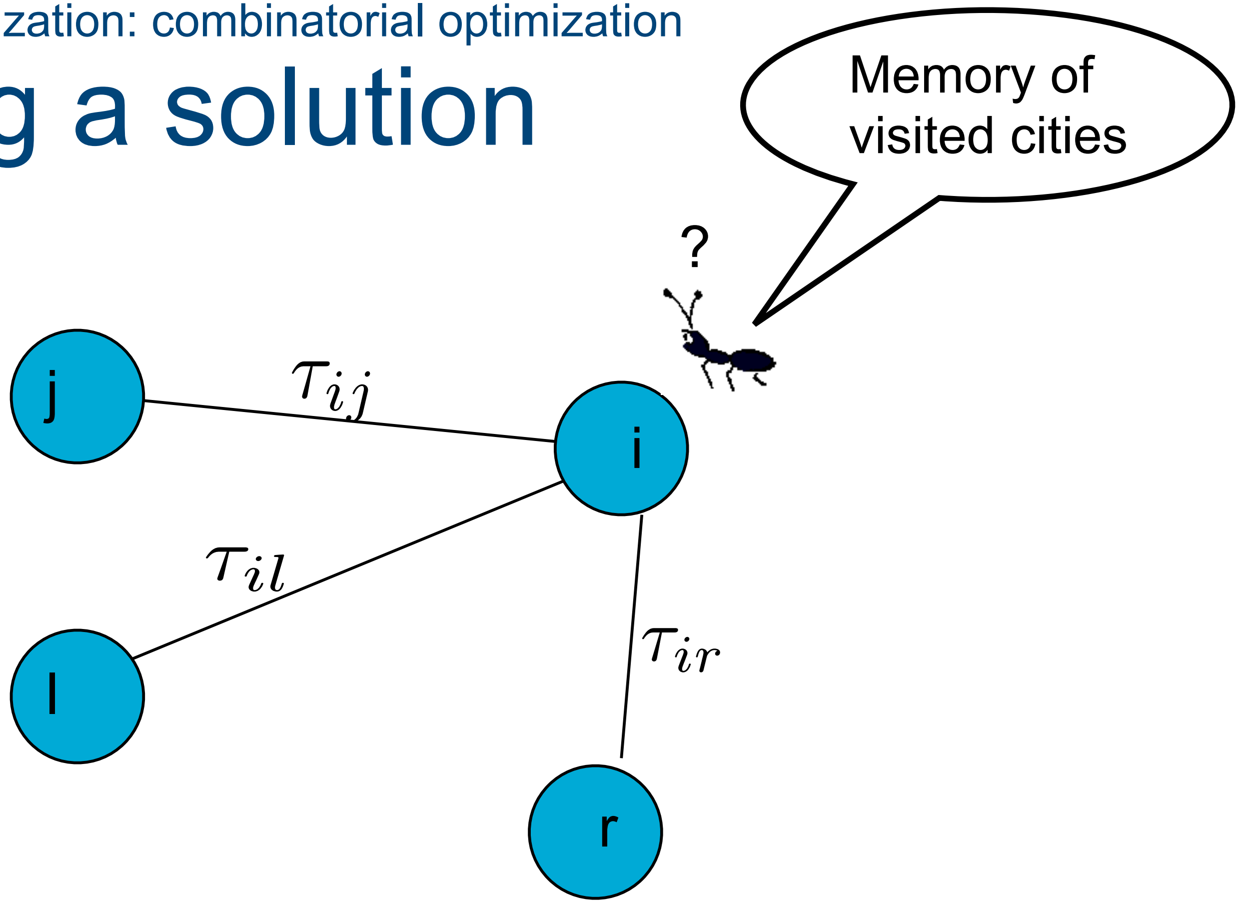


Construction graph used by artificial ants



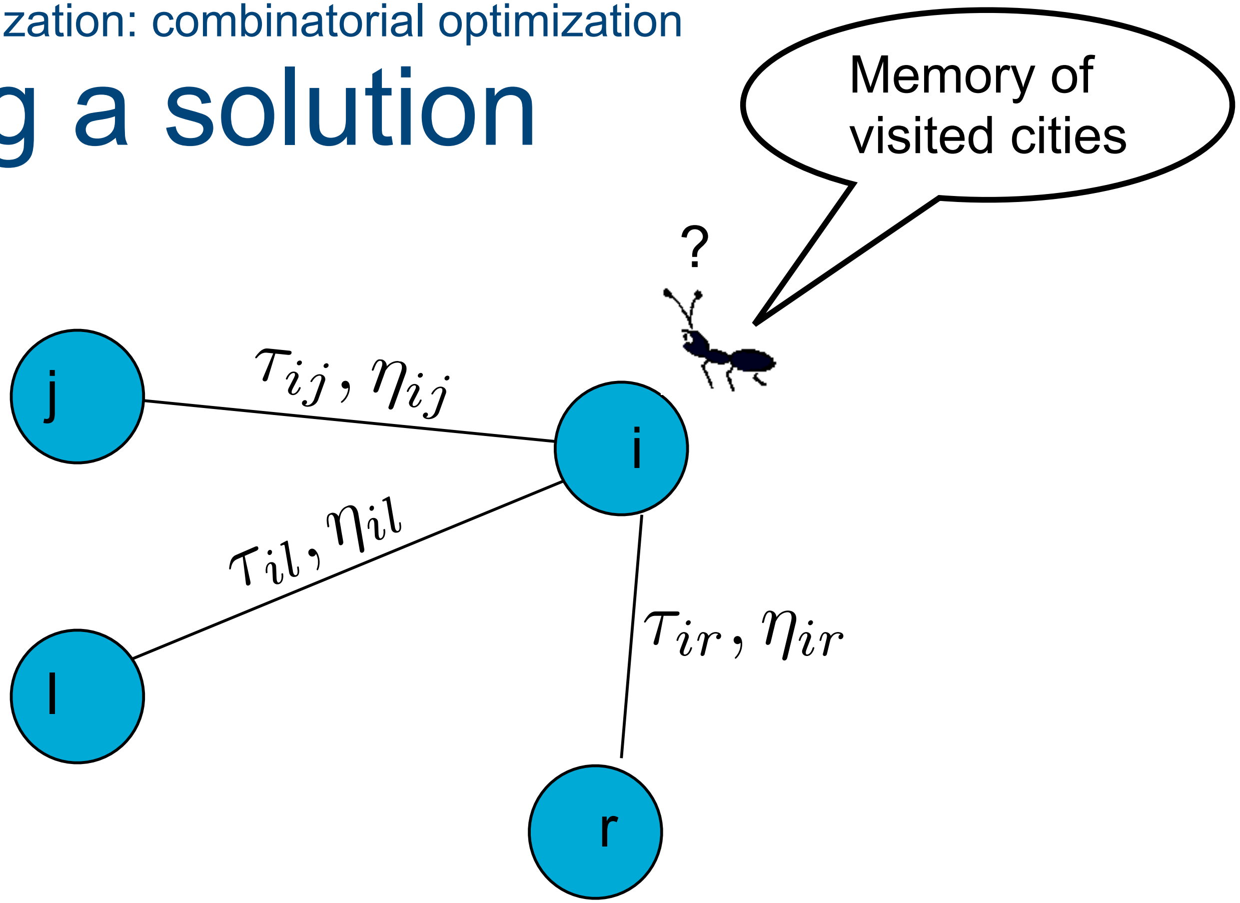
The TSP

