

# MIRWORKS

Machine Intelligence Research Group



## Nature Inspired Computing collective behaviour - ACO

Vinod Chandra S S

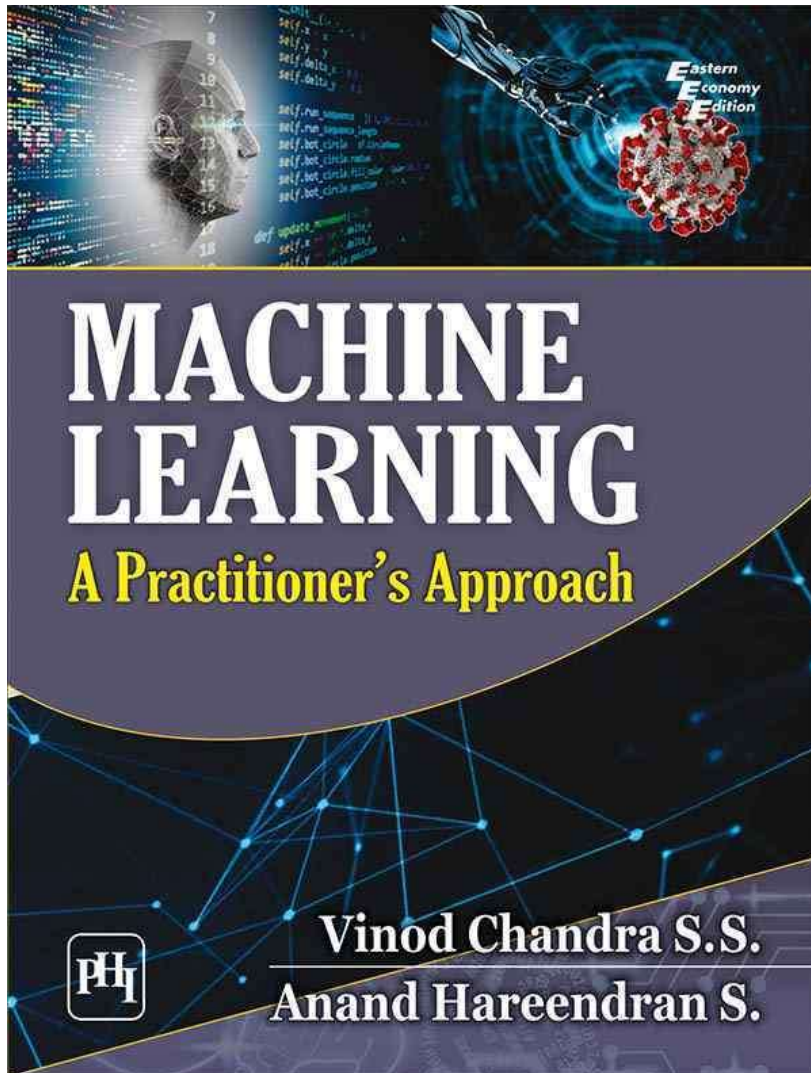
Machine Intelligence Research Laboratory

Department of Computer Science

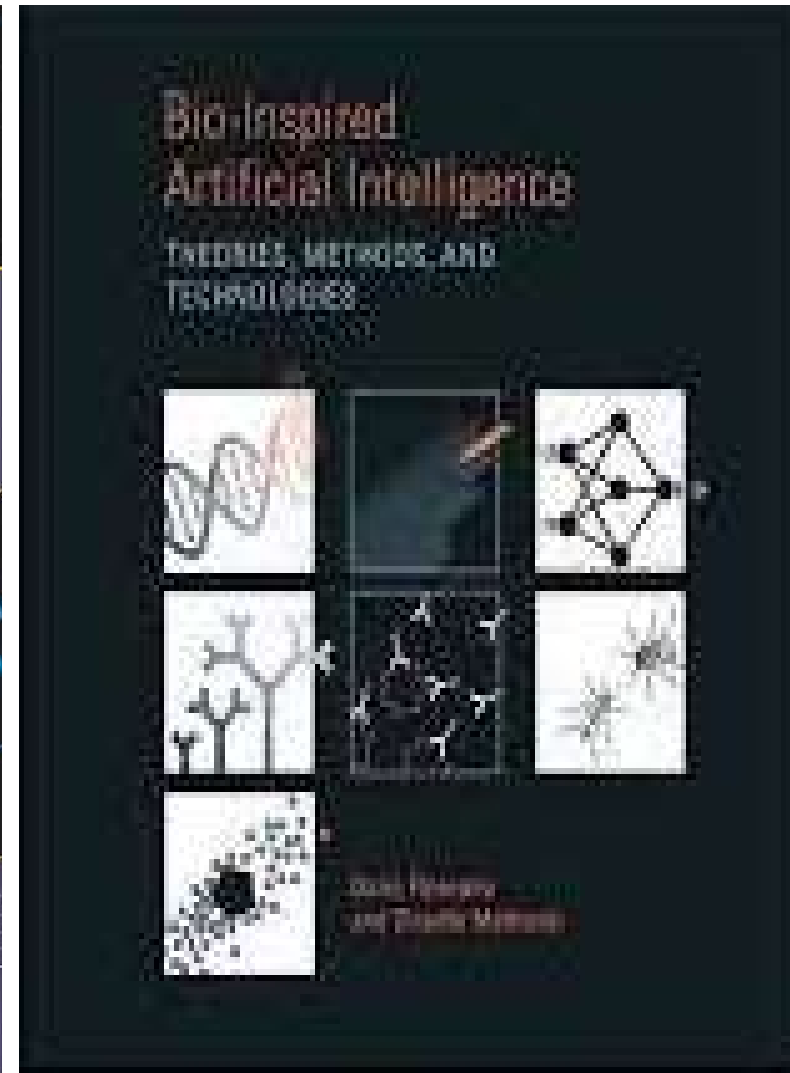
University of Kerala

<http://vinod.mirworks.in>

# Reference



Vinod Chandra S S



# Topic coverage

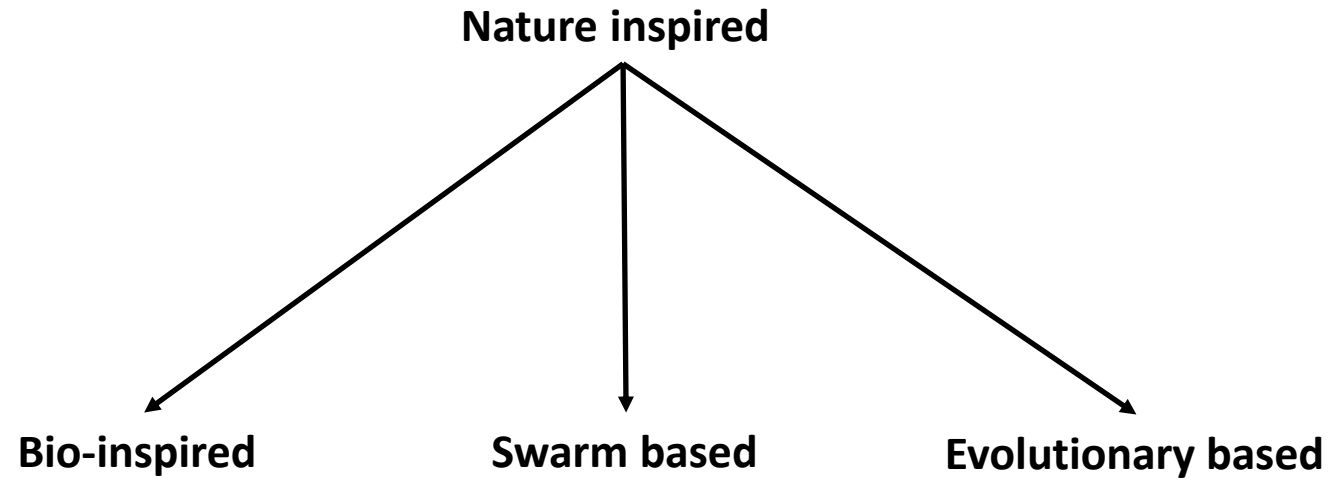
- Ant colony basics
- Hybrid ant system, ACO basics
- ACO in combinatorial optimization
- Variations of ACO, case studies, applications of ACO
- Particle Swarm algorithms - basics of particles moves
- Particle swarm optimization
- Variable length PSO and variants of PSO
- Applications of PSO, case studies
- Artificial Bee Colony algorithms - ABC basics
- ABC in optimization
- Multi-dimensional bee colony algorithms
- Applications of ACO, PSO, ABC algorithms

# Nature inspired algorithms

Nature provide some of the efficient ways to solve many practical optimization problems.

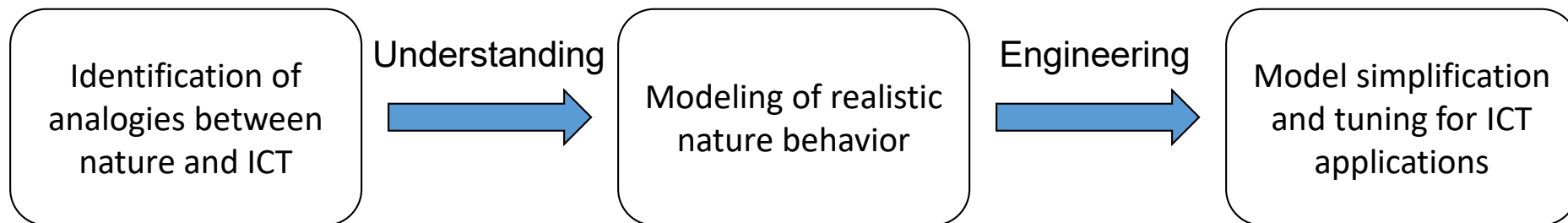
Algorithms imitating processes in nature/inspired from nature -  
**Nature inspired algorithms**

# Classification

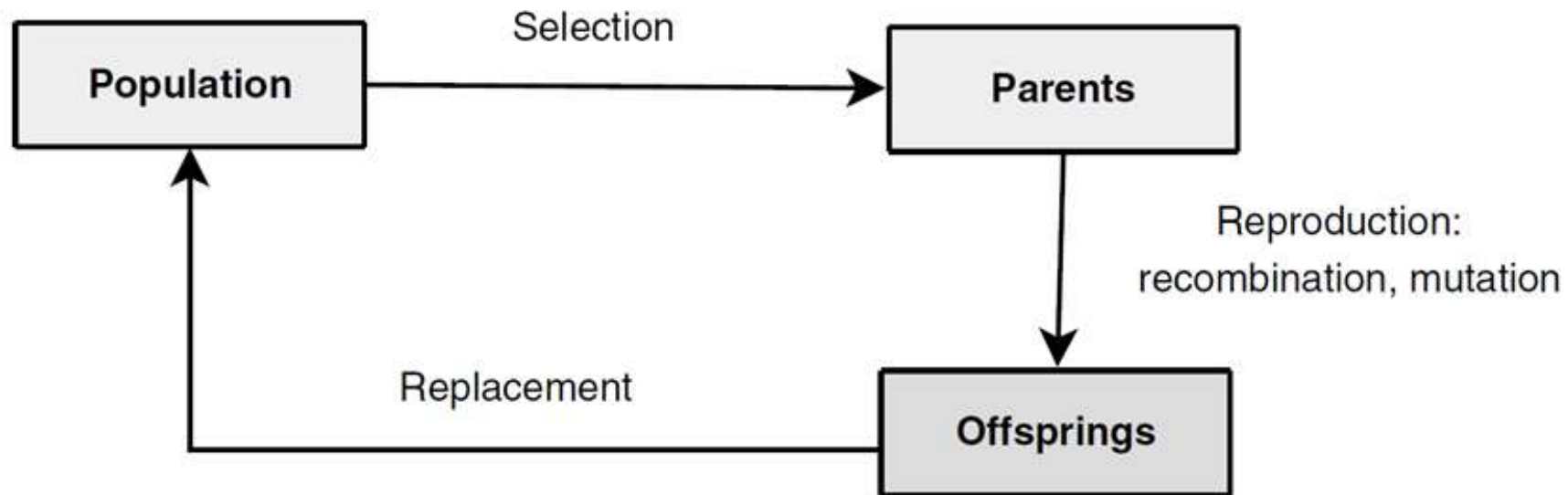


# Design of nature inspired solutions

- Identification of analogies
  - In swarm or molecular biology and IT systems
- Understanding
  - Computer modeling of realistic biological behavior
- Engineering
  - Model simplification and tuning for IT applications



# Evolutionary algorithms



A generation in evolutionary algorithm

# Basic steps

$P \leftarrow \text{Generate Initial Population}$

Evaluate ( $P$ )

while termination conditions not met do

$P' \leftarrow \text{Recompute}(P)$

$P'' \leftarrow \text{Mutate}(P')$

    Evaluate ( $P$ )

$P \leftarrow \text{Select}(P'' \cup P)$

endwhile



# Swarm intelligence

- Swarm Intelligent has two fundamental concepts:

## Self organizing:

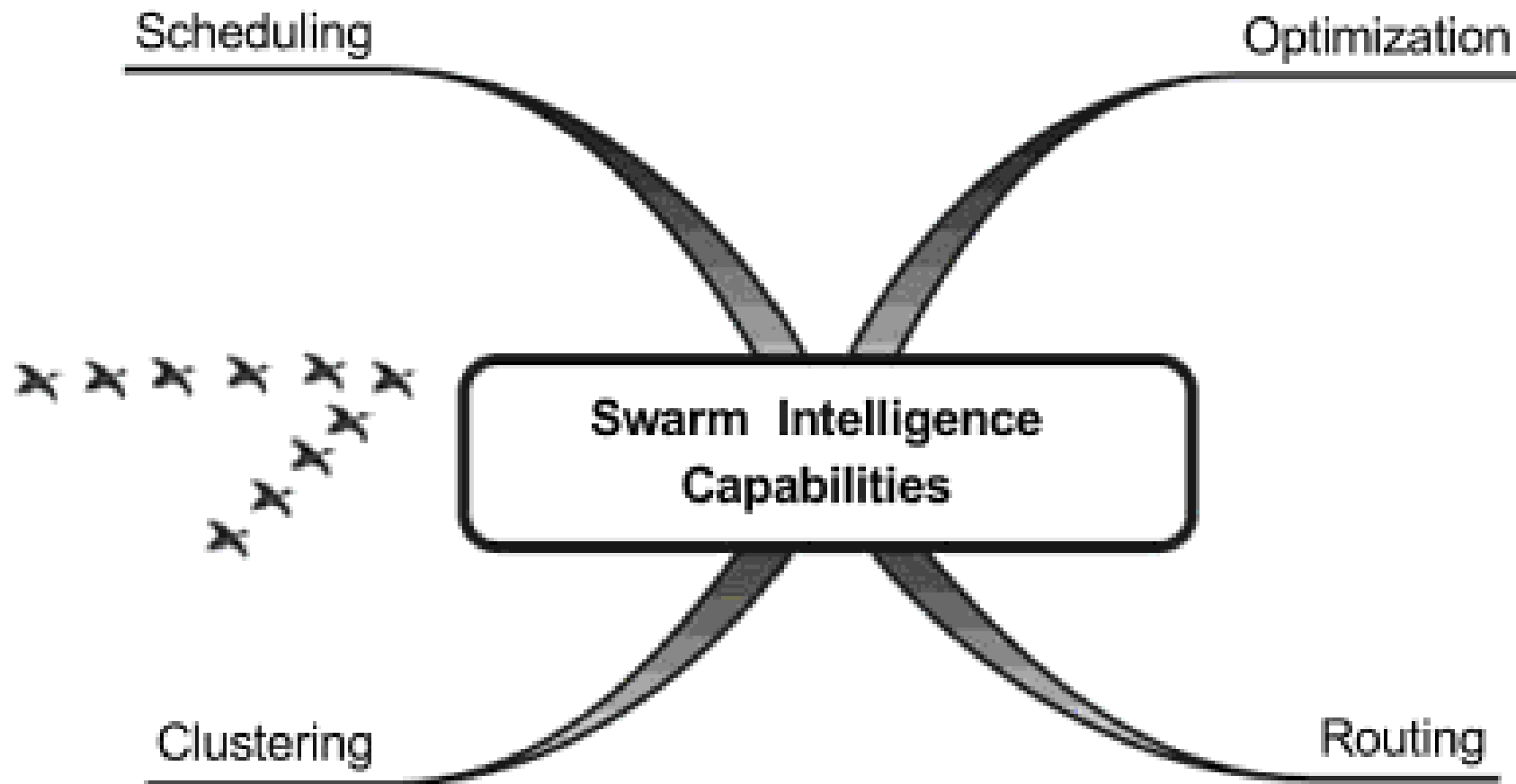
- Positive feedback
- Negative feedback
- Fluctuations
- Multiple interactions

## Division of labor:

- Simultaneous task performance by cooperating specialized individuals
- Enables the swarm to respond to changed conditions in the search space.



# Swarm intelligence



# Basic principles

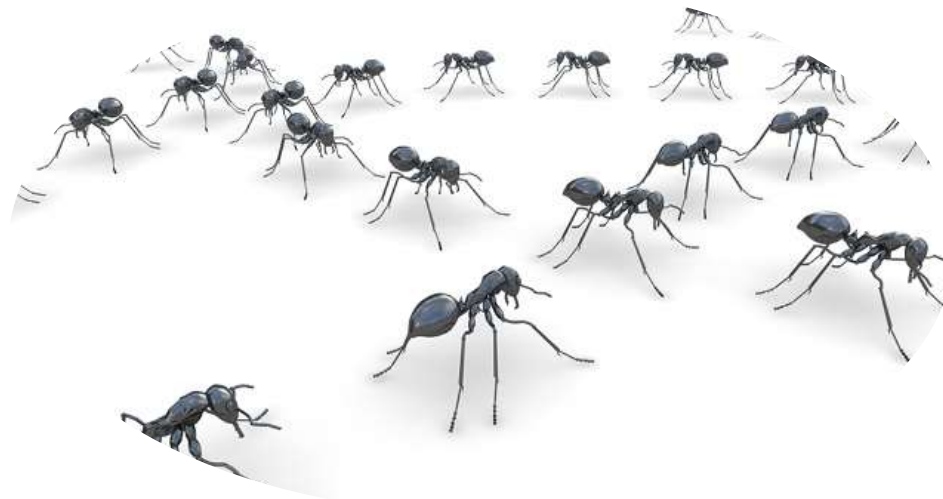
- **Proximity principle:** swarm is capable of doing elementary space and time computations related to its neighborhood
- **Quality Principle:** swarm should respond to quality factors of food and safety of a location
- **Principle of diverse response:** Resources should not be concentrated in a narrow region
- **Principle of stability:** population should not change its mode of behavior every time the environment changes
- **Principle of adaptability:** swarm is sensitive to the changes in the environment that result in different swarm behavior

# Properties

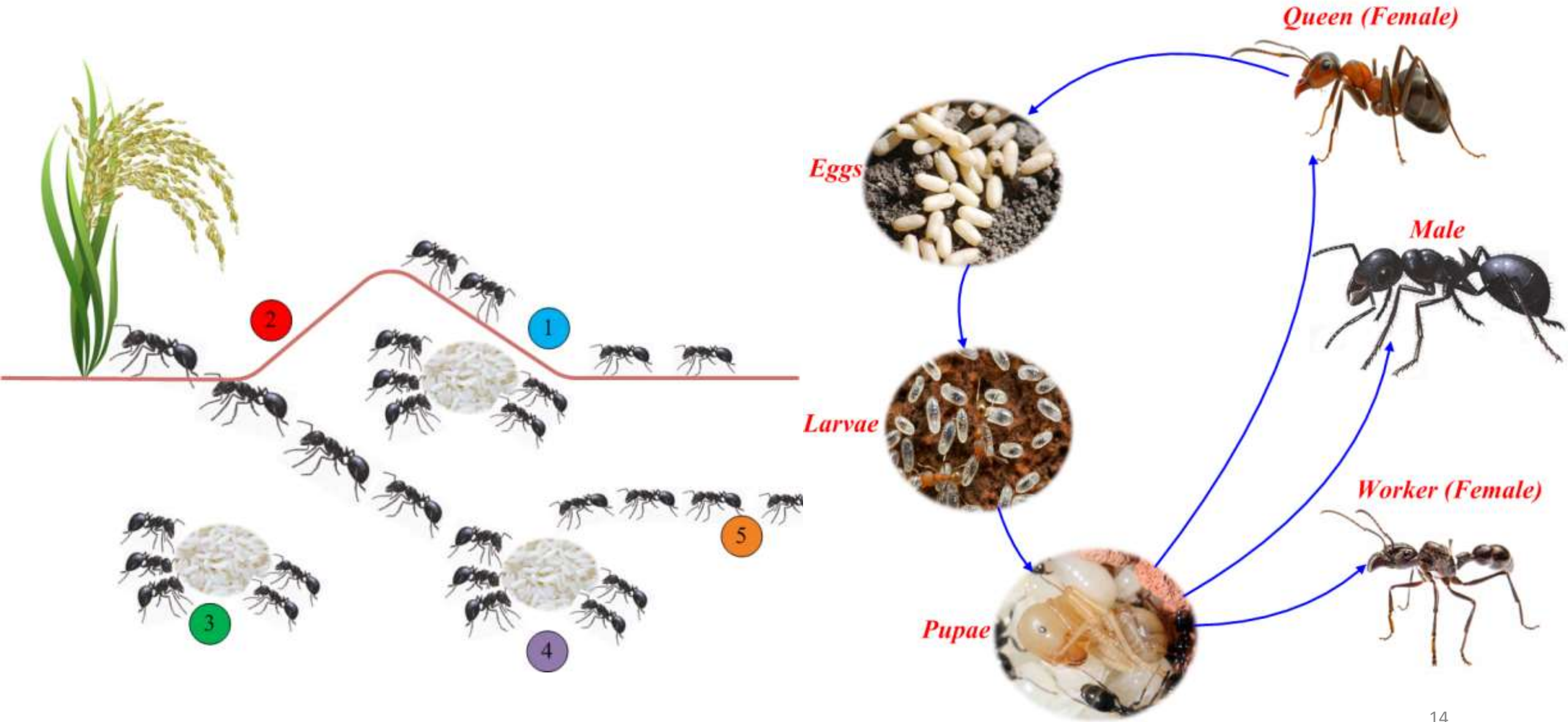
- **Scalable:** Intelligence arises due to the individual interaction of each agent with its neighbors
- **Parallel:** Individual members of the swarm can perform different functions at various locations at the same time
- **Fault Tolerance:** Self-organized and decentralized nature of swarms leads to fault tolerant systems as the agents are interchangeable and no one is in total control of the entire system
- **Adaptability:** Swarm can dynamically accommodate any kind of changes in the environment very promptly
- **Decentralized:** There is no single point of failure in a swarm subsystem as nobody is in full charge of the system

