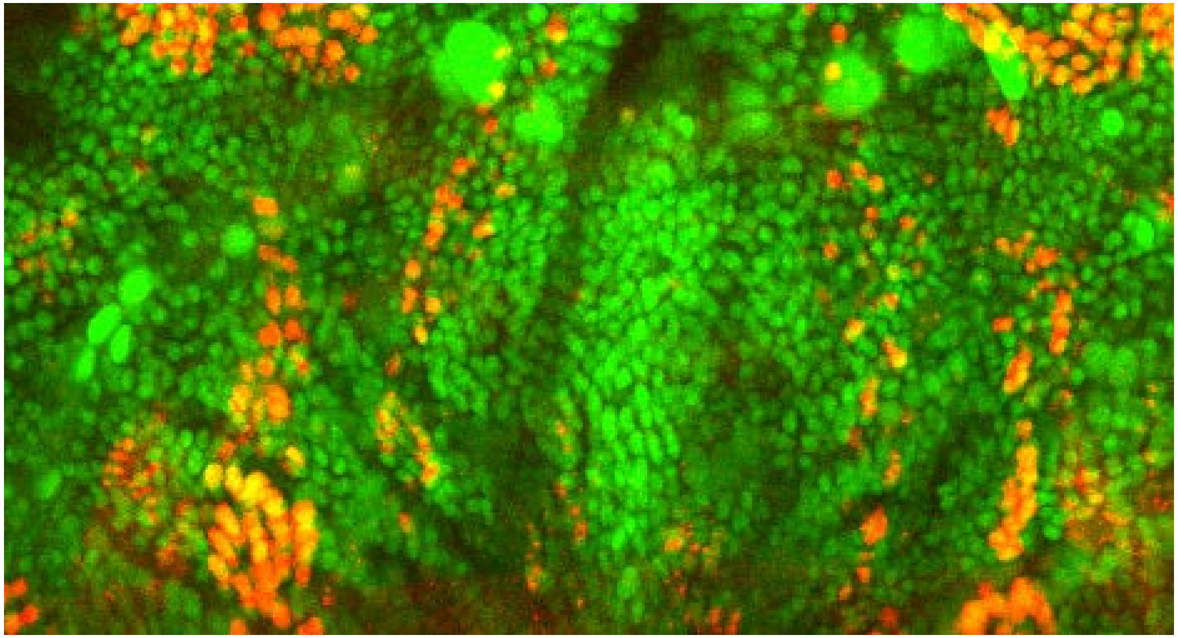
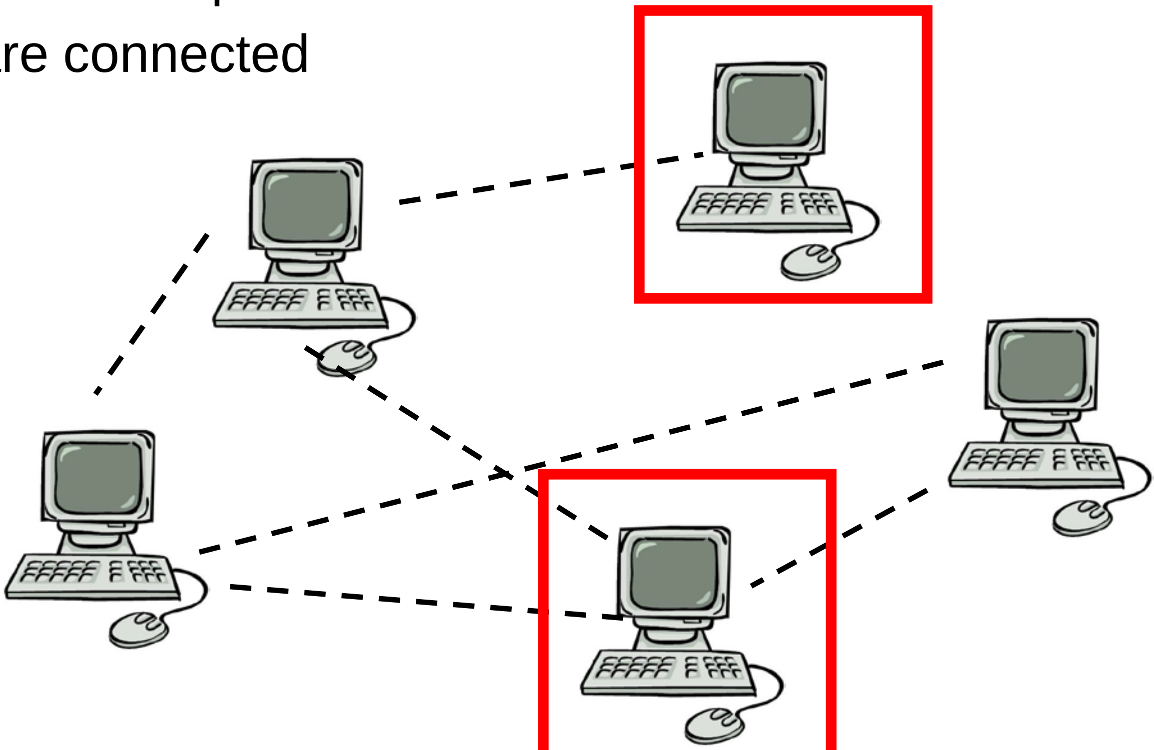
Algorithms in Nature

SOP & MIS



Maximal Independent Set (MIS)

- A fundamental problem in distributed computing:
  - Often required to establish wireless networks
  - Used for routing messages, grouping sensors etc.
- For a set of nodes select a subset *A* such that:
  - Each processor is connected to a processor in *A*
  - No two processors in *A* are connected



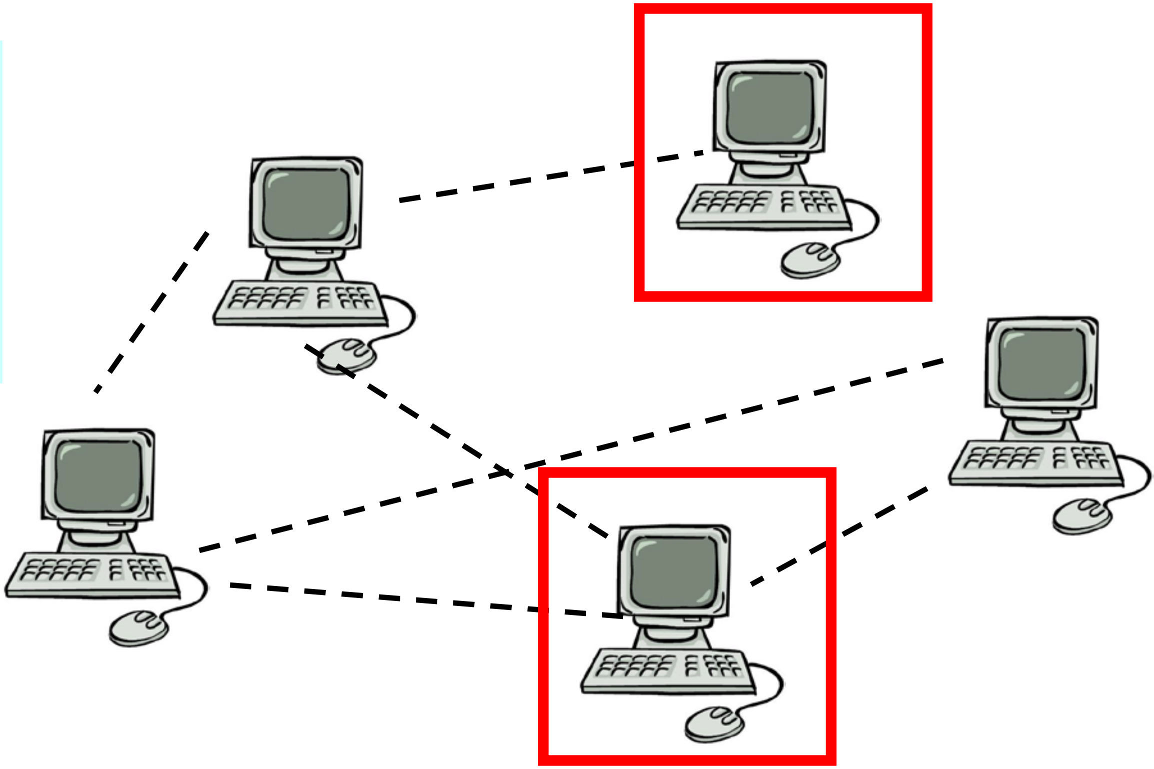
Maximal Independent Set (MIS)