Building a solution



$$P_{ij}(t) = \frac{[\tau_{ij}]^{\alpha}}{\sum_{q \in J} [\tau_{iq}]^{\alpha}}$$

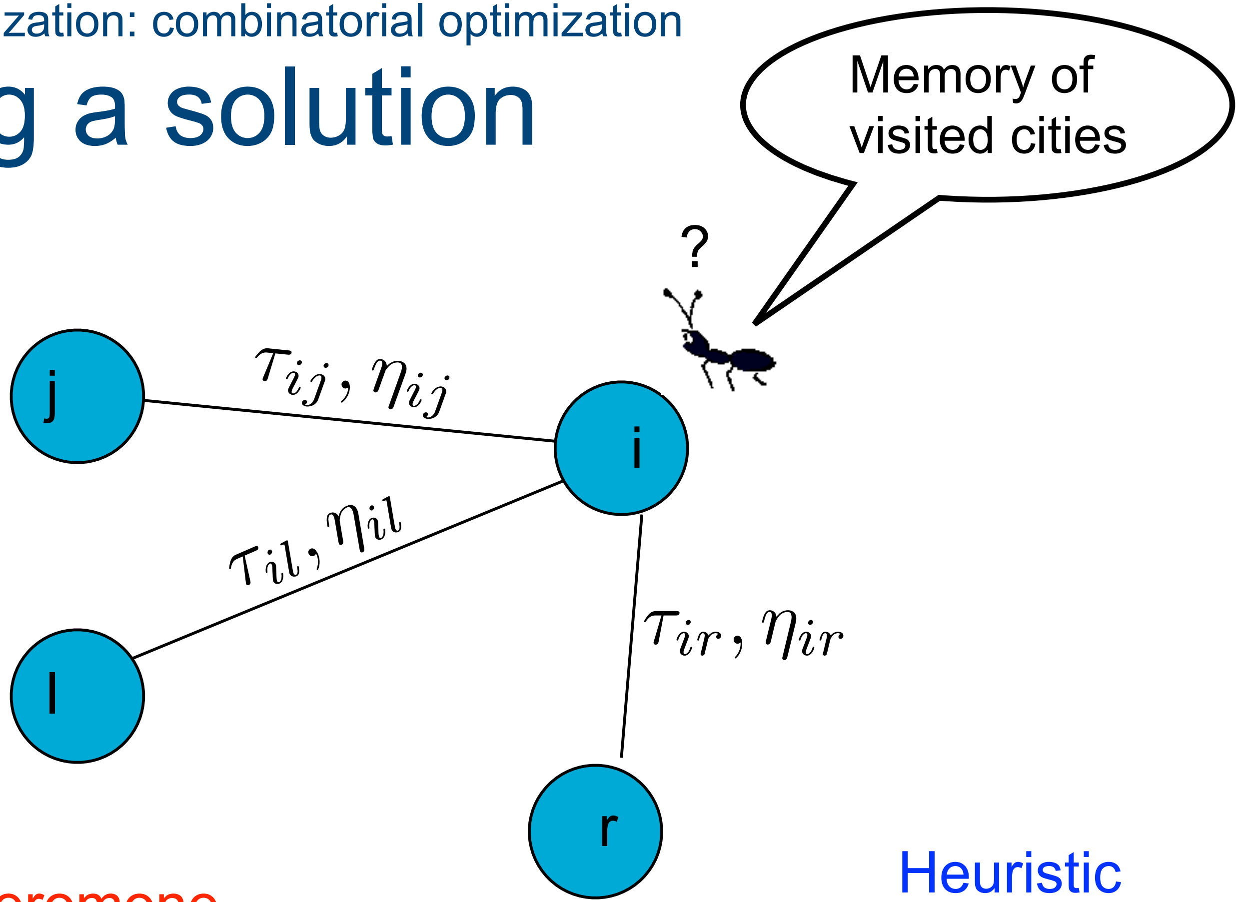
Building a solution



$$P_{ij}(t) = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{q \in J} [\tau_{iq}]^{\alpha} \cdot [\eta_{iq}]^{\beta}}$$