# Swarm Algorithms

## Ant Colony Algorithms

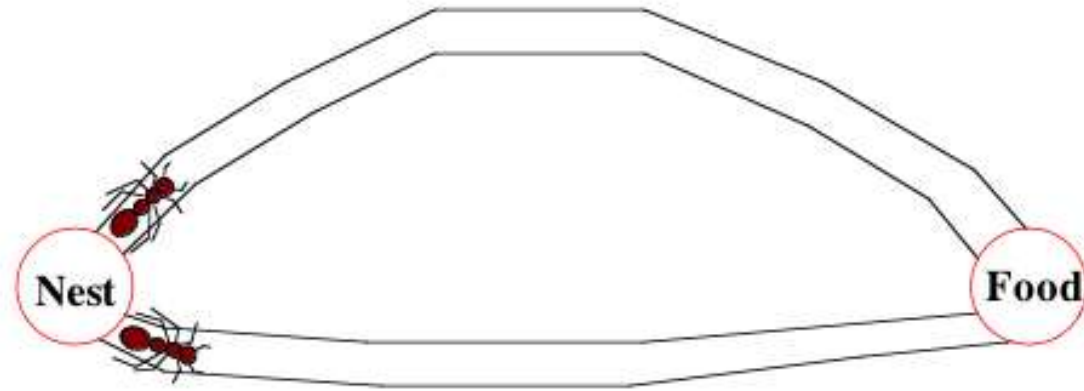


# Ant - biology



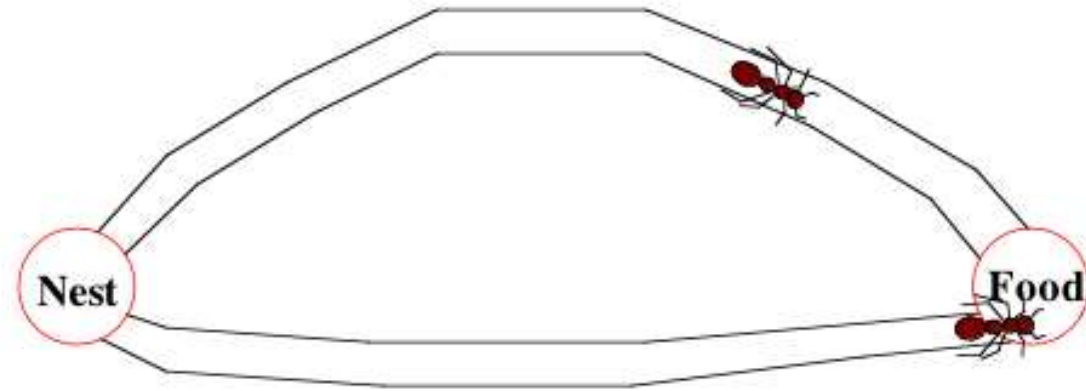
# Ant foraging

Cooperative search by pheromone trails



# Ant foraging

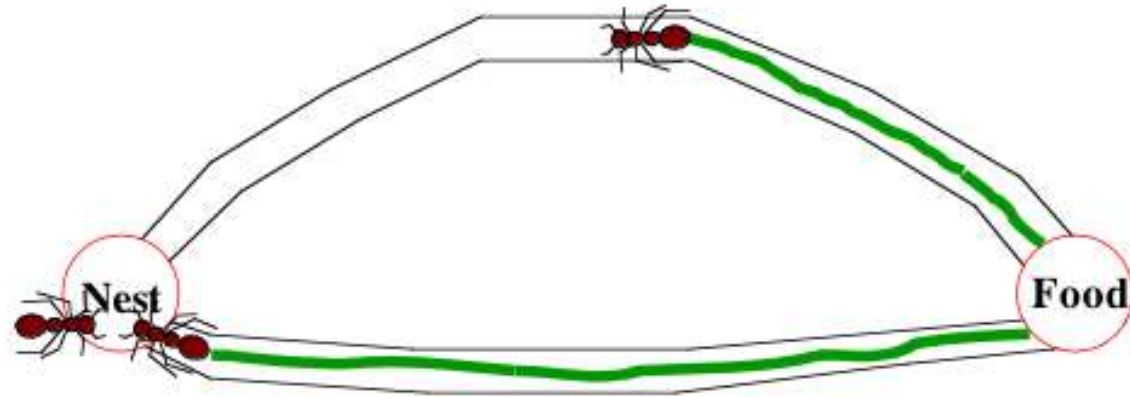
Cooperative search by pheromone trails





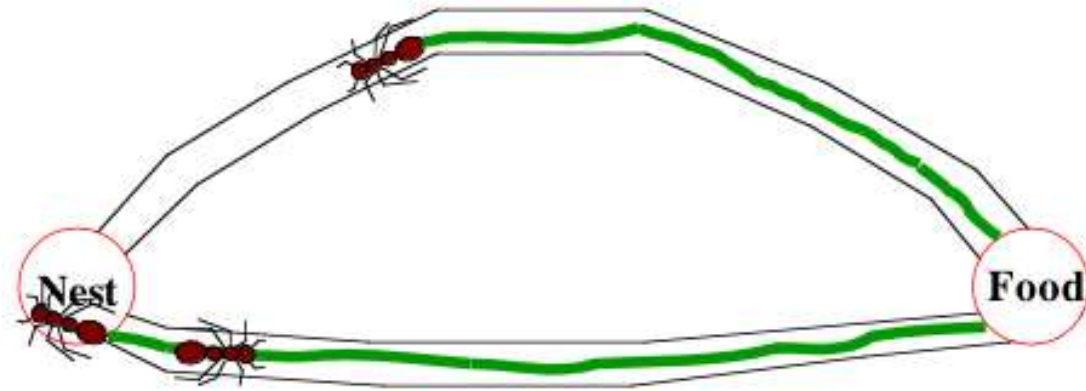
# Ant foraging

Cooperative search by pheromone trails



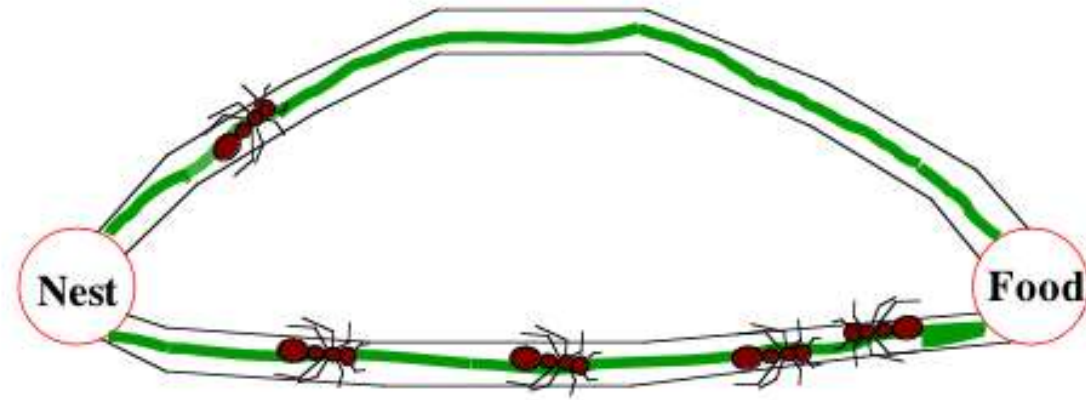
# Ant foraging

Cooperative search by pheromone trails



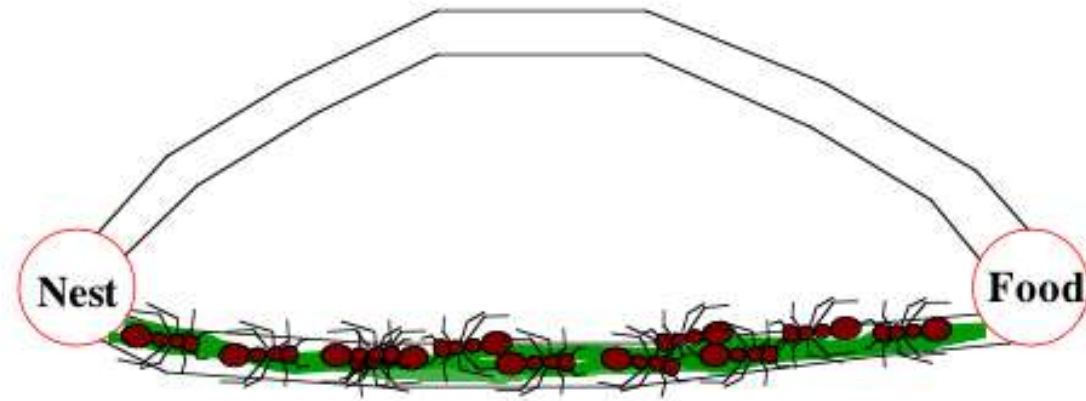
# Ant foraging

Cooperative search by pheromone trails



# Ant foraging

Cooperative search by pheromone trails



# Ants $\equiv$ Agents

- Stigmergy can be operational
  - Coordination by indirect interaction is more appealing than direct communication
- Stigmergy reduces (or eliminates) communications between agents

# 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 ...

# 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?

# From ants to algorithms

- Swarm intelligence information allows us to address modeling via:
  - Problem solving
  - Algorithms
  - Real world applications



# Modeling

- Observe Phenomenon
- Create a biologically motivated model
- Explore model without constraints

# Modeling...

- Creates a simplified picture of reality
- Observable relevant quantities become variables of the model
- Other (hidden) variables build connections

# A good model has...

- Parsimony (simplicity)

Coherence

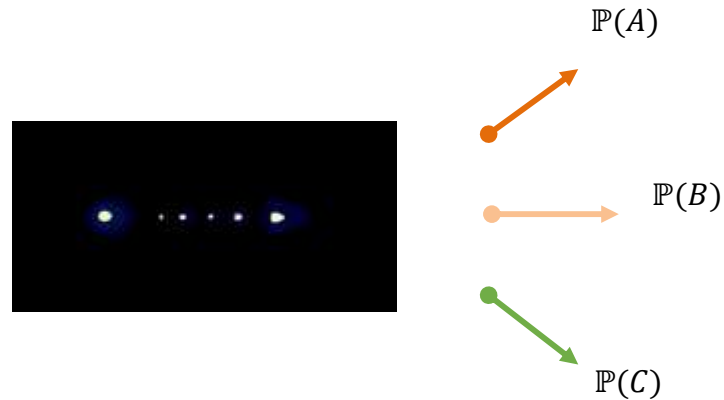
Refutability

Parameter values correspond to values of their natural counterparts



# Pheromone trails

It is well known that the primary means for ants to form and maintain the line is a pheromone trail. Ants deposit a certain amount of pheromone while walking, and each ant probabilistically prefers to follow a direction rich in pheromone.



$$\mathbb{P}(C) < \mathbb{P}(B) < \mathbb{P}(A)$$



# Pheromone trails

## Shortest path around an obstacle

This elementary behavior of real ants can be used to explain how they can find the shortest path that reconnects a broken line after the sudden appearance of an unexpected obstacle has interrupted the initial path.

Let us consider the following scenario:

Ants are moving on a straight line that connects a food source to their nest.





# Pheromone trails

## Shortest path around an obstacle

This elementary behavior of real ants can be used to explain how they can find the shortest path that reconnects a broken line after the sudden appearance of an unexpected obstacle has interrupted the initial path.

An obstacle appears on the path.





# Pheromone trails

## Shortest path around an obstacle

This elementary behavior of real ants can be used to explain how they can find the shortest path that reconnects a broken line after the sudden appearance of an unexpected obstacle has interrupted the initial path.

Those ants which are just in front of the obstacle cannot continue to follow the pheromone trail and therefore they have to choose between turning right or left. In this situation we can expect half the ants to choose to turn right and the other half to turn left.





# Pheromone trails

## Shortest path around an obstacle

This elementary behavior of real ants can be used to explain how they can find the shortest path that reconnects a broken line after the sudden appearance of an unexpected obstacle has interrupted the initial path.

Those ants which choose, by chance, the shorter path around the obstacle will more rapidly reconstitute the interrupted pheromone trail compared to those which choose the longer path. Thus, the shorter path will receive a greater amount of pheromone per time unit and in turn a larger number of ants will choose the shorter path.







# Pheromone trails

Shortest path around an obstacle

This elementary behavior of real ants can be used to explain how they can find the shortest path that reconnects a broken line after the sudden appearance of an unexpected obstacle has interrupted the initial path.

Shortest path is being obtained.





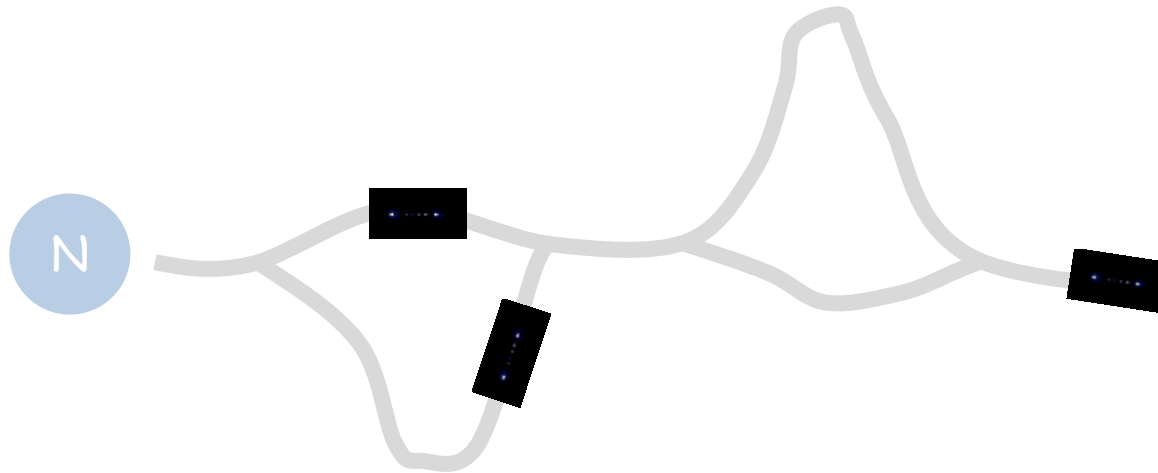
# Pheromone trails

Shortest path from the nest to the food source

Ants are able, without using any spatial Information, to identify a sudden appearance of a food source around their nest, and to find the shortest available path to it.

Let us describe the algorithm:

A small amount of ants travel **randomly** around the nest.





# Pheromone trails

Shortest path from the nest to the food source

Ants are able, without using any spatial Information, to identify a sudden appearance of a food source around their nest, and to find the shortest available path to it.

One of the ants find food source.



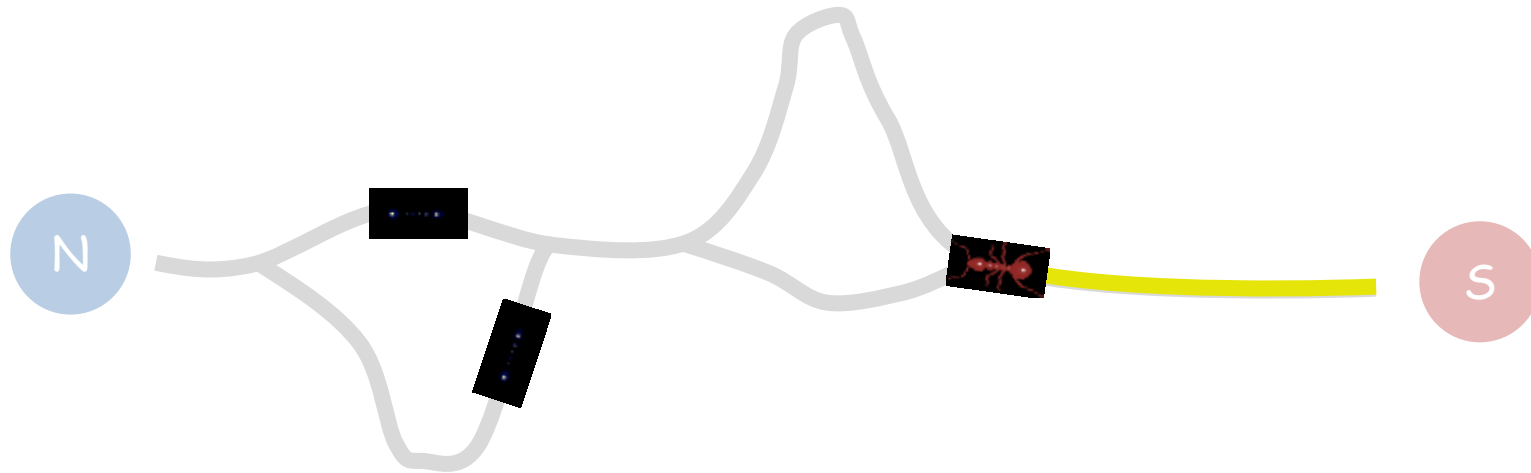


# Pheromone trails

Shortest path from the nest to the food source

Ants are able, without using any spatial Information, to identify a sudden appearance of a food source around their nest, and to find the shortest available path to it.

When ant finds food, it returns to the nest while laying down pheromones trail.



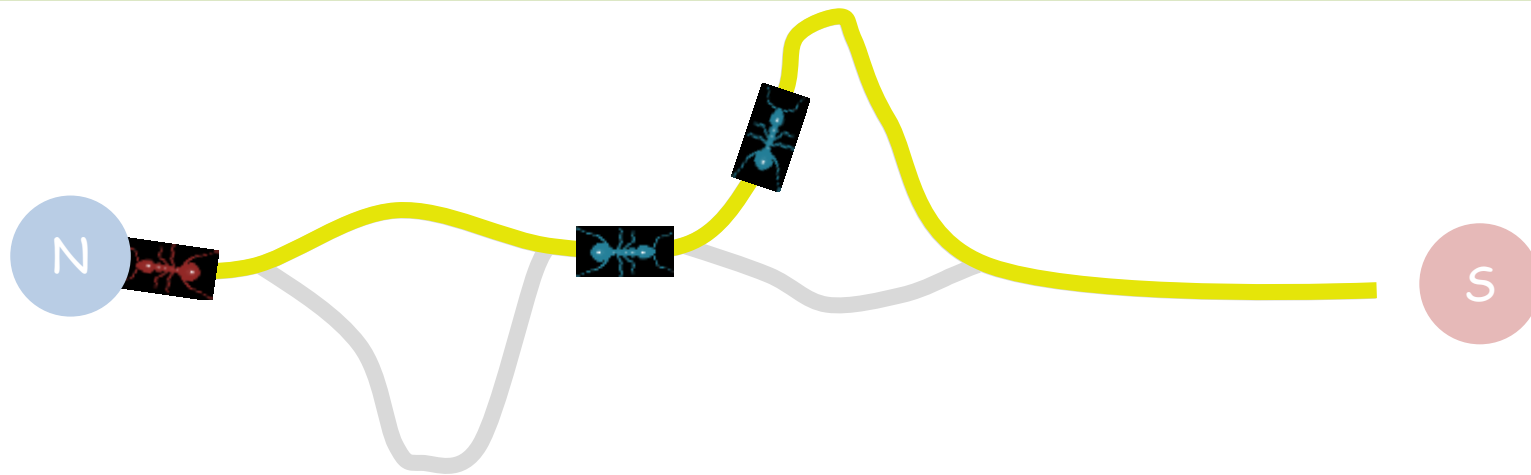


# Pheromone trails

Shortest path from the nest to the food source

Ants are able, without using any spatial Information, to identify a sudden appearance of a food source around their nest, and to find the shortest available path to it.

When other ants find a pheromone trail, they are likely not to keep travelling at random, but to instead follow the trail.



