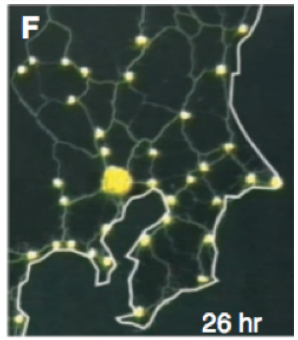
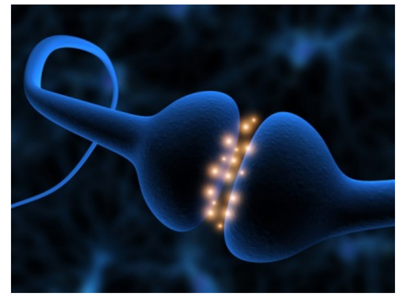
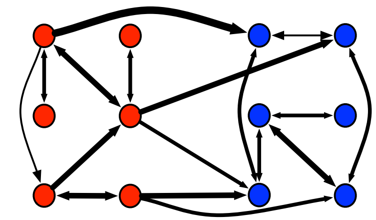
4 examples

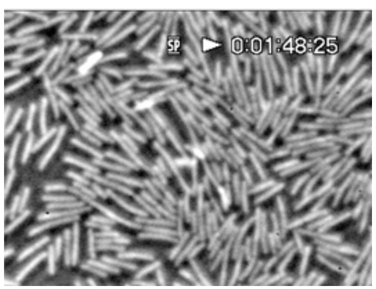


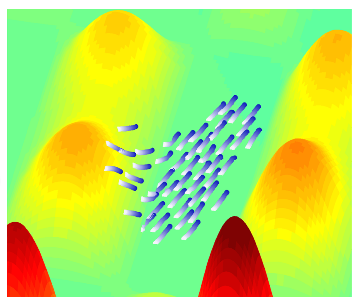
Slime mold foraging & MST construction
[*Tero et al. Science 2010*]



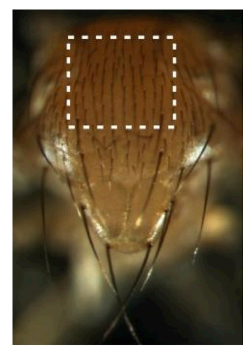


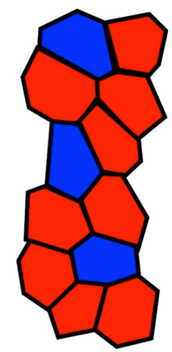
Synaptic pruning & network design



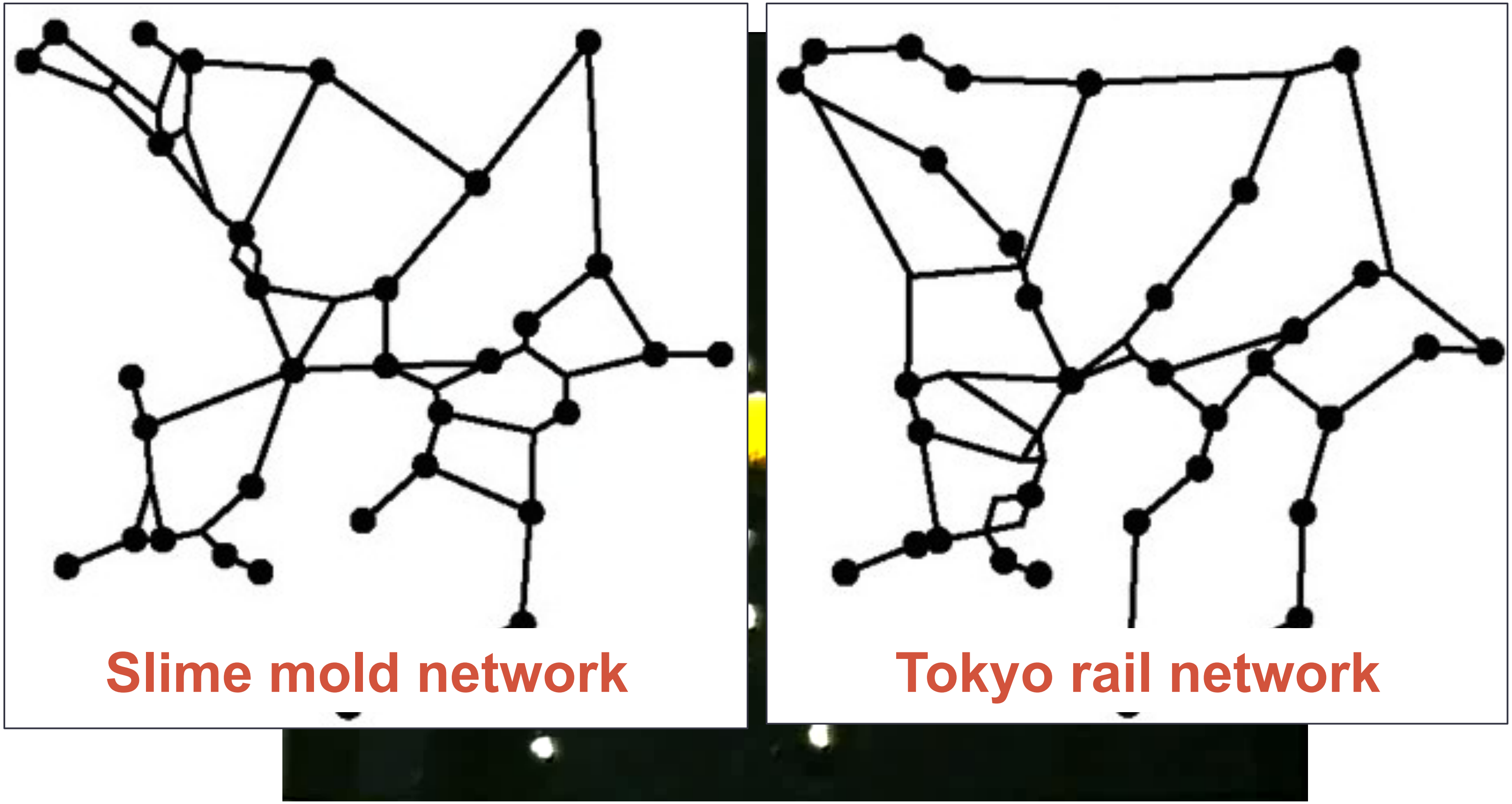


E coli foraging & consensus navigation
[*Shklarsh et al. PLoS Comput. Biol 2013*]





Fly brain development & MIS
[*Afek et al. Science 2010*]



Very similar transport efficiency and resilience

Slime mold model

- Problem:
  - Design a network to connect food
- Platform:
  - Distributed (no centralized controller)
  - Food locations unknown
  - Message passing between nodes
- Algorithm:
  - Feedback: the greater the internal protoplasmic flow, the thicker the tube
  - Idea: reinforce preferred routes; remove unused or overly redundant edges
- Evaluation:
  - network efficiency, robustness, wiring

Slime mold algorithm

Start with meshed lattice

The flux through a tube (edge) is calculated as:

Pressure difference between ends of tube

$$Q_{ij} = \frac{D_{ij}(p_i - p_j)}{L_{ij}}$$

Conductance of tube

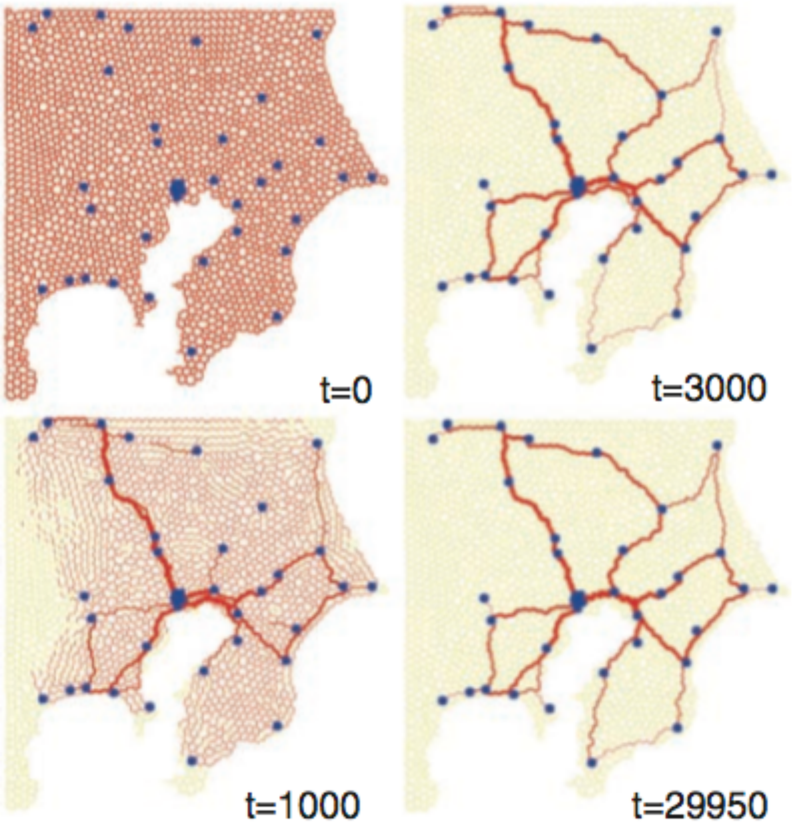
Length of tube

Think "network flow": in each time step, choose two random food sources:

$\sum_j Q_{1j} = I_0$ Source pumps flow

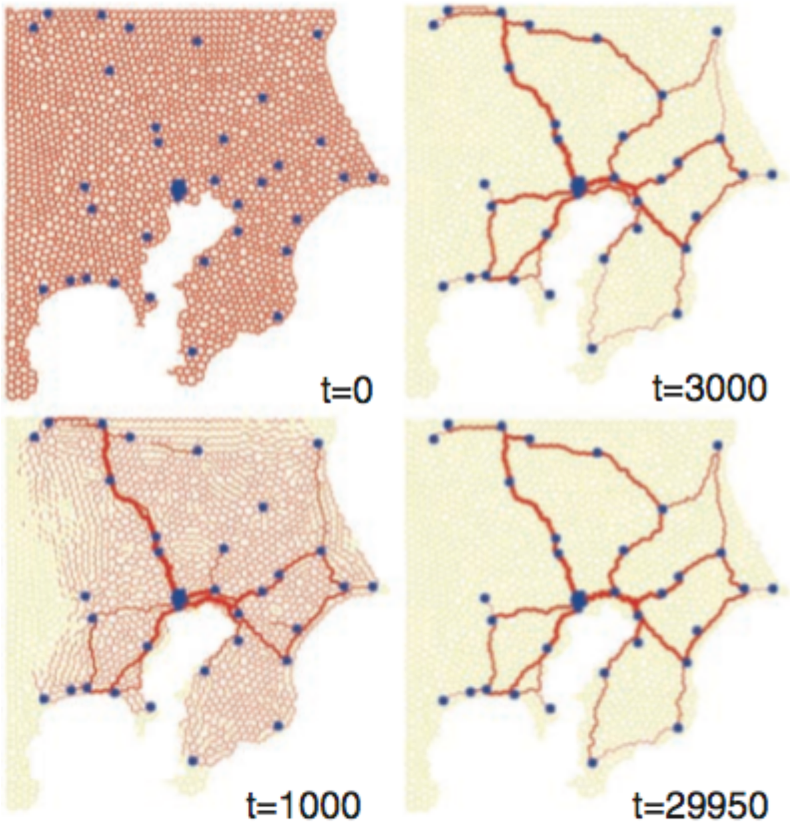
$\sum_j Q_{2j} = -I_0$ Sink consumes flow

$\sum_j Q_{ij} = 0$ Else pass flow along (conservation)



Update rule for tube weights

$$\frac{\mathrm{d}}{\mathrm{d}t} D_{ij} = f(|Q_{ij}|) - D_{ij},$$



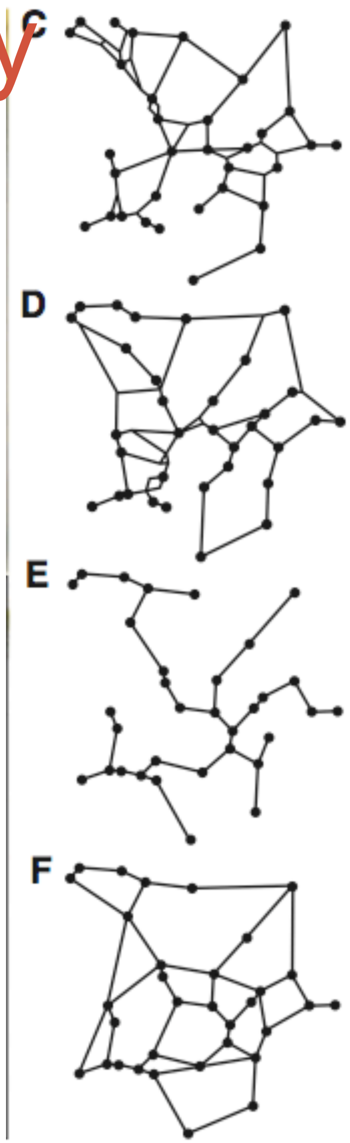
First term: expansion of tubes in response to the flux

Second term: the rate of tube constriction; the tube gradually disappears if no flow

f(|Q|) = sigmoidal curve

Evaluating network quality

- TL = wiring length used
- MD = avg minimum distance between any pair of food sources
- FT = tolerance to disconnection after single link failure