Ant colony optimization: combinatorial optimization

Building a solution



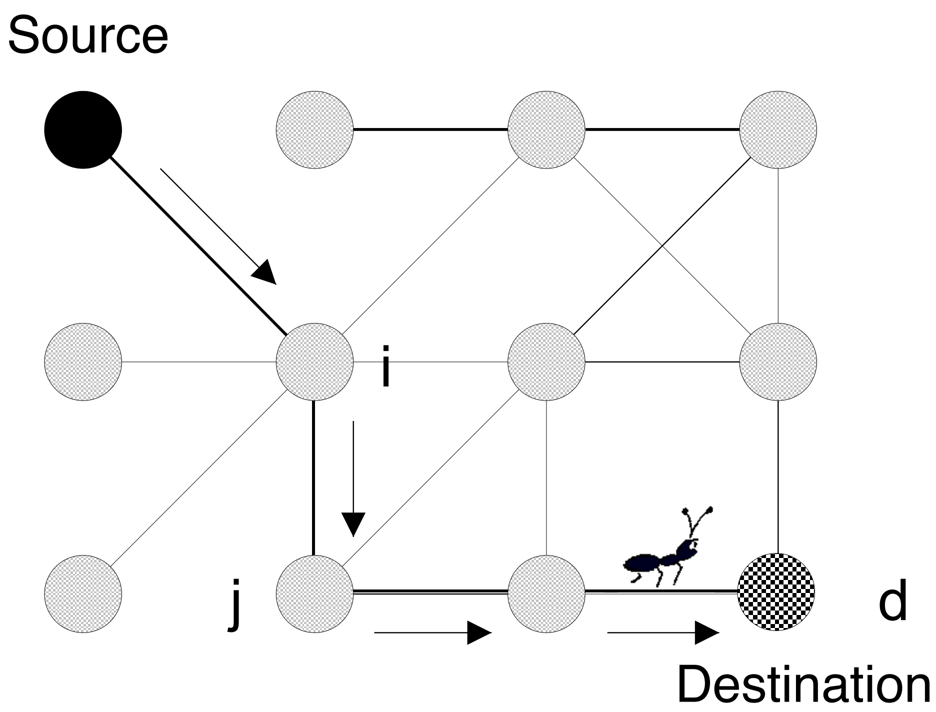
Pheromone

Heuristic

$$P_{ij}(t) = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{q \in J} [\tau_{iq}]^{\alpha} \cdot [\eta_{iq}]^{\beta}}$$

Memory

Updating pheromones



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

Ants' Pheromone Trail Depositing

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

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

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

where $$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

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

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

The ***Ant System*** Algorithm

Dorigo, Maniezzo, Colorni, 1991

Loop

Place one ant on each city *there are #_cities = #_ants cities *

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

Successors and Extensions of Ant System

- **Elitist AS (EAS)**(Dorigo et al., 1991; 1996)
  - The iteration best solution adds more pheromone
- **Rank-Based AS (AS*rank*)**(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; Dorigo & Gambardella, 1997)
  - Pheromones are updated also while building solutions
  - Only iteration best or best-so-far ants can add pheromone
  - Local search added
- **ANTS**(Maniezzo, 1999)
  - Use of bounds to direct the search

Elitist AS

Dorigo et al., 1991; 1996

Idea: give additional pheromone to good solutions

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

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

e = number of elitist ants

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\cdot\Delta\tau_{ij}^{bs}$$

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1/L^{bs} & \text{if } arc(i,j) \in T^{bs} \\ 0 & \text{otherwise} \end{cases}$$

best tour found so-far

Rank-Based AS (AS$_{rank}$)

Bullnheimer et al., 1997; 1999

Idea: give pheromone to solutions proportionally to their rank

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

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

$$\Delta\tau_{ij} = \boxed{\sum_{r=1}^{w-1}(w-r)\cdot\Delta\tau_{ij}^{r} + w\cdot\Delta\tau_{ij}^{bs}}$$

$$\Delta\tau_{ij}^{r} = \begin{cases} 1/L^{r} & \text{if } arc(i,j) \in T^{r} \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1/L^{bs} & \text{if } arc(i,j) \in T^{bs} \\ 0 & \text{otherwise} \end{cases}$$

MaxMin Ant System (MMAS)

Stützle and Hoos, 1997

Idea: give pheromone only to the best solution
(either the best-so-far or the iteration-best)

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

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

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/L^{r} & \text{if } arc(i,j) \in T^{best} \\ 0 & \text{otherwise} \end{cases}$$

MaxMin Ant System (MMAS)

Stützle and Hoos, 1997

- Pheromone trail values τ_{ij} are limited to an interval $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$

$$\tau_{max} = \frac{1}{\rho \cdot L^{opt}} \quad \Rightarrow \quad \hat{\tau}_{max} = \frac{1}{\rho \cdot L^{bs}}$$

$$\tau_{min} = \frac{\hat{\tau}_{max}}{a} \qquad \tau_0 = \hat{\tau}_{max}$$

MaxMin Ant System (MMAS)

Stützle and Hoos, 1997

- Pheromones are reinitialized:
  - when the system stagnates or
  - when no improved solution has been generated for a certain number of consecutive iterations

Ant Colony System (ACS)

Gambardella & Dorigo, 1996; Dorigo & Gambardella, 1997

Three main ideas:

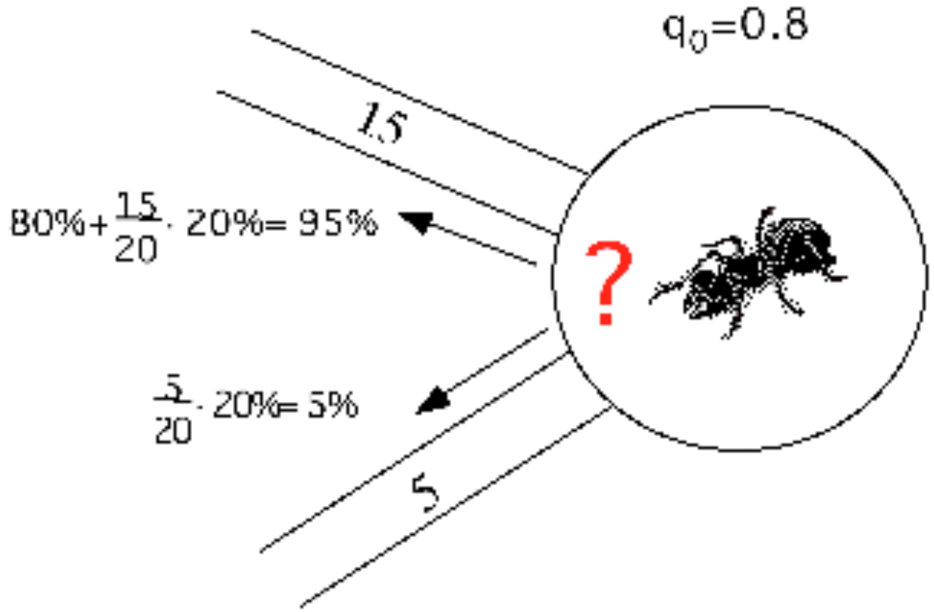
- Different **state transition rule**
- Different **global pheromone trail update** rule
- New **local pheromone trail update** rule

ACS's State Transition Rule

Next state:

with probability q_0
exploitation

with probability (1-q_0)
biased exploration



ACS's State Transition Rule

$$j = \begin{cases} \arg\max_{j \in J_i^k} \left\{ [\tau_{ij}] \cdot [\eta_{ij}]^{\beta} \right\} & \text{if } q \leq q_0 \quad \text{(Exploitation)} \\ J & \text{otherwise} \quad \text{(Exploration)} \end{cases}$$

where *J* is a stochastic variable distributed as follows:

$$p_{ij}^k = \frac{[\tau_{ij}] \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}] \cdot [\eta_{il}]^{\beta}}$$

and β and q_0 are parameters

ACS's State Transition Rule

$$j=\begin{cases} \arg\max_{j\in J_i^k}\left\{[\tau_{ij}]\cdot[\eta_{ij}]^{\beta}\right\} & \text{if } q\le q_0 \quad \text{(Exploitation)} \\ J & \text{otherwise} \quad \text{(Exploration)} \end{cases}$$

$\alpha = 1$

where J is a stochastic variable distributed as follows:

$$p_{ij}^{k}=\frac{[\tau_{ij}]\cdot[\eta_{ij}]^{\beta}}{\sum_{l\in J_i^k}[\tau_{il}]\cdot[\eta_{il}]^{\beta}}$$

and β and q_0 are parameters

ACS's Global Pheromone Trail Update

Pheromone modified only on edges of the best tour so far

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} \qquad \forall (i,j) \in T^{bs}$$

$$\tau_{ij} \leftarrow \tau_{ij} + \rho \cdot \Delta\tau_{ij}^{bs} \quad \forall (i,j) \in T^{bs}$$

where $\Delta\tau_{ij}^{\mathrm{bs}} = \dfrac{1}{L^{\mathrm{bs}}}$

ACS's Local Pheromone Trail Update

While building a solution each ants updates pheromones on visited edges:

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

This update rule introduces diversification

The ACS Algorithm

Loop
  Randomly position #ants ants on #cities cities
  For step=1 to #cities
    For k=1 to #ants
      Apply the **state transition rule**
      Apply the **online trail updating rule**
    End-for
  End-for
  Apply the **offline trail updating rule**
Until End_condition

ACS plus local search

(Gambardella & Dorigo, 1996; Dorigo & Gambardella, 1997)

