Biologically Inspired Computation

Ant Colony Optimisation

Swarm Algorithms

Inspiration from swarm intelligence has led to some highly successful optimisation algorithms.

- **Ant Colony (-based) Optimisation** – a way to solve optimisation problems based on the way that ants indirectly communicate directions to each other.

- **Particle Swarm Optimisation** — a different way to solve optimisation problems, based on the swarming behaviour of several kinds of organisms.

Emergent Problem Solving in *Lasius Niger* ants,

For *Lasius Niger* ants, [Franks, 89] observed:

- regulation of nest temperature within 1 degree celsius range;
- forming bridges;
- raiding specific areas for food;
- building and protecting nest;
- sorting brood and food items;
- cooperating in carrying large items;
- emigration of a colony;
- finding shortest route from nest to food source;
- preferentially exploiting the richest food source available.

These are swarm behaviours – beyond what any individual can do.

Explainable (arguably) as emergent property of many individuals operating simple rules?

For *Lasius Niger* ants, [Franks, 89] observed:

- regulation of nest temperature within 1 degree celsius range;
- **forming bridges**;
- raiding specific areas for food;
- building and protecting nest;
- **sorting brood and food items;**
- **cooperating in carrying large items**;
- emigration of a colony;
- **finding shortest route from nest to food source**;
- preferentially exploiting the richest food source available.

The ACO algorithm is inspired by this:

A key concept: Stigmergy

Stigmergy is:

indirect communication via interaction with the environment.

- A problem gets solved bit by bit ..
- Individuals communicate with each other in the above way, affecting what each other does on the task.
- Individuals 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 …
- E.g. as we will see, this is how ants find shortest paths.

Stigmergy in Ants

Ants are behaviorally unsophisticated, but collectively they can perform complex tasks.

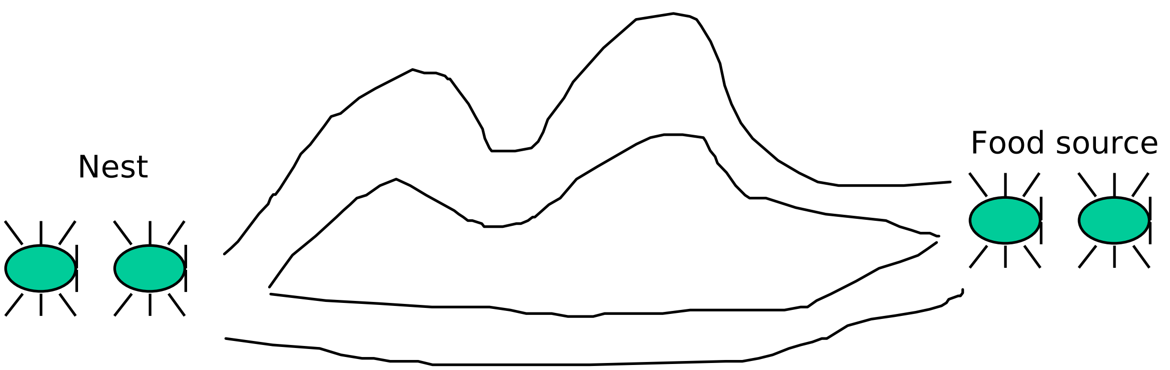
Ants have *highly developed sophisticated* *sign-based stigmergy*

- They communicate using pheromones;
- They **lay trails of pheromone** that can be followed by other ants.

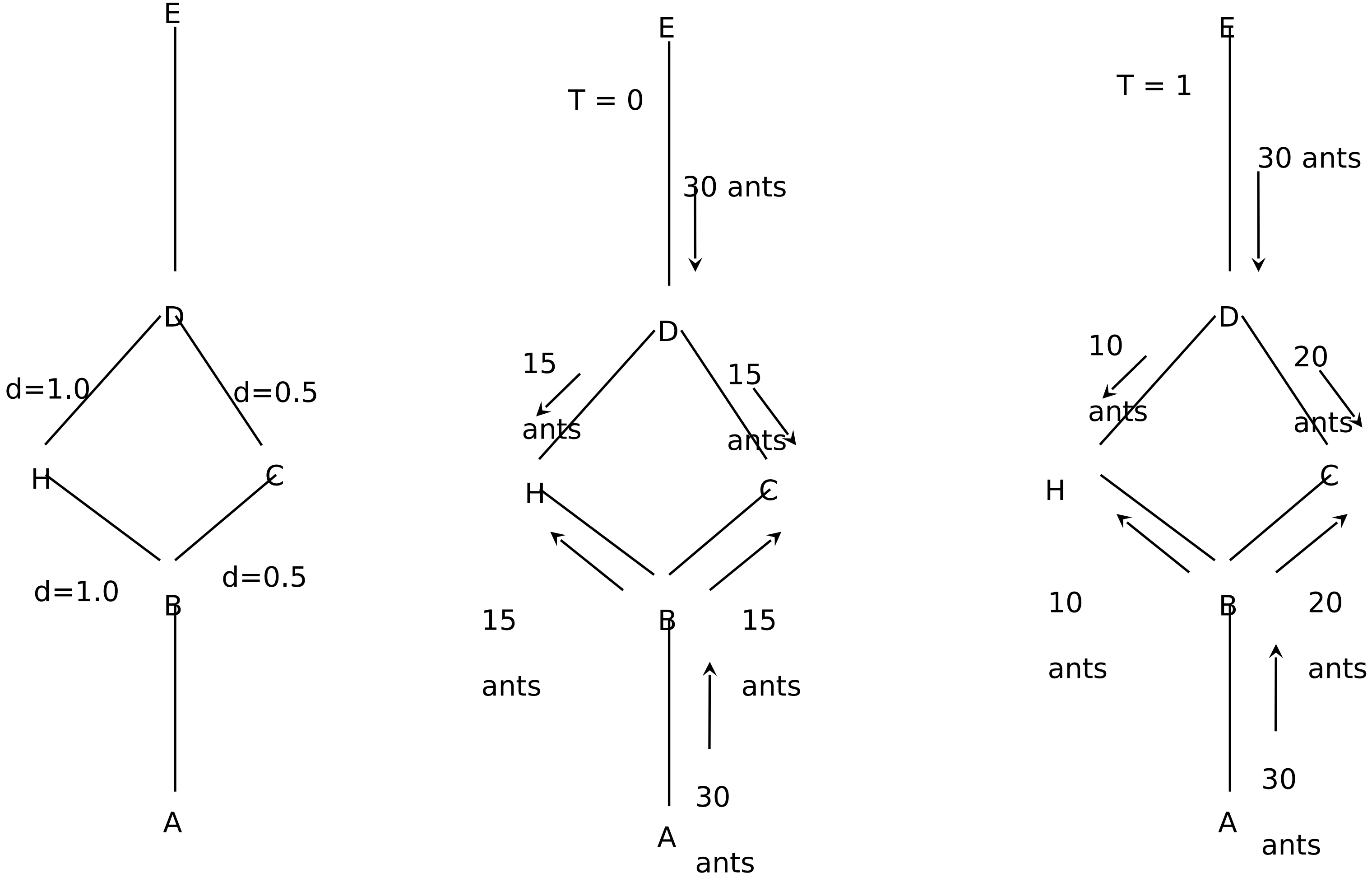
- If an ant has a **choice of two pheromone trails** to follow, one to the NW, one to the NE, but the NW one is *stronger* – which one will it follow?

