Master of Technology in Knowledge Engineering

Unit 7: Developing Intelligent Systems for Performing Business Analytics

Swarm Intelligence : A Different Perspective of Distributed Intelligent Systems

Fan Zhenzhen
Institute of Systems Science
National University of Singapore
email: zhenzhen@nus.edu.sg

Agenda

- Swarm Intelligence
- The Objective: Self-Organising behavior
- Examples in Nature: Ants, Wasps, Termites, Birds
- 7 principles of highly effective
- Example applications of Swarm Intelligence

Swarm Intelligence

The collective behaviors of simple individuals interacting with their environment and each other

Emergence

– Unique global behavior arising from the interaction of many agents

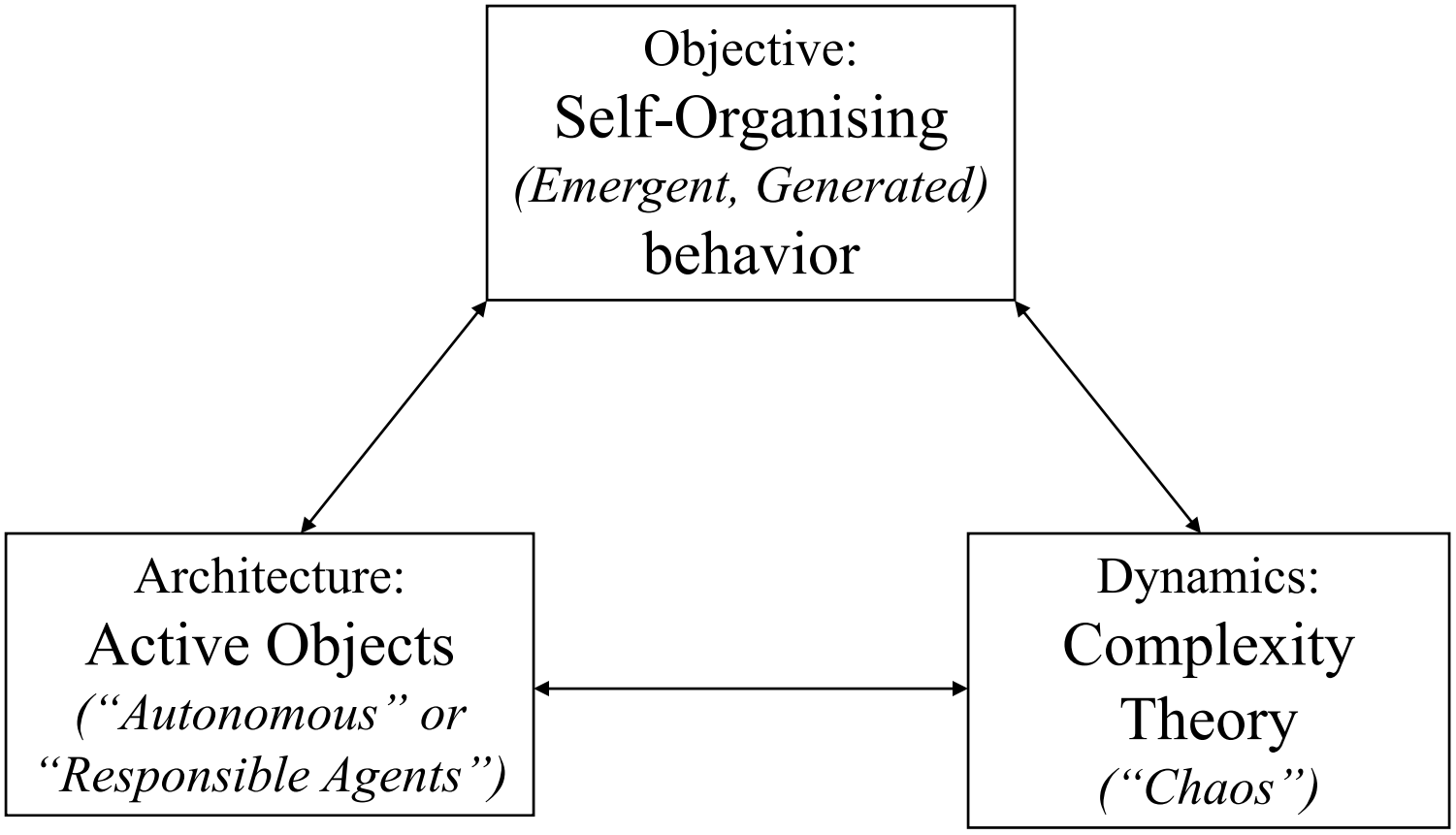
Stigmergy

– Indirect communication through the environment

Suitable for problem domains that

- Do not require precise control over how a goal is achieved
- Require many simple agents
- Need robust behaviors

Context



The Objective: Self-Organising behavior

- Simpler to design agent than system for comparable overall richness, e.g., 100 agents with 10 behaviors each
  - Design 10*100 = 1000 individual behaviors
  - System yields 10^{100} behaviors
- Robust to failure of any replicated entity (redundancy)
- Self-configuring on startup
- Self-adjusting as environment changes

Examples in Nature

- Ants: Path planning
- Ants: Brood sorting
- Army Ants: Transport teaming
- Termites: Nest building
- Wasps: Task differentiation
- Birds & Fish: Flocking
- Wolves: Surrounding prey
- People: Market economy
- Taxicabs: Vehicle distribution

- No central planning
- Simple individual rules
- Complex adaptive behavior

Models for Automation

(Intelligent Agent) Systems?

We are not yet able to build the intelligence of a human into our machines.

But lesser intelligences (e.g., ants) can do pretty impressive things.

