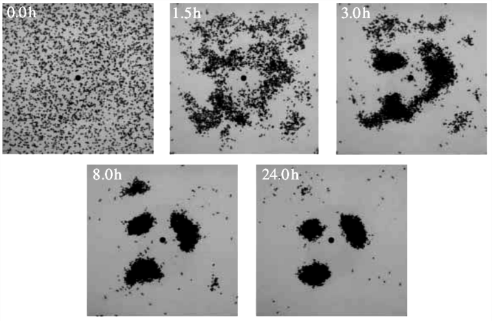
Agent based models

Jean-Luc Falcone

March 2022

Introduction to Agent Based Models

Ant Corpse Piles (*Messor sanctus*)

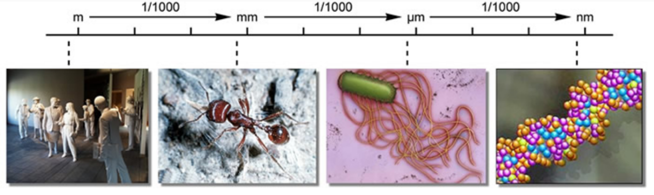


Jost *et al.*, J. R. Soc. Interface, 2007

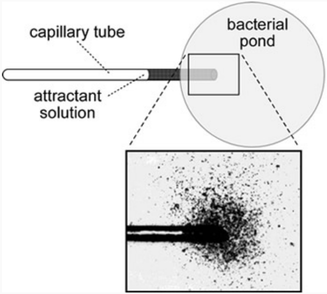
Ant corpse piles: Some questions

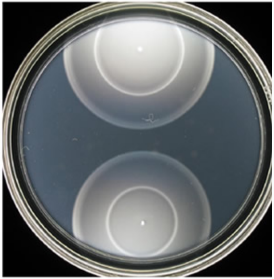
- How does it work ?
- Is it swarm intelligence ?
- What is the simplest model able to explain the process ?

Bacteria

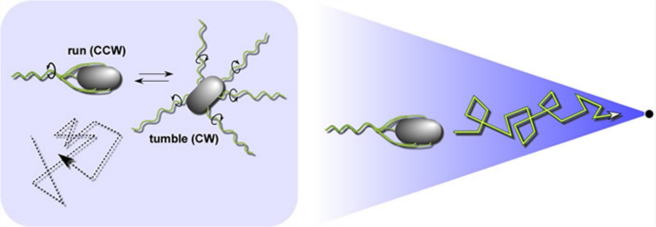


Chemotaxy





Movement



- Each trader makes individual decisions about bids and asks
- Most traders follow individual strategies
- Is it possible to explain market evolution knowing individual strategies.

Paul Jorion, 2007, *Adam Smith's Invisible Hand Revisited. An Agent-Based simulation of the New York Stock Exchange*, `http://www.pauljorion.com/blog/wp-content/uploads/2007/04/adamsmith-kyoto_rev.pdf`

Agent based models

Main idea: Modeling the basic entities as individuals and observe the global *emergent* behavior.

Many more examples:

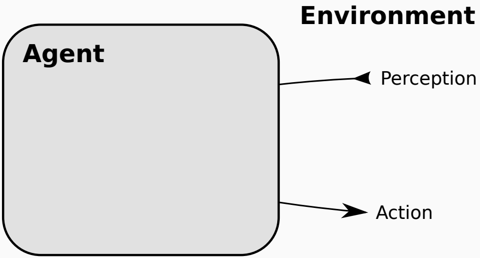
- Pedestrian simulation
- Epidemy propagation
- Ecological modeling
- ...

Agents, what are they ?

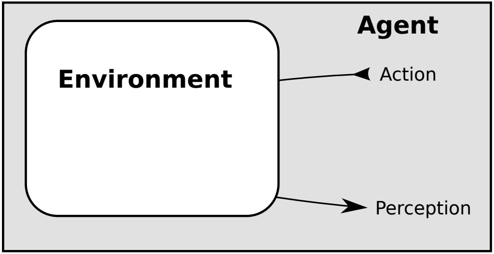
(Intelligent) Agents

- Agents are the fundamental entities of ABM
- Concept introduced in the Artifical Intelligence field
- Autonomous and decentralized
- Interact with an environment