Pheromone Trails

- Individual ants lay pheromone trails while travelling from the nest, to the nest or possibly in both directions.
- The pheromone trail gradually evaporates over time.
- But pheromone trail strength accumulate with multiple ants using path.



Pheromone Trails continued



Ant Colony Optimisation Algorithms: Basic Ideas

Ants are *agents* that:

- Move along between nodes in a graph.
- They choose where to go based on pheromone strength (and maybe other things)
- An ant's path represents a specific candidate solution.
- When an ant has finished a solution, pheromone is laid on its path, according to quality of solution.
- This pheromone trail affects behaviour of other ants by `stigmergy' …

The ACO algorithm for the TSP
[a simplified version with all essential details]

We have a TSP, with n cities.

1. We place some ants at each city. Each ant then does this:
   - It makes a complete tour of the cities, coming back to its starting city, using a *transition rule* to decide which links to follow. By this rule, it chooses each next-city at random, but biased partly by the pheromone levels existing at each path, and biased partly by *heuristic information*.
2. When all ants have completed their tours.

   Global Pheromone Updating occurs.
   - The current pheromone levels on all links are reduced (I.e. pheromone levels decay over time).
   - Pheromone is lain (belatedly) by each ant as follows: it places pheromone on all links of its tour, with strength depending on how good the tour was.

Then we go back to 1 and repeat the whole process many times, until we reach a termination criterion.

A <u>very common variation</u>, which gives the best results

We have a TSP, with *n* cities.

1. We place some ants at each city. Each ant then does this:
   - It makes a complete tour of the cities, coming back to its starting city, using a *transition rule* to decide which links to follow. By this rule, it chooses each next-city at random, but biased partly by the pheromone levels existing at each path, and biased partly by *heuristic information*.

2. When all ants have completed their tours.

Apply some iterations of LOCAL SEARCH to the completed tour; this finds a better solution, which is now treated as the ant's path. Then continue the next steps as normal.

Global Pheromone Updating occurs.

- The current pheromone levels on all links are reduced (I.e. pheromone levels decay over time).
- Pheromone is lain (belatedly) by each ant as follows: it places pheromone on all links of its tour, with strength depending on how good the tour was.

Then we go back to 1 and repeat the whole process many times, until we reach a termination criterion.

The transition rule

T(*r*,s) is the amount of pheromone currently on the path that goes directly from city *r* to city *s*.

H(*r*,*s*) is the heuristic value of this link – in the classic TSP application, this is chosen to be 1/distance(*r*,*s*) -- I.e. the shorter the distance, the higher the heuristic value.

$p_k(r,s)$ is the probability that ant *k* will choose the link that goes from *r* to *s*

β is a parameter that we can call the *heuristic strength*

The rule is:

$$p_k(r,s) = \frac{T(r,s) \cdot H(r,s)^{\beta}}{\sum_{\text{unvisited cities } c} T(r,c) \cdot H(r,c)^{\beta}}$$

Where our ant is at city *r*, and *s* is a city as yet unvisited on its tour, and the summation is over all of *k*'s unvisited cities.

Global pheromone update

$A_k(r,s)$	is amount of pheromone added to the (r, s) link by ant k.
m	is the number of ants
ρ	is a parameter called the pheromone decay rate.
L_k	is the length of the tour completed by ant k

$T(r, s)$ at the next iteration becomes: $\rho \cdot T(r,s) + \sum_{k=1}^{m} A_k(r,s)$

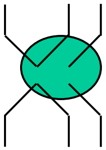
Where $A_k(r,s) = 1/L_k$

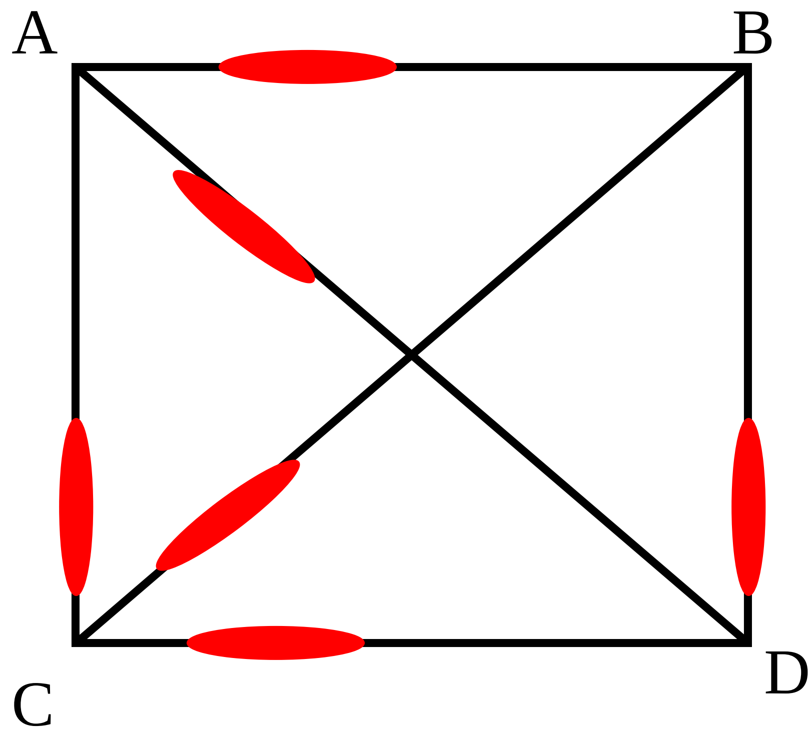
Study these – they're not that hard.

How do you think the parameters *m*, *beta*, *rho* etc … affect the search?