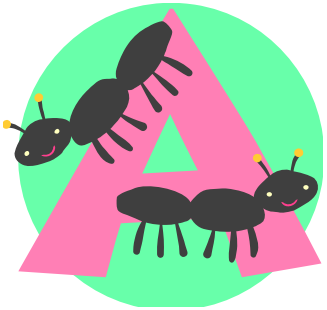
Intelligent (Agent Systems):

Perhaps we can build in the intelligence of an insect!

Schema for Examples

- *System behavior* – What purposeful action emerges from the overall system?
- *Individual Responsibilities* – What rules do individual agents follow in interacting with the environment?
- *Integration* – How does individual exercise of these responsibilities yield the system phenomenon?

Ant Path Planning: System Behavior



nts construct networks of paths among food sources and their nests. These networks form minimum spanning trees …

… but no ant computes the tree!

Ants in real life Ants in Second Life

Ant Path Planning: Individual Responsibilities

- Avoid obstacles
- Wander randomly, in direction of any nearby pheromones
- If not holding food,
  - drop Nest Pheromone at a decreasing rate
  - climb Food Pheromone gradient
- If holding food,
  - drop Food Pheromone at a decreasing rate
  - climb Nest Pheromone gradient
- If at food and not holding any, pick it up
- If at nest and carrying food, drop it

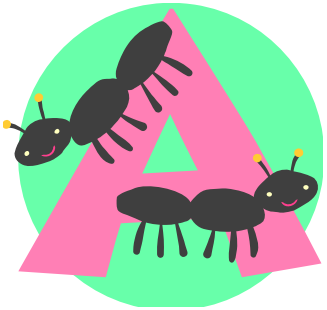
Ant Path Planning: Integration

- Nest Pheromones and Food Pheromones lead to nests and food sources, respectively
- Once these two fields intersect, they attract ants of the opposite state, and condense into paths
- Because pheromones decay through time, paths to depleted food sources disappear
- Stochasticity & width of pheromone line tend to straighten out squiggles

Pheromone

"A chemical secreted by an animal, especially an insect, that influences the behavior or development of others of the same species, often functioning as an attractant of the opposite sex."

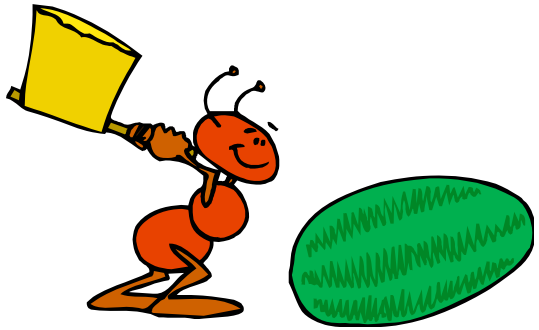
Ant Nest Sorting: System behavior



nt colonies sort out larvae, eggs, cocoons, and food …

… but no ant is running a sorting algorithm!

Ant Nest Sorting: Individual Responsibilities



- Wander around the nest
- Sense presence of objects
- Maintain short memory (about ten steps) of what it has seen
- If not carrying anything, probability of picking up sensed object decreases if it has recently encountered similar objects in the environment
- If carrying something, probability of dropping it increases if it has recently encountered similar items in the environment

Ant Nest Sorting: Integration

- Local concentrations of similar items occur naturally through random scattering
- Once a concentration exists, its presence helps retain current members & attract new ones

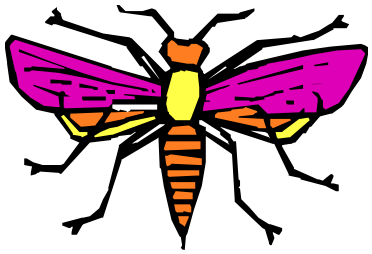
Wasp Task Differentiation: System behavior

Mature wasps in a nest divide into 3 groups: a single chief, a group of foragers who hunt for food, and a group of nurses who care for the brood. The relative proportion of foragers & nurses varies with the abundance of food & the size of the brood …

… but no wasp (not even the chief!) computes what this proportion should be!

Wasp Task Differentiation: Ind. Responsibilities

- Each wasp maintains two parameters:
  - » Force (determines position in hierarchy & degree of mobility)
  - » Foraging threshold (determines how readily the wasp will go hunt)
- Brood maintains a demand parameter that determines the degree of a stimulus it exerts on foragers



Wasp Task Differentiation: Integration

- When a wasp is near the brood:
  - » Its foraging threshold drops
  - » It forages probabilistically depending on its threshold
- When the brood gets food, its demand drops
- When a wasp does not forage, its threshold increases
- When two wasps meet:
  - » Relative hierarchy determined probabilistically by their respective forces
  - » Submissive wasp loses force; dominant wasp gains

Wasp Task Differentiation: Integration

	Force	Threshold
Chief	High	High
Foragers	High	Low
Nurses	Low	High

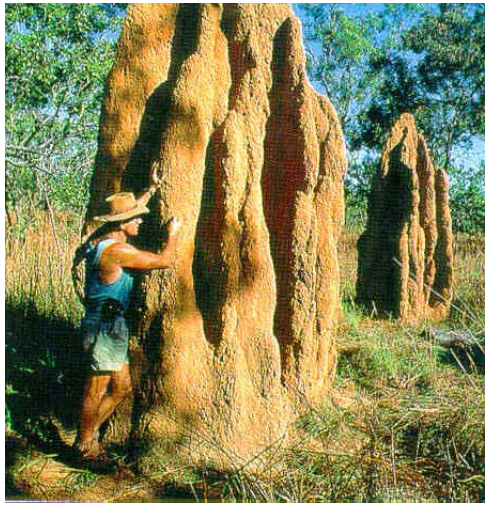
Termite Nest Building: System behaviour

Tropical termites construct hills (>5m high, >10tons) with multiple floors containing intricate networks of passageways & chambers that are used to store food, raise the brood, & house the population

… but there's not a chief engineer among them!