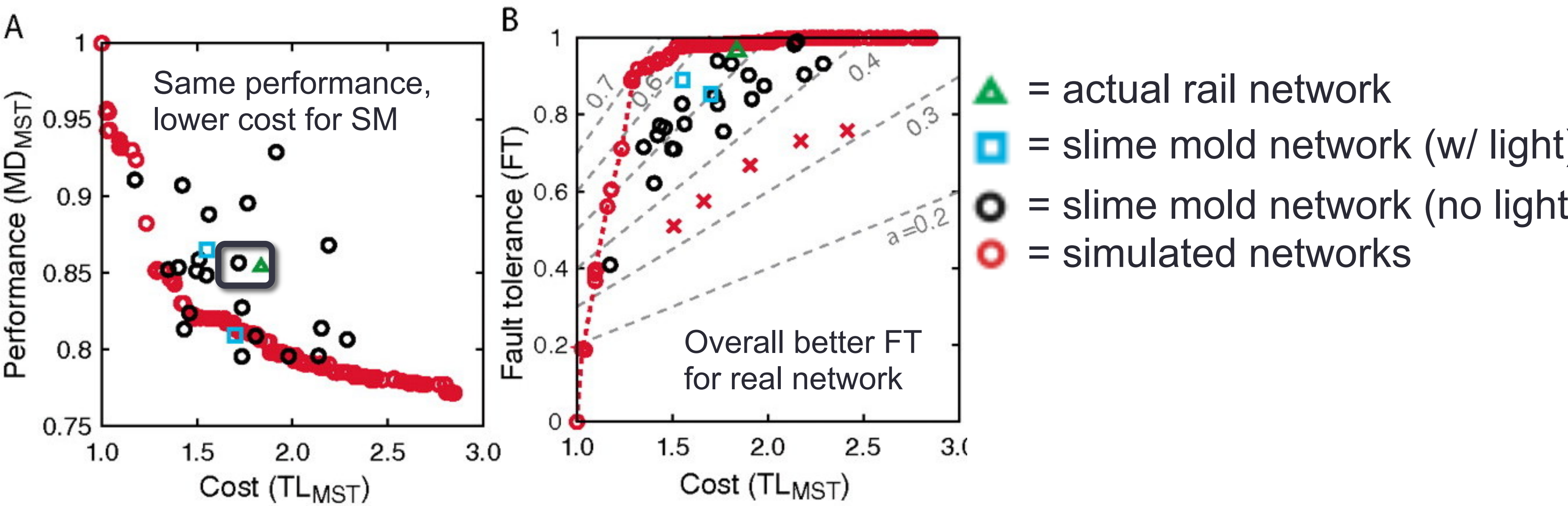
= Final slime mold network

= Tokyo railway network

= minimum spanning tree

= minimum spanning tree + added links

Slime mold and human-engineered networks have similar structural properties

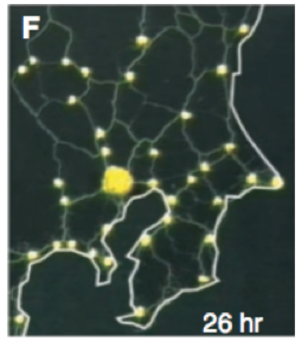


Cost: TL_{MST}(▲) = 1.80 and TL_{MST}(○) = 1.75 +/- 0.30 **SIMILAR**

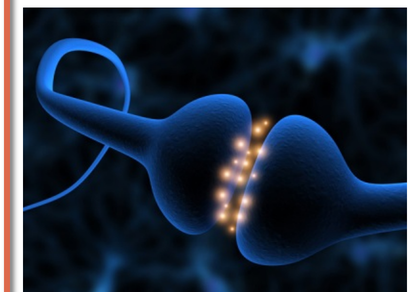
Efficiency: MD_{MST}(▲) = 0.85 and MD_{MST} (○) = 0.85 +/- 0.04 **SIMILAR**

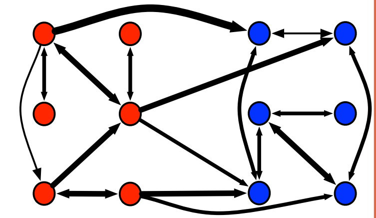
Fault tolerance: 4% of links cause rail network disconnection; 14-20% for mold

4 examples

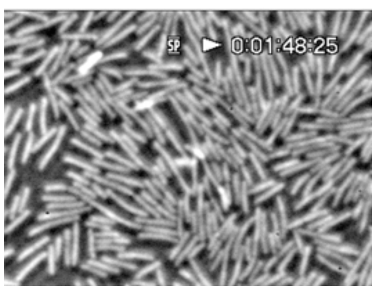


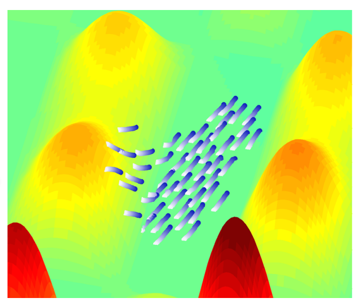
Slime mold foraging & MST construction
[*Tero et al. Science 2010*]



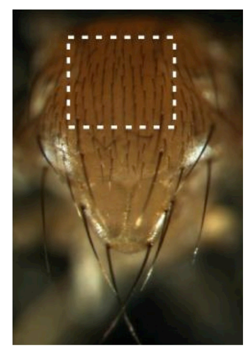


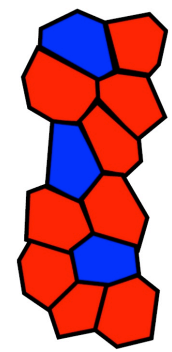
Synaptic pruning & network design





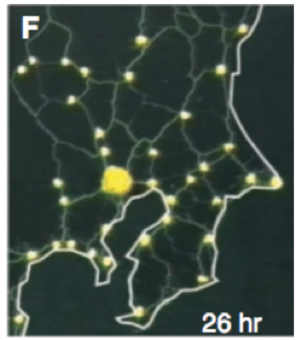
E coli foraging & consensus navigation
[*Shklarsh et al. PLoS Comput. Biol 2013*]



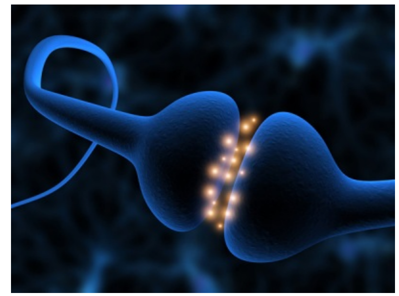


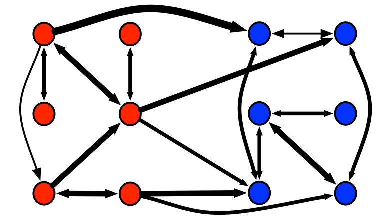
Fly brain development & MIS
[*Afek et al. Science 2010*]

4 examples

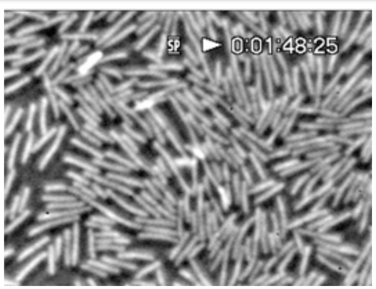


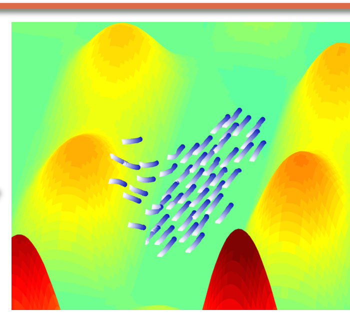
Slime mold foraging & MST construction
[*Tero et al. Science 2010*]