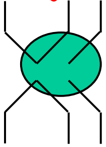
E.g. A 4-city TSP

Initially, random levels of pheromone are scattered on the edges





Pheromone

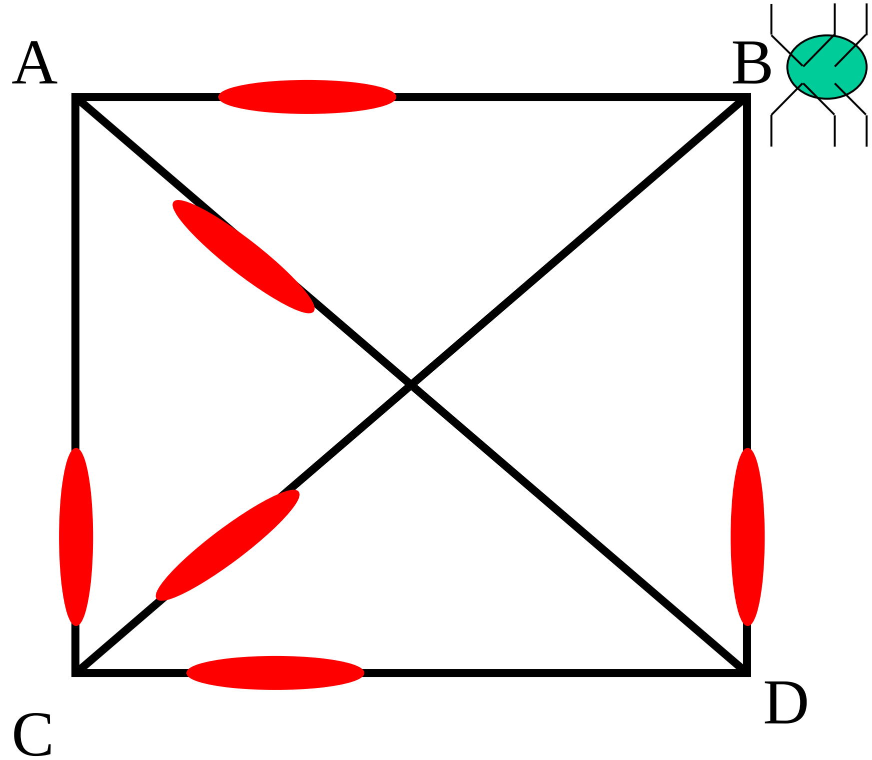


Ant

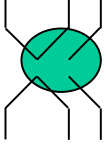
AB: 10, AC: 10, AD, 30, BC, 40, CD 20

E.g. A 4-city TSP

An ant is placed at a random node



Pheromone



Ant

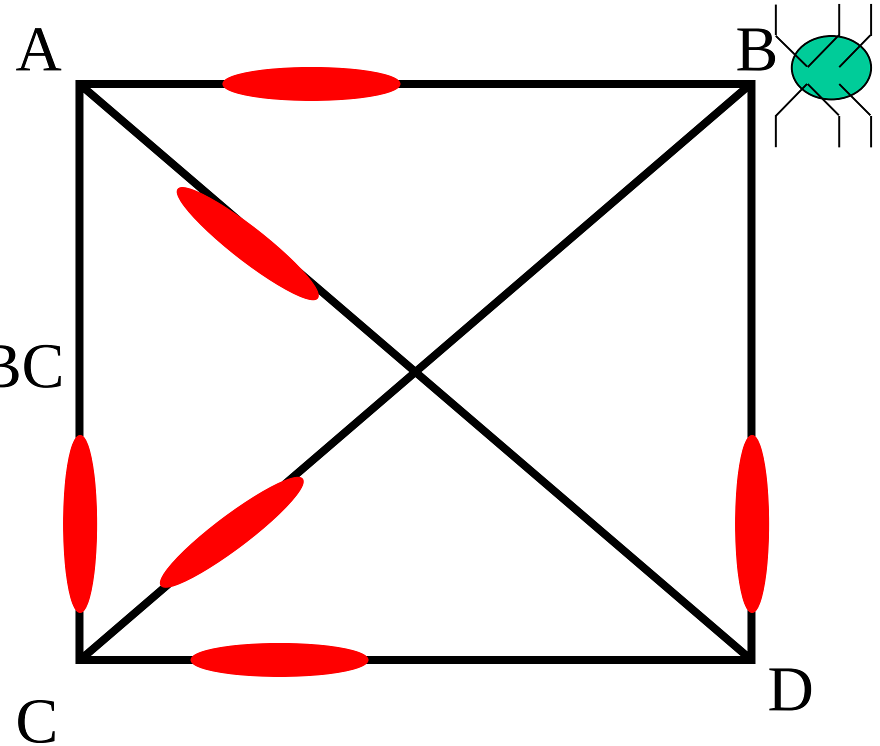
AB: 10,   AC: 10,   AD, 30,   BC, 40,   CD 20

E.g. A 4-city TSP

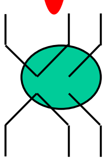
The ant decides where to go from that node, based on probabilities calculated from:

- pheromone strengths,
- next-hop distances.

Suppose this one chooses BC



Pheromone



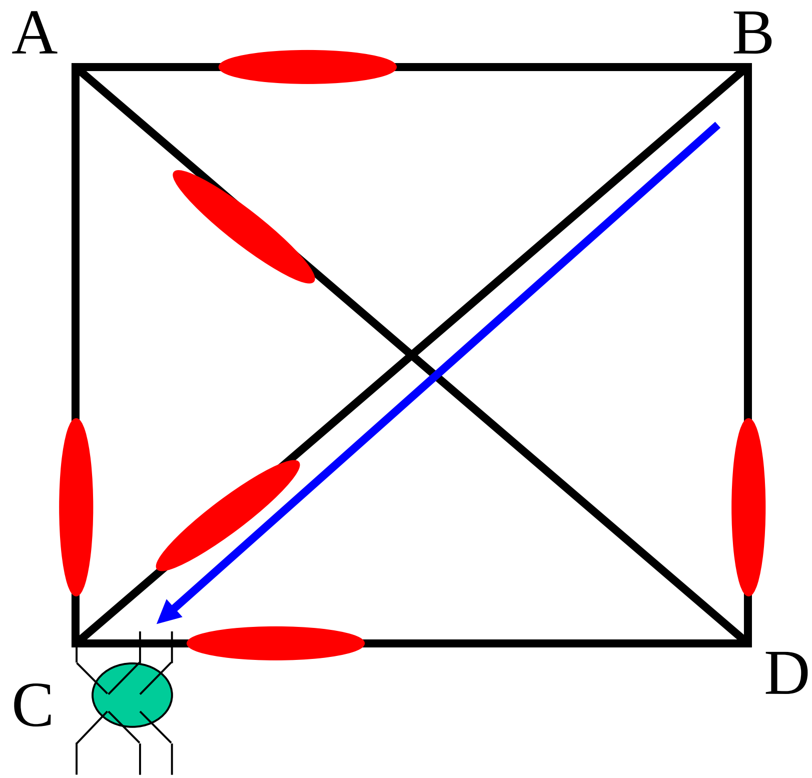
Ant

AB: 10,   AC: 10,   AD, 30,   BC, 40,   CD 20

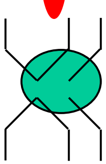
E.g. A 4-city TSP

The ant is now at C, and has a `tour memory' = {B, C} – so he cannot visit B or C again.