Image from: http://www.stanford.edu/~siegelr/tz/tz2005/termitemound.jpg

Termite Mounds!!!

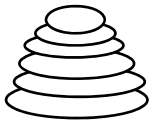


Termite Nest Building: Ind. Responsibilities

- Metabolise bodily waste (which contains pheromones)
- Move toward strongest local pheromone concentration
- Drop waste with probability proportional to
  - » Pheromone density
  - » Amount of waste carried

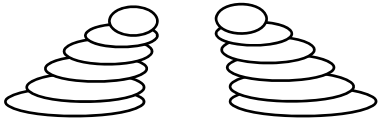
Termite Nest Building: Integration

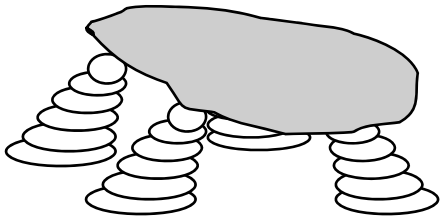
Early deposits attract later ones, yielding ***concentrations***



Newest (i.e., strongest) pheromones on top lead to columns

Nearby columns lean toward one another, building arches



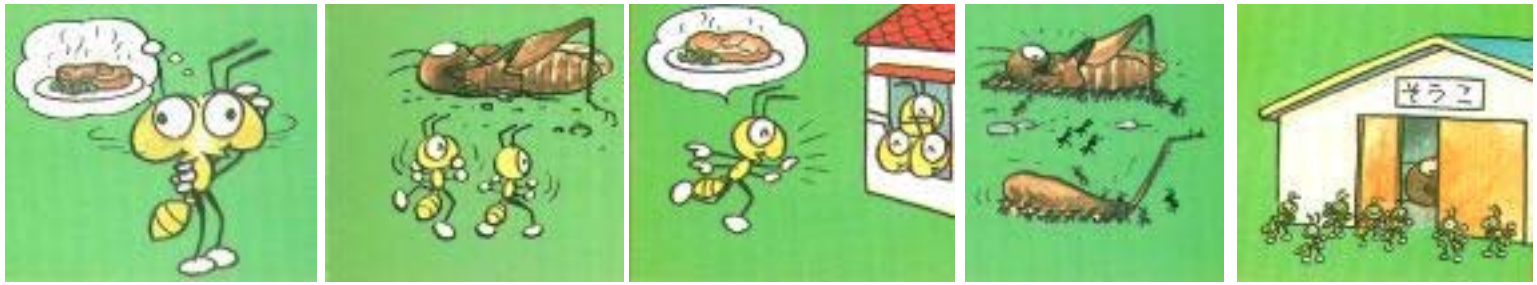


Multiple arches contact to form ***floors***

Cooperative Transport: System Behavior

A team of ants can carry a load 50x their aggregate weight (& no balancing problem!!!), while a single ant can carry only 5x its individual weight

… but there's no foreman to allocate the ants to the optimal positions on the load



Cooperative Transport: Ind. Responsibilities

Nested behaviors:

- Search for food
- Carry the food if possible
- Drag the food if unable to carry
- Try another position or direction if unsuccessful
- Recruit nestmates if still unsuccessful after certain time

Cooperative Transport: Integration

- Ants recruit nestmates by
  - » Releasing gland secretion in the air (short-range recruitment)
  - » Laying a chemical trail with gland secretion from the prey to the nest (long-range recuitment)
- The group size is determined by the difficulty encountered in moving the prey
- Deadlock and stagnation recovery is achieved through individual realigning and repositioning behaviors

Bird Flocking: System behavior

Flocks of birds stay together, coordinate turns, & avoid collisions with obstacles & each other …

… without an air traffic control system!

Bird Flocking

Individual Responsibilities

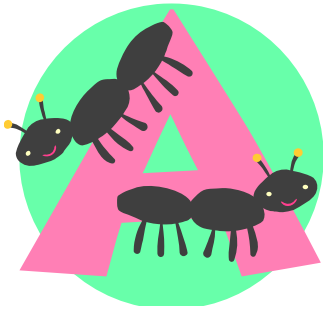
- Maintain minimum separation from nearest object
- Match velocity (magnitude and direction) to nearby birds
- Try to stay close to the centre of the flock

Integration

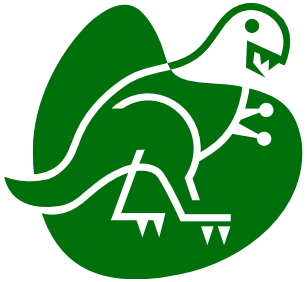
- Each bird's individual actions simultaneously
  - respond to and
  - change

  the overall structure of the flock

Which is more liable to extinction?



The dominant biological family on earth by total body mass is the ants!



