Introduction to Intelligent Systems (CO528)

Collective Bio-inspired Intelligence: Emergence, Swarms and Immune Systems

Collective Intelligence

- Some kinds of intelligence are *collective*, i.e. the intelligent behaviour emerges when lots of elements interact.
- This is apparent in many natural systems such as swarms and flocks, and at a more micro-level in biomolecular systems such as the immune system.

Emergence

- The idea that the system as a whole is more than the sum of its parts.
- Interaction can lead to truly novel phenomena that do not exist at the level of the individual.
- The classical example is “temperature” or “pressure”. Individual molecules do not have this property.
- Complex adaptive systems: Simple rules may be sufficient to lead to highly organised “smart” behaviour at the level of the system.
- New rules or “laws of nature” appear at the higher level.

Example: Tierra

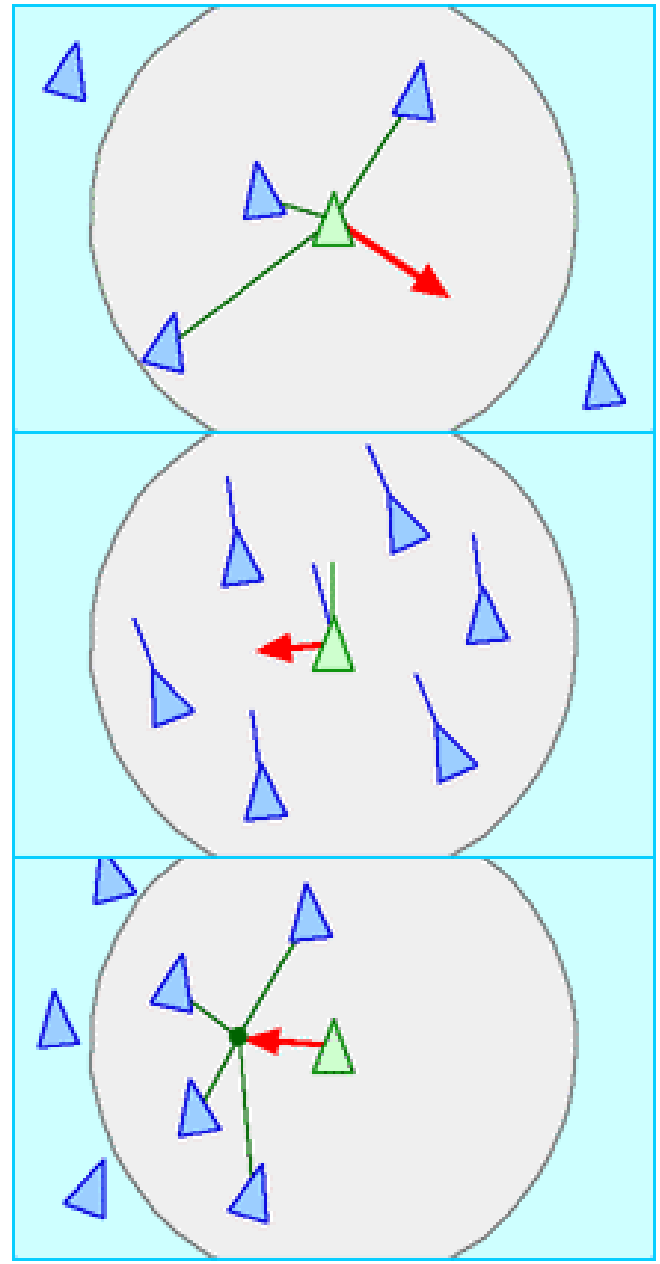
- A system consisting of a virtual CPU with its own instruction set.
- The system is seeded with an ancestor program that can replicate itself, essentially by making a copy of itself.
- A special feature of the system is that copying sometimes goes wrong.
- The result is an evolutionary process.

Tierra

- When playing with the system, it was discovered that evolution leads to emergent phenomena.
- Programs start to evolve to replicate faster, thus driving the ancestor into extinction.
- Then parasitism evolved, where programs use the ressources of other programs in order to copy themselves.
- After some time “immunity” to parasite evolved to prevent this, and even to exploit parasites themselves.
- A more modern version is called Avida: http://avida.devosoft.org/.