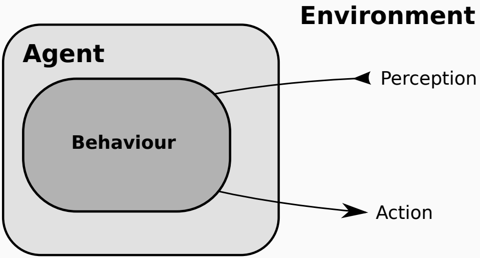
Agent



From an agent point of view



Simple Reflex Agent



Simple Reflex Agent

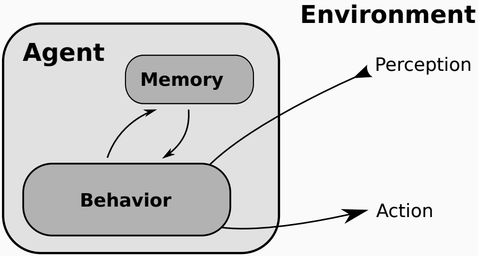
Behaviour: how the agent will react to the environement *perception*.

- Usually rule-based

$$\text{PERCEPTION} \rightarrow \text{ACTION}$$

- May be stochastic
- Perception/Knowledge of environement is limited
- The *action* may affect the environment

Intelligent Agent



Intelligent Agent

The agent has a state, which can be as simple as a boolean or as complex as it needs to be

Behaviour function:

$$\text{PERCEPTION} \times \text{STATE} \rightarrow \text{ACTION} \times \text{STATE}$$

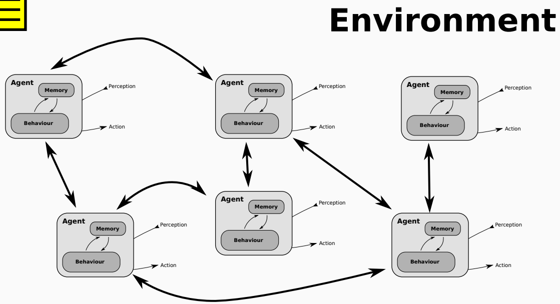
- The state is a kind of memory of past perceptions/actions
- The behaviour depends on memory
- Hence the agent is capable of learning

Example: Trading agent

```
def behavior( price, state ):
  lastTxPrice, cash, stocks = state
  if price - lastTxPrice  > RL:
    n = floor( stocks * Cs )
    return SELL( n, price )
  elif lastTxPrice - price > RL:
    n = floor( ( cash * Cb ) / price )
    return BUY( n, price )
  else:
    return NOP
```

Multi-Agents, what are they ?

Multi-Agent Systems



Multi-Agent Systems

- Usually a single system is modeled by many agents
- They could be identical, or similar, or not...
- They interact, either trough the environement or directly.

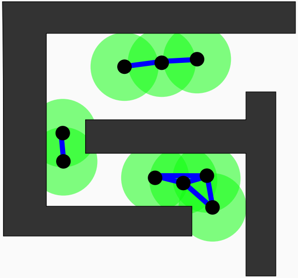
- Multi-agent systems are not synonymous of ABM. For instance:
    - Optimization
    - Network security
    - Videogames

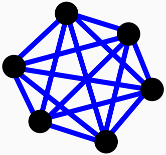
Spatialized Agents (Physical)

- Agent may have a spatial location (2D, 3D, graph)
- They may move across the domain as a result of their actions (mobile)
- The location of the agents may affect:
    - Their environment perception
    - Their interactions with other agents

Environment awareness

Interaction topology





Agent Based Models

- The use of a multi-agent system system to model a natural phenomenon.
- A complex collective behaviour emerges from the simple behaviours of agents.

ABM implementation

Object-Oriented analogy

- Class: type of agent
- Instance: Agent
- Private members: internal state
- Public methods: behaviour

```
class FooAgent {
  private final long ID = 223;
  private int count = 2;
  private double ratio = 1.5;
  public Action behaviour( Perception p ) {
     //...
  }
}
```

Asynchronous update

```
agents, env = initialize()
t = t_init
while t < t_max:
  for agent in agents:
    p = computePerceptionFor(agent, env, agents)
    action = agent.behaviour(p)
    updateEnvironment( env, action )
  increment(t)
```

```
agents, env = initialize()
t = t_init
while t < t_max:
    ps = computeAllPerceptions(env, agents)
    actions = allBehaviours( agents, ps )
    updateEnvironment( env, actions )
    increment(t)
```

Lagrangian approach

- Common approach
- Each agent is aware of its location
- Interactions and environment awareness can be globally computed.

```
agents = [
  Agent( id=1, posX=8.2, posY=0.5, ... ),
  Agent( id=2, posX=9.1, posY=2.7, ... ),
  Agent( id=3, posX=4.6, posY=1.8, ... ),
  ...
]
```

Spatial optimisation

- In most lagrangian models where agents communicate locally, it may be expensive to compute the interaction network.
- Naive approach, $O(n^2)$.