Seven Principles of Highly Effective MAS

1. Agent things, not functions
2. Keep agents small
3. Decentralise system control
4. Support agent diversity
5. Control entropy
6. Enable agents to share information
7. Plan & execute concurrently

1. Agent things, not functions

- Watch out for old SE instincts toward "functional decomposition"!
- Divide agents on the basis of distinct entities in the physical world rather than functional abstractions
- A clear distinction between agent & environment, & a recognition that the environment mediates agent interactions, force the designer to pay attention to topological boundaries between agents & environment

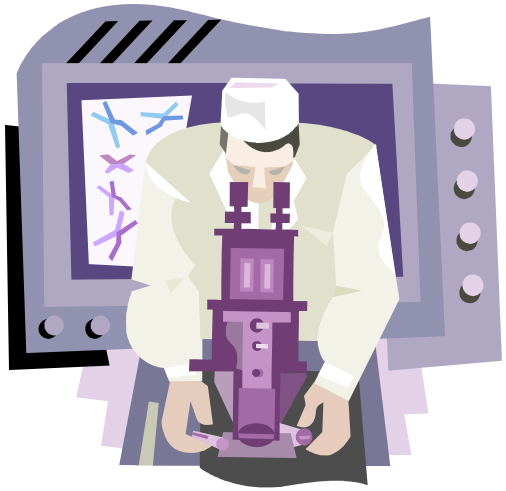
1. Agent things, not functions *(cont.)*

Two important exceptions:

- Legacy systems: industrial agents that are additions to existing systems need to interface with legacy systems (which could be functionally oriented)
- Watchdog agent: to monitor the behavior of a population of physical agents – sense conditions & raise signals; NOT plan or take action!

2. Keep agents small

- In Mass: Small compared with the whole system → agents are numerous
  - » Inexpensive to implement
  - » Easy to understand
  - » Limited impact of unit failure
- In Time: Bounded (forgetful)
  - » Example: pheromone evaporation
  - » Able to adapt to environmental changes



- In Space: Local sensing & action
  - » Global information → processing overload
  - » Self-organisation works better with sparse interconnectivity
  - » Emergent long-range interactions are more robust & agile than built-in ones

Local vs. Global?

Question: Won't the sum of locally optimal decisions often be suboptimal globally?

Answer:

- Strategy: robustness under continual change is worth more than a theoretical optimum in a steady-state that we never reach
- Tactics: life-cycle system costs are lower
- Theory: there are decentralised mechanisms to achieve global coordination
- Track record: SI systems outperform conventional ones!

3. Decentralise system control

(e.g., no king ant)

Why did we centralise computation before?

- Computers were expensive
- Computers were environmentally sensitive
- No way to understand generated behavior

Why not centralise it now?

- Single point of failure
- Performance bottleneck
- Difficult to modify
- Attracts functionality

"Control from the bottom up!"

4. Support Agent Diversity

What is diversity?

- In our examples: lots of {ants, wasps} at different places
- Role differentiation in wasps
- General ecosystem: lots of different kinds of organisms

Agent Diversity: Why?

- Agents are bounded processes that are interacting through a shared environment
- Environment state space >>> agent state space
- Community survival depends on tracking & responding to environmental state
- Agents spread out their coverage
- Adaptive critical populations >> “rational requirements”

  » Example: need more ants than (required amount of food) / (carrying capacity of one ant), in order to explore different regions of food

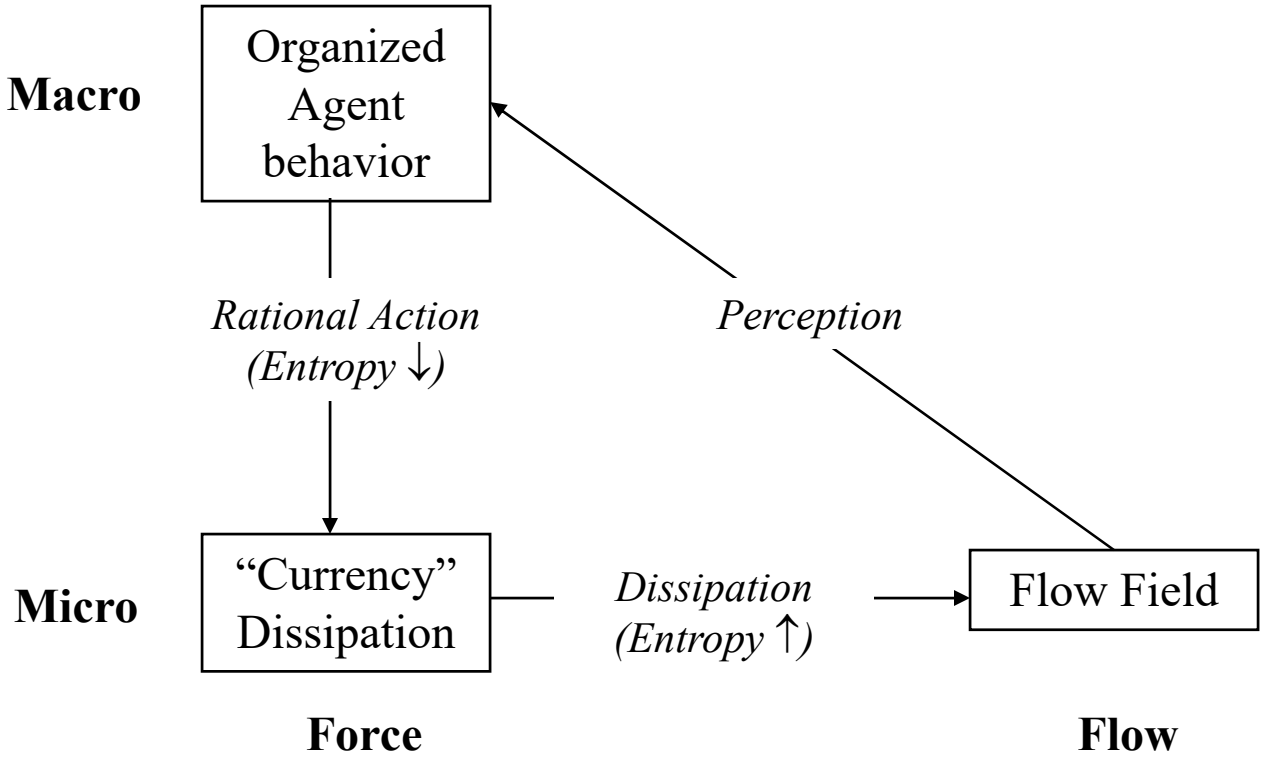
=> Diversity is the cost of adapting to change

Agent Diversity: How?