- Fast algorithms exist for distributively selecting the MIS set but:

  - They Assume nodes know the status of their neighbors and also the topology of the graph (which is changing)

  - Use large messages

"it is difficult to see how this problem can be solved in substantially fewer stages such as O($\sqrt{n}$) …" (Valiant 1982)



Algorithm for MIS

Each process needs to know how many neighbors it has

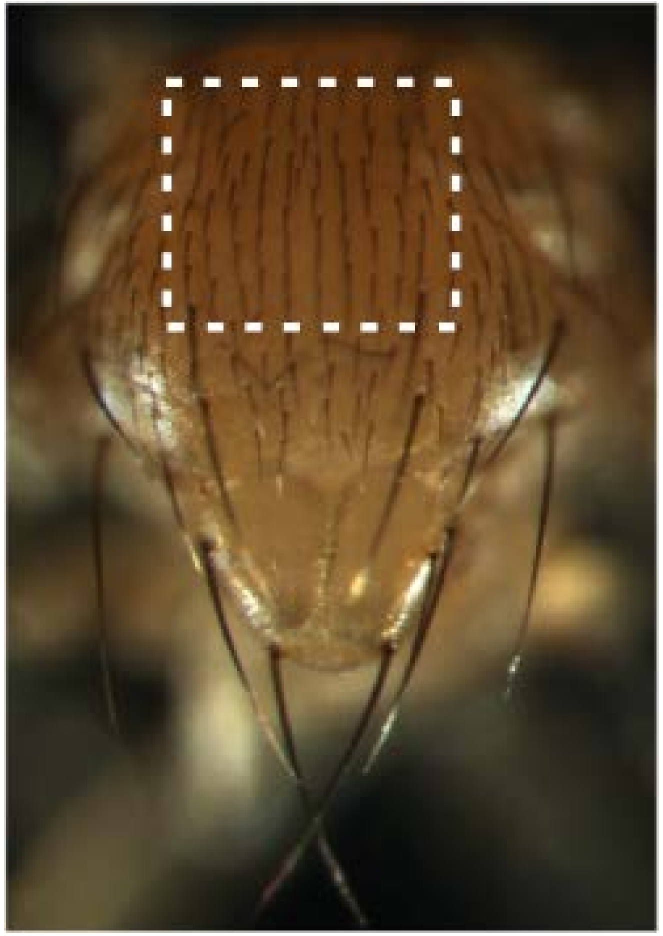
- Algorithm (proceed in rounds)
  - Each processor flips a coin with probability *1/d* where *d* is the number of its neighbors
  - If result is 0, do nothing
  - If result is 1, send to all other processors
    - If no collisions, Leader; all processes exit
    - Otherwise process with highest number of neighbors wins and becomes leader

Each process needs to know how many neighbors its neighbors have

Luby ,*SIAM J. Comput.* 1986
Alon et al *J. Algorithms* 1986

SOP selection in flies

- When the fly’s nervous system develops several cells are selected as sensory organ precursors (SOPs)
- These cells are later attached to the fly's sensory bristles
- Similar to the MIS requirements, in a highly accurate process each cell in a predefined cluster is either:
  - Selected as a SOP
  - Laterally inhibited by a neighboring SOP so it cannot become a SOP



Notch Receptor Signaling Controls Local Cell Fate Decisions

-- **Notch** (a receptor) and its ligands **Delta** and **Serrate** are conserved in all vertebrates as well as complex invertebrates (flies, worms, etc).

-- **Notch** controls cell fate decisions at many times and places in development.

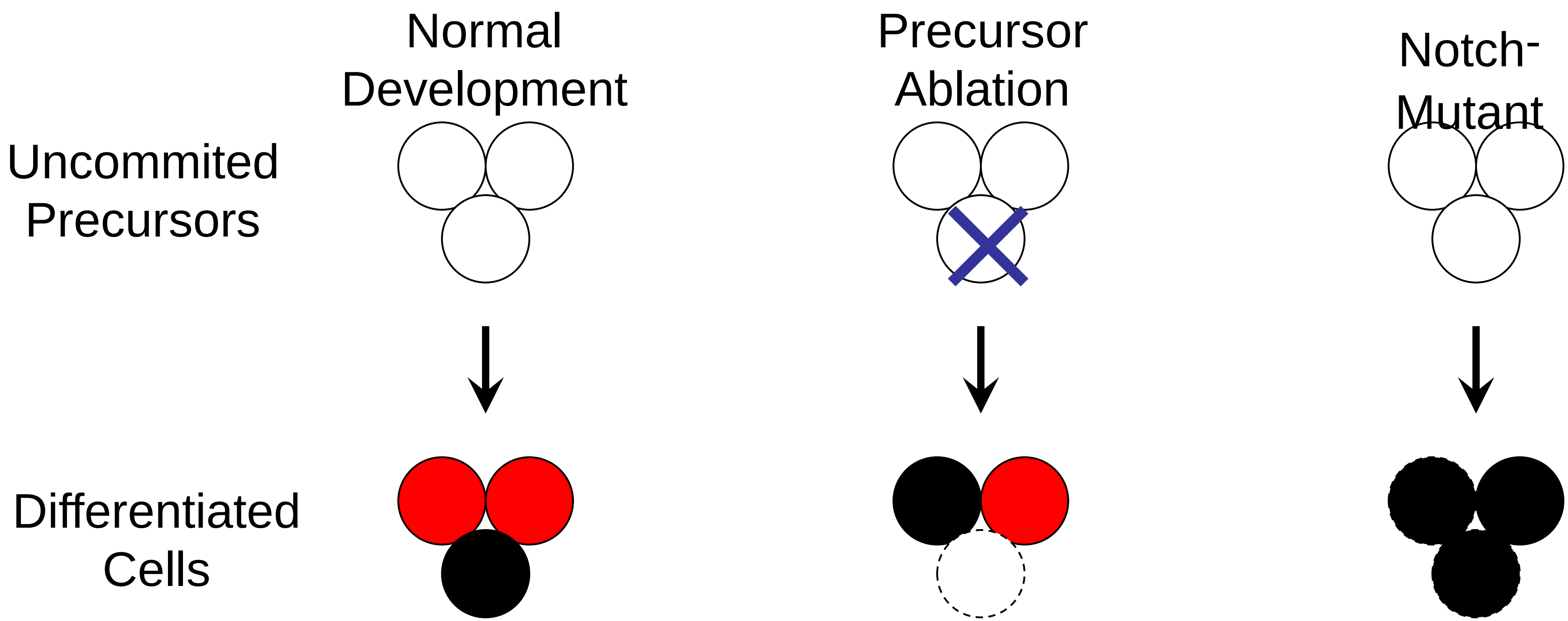
-- The most classic type of decision process mediated by **Notch** is *Lateral Inhibition*: a group of equipotent cells selects some to assume a specific fate, while others of the group are inhibited. *Inhibition requires Notch.*

Notch Receptor Signaling Controls Local Cell Fate Decisions

-- **Notch** (a receptor) and its ligands **Delta** and **Serrate** are conserved in all vertebrates as well as complex invertebrates (flies, worms, etc).