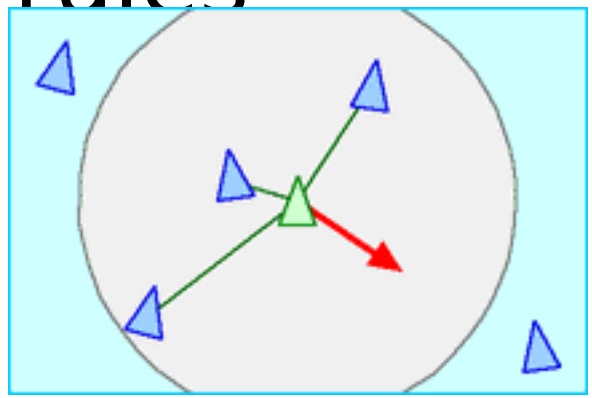
Modelling swarms

- Origins: realization that complex flocking and swarming behaviour can be generated by a small number of rules.
- Example: Craig Reynolds developed the *boids* system for computer animation of flocks of birds.
- Based on three simple rules.

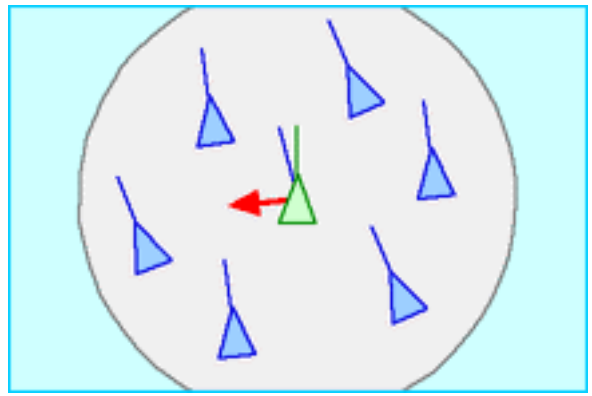


Boids: complex behaviour from simple rules

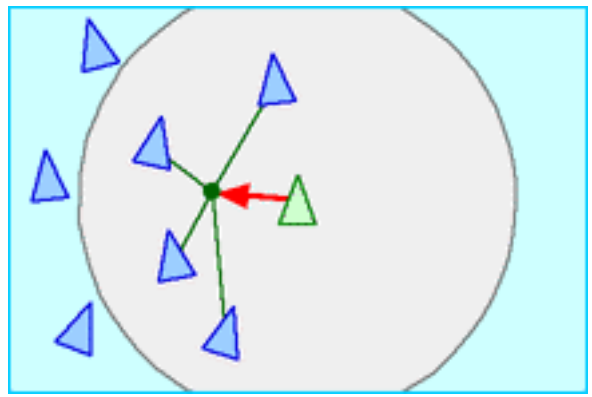
- **Separation**: steer to avoid crowding local flockmates



- **Alignment**: steer towards the average heading of local flockmates



- **Cohesion**: steer to move toward the average position of local flockmates



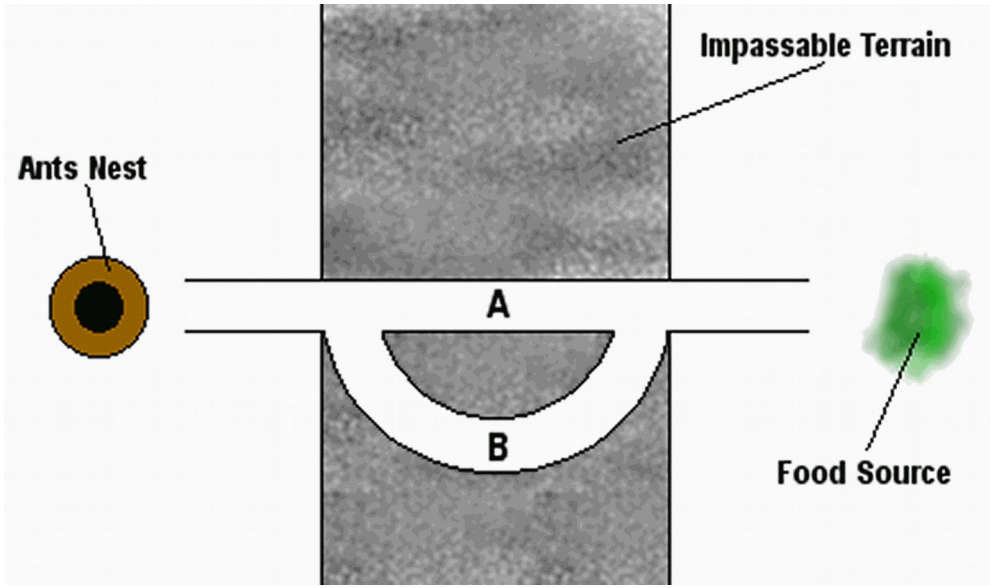
- (www.red3d.com/cwr)

Applying Swarm Intelligence

- How can we apply swarm intelligence in problem solving.
- Let’s start by looking at the collective behaviour of ants.