- Randomness
  - » less dependent on explicit model of the domain
  - » simple processing
- Repulsion: Draw on physical exclusion laws
  - » e.g., wolf & bird examples – diversity of location

5. Control Entropy

- Second Law of Thermodynamics: Overall disorder increases
- If we want self-organisation at macro level, must dissipate order at lower level



5. Control Entropy *(cont.)*

- Natural self-organising systems provide an "entropy leak" to a micro level
  - pheramone dissipation in insect communities
  - currency flow in economies
  - energy flow in plant growth
- Conclusion: engineer a "currency" or "energy" that generates a field to which agents can align

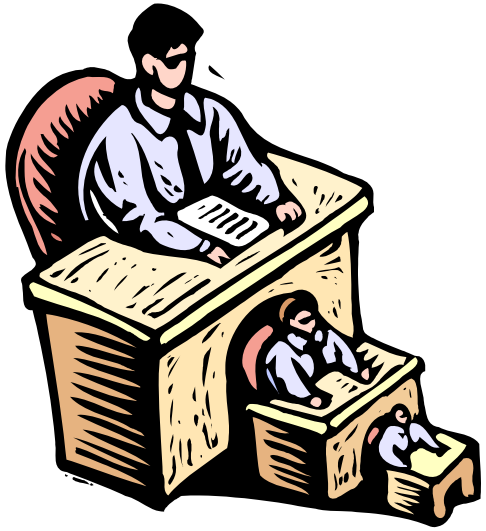
6. Enable Agents to Share Information

In a sense, this is multi-agent learning!

	Ontogenetic	**Phylogenetic**	**Sociogenetic**
Learner	Individual	Species	Society
Mechanism	Manipulate internal representation	Differential reproduction	External signals
Example	This lecture	Moths change colour	Pheromone routes
Technology (examples)	Classical AI; NN	Genetic Algorithms	Market-oriented Programming; NN
Sophistication needed	High	Low	Low

7. Plan & Execute Concurrently

- Avoid the "plan then execute" mode of operation
- Respond dynamically to changes in the environment - replan



Example SI Applications

Technique	Application
Pheromone paths	Packet routing
Wasp division of labour	Truck painting robots
Nest sorting	Data Analysis
Nest construction	Self-assembling robots
Cooperative transport	Robotic transportation systems
Bird & fish flocking	Particle swarm optimization

1. Pheromone Path: Packet Routing

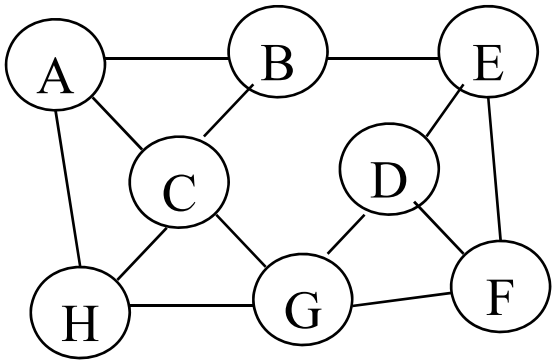
Objective: To direct traffic from sources to destinations maximizing network performance while minimizing costs

Packets of info in a telecom network can follow alternative paths between nodes

Path vary in their capacity, congestion, & length

Nodes maintain a routing table that tells a packet destined for A which next step to take

Problem: Maintain the routing table dynamically, to reflect changing network conditions



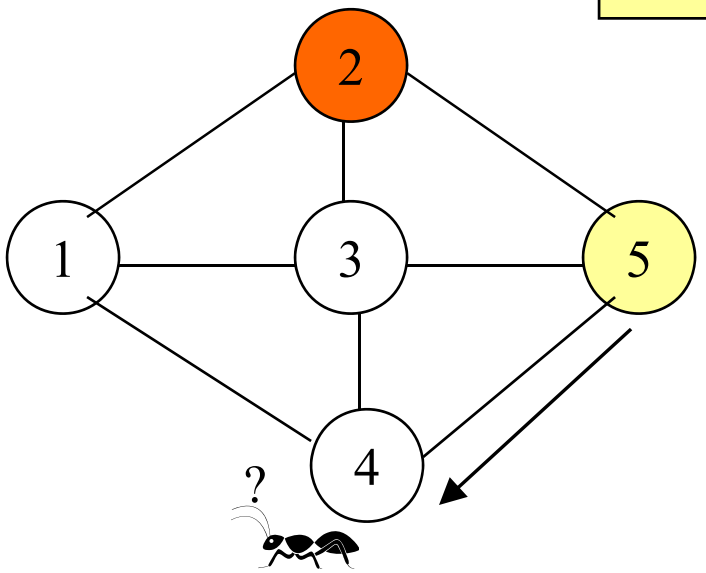
Packet Routing

- There exist many routing algorithms
  - » Ant-Based Control
  - » AntNet
  - » Mobile Ants Based Routing
  - » Ant Colony Based Routing Algorithm
  - » Termite