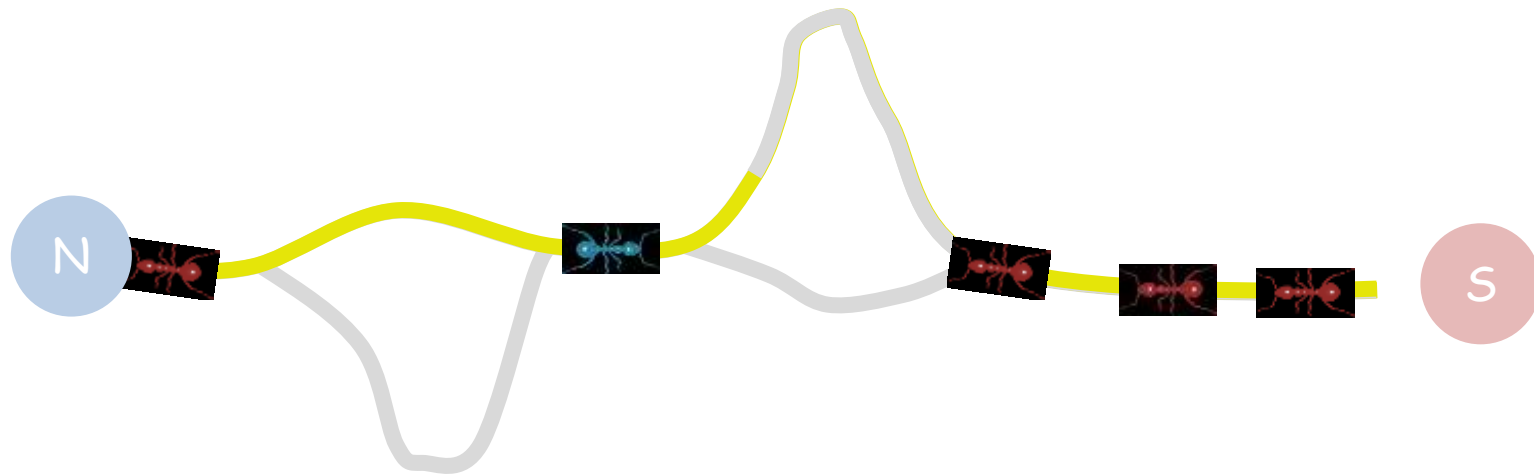


# Pheromone trails

Shortest path from the nest to the food source

Ants are able, without using any spatial Information, to identify a sudden appearance of a food source around their nest, and to find the shortest available path to it.

If an ant eventually find food by following a pheromone trail, it returning to the nest while **reinforcing the trail** with more pheromones.



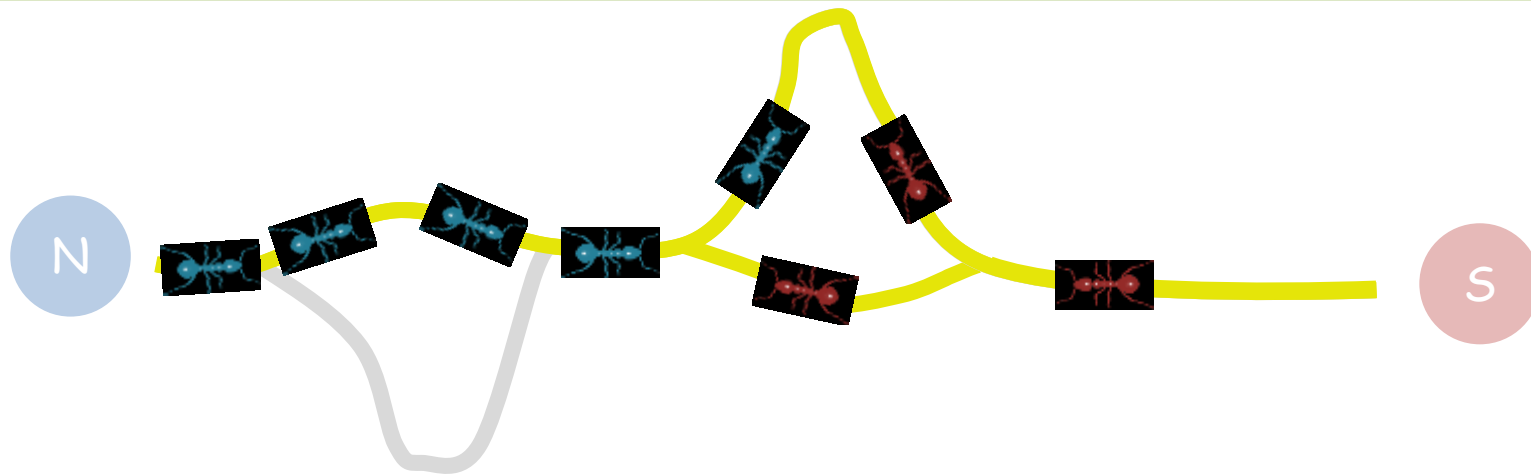


# Pheromone trails

Shortest path from the nest to the food source

Ants are able, without using any spatial Information, to identify a sudden appearance of a food source around their nest, and to find the shortest available path to it.

Due to their **stochastic behavior**, some ants are not following the pheromone trails, and thus uncover more possible paths.



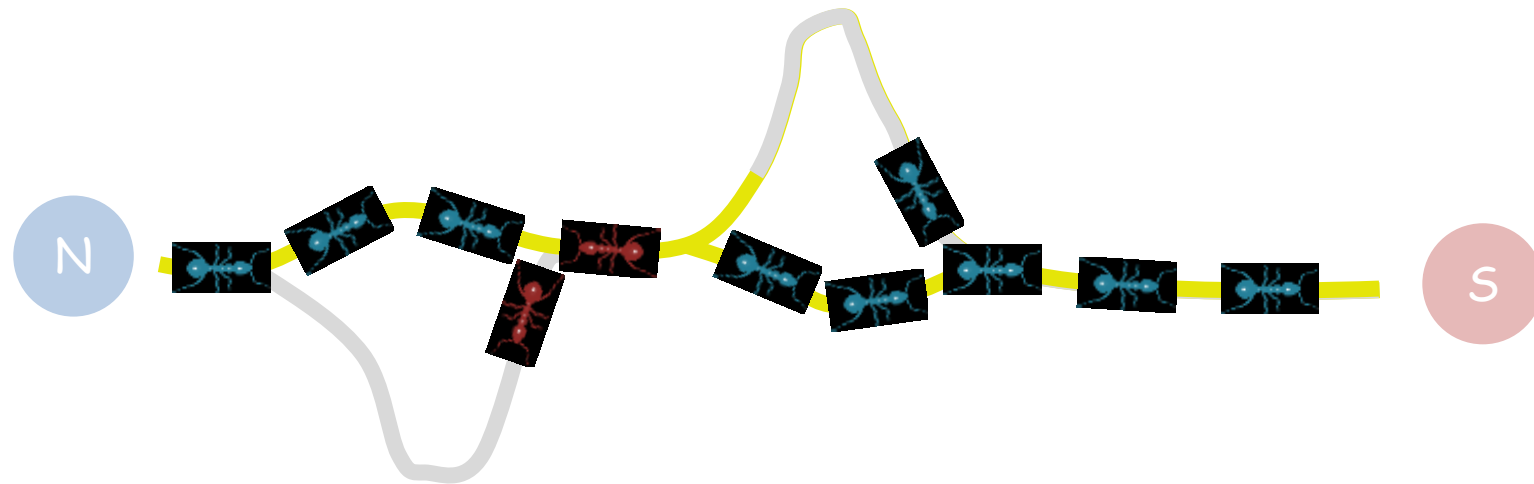


# Pheromone trails

Shortest path from the nest to the food source

Ants are able, without using any spatial Information, to identify a sudden appearance of a food source around their nest, and to find the shortest available path to it.

Over time, however, the pheromones trails starts to **evaporate**, thus reducing its attractive strength.





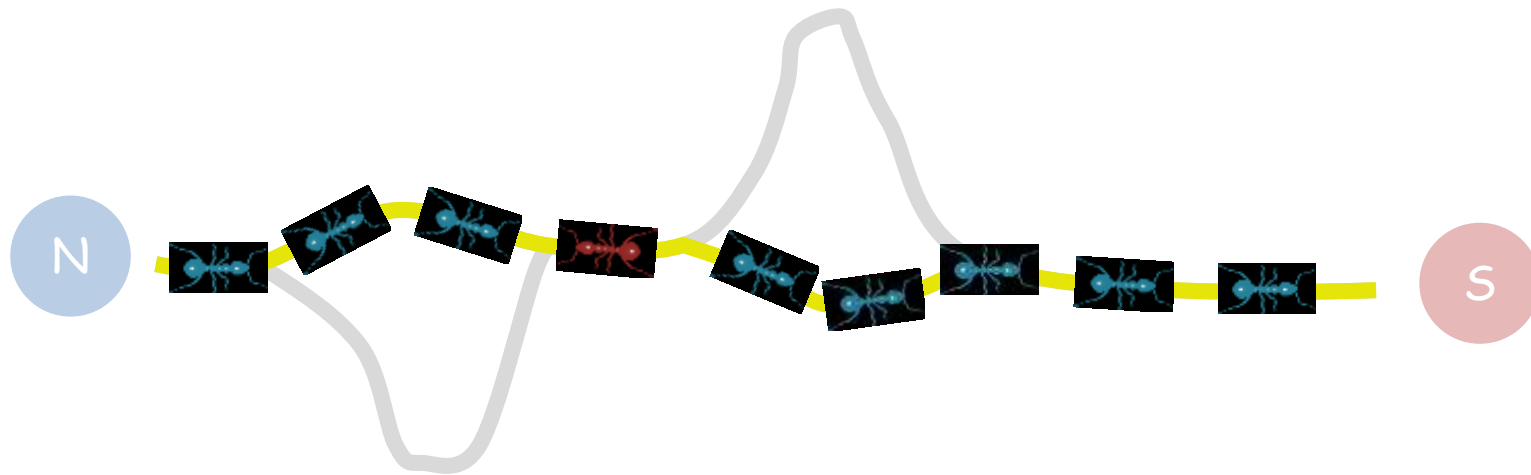


# Pheromone trails

Shortest path from the nest to the food source

Ants are able, without using any spatial Information, to identify a sudden appearance of a food source around their nest, and to find the shortest available path to it.

Shortest path is being obtained.





# Pheromone trails

Shortest path from the nest to the food source

An intuitive proof of correctness

- The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate.

Thus,

- A short path, by comparison, get marched over more frequently, and thus its pheromone density becomes higher on shorter paths than longer ones.



# Pheromone trails

Shortest path from the nest to the food source

Pheromone evaporation also has the advantage of **avoiding the convergence to a locally optimal solution**. If there were no evaporation at all, all paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

Thus, when one ant finds a short path from the colony to a food source, other ants are more likely to follow that path, **and positive feedback eventually leads all the ants following a single path.**

# Ant Colony Optimization (ACO) algorithm

- Proposed by M Dorigo - 1991
- Inspired from natural behavior of an ant Foraging modes
- Proposed in his Ph D Thesis
- Solving multimodal optimization problems



# ACO

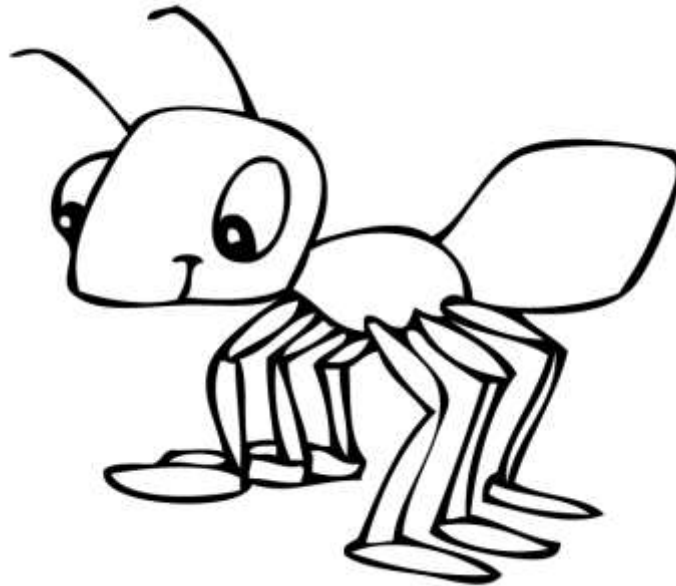
- Working on a connected graph  $G = (V, E)$ , the ACO algorithm is able to find a shortest path between any two nodes
- Capabilities
  - A colony of ants is employed to build a solution in the graph
  - A probabilistic transition rule is used for determining the next edge of the graph on which an ant will move; this moving probability is further influenced by a heuristic desirability
  - The "routing table" is represented by a pheromone level of each edge indicating the quality of the path
- The most important aspect in this algorithm is the **transition probability**  $p_{ij}$  for an ant  $k$  to move from  $i$  to  $j$

# ACO

```
While it < max_it do,  
  for each ant do,  
    build_solution();  
  endfor  
  update_pheromone();  
endwhile
```

Dorigo, M. Optimization, Learning and Natural Algorithms. Politecnico di Milano, Italy, 1992.

# Solving NP-hard combinatorial problems





# ACO algorithms

Ant Colony Optimization (ACO) studies artificial systems that **take inspiration from the behavior of real ant colonies** and which are used to solve discrete optimization problems.

Some applications:



## Set partition problem

Deciding whether a given multiset of positive integers can be partitioned into two subsets  $A$  and  $B$  such that the sum of the numbers in  $A$  equals the sum of the numbers in  $B$ .





# ACO algorithms

Ant Colony Optimization (ACO) studies artificial systems that **take inspiration from the behavior of real ant colonies** and which are used to solve discrete optimization problems.

Some applications:



## **Job - Shop problems**

Given a number of jobs have to be done and every job consists of using a number of machines for a certain amount of time. Find the best planning to do all the jobs on all the different machines in the shortest period of time.



# ACO algorithms

Ant Colony Optimization (ACO) studies artificial systems that **take inspiration from the behavior of real ant colonies** and which are used to solve discrete optimization problems.

Some applications:



## Multiple knapsack problem

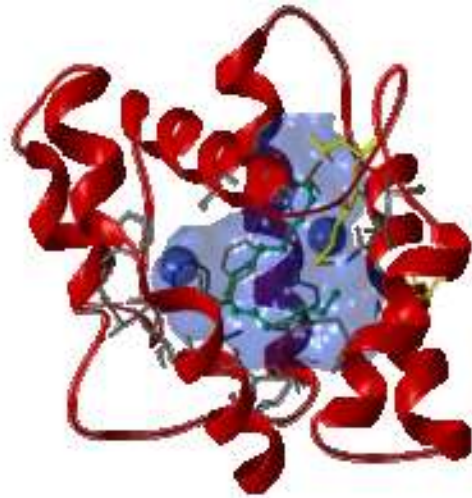
Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.



# ACO algorithms

Ant Colony Optimization (ACO) studies artificial systems that **take inspiration from the behavior of real ant colonies** and which are used to solve discrete optimization problems.

Some applications:



## Protein folding

The process by which a protein structure assumes its functional shape or conformation.



# ACO algorithms

Ant Colony Optimization (ACO) studies artificial systems that **take inspiration from the behavior of real ant colonies** and which are used to solve discrete optimization problems.

Some applications:



## Data mining

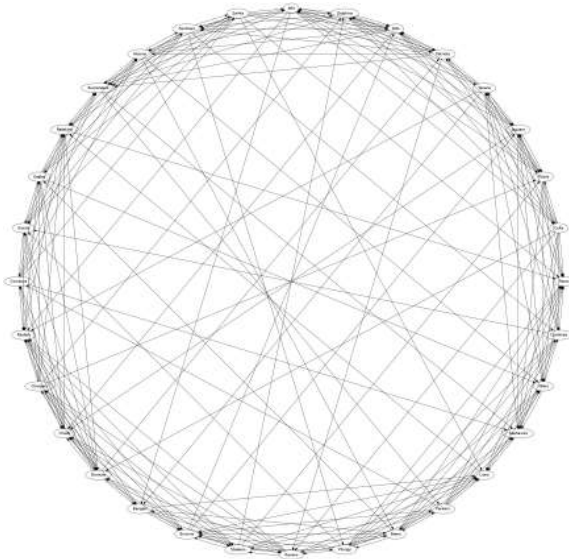
the computational process of discovering patterns in large data sets.



# ACO algorithms

Ant Colony Optimization (ACO) studies artificial systems that **take inspiration from the behavior of real ant colonies** and which are used to solve discrete optimization problems.

Some applications:



## Travelling salesman problem

Given a list of cities and the distances between each pair of cities, determine the shortest possible route that visits each city exactly once and returns to the origin city.

# Travelling Salesperson Problem (TSP)

Initialize

Loop /\* at this level each loop is called an iteration \*/

Each ant is positioned on a starting node

Loop /\* at this level each loop is called a step \*/

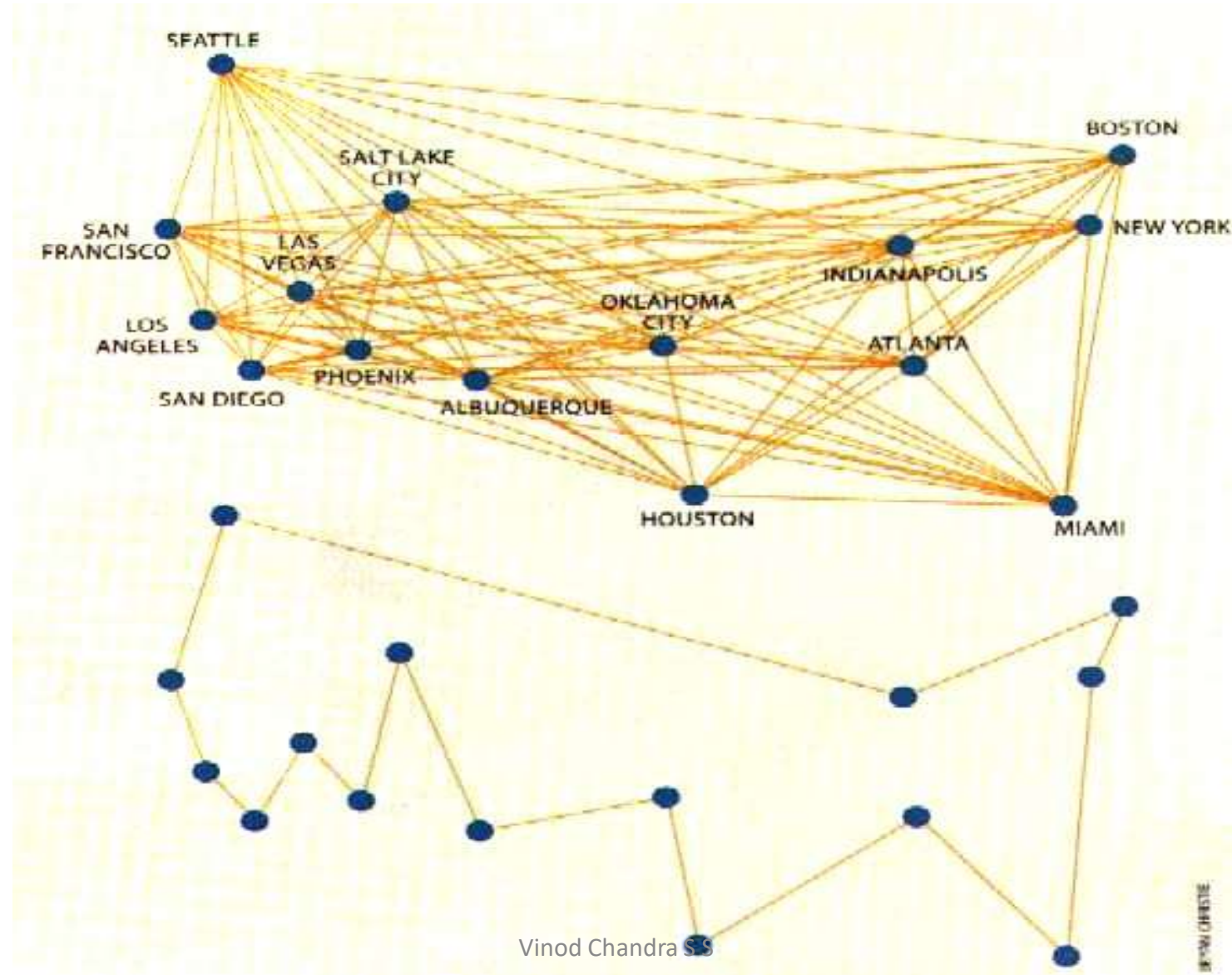
Each ant applies a state transition rule to incrementally  
build a solution and a local pheromone updating rule

