On the Collective Behaviors of Bio-Inspired Distributed Systems

Jury

- Robert Elsässer
- Pierre Fraigniaud
- Alain Jean-Marie
- Frederik Mallmann-Trenn

Supervisors

- Emanuele Natale
- Nicolas Nisse

PhD defense

Francesco d'Amore

COATI team

17 October 2022

Table of contents

Table of contents

Natural algorithms

Algorithms designed by evolution over millions of years [Chazelle, SODA 2009]

- migrating geese
- flocking cranes
- fish bait ball
- prey-predator systems
- synchronously flashing fireflies

flocking cranes

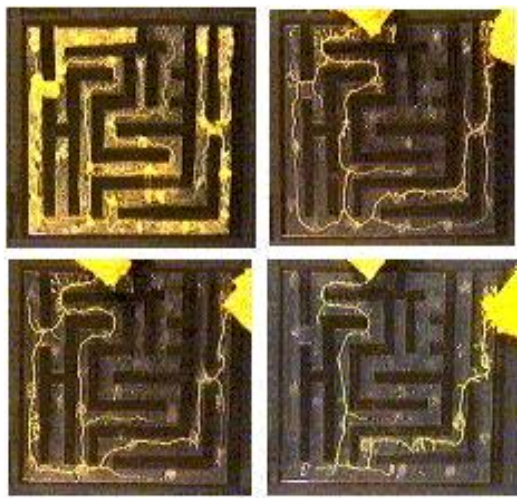
fish bait ball

"Georgia Aquarium Fish" by Mike Johnston

Natural algorithms

The computational lens help catching behavioral properties of biological systems

- bird flocking convergence time [Chazelle, SODA 2009]
- slime mold computing shortest paths [Bonifaci et al., Journal of Theoretical Biology 2012]



Slime mold
John S. Macneil, Science 2000

Flocking birds
"Flocking birds" by davepatten, CC BY-NC-SA 2.0.

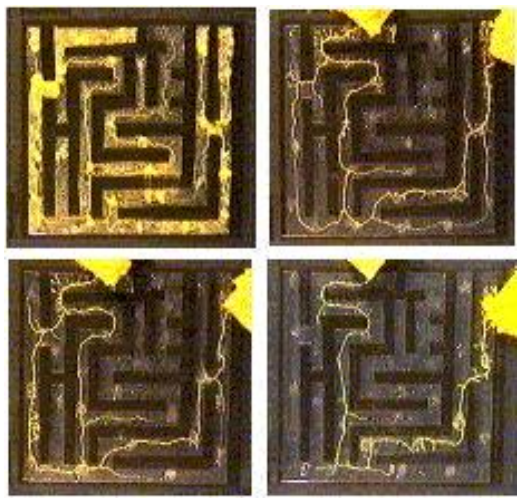
Natural algorithms

The computational lens help catching behavioral properties of biological systems

- bird flocking convergence time [Chazelle, SODA 2009]
- slime mold computing shortest paths [Bonifaci et al., Journal of Theoretical Biology 2012]

On the other hand, biological systems help designing new algorithms for well-known problems

- distributed maximal independent set algorithm from the fly's nervous system [Afek et al., SCIENCE 2011]



Slime mold
John S. Macneil, Science 2000

Flocking birds
"Flocking birds" by davepatten, CC BY-NC-SA 2.0.

Distributed computing tasks

Often, systems of interacting agents performing collective tasks

Other than MIS and bird flocking:

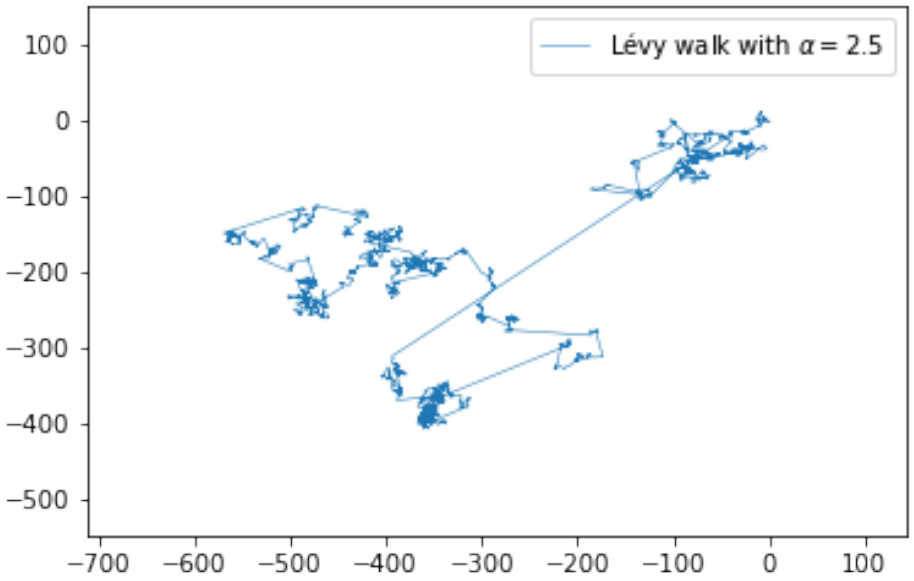
- Information spreading: schooling fish [Rosenthal et al., PNAS 2015]
- Reaching agreement: molecules [Carrol, Nature Immunology 2004], bacteria [Bassler, Cell 2002], social insects [Franks et al., 2002] (e.g. bees [Reina et al., Physical Review E 2017])
- Collective search: ants and bees [Feinermant et Korman, Distributed Computing 2017]

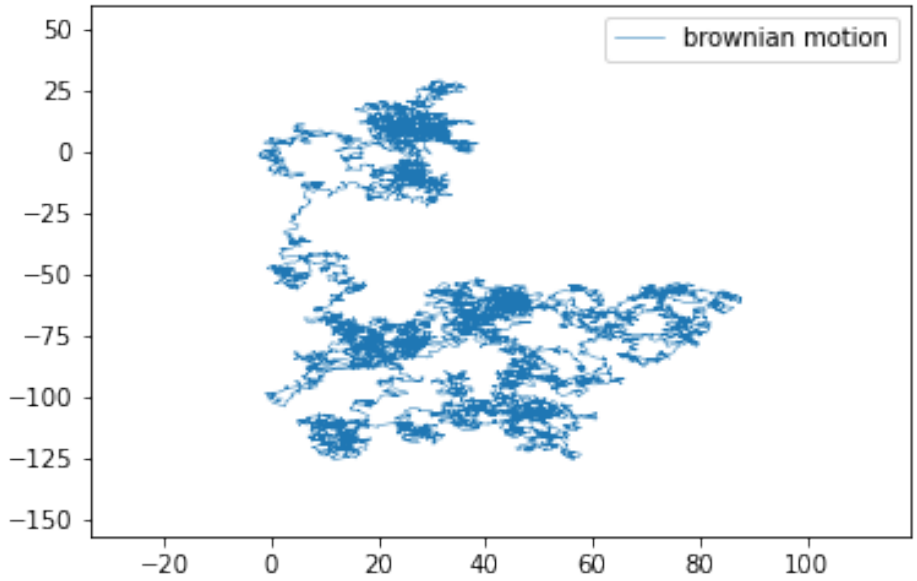
Foraging ants

"The Blueberry Hunters" by bob in swamp, CC BY 2.0.

Table of contents

The Lévy walk





Lévy walk (informal):
A Lévy walk is a random walk whose step-length density distribution is proportional to a power-law, namely, for each $d \in \mathbb{R}^+$, $f(d) \sim 1/d^\alpha$, *for some* $\alpha > 1$

Note: the speed of the walk is constant

Movement models and foraging theory

Lévy walks are used to model **movement patterns** [Reynolds, Biology Open 2018]

Examples:

- T cells within the brain
- swarming bacteria
- midge swarms
- termite broods
- schools of fish
- Australian desert ants
- a variety of molluscs

Desert ants

Rhytidoponera mayri workers. Credit: Associate Professor Heloise Gibb, La Trobe University

Movement models and foraging theory

Lévy walks are used to model **movement patterns** [Reynolds, Biology Open 2018]

Examples:

- T cells within the brain
- swarming bacteria
- midge swarms
- termite broods
- schools of fish
- Australian desert ants
- a variety of molluscs

Desert ants

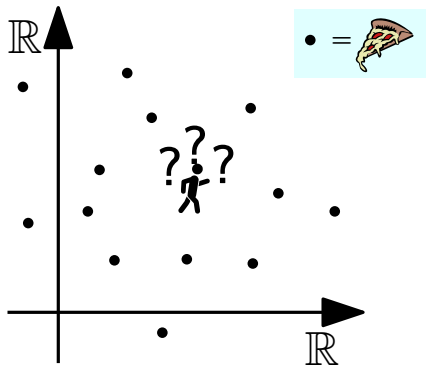
Rhytidoponera mayri workers. Credit: Associate Professor Heloise Gibb, La Trobe University

Widely employed in the foraging theory

- how animals search for food

Lévy walk optimality

Foraging theory



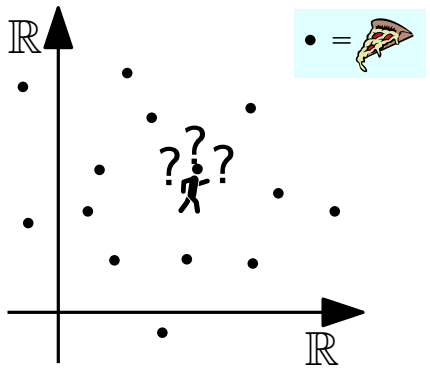
- distribution of food locations in $\mathbb{R}^n$
- uninformed agent searching for food

maximum expected food discovery rate

[Viswanathan et al., Nature 1999]: Lévy walk with exponent $\alpha = 2$ is optimal in any dimension, with some assumptions

Lévy walk optimality

Foraging theory

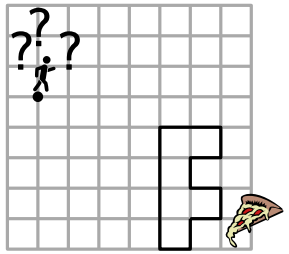


- distribution of food locations in $\mathbb{R}^n$
- uninformed agent searching for food

maximum expected food discovery rate

[Viswanathan et al., Nature 1999]: Lévy walk with exponent $\alpha = 2$ is optimal in any dimension, with some assumptions

Other search problems



- a target in the bidimensional torus $\mathbf{T}$
- uninformed agent searching for it

as fast as possible

[Guinard et Korman, Sciences Advances 2021]: (truncated) Lévy walk with exponent $\alpha = 2$ is optimal

The Lévy flight foraging hypothesis

Formulation of an evolutionary hypothesis

The Lévy flight foraging hypothesis [Viswanathan et al., Physics of Life Reviews 2008]: since Lévy flights/walks optimize random searches, biological organisms must have therefore evolved to exploit Lévy flights/walks

- Special exponent $\alpha = 2$

The Lévy flight foraging hypothesis

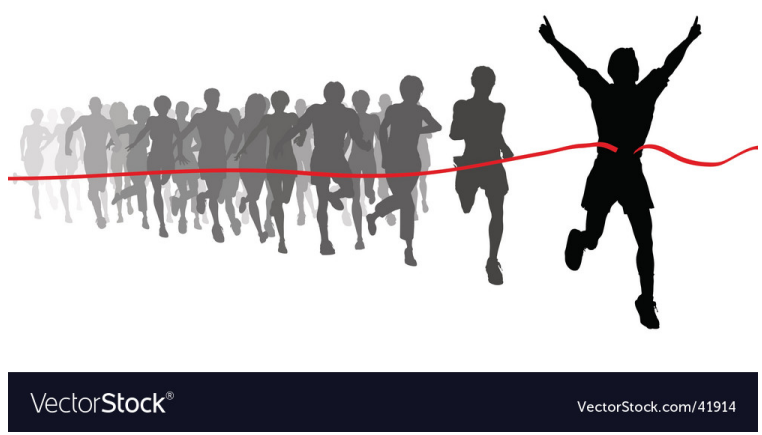
Formulation of an evolutionary hypothesis

The Lévy flight foraging hypothesis [Viswanathan et al., Physics of Life Reviews 2008]: since Lévy flights/walks optimize random searches, biological organisms must have therefore evolved to exploit Lévy flights/walks

- Special exponent $\alpha = 2$

We test this hypothesis by focusing on a distributed search problem:

- the ANTS (Ants Nearby Treasure Search) problem



The ANTS problem

Introduced by [Feinerman et al., PODC 2012]:

Setting:

- k (mutually) independent agents start moving on $\mathbb{Z}^2$ from the origin
- time is synchronous and marked by a global clock
- one special node $\mathcal{P} \in \mathbb{Z}^2$, the *target*, placed by an adversary at unknown (Manhattan) distance ℓ from the origin

The ANTS problem

Introduced by [Feinerman et al., PODC 2012]:

Setting:
- k (mutually) independent agents start moving on $\mathbb{Z}^2$ from the origin
- time is synchronous and marked by a global clock
- one special node $\mathcal{P} \in \mathbb{Z}^2$, the *target*, placed by an adversary at unknown (Manhattan) distance ℓ from the origin

Task: find the target as fast as possible

Lower bound: for any $k \geq 1$, and for any search algorithm $\mathcal{A}$, the hitting time to find $\mathcal{P}$ is $\Omega\left(\ell^2/k + \ell\right)$ both with constant probability and in expectation

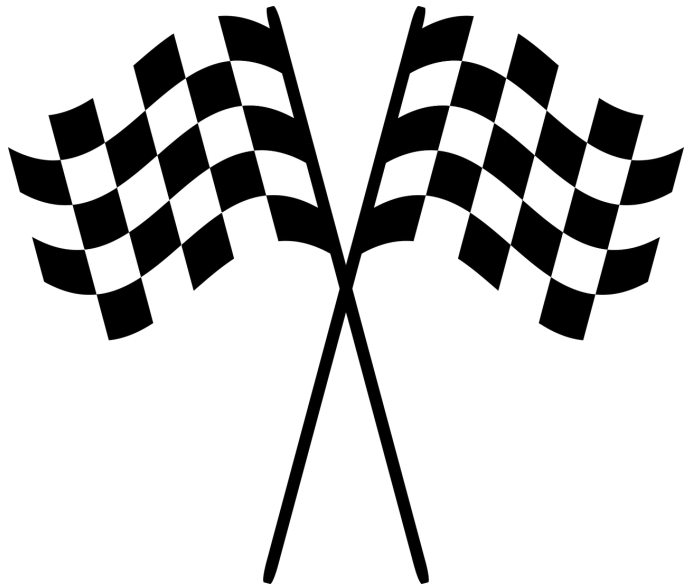


Image by OpenClipart-Vectors from Pixabay

Our contributions

Based on the work [Clementi et al., PODC 2021]

(i) first definition of Lévy walk in the discrete setting in $\mathbb{Z}^2$, time-homogeneus

(ii) first analysis of the hitting time distribution of k parallel walks

(iii) the Lévy walks can be employed to give a natural, almost-optimal solution to the ANTS problem (no advice, no communication)

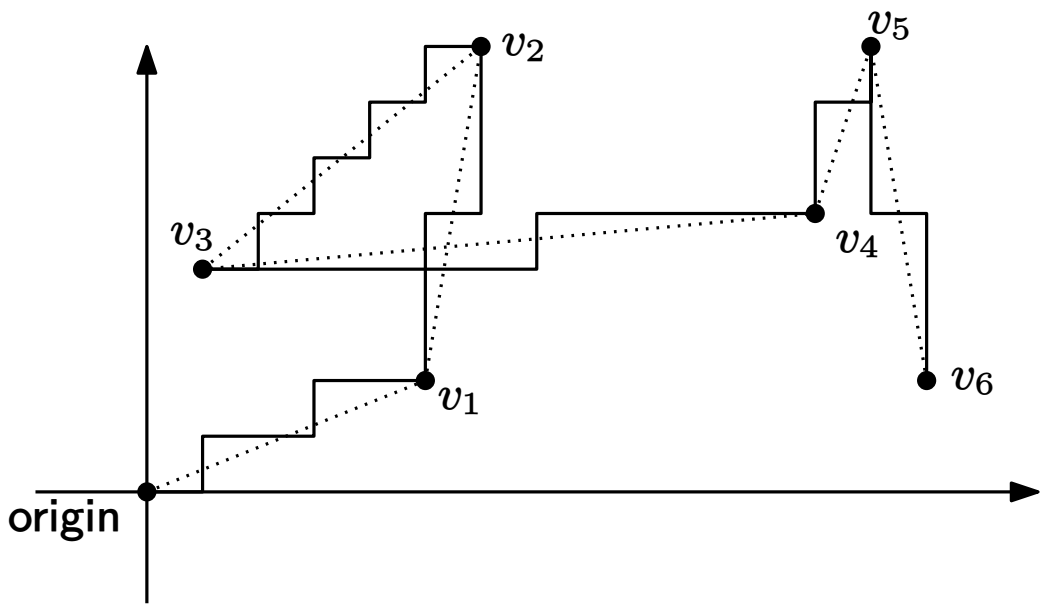
Our contributions

(i) DEFINITION OF DISCRETE LÉVY WALK

(ii) ANALYSIS OF THE PARALLEL HITTING TIME

(iii) ALGORITHM FOR THE ANTS PROBLEM

Discrete Lévy walk



direction chosen
path performed

Six jumps of a discrete Lévy walk

Let $\alpha > 1$ be a real value

Lévy walk: the agent

a) chooses a distance $d \in \mathbb{N}$ as follows: $d = 0$ w.p. $1/2$, and $d \geq 1$ w.p. c_α / d^α
b) chooses a destination u.a.r. among those at distance d
c) walks along an approximating path for d steps, one edge at a time, crossing d nodes
d) repeats the procedure

Our contributions

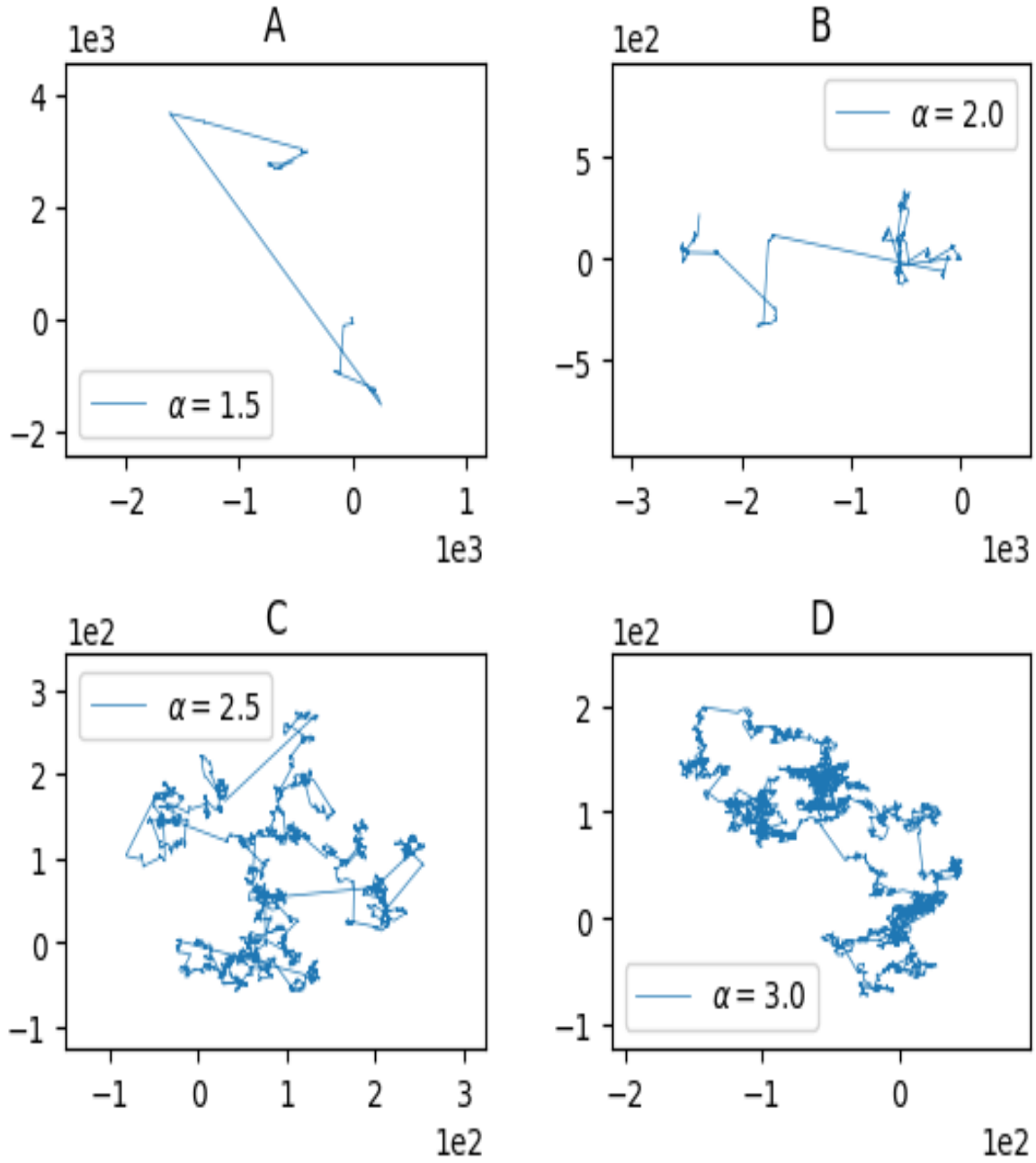
(i) DEFINITION OF DISCRETE LÉVY WALK

(ii) ANALYSIS OF THE PARALLEL HITTING TIME

(iii) ALGORITHM FOR THE ANTS PROBLEM

α-behavior of Lévy walks

- ballistic diffusion: fig.s A and B
  - $1 < \alpha \leq 2$
- super diffusion: fig. C
  - $2 < \alpha < 3$
- normal diffusion: fig. D
  - $3 \leq \alpha$



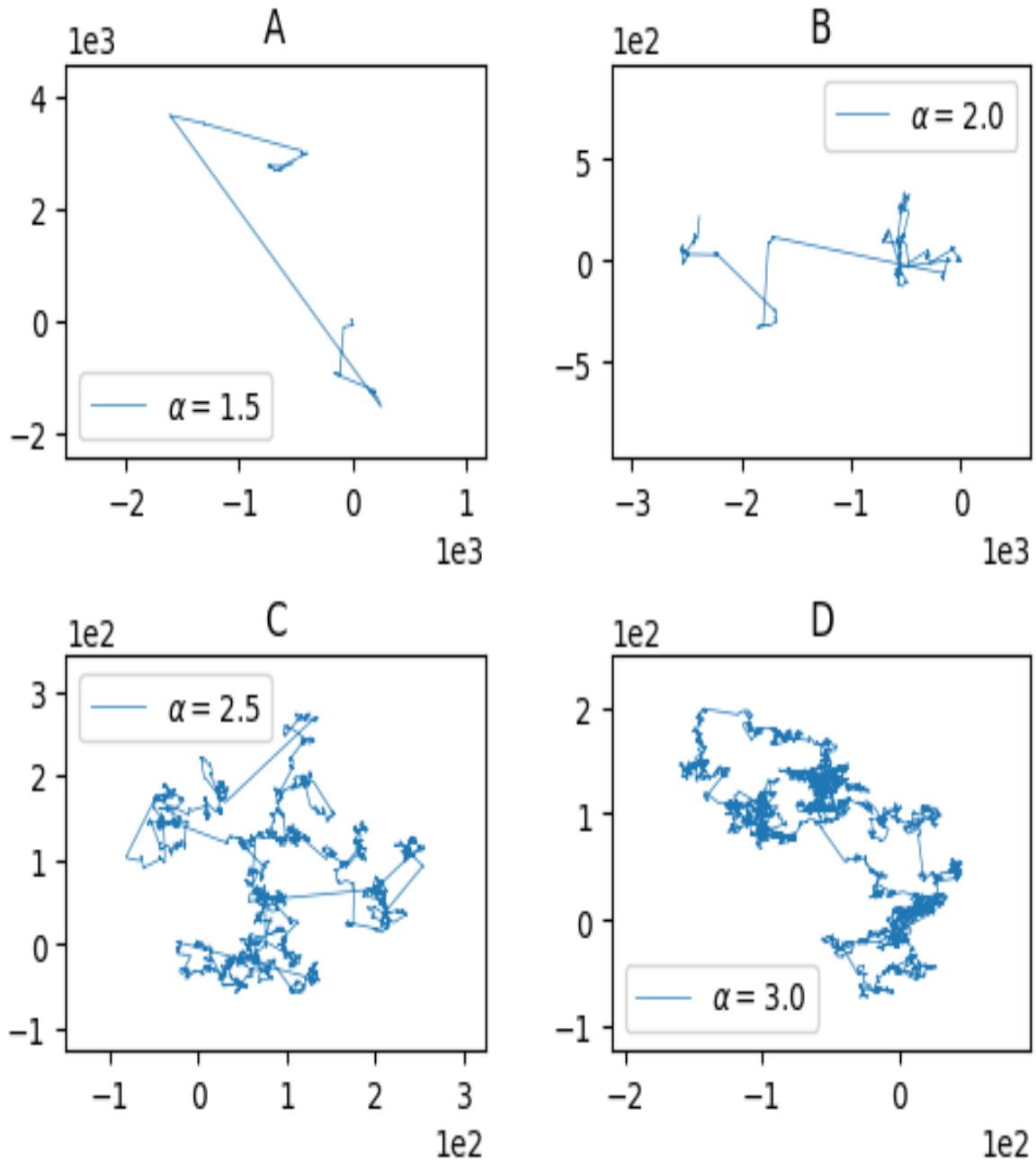
α-behavior of Lévy walks

- ballistic diffusion: fig.s A and B
  - $1 < \alpha \leq 2$
- super diffusion: fig. C
  - $2 < \alpha < 3$
- normal diffusion: fig. D
  - $3 \leq \alpha$

Expected jump-length

$\int_1^\infty x^{-\alpha+1} \mathrm{d}x$

- $+\infty$ if $\alpha \leq 2$
- $\frac{1}{\alpha-2}$ if $\alpha > 2$



α-behavior of Lévy walks

- ballistic diffusion: fig.s A and B
  - $1 < \alpha \leq 2$
- super diffusion: fig. C
  - $2 < \alpha < 3$
- normal diffusion: fig. D
  - $3 \leq \alpha$