Packet Routing : Ant-Based Control

- The first SI routing algorithm for telecom networks
- Use of many simple agents (ants) to modify the routing policy at each node by depositing a virtual pheromone trail on routing table entries
- Ants are launched regularly from any node to random destinations in the network, while actual calls are placed onto the network after the routes are established
- The next node an ant will move to is selected probabilistically according to the routing table of its current node
- Each visited node's routing table is updated
- An ant is deleted from the network when it reaches its destination node

Packet Routing : Ant-Based Control



Determine its next step

Destination nodes

Neighbor nodes

	1	2	3	5
1	0.8	0.3	0.1	0.1
3	0.1	0.4	0.8	0.1
5	0.1	0.3	0.1	0.8

Routing table for node4

An ant (destination=node2, source=node5) arrives in node4 from node5

Example

Reinforce its previous step

Packet Routing : Ant-Based Control

- Updating of routing table at current node *i* by an ant from source node *s*; the node just visited is *i-1*
- Reinforce the entry $p_{i-1,s}$ (from previous node to source node)

$$p = \frac{p_{old} + \Delta p}{1 + \Delta p}$$

- Other entries $p_{n,s}$ $(n \neq i-1)$ decay

$$p = \frac{p_{old}}{1 + \Delta p}$$

- Δp decreases with age of the packet
- Therefore the updated values of probabilities for destination 5 in the example is $0.1/(1+\Delta p)$, $0.1/(1+\Delta p)$, $(0.8 + \Delta p)/(1+\Delta p)$

Packet Routing : Ant-Based Control

- Effect
  - » routes which are visited frequently and by “young” ants will be favored routes when building paths to route new calls
  - » Links close to the source node get higher reinforcement than links that are far away => shorter path

2. Wasp Division of Labour

Two fundamental principles:

- If you're not busy, find something useful to do
- If you're already doing something, do more of the same thing rather than changing tasks

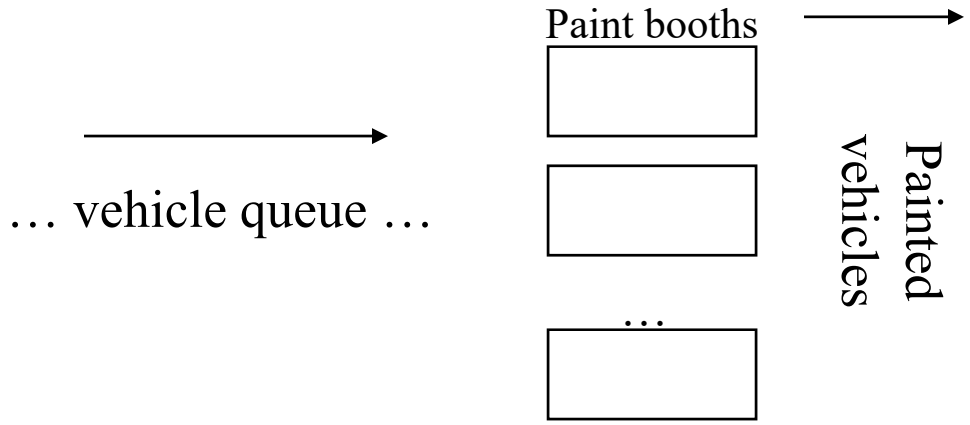
Similar to market-based algorithms in the sense that:

low threshold = high bid

high threshold = low bid

Applied by Flavors Technology to assignment of paint booths to truck bodies

Application: GM Paint Booth Scheduling



- To assign trucks to paint booths
- To minimize color changeover at a booth which causes:
  - » the booth to be down for a few minutes
  - » high cost

GM Paint Booth Scheduling

For each paint booth:

- The stimulus – an arriving truck
- The response threshold is **low** if
  - » The truck is the same color as its current color
  - » The job is urgent
  - » The queue is empty or small
- The threshold is **high** if
  - » The truck's color is different
  - » The queue is large
  - » The booth is down (infinite)
- Thresholds determine magnitude of bids

GM Paint Booth Scheduling

Benefits, compare with conventional systems:

- Shorter, simpler code (maintained by electrician)
- Colour changeovers @ paint booths ↓50% (i.e., emergent booth specialisation) => Lower cost ($1M/year)
- More robust (less affected by the break-down of a booth)

3. Net Sorting

- Clustering and sorting is achieved through individuals moving randomly in space and pick up and deposit items on the basis of local information
- Can be applied to data analysis (sorting objects with different attributes), graph partitioning (sorting nodes of a graph), distributed clustering and sorting using robots

Deneubourg's Basic Model for Clustering

$$p_p = \left(\frac{k_1}{k_1 + f} \right)^2$$

- p_p – probability for a randomly moving, unladen agent (ant) to *pick up* an item
- f – the perceived fraction of items in the neighborhood (the number of items encountered during T time units / the largest possible number of items that can be encountered during T time units)
- k_1 – a threshold constant
- When $f \ll k_1$, p_p is close to 1 → Isolated items should be picked up
- When $f \gg k_1$, p_p is close to 0 → Items are unlikely to be removed from dense clusters

Basic Model for Clustering

$$p_d = \left(\frac{f}{k_2 + f} \right)^2$$

- p_d – probability for a randomly moving loaded agent to *deposit* an item
- k_2 – another threshold constant
- When $f \ll k_2$, p_d is close to 0 → Items are unlikely to be dropped when there are not many similar items in the neighborhood
- When $f \gg k_2$, p_d is close to 1 → Items are to be dropped where more items of that type are present

- Model for sorting is similar except using f_A and f_B, for example, to represent respective fractions of items of types A and B encountered during the last T time units