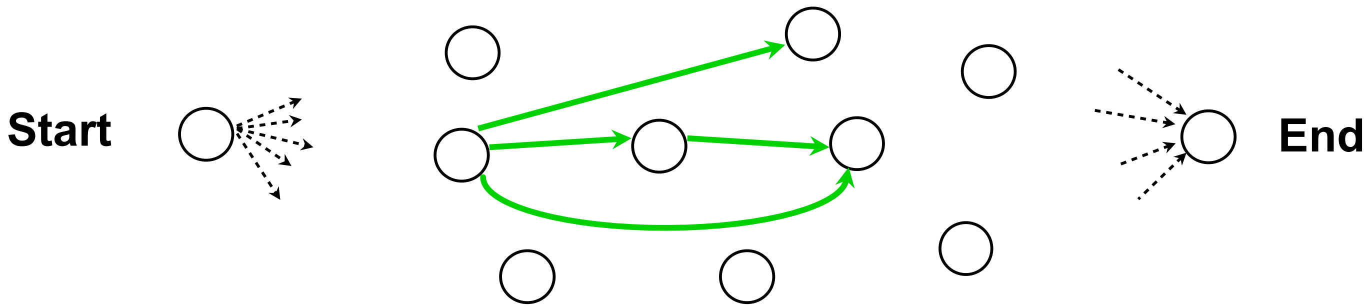
```
Loop
    Randomly position #ants ants on #cities cities
    For step=1 to #cities
        For k=1 to #ants
            Apply the state transition rule
            Apply the online trail updating rule
        End-for
    End-for
    Apply local search
    Apply the offline trail updating rule
Until End_condition
```

Applying ACO to Other COPs

- Graph representation
- Constructive heuristic
- Pheromone trail distribution mechanism
- Mechanism to force feasible solutions

Example:
The Sequential Ordering Problem

- Find a minimum weight Hamiltonian path on a directed graph with weights on arcs, subject to multiple precedence constraints among nodes

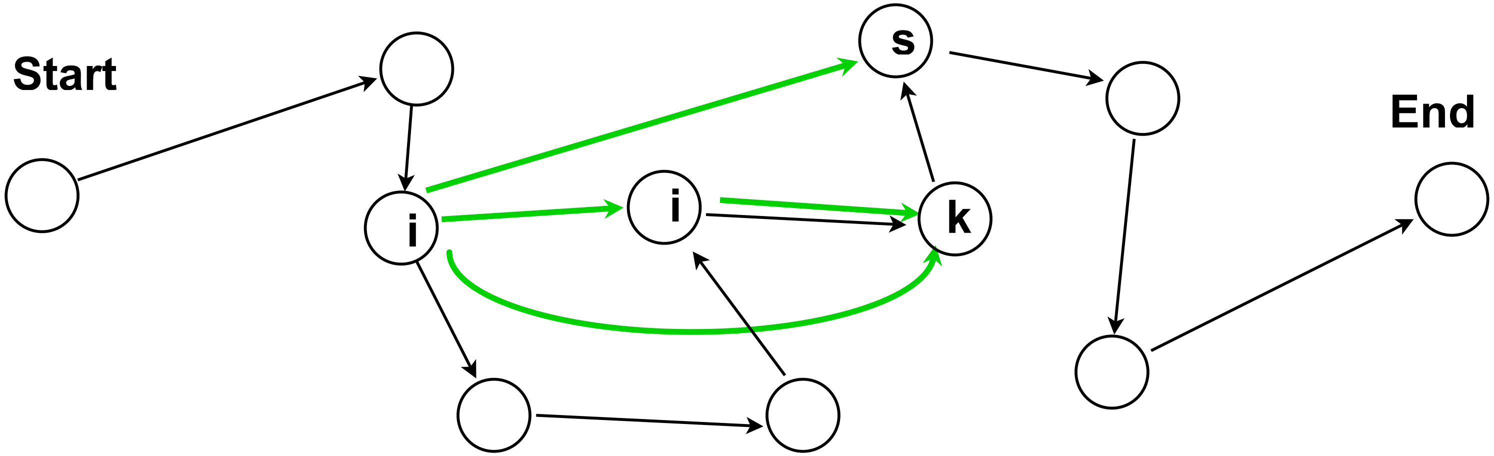


- Very similar to an ATSP in which a solution is a path which visits all cities once and connects an origin to a destination node without closing the tour

- The SOP models real-world problems like single-vehicle routing problems with pick-up and delivery constraints, production planning, and transportation problems in flexible manufacturing systems

The Sequential Ordering Problem

Find the minimal sequence from node **Start** to node **End** that visits all the nodes and do not violate precedence constraints



The HAS-SOP Algorithm

Loop
  Position *m* ants on the first city
  For step=1 to *n*
    For k=1 to *m*
      Apply the **state transition rule**
    End-for
  End-for
  Apply local search
  Apply the **trail updating rule**
Until End_condition

HAS-SOP:

Results on a Set of 23 TSPLIB Test Problems

HAS-SOP results:

Red: In 14 problems out of 23 it found a result which impro- ved the TSPLIB upper bound

Blue: In 6 problem it found the known optimal solution

Green: In 3 problems it found the same value as the TSPLIB upper bound

PROB	n	|R|	TSPLIB Bounds	Best Result	Avg Result	Std.Dev.	Avg Time (sec)
ESC63.sop	65	95	62	62	62.0	0	0.1
ESC78.sop	80	77	18230	18230	18230.0	0	6.9
ft53.1.sop	54	12	[7438,7570]	7531	7531.0	0	9.9
ft53.2.sop	54	25	[7630,8335]	8026	8026.0	0	18.4
ft53.3.sop	54	48	[9473,10935]	10262	10262.0	0	2.9
ft53.4.sop	54	63	14425	14425	14425.0	0	0.4
ft70.1.sop	71	17	39313	39313	39313.0	0	29.8
ft70.2.sop	71	35	[39739,40422]	40419	40433.5	24.6	114.1
ft70.3.sop	71	68	[41305,42535]	42535	42535.0	0	64.4
ft70.4.sop	71	86	[52269,53562]	53530	53566.5	7.6	38.2
kro124p.1.sop	101	25	[37722,40186]	39420	39420.0	0	115.2
kro124p.2.sop	101	49	[38534,41677]	41336	41336.0	0	119.3
kro124p.3.sop	101	97	[40967,50876]	49499	49648.8	249.7	262.8
kro124p.4.sop	101	131	[64858,76103]	76103	76103.0	0	57.4
prob.100.sop	100	41	[1024,1385]	1226	1302.4	39.4	1918.7
rbg109a.sop	111	622	1038	1038	1038.0	0	14.6
rbg150a.sop	152	952	[1748,1750]	1750	1750.0	0	159.1
rbg174a.sop	176	1113	2033	2033	2034.7	1.4	99.3
rbg253a.sop	255	1721	[2928,2987]	2950	2950.0	0	81.5
rbg323a.sop	325	2412	[3136,3157]	3141	3146.0	1.4	1685.5
rbg341a.sop	343	2542	[2543,2597]	2580	2591.9	11.8	2149.6
rbg358a.sop	360	3239	[2518,2599]	2555	2561.2	5.2	2169.3
rbg378a.sop	380	3069	[2761,2833]	2817	2834.3	10.7	2640.3

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 (minimum cost) problem on graphs
- The resulting extension is called ***Ant Colony Optimization metaheuristic***
- 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

ACO: Academic applications

- **TSP-like problems**
  - sequential ordering, vehicle routing
- **Assignment problems**
  - QAP, generalized assignment, frequency assignment, timetabling, graph coloring
- **Scheduling problems**
  - single machine total weighted tardiness, permutation flow shop, job-shop, open-shop, group-shop, resource constrained project scheduling
- **Subset problems**
  - set covering, multiple knapsack, maximum independent set
- **Other problems**
  - learning Bayesian networks, learning fuzzy rules, and many more …

ACO: Real world applications

- **Vehicle routing with time windows**
  - AntOptima, Migros Supermarkets, Switzerland;
  - AntOptima, N1-Barilla, Italy
- **Optimization of production schedule**
  - NUTECH, in use at Air Liquide, USA
- **Routing of gasoline trucks in Canton Ticino**
  - AntOptima, in use by Pina Petroli, Switzerland
- **Job-shop scheduling**
  - EuroBios, in use at Unilever, France
- **Project scheduling**
  - Dr. Kouranos, in use at Intracom S.A , Greece

Theoretical results

- Gutjahr (2000; 2002) and Stützle & Dorigo (2002) have proved **convergence with prob 1 to the optimal solution** of different versions of ACO
- Meuleau & Dorigo (2002) have shown strong relations between ACO and **stochastic gradient descent** in the space of pheromone trails, proving **convergence to a local optimum with prob 1**
- Birattari et al. (2002) have shown the tight relationship between ACO and **dynamic programming**
- Zlochin et al. (2004) have shown the tight relationship between ACO and **estimation of distribution algorithms**
- Blum & Dorigo (2004) have shown that, in case of unconstrained problems, the expected value of solutions generated by AS increases over time
- Blum & Dorigo (2005) have studied the role of **search bias**
- Merkle & Middendorf (2002; 2004) have studied the **dynamics of ACO** models