- Some specialised data structure may speed-up the process.
- For instance k-d trees:
  - Construction: $O(n \log n)$
  - Range search (in 2D): $O(n\sqrt{n})$

Eulerian approach

- Environment is a regular grid of cells
- Each cell contains a list of agents

Advantages:

- Interaction network easy to compute (neighboring cells).
- Interations are local (simple parallelism)

Disadvantages:

- Loss of spatial precision

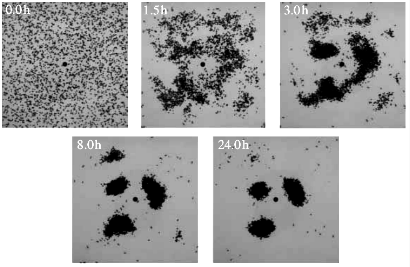
Time

- Usually continuous time
- But ABM can be used inside a Discrete Event System to update its state and produce new events.

```
def behavior( event, state ):
  ...
  return newState, [events]
```

Ant Corpse Clustering

Ant Corpse Piles (*Messor sanctus*)

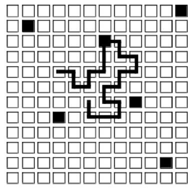


Jost *et al.*, J. R. Soc. Interface, 2007

Questions

- How does it work ?
- Is it swarm intelligence ?
- What is the simplest model able to explain the process ?

- Ants on a regular grid, with 4 directions
- Random walk, can walk over a corpse
- Sequential (asynchronous) updating scheme



Ant Behaviour (i)

- With propability P_p, the workers pick up a corpse if it is isolated or in a small cluster
- With probability P_d, the workers deposit a corpse in large cluster of dead bodies

- How the ant does evaluate the cluster size ?
  - Each ant has a memory M of size n:
  - The memory locations indicate the state of the cells visited by the ant during the last n steps: $M(i) = 1$ if there was a corps at time $t - i$, 0 otherwise

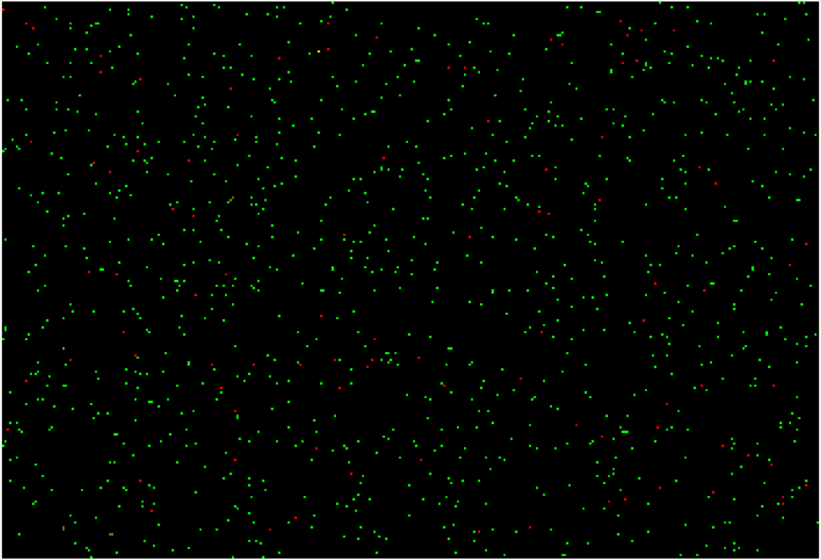
Ant Behaviour (ii)

- The probabilities are computed at each step as:

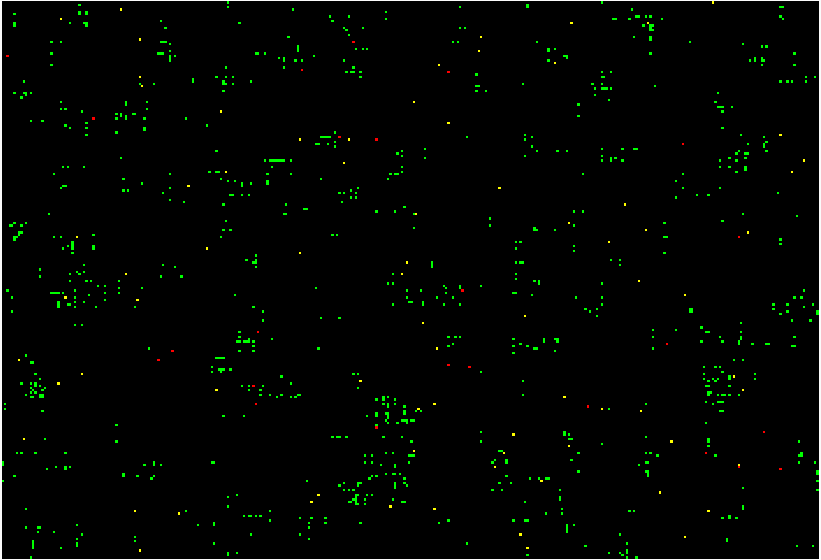
$$
\begin{aligned}
f &= \sum_{i=1}^{n} M(i) \\
P_p &= \left(\frac{k_1}{k_1 + f}\right)^2 \\
P_d &= \left(\frac{f}{f + k_2}\right)^2
\end{aligned}
$$