Expected jump-length

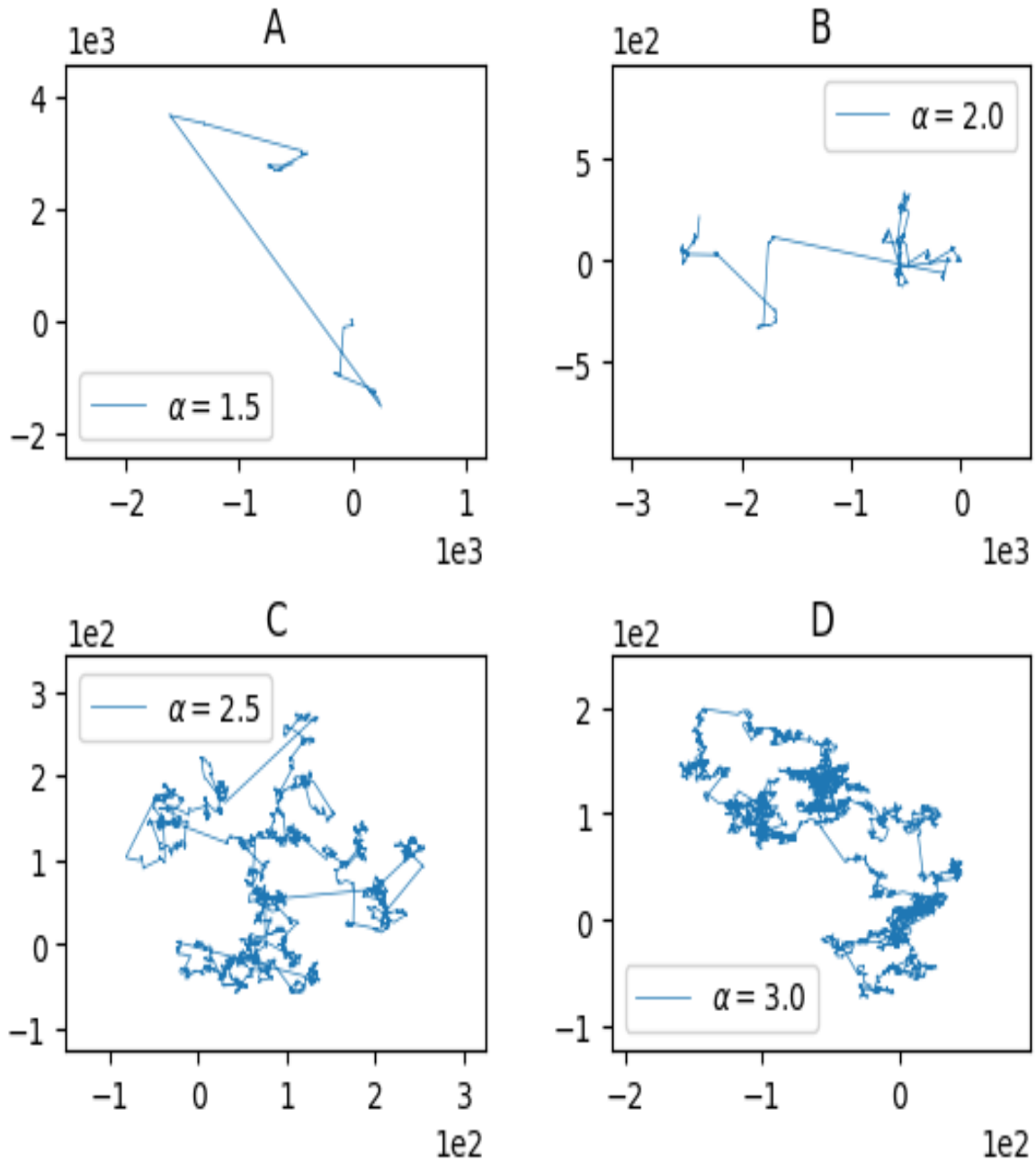
$\int_1^\infty x^{-\alpha+1} \mathrm{d}x$

- $+\infty$ if $\alpha \leq 2$
- $\frac{1}{\alpha-2}$ if $\alpha > 2$

Jump-length second moment

$\int_1^\infty x^{-\alpha+2} \mathrm{d}x$

- $+\infty$ if $\alpha \leq 3$
- $\frac{1}{\alpha-3}$ if $\alpha > 3$

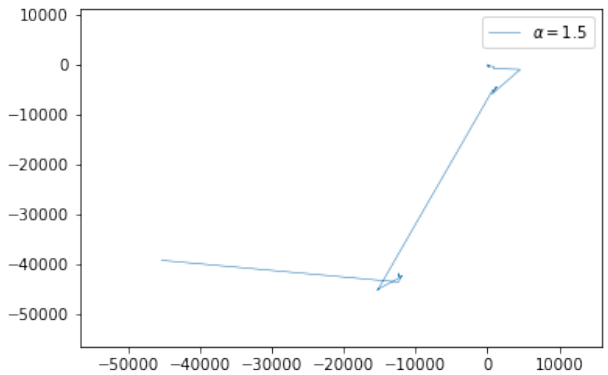


Three ranges for k and ℓ

Recall: ℓ target distance, k number of agents, $\Omega\left(\ell^2/k+\ell\right)$ time lower bound

Three different possible settings:

1. **Close target**: $\ell \leq k/\text{polylog}\,(k)$



Best strategy = ballistic walks

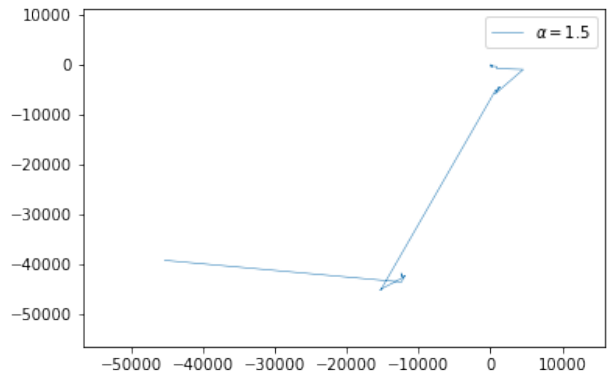
- any α in $(1,2]$

Three ranges for k and ℓ

Recall: ℓ target distance, k number of agents, $\Omega\left(\ell^2/k+\ell\right)$ time lower bound

Three different possible settings:

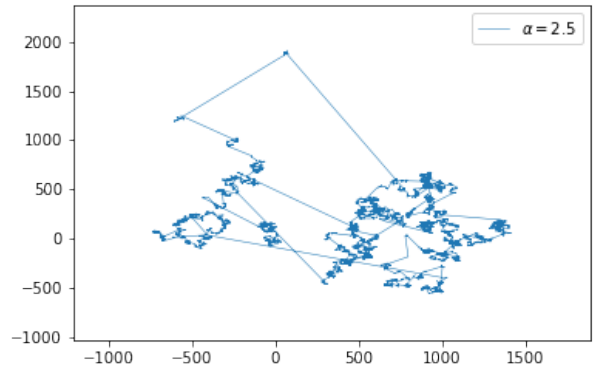
1. **Close target**: $\ell \leq k/\text{polylog}\,(k)$



Best strategy = ballistic walks

- any α in $(1,2]$

2. **Far target**: $k/\text{polylog}\,(k) \leq \ell \leq \exp\left(k^{\Theta(1)}\right)$



Best strategy = super-diffusive range

- $\alpha = \underbrace{3-\log k/\log \ell}_{\alpha^\star} + \mathcal{O}\left(\frac{\log\log \ell}{\log \ell}\right)$

Three ranges for k and ℓ

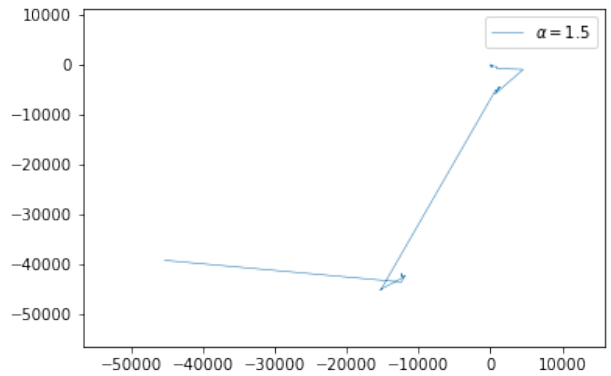
Recall: ℓ target distance, k number of agents, $\Omega\left(\ell^2/k+\ell\right)$ time lower bound

Three different possible settings:

1. **Close target**: $\ell \leq k/\text{polylog}\,(k)$

Best strategy = ballistic walks

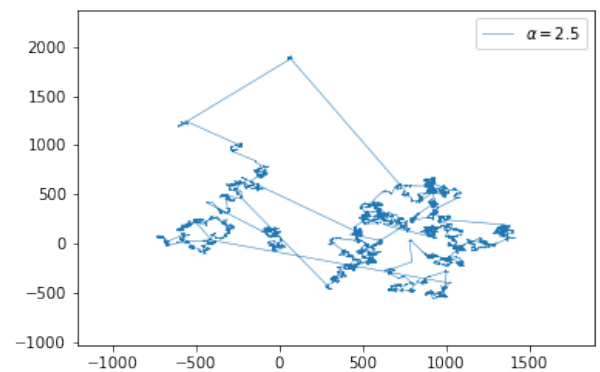
- any α in $(1,2]$



2. **Far target**: $k/\text{polylog}\,(k) \leq \ell \leq \exp\left(k^{\Theta(1)}\right)$

Best strategy = super-diffusive range

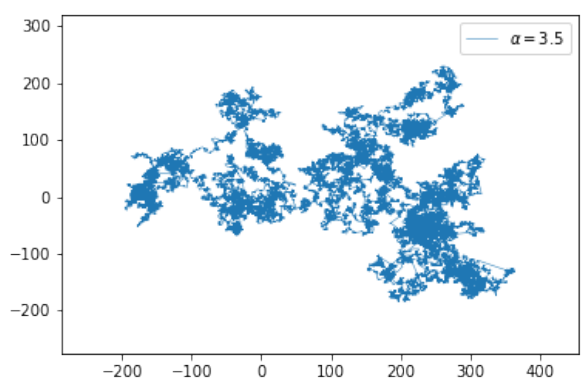
- $\alpha = \underbrace{3 - \log k/\log \ell}_{\alpha^\star} + \mathcal{O}\left(\frac{\log\log \ell}{\log \ell}\right)$



3. **Very far target**: $\exp\left(k^{\Theta(1)}\right) \leq \ell$

Best strategy = diffusive walks

- $[3,+\infty)$ (brownian-like behavior)



Our contributions

(i) DEFINITION OF DISCRETE LÉVY WALK

(ii) ANALYSIS OF THE PARALLEL HITTING TIME

- proof's sketch

(iii) ALGORITHM FOR THE ANTS PROBLEM

Far target case: proof's sketch

$\alpha^\star = 3 - \log k / \log \ell$: super-diffusive range $\in (2, 3)$

Far target case: proof's sketch

$\alpha^\star = 3 - \log k / \log \ell$: super-diffusive range $\in (2,3)$

Reverse engineering: $\alpha \in (2,3)$

- analyze the hitting time of a single Lévy flight (Markov chain on $\mathbb{Z}^2$)
- derive bounds for a single Lévy walk through a coupling argument
- exploit independence to get results for k walks

Remark: Here, we DON'T show why other values for α are worse than $\alpha^\star$

Far target case: proof's sketch

$\alpha^\star = 3 - \log k / \log \ell$: super-diffusive range $\in (2, 3)$

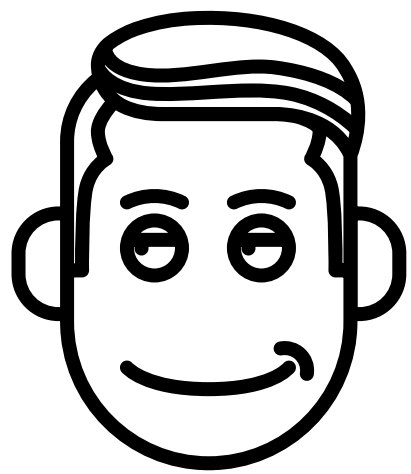
Reverse engineering: $\alpha \in (2, 3)$

- analyze the hitting time of a single Lévy flight (Markov chain on $\mathbb{Z}^2$)
- derive bounds for a single Lévy walk through a coupling argument
- exploit independence to get results for k walks

Remark: Here, we DON'T show why other values for α are worse than $\alpha^\star$

- $\mathcal{E}_t$ event: first t jumps have length $\leq (t \log t)^{\frac{1}{\alpha - 1}}$

Coupling argument: if a Lévy flight finds a node with probability p in time t (conditional on $\mathcal{E}_t$), then a Lévy walk finds it with prob. $\approx p$ in time $\mathcal{O}(t)$



Single Lévy flight analysis

- $\mathcal{P}$ target node

Let
- $Z_u(t) =$ random variable of number of visits in u until time t
- $\mathcal{E}_t =$ the event first t jumps have length $\leq (t \log t)^{\frac{1}{\alpha-1}}$
- $a_t = \mathbb{E}\left[Z_{(0,0)}(t) \mid \mathcal{E}_t\right]$
- $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 \mid \mathcal{E}_t\right)$

Single Lévy flight analysis

- $\mathcal{P}$ target node

Let
- $Z_u(t) =$ random variable of number of visits in u until time t
- $\mathcal{E}_t =$ the event first t jumps have length $\leq (t \log t)^{\frac{1}{\alpha-1}}$
- $a_t = \mathbb{E}\left[Z_{(0,0)}(t) \mid \mathcal{E}_t\right]$
- $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 \mid \mathcal{E}_t\right)$

Lemma: $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 \mid \mathcal{E}_t\right) \geq \mathbb{E}\left[Z_{\mathcal{P}}(t) \mid \mathcal{E}_t\right] / a_t$

Comes from two facts

(i) $\mathbb{E}\left[Z_{\mathcal{P}}(t) | \mathcal{E}_t\right] = \mathbb{E}\left[Z_{\mathcal{P}}(t) | Z_{\mathcal{P}}(t) > 0, \mathcal{E}_t\right] \cdot \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 | \mathcal{E}_t\right)$

(ii) $\mathbb{E}\left[Z_{\mathcal{P}}(t) | Z_{\mathcal{P}}(t) > 0, \mathcal{E}_t\right] \leq a_t$

Single Lévy flight analysis

- $\mathcal{P}$ target node

Let
- $Z_u(t) =$ random variable of number of visits in u until time t
- $\mathcal{E}_t =$ the event first t jumps have length $\leq (t \log t)^{\frac{1}{\alpha-1}}$
- $a_t = \mathbb{E}\left[Z_{(0,0)}(t) \mid \mathcal{E}_t\right]$
- $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 \mid \mathcal{E}_t\right)$

Lemma: $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 \mid \mathcal{E}_t\right) \geq \mathbb{E}\left[Z_{\mathcal{P}}(t) \mid \mathcal{E}_t\right] / a_t$

Comes from two facts

(i) $\mathbb{E}\left[Z_{\mathcal{P}}(t) | \mathcal{E}_t\right] = \mathbb{E}\left[Z_{\mathcal{P}}(t) | Z_{\mathcal{P}}(t) > 0, \mathcal{E}_t\right] \cdot \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 | \mathcal{E}_t\right)$

(ii) $\mathbb{E}\left[Z_{\mathcal{P}}(t) | Z_{\mathcal{P}}(t) > 0, \mathcal{E}_t\right] \leq a_t$

Lemma: $a_t = \Theta(1)$

The process is transient

Single Lévy flight analysis

- $\mathcal{P}$ target node

Let
- $Z_u(t) =$ random variable of number of visits in u until time t
- $\mathcal{E}_t =$ the event first t jumps have length $\leq (t \log t)^{\frac{1}{\alpha-1}}$
- $a_t = \mathbb{E}\left[Z_{(0,0)}(t) \mid \mathcal{E}_t\right]$
- $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 \mid \mathcal{E}_t\right)$

We look for this expectation

Lemma: $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 \mid \mathcal{E}_t\right) \geq \mathbb{E}\left[Z_{\mathcal{P}}(t) \mid \mathcal{E}_t\right] / a_t$

Comes from two facts

(i) $\mathbb{E}\left[Z_{\mathcal{P}}(t) | \mathcal{E}_t\right] = \mathbb{E}\left[Z_{\mathcal{P}}(t) | Z_{\mathcal{P}}(t) > 0, \mathcal{E}_t\right] \cdot \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 | \mathcal{E}_t\right)$

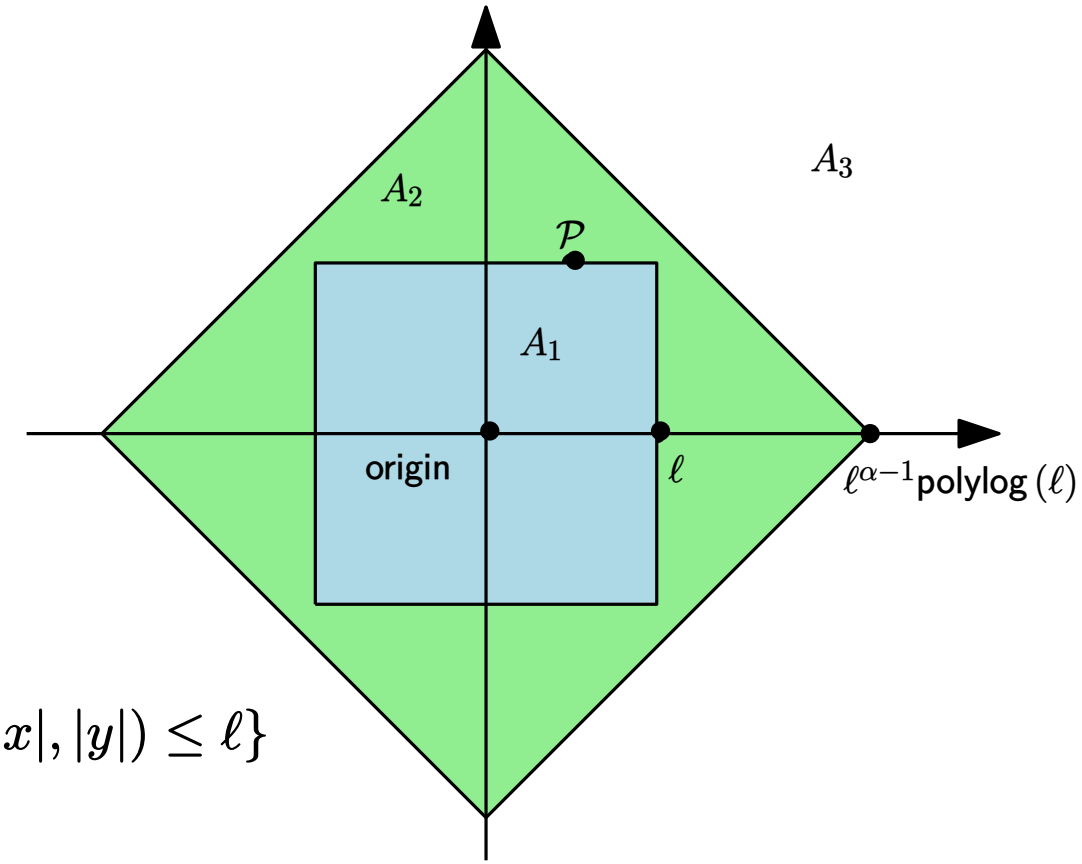
(ii) $\mathbb{E}\left[Z_{\mathcal{P}}(t) | Z_{\mathcal{P}}(t) > 0, \mathcal{E}_t\right] \leq a_t$

Lemma: $a_t = \Theta(1)$

The process is transient

Partition of the space

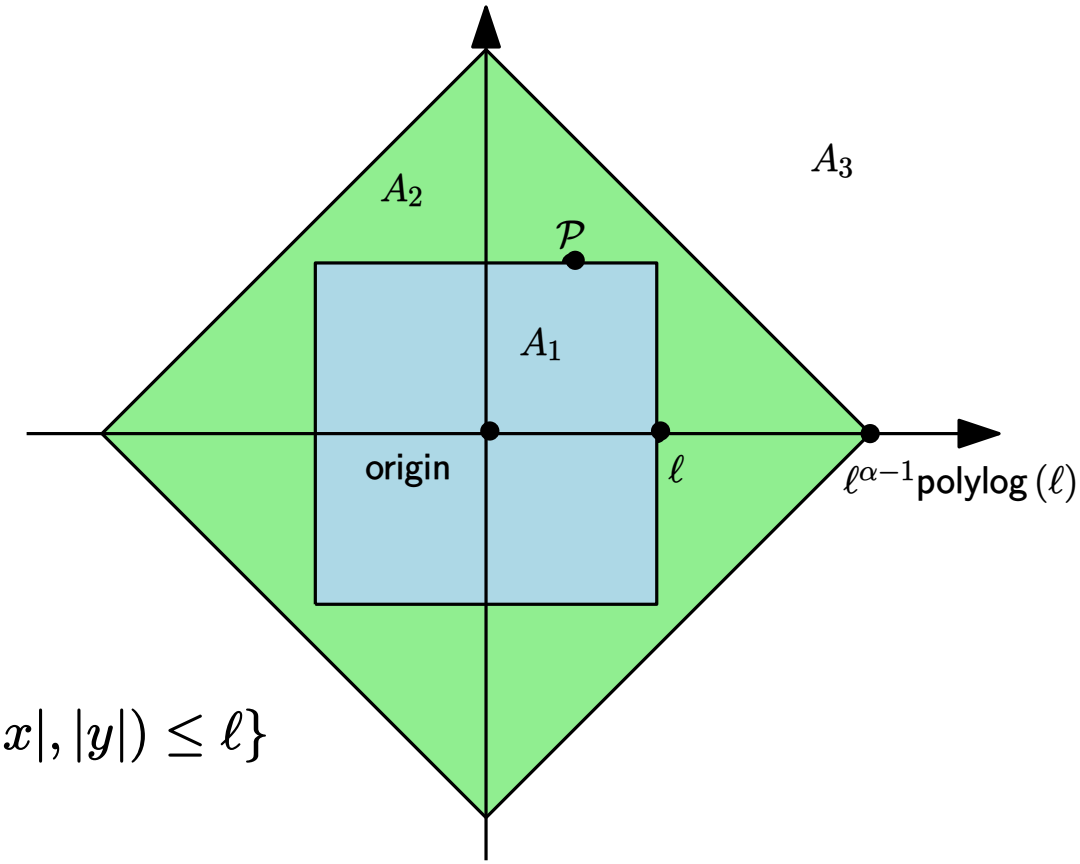
We partition $\mathbb{Z}^2$ in three areas in the following way



- $A_1 = Q(\ell) = \{(x, y) \in \mathbb{Z}^2 : \max(|x|, |y|) \leq \ell\}$
- $A_2 = B_{\ell^{\alpha-1}\text{polylog}(\ell)}((0, 0)) \setminus A_1$
- $A_3 = \mathbb{Z}^2 \setminus (A_1 \cup A_2)$

Partition of the space

We partition $\mathbb{Z}^2$ in three areas in the following way



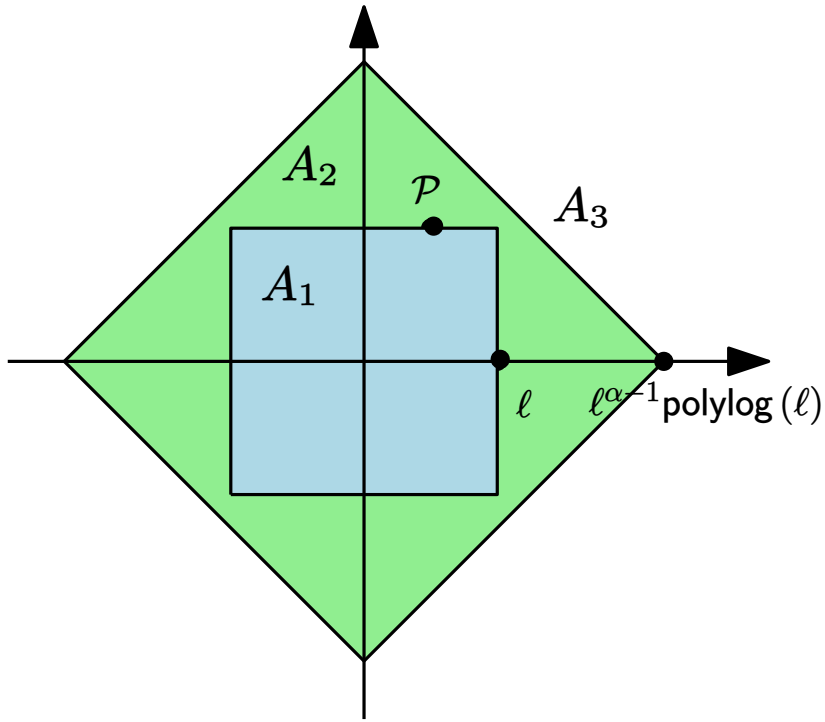
- $A_1 = Q(\ell) = \{(x, y) \in \mathbb{Z}^2 : \max(|x|, |y|) \leq \ell\}$
- $A_2 = B_{\ell^{\alpha-1}\text{polylog}(\ell)}((0,0)) \setminus A_1$
- $A_3 = \mathbb{Z}^2 \setminus (A_1 \cup A_2)$

Why: roughly $\ell^{\alpha-1}$ jumps to first jump of length $\geq \ell$

- Afterwards, the walk is lost

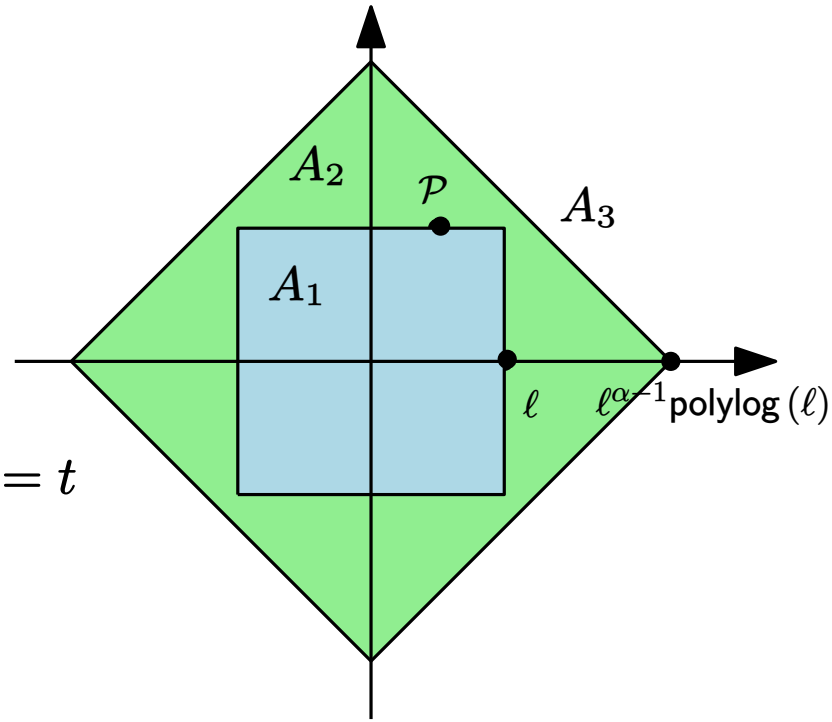
Getting $\mathbb{E}\left[Z_{\mathcal{P}}\left(t\right)|\mathcal{E}_t\right]$

$Z_S\left(t\right)$: total number of visits in the set S until time t



Getting $\mathbb{E}\left[Z_{\mathcal{P}}\left(t\right)|\mathcal{E}_t\right]$

$Z_S\left(t\right)$: total number of visits in the set S until time t

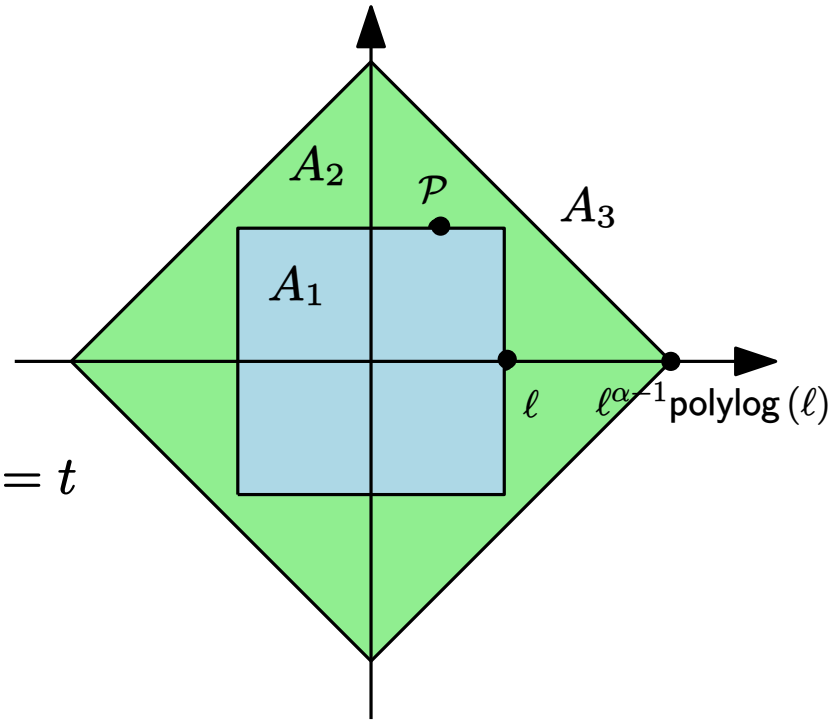


As $Z_{\mathbb{Z}^2}\left(t\right) = t$, then

a) $\mathbb{E}\left[Z_{A_1}\left(t\right) \mid \mathcal{E}_t\right] + \mathbb{E}\left[Z_{A_2}\left(t\right) \mid \mathcal{E}_t\right] + \mathbb{E}\left[Z_{A_3}\left(t\right) \mid \mathcal{E}_t\right] = t$