-- **Notch** controls cell fate decisions at many times and places in development.

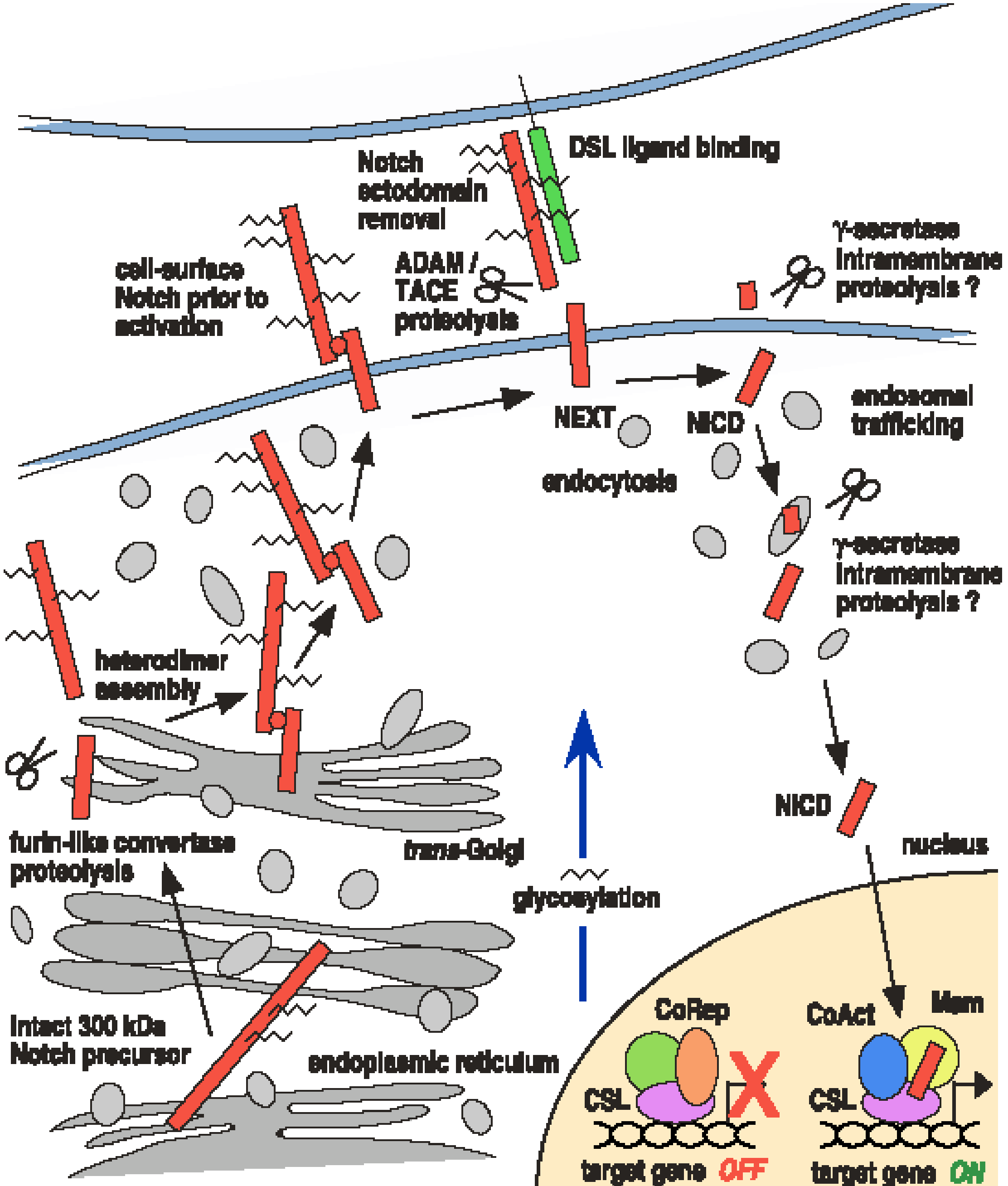
-- The most classic type of decision process mediated by **Notch** is *Lateral Inhibition*: a group of equipotent cells selects some to assume a specific fate, while others of the group are inhibited. *Inhibition requires Notch.*



-- Ligand for **Notch** produced in cell assuming *"black"* fate, acting to inhibit neighbors from assuming same fate.

-- **Notch** signaling thereby induces the *"red"* differentiated state

Canonical Notch Signaling Pathway



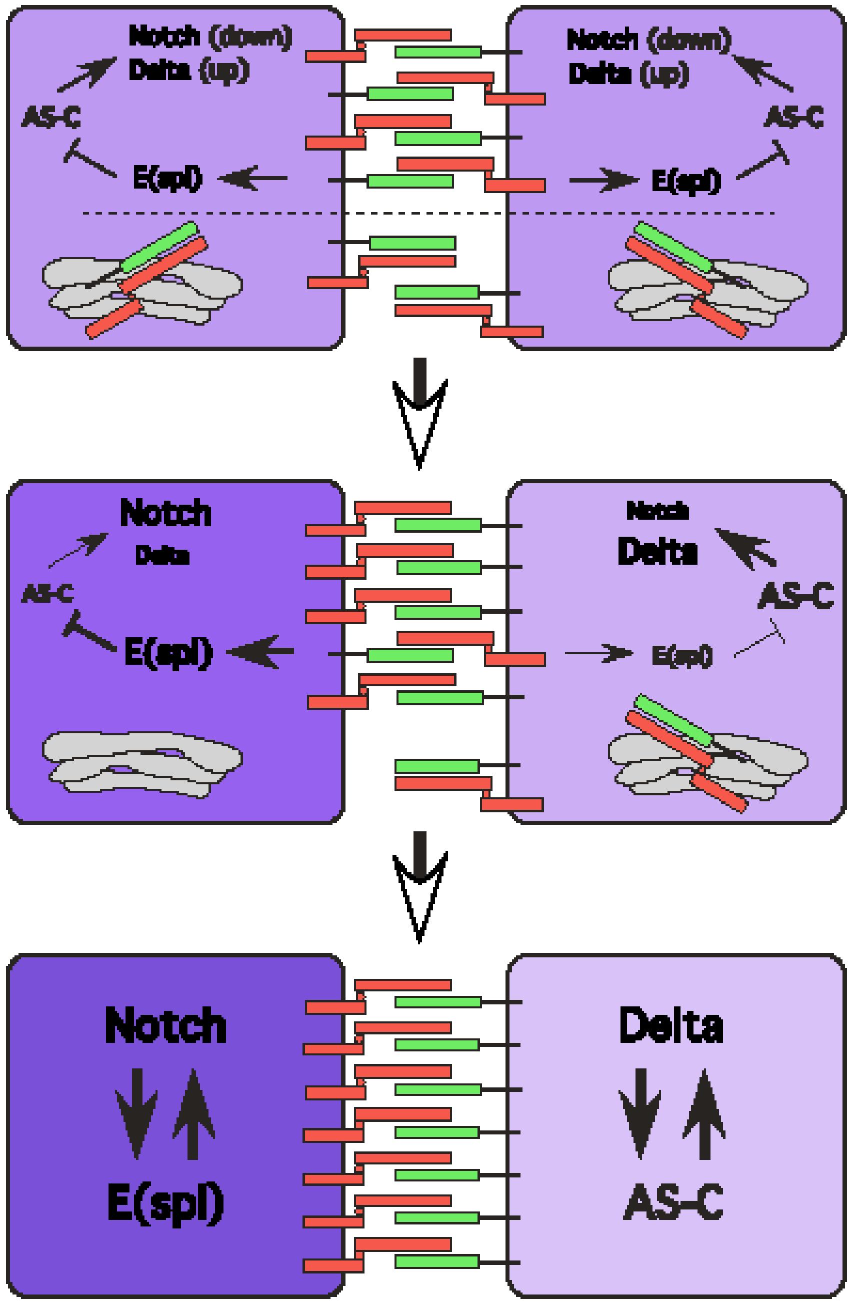
Transmembrane Notch binds a membrane-bound ligand (Delta or Serrate) on neighboring cell.