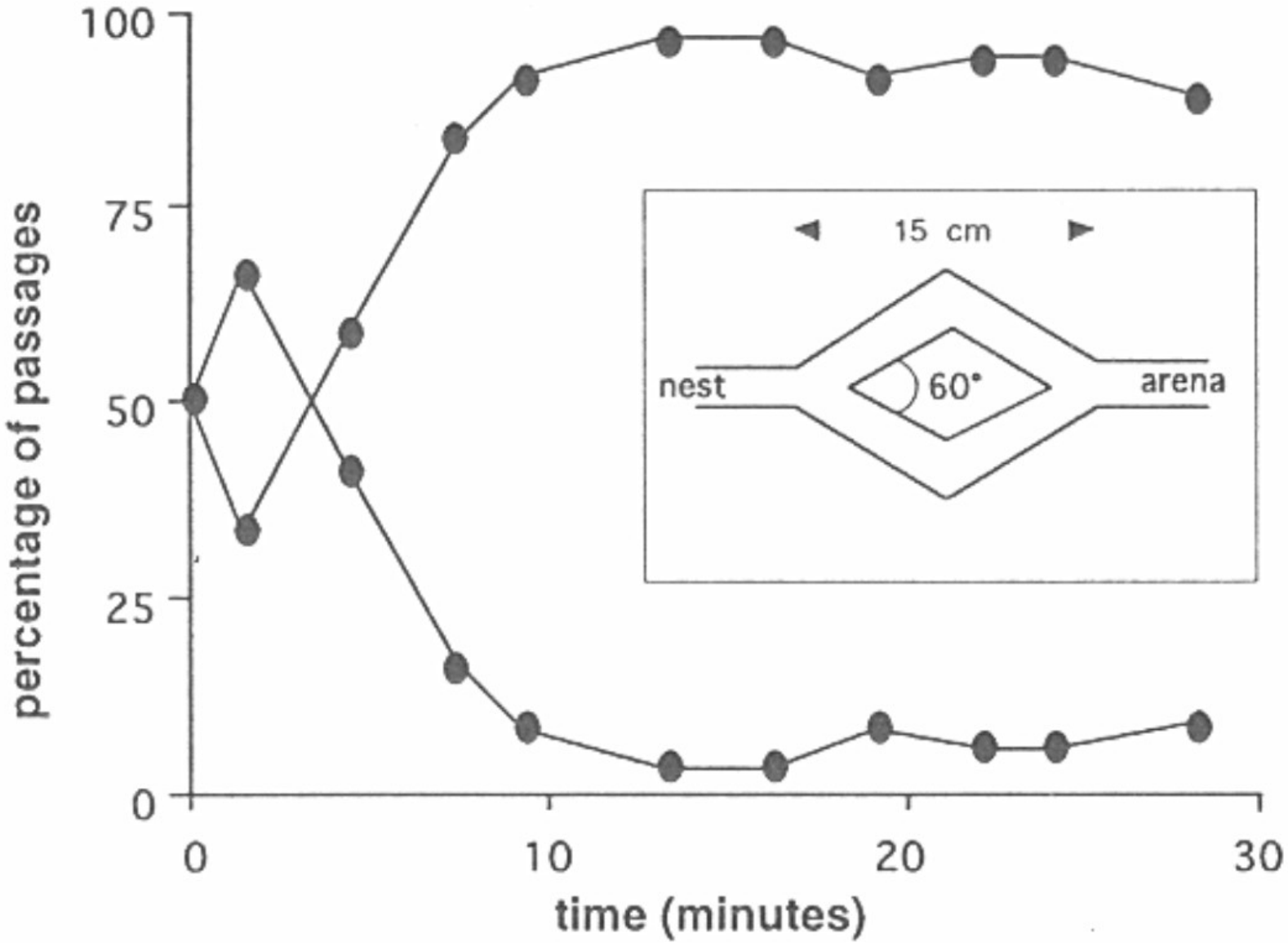
Ant colony optimization

- We use the food-seeking behaviour of ants as an *inspiration* for a computational idea.
- Ant colonies find their way to food sources in an efficient manner.
- Initially ants wander at random.
- As they wander they lay pheromone trails.
- The ants follow pheromone once it has been laid down



• Initially the ants wander over A and B with equal frequency.
• The distance across A is shorter, so more ants go back and forth that way, depositing more pheromone.