Getting $\mathbb{E}\left[Z_{\mathcal{P}}\left(t\right)|\mathcal{E}_t\right]$

$Z_S\left(t\right)$: total number of visits in the set S until time t



As $Z_{\mathbb{Z}^2}\left(t\right)=t$, then

a) $\mathbb{E}\left[Z_{A_1}\left(t\right)\mid\mathcal{E}_t\right]+\mathbb{E}\left[Z_{A_2}\left(t\right)\mid\mathcal{E}_t\right]+\mathbb{E}\left[Z_{A_3}\left(t\right)\mid\mathcal{E}_t\right]=t$

For some $t=\Theta\left(\ell^{\alpha-1}\right)$, we prove that:

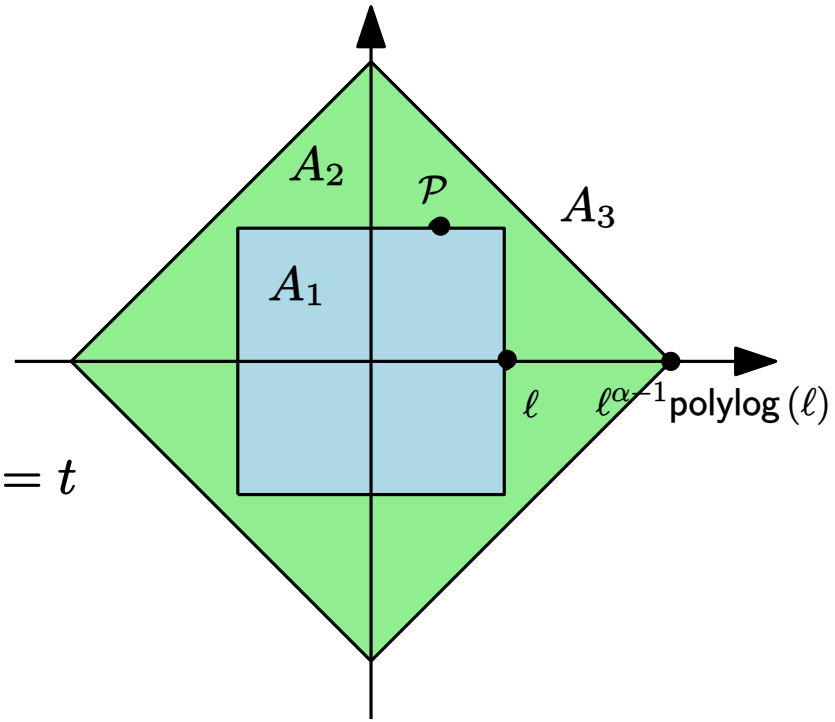
b) $\mathbb{E}\left[Z_{A_1}\left(t\right)\mid\mathcal{E}_t\right]\leq\frac{3}{4}t$

c) $\mathbb{E}\left[Z_{A_2}\left(t\right)\mid\mathcal{E}_t\right]\leq\mathbb{E}\left[Z_{\mathcal{P}}\left(t\right)\mid\mathcal{E}_t\right]\cdot\overbrace{\ell^2\text{polylog}\left(\ell\right)}^{|A_2|}$ ← Monotonicity property

d) $\mathbb{E}\left[Z_{A_3}\left(t\right)\mid\mathcal{E}_t\right]=\mathcal{O}\left(t/\log t\right)$

Getting $\mathbb{E}\left[Z_{\mathcal{P}}\left(t\right)|\mathcal{E}_t\right]$

$Z_S\left(t\right)$: total number of visits in the set S until time t



As $Z_{\mathbb{Z}^2}\left(t\right)=t$, then

a) $\mathbb{E}\left[Z_{A_1}\left(t\right)\mid\mathcal{E}_t\right]+\mathbb{E}\left[Z_{A_2}\left(t\right)\mid\mathcal{E}_t\right]+\mathbb{E}\left[Z_{A_3}\left(t\right)\mid\mathcal{E}_t\right]=t$

For some $t=\Theta\left(\ell^{\alpha-1}\right)$, we prove that:

b) $\mathbb{E}\left[Z_{A_1}\left(t\right)\mid\mathcal{E}_t\right]\leq\frac{3}{4}t$

c) $\mathbb{E}\left[Z_{A_2}\left(t\right)\mid\mathcal{E}_t\right]\leq\mathbb{E}\left[Z_{\mathcal{P}}\left(t\right)\mid\mathcal{E}_t\right]\cdot\overbrace{\ell^2\text{polylog}\left(\ell\right)}^{|A_2|}$ ← Monotonicity property

d) $\mathbb{E}\left[Z_{A_3}\left(t\right)\mid\mathcal{E}_t\right]=\mathcal{O}\left(t/\log t\right)$

Combine (a) with (b), (c), and (d) to get

$$\mathbb{E}\left[Z_{\mathcal{P}}\left(t\right)\mid\mathcal{E}_t\right]\geq 1/\left(\ell^{3-\alpha}\text{polylog}\left(\ell\right)\right)$$

Getting $p(t)$ and $\alpha^\star$

Reminder: $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}\left(t\right) > 0 \mid \mathcal{E}_t\right) \geq \mathbb{E}\left[Z_{\mathcal{P}}\left(t\right) \mid \mathcal{E}_t\right] / a_t$

Getting $p(t)$ and $\alpha^\star$

Reminder: $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}\left(t\right) > 0 \mid \mathcal{E}_t\right) \geq \mathbb{E}\left[Z_{\mathcal{P}}\left(t\right) \mid \mathcal{E}_t\right] / a_t$

Lemma: for $t = \Theta\left(\ell^{\alpha-1}\right)$, it holds that

$$p(t) \geq 1/\left(\ell^{3-\alpha}\text{polylog}\left(\ell\right)\right)$$

Note: the coupling result gives us the same asymptotic bound for the Lévy walk

Getting $p(t)$ and $\alpha^\star$

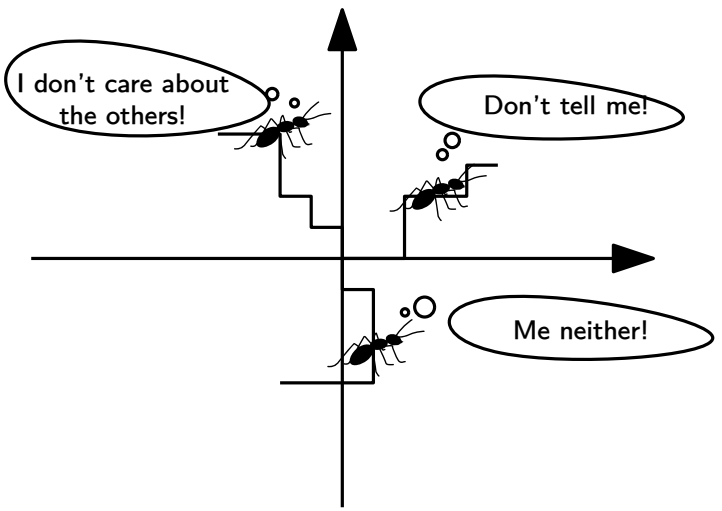
Reminder: $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}\left(t\right) > 0 \mid \mathcal{E}_t\right) \geq \mathbb{E}\left[Z_{\mathcal{P}}\left(t\right) \mid \mathcal{E}_t\right] / a_t$

Lemma: for $t = \Theta\left(\ell^{\alpha-1}\right)$, it holds that

$$p(t) \geq 1/\left(\ell^{3-\alpha}\text{polylog}\left(\ell\right)\right)$$

Note: the coupling result gives us the same asymptotic bound for the Lévy walk

Independence: $k = \ell^{3-\alpha}\text{polylog}\left(\ell\right)$ agents to have high probability



Getting $p(t)$ and $\alpha^\star$

Reminder: $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}(t) > 0 \mid \mathcal{E}_t\right) \geq \mathbb{E}\left[Z_{\mathcal{P}}(t) \mid \mathcal{E}_t\right] / a_t$

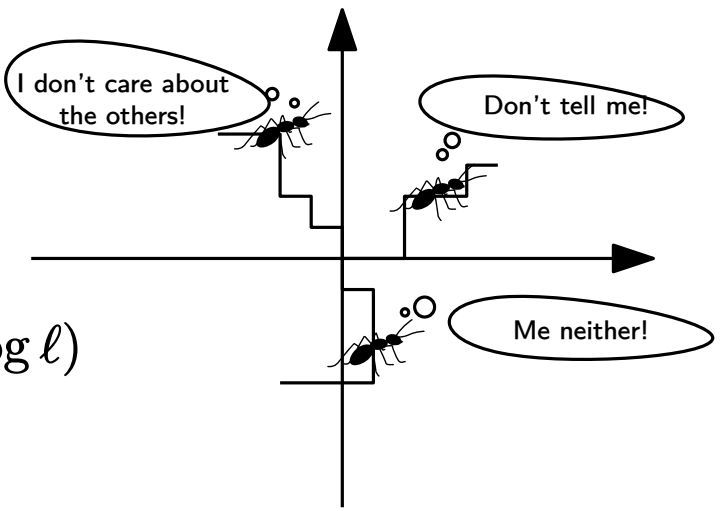
Lemma: for $t = \Theta\left(\ell^{\alpha-1}\right)$, it holds that

$$p(t) \geq 1/\left(\ell^{3-\alpha}\text{polylog}\left(\ell\right)\right)$$

Note: the coupling result gives us the same asymptotic bound for the Lévy walk

Independence: $k = \ell^{3-\alpha}\text{polylog}\left(\ell\right)$ agents to have high probability

$$\alpha = 3 - \log k / \log \ell + \mathcal{O}\left(\log\log \ell / \log \ell\right)$$



Getting $p(t)$ and $\alpha^\star$

Reminder: $p(t) = \mathbb{P}\left(Z_{\mathcal{P}}\left(t\right) > 0 \mid \mathcal{E}_t\right) \geq \mathbb{E}\left[Z_{\mathcal{P}}\left(t\right) \mid \mathcal{E}_t\right] / a_t$

Lemma: for $t = \Theta\left(\ell^{\alpha-1}\right)$, it holds that

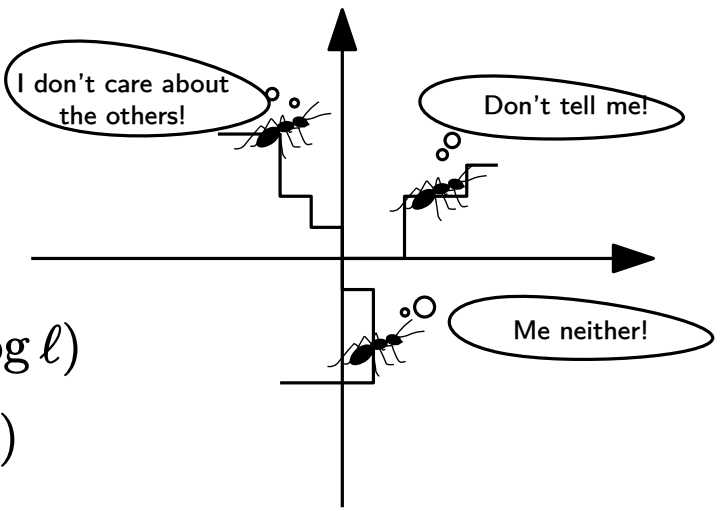
$$p(t) \geq 1/\left(\ell^{3-\alpha}\text{polylog}\left(\ell\right)\right)$$

Note: the coupling result gives us the same asymptotic bound for the Lévy walk

Independence: $k = \ell^{3-\alpha}\text{polylog}\left(\ell\right)$ agents to have high probability

$$\alpha = 3 - \log k / \log \ell + \mathcal{O}\left(\log\log \ell / \log \ell\right)$$

$$t = \mathcal{O}\left(\ell^{\alpha-1}\right) \leq \left(\ell^2/k + \ell\right)\text{polylog}\left(\ell\right)$$



Our contributions

(i) DEFINITION OF DISCRETE LÉVY WALK

(ii) ANALYSIS OF THE PARALLEL HITTING TIME

(iii) ALGORITHM FOR THE ANTS PROBLEM

But... No advice, no communication!

How can agents find $\alpha^\star$?

But... No advice, no communication!

How can agents find $\alpha^\star$?

They don't have to!

Algorithm: *each agent u samples u.a.r. a real number $\alpha_u \in (2,3)$. Then, it performs a discrete Lévy walk with exponent α_u*

But... No advice, no communication!

How can agents find $\alpha^\star$?

They don't have to!

Algorithm: *each agent u samples u.a.r. a real number $\alpha_u \in (2,3)$. Then, it performs a discrete Lévy walk with exponent α_u*

If $\ell \leq \exp\left(k^{\Theta(1)}\right)$, the hitting time is $\mathcal{O}\left(\left(\ell^2/k + \ell\right) \text{polylog}\left(\ell\right)\right)$ w.h.p.

But... No advice, no communication!

How can agents find $\alpha^\star$?

They don't have to!

Algorithm: *each agent u samples u.a.r. a real number $\alpha_u \in (2,3)$. Then, it performs a discrete Lévy walk with exponent α_u*

If $\ell \leq \exp\left(k^{\Theta(1)}\right)$, the hitting time is $\mathcal{O}\left(\left(\ell^2/k + \ell\right) \text{polylog}\left(\ell\right)\right)$ w.h.p.

Proof outline:

Fix some $\epsilon = \mathcal{O}\left(\log\log\ell / \log\ell\right)$

We use: $\ell < \exp\left(k^{\Theta(1)}\right)$ ($\iff$ $k \geq \text{polylog}\left(\ell\right)$) + Chernoff bound

$\implies$ at least $\Theta\left(\epsilon k\right)$ agents sample an exponent in the range $(\alpha^\star - \epsilon, \alpha^\star + \epsilon)$ w.h.p.

$\Theta\left(\epsilon k\right)$ agents are sufficient to ensure high probability to find the target fast enough

Recap

We

- provide a definition of a discrete version of the Lévy walk
- analyze the hitting time of k parallel Lévy walks
- show that for any choices of k and ℓ from a wide range, Lévy walks are an almost-optimal search strategy for the ANTS problem

Recap

We

- provide a definition of a discrete version of the Lévy walk
- analyze the hitting time of k parallel Lévy walks
- show that for any choices of k and ℓ from a wide range, Lévy walks are an almost-optimal search strategy for the ANTS problem

- mathematically corroborate the Lévy flight foraging hypothesis
- argue the non (universal) optimality of exponent $\alpha = 2$

Table of contents

The consensus problem

Input: system of n agents supporting opinions, with a communication network

Task: designing a protocol which **brings the system** in finite time to a configuration such that

1. all agents support the same opinion (AGREEMENT)
2. the final opinion is among the initial ones (VALIDITY)
3. the agreement keeps on unless external events occur (STABILITY)

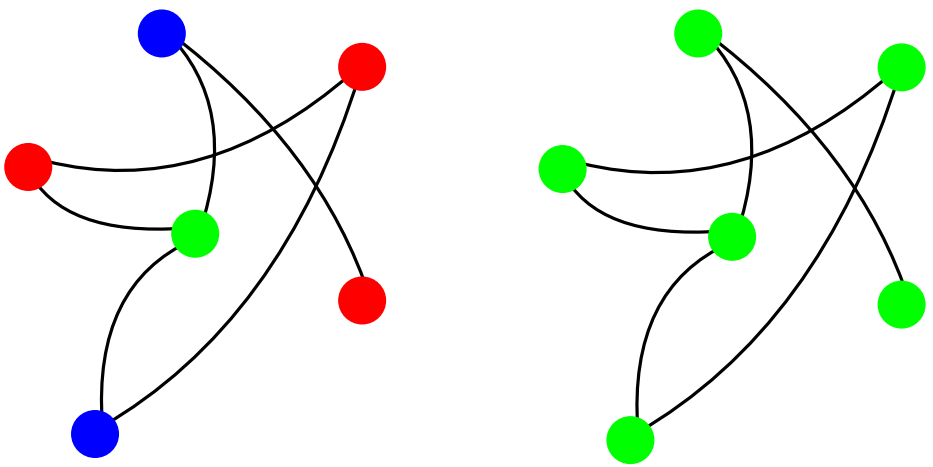
The consensus problem

Input: system of n agents supporting opinions, with a communication network

Task: designing a protocol which **brings the system** in finite time to a configuration such that

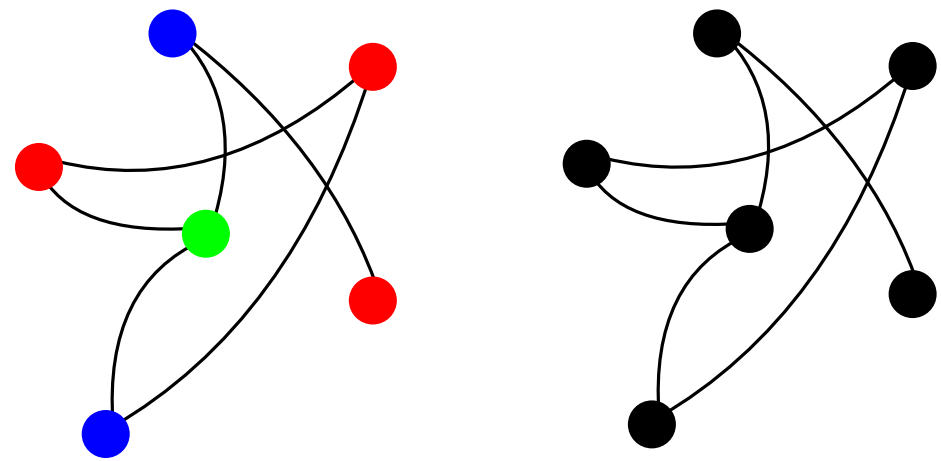
1. all agents support the same opinion (AGREEMENT)
2. the final opinion is among the initial ones (VALIDITY)
3. the agreement keeps on unless external events occur (STABILITY)

time $t = 0$ ⟶ time $t > 0$



valid consensus

time $t = 0$ ⟶ time $t > 0$



non valid consensus

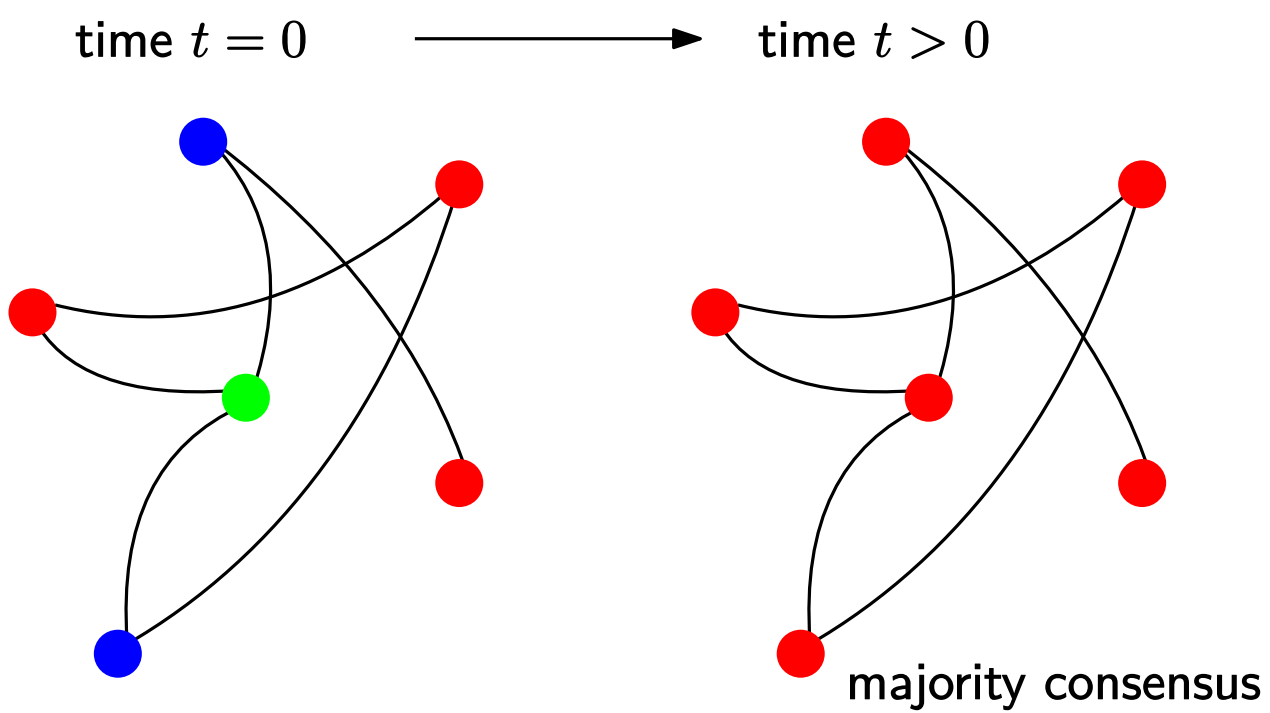
The majority consensus problem

1. AGREEMENT
2. ~~VALIDITY~~
3. STABILITY

→

1. AGREEMENT
2. **MAJORITY**
3. STABILITY

2. MAJORITY property: the final opinion is the **initial majority** one



Opinion dynamics

Opinion dynamics: class of simple, lightweight parallel protocols for distributed tasks (e.g., consensus, clustering, etc.)

Many have been investigated, including:

- Voter Model [Hassin and Peleg, Inf. Comput. 2001]
- Averaging dynamics [Becchetti et al., SODA 2017]

} linear dynamics

- 3-Majority [Becchetti et al., SODA 2016]
- 2-Choices [Berenbrink et al., PODC 2017]
- Undecided-State [Becchetti et al., SODA 2015]

} non-linear dynamics

Opinion dynamics

Opinion dynamics: class of simple, lightweight parallel protocols for distributed tasks (e.g., consensus, clustering, etc.)

Many have been investigated, including:

- Voter Model [Hassin and Peleg, Inf. Comput. 2001]
- Averaging dynamics [Becchetti et al., SODA 2017]

} linear dynamics

- 3-Majority [Becchetti et al., SODA 2016]
- 2-Choices [Berenbrink et al., PODC 2017]
- Undecided-State [Becchetti et al., SODA 2015]

} non-linear dynamics

Majority update-rules and the undecided state dynamics have biological inspirations [Reina et al., Physical Review 2017] [Condon et al., Nat. Computing 2020] [Chaouiya et al., PLOS ONE 2013]

Opinion dynamics

Opinion dynamics: class of simple, lightweight parallel protocols for distributed tasks (e.g., consensus, clustering, etc.)

Many have been investigated, including:

- Voter Model [Hassin and Peleg, Inf. Comput. 2001]
- Averaging dynamics [Becchetti et al., SODA 2017]

} linear dynamics

- 3-Majority [Becchetti et al., SODA 2016]
- 2-Choices [Berenbrink et al., PODC 2017]
- Undecided-State [Becchetti et al., SODA 2015]

} non-linear dynamics

Majority update-rules and the undecided state dynamics have biological inspirations [Reina et al., Physical Review 2017] [Condon et al., Nat. Computing 2020] [Chaouiya et al., PLOS ONE 2013]

Often, settings with adversarial Byzantine failures are investigated

Not realistic in biological scenarios; rather, uniform noise [Feinerman et al., PODC 2014]

Uniform communication noise

Inspired by [Feinerman et al., PODC 2014], [Fraigniaud and Natale, PODC 2016]

Σ set of k opinions, $p \in [0, 1]$ constant

When u looks at v's opinion x

a) with probability $1 - p$, u sees x
b) with probability p, u sees y where y is chosen u.a.r. in Σ

Uniform communication noise

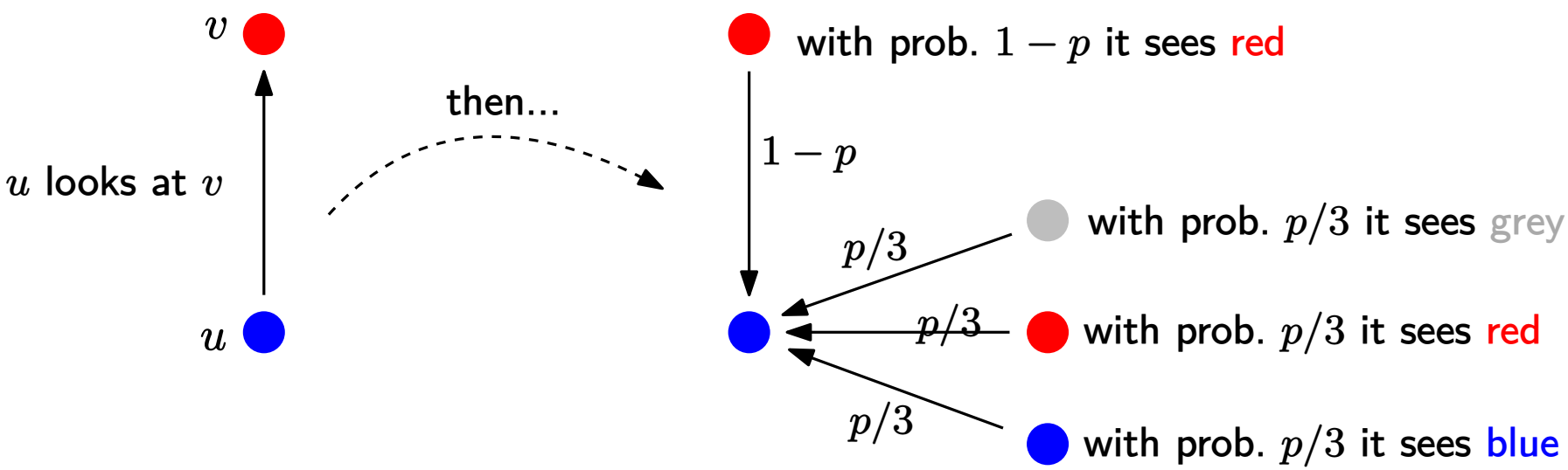
Inspired by [Feinerman et al., PODC 2014], [Fraigniaud and Natale, PODC 2016]

Σ set of k opinions, $p \in [0, 1]$ constant

When u looks at v's opinion x

a) with probability $1 - p$, u sees x
b) with probability p, u sees y where y is chosen u.a.r. in Σ

Example: $k = 3$



3-Majority and Undecided-State

3-Majority dynamics: each node u

1. samples 3 neighbors u.a.r.
2. pulls their opinions
3. updates its opinion to the majority one, if any

3-Majority and Undecided-State

3-Majority dynamics: each node u

1. samples 3 neighbors u.a.r.
2. pulls their opinions
3. updates its opinion to the majority one, if any

Undecided-State dynamics: each node u

1. samples a neighbor v u.a.r.
2. pulls v's opinion
3. updates its opinion according to the following **table**

$u \backslash v$	opinion i	opinion j	undecided
opinion i	i	undecided	i
opinion j	undecided	j	j
undecided	i	j	undecided