Application to Data Analysis

- Exploratory Data Analysis - to gain insight into a data set, to uncover underlying structures, etc.
- The basic clustering model can be extended to map high-dimensional data onto a two-dimensional representation, to make clusters appear
- Intra-cluster distances should be smaller than inter-cluster distances
- Example – the Lumer Faieta algorithm
- Define a distance or dissimilarity $d(o_i, o_j)$ between two objects o_i and o_j in the space of object attributes
- Define $f(o_i)$ as the measure of the average similarity of object o_i at site r with the other objects o_j present in the neighborhood of o_i.
- $f(o_i)$ Replaces the fraction f in the basic model

Lumer Faieta algorithm

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum_{o_j \in Neigh_{(s\times s)}(r)} \left[1 - \frac{d(o_i, o_j)}{\alpha} \right] & \text{if } f > 0, \\ 0 & \text{otherwise.} \end{cases}$$

- $Neigh_{(s\times s)}(r)$ – the region of area s^2 surrounding site r
- α - a factor that defines the scale for dissimilarity
  - » If too small, clusters may not form
  - » If too large, clusters may be composed of items that should not belong to the same cluster
- $\forall o_j \in Neigh_{(s\times s)}(r)$ $\quad d(o_i, o_j)=0 \rightarrow f(o_i)=1$
- $\forall o_j \in Neigh_{(s\times s)}(r)$ $\quad d(o_i, o_j)=d_{max} \rightarrow$ small $f(o_i)$
- No objects around $r \rightarrow f(o_i)=0$

Lumer Faieta algorithm

- Definition of picking up probability:

$$p_p(o_i) = \left(\frac{k_1}{k_1 + f(o_i)} \right)^2$$

$f(o_i)$=1 → low probability to be picked up

$f(o_i)$=0 → high probability to be picked up

- Definition of dropping probability:

$$p_d(o_i) = \begin{cases} 2f(o_i) & \text{if } f(o_i) < k_2 \\ 1 & \text{if } f(o_i) \geq k_2 \end{cases}$$

LF algorithm : High Level Description

Randomly distribute items & randomly distribute agents

For all agents **do**

if ((agent unladen) and (site occupied by item o_i)) **then**

Compute $f(o_i)$ and $p_p(o_i)$

Draw random real number R between 0 and 1

if ($R \leq p_p(o_i)$) **then**

pick up item o_i

end if

else if ((agent carrying item o_i) and (site empty)) **then**

compute $f(o_i)$ and $p_d(o_i)$

draw random real number R between 0 and 1

if ($R \leq p_d(o_i)$) **then**

drop item

end if

end if

move to randomly selected neighboring site not occupied by other agent

End For

LF Algorithm

- Applied to a database containing the profiles of 1650 bank customers and found interesting corelations
- A simple way of implementing multidimensional scaling
- But computationally expensive

4. Nest Construction

Basic idea: Stigmergy

- Existing units of work guide & orient subsequent ones

Applications:

- Nanotechnology, “smart materials”
- Self-assembling structures in hostile environments (space, emergency situations)

Model of Self-Assembly

- High-level description of construction algorithm

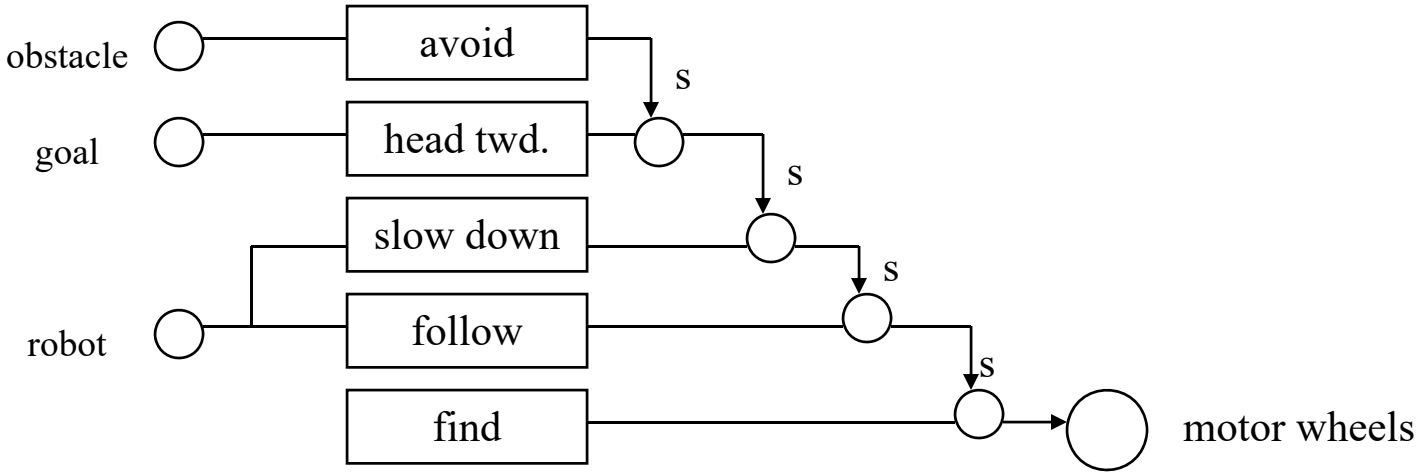
```
Construct lookup table (micro-rules: configuration -> action)
Put one initial brick at predefined site
Distribute m agents at unoccupied sites randomly
For t=1 to tmax do
        For k=1 to m do
            sense local configuration
            If (local configuration is in lookup table) then
                deposit brick specified by lookup table
                draw new brick
            Else
                do not deposit brick
            End If
            move to randomly selected, unoccupied, neighboring site
        End For
End For
```

5. Cooperative Transport

Application: cooperative box-pushing by a swarm of robots

•To locate a lit box within a given arena, and to push it toward an edge of the arena