Results with Real Ants

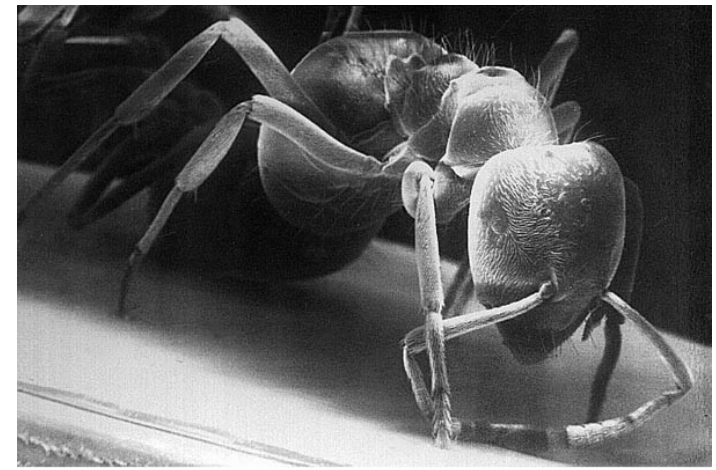


Features of ACO

- There is use of *positive feedback*. Success is rewarded by encouraging further exploration of this option.
- Exploration vs. exploitation.
- Swarm intelligence techniques rely on the *amplification of fluctuations*.
- There is no direct communication:
  - Communication is via the environment.
  - This is referred to as *stigmergy.*
- There is no “central control” for the behaviour; it is self-organized and the results are emergent.

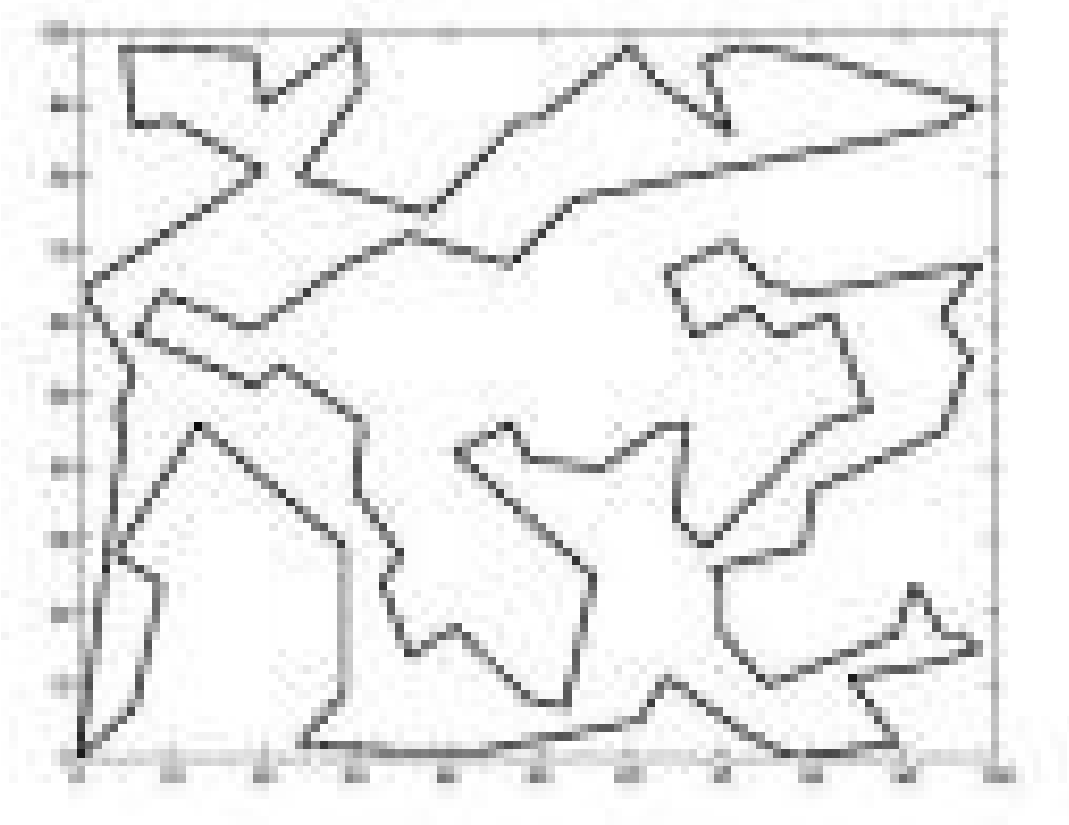
Route finding via ants.