Until all ants have built a complete solution

A global pheromone updating rule is applied

Until End\_condition

# Traveling sales ants

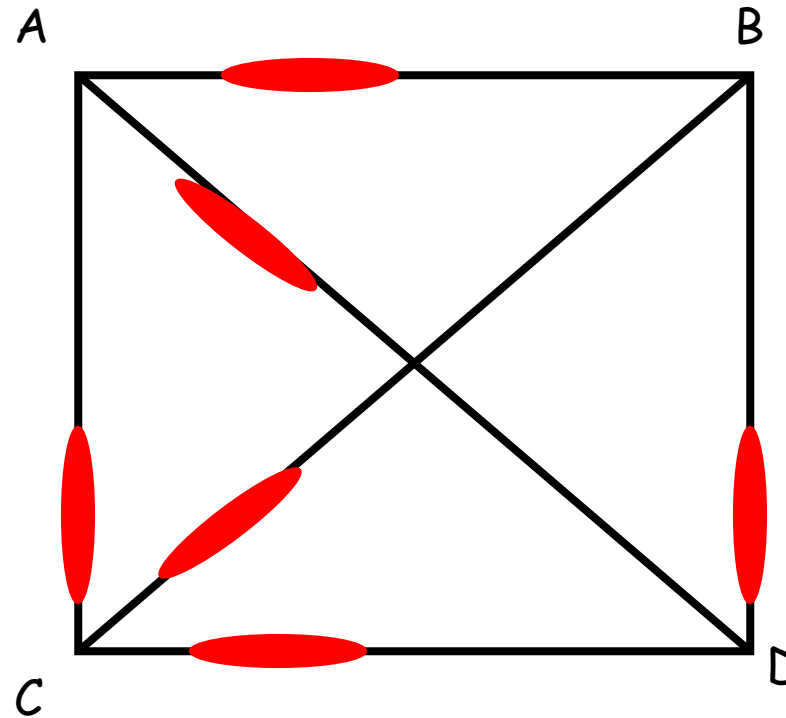


# 4-city TSP

Initially, random levels of pheromone are scattered on the edges



Pheromone

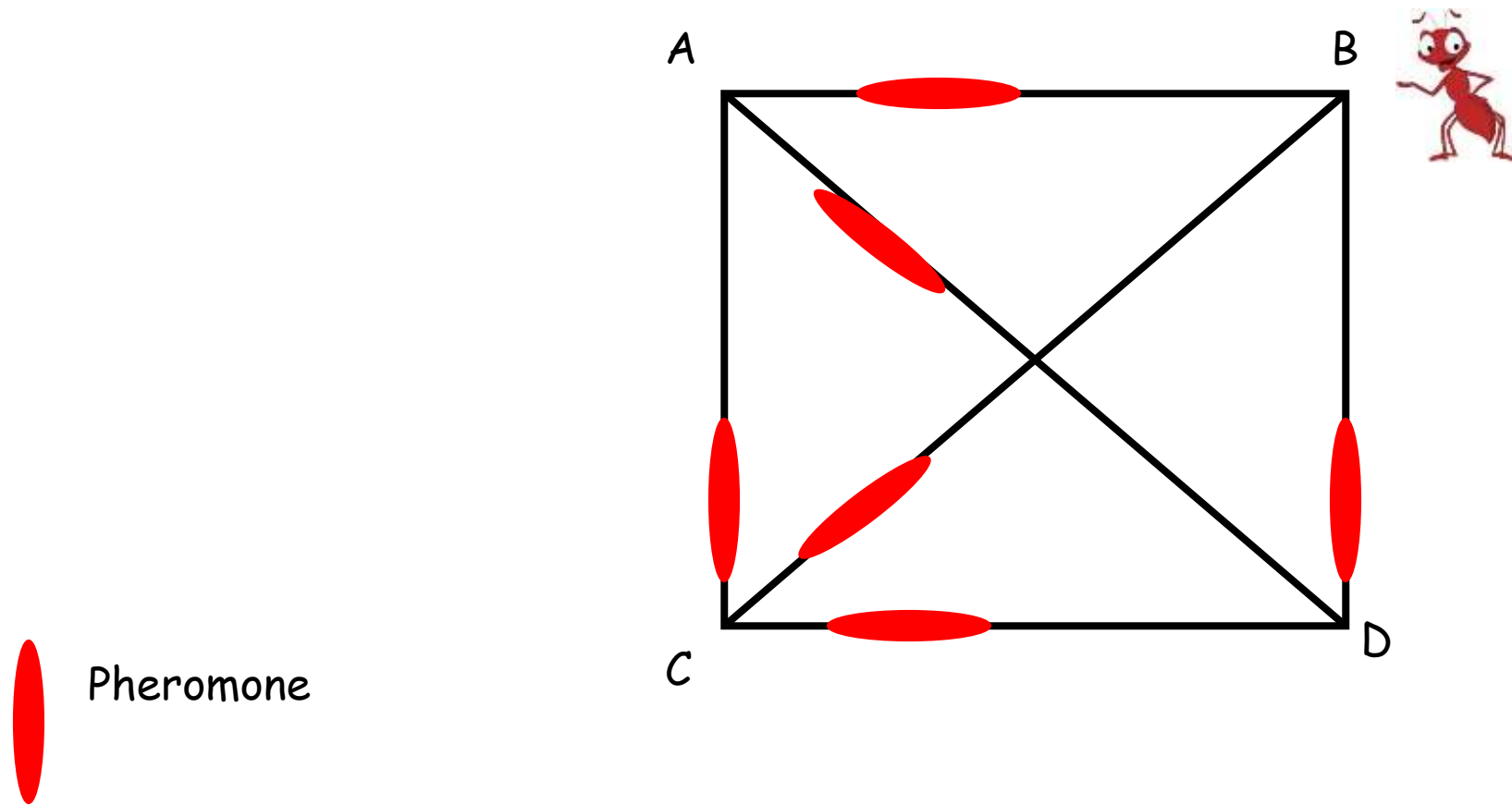


AB: 10, AC: 10, AD: 30, BC: 40, CD: 20



# 4-city TSP

An ant is placed at a random node



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

# 4-city TSP

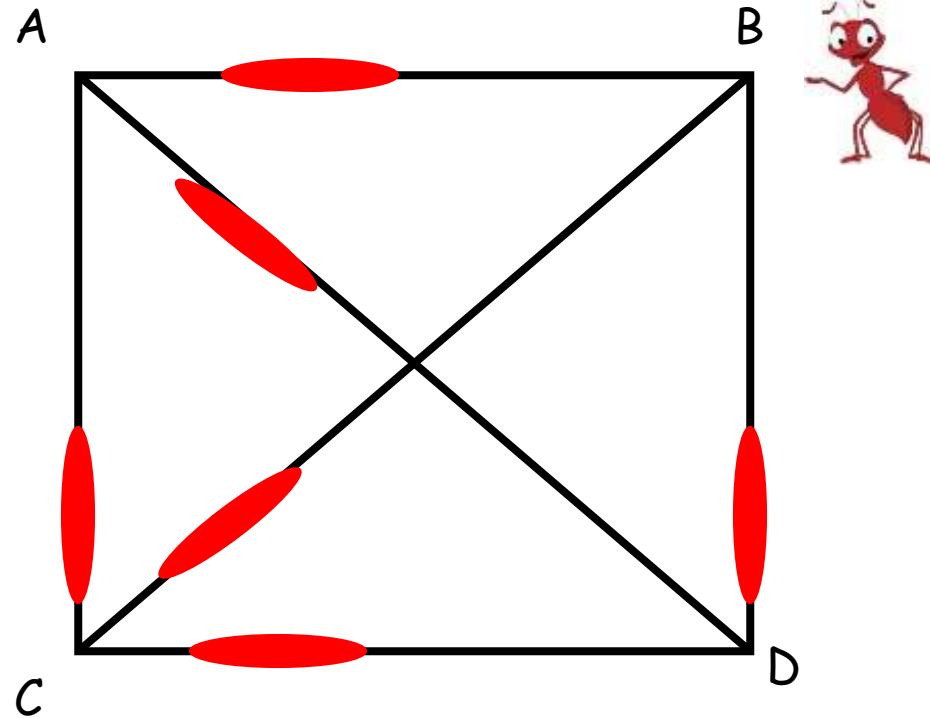
The ant decides where to go from that node, based on probabilities calculated from:

- pheromone strengths,
- next-hop distances.

Suppose this one chooses BC



Pheromone



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

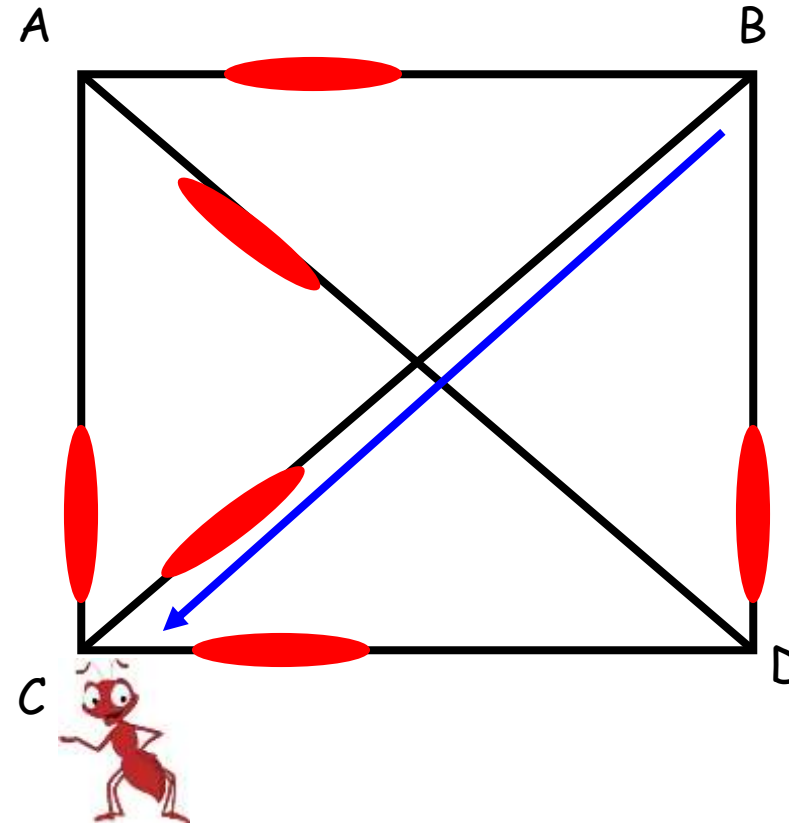
# 4-city TSP

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  $CD$



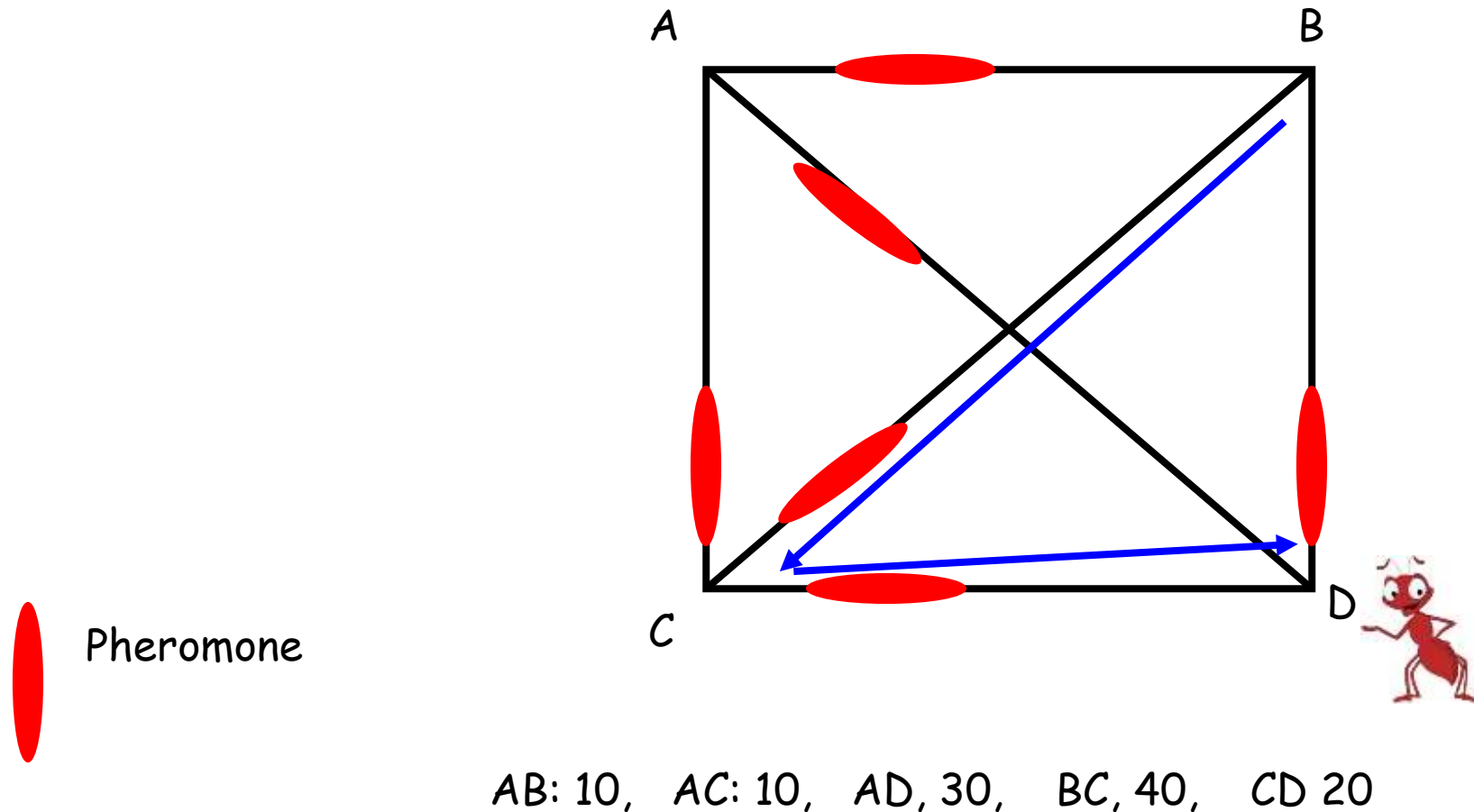
Pheromone



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

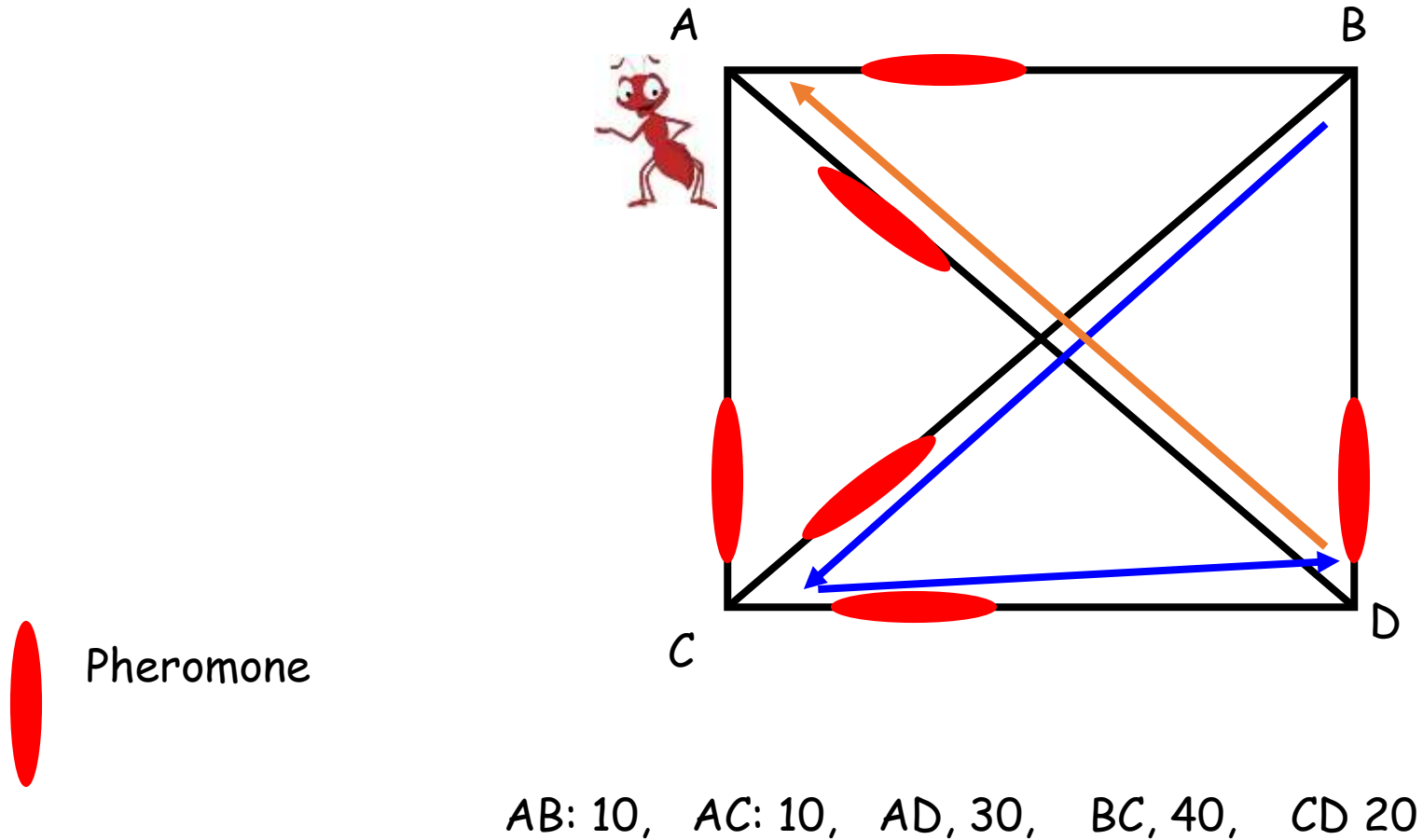
# 4-city TSP

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



# 4-city TSP

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



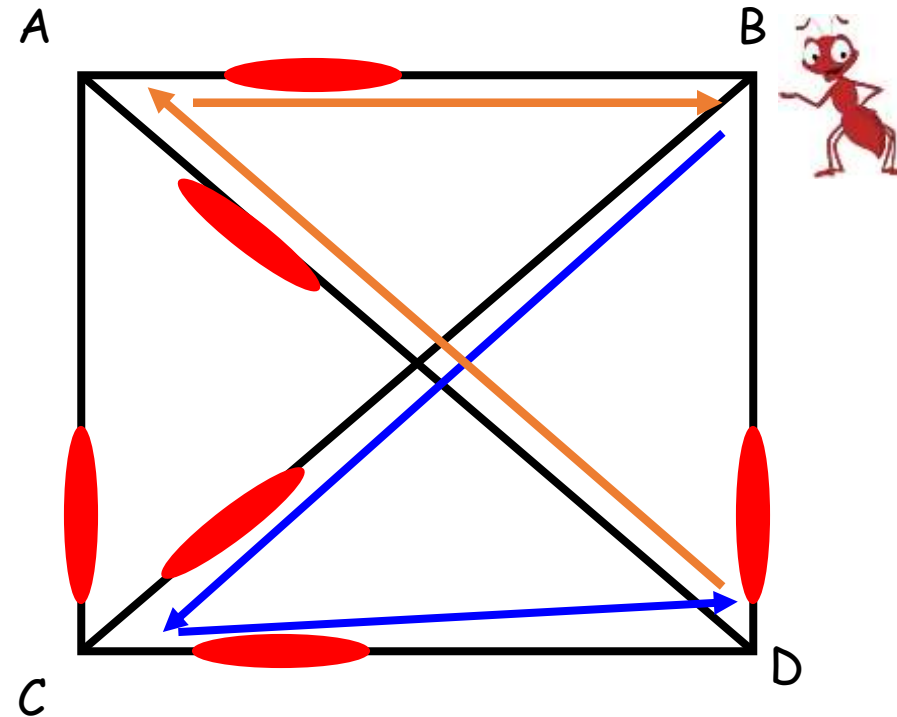
# 4-city TSP

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.