Again, he decides next hop (from those allowed) based on pheromone strength and distance; suppose he chooses CD



Pheromone

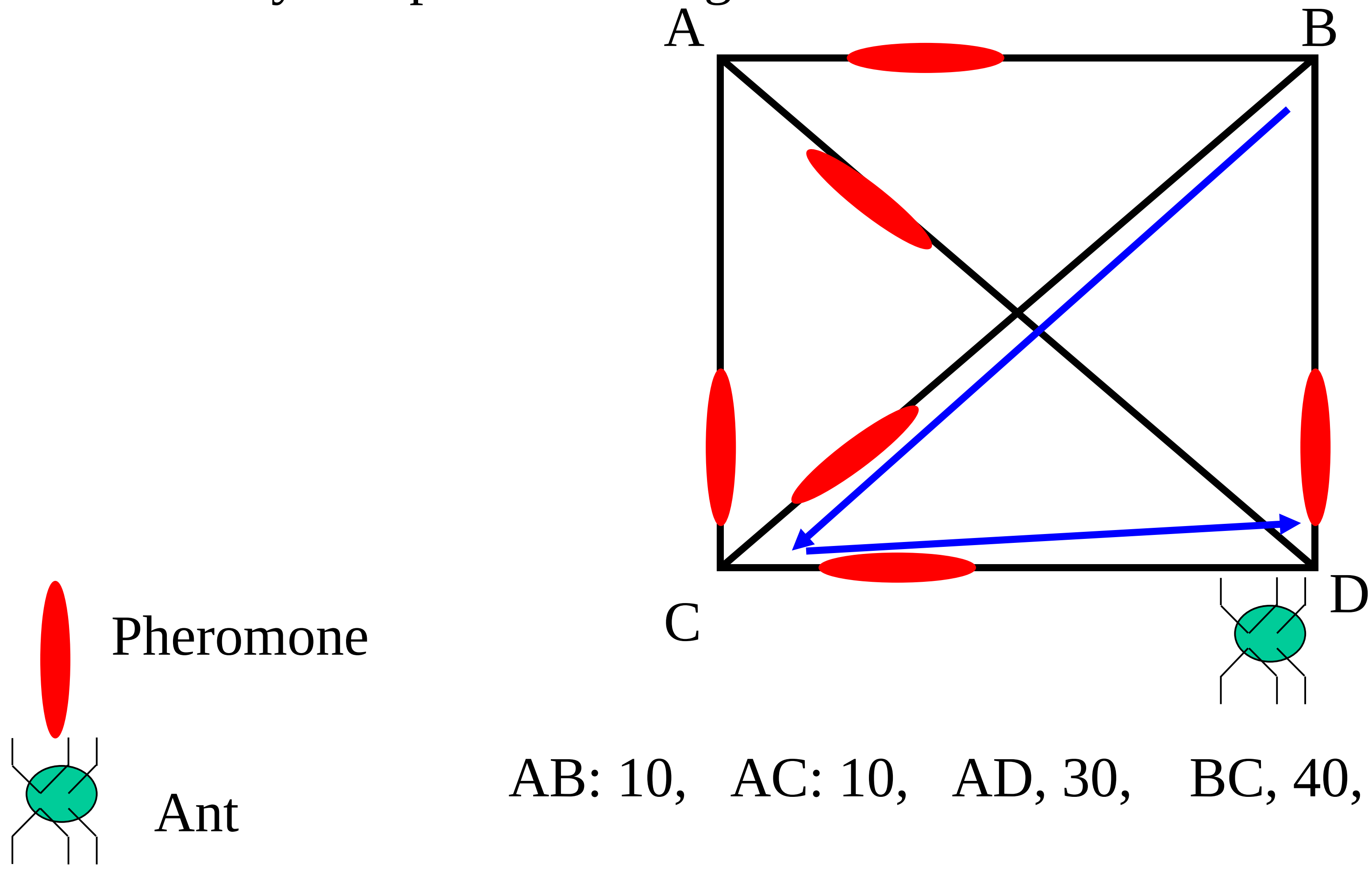


Ant

AB: 10, AC: 10, AD, 30, BC, 40, CD 20

E.g. A 4-city TSP

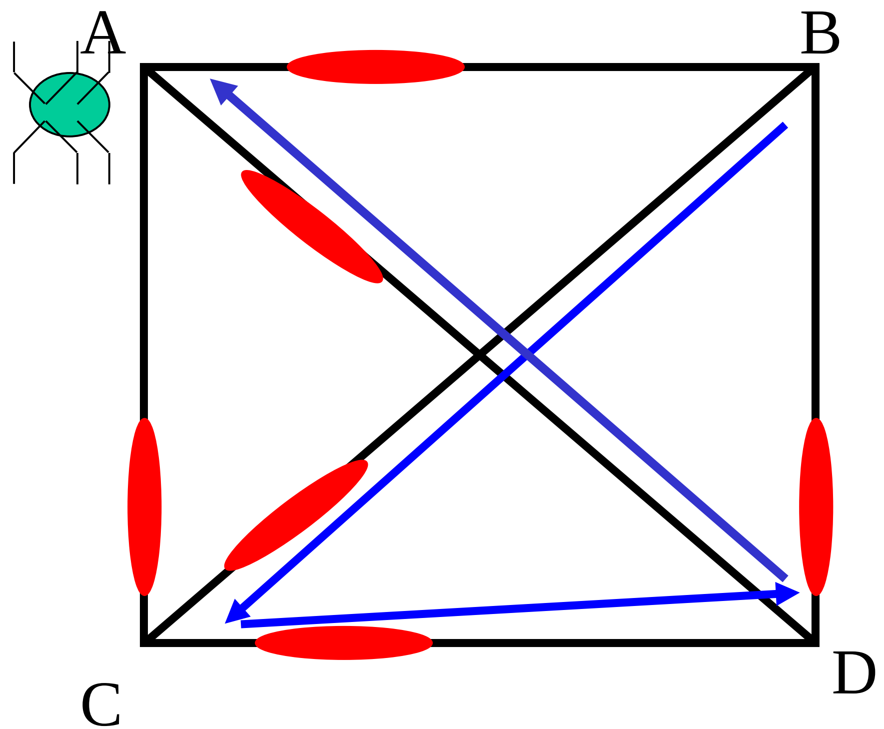
The ant is now at D, and has a `tour memory' = {B, C, D}
There is only one place he can go now:



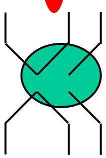
AB: 10,   AC: 10,   AD, 30,   BC, 40,   CD 20

E.g. A 4-city TSP

So, he has nearly finished his tour, having gone over the links: BC, CD, and DA.