- A number of ACO algorithms have been developed for route finding.
- In particular “shortest path” problems such as the TSP have proven to be effective applications of such methods.
- The idea is that ants wander around the graph, leaving chemical trails that evolve. These trails evolve with time, so the trails which have been traversed most often will have most chemical.
- The communication here is indirect: so called *stigmergic* communication via the environment.



Example of ACO: the TSP

- Reminder: the TSP problem
- Visit all nodes on a weighted graph in some order, without going back on yourself, so as to minimize the total distance travelled.



TSP with ACO

- Dorigo (1996) invented an algorithm for solving the TSP with ACO techniques (called the *Ant System*).
- Ants move around the graph based on three ideas:
  1. Each ant has a *memory* list of cities that have already been visited. It does not choose to go back to cities where it has already been.
  2. The ants use a *local* distance metric: closer cities are more likely to be visited than more distant ones ("whilst I'm in the area...").
  3. They also read the local pheromone trail. The stronger the pheromon trail the more likely they take this route.
- There is a *trade-off* between factors 2 and 3 that can be tuned by adjusting a parameter in the algorithm.
- Another parameter that can be tweaked is the relative strength of pheromone-following vs. random wandering.

TSP with ACO (cont.)

- After the ant has completed a tour, it lays down a pheromone trail.
- At this point this algorithm departs from the biological inspiration: the strength of pheromone laid down depends on the quality of the trail found. Clearly a (non-psychic!) ant cannot do this.
- This is important: we need to know how far to take the biological inspiration, and when to give up on it and use information that could not be got in the real bio-system.
- Also there is a pheromone *evaporation* process between each round.

TSP with ACO (cont.)

- The number of ants is important:
  - Too many ants and sub-optimal trails get reinforced too quickly, so we get an early convergence to bad solutions.
  - Too few ants and there is not enough communication between ants and trails.
  - An empirically good heuristic seems to be to take the number of ants equal to the number of cities.
- An extension: we can also introduce “elite ants” which constantly walk along the best-so-far trail.
- Again, these elite ants are an example of “leaping outside” the original biology.

Particle Swarm Optimization

- ACO is good for graph based problems
- Particle swarm optimization - PSO - is another swarm intelligence algorithm
- It is particularly useful for searching in a “continuous” space of parameters.
- For example, imagine optimizing performance of a sporting action like a tennis serve:

Particle Swarm Optimization (cont.)

- Sometimes there is a simple minimization procedure that can find the best position in the configuration space, but sometimes the problem is more complex.

Particle Swarm Optimization (cont.)

- For many problems, the best solution is a mixture of *local* information and *global* information.
- *Local* information describes how a solution compares to other solutions which are nearby in the space.
- *Global* information describes how a solution compares to all the solutions in the space.
- (Why should we use both?)
- The PSO algorithm attempts to combine the two, by moving a set of particles around the space.
- Each particle has a *position* and a *velocity*.

Particle Swarm Optimization (cont.)

- Pseudocode for a PSO:

```
Initialize a number of particles with random
   positions and velocities
LOOP (until converged):
   Move the particle according to its velocity
   Evaluate the quality of the solution
     represented by each particle
   Adjust the velocity of each particle by:
     1) a vector pointing to the best
          nearby solution
     2) a vector pointing to the best
          overall solution
END OF LOOP
```

Particle Swarms in Data Clustering

- An example of the application of PSO is concerned with analysing data using particle swarms.
- In particular the current main interest is in *unsupervised* machine learning, in particular picking out *clusters* in complex data sets. We can alternatively see this as *data compression.*
- The idea is that particles are attracted to data points. However when they get too close to each other they push each other apart. Also particles are in different *colours*: two particles of different colour will have a repulsive force between them if they get too close to each other.
- This demonstrates the beginnings of a way of “programming” particle systems.

Collective Robotics

- Instead of constructing one complex robot a task is achieved by lots of small, simple robots acting together.
- The behaviour for such robots can be evolved using genetic algorithms.
- In particular, it is known from observational studies of real ants that a swarm of ants can carry something heavier than they could if the object was split up and distributed between the ants.