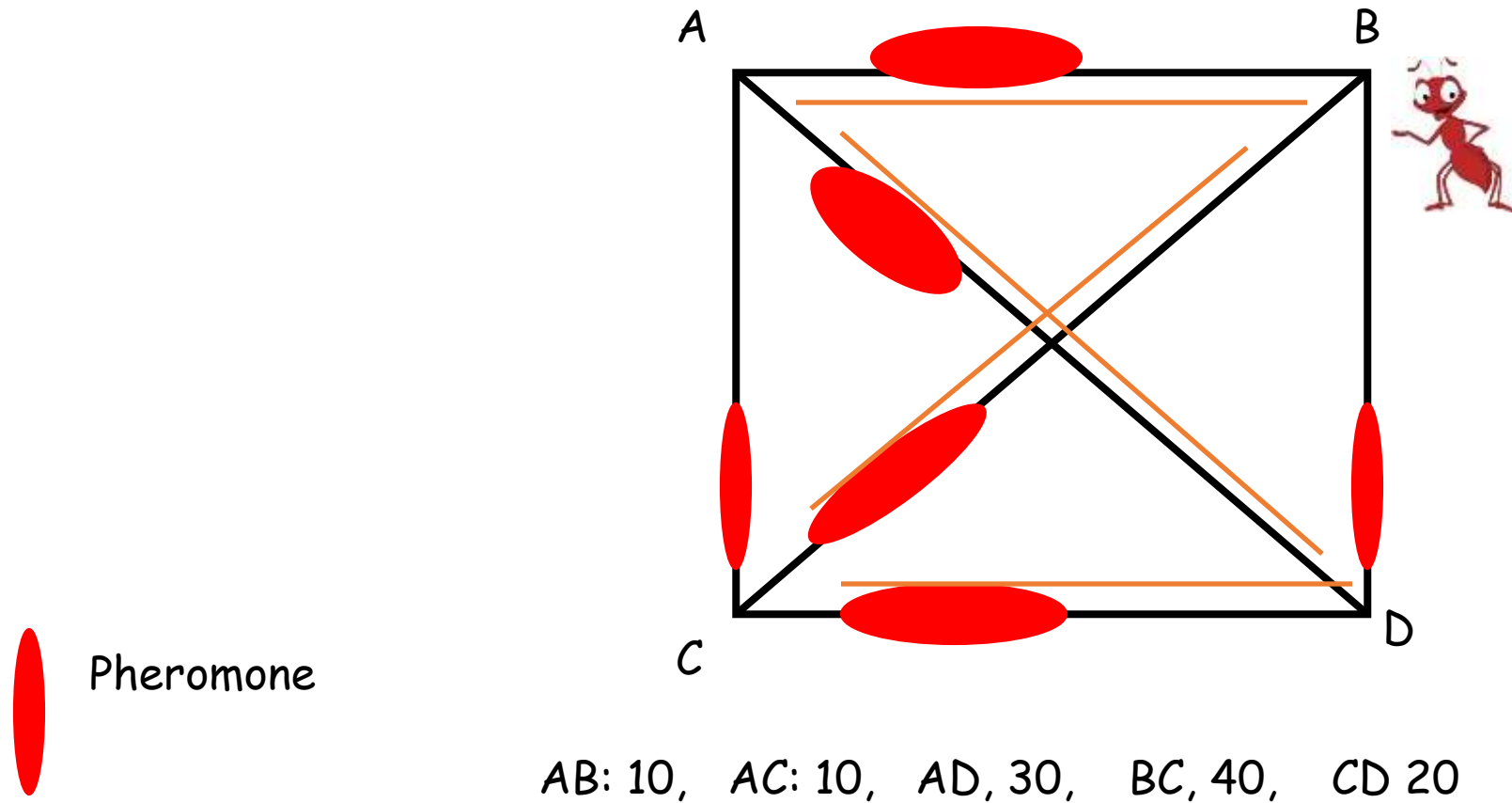
Pheromone



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

# 4-city TSP


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

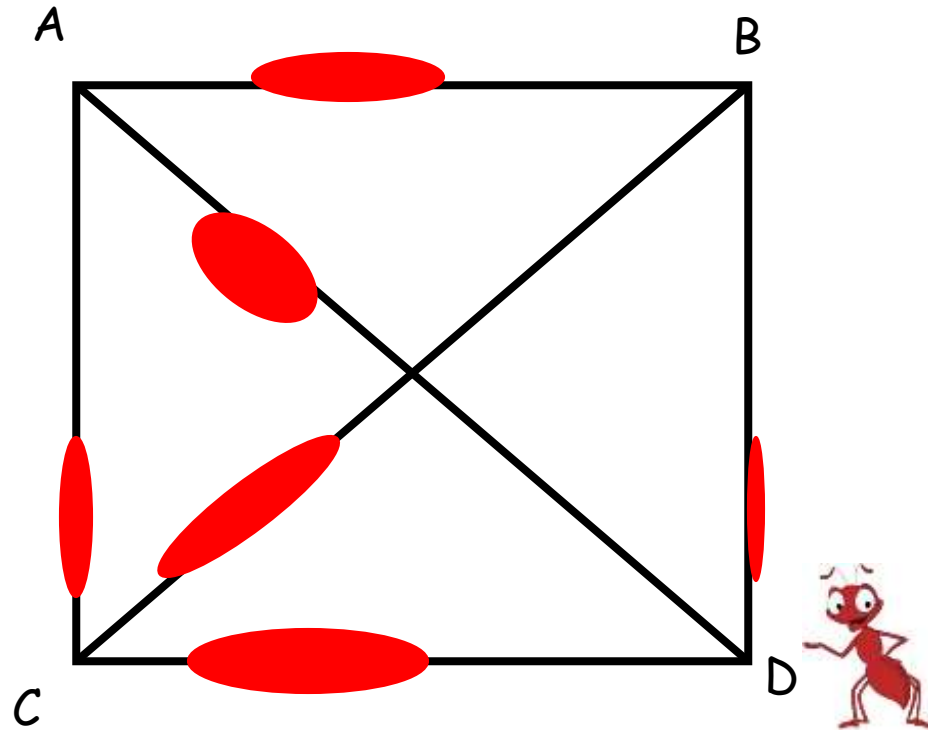


# 4-city TSP

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

Where will he go?

 Pheromone



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



# ACO algorithm for 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

# ACO algorithm for TSP

[a simplified version with all essential details]

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 laid (belatedly) by each ant as follows: it places pheromone on all links of its tour, with strength depending on how good the tour was

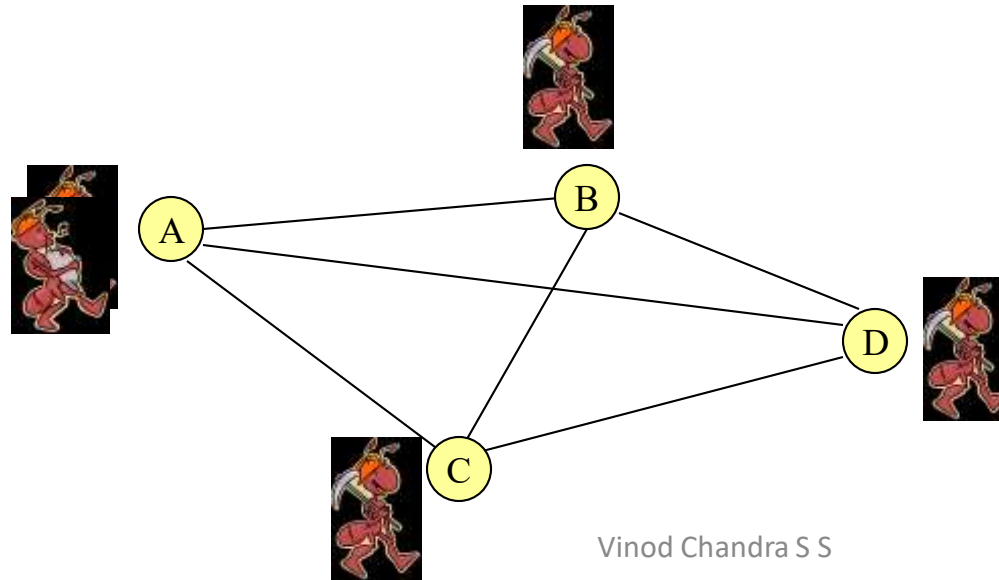
3. Then we go back to 1 and repeat the whole process many times, until we reach a termination criterion

# ACO algorithm for TSP

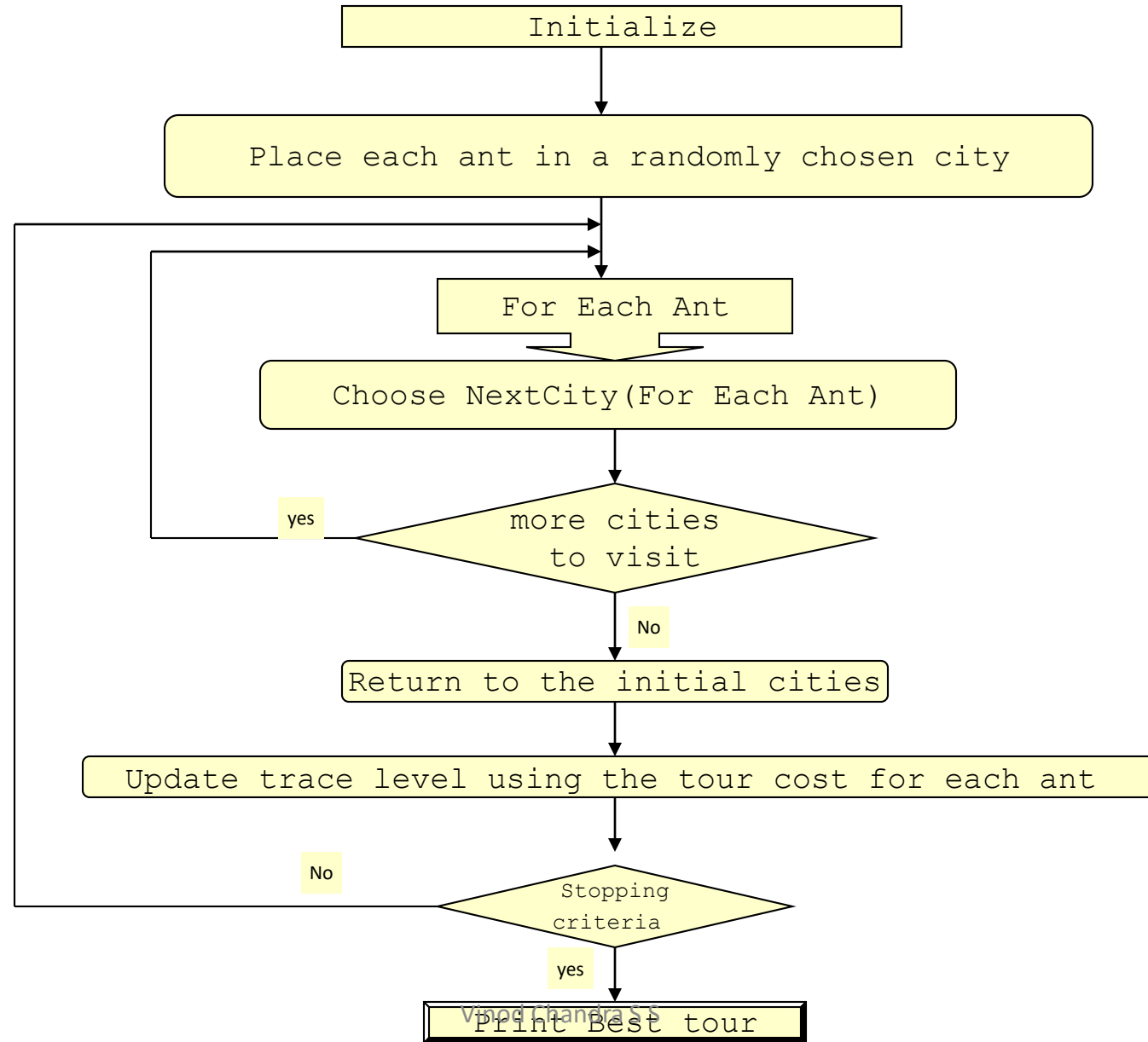
Graph (N,E): where N = cities(nodes), E = edges

$\tau_{ij}$  = tour cost from city i to city j (edge weight)

Ant move from one city i to the next j with some transition probability.



# ACO algorithm for TSP



# ACO algorithm for TSP

- $N$  - set of nodes (cities),  $|N| = n$
- $A$  - set of arcs, fully connecting  $N$
- Weighted graph  $G = (N, A)$
- Each arc has a weight  $d_{ij}$  - distance
- Problem:

Find minimum length Hamiltonian circuit



# Ant system: Pheromone initialization

- Pheromone initialization

$$T_{ij} = m / C^{nn},$$

where:

- $m$  - number of ants
- $C^{nn}$  - path length of nearest-neighbor algorithm



# Ant system: Tour construction

- Ant  $k$  is located in city  $i$
- $J_i^k$  is the neighborhood of city  $i$
- Probability to go to city  $j$  :

$$p_{ij}^k = \left\{ \begin{array}{ll} \frac{\tau_{ij}^\alpha(t) \times \eta_{ij}^\beta(t)}{\sum_{l \in J_i^k} \tau_{il}^\alpha(t) \times \eta_{il}^\beta(t)}; & \text{if } j \in J_i^k \\ 0; & \text{Otherwise} \end{array} \right\}$$



$\alpha$  and  $\beta$  are adjustable parameters that control the relative weight of the trail intensity  $\tau_{ij}$  and the visibility  $\eta_{ij}$ , respectively

# Tour construction: Comprehension



- $\alpha = 0$  - greedy algorithm
- $\beta = 0$  - only pheromone is at work
  - quickly leads to stagnation





# Ant system: Update pheromone trails - evaporation

Evaporation for all connections  $\forall (i, j) \in L$ :

$$\tau_{ij} \leftarrow (1 - \rho) \tau_{ij},$$

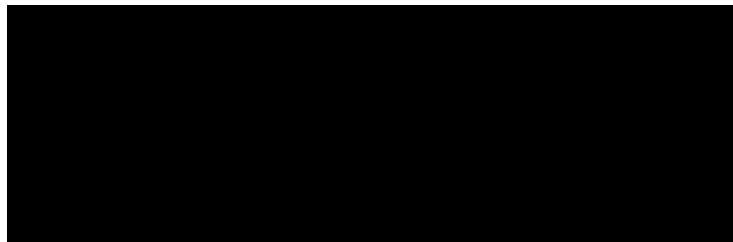
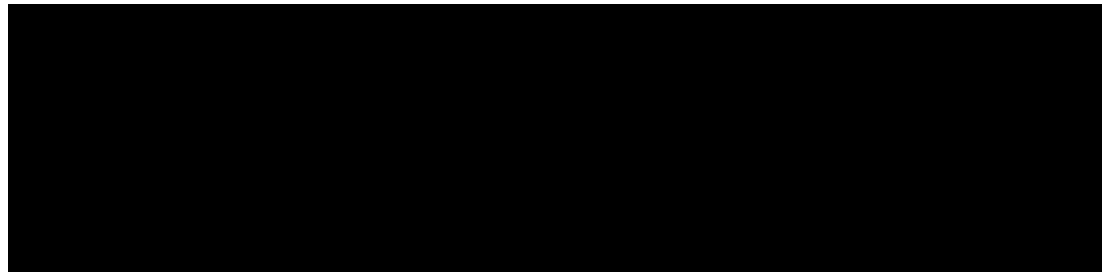
$\rho \in [0, 1]$  - evaporation rate

Prevents convergence to suboptimal solutions



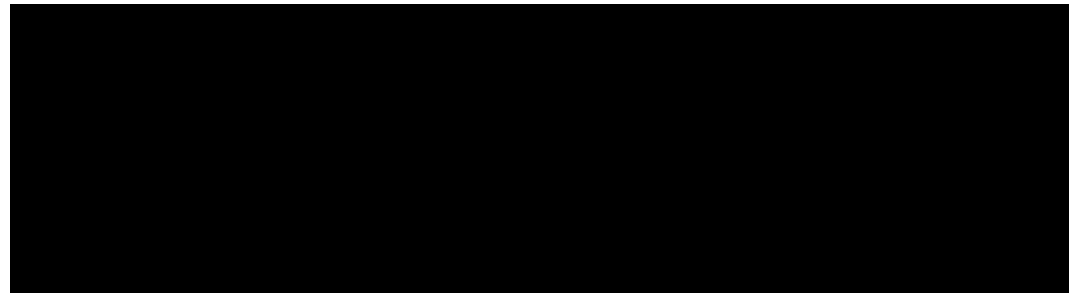
# Ant system: Update pheromone trails - deposit

- $T^k$  - path of ant  $k$
- $C^k$  - length of path  $T^k$
- Ants deposit pheromone on visited arcs:



# Elitist ant system

- Best-so-far ant deposits pheromone on each iteration:



# Rules for transition probability

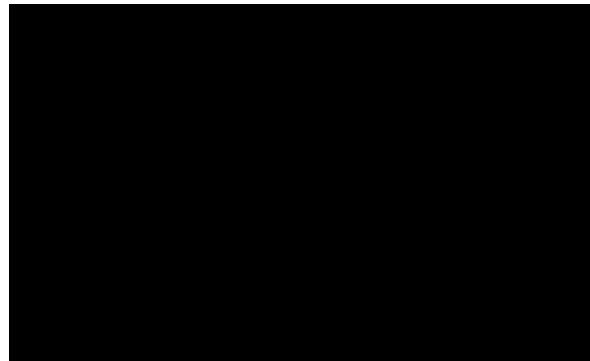
1. Whether or not a city has been visited

Use of a **memory**(tabu list):  $T$  : set of all cities that are to be visited

2.  $P_{ij}$  =  $\eta_{ij}$  **visibility** : Heuristic desirability of choosing city j when in city i.

3. **Pheromone trail**:  $\tau_{ij}$  This is a global type of information

**Transition probability for ant k to go from city i to city j while building its route.**



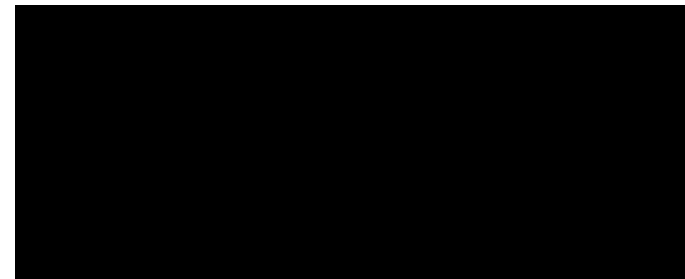
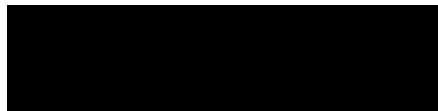
$a = 0$ : closest cities are selected

Vinod Chandra S S

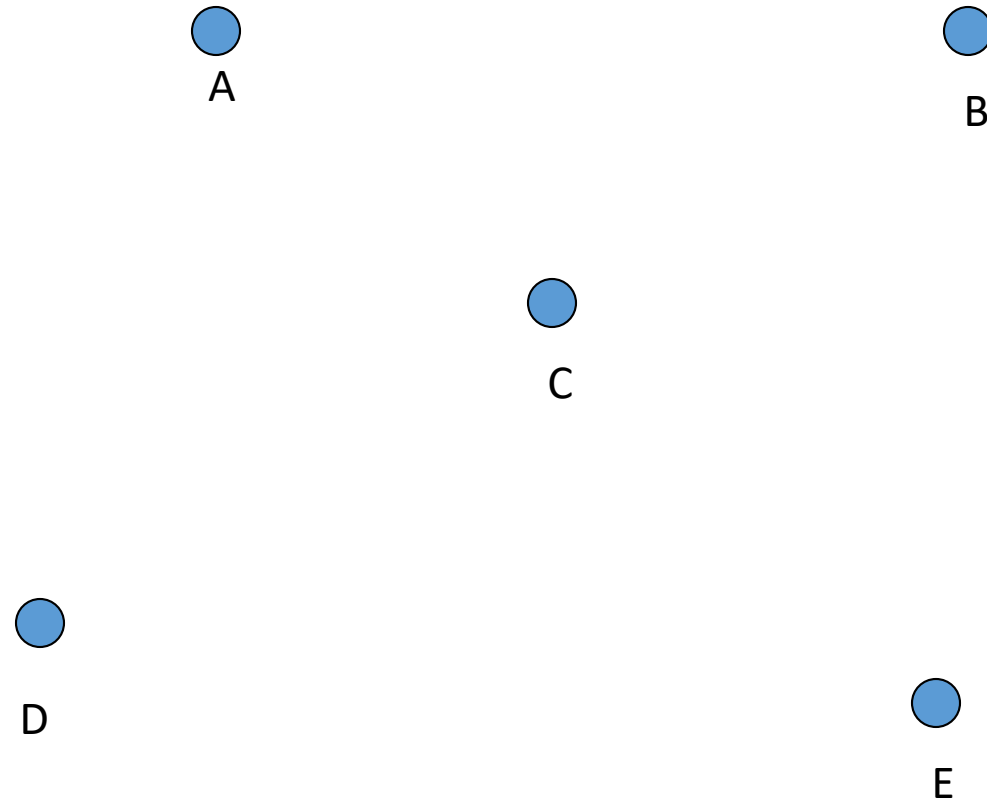
# Global pheromone update

$A_k(r,s)$	<i>is amount of pheromone added to the <math>(r, s)</math> link by ant <math>k</math></i>
$m$	<i>is the number of ants</i>
$\rho$	<i>is a parameter called the pheromone decay rate.</i>
$L_k$	<i>is the length of the tour completed by ant <math>k</math></i>
$T(r, s)$	<i>at the next iteration becomes:</i>