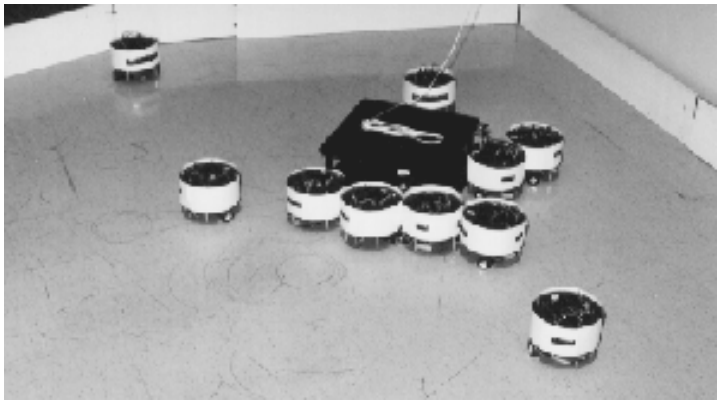
•Basic behavior model:



Subsumption architecture: Lower-level behaviors run unless suppressed by higher-level ones

Cooperative Transport

- Stagnation recovery
  - » Randomly change the direction of the applied force if the box has not been moved during a time greater than the realignment time-out threshold
  - » Reposition randomly on the box if realignment is not sufficient to move the box



6. Particle Swarm Optimization

- A population based stochastic optimization technique by Dr. Eberhart and Dr. Kenndy in 1995
- Share similarities with evolutionary computation techniques such as Genetic Algorithm
  - » Initialized with a population of random solutions (particles)
  - » Have fitness values to evaluate the population
  - » Search for optima by updating generations with random techniques
- GA vs. PSO
  - » GA is inspired by evolution of nature, while PSO is inspired by social behaviour such as bird flocking
  - » PSO doesn't have genetic operators such as mutation and crossover
- It's now a rival of GA!

Particle Swarm Optimization

- A PSO system combines local search methods with global search methods, attempting to balance exploration & exploitation
- A flock of agents swarm over the search space, guided by
  - » Their own best experience so far (*pbest*);
  - » The best experience of their "neighbours" (*gbest*)
  - » Random movements
- Neighbours : agent closest to me in data structure
- Fitness / Velocity
  - » At each step, change the velocity of each particle toward its *pbest* and *gbest* locations, with acceleration weighted by a random term

Particle Swarm Optimization

General algorithm

For each particle Initialize particle END

Do

For each particle

Calculate fitness value

If the fitness value is better than the best fitness value (pBest) in history

set current value as the new pBest

End

Choose the particle with the best fitness value of all the particles as the gBest

For each particle

Calculate particle velocity according to equation (a)

Update particle position according to equation (b)

End

While maximum iterations or minimum error criteria is not attained

Particle Swarm Optimization

Equation (a):

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[])

Equation (b):

present[] = persent[] + v[]

- v[] : the particle velocity
- persent[] : the current particle (solution).-
- pbest[] : the best solution the particle has achieved so far
- gbest[] : the best solution in the neighborhood so far
- rand () is a random number between (0,1)
- c1, c2 are learning factors; usually c1 = c2 = 2.

Particle Swarm Optimisation: Applications

Artificial Neural Network training –

- The particle is a group of weights
- The fitness value could be the misclassified cases (for classification), or error function
- Minimize the misclassified cases
- As effectively as usual error back-propagation method; perhaps more accurate and faster!
- Can take real numbers as particle
- Less parameters to adjust
  - » Number of particles (typically 20~40)
  - » Dimension of particles
  - » Range of particles
  - » Vmax, the maximum change for a particle (usually same as range)
  - » Learning factors: c1 & c2
  - » Stop condition (max no. of iterations and min error requirement)

Particle Swarm Optimisation: Applications

- Non-linear functions optimization
  - » as efficient as genetic algorithm!
- Engineering design optimization
  - » design of antennas, power system stabilizers, aircraft wings, circuits, amplifiers, truss systems, etc.
- Data mining
  - » as efficient clustering algorithms, e.g. for image clustering
  - » Rule extraction
- Game learning
  - » to co-evolve game-playing agents

Particle Swarm Optimisation: Applications

- Scheduling and planning
  - » Aircraft mission route planning, economic dispatch in power systems, generator maintenance scheduling, operational planning, optimal power flow, power transmission network expansion planning, etc.
- Controller applications
  - » Reactive power & voltage control, greenhouse air temperature control, flight paths for missiles, industrial process control, PID controllers, etc.

Conclusions: The Wisdom of the Ages

"Go to the ant ...; consider her ways, and be wise. Having no guide, overseer, or ruler, she prepares her bread in the summer, and gathers her food at harvest time."

"The locusts have no king, yet all of them go forth by companies."

- King Solomon, 1000 B.C.

"*And that swarm intelligence offers an alternative way of designing "intelligent" systems in which autonomy, emergence, and distributedness replace control, preprogramming, and centralization.*"

- Eric Bonabeau, 2003

Reference

- V. Parunak, "*Go to the Ants: Engineering Principles from Natural Agent Systems*", Annals of Operations Research, Vol. 75, pp. 69-101, 1997. (http://www.agentgroup.unimo.it/didattica/cas/L12/Parunak_Gotoant.pdf)
- E. Bonabeau, M. Dorigo & G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, New York, Oxford University Press, 1999
- J. Kennedy & R.C. Eberhart, *Swarm Intelligence*, San Francisco, Morgan Kaufmann Publishers, 2001
- A.P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, New Jersey, Wiley, 2005
- A good tutorial on PSO: http://www.swarmintelligence.org/tutorials.php