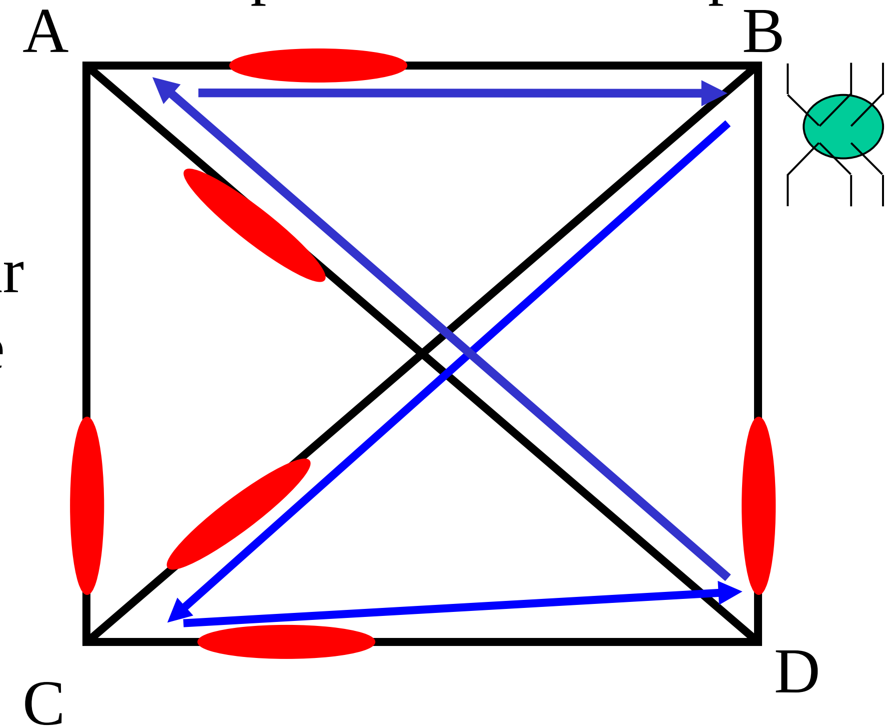

Pheromone


Ant

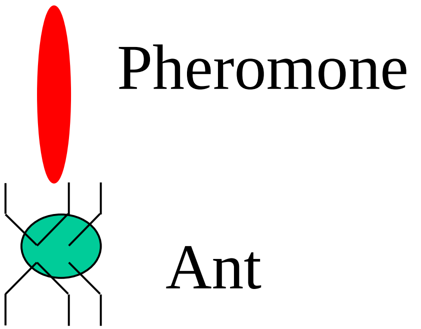
AB: 10,   AC: 10,   AD, 30,   BC, 40,   CD 20

E.g. A 4-city TSP

So, he has nearly finished his tour, having gone over the links: BC, CD, and DA. AB is added to complete the round trip.



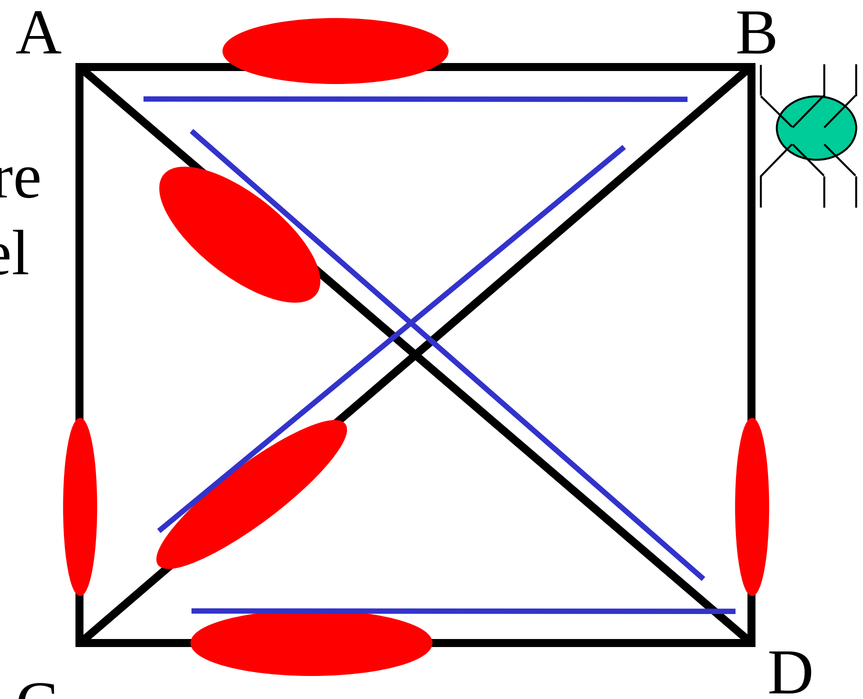
Now, pheromone on the tour is increased, in line with the fitness of that tour.



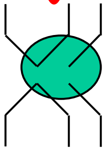
AB: 10, AC: 10, AD, 30, BC, 40, CD 20

E.g. A 4-city TSP

Next, pheromone everywhere is decreased a little, to model decay of trail strength over time