Induces Notch proteolysis, freeing cytoplasmic NICD.

NICD goes to nucleus, acting as cofactor for activation transcription.

Signaling is terminated by NICD phosphorylation, ubiquitination, and proteasomal degradation.

The Logic of Notch-Mediated Lateral Inhibition: Classical model



Equipotent precursors may have *random* small differences in Notch and Delta expression levels.

Notch signaling mediates a pathway through E(spl) and AS-C that enhances Notch expression and decreases Delta expression

Loss of Notch signaling decreases Notch expression and enhances Delta expression.

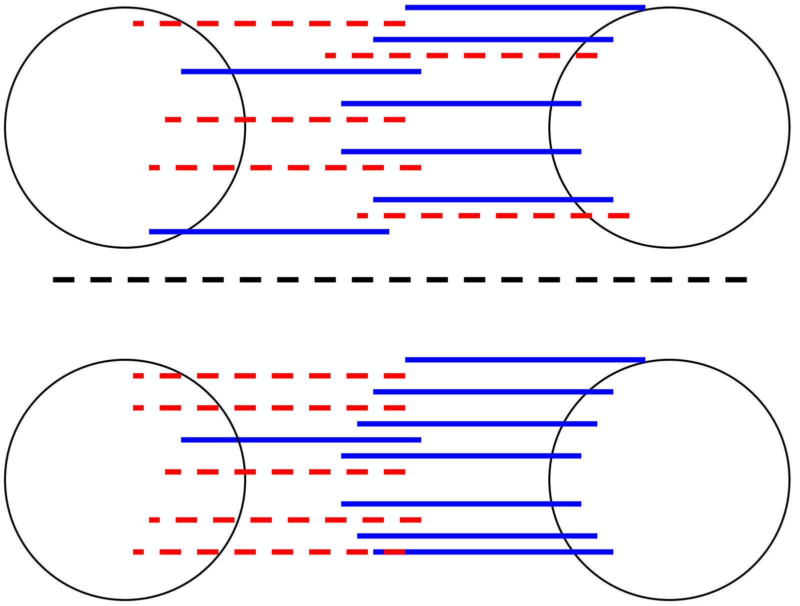
The AS-C transcription factor also controls adoption of a particular differentiated state, so that Notch signaling blocks this differentiated state.

Lateral inhibition

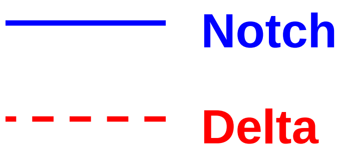
Notch

Delta

Trans model

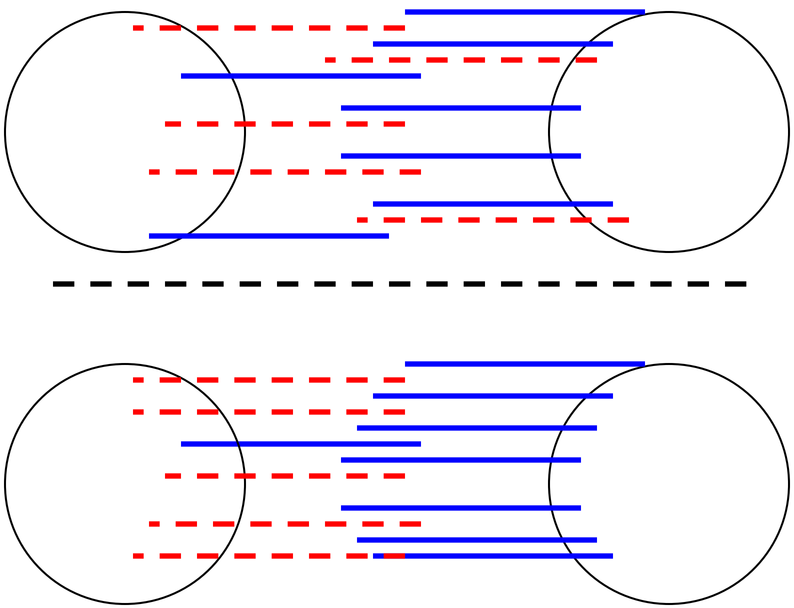


Trans vs. cis inhibition

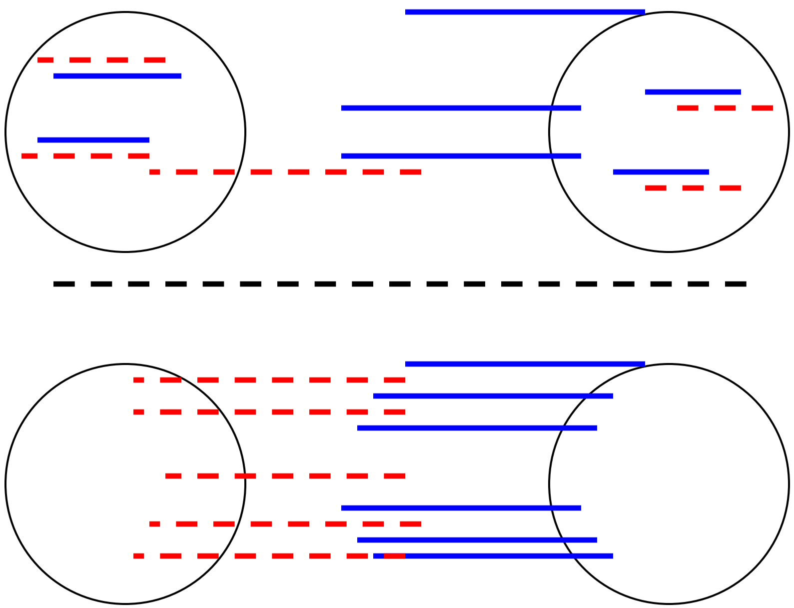


- Recent findings suggest that Notch is also suppressed in cis by delta’s from the same cell
- Only when a cell is ‘elected’ it communicates its decision to the other cell