where k_1 and k_2 are model parameters.

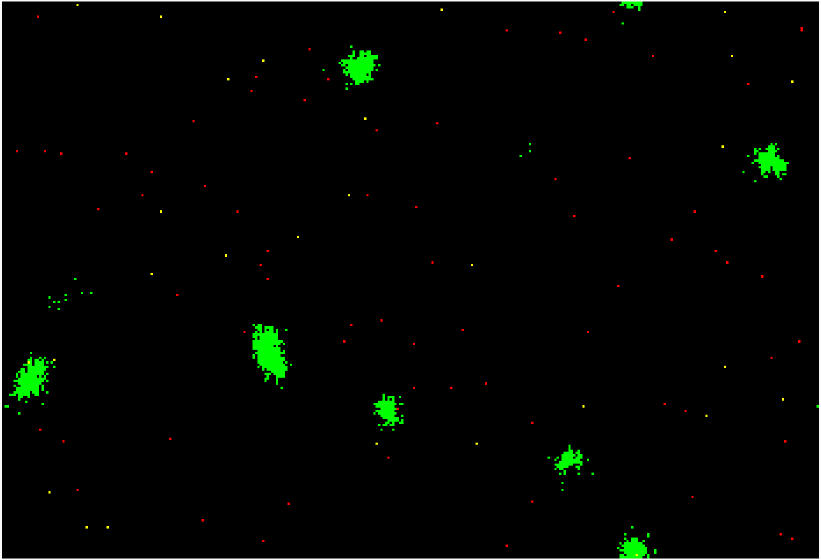
Result (i)



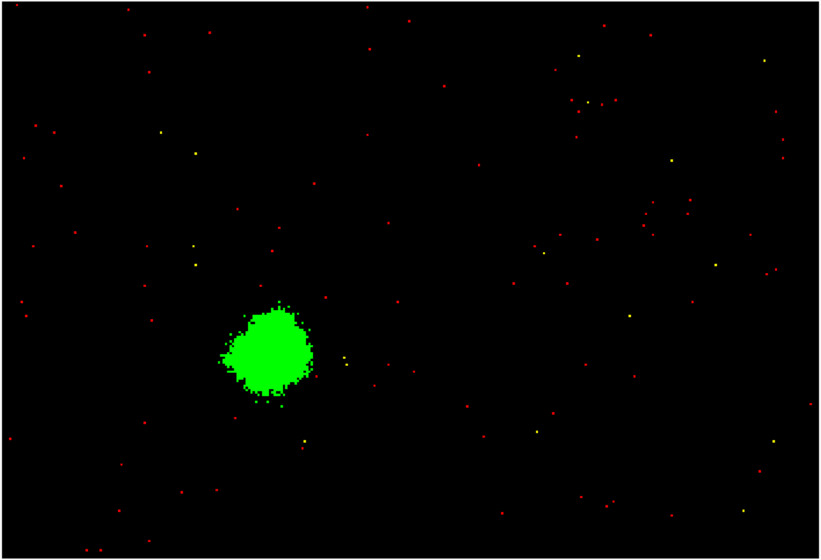
Result (ii)



Result (iii)



Result (iv)



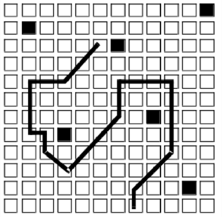
Smart ants

- Deneubourg's model works well
- Basic mechanism is intuitive
- But it requires a lot of "intelligence" from ants

- What about dumber ants ?