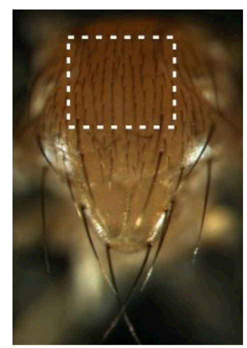


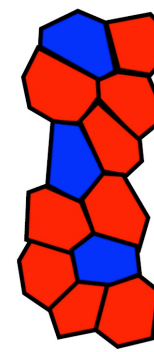
Synaptic pruning & network design





E coli foraging & consensus navigation
[*Shklarsh et al. PLoS Comput. Biol 2013*]

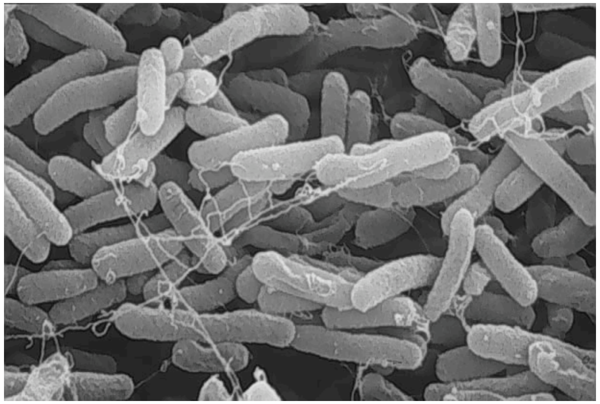




Fly brain development & MIS
[*Afek et al. Science 2010*]

Bacterial foraging

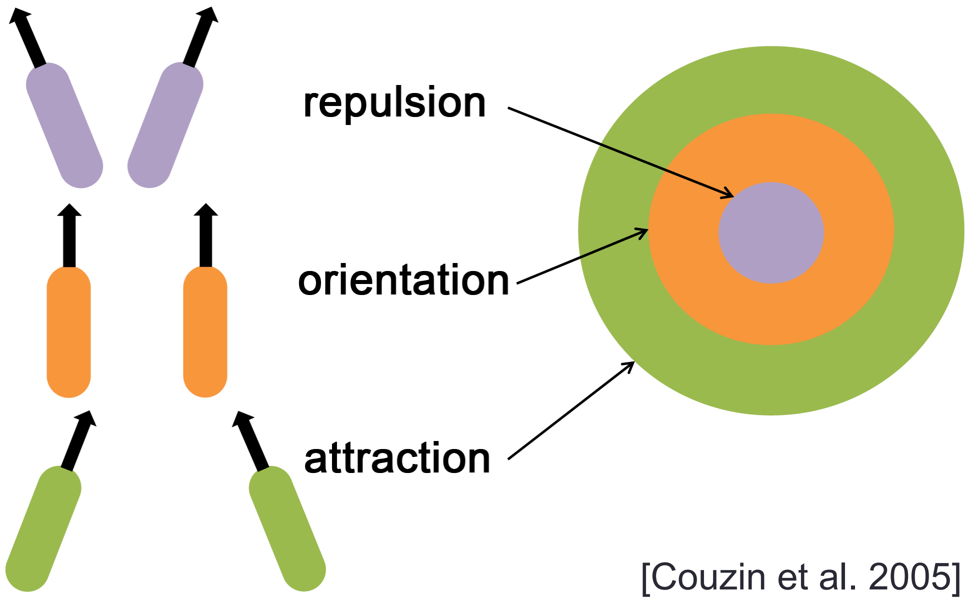
- Problem: how does a collection of bacteria collectively navigate to find food in a complicated terrain?
- Platform:
  - Distributed (no centralized controller)
  - Food location unknown
  - Broadcast-like messages: individual and neighbor knowledge
  - Bounded message complexity [see talk by Shashank Singh tomorrow]
- Algorithm: (next slides)
- Evaluation:
  - Detection accuracy and time



Bacterial chemotaxis

- Bacteria navigate via chemotaxis: move according to gradients in the chemical concentration (food)
- If low food concentration, tumble more (move randomly)

Bacteria also acquire cues from neighbors:
- Repulsion to avoid collision
- Orientation
- Attraction to avoid fragmentation



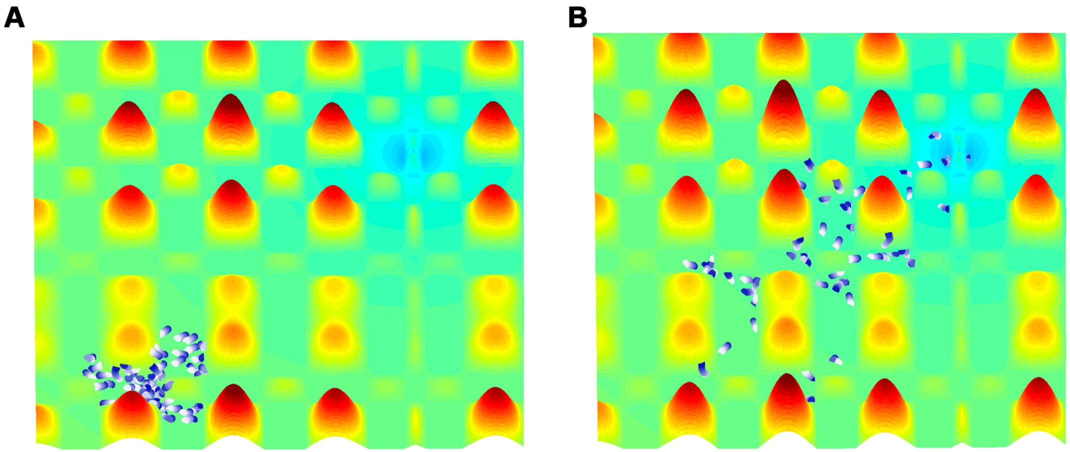
[Couzin et al. 2005]

Bacterial automata

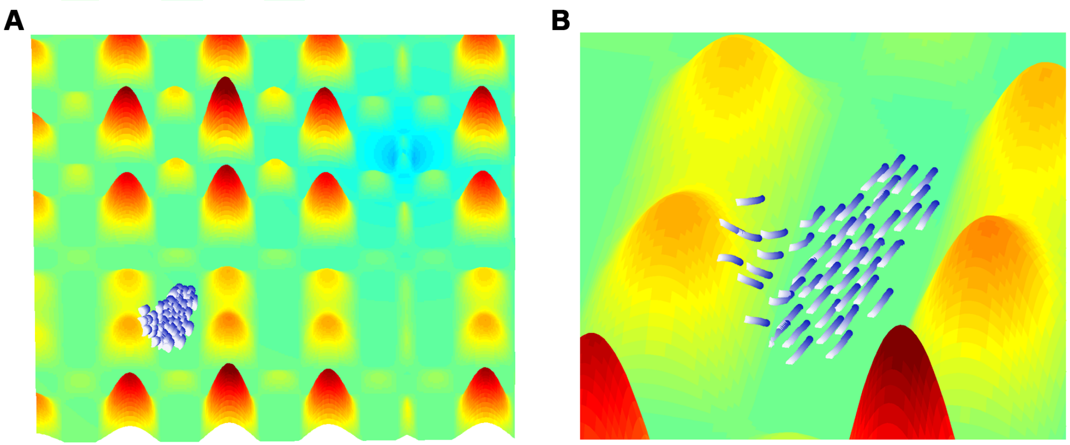
- Treat bacteria as automata with two information sources:
  - Individual belief based on food source gradient
  - Interaction with neighbors' beliefs