Trans model



Cis+Trans model



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

Similarities between MIS and SOP selection

- Both are performed using a stochastic processes

  - Proven for MIS, experimentally validated for SOP
- Both are constrained by time

  - A cell that is not inhibited by certain time becomes a SOP
- Both only send messages if a node (cell) decides to join *A*

  - Reduces communication in computational systems, based on *cis* interactions for cells

Differences between SOP and MIS selection

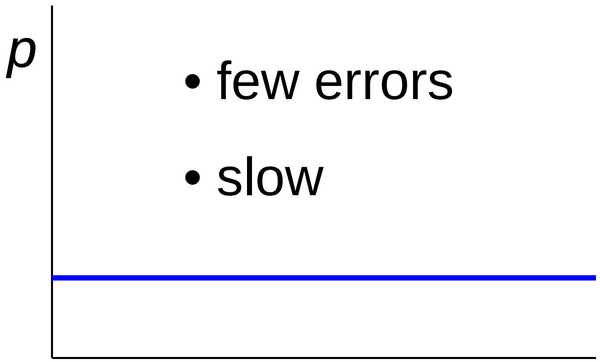
- In SOP selection cells do not know the status of their neighbors and the overall topology
- Messages in SOP selection are binary

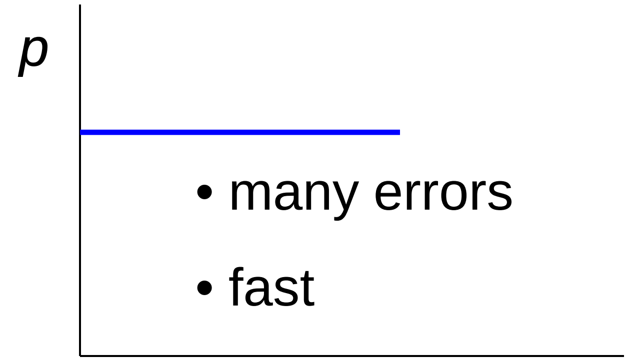
Can we improve current algorithms for MIS by understating how the biological process is performed?

Leader election

- Algorithm (proceed in rounds)
  - Each processor flips a coin with probability p
  - If result is 0, do nothing
  - If result is 1, send to all other processors
    - If no collisions, Leader; all processes exit
    - Otherwise proceed to next round

Probability distribution: Lateral inhibition and leader election





- Optimal solution for *n* processes: *p = 1/n*

$$p(only \quad 1 \quad elected) = \binom{n}{1}\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \cong \frac{1}{e}$$

- A unique leader is very likely elected after a constant (c) number of rounds (the probability that no leader is elected after c rounds is (1/e^c))

Algorithm for MIS

Each process needs to know how many neighbors it has

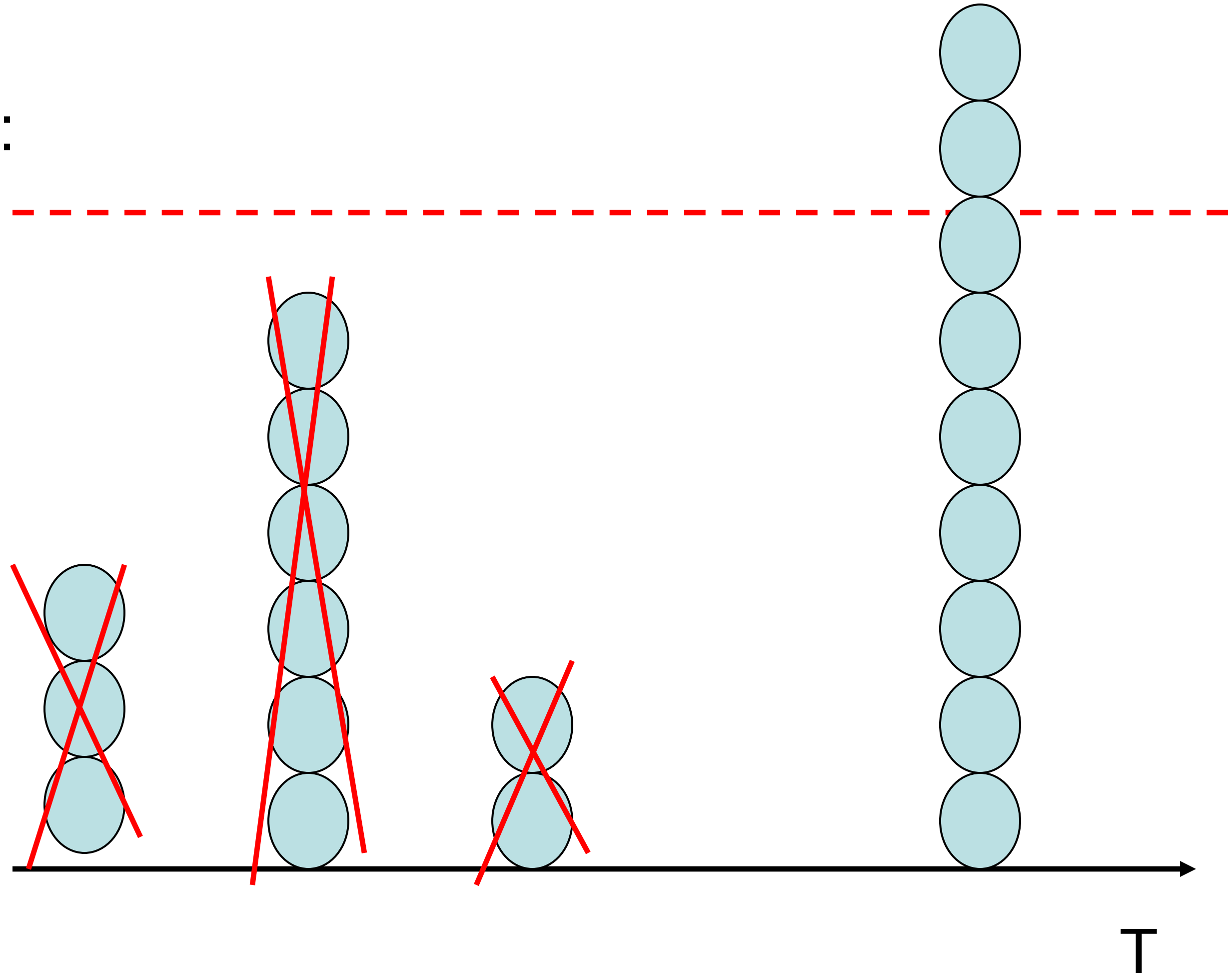
- Algorithm (proceed in rounds)
  - Each processor flips a coin with probability *1/d* where *d* is the number of its neighbors
  - If result is 0, do nothing
  - If result is 1, send to all other processors
    - If no collisions, Leader; all processes exit
    - Otherwise process with highest number of neighbors wins and becomes leader

Each process needs to know how many neighbors its neighbors have

But what about cells?