Our contribution

Based on [d'Amore et al., SIROCCO 2020], [d'Amore et Ziccardi, SIROCCO 2022], [d'Amore et al., Swarm Intelligence 2022]

We study the **Undecided-State** dynamics and the **3-Majority** dynamics with $k = 2$ opinions in the presence of uniform noise in the complete graph

Our contribution

Based on [d'Amore et al., SIROCCO 2020], [d'Amore et Ziccardi, SIROCCO 2022], [d'Amore et al., Swarm Intelligence 2022]

We study the **Undecided-State** dynamics and the **3-Majority** dynamics with $k = 2$ opinions in the presence of uniform noise in the complete graph

For the **3-Majority** dynamics: phase-transition

$p < 1/3$:
- a value $\bar{s} = \Theta(n)$ exists such that the bias of the system reaches the interval $I_\varepsilon = [(1-\varepsilon)\bar{s}, (1+\varepsilon)\bar{s}]$ in time $\mathcal{O}(\log n)$ w.h.p., and keeps in I_ε for time poly(n) w.h.p. $\longrightarrow$ **almost-consensus**
- if the initial bias is $\Omega(\sqrt{n \log n})$, we have almost-majority consensus

$p > 1/3$:
- in time $\mathcal{O}(\log n)$ the bias becomes bounded by $\mathcal{O}(\sqrt{n \log n})$ and keeps bounded for time poly(n) wh.p. $\longrightarrow$ **victory of noise**
- there is constant probability to switch majority within time $\mathcal{O}(\log n)$

Our contribution

Based on [d'Amore et al., SIROCCO 2020], [d'Amore et Ziccardi, SIROCCO 2022], [d'Amore et al., Swarm Intelligence 2022]

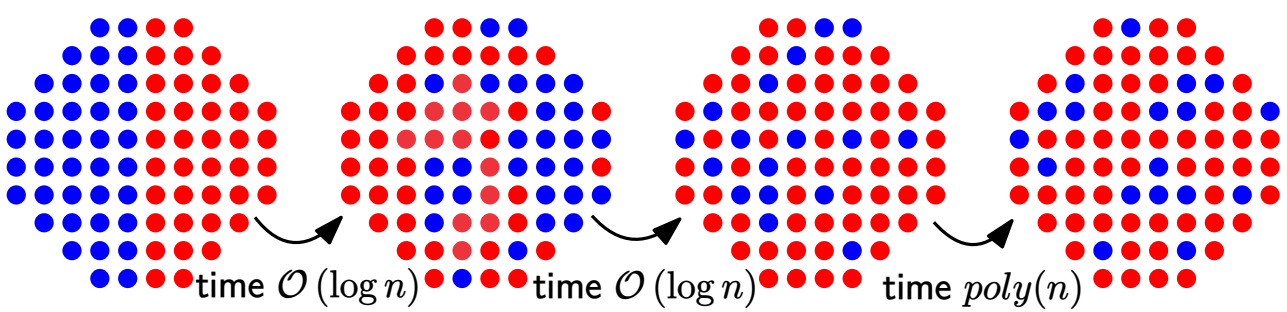
We study the **Undecided-State** dynamics and the **3-Majority** dynamics with $k = 2$ opinions in the presence of uniform noise in the complete graph

For the **3-Majority** dynamics: phase-transition

$p < 1/3$:
- a value $\bar{s} = \Theta(n)$ exists such that the bias of the system reaches the interval $I_\varepsilon = [(1-\varepsilon)\bar{s}, (1+\varepsilon)\bar{s}]$ in time $\mathcal{O}(\log n)$ w.h.p., and keeps in I_ε for time poly(n) w.h.p. ⟶ **almost-consensus**
- if the initial bias is $\Omega(\sqrt{n \log n})$, we have almost-majority consensus

$p > 1/3$:
- in time $\mathcal{O}(\log n)$ the bias becomes bounded by $\mathcal{O}(\sqrt{n \log n})$ and keeps bounded for time poly(n) wh.p. ⟶ **victory of noise**
- there is constant probability to switch majority within time $\mathcal{O}(\log n)$

Example: consensus



Our contribution

Based on [d'Amore et al., SIROCCO 2020], [d'Amore et Ziccardi, SIROCCO 2022], [d'Amore et al., Swarm Intelligence 2022]

We study the **Undecided-State** dynamics and the **3-Majority** dynamics with $k = 2$ opinions in the presence of uniform noise in the complete graph

For the **3-Majority** dynamics: phase-transition

$p < 1/3$:
- a value $\bar{s} = \Theta(n)$ exists such that the bias of the system reaches the interval $I_\varepsilon = [(1-\varepsilon)\bar{s}, (1+\varepsilon)\bar{s}]$ in time $\mathcal{O}(\log n)$ w.h.p., and keeps in I_ε for time poly(n) w.h.p. $\longrightarrow$ **almost-consensus**
- if the initial bias is $\Omega(\sqrt{n \log n})$, we have almost-majority consensus

$p > 1/3$:
- in time $\mathcal{O}(\log n)$ the bias becomes bounded by $\mathcal{O}(\sqrt{n \log n})$ and keeps bounded for time poly(n) wh.p. $\longrightarrow$ **victory of noise**
- there is constant probability to switch majority within time $\mathcal{O}(\log n)$

For the **Undecided-State**: similar behavior

- phase-transition at $p = 1/2$
- less characterized, more complex

Recap

- **Undecided-State** dynamics and **3-Majority** dynamics are not implemented by biological systems
  - despite the bio-inspiration, highly abstract model
  - aiming to capture fundamental phenomena that (**very loosely**) relates to many biological systems

Recap

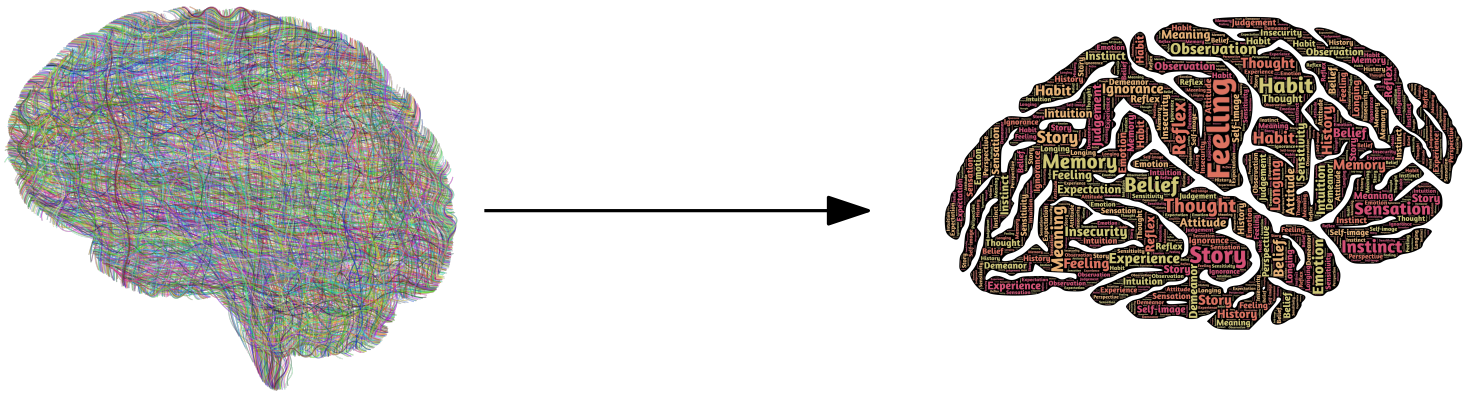
- **Undecided-State** dynamics and **3-Majority** dynamics are not implemented by biological systems
  - despite the bio-inspiration, highly abstract model
  - aiming to capture fundamental phenomena that (**very loosely**) relates to many biological systems

- **Undecided-State** dynamics more resilient to noise than **3-Majority** dynamics
  - phase-transitions: $p = 1/2$ vs $p = 1/3$
    - $p = 1/2$ (USD) means half of the communications are non-noisy on average
    - $p = 1/3$ (3-Maj) means 2 out of 3 pulled opinions are non-noisy on average

Table of contents

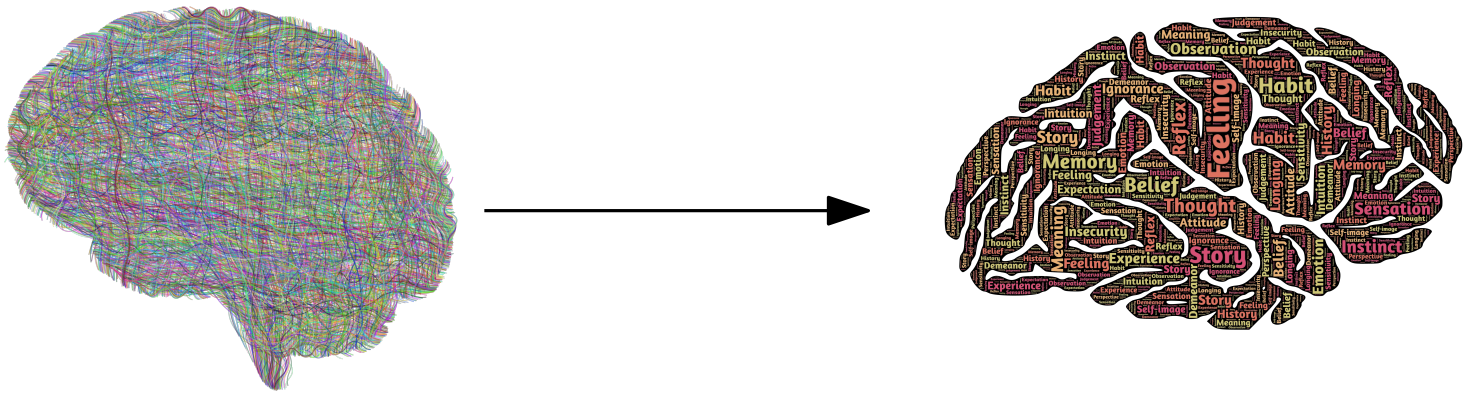
Brain & Mind

In the reductionist/materialistic approach, the mind is an emergent phenomenom of the brain (i.e., neurons and synapses)



Brain & Mind

In the reductionist/materialistic approach, the mind is an emergent phenomenom of the brain (i.e., neurons and synapses)

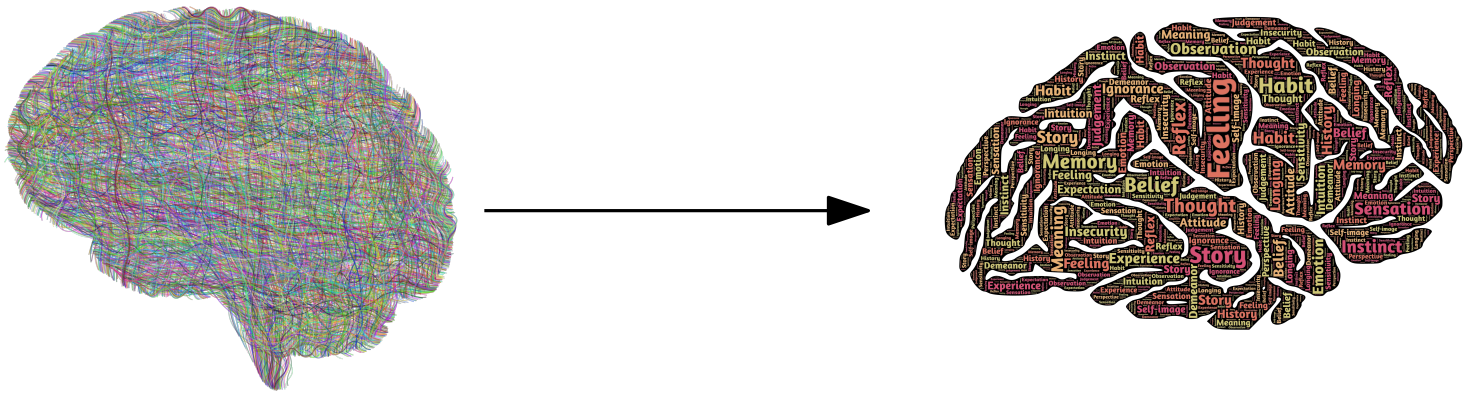


Nobel laureate Richard Axel: *We do not have a* **logic** *for the transformation of neural activity to thought and action. I consider discerning [this* **logic***] as the most important future direction in Neuroscience* [Axel, Neuron 2018]

Huge gap of scale and methodology between Experimental and Cognitive neuroscience

Brain & Mind

In the reductionist/materialistic approach, the mind is an emergent phenomenom of the brain (i.e., neurons and synapses)



Nobel laureate Richard Axel: *We do not have a* **logic** *for the transformation of neural activity to thought and action. I consider discerning [this* **logic***] as the most important future direction in Neuroscience* [Axel, Neuron 2018]

Huge gap of scale and methodology between Experimental and Cognitive neuroscience

Assembly Calculus: recently proposed formal computational system [Papadimitriou et al., PNAS 2020]

- explicit purpose of fitting Axel's logic
- bridging through computation the gap between neurons and intelligence

Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

Assemblies

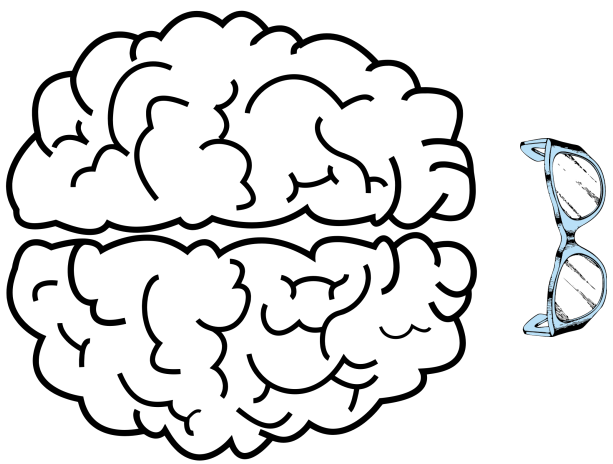
Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]

Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

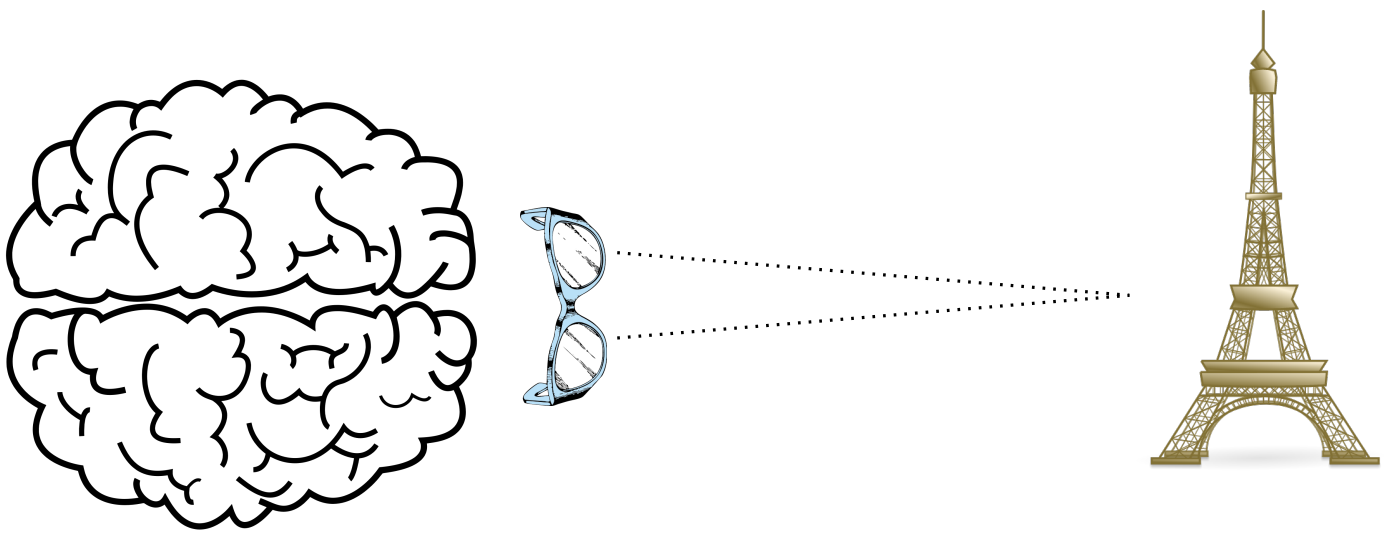
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

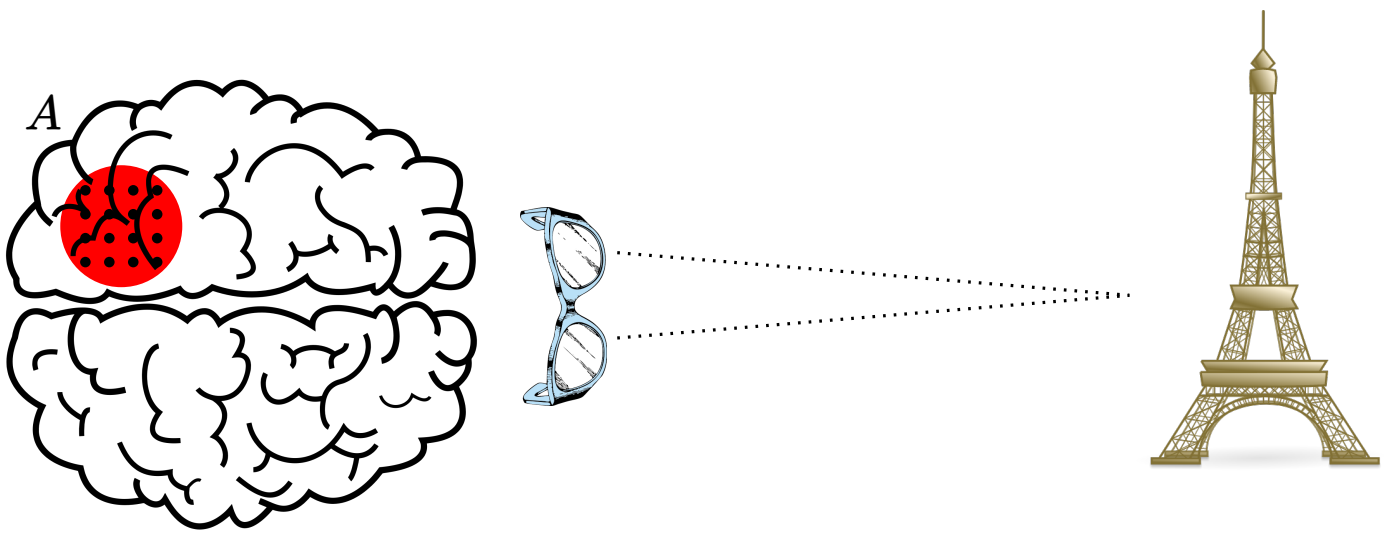
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

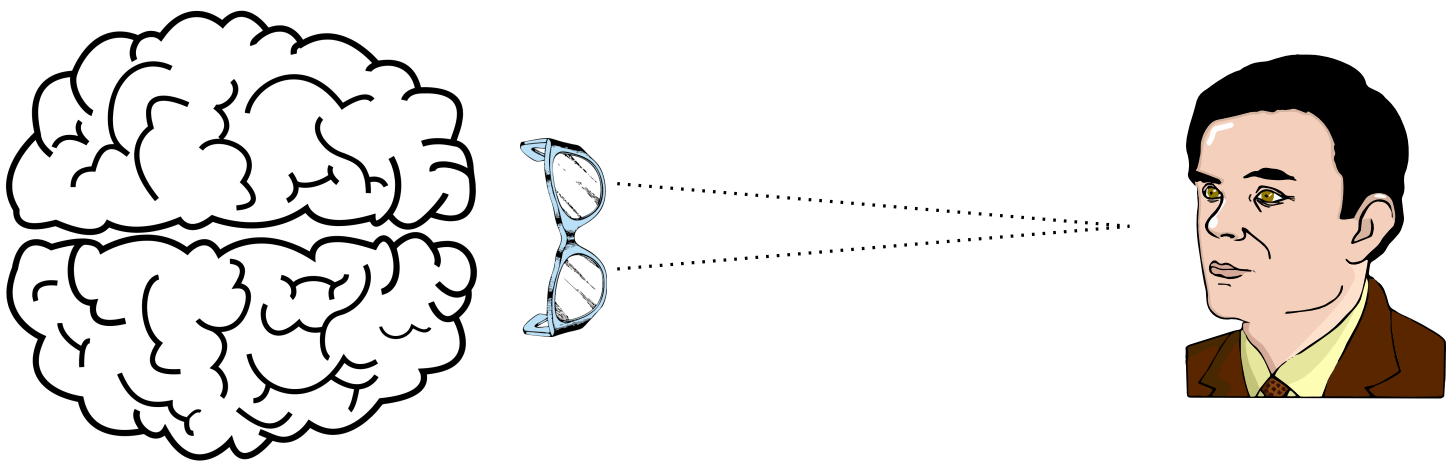
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

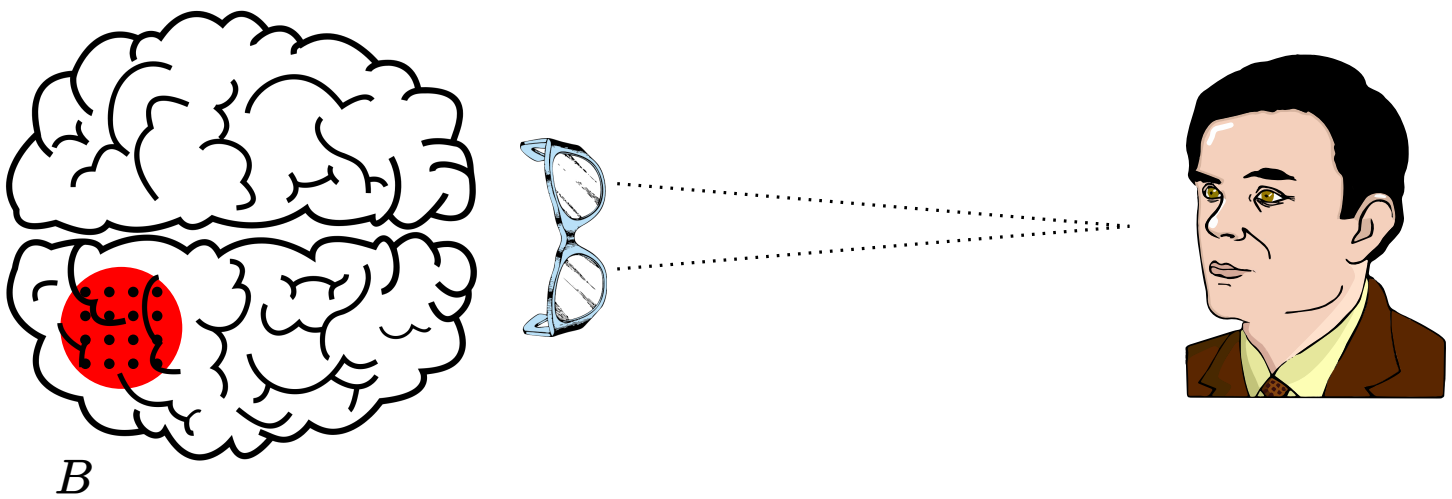
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

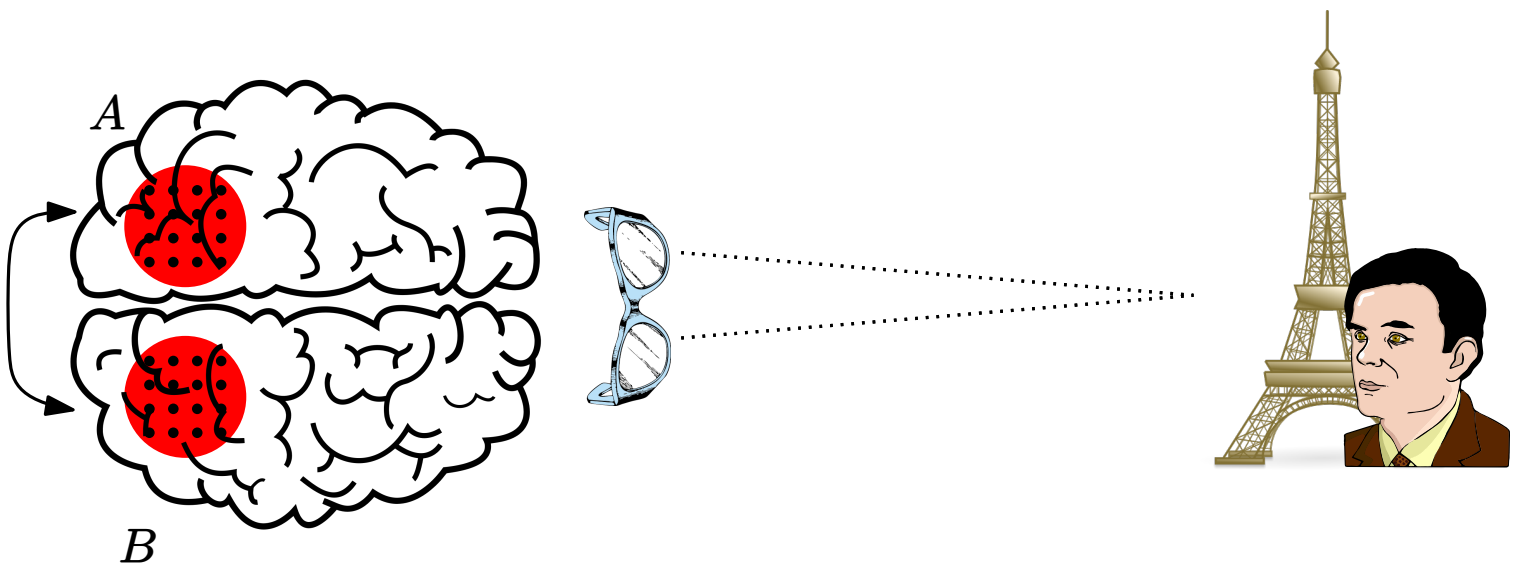
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Association: experiment by [Ison et al., Neuron 2015]

Contribution

We empirically **demonstrate** that reasonably large and complex programs run correctly and reliably in the assembly calculus by implementing planning strategies in the blocks world (**Julia** programming language) [d'Amore et al., AAAI 2022]

Possible moves:
- move a block from the top of a stack to the table
- move a block from the top of a stack to the top of another stack
- move a block from the table to the top of a stack

Input:
- set of initial stacks
- set of goal stacks

Output:
- the sequence of moves

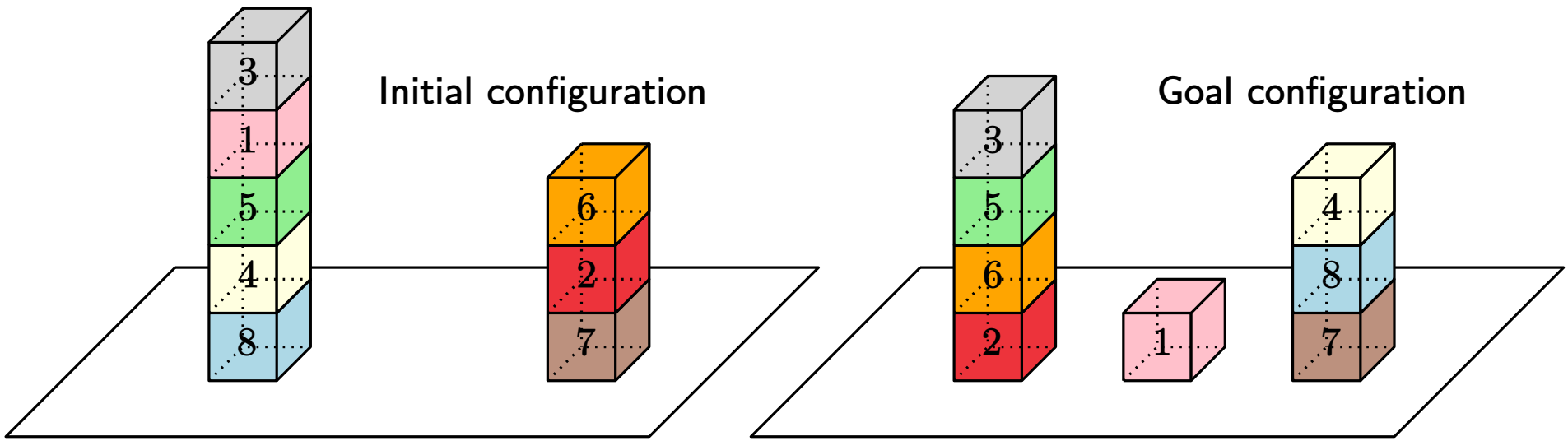


Table of contents

Conclusions & perspectives

Lévy walks	Opinion dynamics	Assembly Calculus
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].		