Where



# Simple TSP example

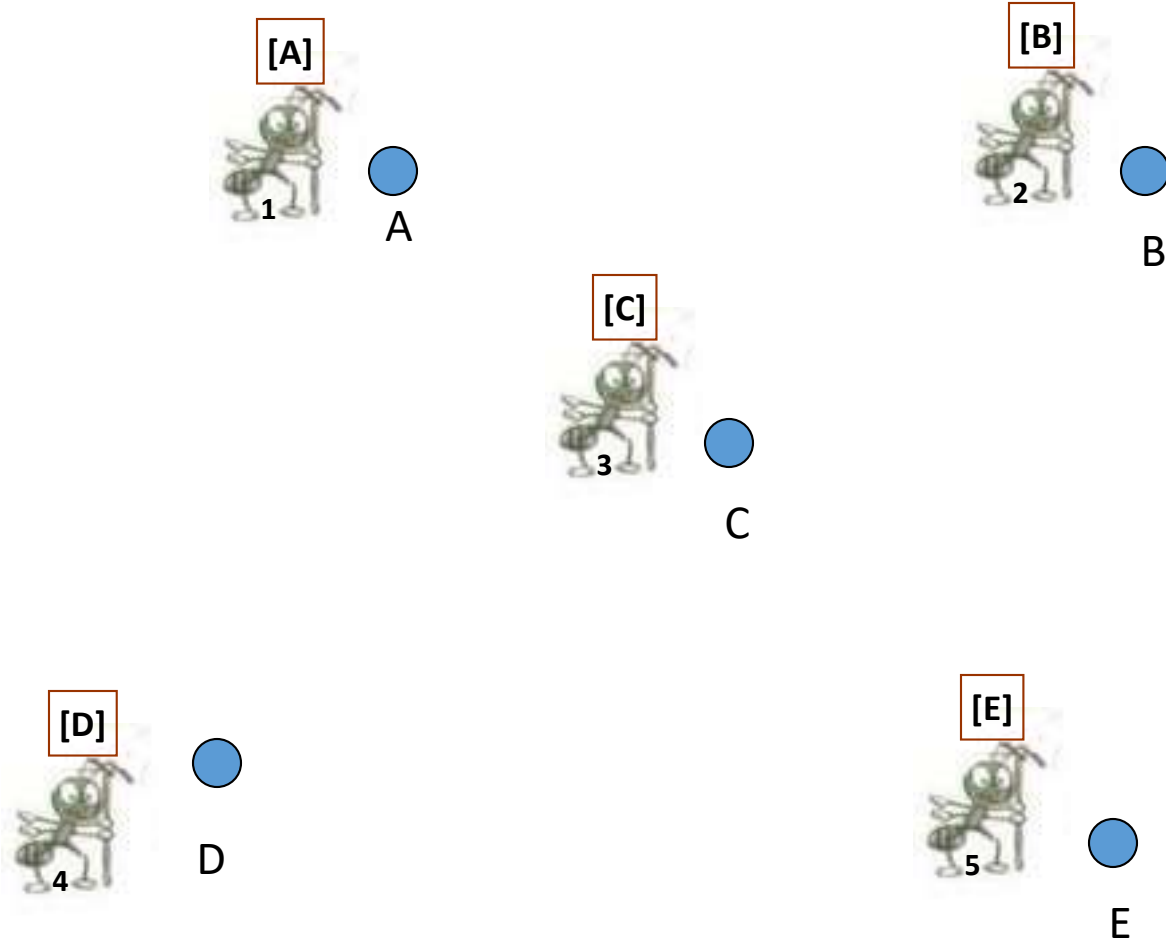


$d_{AB}=100; d_{BC}=60...; d_{DE}=150$

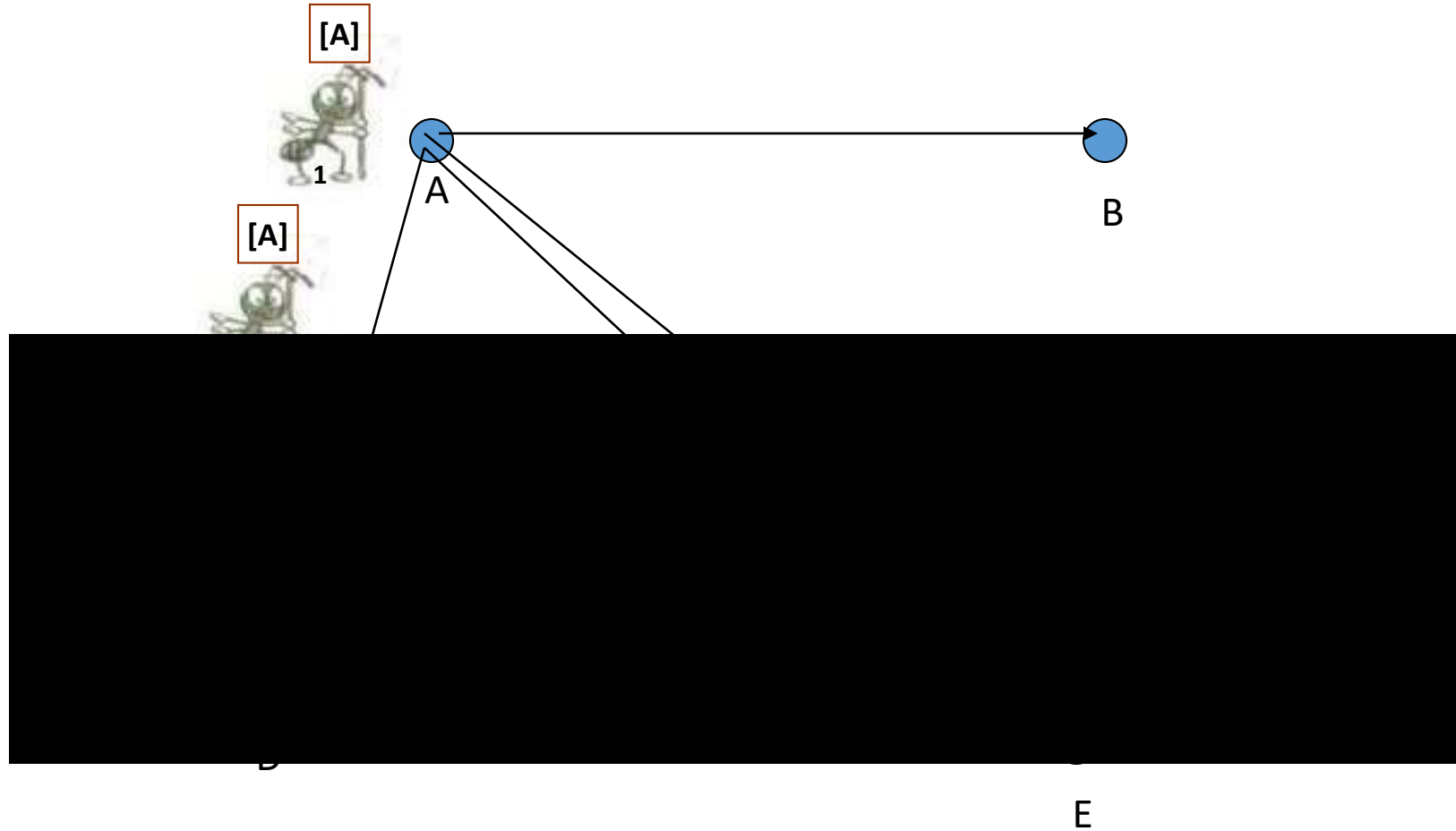
Vinod Chandra S S



# Iteration 1

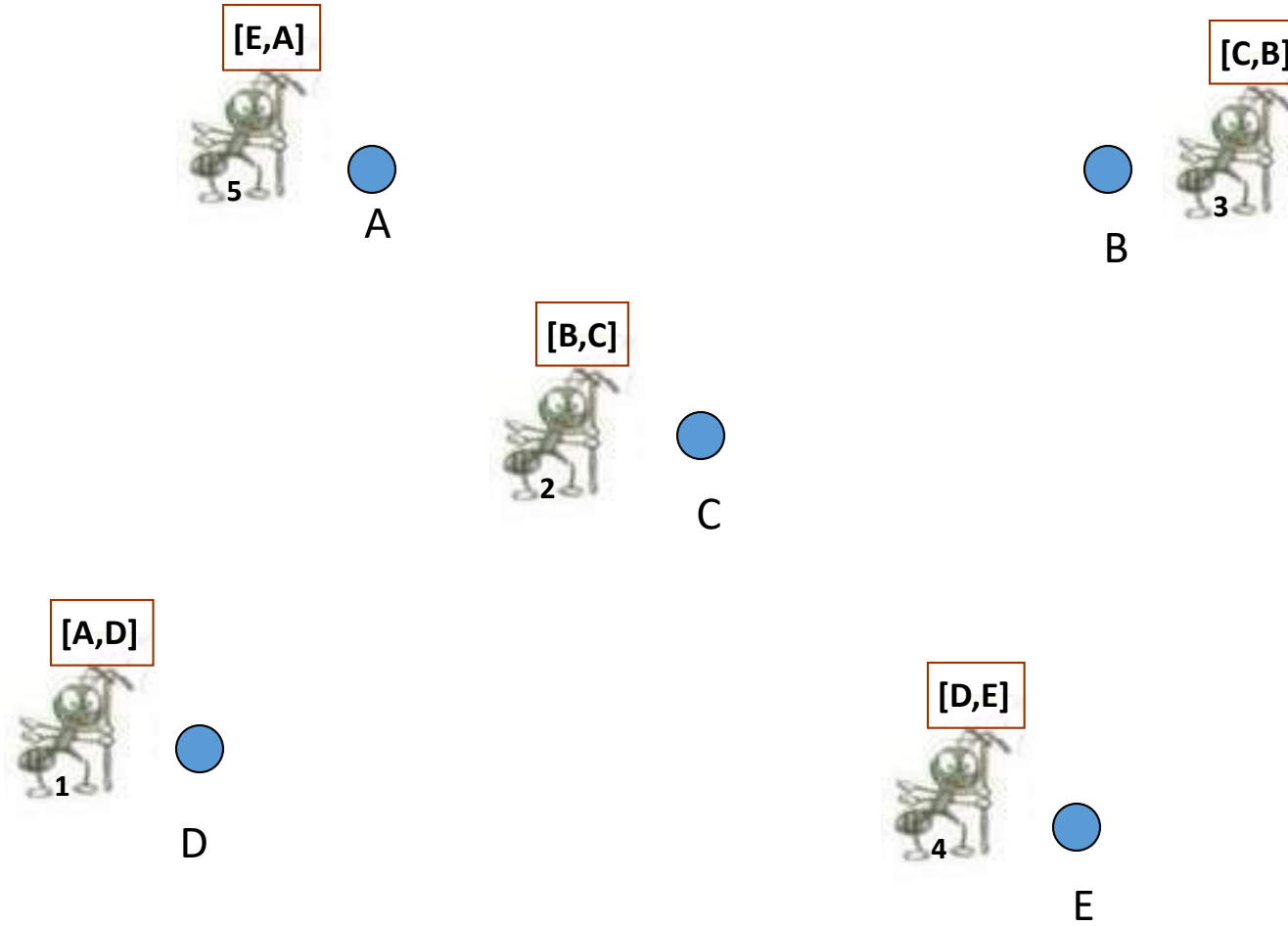


# How to choose next city?

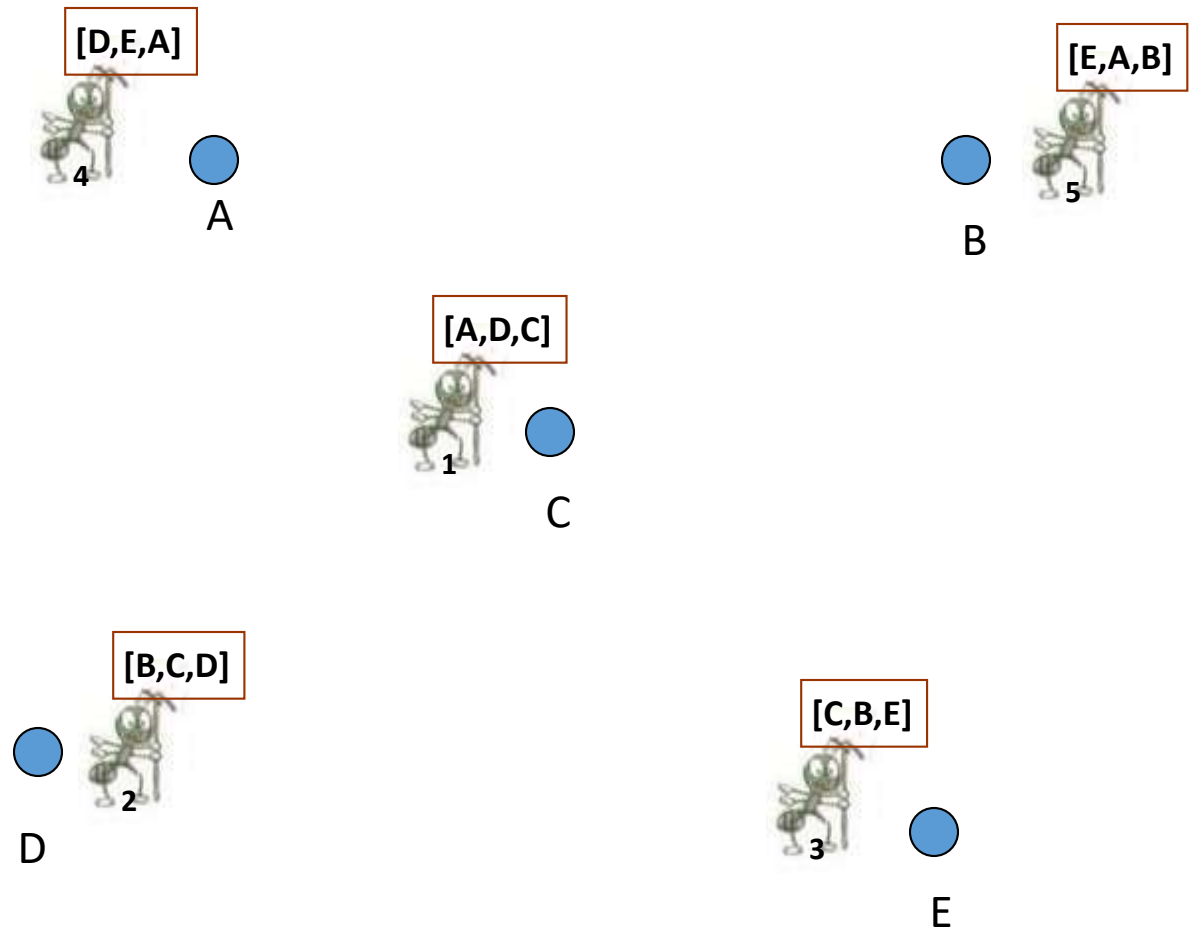




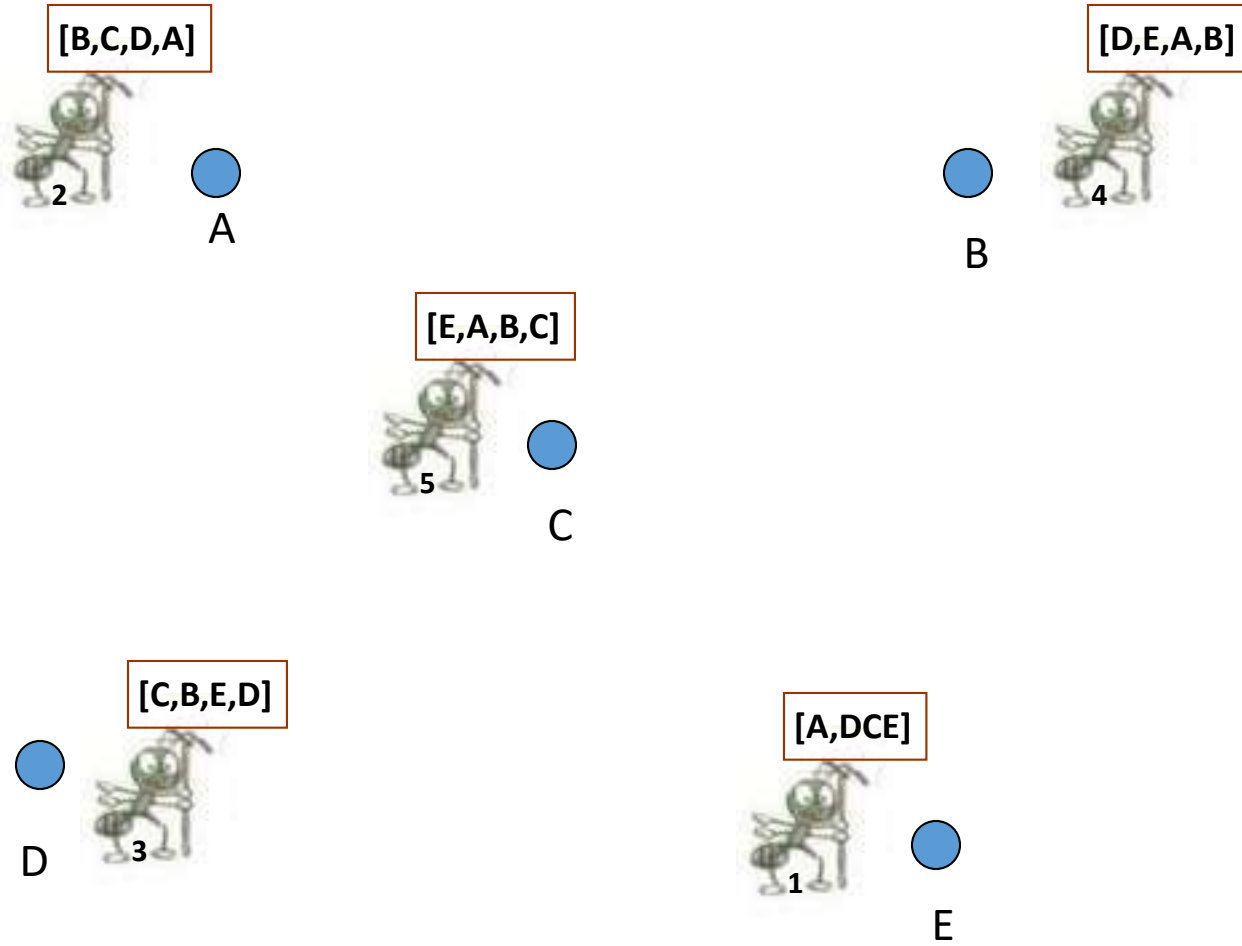
# Iteration 2



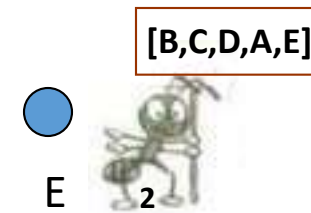
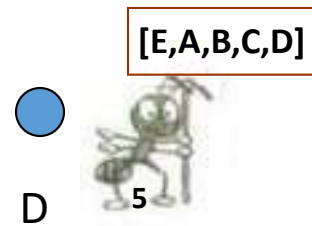
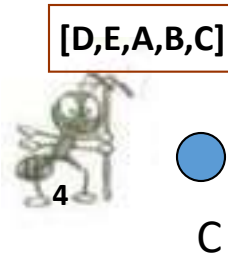
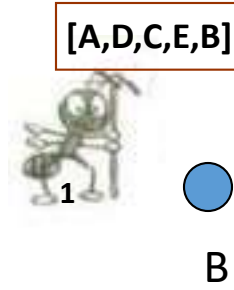
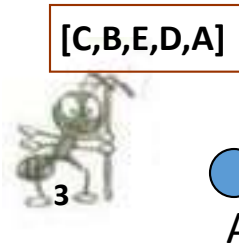
# Iteration 3



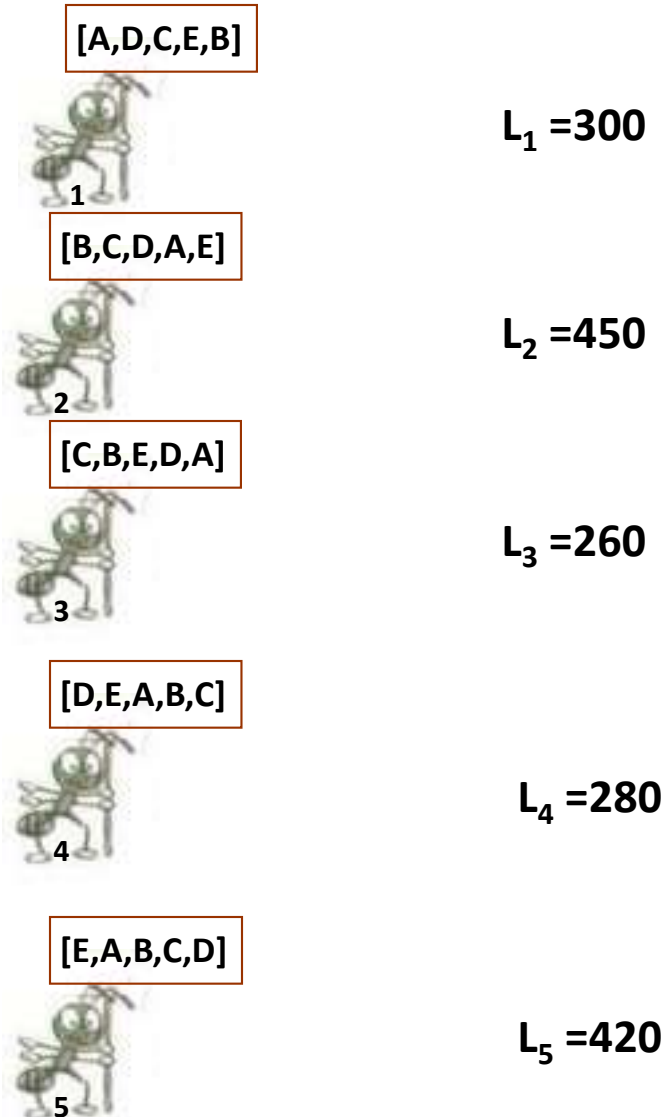
# Iteration 4

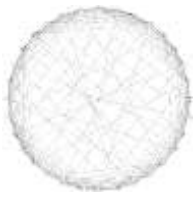


# Iteration 5



# Path and trace update

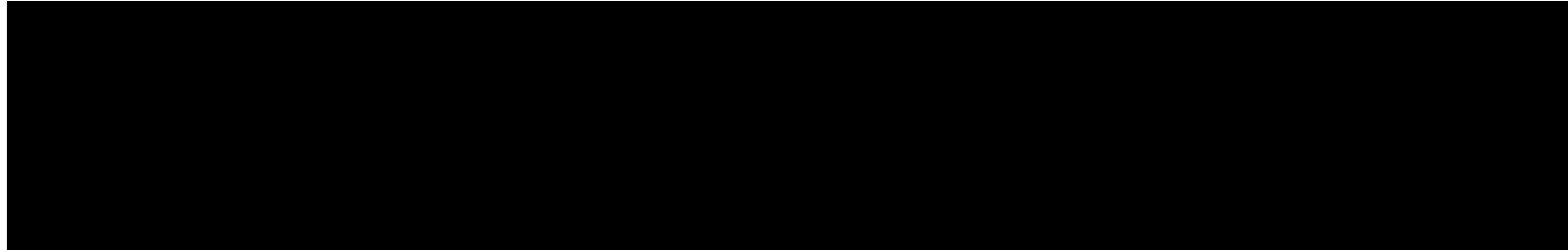




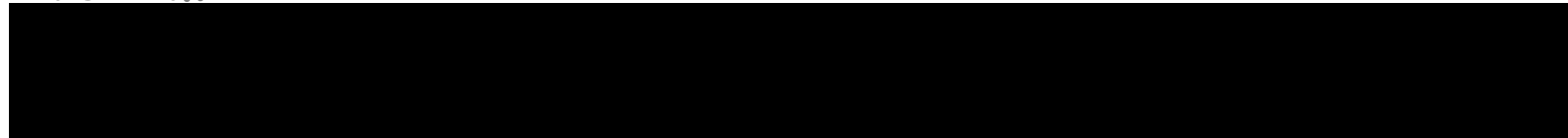
# TSP

For a complete undirected weighted graphs

## Definitions

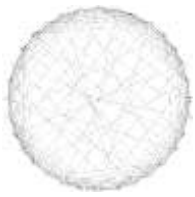


## Problem



## Complexity class

**NP-Hard**

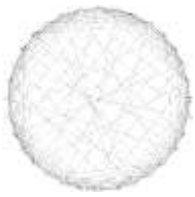


# TSP

For an undirected complete weighted graphs

Algorithms for **deterministic** solution

- **Brute force search**  
Compute all possible cycles and get the minimal path.  
Time complexity:  $O(|V|!)$   
**Exact solution**
- **Dynamic programming**  
Recursive algorithms such as the Held-Karp algorithm.  
Time complexity:  $O(|V|^2 2^{|V|})$   
**Exact solution**
- **Greedy algorithm**  
Such as the nearest neighbor (NN) algorithm.  
Time complexity:  $O(\log |V|)$   
**Approximate solution**

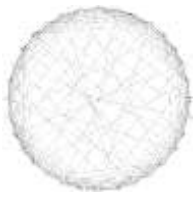


# TSP

## ACS solution

The Ant Colony System (ACS) is one of the ACO algorithms, which gives a **non - deterministic approximate solution** to the TSP problem with a time complexity of  $O(|V|^3 \cdot c)$ , where  $c$  is a constant.