- Parameter $w(i)_t$ controls how much bacterium *i* "listens" to its neighbors at time *t*

Analysis of different interaction weights



<u>No interactions</u>: inefficient collective navigation



<u>Static interactions</u>: erroneous positive feedback leads bacteria astray: a subgroup gets “bad” information and leads others along an incorrect trajectory

Solution: adaptive interaction weights

- Parameter $w(i)_t$ controls how much bacterium *i* “listens” to its neighbors at time *t*
- Adjust interaction weight $w(i)_t$ based on “self-confidence”:
  - When a bacteria finds a beneficial path (strong gradient), downweight $w(i)_t$ and listen less to neighbors
  - When unsure, upweight $w(i)_t$ to increase neighbor influence

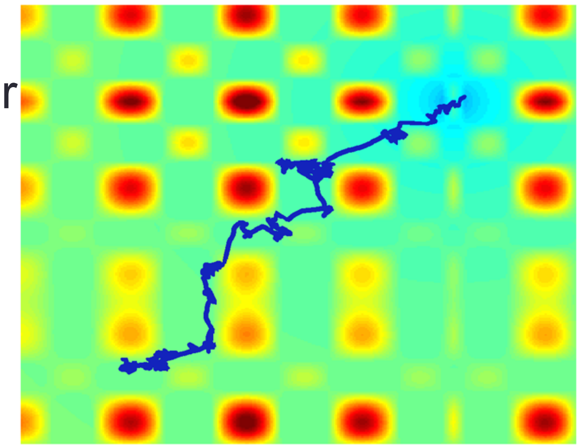
- A simple interaction rule:

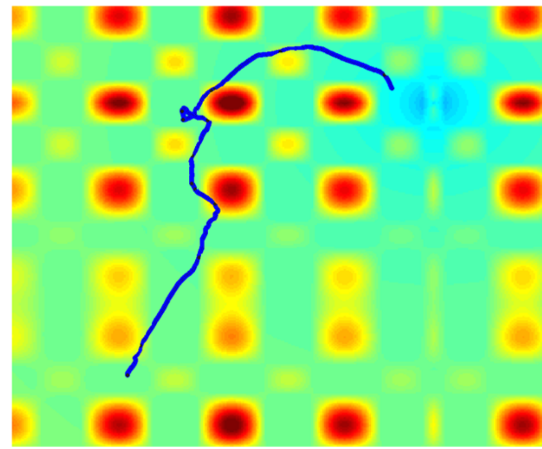
Change in gradient detected

$$w_i(t) = \begin{cases} 1 & \text{if } \Delta c_i(t) \gg 0 \\ 0 & \text{else} \end{cases}$$

Plasticity of the interaction network leads to more efficient collective navigation

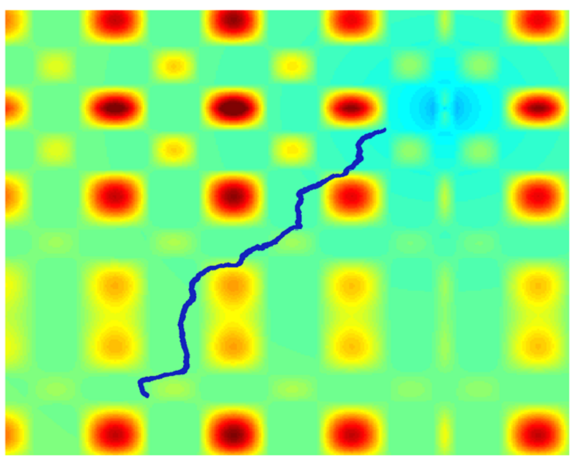
Random walk for independent agents

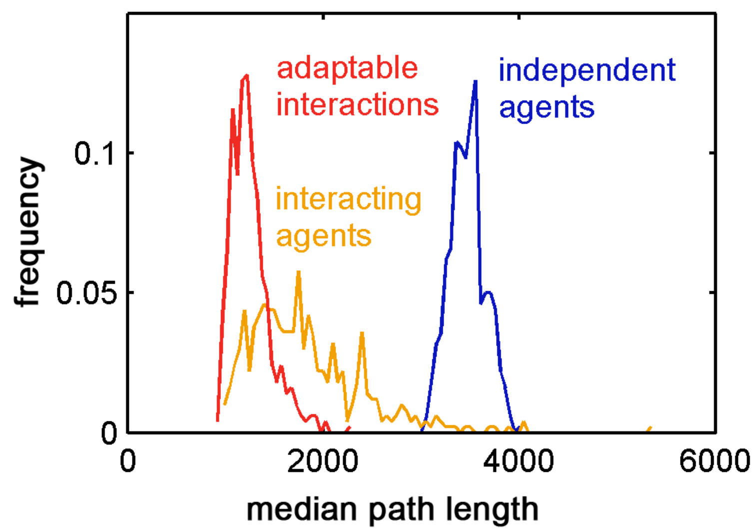




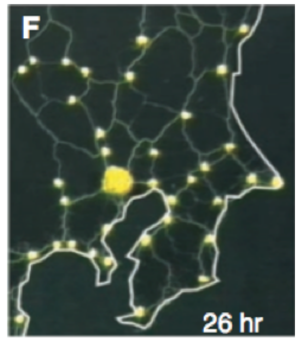
Some errors with fixed weights (static interactions)

Fewer errors with adaptable interactions

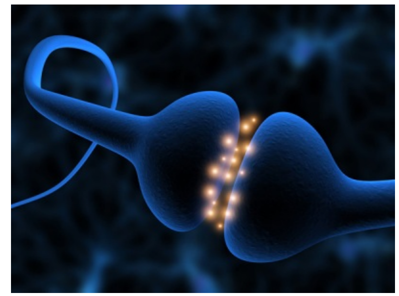


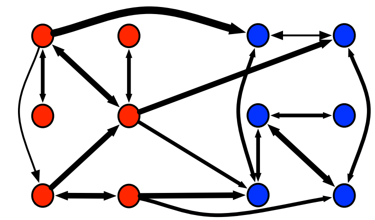


4 examples

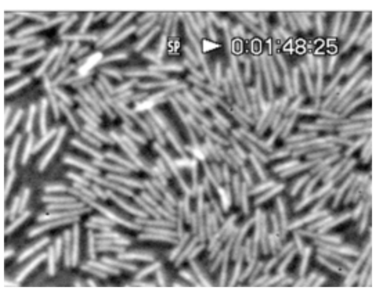


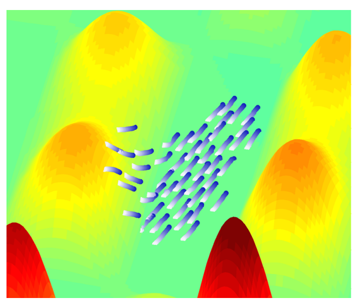
Slime mold foraging & MST construction
[*Tero et al. Science 2010*]