Conclusions & perspectives

Lévy walks	Opinion dynamics	Assembly Calculus
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].		
Generalization to higher dimension. Possible follow-up with experiments in biology.		

Conclusions & perspectives

Lévy walks	Opinion dynamics	Assembly Calculus
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].		
Generalization to higher dimension. Possible follow-up with experiments in biology.		
Wider spectrum: broadcasting, epidemic proccesses, space-covering. [Berenbrink et al., SODA 2011; Elsässer et Sauerwald, TCS 2011].		

Conclusions & perspectives

Lévy walks	Opinion dynamics	Assembly Calculus
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].	Contribution to the study of noisy communications in the consensus problem. Hint at general behavior of non-linear OD.	
Generalization to higher dimension. Possible follow-up with experiments in biology.		
Wider spectrum: broadcasting, epidemic proccesses, space-covering. [Berenbrink et al., SODA 2011; Elsässer et Sauerwald, TCS 2011].		

Conclusions & perspectives

Lévy walks	**Opinion dynamics**	**Assembly Calculus**
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].	Contribution to the study of noisy communications in the consensus problem. Hint at general behavior of non-linear OD.	
Generalization to higher dimension. Possible follow-up with experiments in biology.	Higher comprehension of information exchange processes in biological system: **asymmetric** noise? More opinions?	
Wider spectrum: broadcasting, epidemic proccesses, space-covering. [Berenbrink et al., SODA 2011; Elsässer et Sauerwald, TCS 2011].		

Conclusions & perspectives

Lévy walks	Opinion dynamics	Assembly Calculus
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].	Contribution to the study of noisy communications in the consensus problem. Hint at general behavior of non-linear OD.	
Generalization to higher dimension. Possible follow-up with experiments in biology.	Higher comprehension of information exchange processes in biological system: **asymmetric** noise? More opinions?	
Wider spectrum: broadcasting, epidemic proccesses, space-covering. [Berenbrink et al., SODA 2011; Elsässer et Sauerwald, TCS 2011].	Distributed broadcast or consensus in dynamic environments: random walks via averaging, voter, etc. [Berenbrink et al, ICALP 2016; Giakkoupis et al., PODC 2019]	

Conclusions & perspectives

Lévy walks	Opinion dynamics	Assembly Calculus
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].	Contribution to the study of noisy communications in the consensus problem. Hint at general behavior of non-linear OD.	Experimental investigation of the AC framework, establishing some computational limits
Generalization to higher dimension. Possible follow-up with experiments in biology.	Higher comprehension of information exchange processes in biological system: **asymmetric** noise? More opinions?	
Wider spectrum: broadcasting, epidemic proccesses, space-covering. [Berenbrink et al., SODA 2011; Elsässer et Sauerwald, TCS 2011].	Distributed broadcast or consensus in dynamic environments: random walks via averaging, voter, etc. [Berenbrink et al, ICALP 2016; Giakkoupis et al., PODC 2019]	

Conclusions & perspectives

Lévy walks	Opinion dynamics	Assembly Calculus
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].	Contribution to the study of noisy communications in the consensus problem. Hint at general behavior of non-linear OD.	Experimental investigation of the AC framework, establishing some computational limits
Generalization to higher dimension. Possible follow-up with experiments in biology.	Higher comprehension of information exchange processes in biological system: **asymmetric** noise? More opinions?	How can other cognitive functions be carried out? How to provide more robustness at the microscopic level?
Wider spectrum: broadcasting, epidemic proccesses, space-covering. [Berenbrink et al., SODA 2011; Elsässer et Sauerwald, TCS 2011].	Distributed broadcast or consensus in dynamic environments: random walks via averaging, voter, etc. [Berenbrink et al, ICALP 2016; Giakkoupis et al., PODC 2019]	

Conclusions & perspectives

Lévy walks	Opinion dynamics	Assembly Calculus
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].	Contribution to the study of noisy communications in the consensus problem. Hint at general behavior of non-linear OD.	Experimental investigation of the AC framework, establishing some computational limits
Generalization to higher dimension. Possible follow-up with experiments in biology.	Higher comprehension of information exchange processes in biological system: **asymmetric** noise? More opinions?	How can other cognitive functions be carried out? How to provide more robustness at the microscopic level?
Wider spectrum: broadcasting, epidemic proccesses, space-covering. [Berenbrink et al., SODA 2011; Elsässer et Sauerwald, TCS 2011].	Distributed broadcast or consensus in dynamic environments: random walks via averaging, voter, etc. [Berenbrink et al, ICALP 2016; Giakkoupis et al., PODC 2019]	What are the algorithmic primitives in biological NN? Computational dynamics for k-Winners take all networks [Lynch et al., ITCS 2017].

Conclusions & perspectives

Lévy walks	**Opinion dynamics**	**Assembly Calculus**
Power law distributions extremely powerful [Kleinberg, STOC 2000; Fraigniaud et Giakkoupis, STOC 2010, DC 2014].	Contribution to the study of noisy communications in the consensus problem. Hint at general behavior of non-linear OD.	Experimental investigation of the AC framework, establishing some computational limits
Generalization to higher dimension. Possible follow-up with experiments in biology.	Higher comprehension of information exchange processes in biological system: **asymmetric** noise? More opinions?	How can other cognitive functions be carried out? How to provide more robustness at the microscopic level?
Wider spectrum: broadcasting, epidemic proccesses, space-covering. [Berenbrink et al., SODA 2011; Elsässer et Sauerwald, TCS 2011].	Distributed broadcast or consensus in dynamic environments: random walks via averaging, voter, etc. [Berenbrink et al, ICALP 2016; Giakkoupis et al., PODC 2019]	What are the algorithmic primitives in biological NN? Computational dynamics for k-Winners take all networks [Lynch et al., ITCS 2017].

The End (or not...?)

Post-doc position at **Aalto University**, Helsinki, working with prof. **Jukka Suomela**

The End (or not...?)

Post-doc position at **Aalto University**, Helsinki, working with prof. **Jukka Suomela**

Appendix

1. Lévy walks

2. Opinion dynamics

3. Assembly calculus

Defining the discrete Lévy walk

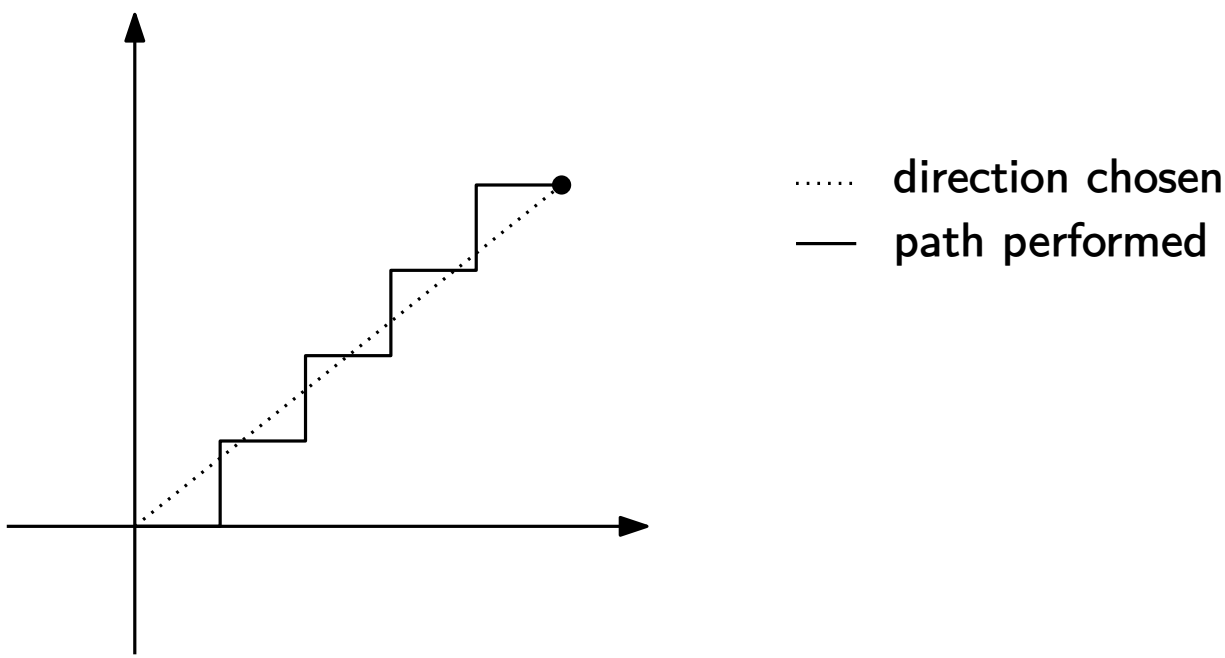
Two choices to make:

- define the jump-length distribution
- define a notion of **approximating** a line-segment

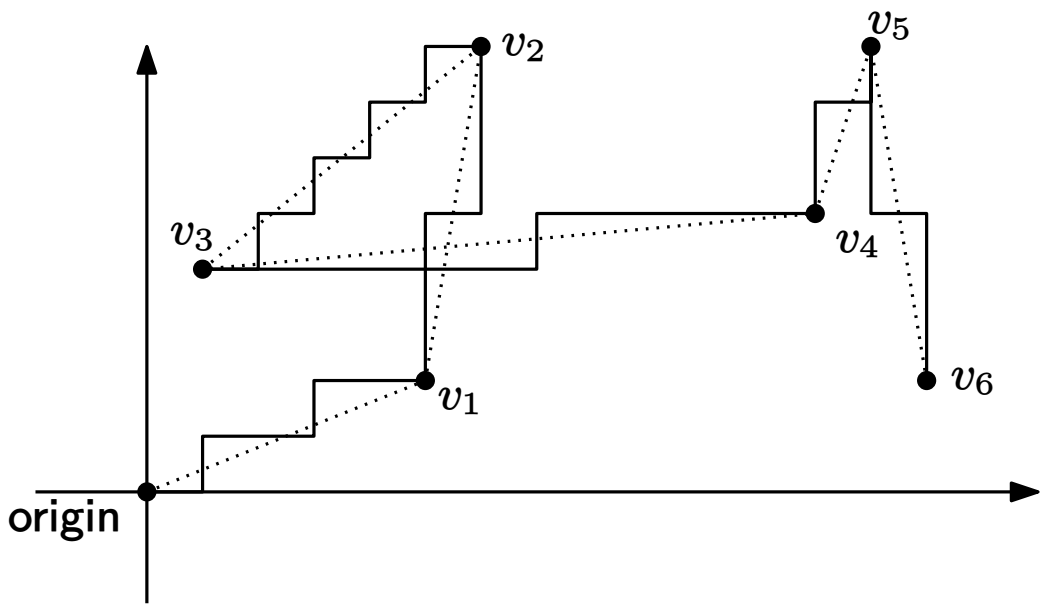
Jump length distribution

- $d = 0$ with probability $1/2$
- $d \geq 1$ with probability c_α / d^α

Approximation of a line-segment



Discrete Lévy walk



direction chosen
path performed

Six jumps of a discrete Lévy walk

Let $\alpha > 1$ be a real value

Lévy walk: the agent

a) chooses a distance $d \in \mathbb{N}$ as follows: $d = 0$ w.p. $1/2$, and $d \geq 1$ w.p. c_α / d^α
b) chooses a destination u.a.r. among those at distance d
c) walks along an approximating path for d steps, one edge at a time, crossing d nodes
d) repeats the procedure

No advice, no communication

[Feinerman et Korman, DC 2017] proposes many solutions to the problem

Many considered settings, in which

- agents exchange information at the source node
- agents receive some advice on the number of agents k
- there is no communication and no advice

No advice, no communication

[Feinerman et Korman, DC 2017] proposes many solutions to the problem

Many considered settings, in which

- agents exchange information at the source node
- agents receive some advice on the number of agents k
- there is no communication and no advice

We focus on the case **no advice, no communication**

No advice, no communication

[Feinerman et Korman, DC 2017] proposes many solutions to the problem

Many considered settings, in which

- agents exchange information at the source node
- agents receive some advice on the number of agents k
- there is no communication and no advice

We focus on the case **no advice, no communication**

Their best algorithm in this case achieves expected hitting time

$$\mathcal{O}\left(\left(\ell^2/k+\ell\right)\log^{1+\epsilon}\ell\right),$$

for any fixed constant $\epsilon > 0$

No advice, no communication

Uniform algorithm proposed in [Feinerman et Korman, DC 2017]

(idea)

i fix a subset of k_i agents to be moved
ii fix a ball of some radius ℓ_i
iii agents go to random nodes in the ball
iv agents perform a spiral search of length d_i around the chosen nodes
v agents return to the source node
vi increase k_i, ℓ_i, d_i and repeat (i)-(v)

However, the above algorithm is not that natural

No advice, no communication

Uniform algorithm proposed in [Feinerman et Korman, DC 2017]

(idea)

i fix a subset of k_i agents to be moved
ii fix a ball of some radius ℓ_i
iii agents go to random nodes in the ball
iv agents perform a spiral search of length d_i around the chosen nodes
v agents return to the source node
vi increase k_i, ℓ_i, d_i and repeat (i)-(v)

However, the above algorithm is not that natural

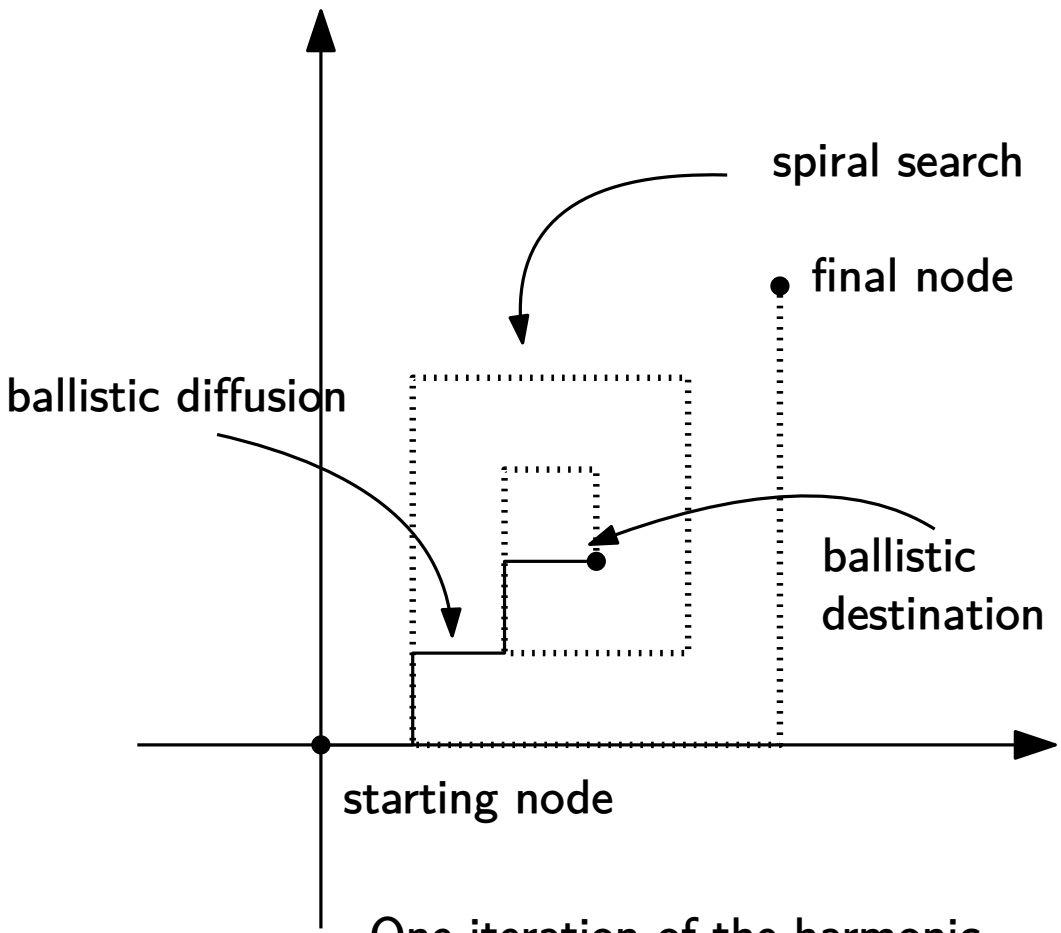
[Feinerman et Korman, DC 2017] proposes a more natural algorithm, the Harmonic search algorithm (HSA)

The Harmonic search algorithm

HSA worsens performance, but increases probability: the hitting time is

$$\mathcal{O}\left(\ell^{2+\delta}/k+\ell\right)$$

with probability $1-\epsilon$ for any fixed constants $0<\delta,\ \epsilon<1$



One iteration of the harmonic search algorithm:

The Harmonic search algorithm

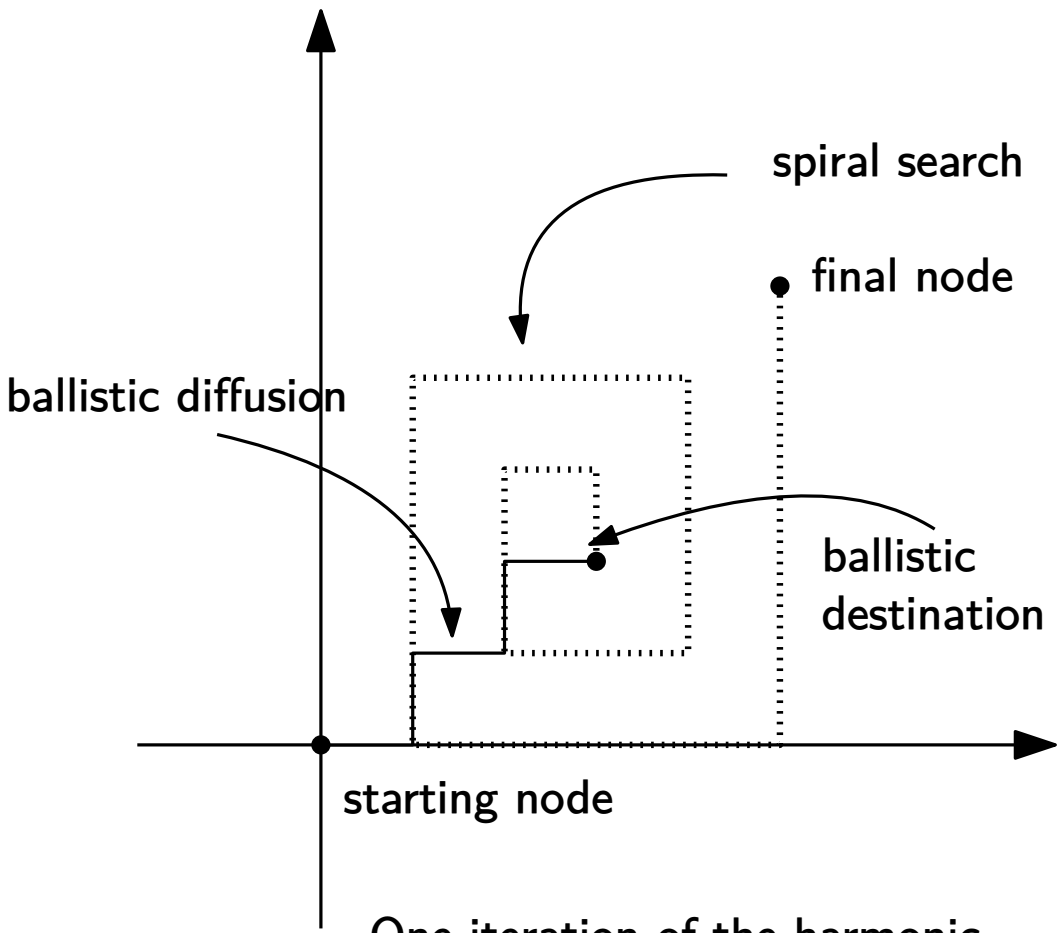
HSA worsens performance, but increases probability: the hitting time is

$$\mathcal{O}\left(\ell^{2+\delta}/k + \ell\right)$$

with probability $1-\epsilon$ for any fixed constants $0 < \delta, \epsilon < 1$

HSA: each agent

a) samples a jump-length d with a power-law distribution with exponent $\alpha = 1 + \delta$ (small)
b) (ballistic diffusion) moves to a destination at distance d chosen u.a.r.
c) (normal diffusion) starts a spiral search for $d^{\delta+2}$ steps
d) returns in the origin and repeats



One iteration of the harmonic search algorithm:

Appendix

1. Lévy walks

2. Opinion dynamics

3. Assembly calculus

Techniques

For **consesus**:

- symmetry breaking: the bias has enough standard deviation to break symmetry (drift analysis results)
- applying concentration inequalities to construct a process M_t such that $M_{t+1} \leq (1-\delta)M_t$ w.h.p. as long as the bias is outside $I_\varepsilon = [(1-\varepsilon)\bar{s}, (1+\varepsilon)\bar{s}]$ but at least $\Omega(\sqrt{n \log n})$
- chain rule + union bound

Techniques

For **consesus**:

- symmetry breaking: the bias has enough standard deviation to break symmetry (drift analysis results)
- applying concentration inequalities to construct a process M_t such that $M_{t+1} \leq (1-\delta)M_t$ w.h.p. as long as the bias is outside $I_\varepsilon = [(1-\varepsilon)\bar{s}, (1+\varepsilon)\bar{s}]$ but at least $\Omega(\sqrt{n \log n})$
- chain rule + union bound

For **victory of noise**:

- constructing a super-martingale N_t such that $\mathbb{E}[N_{t+1}|\mathcal{F}_t] \leq (1-\delta)N_t$ involving the bias
- concentration arguments for super-martingales [Lehre and Witt, ISAAC 2014] implying that the bias reaches 0

Techniques

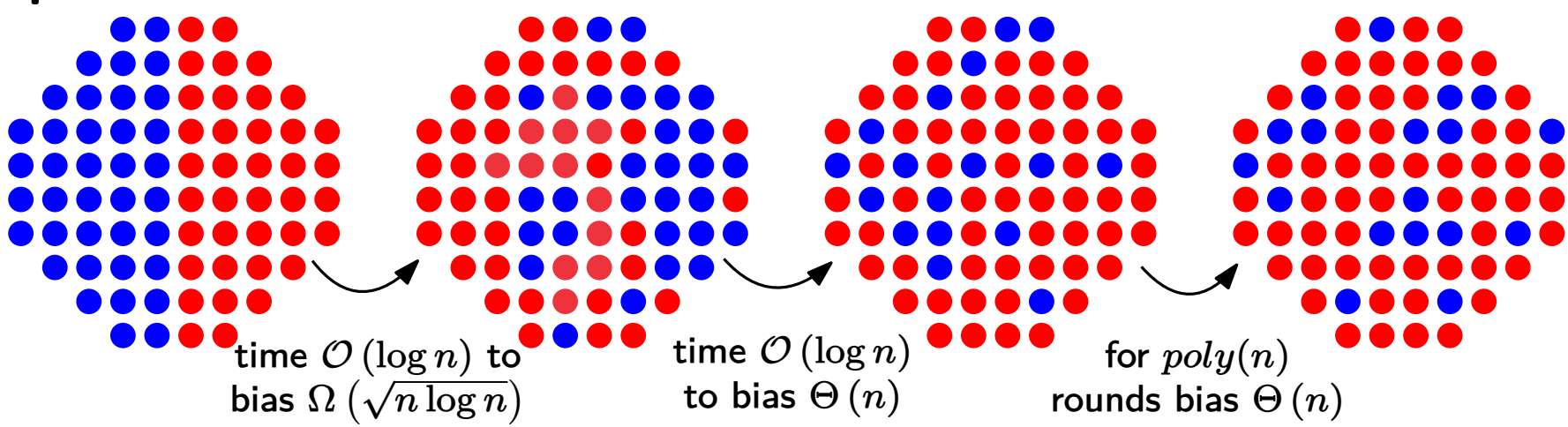
For **consesus**:

- symmetry breaking: the bias has enough standard deviation to break symmetry (drift analysis results)
- applying concentration inequalities to construct a process M_t such that $M_{t+1} \leq (1-\delta)M_t$ w.h.p. as long as the bias is outside $I_\varepsilon = [(1-\varepsilon)\bar{s}, (1+\varepsilon)\bar{s}]$ but at least $\Omega(\sqrt{n \log n})$
- chain rule + union bound

For **victory of noise**:

- constructing a super-martingale N_t such that $\mathbb{E}[N_{t+1}|\mathcal{F}_t] \leq (1-\delta)N_t$ involving the bias
- concentration arguments for super-martingales [Lehre and Witt, ISAAC 2014] implying that the bias reaches 0

Example: consensus



Appendix

1. Lévy walks

2. Opinion dynamics

3. Assembly calculus

Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

Assemblies

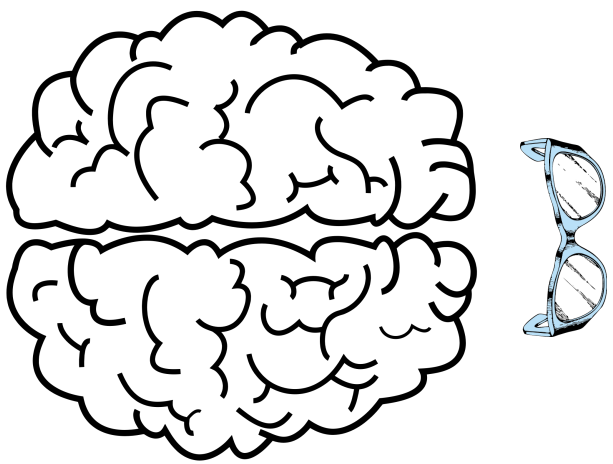
Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]

Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

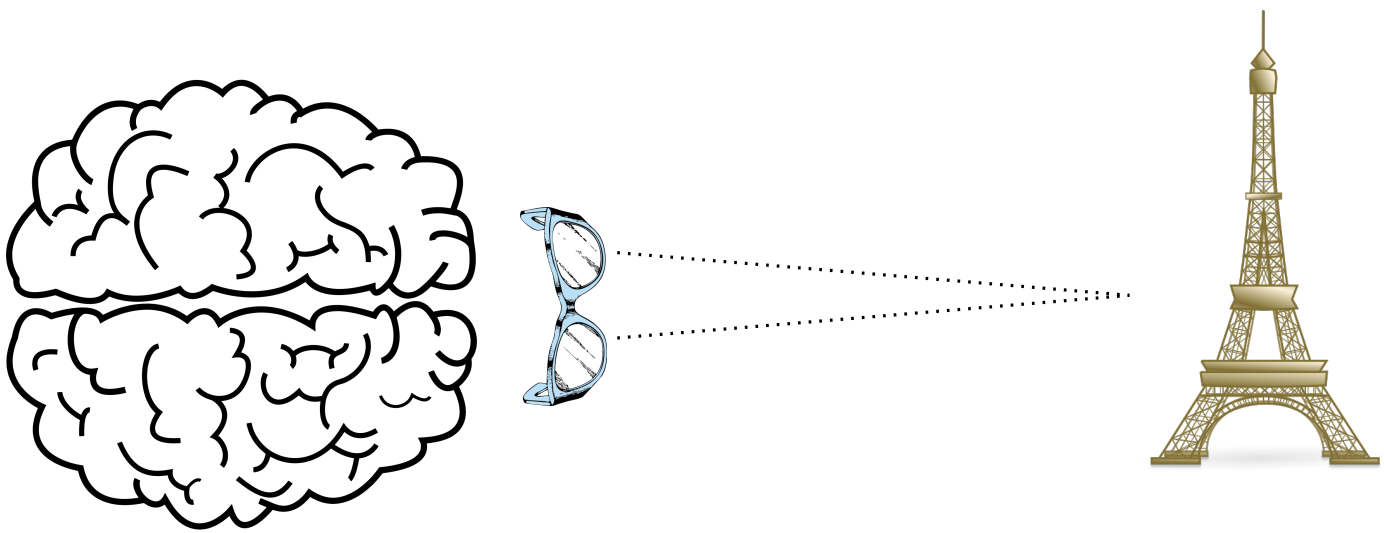
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

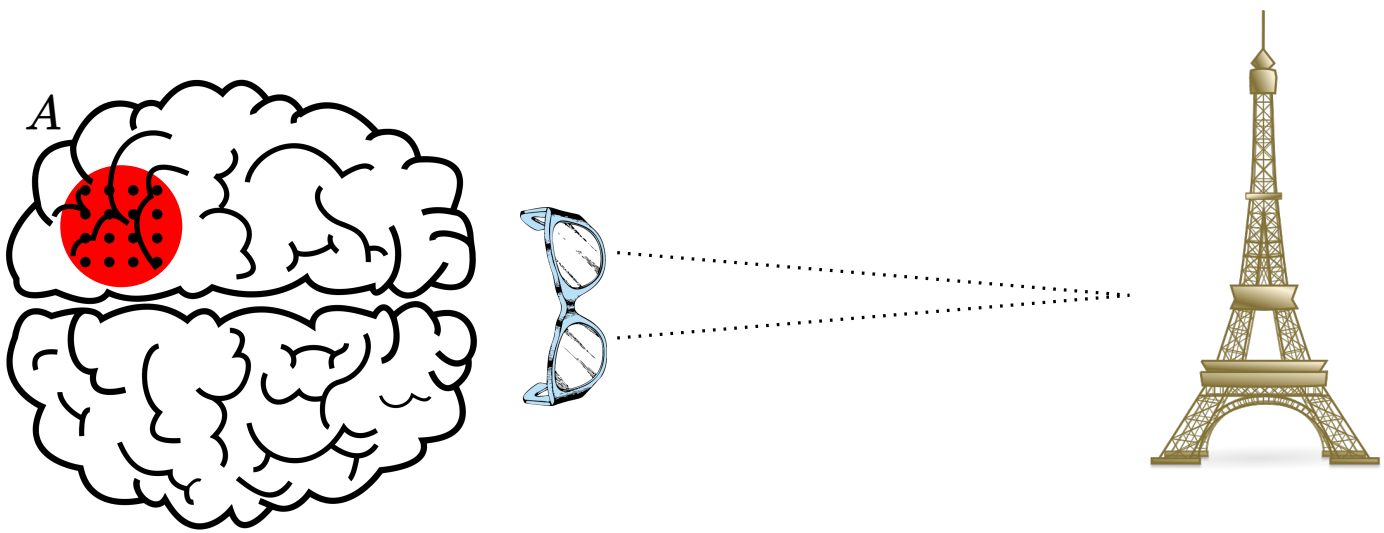
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

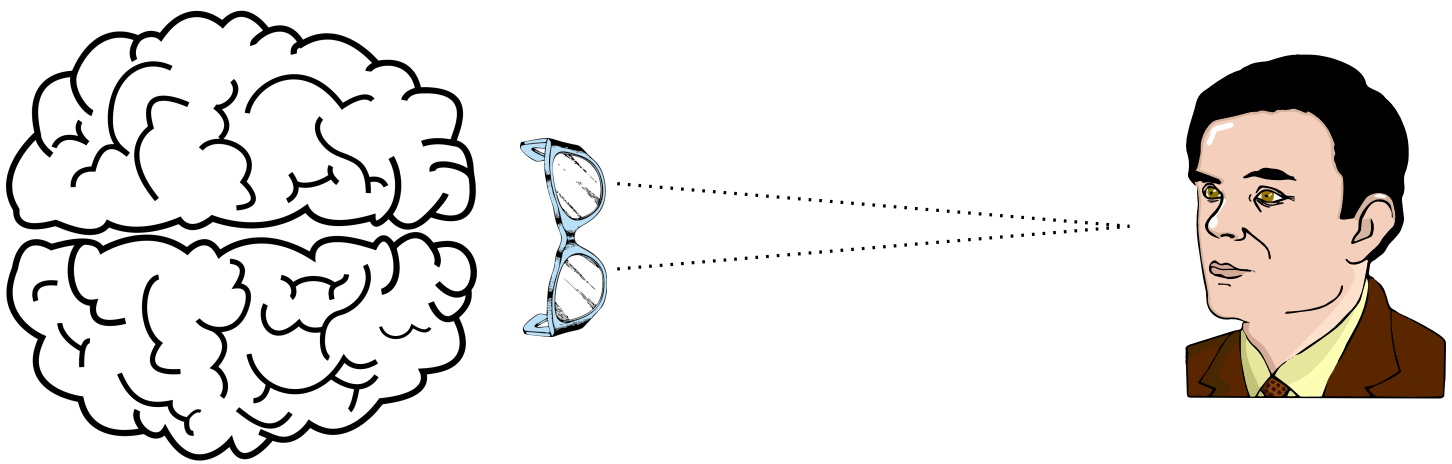
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

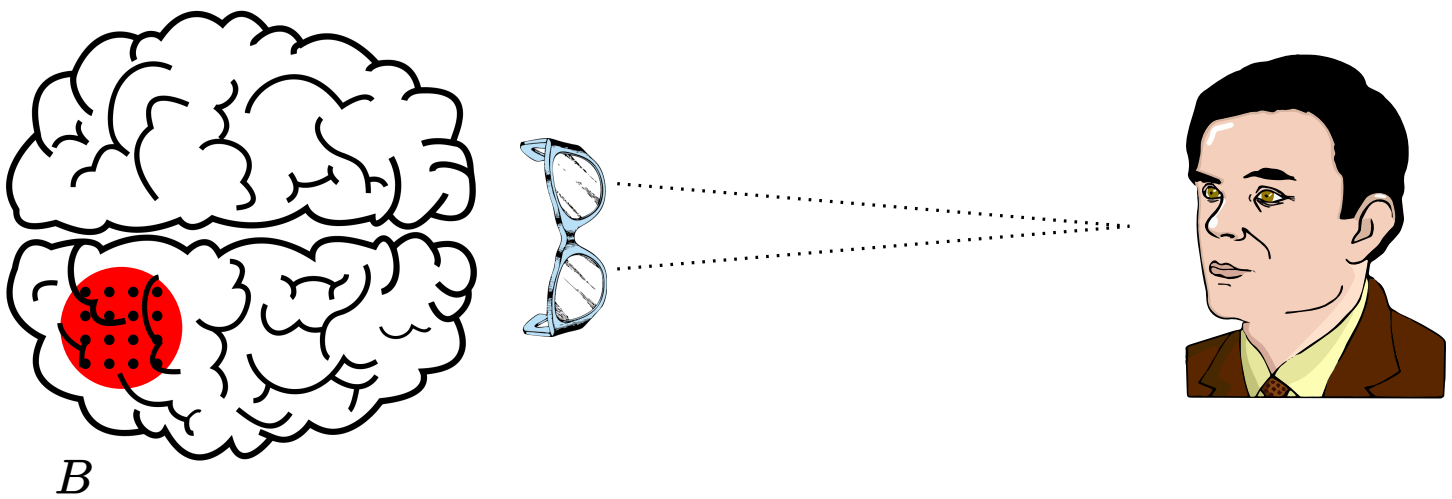
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

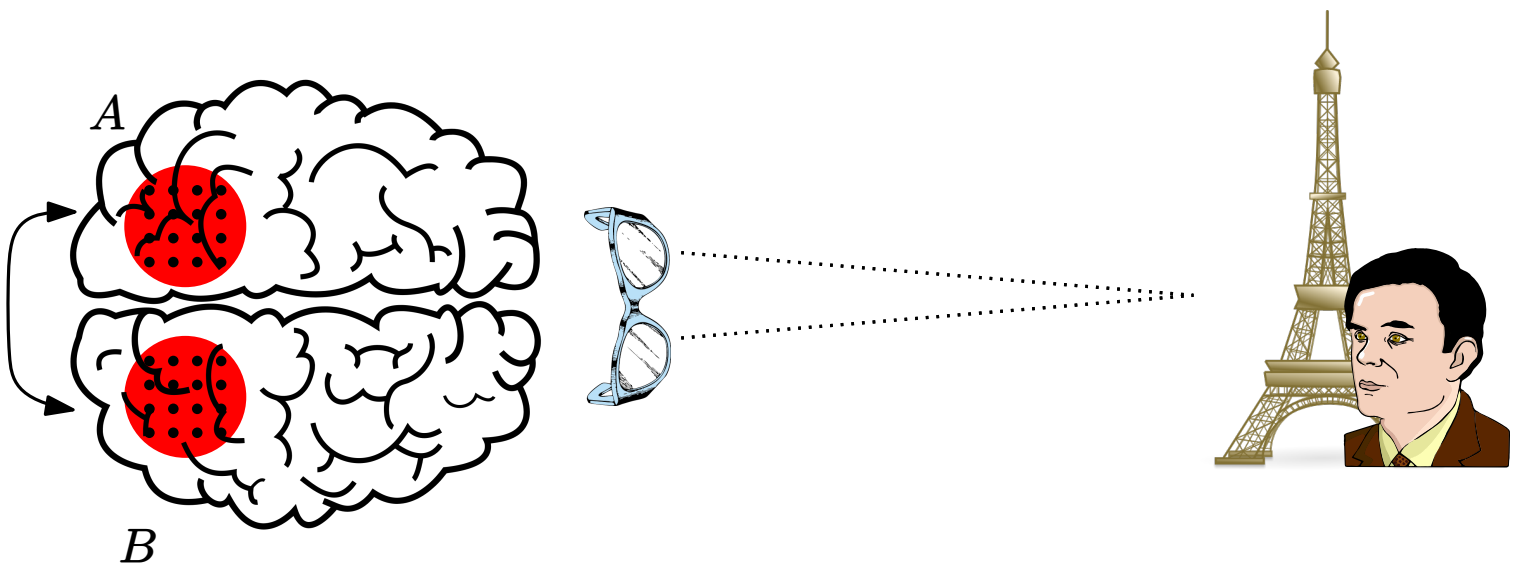
Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Assemblies

Hypothized by [Hebb, 1949]: densely interconnected sets of neurons whose loosely synchronized firing in a pattern is simultaneous with the subject thinking of a particular **concept** or **idea**

Confirmed by [Harris, Nat. Rev. Neurosci. 2005], [Buzsaki, Neuron 2010]



Association: experiment by [Ison et al., Neuron 2015]

The Model

- **Brain area**: $A = G_{n_A, p_A}$ Erdös Rényi graph

- *Recall* $G_{n,p} = (V, E)$:
  - $|V| = n$
  - $\forall x, y \in V$, $(x, y) \in E$ with probability p

The Model

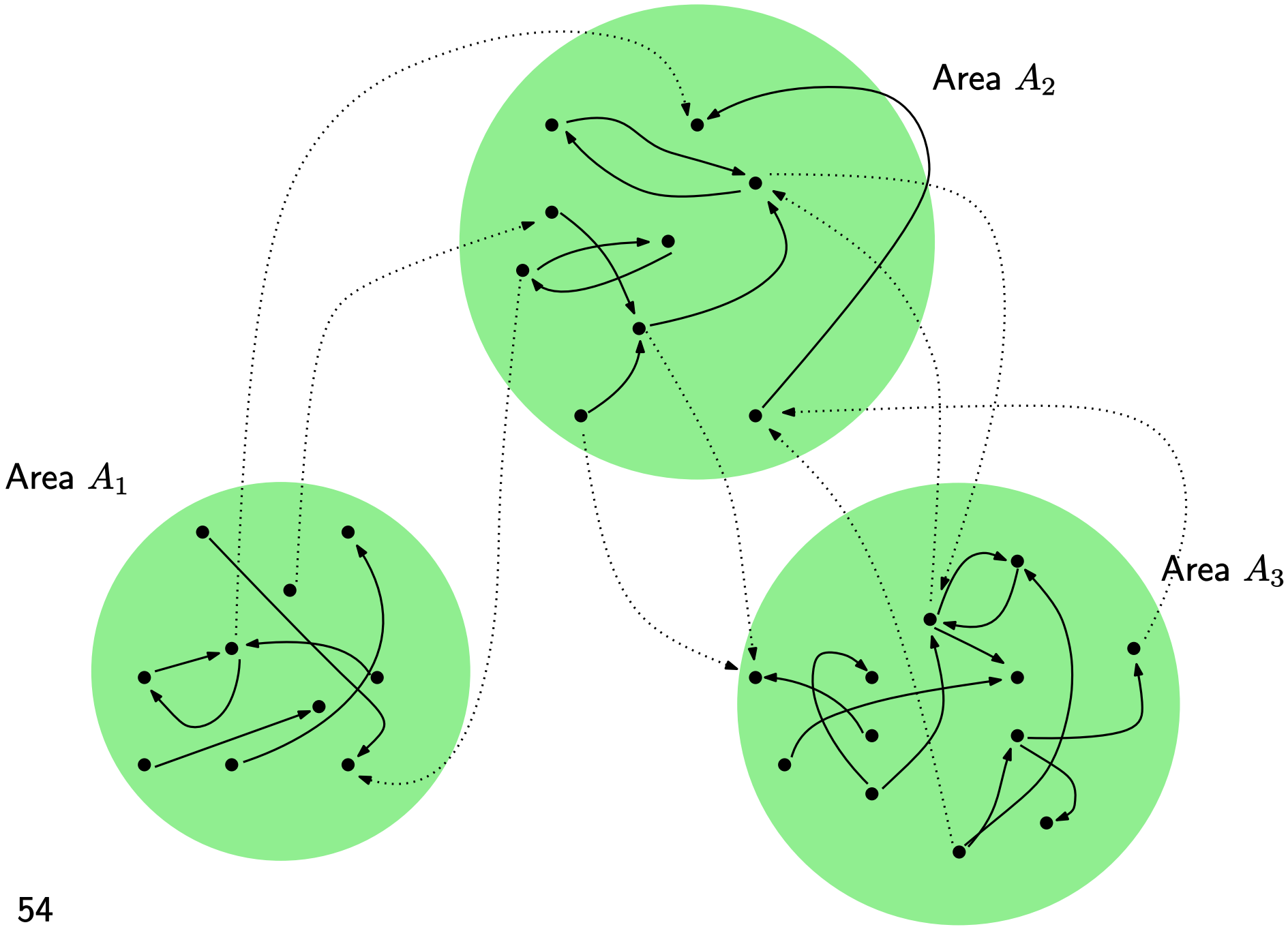
- **Brain area**: $A = G_{n_A, p_A}$ Erdös Rényi graph

- *Recall* $G_{n,p} = (V, E)$:
  - $|V| = n$
  - $\forall x, y \in V$, $(x, y) \in E$ with probability p

- set of brain areas $\mathcal{S} = \{A_1, A_2, \ldots, A_m\}$
- $\mathcal{C} \subseteq \mathcal{S} \times \mathcal{S}$ set of ordered pairs of brain regions

- **The Brain**: $\mathcal{B} = (V_{\mathcal{B}}, E_{\mathcal{B}})$ with
  - $V_{\mathcal{B}} = V_{A_1} \cup \cdots \cup V_{A_m}$
  - $E_{\mathcal{B}} = E \cup E_{A_1} \cup \ldots E_{A_m}$
  - E: $\forall (A, B) \in \mathcal{C}$, $\forall x \in A, y \in B$, $(x, y) \in E$ with probability $p_{A,B}$

Example of a Brain



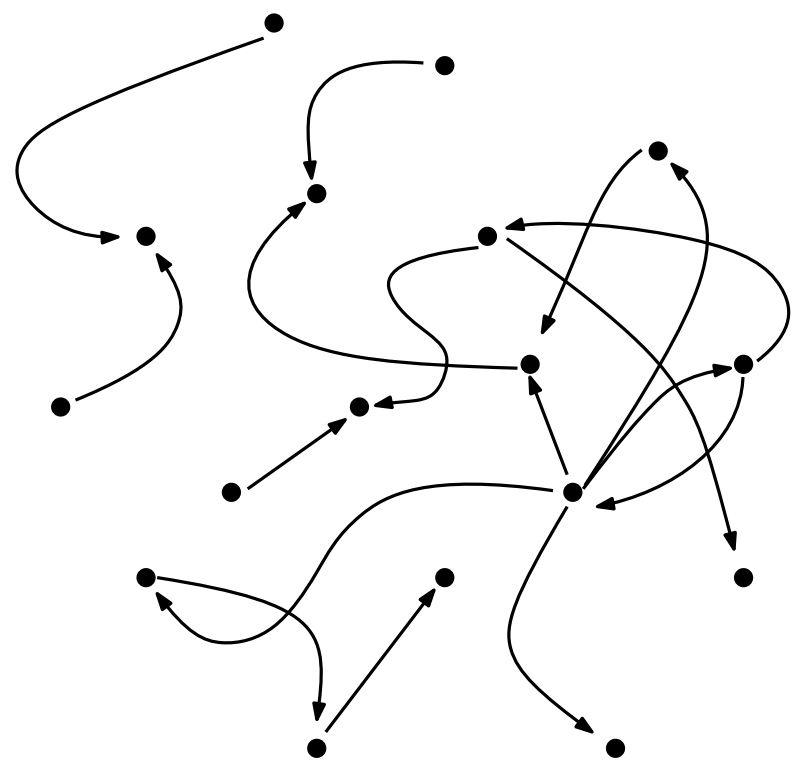
The AC dynamics

Neuron = vertex **Fiber** = edge

All fiber weights initialized to be 1

Input for a neuron:

- external input (number)
- sum of incoming fiber weights whose sources have fired



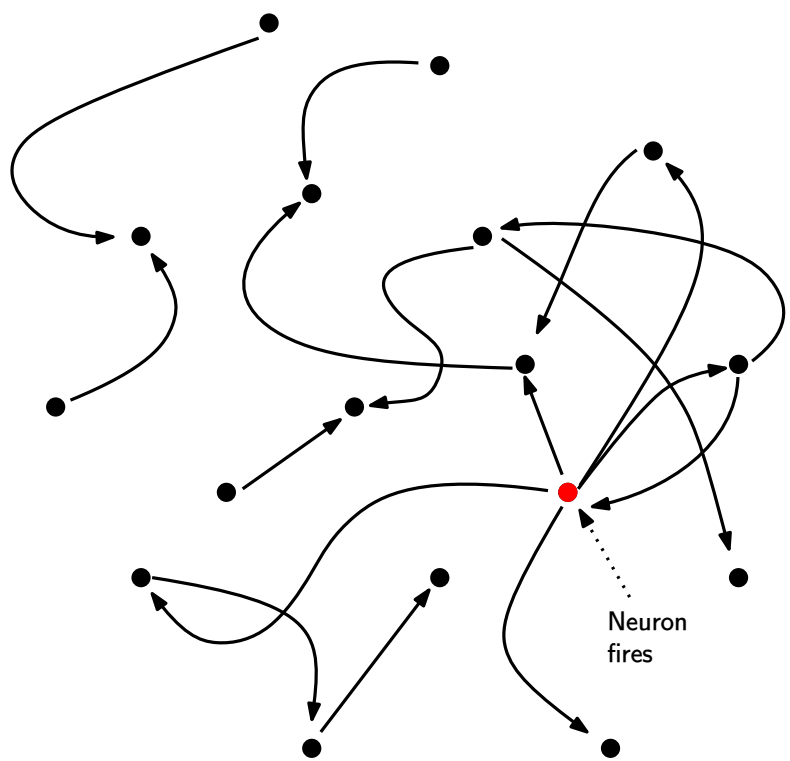
The AC dynamics

Neuron = vertex **Fiber** = edge

All fiber weights initialized to be 1

Input for a neuron:

- external input (number)
- sum of incoming fiber weights whose sources have fired



The AC dynamics

Neuron = vertex **Fiber** = edge

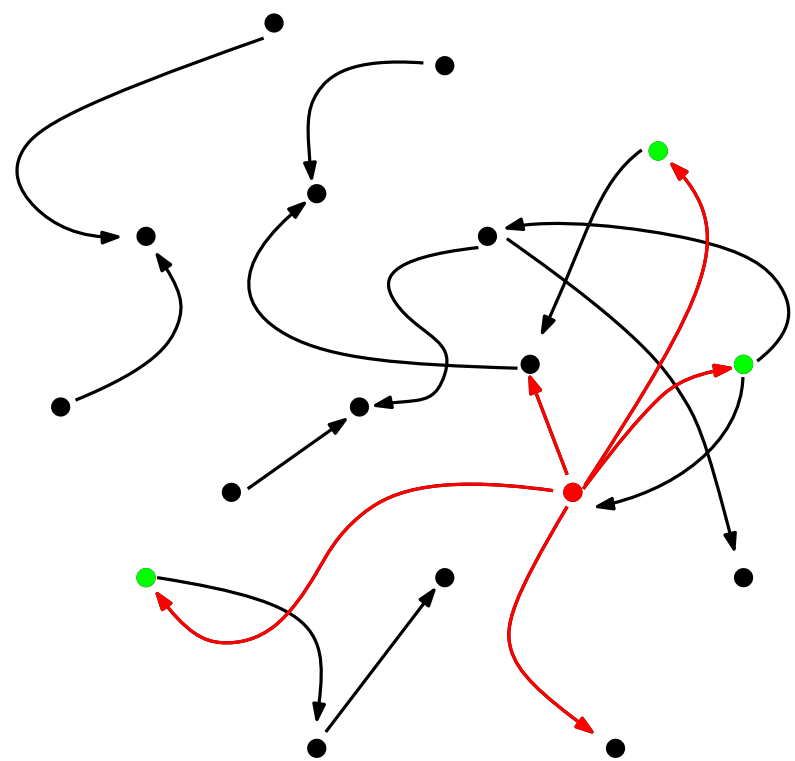
All fiber weights initialized to be 1

Input for a neuron:

- external input (number)
- sum of incoming fiber weights whose sources have fired

k-CAP rule: neuron fires if it is among the k neurons with the highest input

Here (toy): $k = 3$ (ties broken u.a.r.)



The AC dynamics

Neuron = vertex **Fiber** = edge

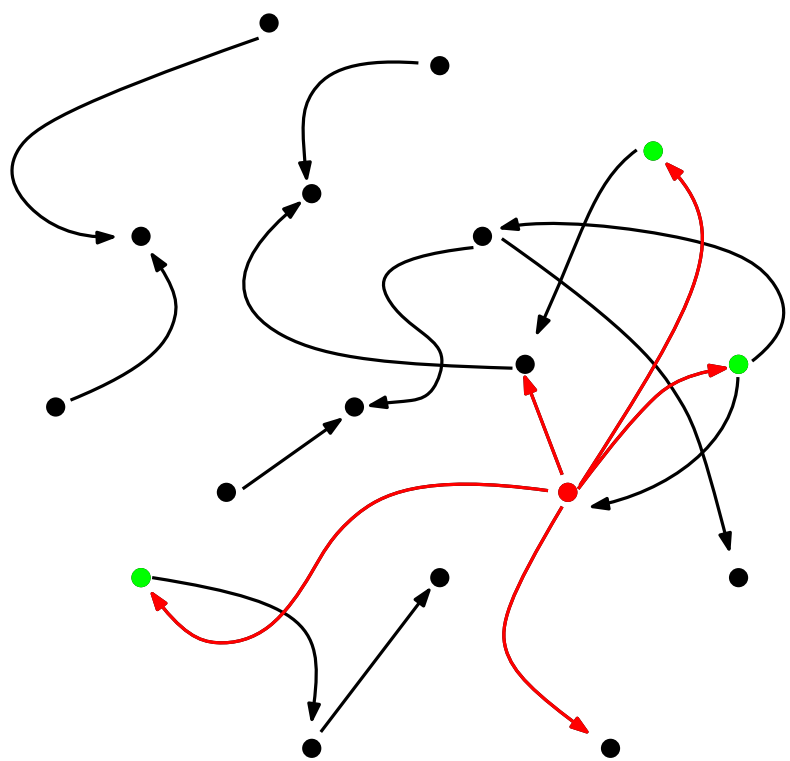
All fiber weights initialized to be 1

Input for a neuron:

- external input (number)
- sum of incoming fiber weights whose sources have fired

k-CAP rule: neuron fires if it is among the k neurons with the highest input

Here (toy): $k = 3$ (ties broken u.a.r.)



Upon fiber endpoints activation:

weight = old_weight $\cdot (1 + \beta)$

Hebbian plasticity

The AC dynamics

Neuron = vertex **Fiber** = edge

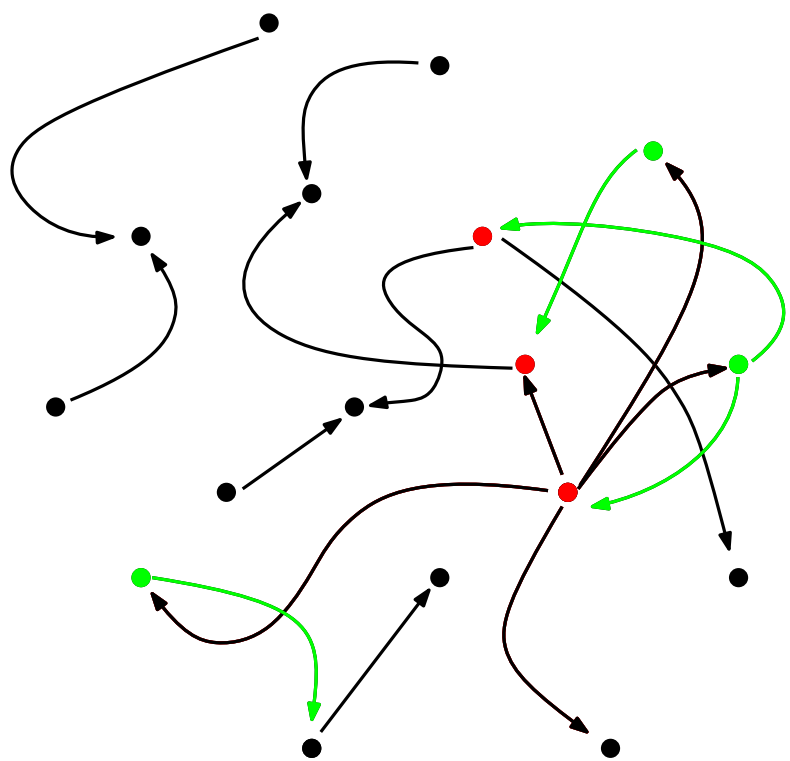
All fiber weights initialized to be 1

Input for a neuron:

- external input (number)
- sum of incoming fiber weights whose sources have fired

k-CAP rule: neuron fires if it is among the k neurons with the highest input

Here (toy): $k = 3$ (ties broken u.a.r.)