Pheromone



Ant

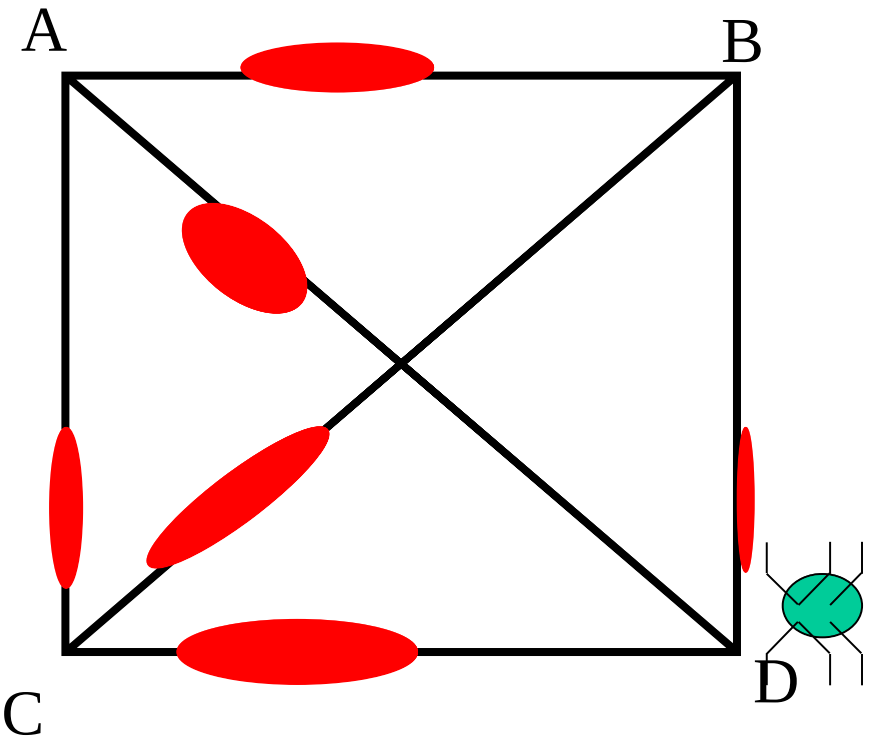
AB: 10, AC: 10, AD, 30, BC, 40, CD 20

E.g. A 4-city TSP

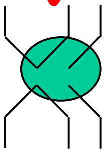
We start again, with another ant in a random position.

Where will he go?

Next , the actual algorithm and variants.



Pheromone



Ant

AB: 10, AC: 10, AD, 30, BC, 40, CD 20

Not just for TSP of course

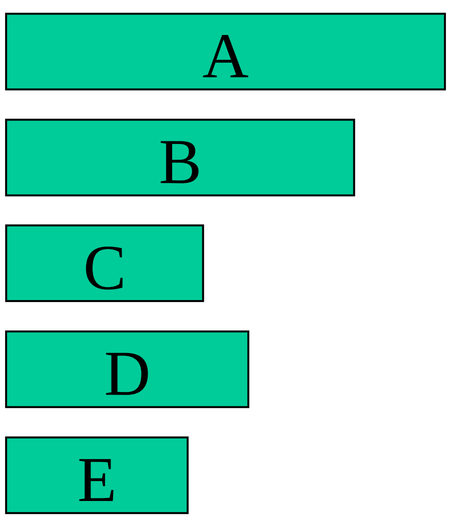
ACO is naturally applicable to any sequencing problem, or indeed *any* problem

All you need is some way to represent solutions to the problem as paths in a network.

E.g.

Single machine scheduling with due-dates

These jobs have to be done; their length represents the time they will take.

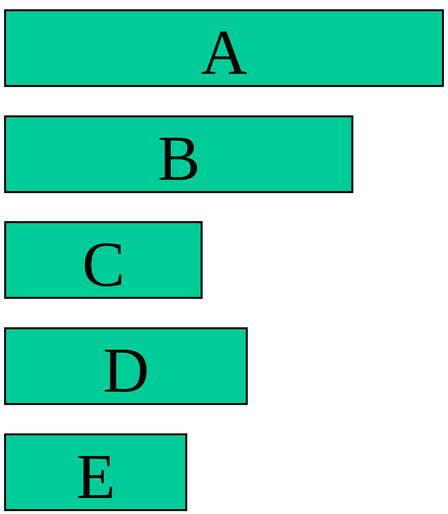


E.g.

Single machine scheduling with due-dates

These jobs have to be done; their length represents the time they will take.

Each has a `due date', when it needs to be finished

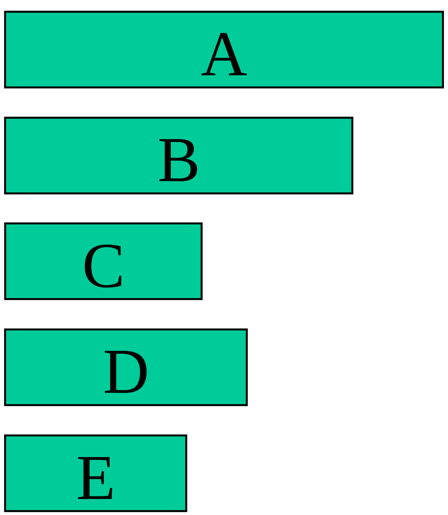


3pm
3:30pm
5pm
4pm
4:30pm

E.g.
Single machine scheduling with due-dates

These jobs have to be done; their length represents the time they will take.

Each has a `due date', when it needs to be finished

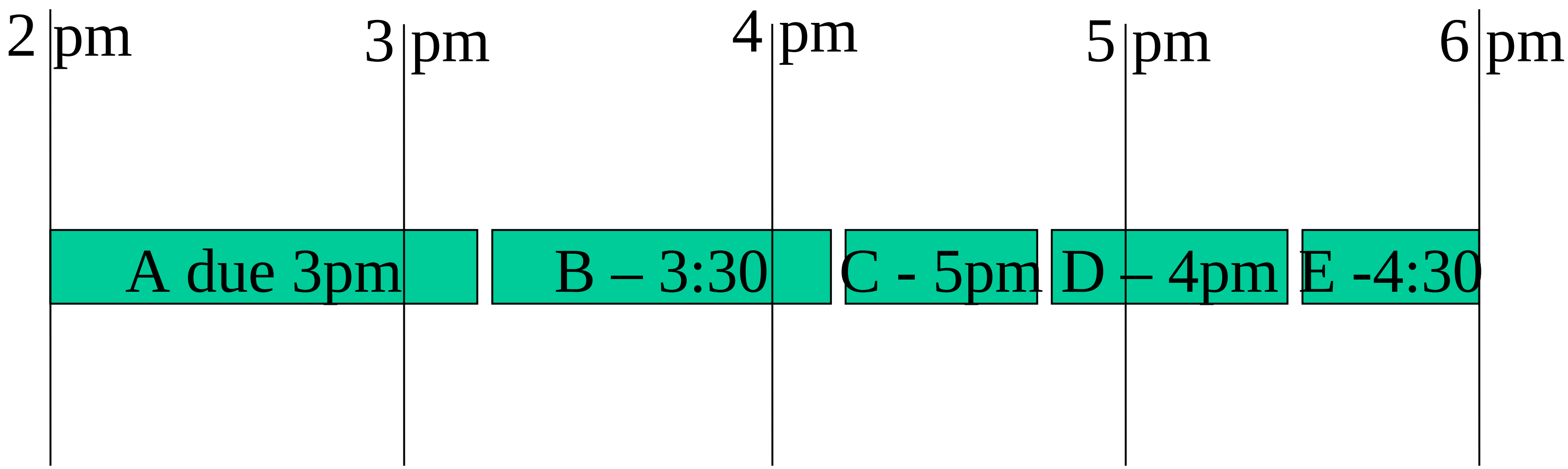


3pm
3:30pm
5pm
4pm
4:30pm

Only one `machine' is available to process these jobs, so can do just one at a time.