Unige Model (2000)

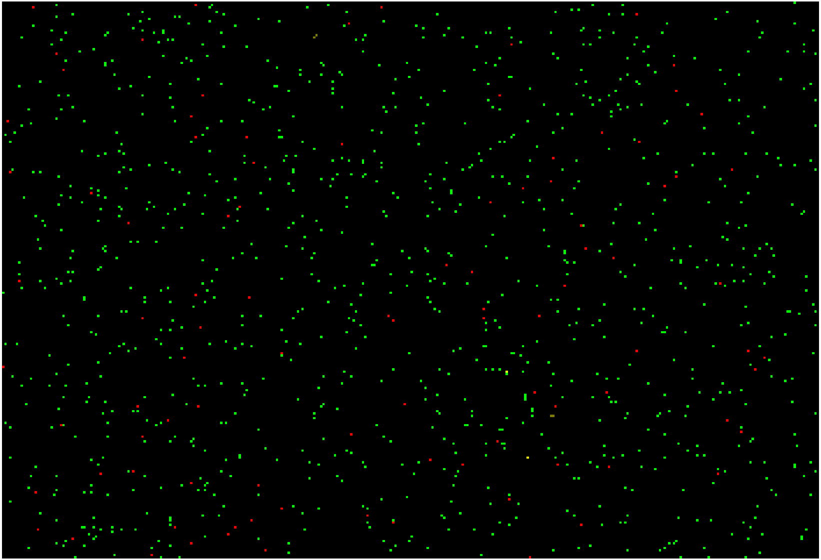
- Regular Grid, with 8 directions
- Random Walk with large diffusion constant
- Asynchronous updating



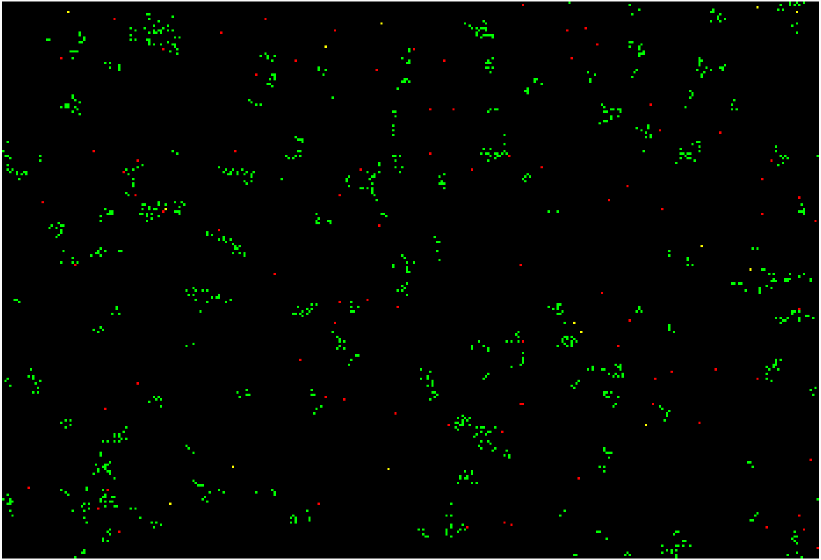
Behavior

- The ants avoids all obstacles:
  - ant corpses
  - other working ants
  - boundaries and walls
- An unloaded ant always picks a found corpse
- A loaded ant who finds another corpse always drops the carried corpse.

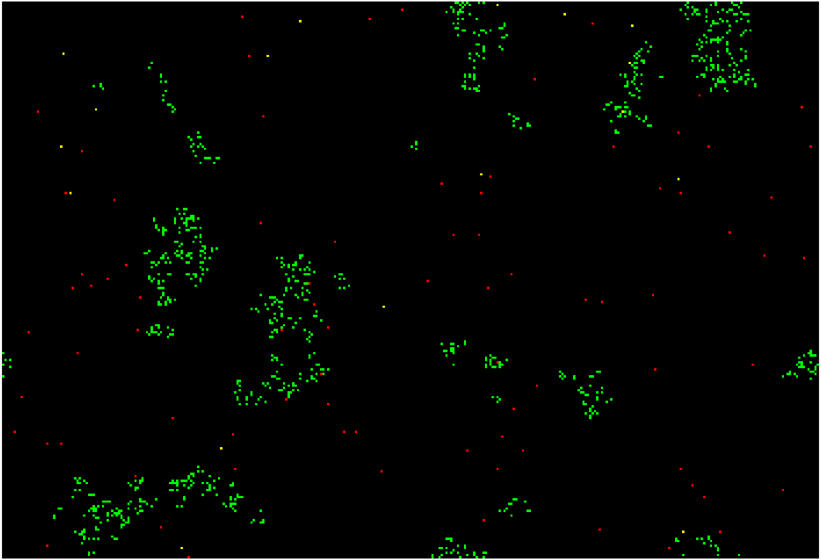
Result (i)