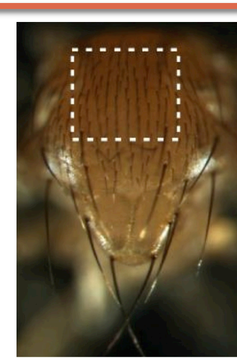


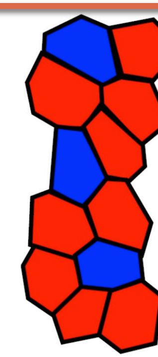
Synaptic pruning & network design





E coli foraging & consensus navigation
[*Shklarsh et al. PLoS Comput. Biol 2013*]





Fly brain development & MIS
[*Afek et al. Science 2010*]

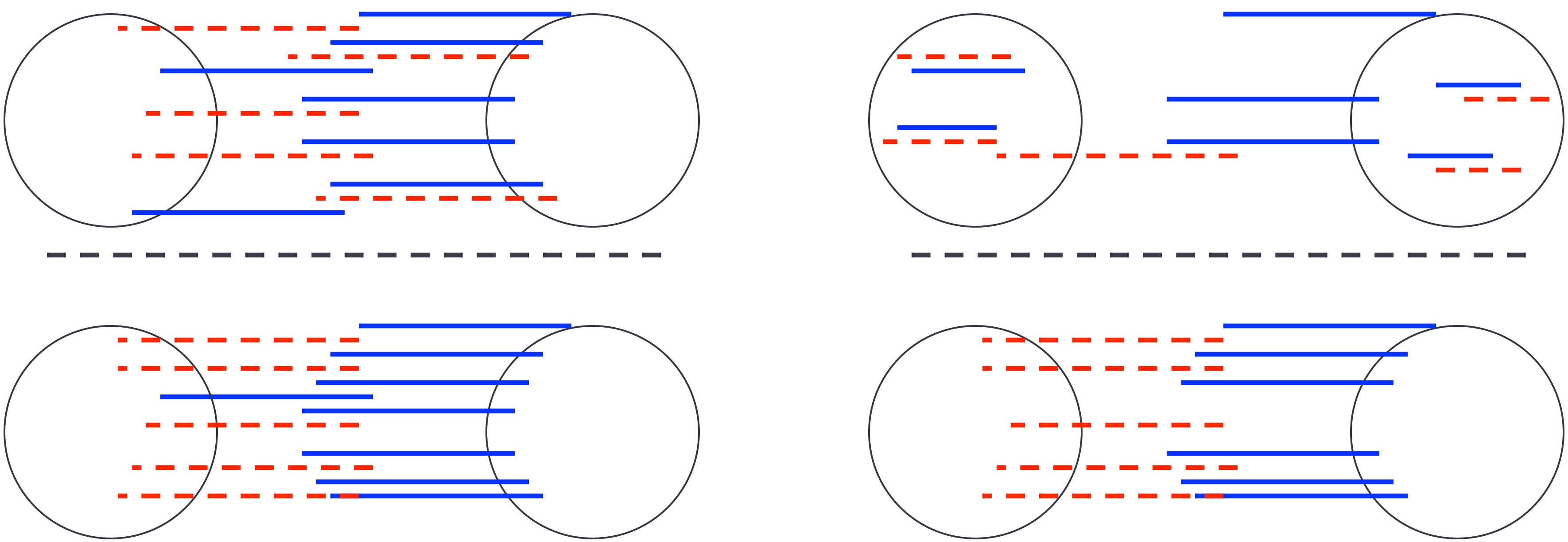
Trans vs. cis inhibition

Notch

Delta

- Recent findings suggest that Notch is also suppressed in cis by delta’s from the same cell
- Only when a cell is ‘elected’ it communicates its decision to the other cell

Trans model **Cis+Trans model**



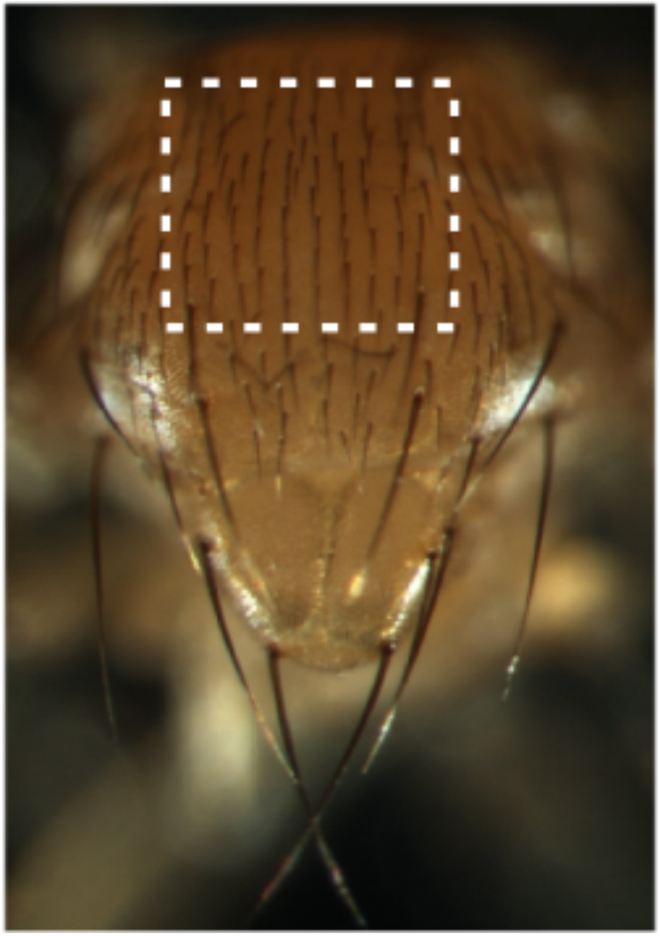
Miller et al *Current Biology* 2009, Sprinzak et al *Nature* 2010, Barad et al *Science Signaling* 2010

SOP selection in fruit flies

- During nervous system development, some cells are selected as sensory organ precursors (SOPs)
- SOPs are later attached to the fly's sensory bristles

- Like MIS, each cell is either:
  - Selected as a SOP; or
  - Laterally inhibited (via Delta-Notch signaling) by a neighboring SOP so it cannot become a SOP

No two SOPs connected



MIS vs SOP

- Stochastic
  - Proven for MIS
  - Experimentally validated for SOP
- Constrained by time
  - An uninhibited cell eventually becomes a SOP
- Reduced communication
  - A node (cell) only sends messages if it joins the MIS