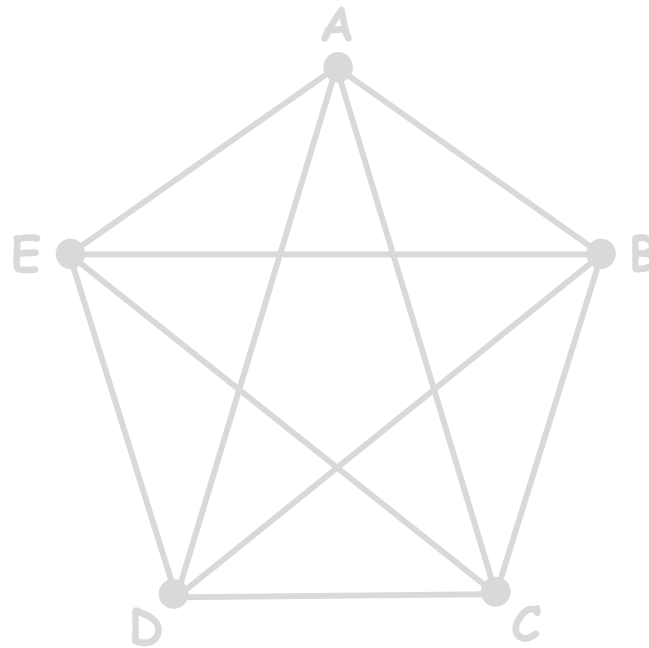
# TSP

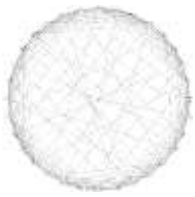
For an undirected complete weighted graphs

## Algorithm - Initialization

Choose constant  $k$  between 1 and  $|V|$  to be the number of ants.

$$k = 3$$



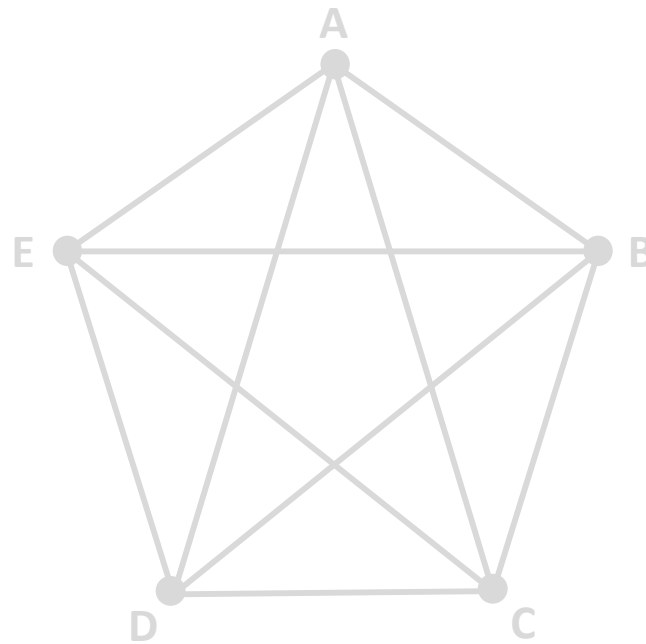


# TSP

For an undirected complete weighted graphs

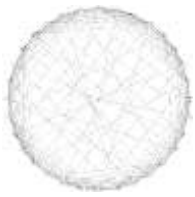
## Algorithm - Initialization

Place each ant in a randomly chosen city.



$k = 3$



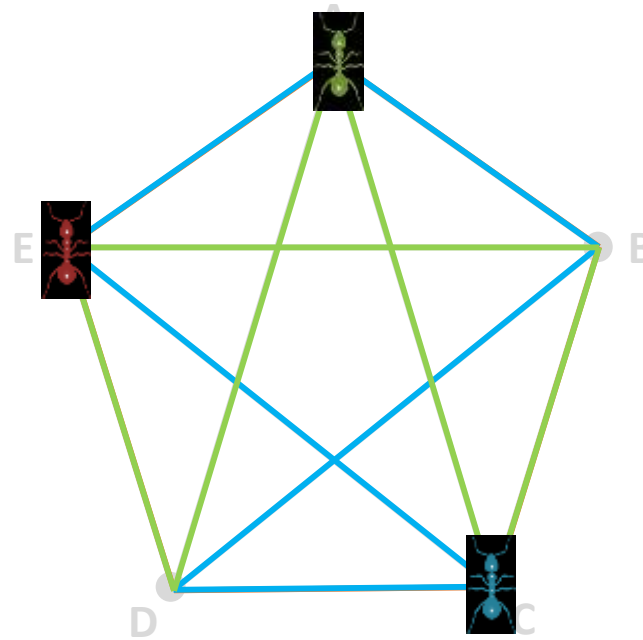


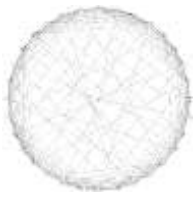
# TSP

For an undirected complete weighted graphs

## Algorithm - Step 1

Each ant preforms a Hamiltonian cycle according to the **navigation function**  $F(s)$ , while applying the **local update rule** for each visited edge.



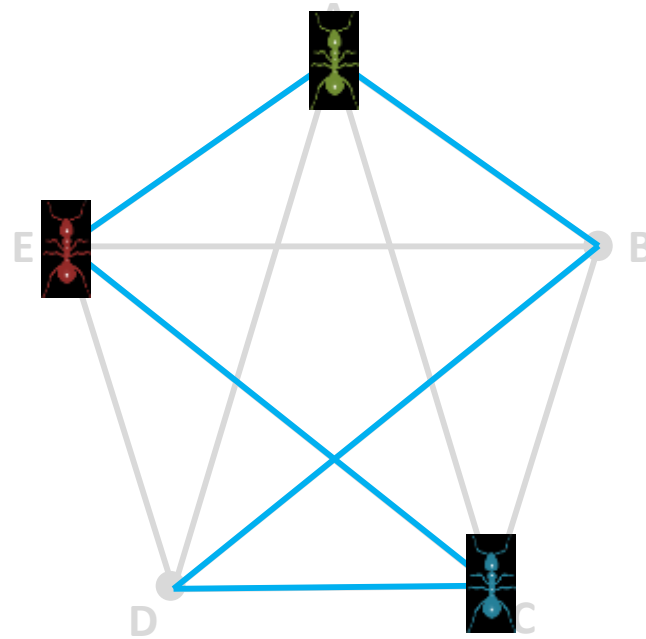


# TSP

For an undirected complete weighted graphs

Algorithm - Step 2

Apply the **Global Update Rule**.



# ACO algorithms

- Ant System (AS)
- Elitist Ant System (EAS)
- Rank-Based Ant System ( $AS_{rank}$ )
- Min-Max Ant System (MMAS)
- Ant Colony System (ACS)
- Approximate Nondeterministic Tree Search (ANTS)
- Hyper-Cube Framework for ACO

# Ant System (AS)

- $m$  ants concurrently build tour.
- Pheromone initialized to  $m/C^{nn}$ .
- Ants initially in randomly chosen sites.
- Random proportional rule used to decide which city to visit next

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \quad \text{if } j \in \mathcal{N}_i^k$$

# Ant System (AS)

- Each ant  $k$  maintains a memory  $M^k$  for its neighborhood.
- After all ants have constructed their tours, the pheromone trails are updated.
- Pheromone evaporation:

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

# Ant System (AS)

- Pheromone update:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k, \quad \forall (i, j) \in L$$

$$\Delta\tau_{ij}^k = \begin{cases} 1/C^k, & \text{if arc } (i, j) \text{ belongs to } T^k; \\ 0, & \text{otherwise;} \end{cases}$$



# Elitist Ant System (EAS)

- First improvement on AS.
- Provide strong additional reinforcement to the arcs belonging to the best tour found since the start of the algorithm.

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

# Rank-Based Ant System ( $AS_{\text{rank}}$ )

- Another improvement over AS.
- Each ant deposits an amount of pheromone that decreases with its rank.
- In each iteration, only the best  $(w-1)$  ranked ants and the best-so-far ant are allowed to deposit pheromone.

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

# Min-Max Ant System (MMAS)

- Four modifications with respect to AS.
  - Strongly exploits the best tours found.
    - This may lead to stagnation. So...
  - Limits the possible range of pheromone values.
  - Pheromone values initialized to upper limit.
  - Pheromone values are reinitialized when system approaches stagnation.

# Min-Max Ant System (MMAS)

- After all ants construct a solution, pheromone values are updated.  
(Evaporation is the same as in AS)

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}$$

- Lower and upper limits on pheromones limit the probability of selecting a city.
- Initial pheromone values are set to the upper limit, resulting in initial exploration.
- Occasionally pheromones are reinitialized.

# Ant Colony System (ACS)

- Uses ideas not included in the original AS.
- Differs from AS in three main points:
  - Exploits the accumulated search experience more strongly than AS.
  - Pheromone evaporation and deposit take place only on the best-so-far tour.
  - Each time an ant uses an arc, some pheromone is removed from the arc.

# Ant Colony System (ACS)

- Pseudorandom proportional rule used to decide which city to visit next.

$$j = \begin{cases} \operatorname{argmax}_{l \in \mathcal{N}_i^k} \{ \tau_{il} [\eta_{il}]^\beta \}, & \text{if } q \leq q_0; \\ J, & \text{otherwise;} \end{cases}$$

- Only best-so-far ant adds pheromone after each iteration. Evaporation and deposit only apply to best-so-far.

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

# Ant Colony System (ACS)

- The previous pheromone update was global. Each ant in ACS also uses a local update that is applied after crossing an arc.

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

- Makes arc less desirable for following ants, increasing exploration.

# Approximate Nondeterministic Tree Search (ANTS)

- Uses ideas not included in the original AS.
- Not applied to TSP.
- Computes lower bounds on the completion of a partial solution to define the heuristic information that is used by each ant during the solution construction.
- Creates a dynamic heuristic where the lower the estimate the more attractive the path.



# Approximate Nondeterministic Tree Search (ANTS)

- Two modifications with respect to AS:
  - Use of a novel action choice rule.

$$p_{ij}^k = \frac{\zeta \tau_{ij} + (1 - \zeta) \eta_{ij}}{\sum_{l \in \mathcal{N}_i^k} \zeta \tau_{il} + (1 - \zeta) \eta_{il}}, \quad \text{if } j \in \mathcal{N}_i^k$$

- Modified pheromone trail update rule. (No explicit pheromone evaporation)

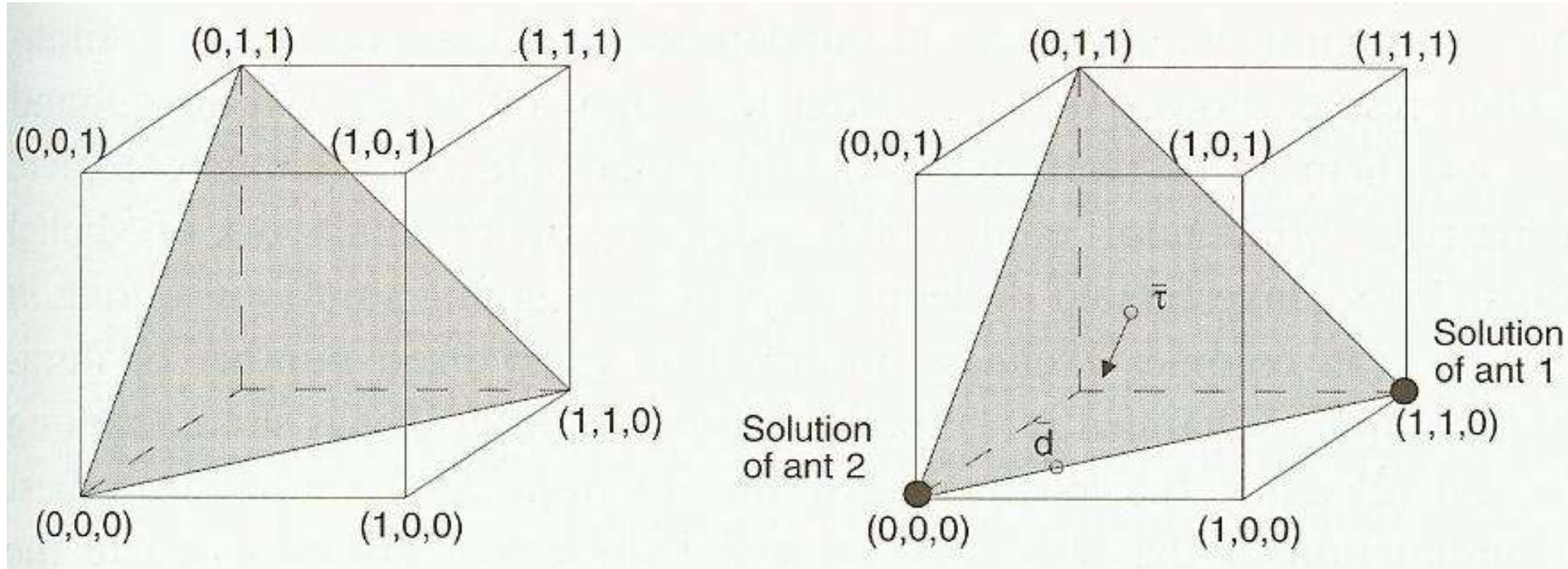
$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^m \Delta \tau_{ij}^k$$

$$\Delta \tau_{ij}^k = \begin{cases} \vartheta \left( 1 - \frac{C^k - LB}{L_{\text{avg}} - LB} \right), & \text{if arc } (i, j) \text{ belongs to } T^k; \\ 0, & \text{otherwise;} \end{cases}$$

# Hyper-cube Framework for ACO

- Uses ideas not included in the original AS.
- Not applied to TSP.
- Automatically rescales the pheromone values for them to lie always in the interval  $[0,1]$ .
- Decision variables  $\{0, 1\}$  typically correspond to the components used by the ants for construction.
- A solution problem then corresponds to one corner of the  $n$ -dimensional hyper-cube, where  $n$  is the number of decision variables.

# Hyper-cube Framework for ACO



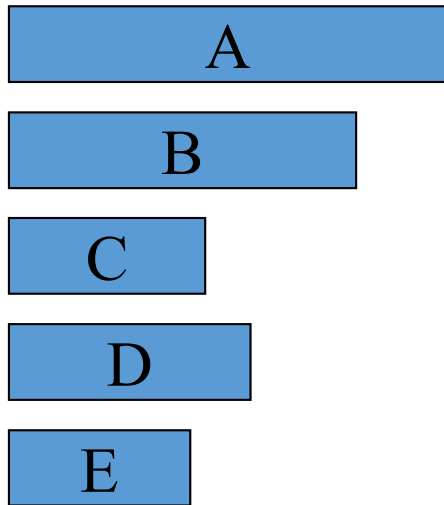
# Not just for TSP of course

- 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.