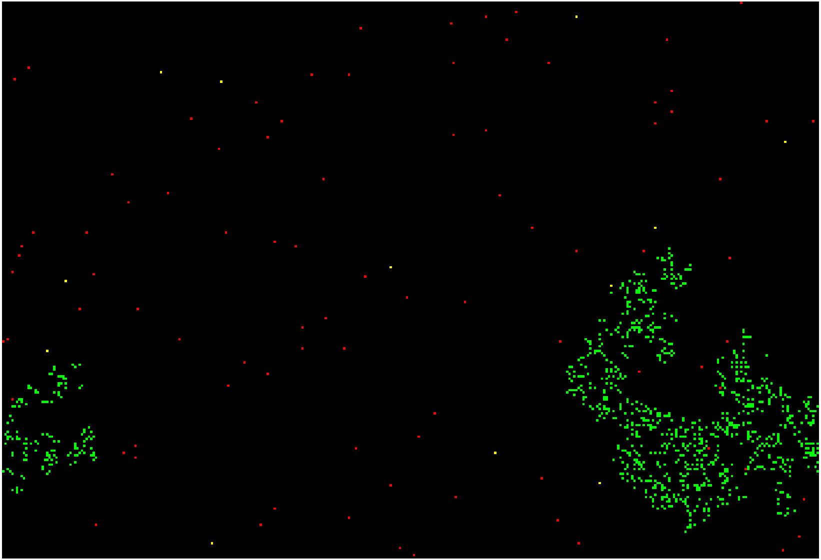
Result (ii)



Result (iii)



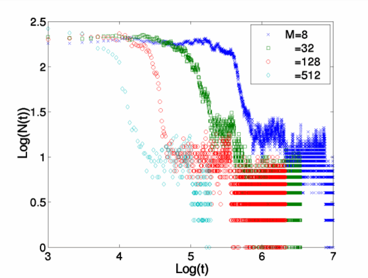
Result (iv)



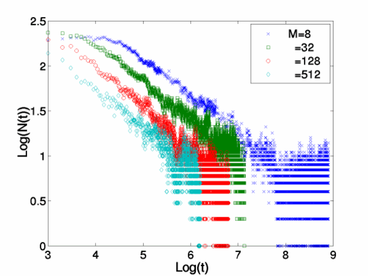
It works... but why ?

- The probabilities to remove a corpse from a cluster, or to add a new corpse are the same.
- Ants make no difference between a large or a small cluster
- When a cluster is emptied it will never reappear.
- Due to fluctuations, all clusters but one will eventually reach a zero size

Quantitative results



Deneubourg, with 8 directions

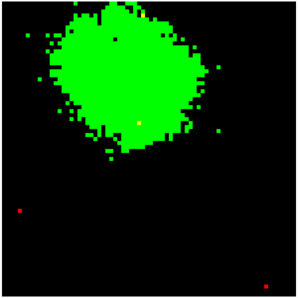


Unige

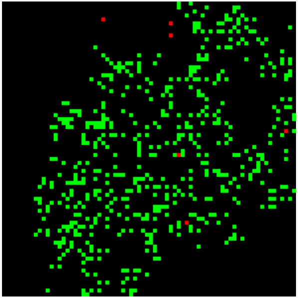
Quantitative results

- In Deneubourg's model converges ~10x faster (using better random walk).
- In both models: not a collective behavior, $N(t) = f(Mt)$
- One single ant would make it, but slower