[e.g. machine might be human tailor, photocopier, Hubble Space Telescope, Etc …]

An example schedule

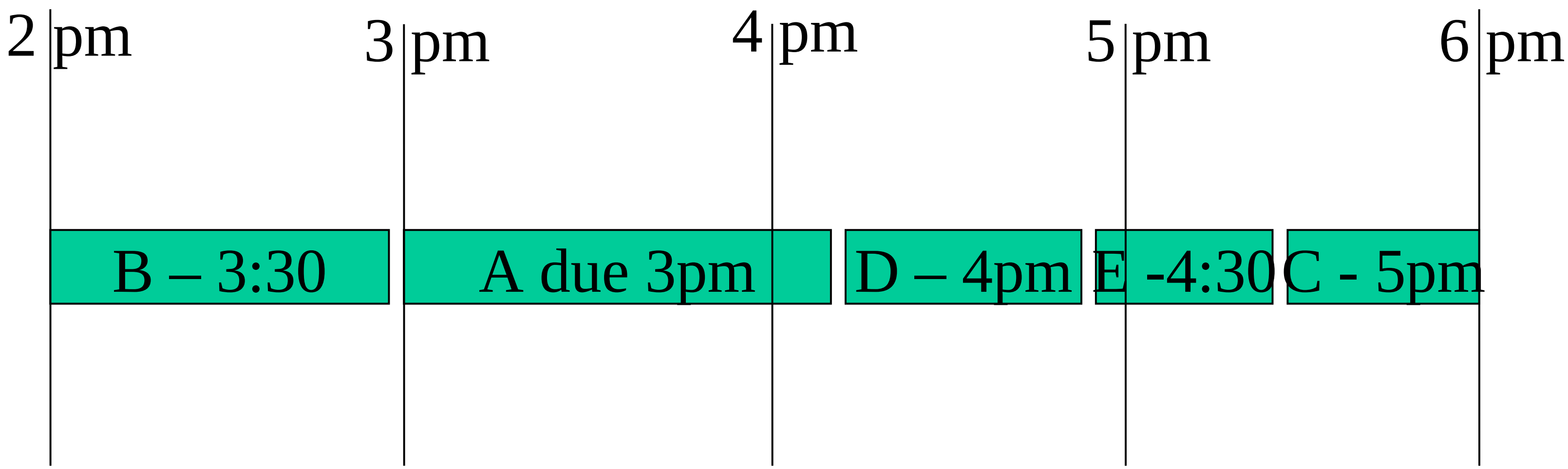


A is 10min late
B is 40min late
C is 20min early (lateness = 0)
D is 90min late
E is 90min late

Fitness might be average lateness; in this case 46min

or fitness could be Max lateness, in this case 90min

Another schedule



A is 70min late
B is 30min early (0 lateness)
C is 60min late
D is 50min late
E is 50min late

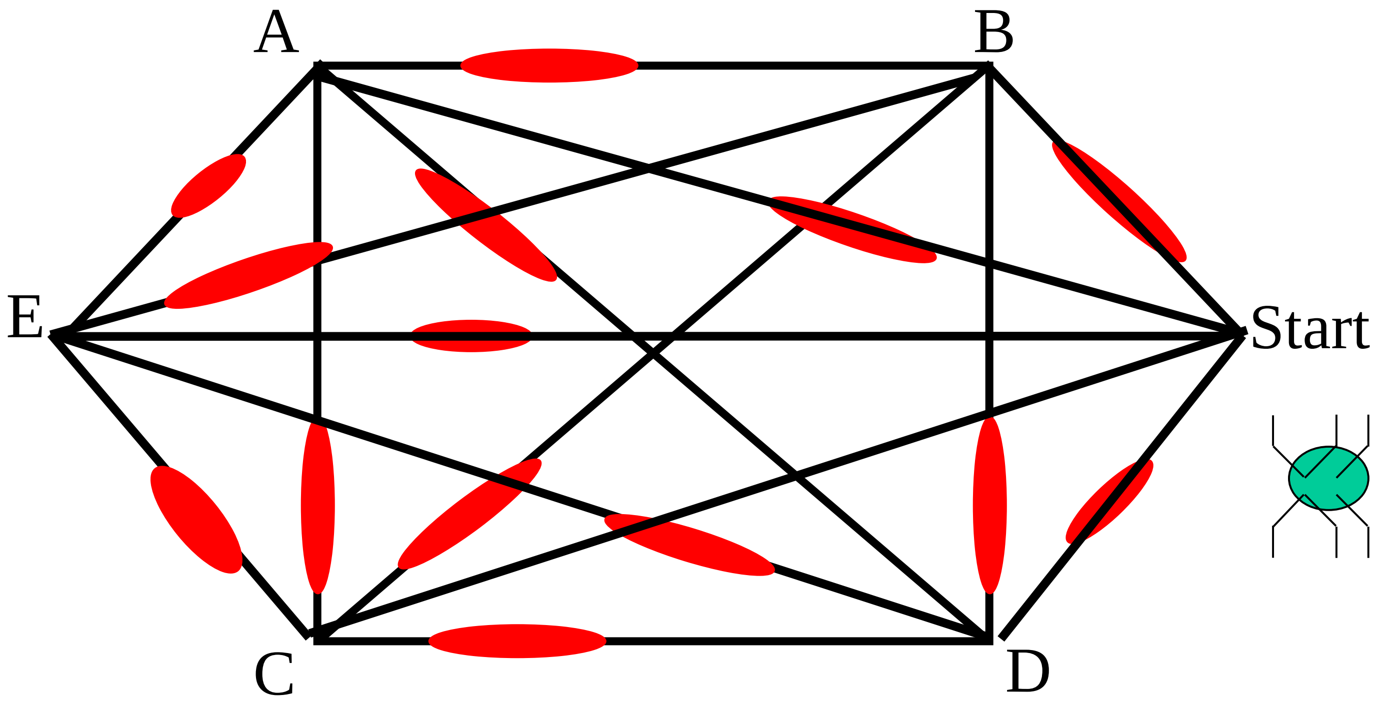
Fitness might be average lateness; in this case again 46min

or fitness could be Max lateness, in this case 70min

Applying ACO to this problem

Just like with the TSP, each ant finds paths in a network, where, in this case, each job is a node. Also, no need to return to start node – path is complete when every node is visited.