Upon fiber endpoints activation:

weight = old_weight $\cdot (1 + \beta)$

Hebbian plasticity

The AC dynamics

Neuron = vertex **Fiber** = edge

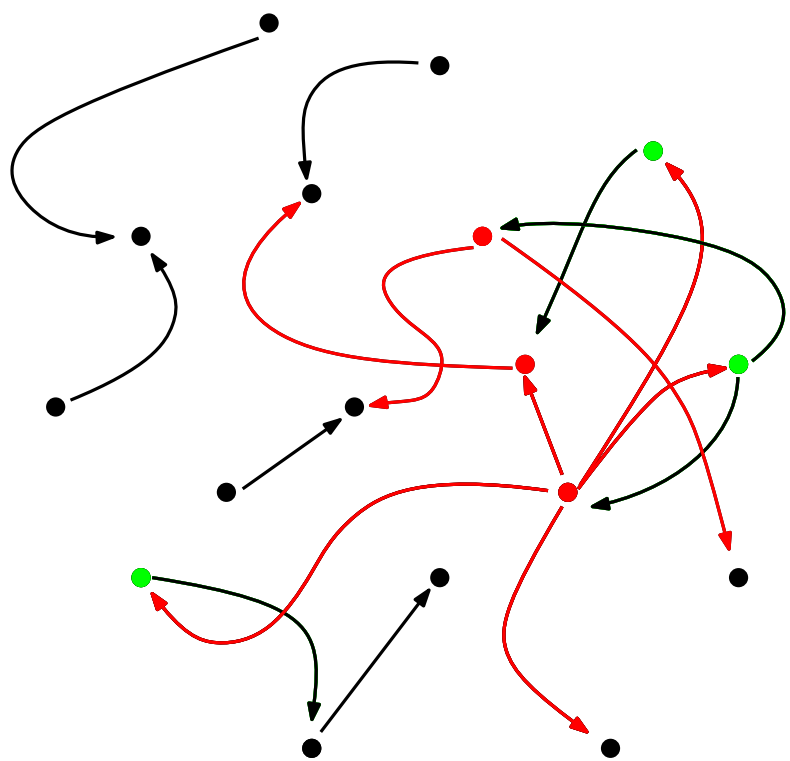
All fiber weights initialized to be 1

Input for a neuron:

- external input (number)
- sum of incoming fiber weights whose sources have fired

k-CAP rule: neuron fires if it is among the k neurons with the highest input

Here (toy): $k = 3$ (ties broken u.a.r.)



Upon fiber endpoints activation:

weight = old_weight $\cdot (1 + \beta)$

Hebbian plasticity

The AC dynamics

Neuron = vertex **Fiber** = edge

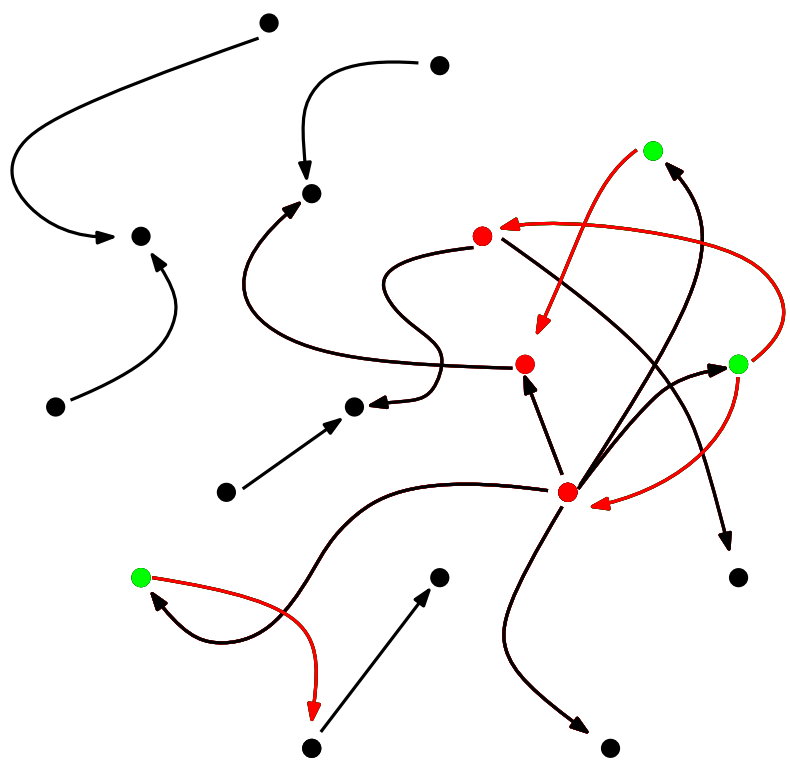
All fiber weights initialized to be 1

Input for a neuron:

- external input (number)
- sum of incoming fiber weights whose sources have fired

k-CAP rule: neuron fires if it is among the k neurons with the highest input

Here (toy): $k = 3$ (ties broken u.a.r.)



Upon fiber endpoints activation:

weight = old_weight $\cdot (1 + \beta)$

Hebbian plasticity

The AC dynamics

Neuron = vertex **Fiber** = edge

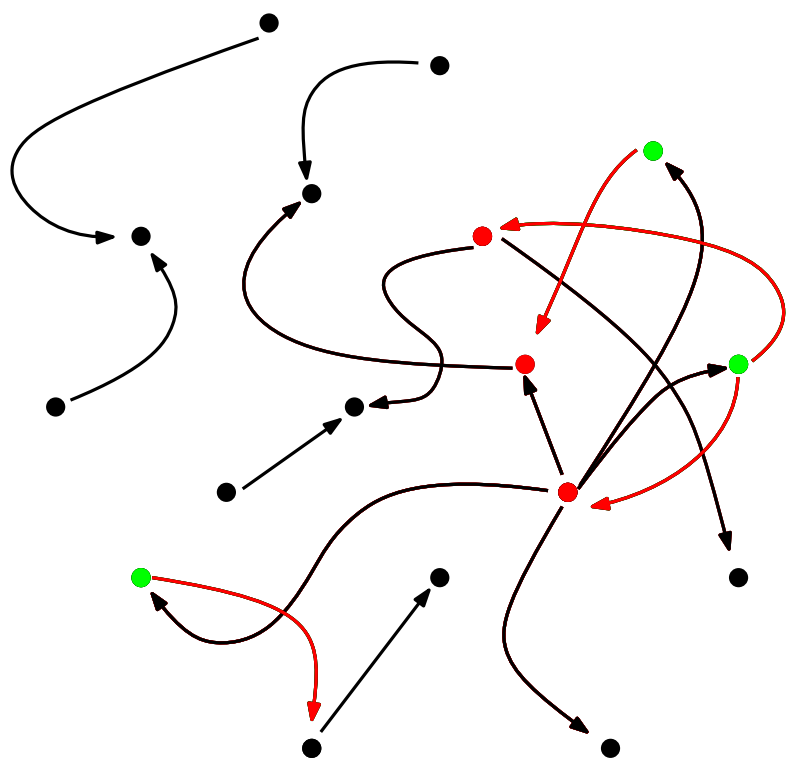
All fiber weights initialized to be 1

Input for a neuron:

- external input (number)
- sum of incoming fiber weights whose sources have fired

k-CAP rule: neuron fires if it is among the k neurons with the highest input

Here (toy): $k = 3$ (ties broken u.a.r.)



Upon fiber endpoints activation:

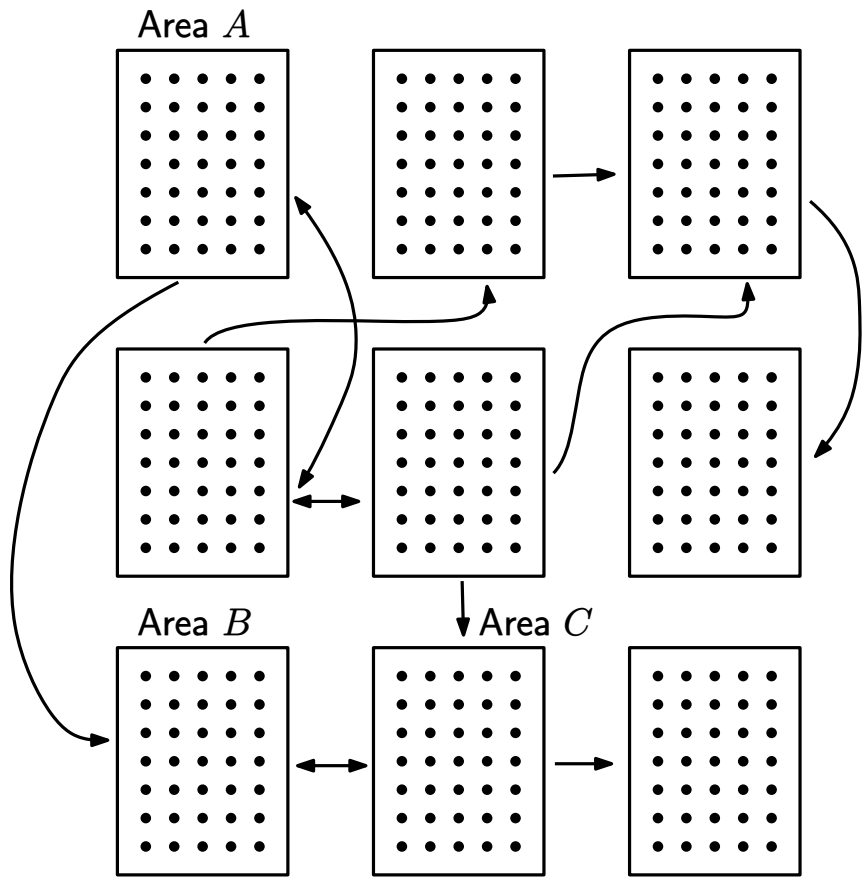
weight = old_weight $\cdot (1 + \beta)$

Hebbian plasticity

Assembly: *stable* set of k neurons

Operations

The brain and its areas

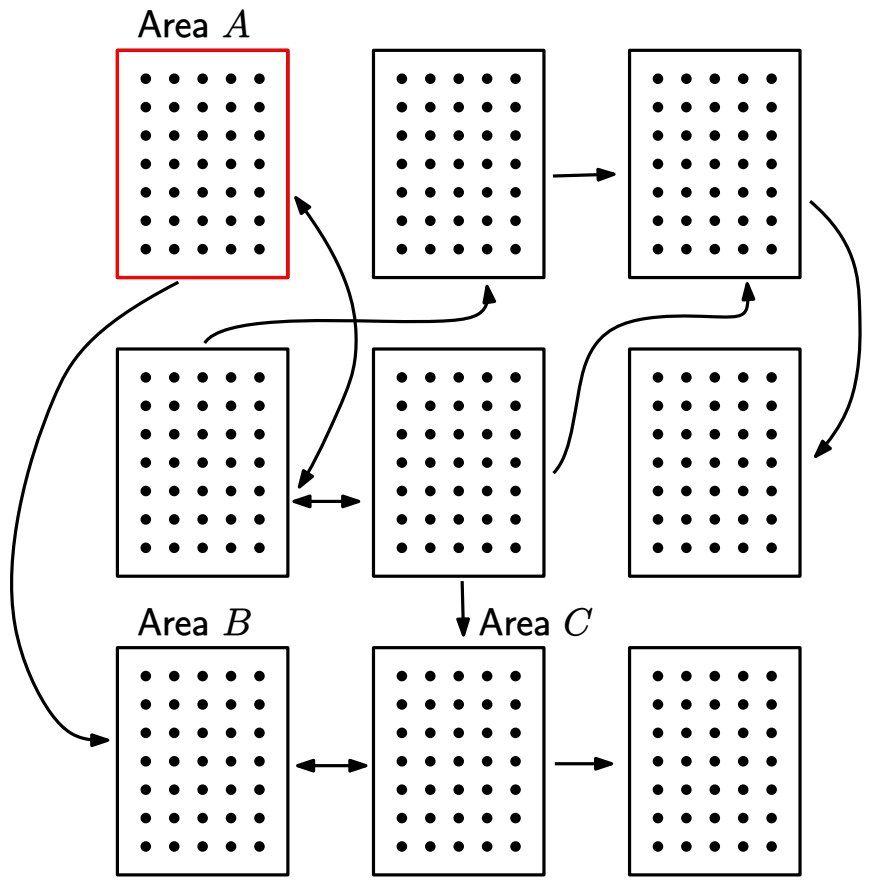


Operations

- **Inhibition**: an area can be inhibited, that is, its neurons cannot fire

Inhibition is accomplished through populations of inhibitory neurons, whose firing prevents other neurons from firing [Mitropolsky et al., TACL 2021]

The brain and its areas



For areas A, B, C:

- `inhibitArea` (A)

Operations

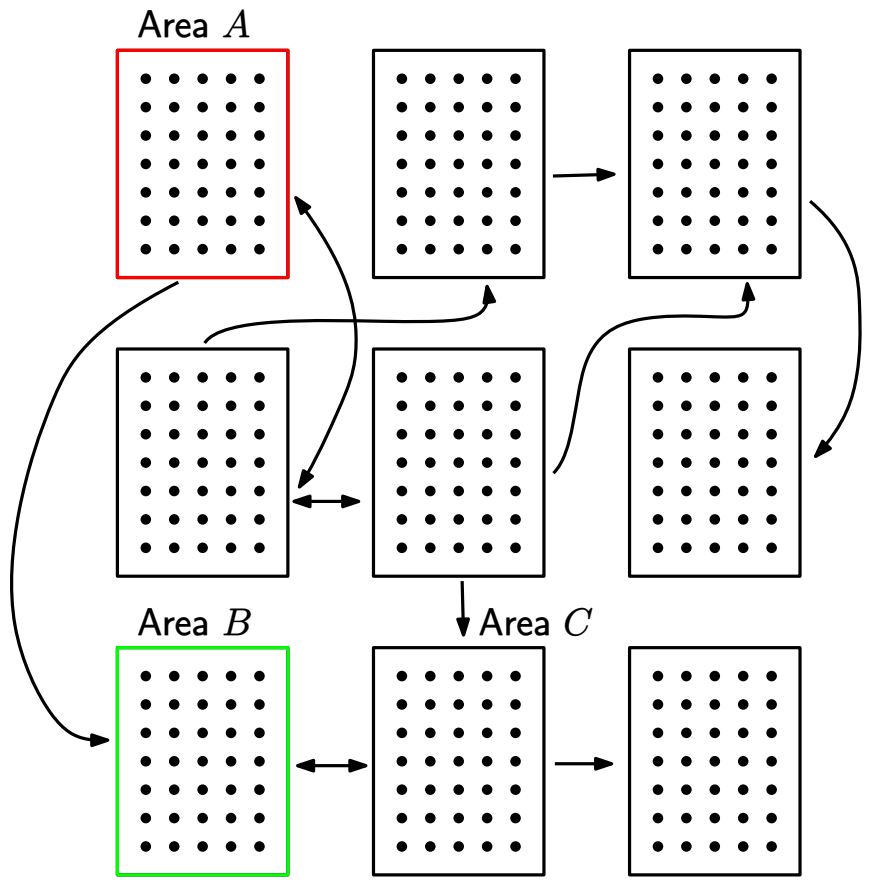
- **Inhibition**: an area can be inhibited, that is, its neurons cannot fire

Inhibition is accomplished through populations of inhibitory neurons, whose firing prevents other neurons from firing [Mitropolsky et al., TACL 2021]

- **Disinhibition**: an inhibited area can be disinhibited, that is, its neurons can now fire

Disinhibition inhibits a population of inhibitory neurons, which currently inhibit the area [Mitropolsky et al., TACL 2021]

The brain and its areas



For areas A, B, C:

- `inhibitArea` (A)
- `disinhibitArea` (B)

Operations

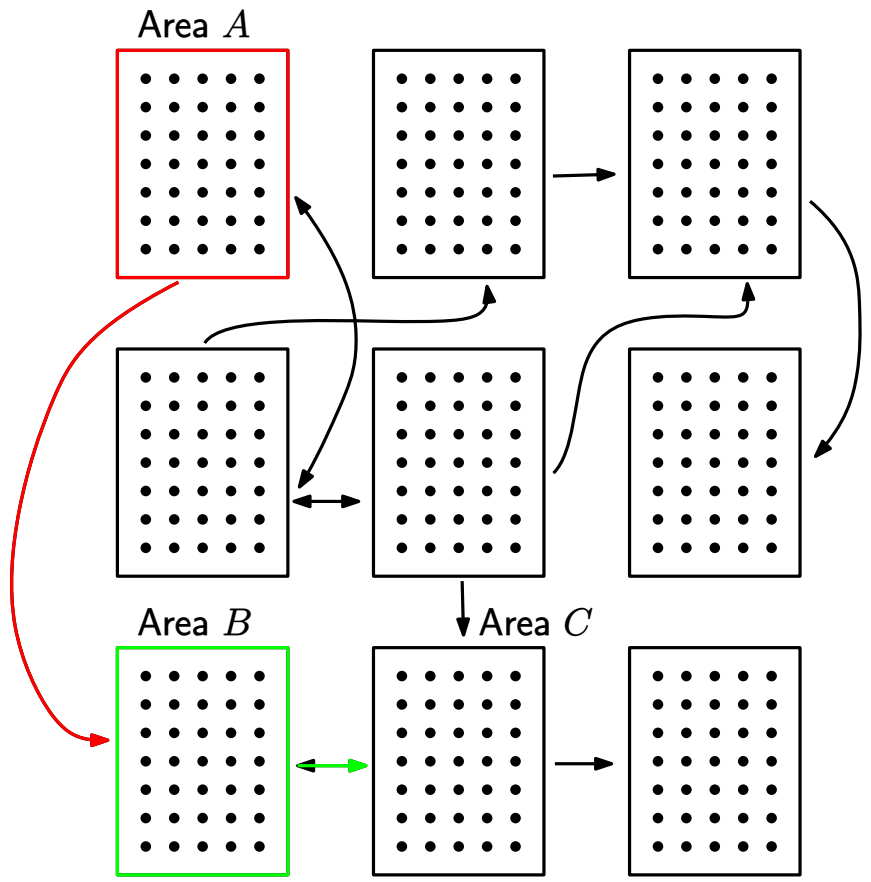
- **Inhibition**: an area can be inhibited, that is, its neurons cannot fire

Inhibition is accomplished through populations of inhibitory neurons, whose firing prevents other neurons from firing [Mitropolsky et al., TACL 2021]

- **Disinhibition**: an inhibited area can be disinhibited, that is, its neurons can now fire

Disinhibition inhibits a population of inhibitory neurons, which currently inhibit the area [Mitropolsky et al., TACL 2021]

The brain and its areas



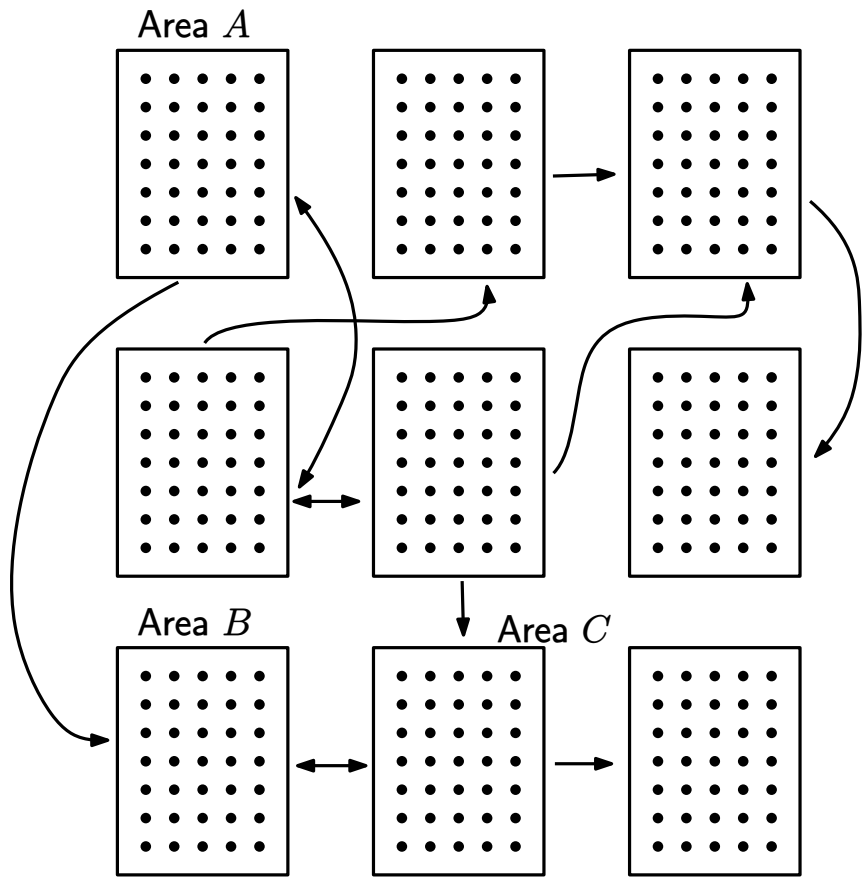
We assume we can do the same with fibers

For areas A, B, C:

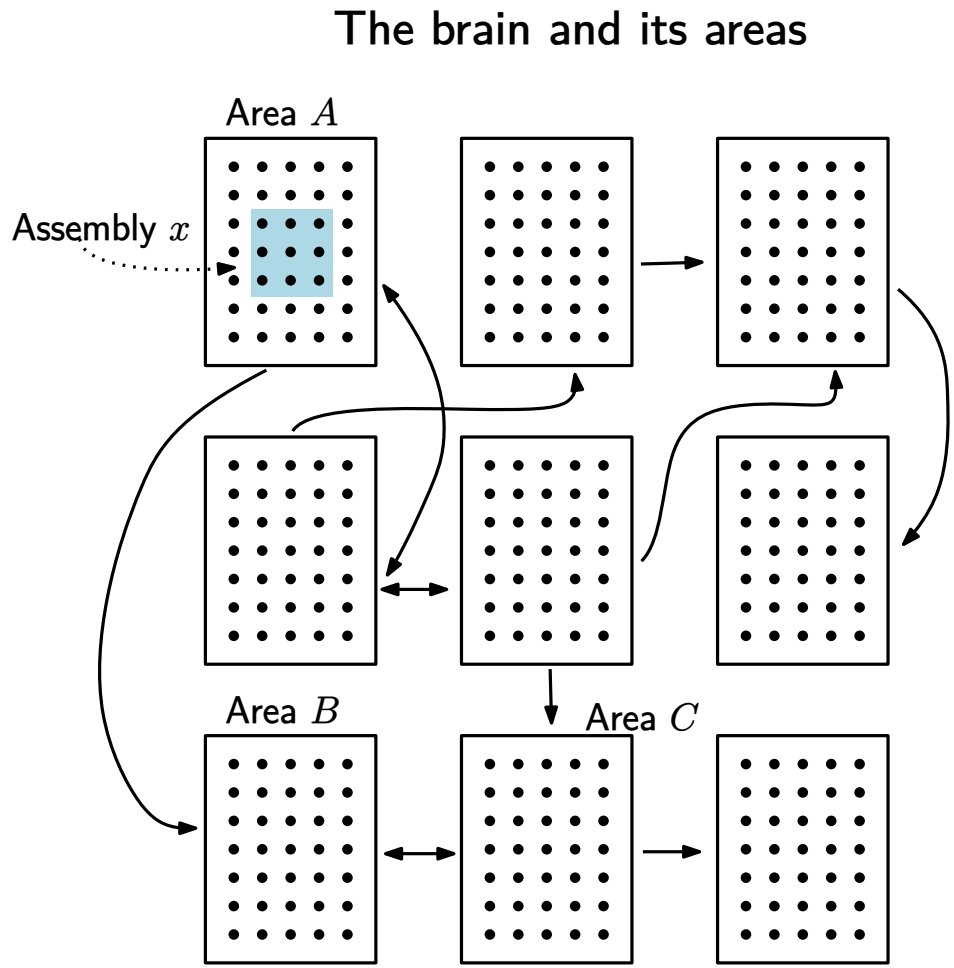
- `inhibitArea` (A)
- `inhibitFiber` (A, B)
- `disinhibitArea` (B)
- `disinhibitFiber` (B, C)

Operations

The brain and its areas



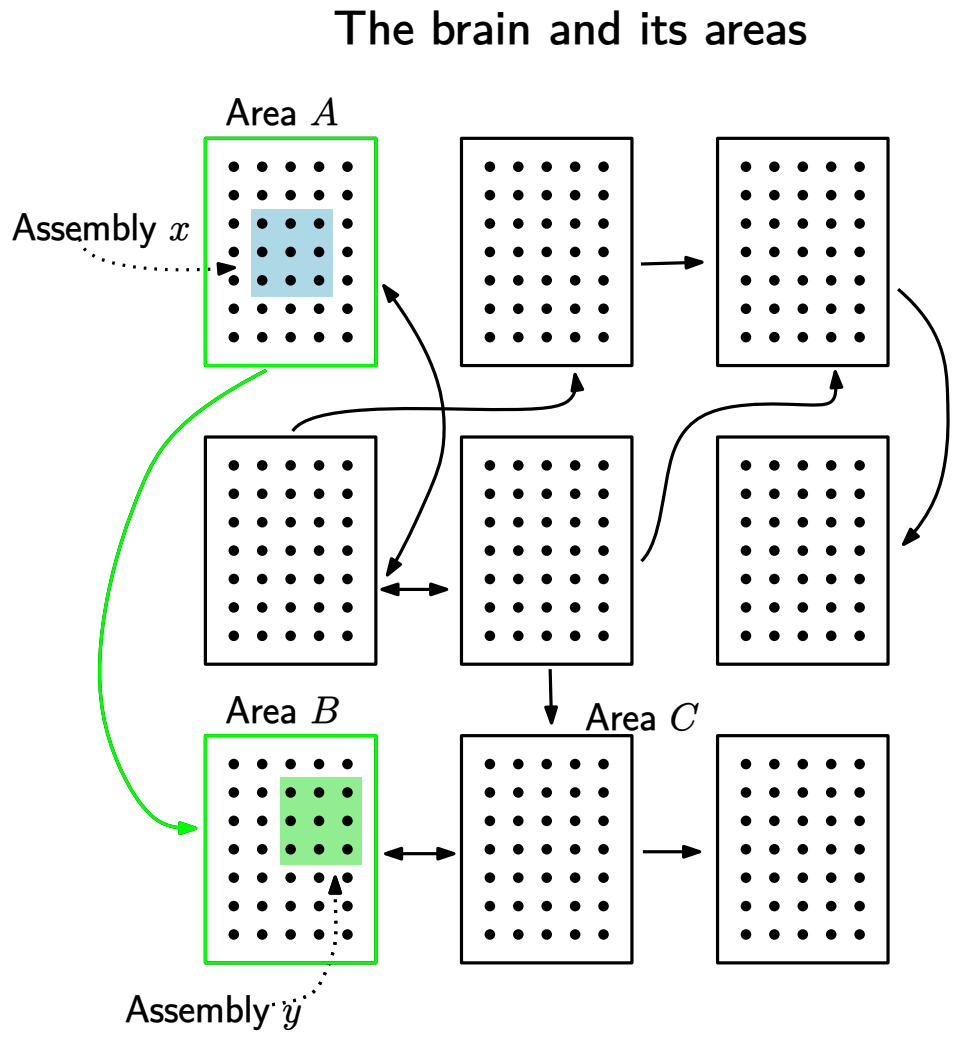
Operations



Operations

- **Projection**: the last active assembly x in area A starts firing repeatedly into area B until an assembly y is formed