Final Cluster

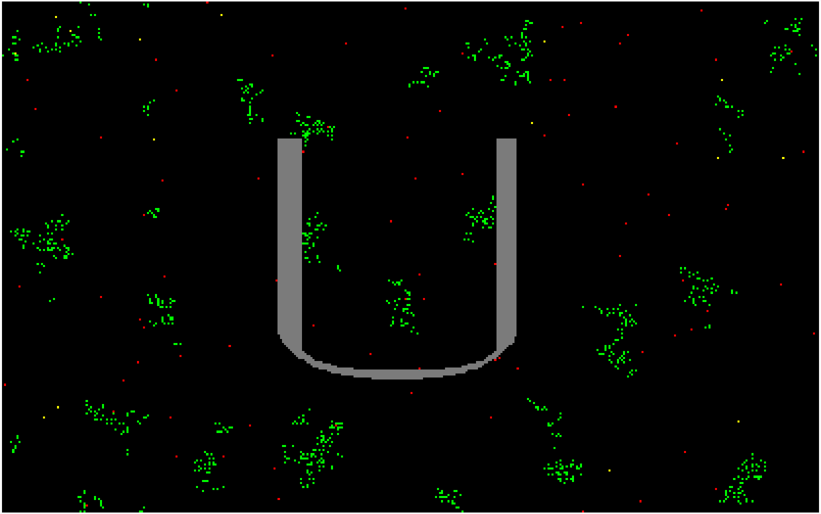


Deneubourg, with 8 directions

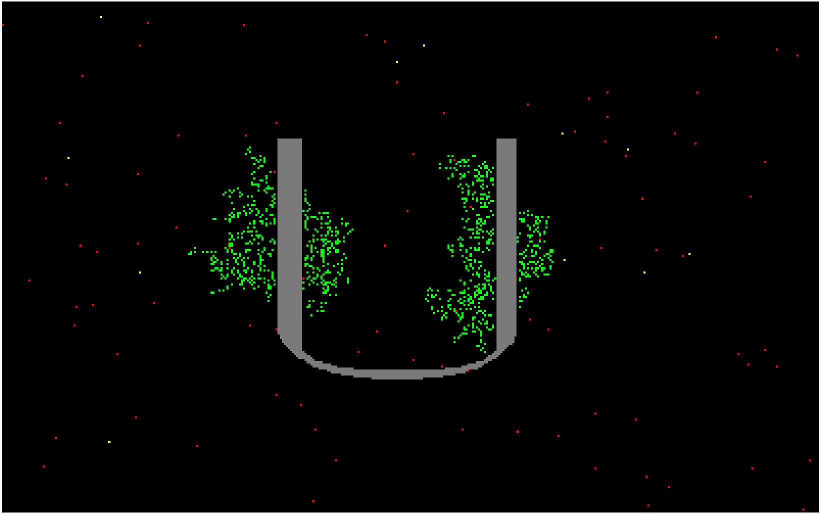


Unige

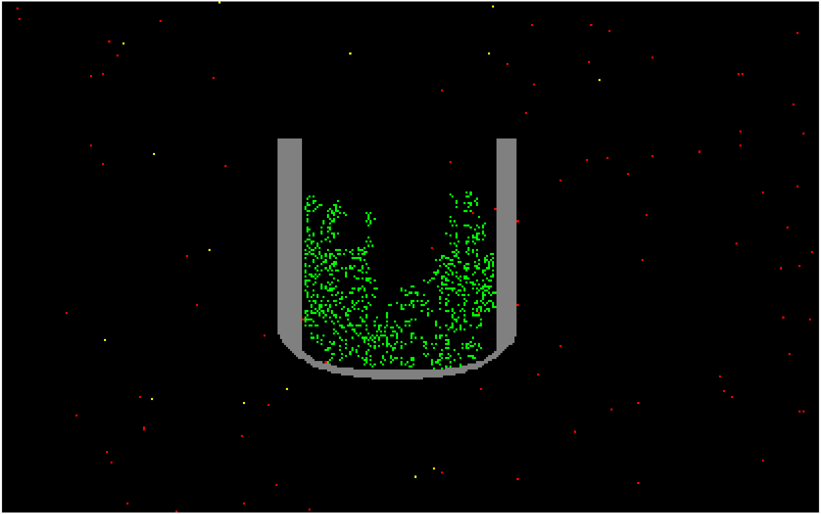
With obstacle (i)



With obstacle (ii)



With obstacle (iii)

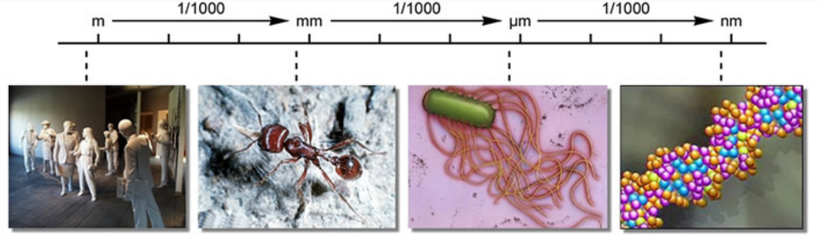


Conclusions

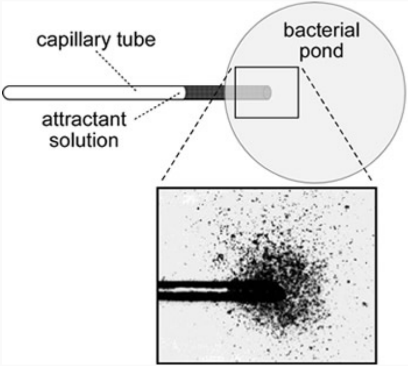
- Ant corps pile construction can be explained by statistical fluctuations
- Yet, intelligence speeds up the process
- Not a collective effect, just a collaboration with a linear speedup