Possible mechanisms for stochastic process:

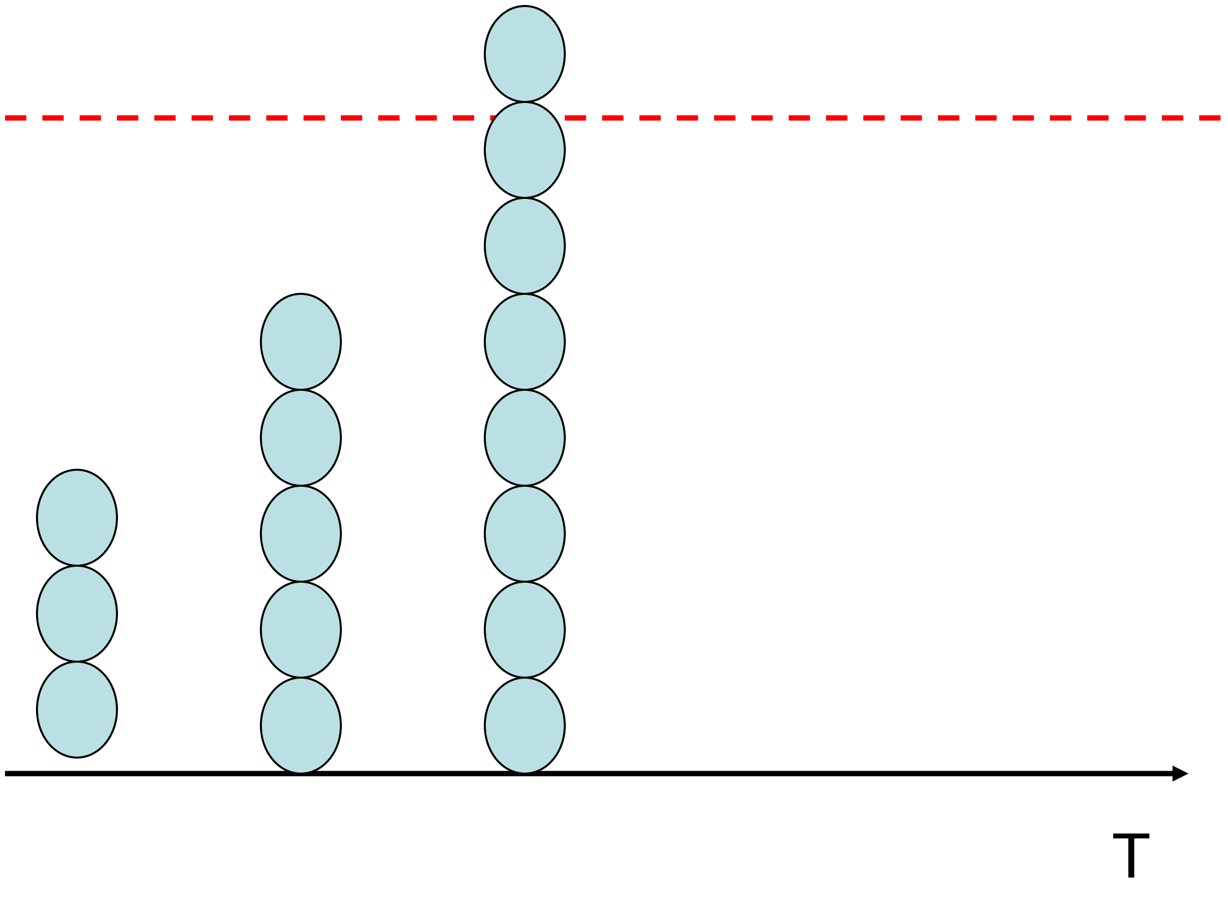
1. Burst model



But what about cells?

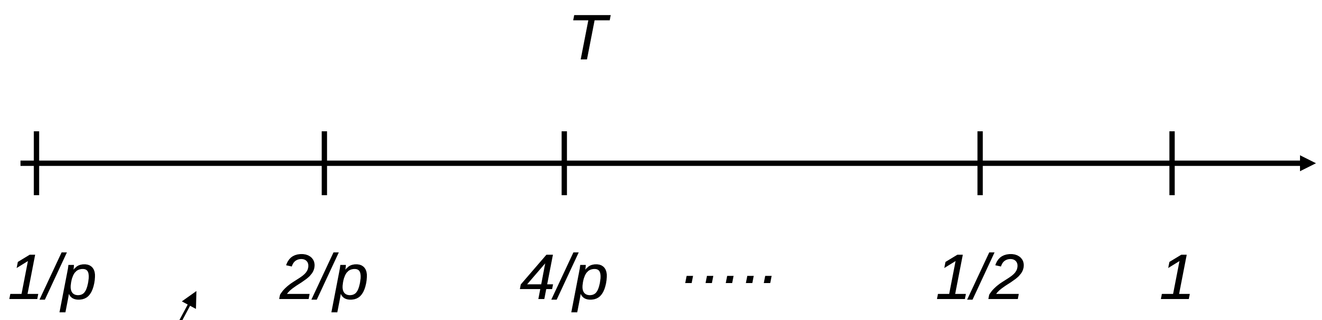
Possible mechanisms for stochastic process:

1. Burst model
2. Accumulation model



Using time

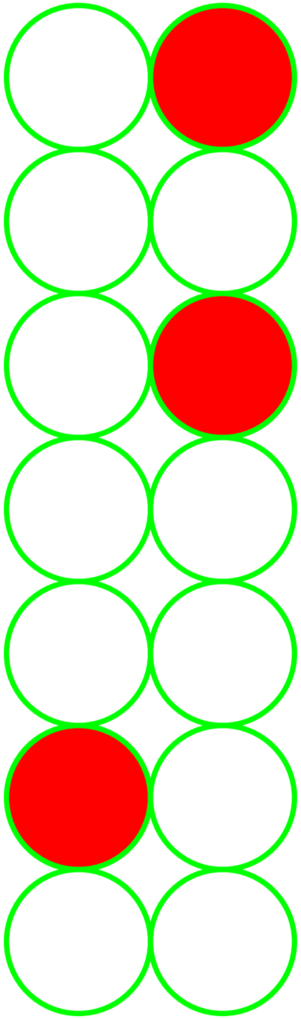
Increase probability with time



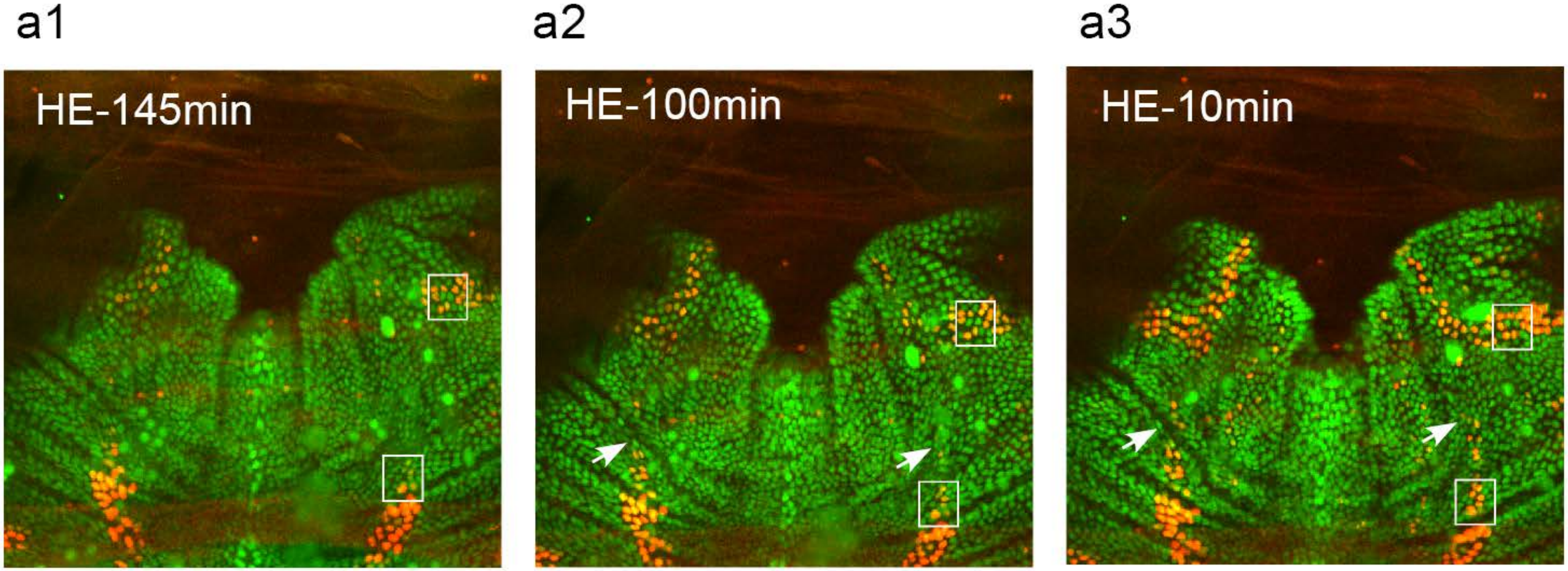
Decision to increase probability is only a function of time and does not depend on how many neighbors are competing

Do cells really do this?

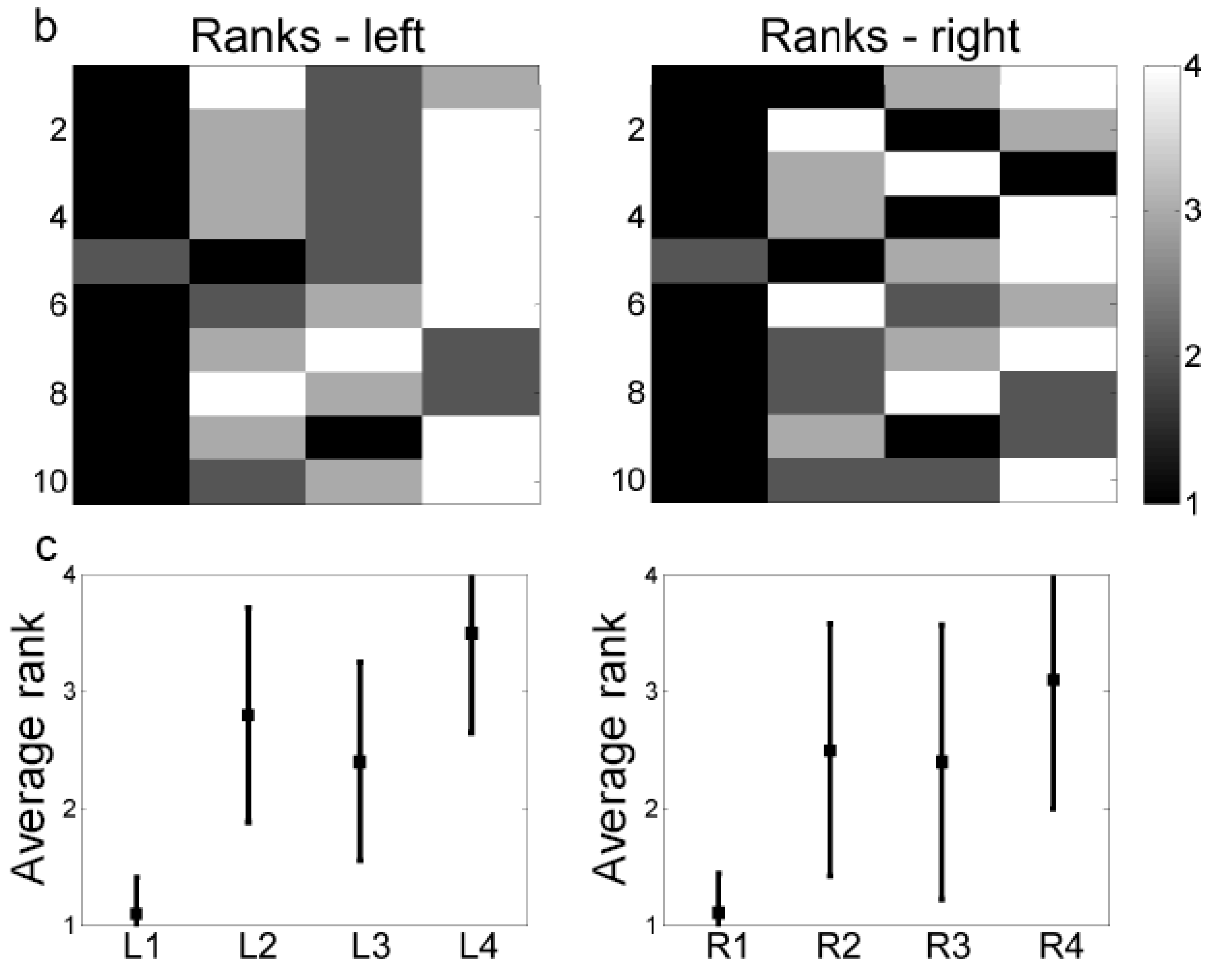
Possible way to test: Compute the time between first and second selection and compare to time between second and third selections



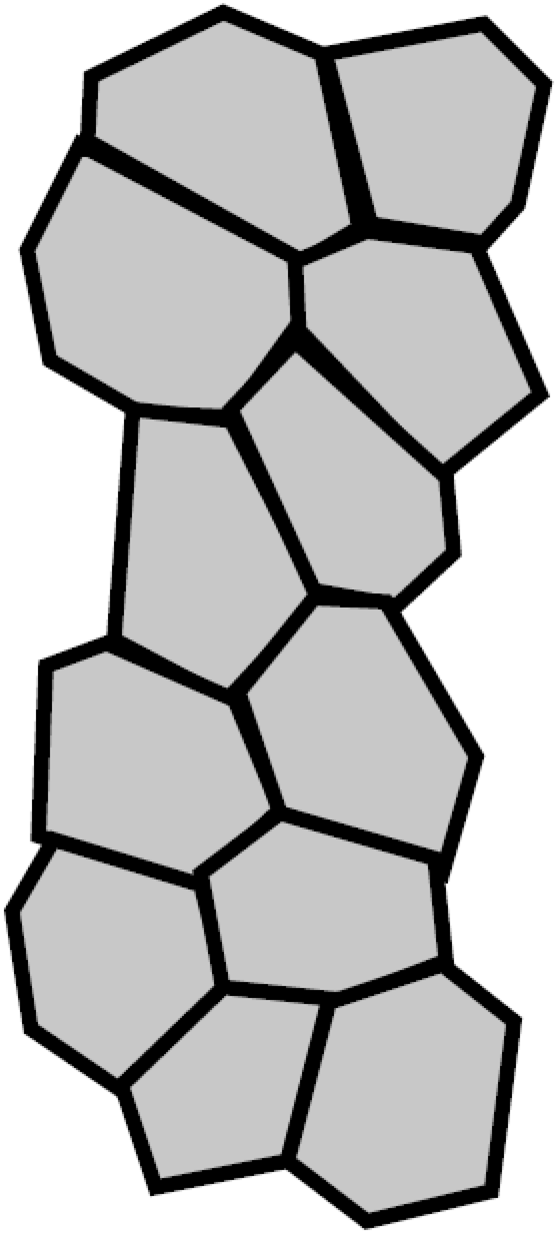
Movie



Observation 1: SOP selection is stochastic



Simulations



- 2 by 6 grid (also tried 2 by 7)
- Each cell touches all adjacent and diagonal neighbors