- Compared to previous algs:
  - Unlike Luby, SOP cells do not know its number of neighbors (nor network topology)
  - For SOP, messages are binary

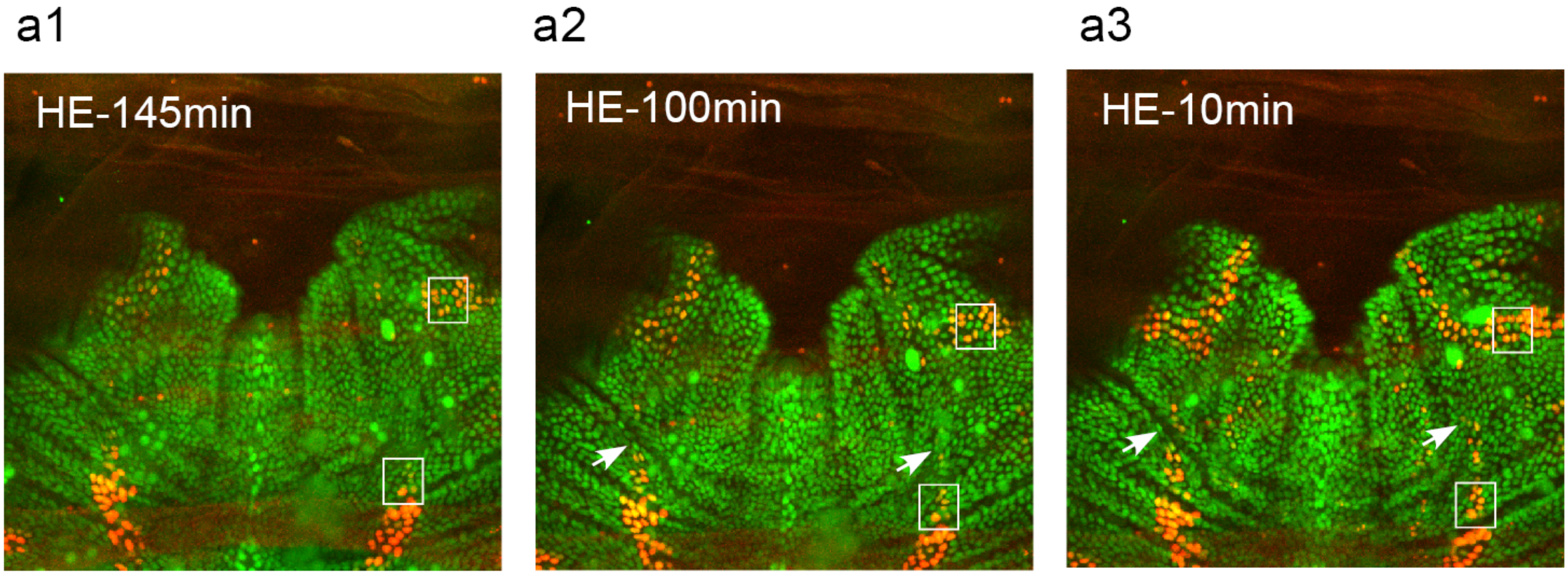
Can we improve MIS algorithms by understating how the biological process is performed?

Maximal independent set 'on the fly'

- Problem: Elect a MIS
- Platform:
  - Distributed (no node receives all inputs or observes all outputs)
  - Binary message passing between nodes, and no knowledge of topology or number of neighbors (unlike Luby)
- Algorithm ingredients:
  - Stochastic (proven for MIS; experimentally validated for SOP)
  - Constrained by time (an uninhibited cell eventually becomes a SOP)
  - Low communication: a node only sends messages if it joins the MIS
- Evaluation:
  - Message complexity
  - Running time

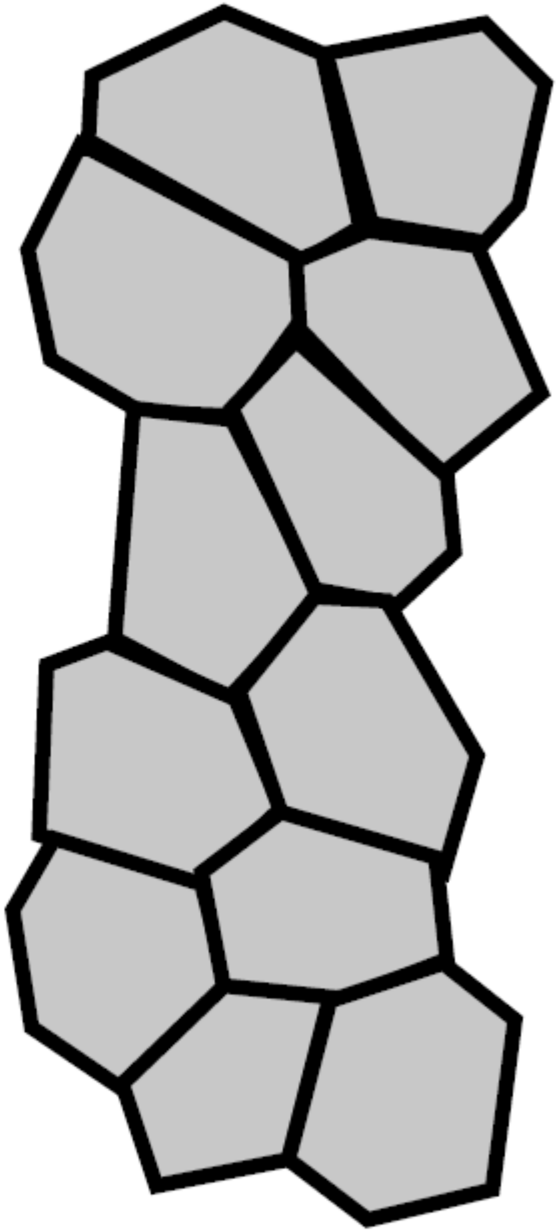
Can we improve MIS algorithms by understating how the fly solves this problem?

Movie



Simulations

- 2 by 6 grid
- Each cell touches all adjacent and diagonal neighbors



Simulations

- A cell becomes a SOP by accumulating the protein Delta until it passes some threshold (nodes increase prob. of being elected as # of active nodes decreases)

Four models:

1. Accumulation

   - Accumulating Delta based on a Gaussian distribution

2. Fixed Accumulation

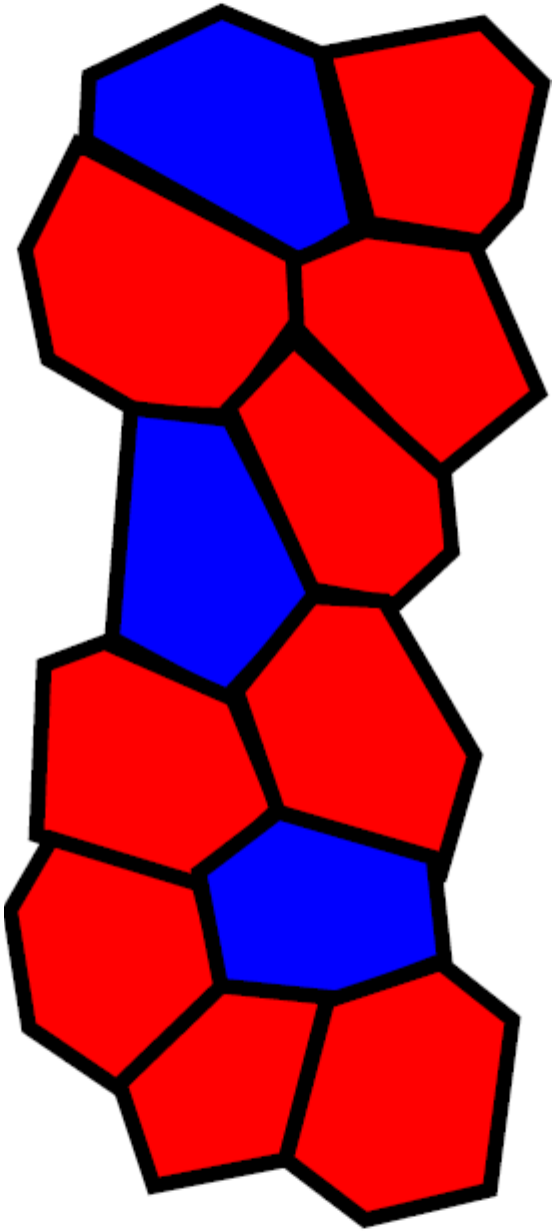
   - Randomly select an accumulation rate only once

3. Rate Change

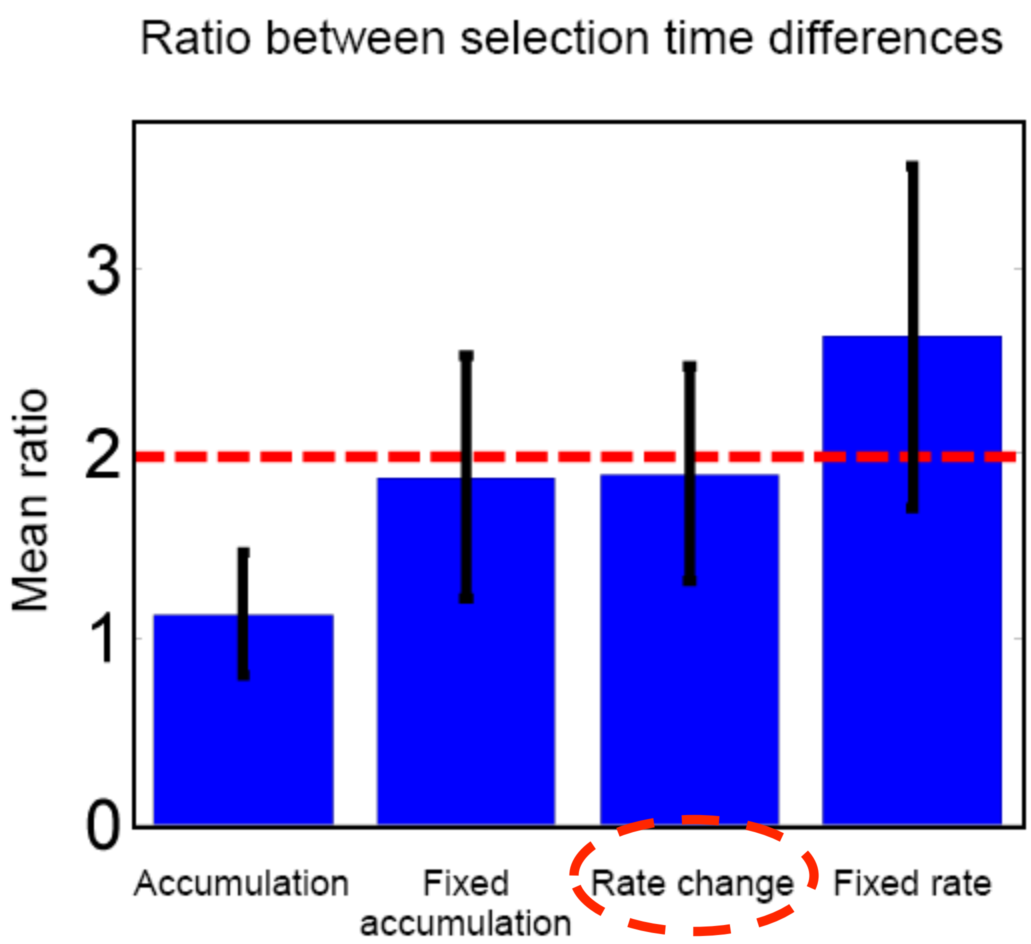
   - Increase accumulation probability as time goes by using feedback loop

4. Fixed rate

   - Fix accumulation probability, use the same probability in all rounds



Comparing time of selection experimentally and via simulations



New MIS Algorithm

MIS Algorithm (n,D) // n – upper bound on number of nodes

D - upper bound on number of neighbors

Table 1. MIS algorithm.

1. Algorithm: MIS (n, D) at node u
2. For i = 0: log D
   3. For j = 0: M log n // M is constant derived below
      4. * exchange 1*
      5. $v = 0$
      6. $\boxed{\text{With probability } \frac{1}{2^{\log D - i}} \text{ broadcast } \boldsymbol{B}}$ to neighbors and set $v = 1$ // ***B*** is one bit
      7. If received message from neighbor, then $v = 0$
      8. * exchange 2 *
      9. If $v = 1$ then
         10. Broadcast B; join MIS; exit the algorithm
      11. Else
         12. If received message B in this exchange, then mark node u inactive; exit the algorithm
      13. End
   14. End
15. End

- W.h.p., the algorithm computes a MIS in O(log^{2}n) rounds
- All msgs are 1 bit

Afek et al *Science* 2011, *DISC* 2011

Biological distributed algorithms

- Problem: what computational problem is the system trying to solve?
  - MIS, network construction, distributed search & consensus, task allocation
- Platform: what are the constraints and assumptions that need to be abided by?
  - Distributed, simple messages, dynamic networks, unknown environments, no UIDs
- Algorithm: what strategy solves the problem within the platform?
  - Exploring broadly to deal with uncertainty, and then exploiting [see also Chris Reid tomorrow]
  - Feedback processes, to reinforce good solutions/edges/paths [slime mold, pruning]
  - Rates of communication/contact [MIS, pruning, ants]
  - The importance of stochasticity, to overcome noise & break symmetry
- Evaluation: what needs to be optimized?
  - Run-time efficiency, communication cost, flexibility, robustness, adaptation, resources
  - And their trade-offs! [MIS: higher run-time, lower complexity; pruning: wasteful but adaptive]

Conclusions

- What can biology contribute to distributed algorithms research?
  - New robust/flexible/adaptive algorithms
  - Revisiting problems with more or different constraints

- What can distributed algorithms contribute to biology research?
  - Formal models to evaluate performance and predict behavior
  - Identification of parameters critical for algorithmic optimization but ignored; raise new, testable hypotheses

Thanks!