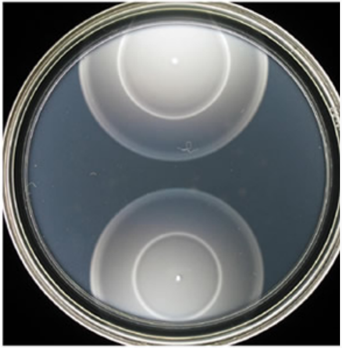
Bacteria chemotaxy

Bacteria

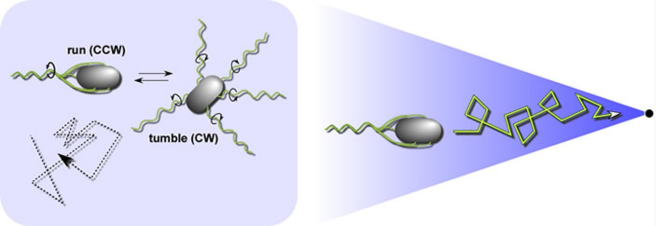


Chemotaxy

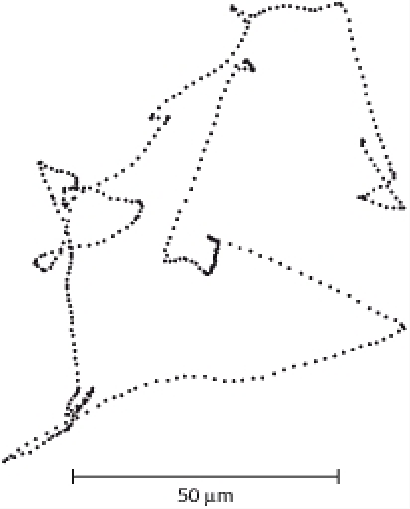




Movement (1)



Movement (2)



Model

- Eulerian 2D Grid
- In each cell (x, y) we have:
    - List of Bacteria in the (x, y)
    - Concentration of nutrient $\rho_{x,y}$
- Bacteria are agents i with state (d_i, m_i):
    - d_i: last direction taken (N, S, E, W)
    - m_i: last concentration of nutrient

Behaviour

- Bacteria remember last concentration (d_i)
- Bacteria at position (x, y) perceive the current concentration $\rho_{x,y}$

- There are two model parameters:
    - p_i : probability of tumbling when concentration increases
    - p_d : probability of tumbling when concentraion decreases
    - with $p_d > p_i$

Behaviour function

```
def behaviour( rho, m_i, d_i ):
 if rho <= m_i:
    p = p_d
  else:
    p = p_i
  if random() <= p:
    return rho, randomDirection()
  else:
    return rho, d_i
```

Environment

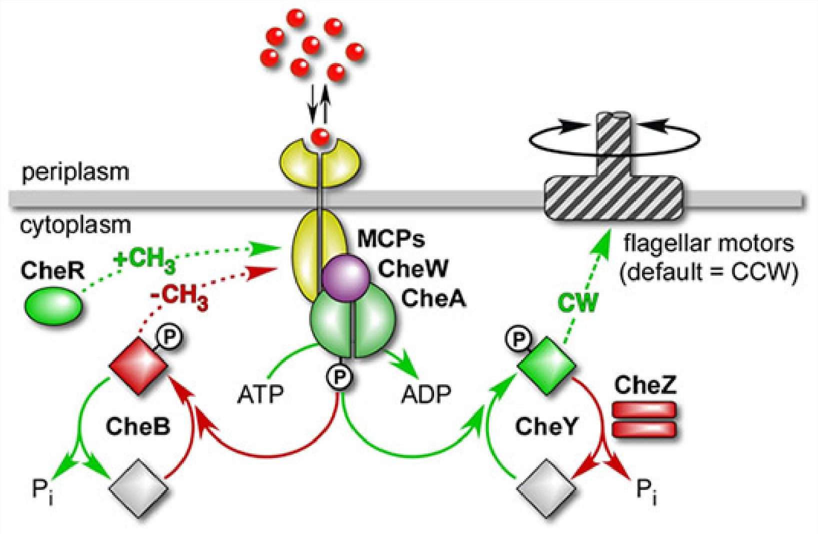
Nutrient diffusion: (solved with finite differences)

$$\frac{\partial \rho_{x,y}(t)}{\partial t} = D\nabla^2 \rho_{x,y}(t)$$

Bacteria movement:

- Each bacteria i is moved to the next cell in the direction d_i

Molecular Mechanism



See also

See also

- An overview of E. coli chemotaxis
- Robustness in bacterial chemotaxis, Alon *et al.*, Nature 397, 1999

`https://www.youtube.com/watch?v=Hc6kng5A8lQ`