# Example:

## Single machine scheduling with due-dates

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

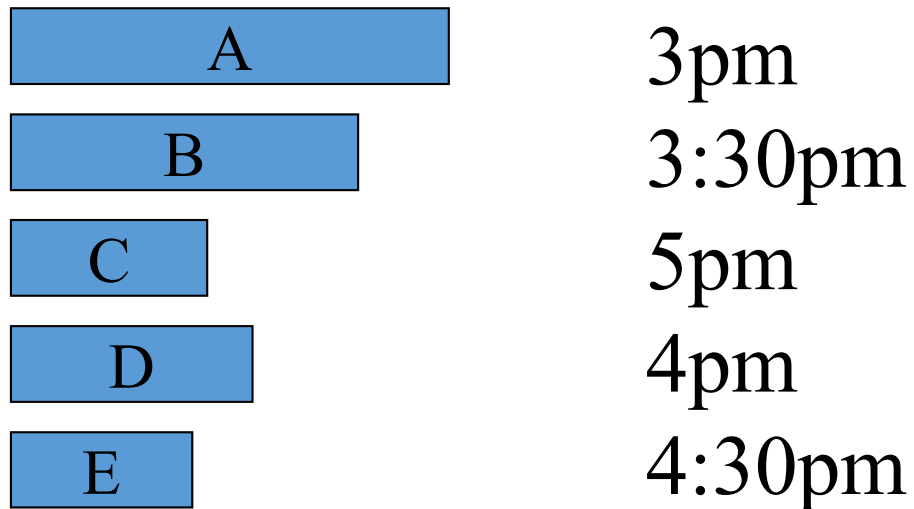


# E.g.

## 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

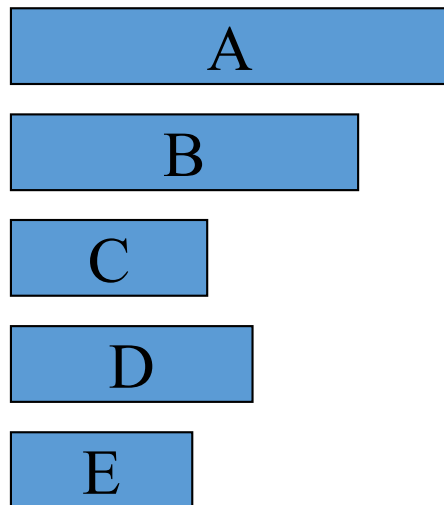


# E.g.

## 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

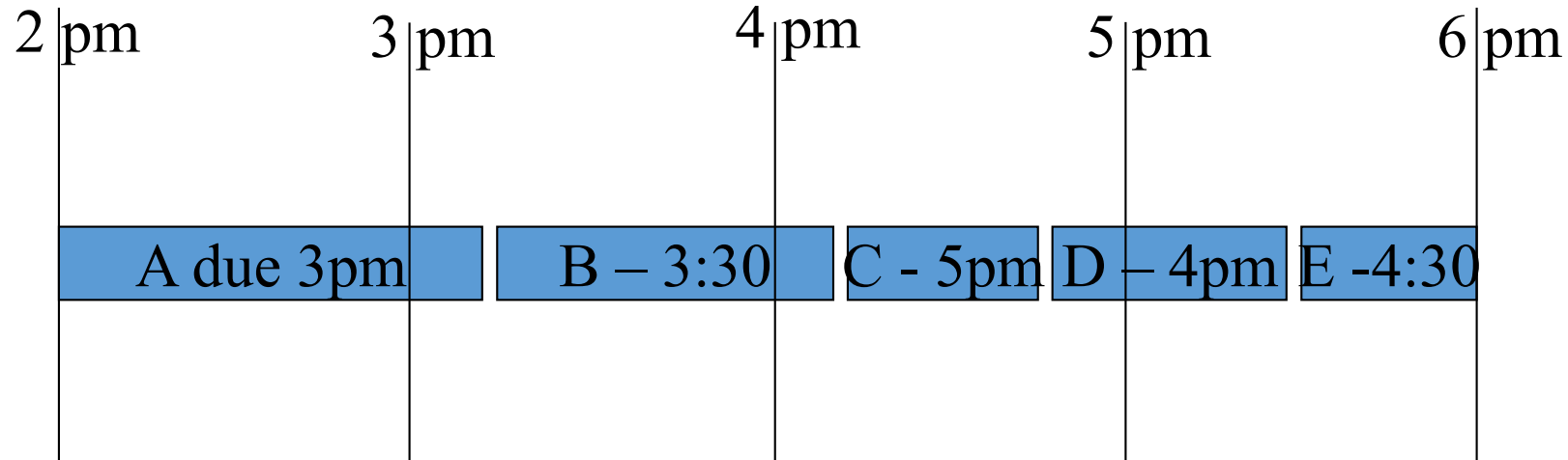
4pm

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

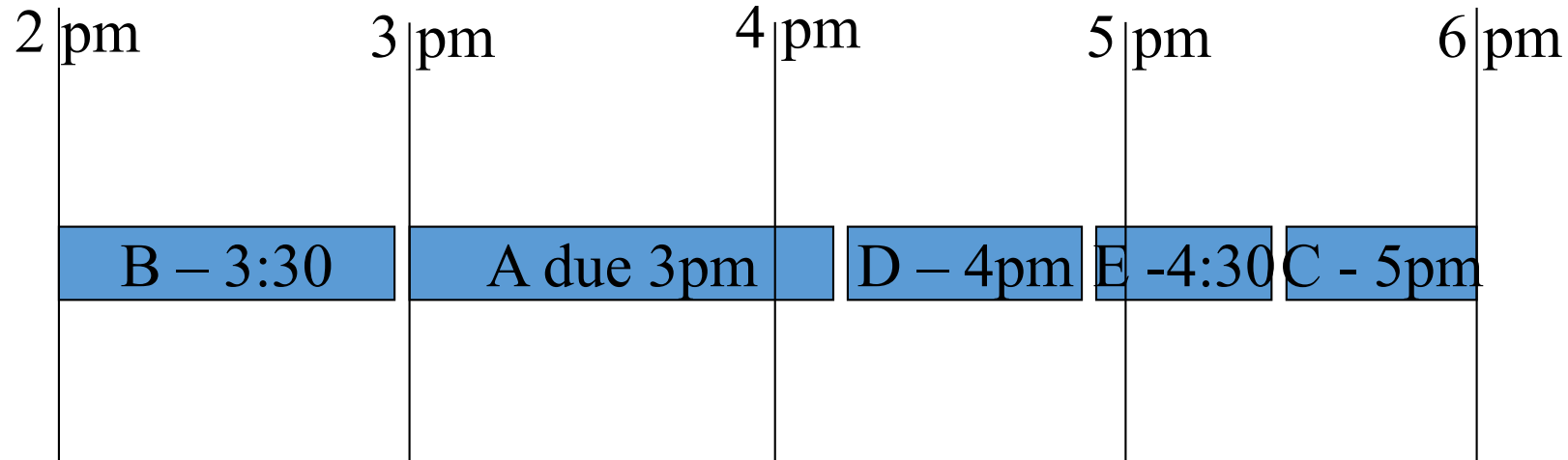
E is 90min late

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

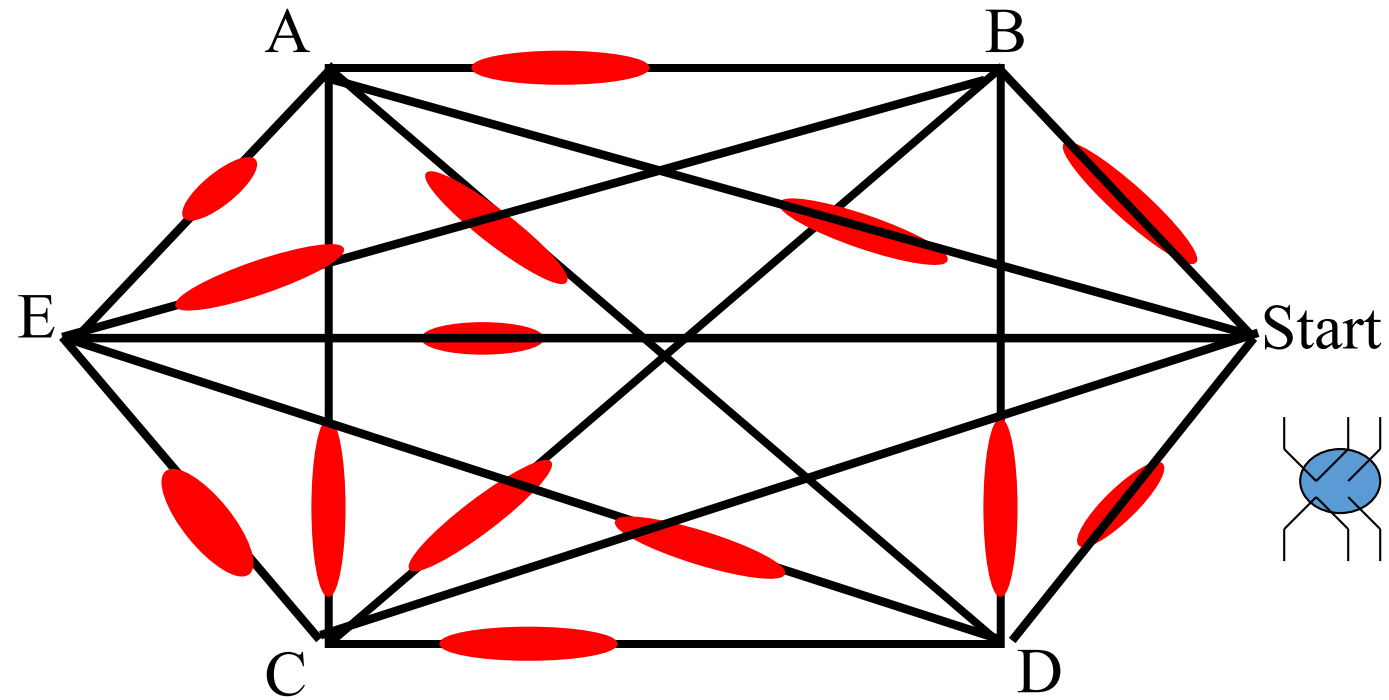
Fitness might be average lateness;  
in this case again 46min

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

# Applying ACO to this problem

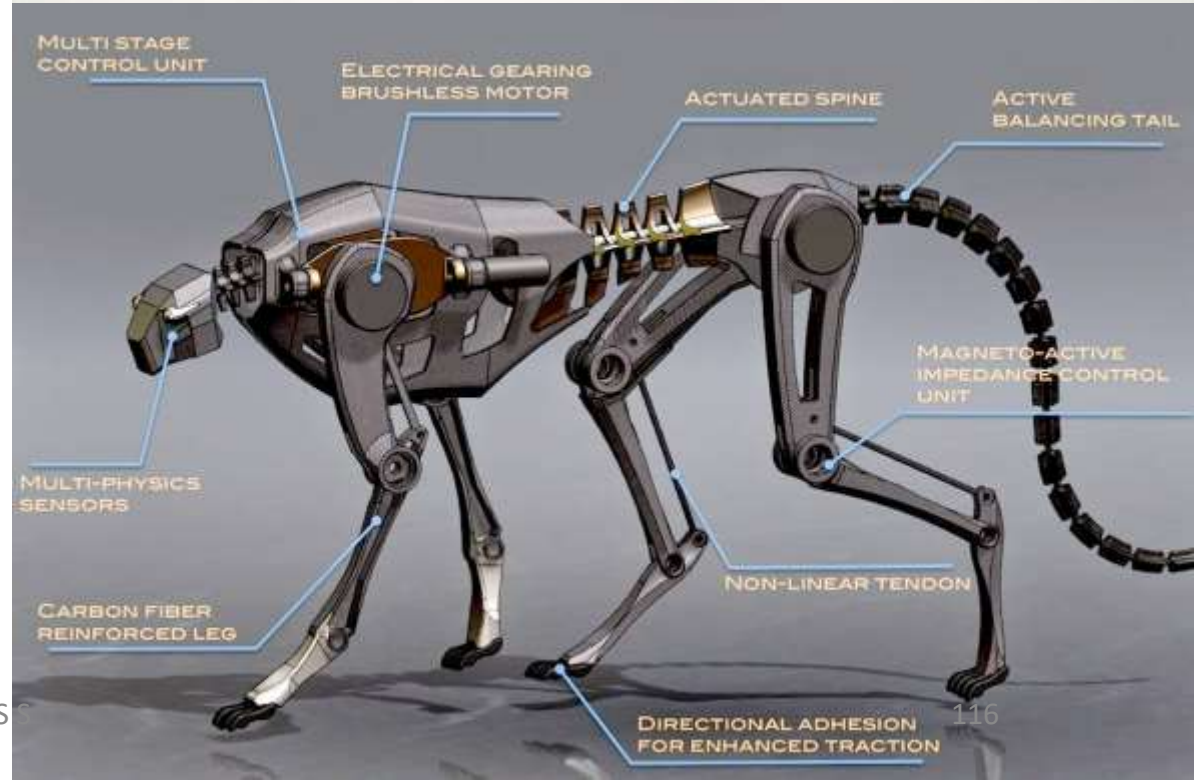
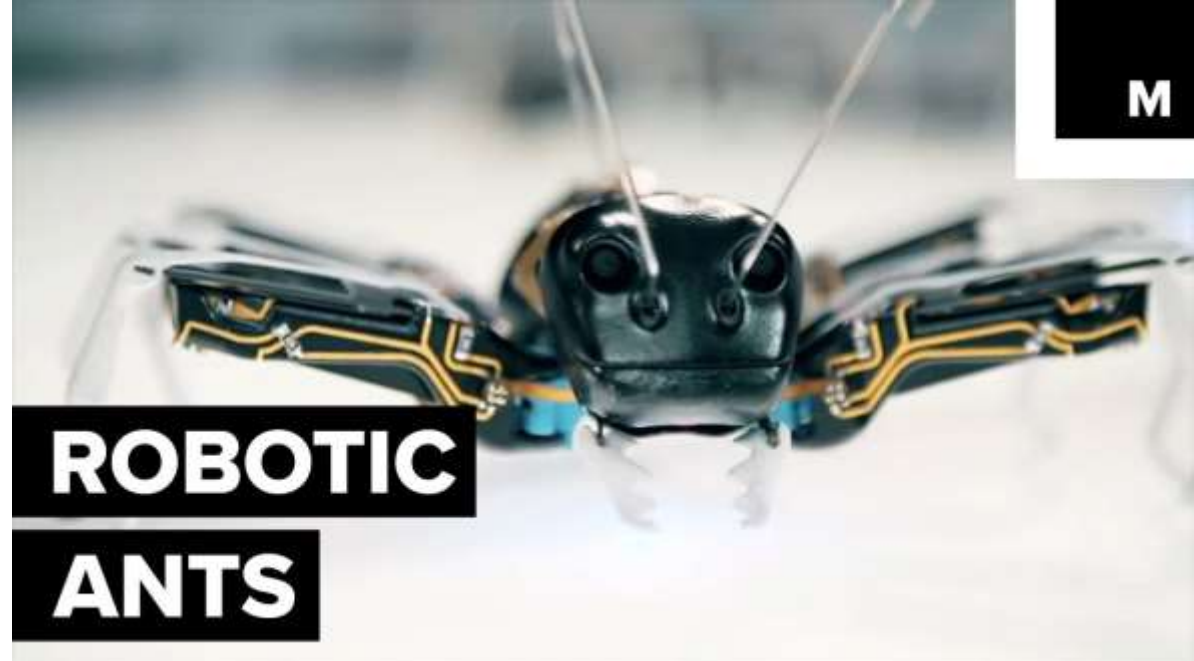
- Just like with the TSP, each ant finds paths in a network, where, in this case, each job is a node
- No need to return to start node - path is complete when every node is visited.

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?

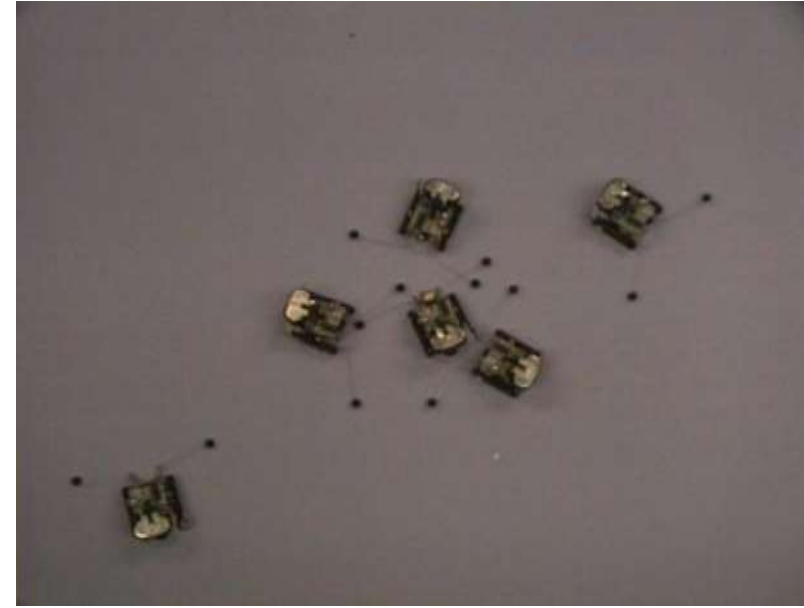
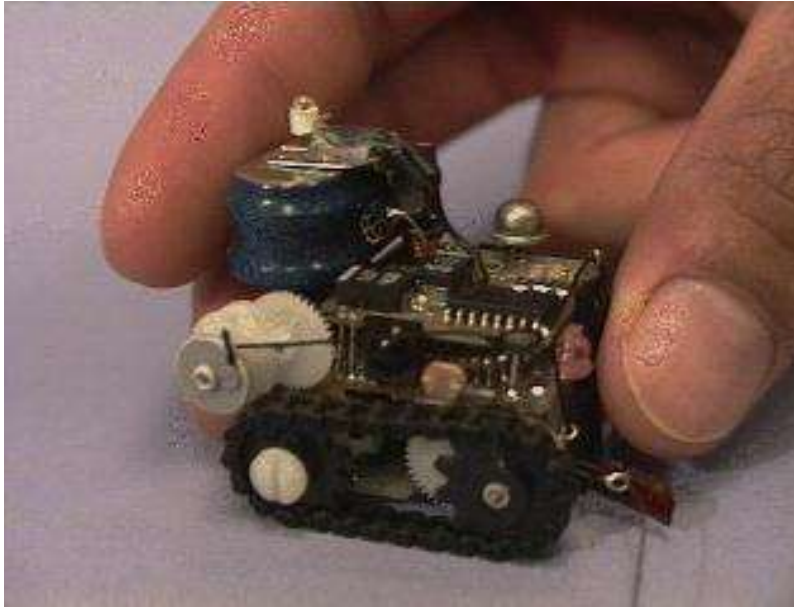


Why is it sensible to have a Start node for this problem

In real world



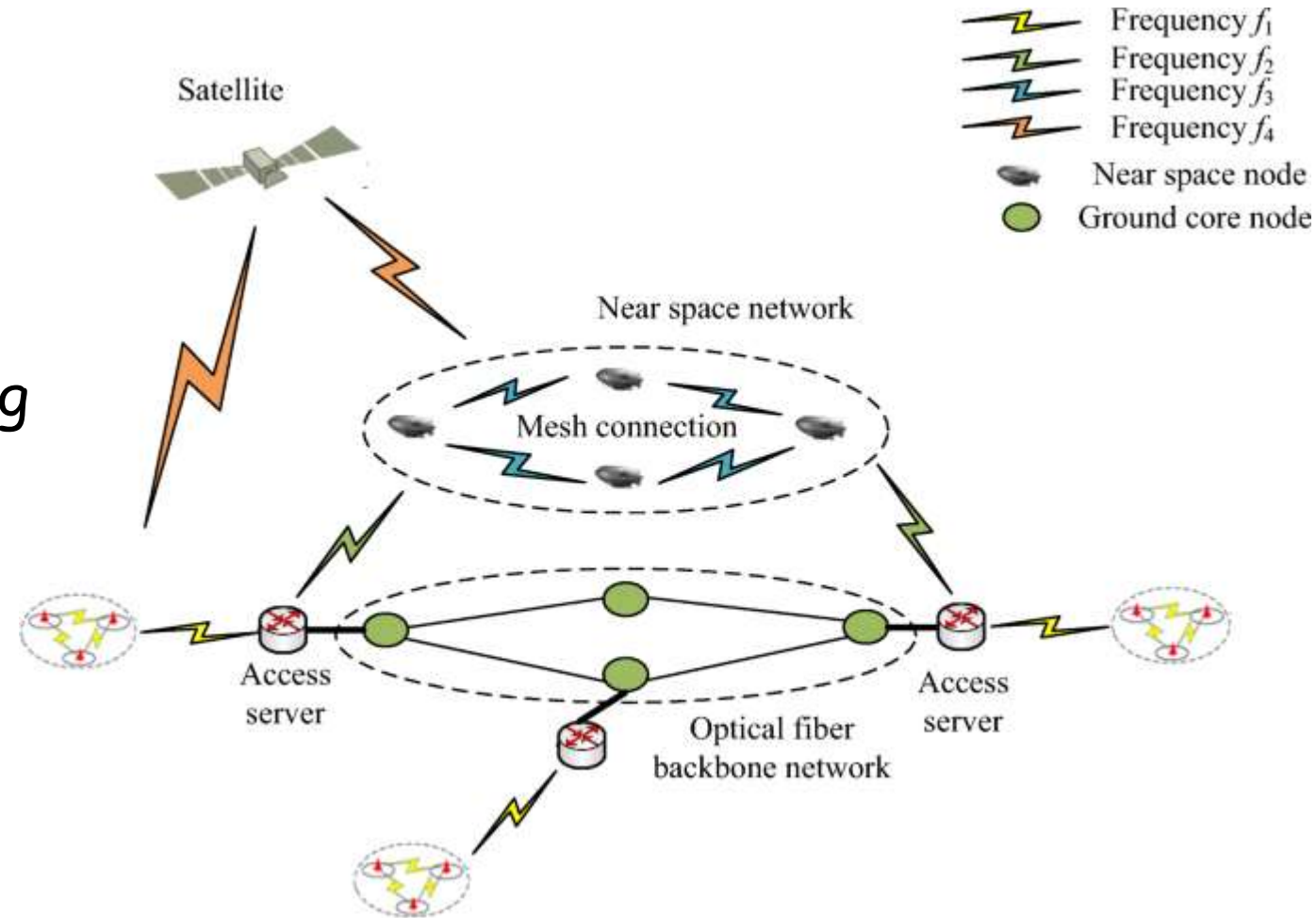
# Ant robots



- Collective task completion
- No need for overly complex algorithms
- Adaptable to changing environment

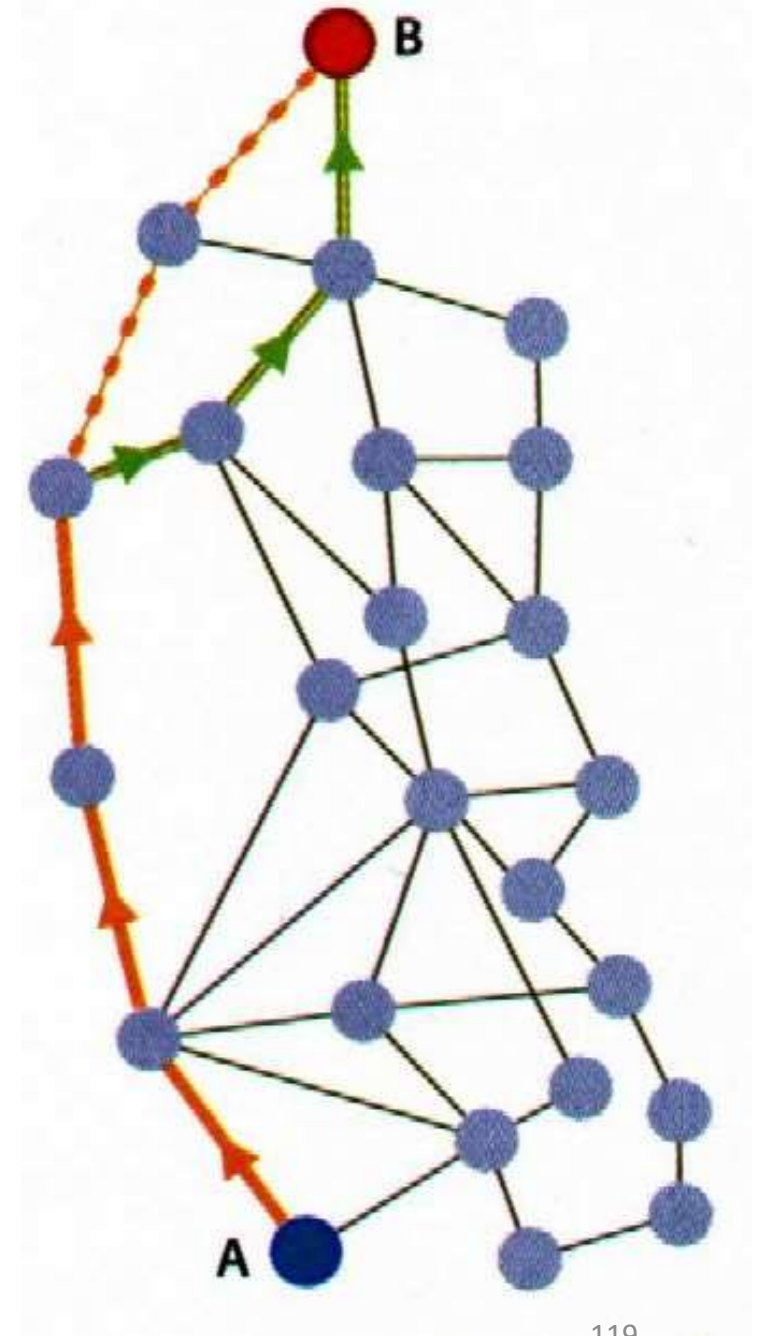
# Communication networks

- Routing packets to destination in shortest time
- Similar to Shortest Route Statistics kept from prior routing (learning from experience)



# Communication networks

- Shortest Route
- Congestion
- Adaptability
- Flexibility



# Antiflying website searching

- Digital-Information Pheromones (DIPs)
- Ant world server
- Transform the web into a gigANTic neural net



# The future?

Miniaturization

Pipe Inspection

Satellite  
Maintenance

Job Scheduling

Interacting Chips in  
Mundane Objects

Pest Eradication

Vehicle Routing

Optimal Resource  
Allocation

Data Clustering

Distributed Mail  
Systems

Engine Maintenance

Cleaning Ship Hulls

Self-Assembling Robotic  
Combinatorial  
Optimization

Medical

Telecommunications



*to be continued*