Initially, random levels of pheromone are scattered on the edges, an ant starts at a *Start* node (so the first link it chooses defines the first task to schedule on the machine); as before it uses a transition rule to take one step at a time, biased by pheromone levels, and also a heuristic score, each time choosing the next machine to schedule. What heuristic might you use in this case?



Example table from a research paper comparing ACO with

Table 3 Comparison of the performance of the ACO algorithm with the methods reported in Tan *et al*:[7] branch-and-bound (B&B), genetic algorithm (GA), simulated annealing (SA), and the RSPI local improvement method

						GA (Rubin and Ragatz[16])			*SA (Tan and Narasimhan[27])*			*RSPI (Rubin and Ragatz[16])*				*ACO*			
						% to B&B			*% to B&B*			*% to B&B*				*% to B&B*			
Problem	*# Jobs*	*PTV*	*TF*	*DDK*	*B&B*	*Best*	*Median*	*Worst*	*Best*	*Median*	*Worst*	*Best*	*Medium*	*Worst*	*Average run time (s)*†	*Best*	*Medium*	*Worst*	*Average run time (s)*‡
Prob401	15	L	L	N	90*	0.0	4.4	4.4	0.0	3.3	7.8	0.0	0.0++	3.3++	360	0.0	4.4	7.8	11.80
Prob402	15	L	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	360	0.0	0.0	0.0	0.35
Prob403	15	L	M	N	3418*	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.0++	0.2++	360	0.0	1.1	2.1	13.50
Prob404	15	L	M	W	1067*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	12.7	360	0.0	0.0	0.0+	11.05
Prob405	15	H	L	N	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	360	0.0	0.0	0.0	0.35
Prob406	15	H	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	360	0.0	0.0	0.0	0.30
Prob407	15	H	M	N	1861*	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.4++	360	0.0	0.0	1.1	13.75
Prob408	15	H	M	W	5660*	0.0	0.0	0.9	0.0	0.0	0.6	0.0	0.0++	0.9++	360	0.0	1.1	1.5	13.20
Prob501	25	L	L	N	264	0.0	1.5	3.8	0.8	1.9	4.2	0.8	0.8	1.5++	960	−1.1+	0.8	1.9	85.90
Prob502	25	L	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	960	0.0	0.0	0.0	1.70
Prob503	25	L	M	N	3511	−0.4	0.2	0.9	−10.4	−9.9	−9.6	−0.4	−0.4++	−0.4++	960	−0.4	0.3	0.9	88.65
Prob504	25	L	M	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	960	0.0	0.0	0.0	1.50
Prob505	25	H	L	N	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	–#	960	0.0	0.0	0.0+	1.60
Prob506	25	H	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	960	0.0	0.0	0.0	1.60
Prob507	25	H	M	N	7225	2.1	6.1	9.6	0.0	1.1	2.4	0.0++	0.1++	0.2++	960	0.7	1.8	3.7	126.90
Prob508	25	H	M	W	2067	−5.9	−5.9	−1.5	−7.4	−7.4	−3.1	−7.4++	−7.4++	−7.4++	960	−5.9	5.9	14.2	102.90
Prob601	35	L	L	N	30	76.7	150.0	193.3	20.0	43.3	96.7	20.0	31.7	46.7	1800	−46.7+	−13.3+	6.7+	386.50
Prob602	35	L	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1800	0.0	0.0	0.0	3.75
Prob603	35	L	M	N	17 774	−0.7	0.4	2.2	0.1	0.8	1.4	0.1	0.2	0.7	1800	−0.5+	0.1+	0.3+	776.65
Prob604	35	L	M	W	19 277	0.2	1.0	2.6	−0.2	0.7	2.8	−0.2	−0.1++	0.1++	1800	−0.8+	0.3	1.3	382.05
Prob605	35	H	L	N	291	13.7	37.3	56.7	−6.2	−1.2	8.9	−6.2	−4.1	−2.1++	1800	−15.1	−8.8+	−1.0	413.10
Prob606	35	H	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1800	0.0	0.0	0.0	3.65
Prob607	35	H	M	N	13 274	5.0	6.6	7.6	−1.7	−0.1	1.7	−1.7++	−0.8++	−0.2++	1800	−1.4	−0.1	0.6	715.45
Prob608	35	H	M	W	6704	−29.0	−28.6	−26.7	−29.4	−29.0	−26.9	−29.4	−29.2++	−29.0++	1800	−29.4	−24.8	−20.1	880.35
Prob701	45	L	L	N	116	57.8	82.8	118.1	1.7	29.3	40.5	1.7	22.4	28.4	3600	−11.2+	−5.6+	0.0+	1197.95
Prob702	45	L	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3600	0.0	0.0	0.0	11.05
Prob703	45	L	M	N	27 097	−0.1	1.5	2.5	−1.3	0.2	1.3	−1.3	−0.2	0.1	3600	−1.6+	−1.0+	−0.5+	2638.25
Prob704	45	L	M	W	15 941	−2.4	−1.6	1.0	−3.3	−1.2	1.0	−3.3++	−2.7++	1.8++	3600	−2.8	−1.1	−0.3	1886.65
Prob705	45	H	L	N	234	53.4	89.3	114.5	8.5	20.5	49.1	8.5	18.8	23.1	3600	−5.1+	5.6+	15.4+	1168.85
Prob706	45	H	L	W	0*	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3600	0.0	0.0	0.0	9.60
Prob707	45	H	M	N	25 070	−1.1	1.8	6.3	−3.4	−2.5	−0.8	−3.4	−2.9	−2.6	3600	4.2+	−3.3+	−2.9+	2524.65
Prob708	45	H	M	W	24 123	2.8	7.2	10.1	−4.0	−3.2	−2.0	−4.0++	−3.9++	−3.3++	3600	−3.2	−1.7	−0.6	2336.10

Legend: B&B = branch-and-bound method; PTV = processing time variance; TF = tardiness factor; DDR = due date range; L = low; H = high; M = moderate; N = narrow; W = wide.

*Indicates optimum solution.

+ACO is better than RSPI.

++RSPI is better than ACO.

#Divides by 0, the worst solution is 6.0.

†Pentium 90 MHz personal computer. ‡Pentium 100 MHz personal computer.

ACO is a thriving and maturing research area – it has its own conferences. It gets very good results on some difficult problems. Following the above link will help you find examples.

ACO research and practice tends to concentrate on:

- hybridisation with other methods; e.g. it is common to improve an individual ant's solution by local search, and then lay pheromone.
- New and adaptive ways to control the relative influence of heuristics, pheromone strength and pheromone decay.