Simulations

- All models assume a cell becomes a SOP by accumulating the protein Delta until it passes some threshold

Four different models:

1. Accumulation

   - Accumulating Delta based on a Gaussian distribution

2. Fixed Accumulation

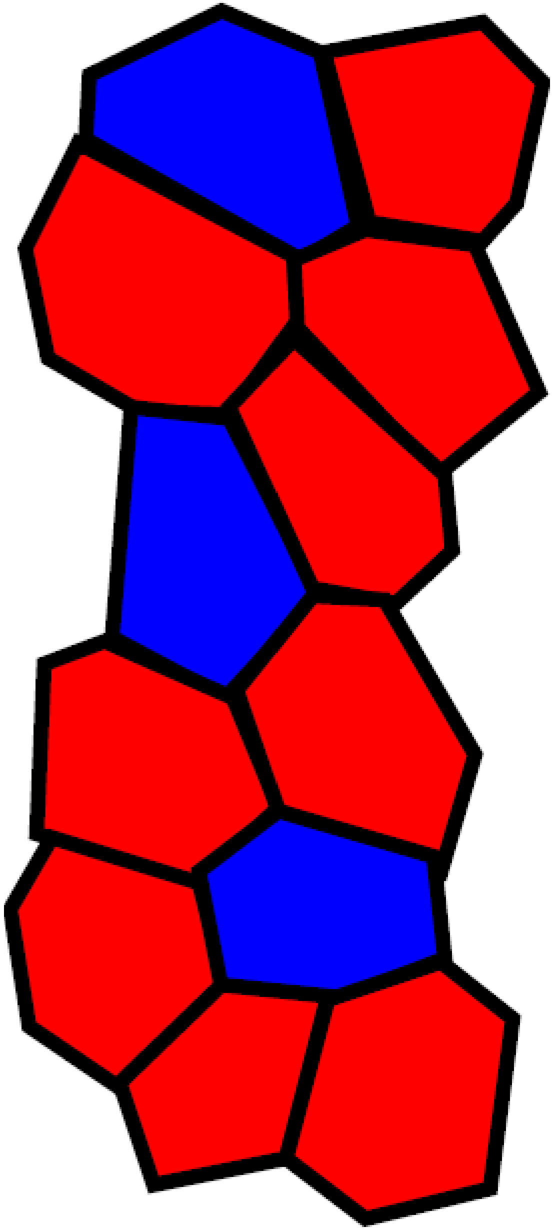
   - Randomly select an accumulation rate only once

3. Rate Change

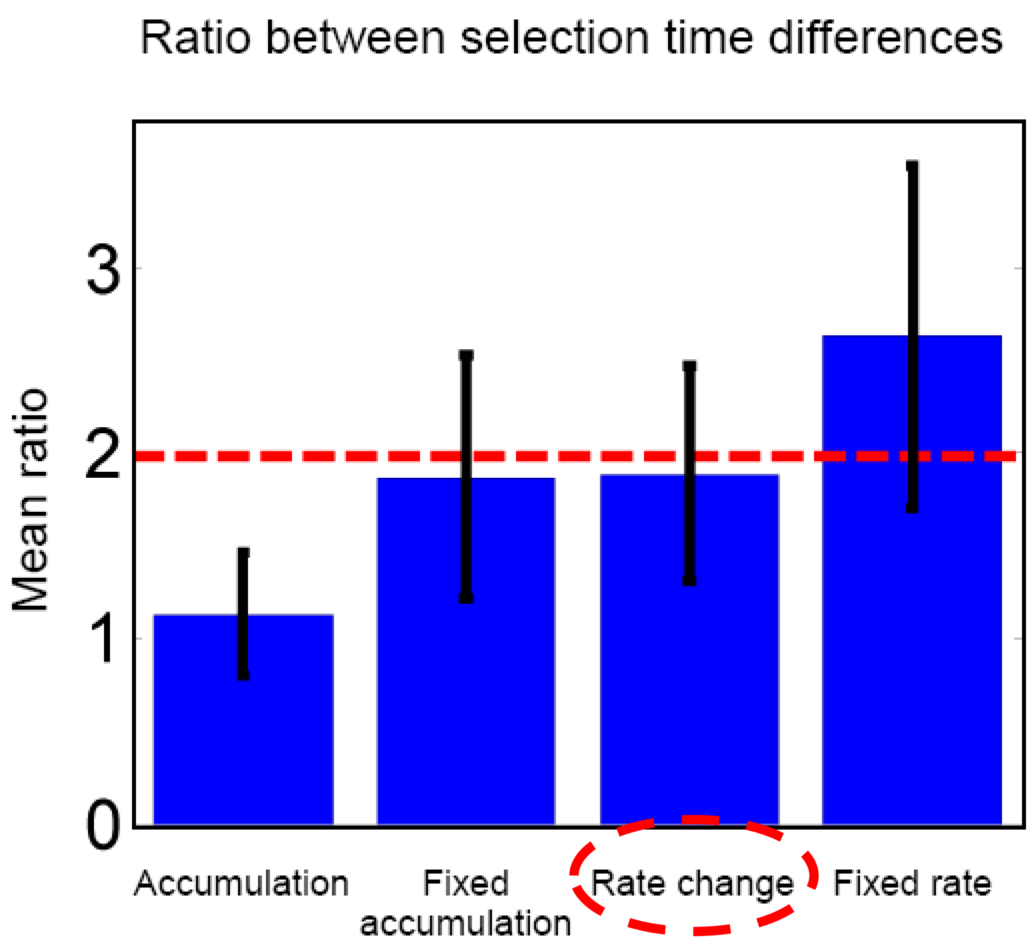
   - Increase accumulation probability as time goes by usind feedback loop

4. Fixed rate

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



Observation 2: Comparing the time of experimental and simulated selection



MIS Algorithm (revised)

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

D - upper bound on number of neighbors

- $p = 1/D$
- round = round +1
- if round > log(n)

  $p = p * 2$; round = 0 // we start a new phase
- Each processor flips a coin with probability p
  - If result is 0, do nothing
  - If result is 1, send to all other processors
    - If no collisions, Leader; all processes exit
    - Otherwise

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

- 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. With probability broadcast ***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**

Why does it work?

- Can show that by phase *i* there are no processes with more $n/2^i$ neighbors
- Overall running time is O(log (n) log(D)) where D is an upper bound on the number of neighbors
- For grids this is as fast as the best known algorithm for this problem.
- Message complexity is also extremely low: O(n)

Can also work with continuous increases

Demo