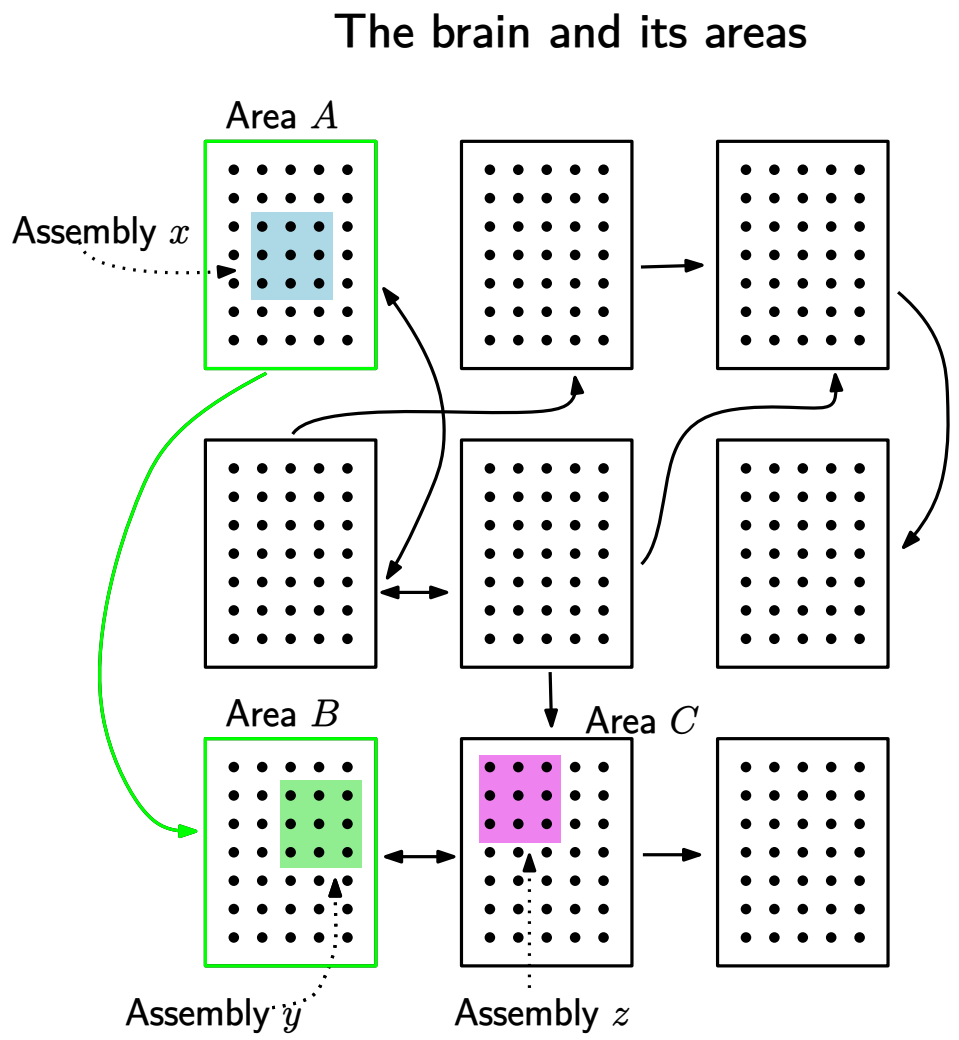
This process eventually converges under a wide range of parameters w.h.p. [Papadimitriou et Vempala, ITCS 2019]



Operations

- **Projection**: the last active assembly x in area A starts firing repeatedly into area B until an assembly y is formed

This process eventually converges under a wide range of parameters w.h.p. [Papadimitriou et Vempala, ITCS 2019]



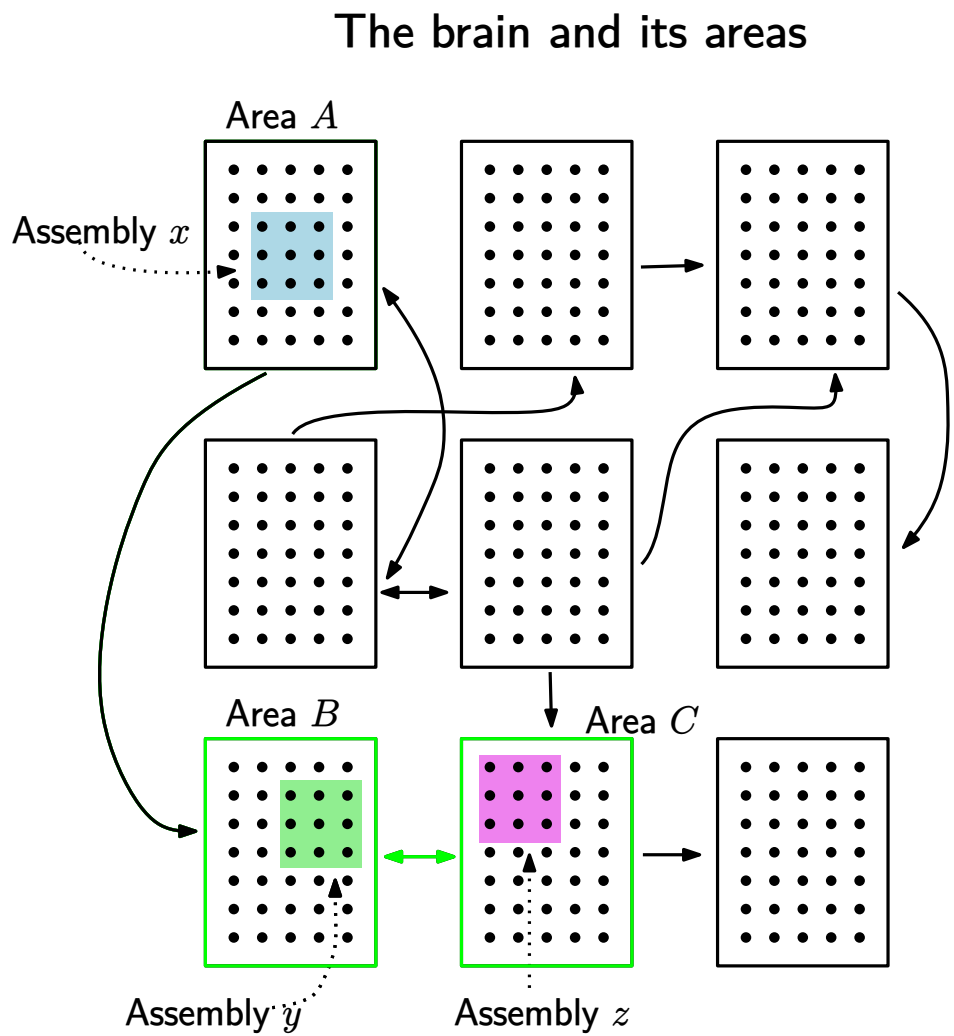
Operations

- **Projection**: the last active assembly x in area A starts firing repeatedly into area B until an assembly y is formed

This process eventually converges under a wide range of parameters w.h.p. [Papadimitriou et Vempala, ITCS 2019]

- **Association**: last active assemblies y in area B and z in area C fire simultaneously until they're linked

Linked: the firing of one results into the firing of the other



Remember

Operations

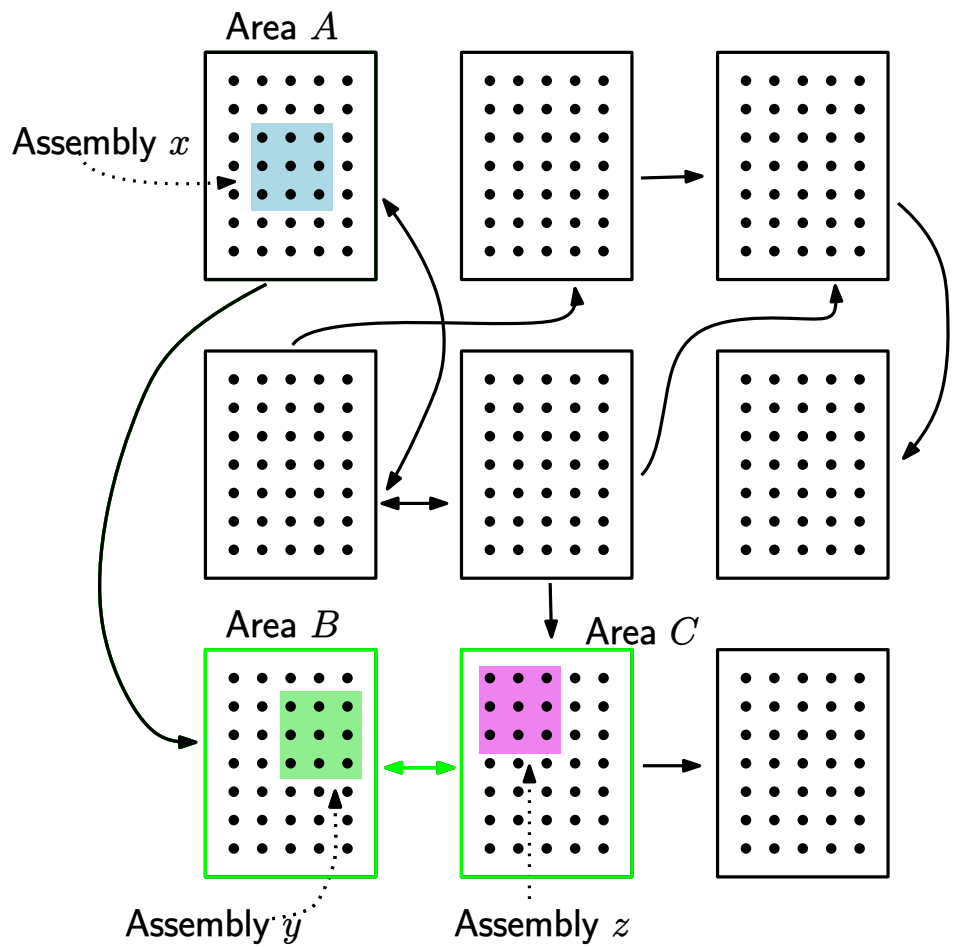
- **Projection**: the last active assembly x in area A starts firing repeatedly into area B until an assembly y is formed

This process eventually converges under a wide range of parameters w.h.p. [Papadimitriou et Vempala, ITCS 2019]

- **Association**: last active assemblies y in area B and z in area C fire simultaneously until they're linked

Linked: the firing of one results into the firing of the other

The brain and its areas



For areas A, B, C:

- `project` (A, B)
- `associate` (B, C)

Remember

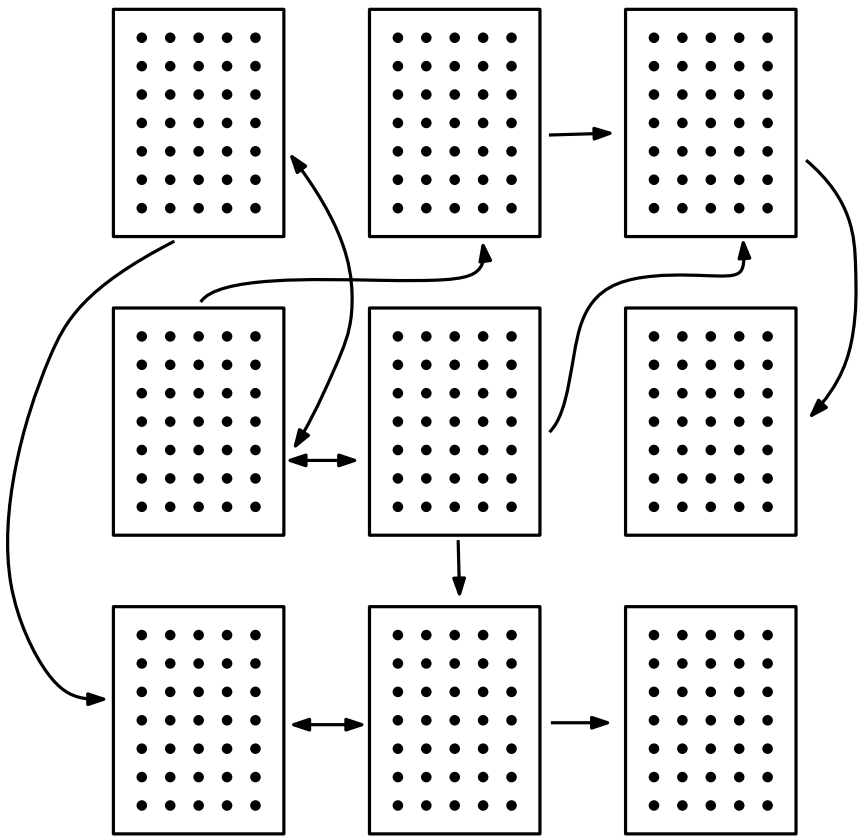
Operations

- **Strong projection**: enhancement of the projection operation

Dynamics on the subgraph induced by all disinhibited areas and fibers:

- **all active assemblies** start projecting in adjacent areas, forming new assemblies, and so on, until stability

The brain and its areas



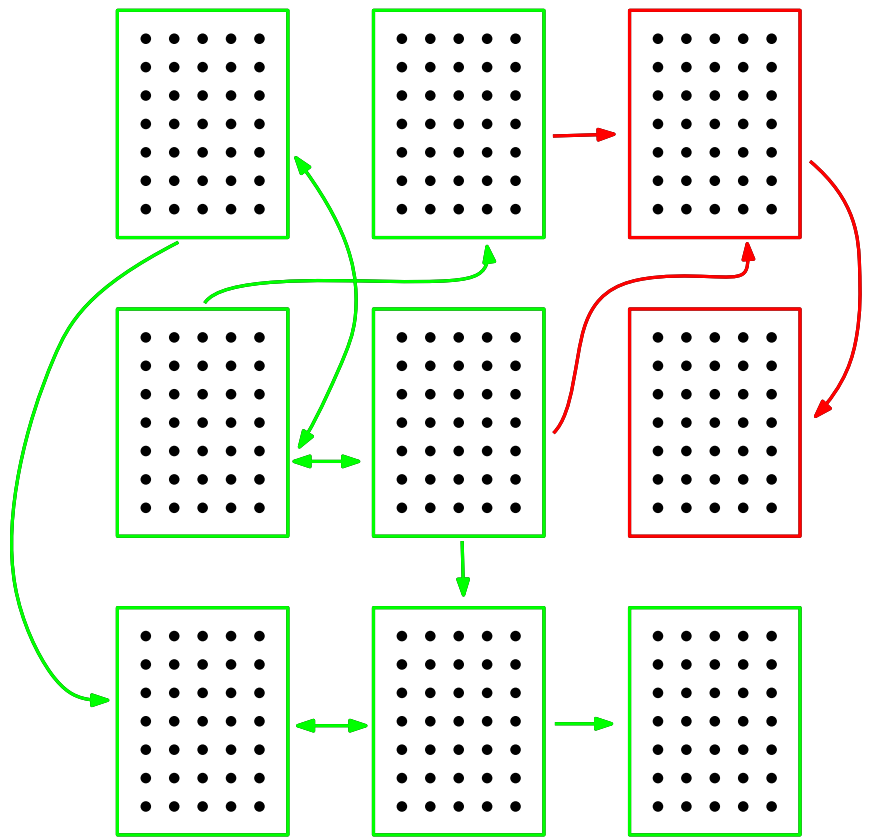
Operations

- **Strong projection**: enhancement of the projection operation

Dynamics on the subgraph induced by all disinhibited areas and fibers:

- **all active assemblies** start projecting in adjacent areas, forming new assemblies, and so on, until stability

The brain and its areas



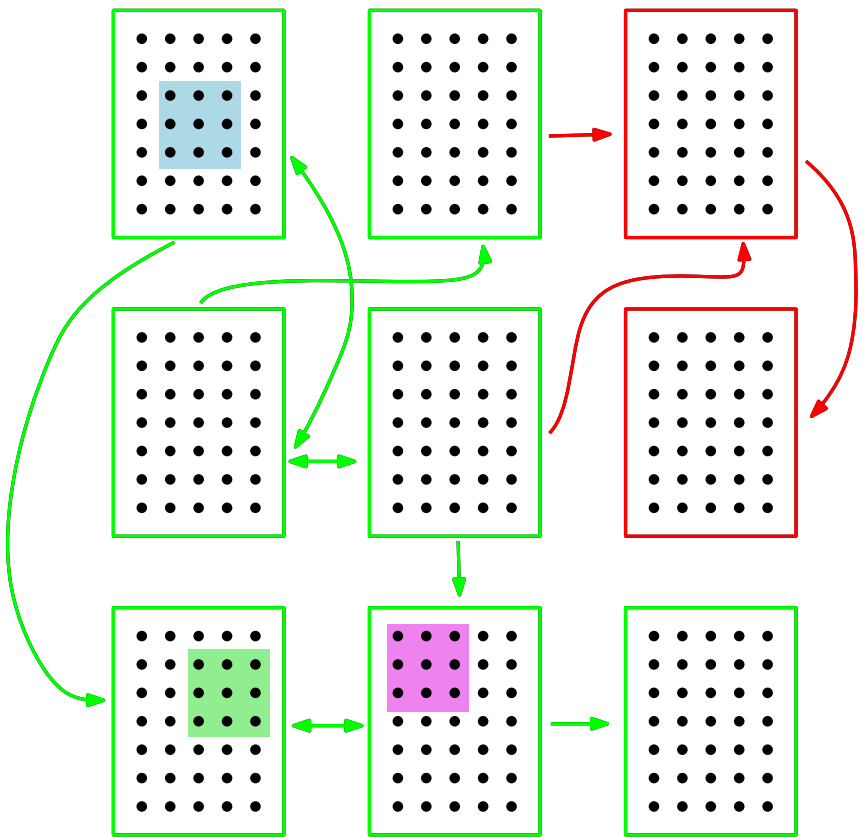
Operations

- **Strong projection**: enhancement of the projection operation

Dynamics on the subgraph induced by all disinhibited areas and fibers:

- **all active assemblies** start projecting in adjacent areas, forming new assemblies, and so on, until stability

The brain and its areas



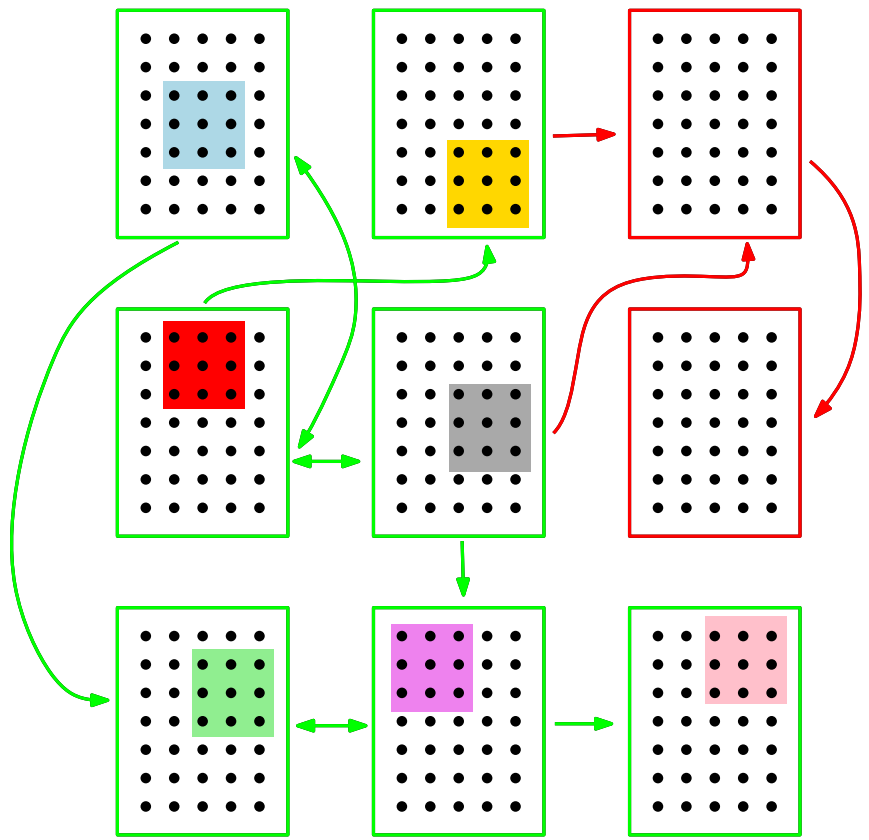
Operations

- **Strong projection**: enhancement of the projection operation

Dynamics on the subgraph induced by all disinhibited areas and fibers:

- **all active assemblies** start projecting in adjacent areas, forming new assemblies, and so on, until stability

The brain and its areas



Operations

- **Strong projection**: enhancement of the projection operation

Dynamics on the subgraph induced by all disinhibited areas and fibers:

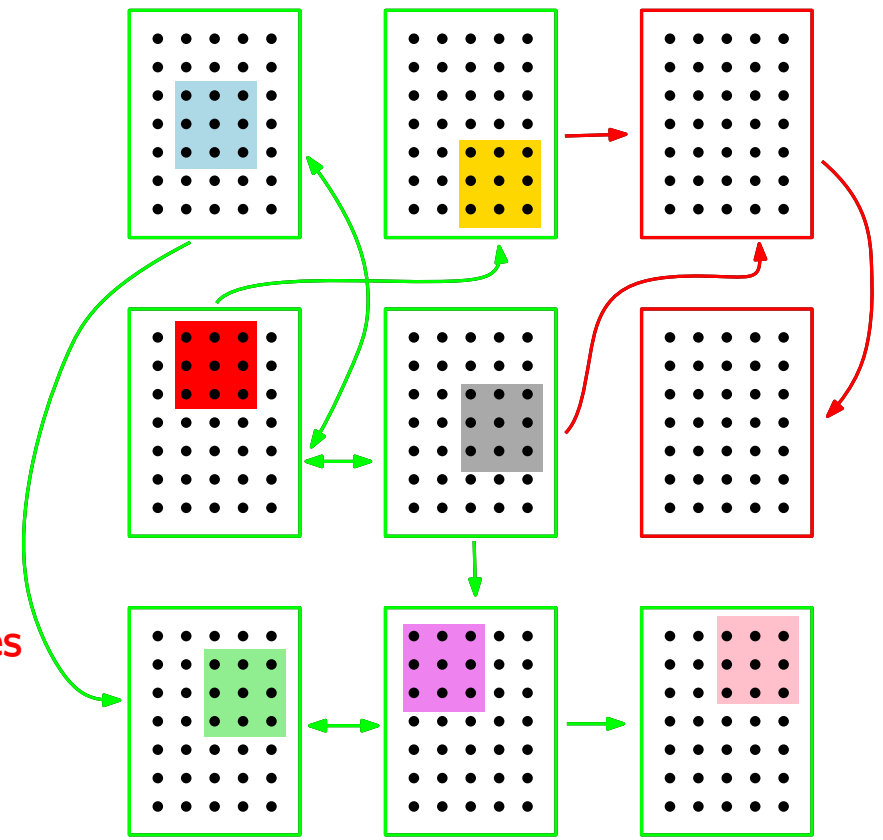
  - **all active assemblies** start projecting in adjacent areas, forming new assemblies, and so on, until stability

- **Read**: identifies whether in a given area an assembly has fired

[Buzsáki, Neuron 2010] proposes that, for assemblies to be functionally useful, readout mechanisms must exist that sense the current state of the assembly system and trigger appropriate further action

  - returns a boolean

The brain and its areas



Operations

- **Strong projection**: enhancement of the projection operation

Dynamics on the subgraph induced by all disinhibited areas and fibers:

  - **all active assemblies** start projecting in adjacent areas, forming new assemblies, and so on, until stability

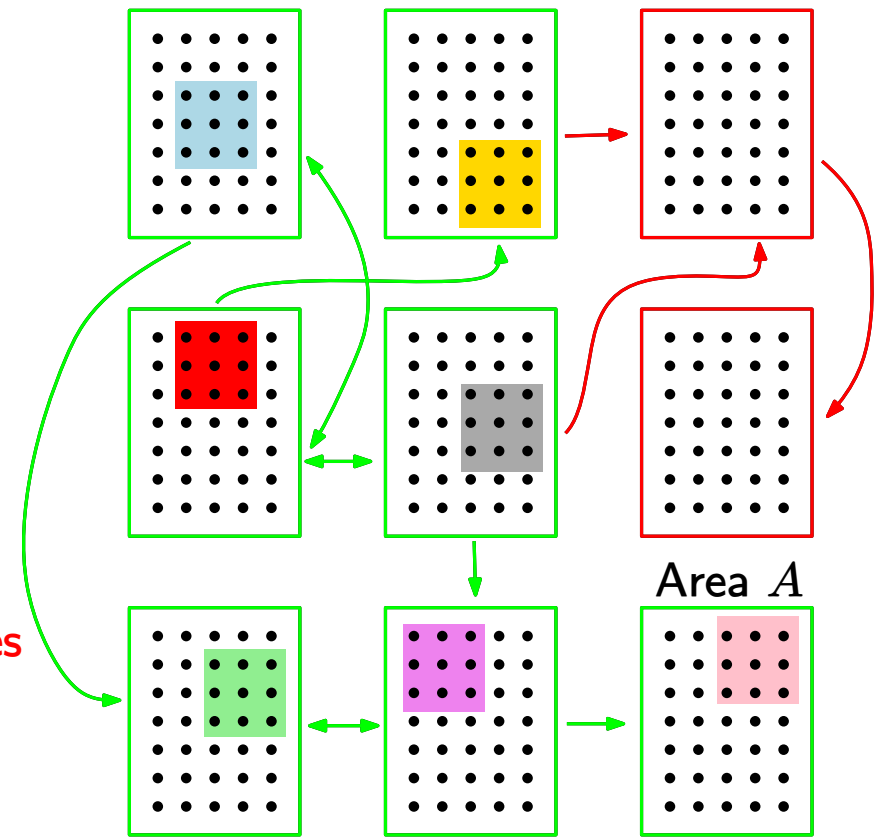
- **Read**: identifies whether in a given area an assembly has fired

[Buzsáki, Neuron 2010] proposes that, for assemblies to be functionally useful, readout mechanisms must exist that sense the current state of the assembly system and trigger appropriate further action

  - returns a boolean

We write `strongProject()` and `read` (A) for a brain area A

The brain and its areas



Related Works

- [Legenstein et al., ITCS 2018] shows analytically how assemblies emerge from stimula

- [Papadimitriou et Vempala, ITCS 2019] analyzes the convergence of processes defined by operations with assemblies

- [Papadimitriou et al., PNAS 2020] introduces the formal model of the assembly calculus and proves it's Turing complete

  - gives a Python code which simulates the model
  - leaves open:
    - language representation
    - planning strategies
    - reasoning
    - learning

Related Works

- [Legenstein et al., ITCS 2018] shows analytically how assemblies emerge from stimula

- [Papadimitriou et Vempala, ITCS 2019] analyzes the convergence of processes defined by operations with assemblies

- [Papadimitriou et al., PNAS 2020] introduces the formal model of the assembly calculus and proves it's Turing complete

  - gives a Python code which simulates the model
  - leaves open:
    - ~~language representation~~
    - planning strategies
    - reasoning
    - learning

- [Mitropolsky et al., TACL 2021; Mitropolsky et al., 2022] deal with language

Related Works

- [Legenstein et al., ITCS 2018] shows analytically how assemblies emerge from stimula

- [Papadimitriou et Vempala, ITCS 2019] analyzes the convergence of processes defined by operations with assemblies

- [Papadimitriou et al., PNAS 2020] introduces the formal model of the assembly calculus and proves it's Turing complete

  - gives a Python code which simulates the model
  - leaves open:
    - ~~language representation~~
    - planning strategies
    - reasoning
    - ~~learning~~

- [Mitropolsky et al., TACL 2021; Mitropolsky et al., 2022] deal with language

- [Dabagia et al., COLT 2022] deal with learning

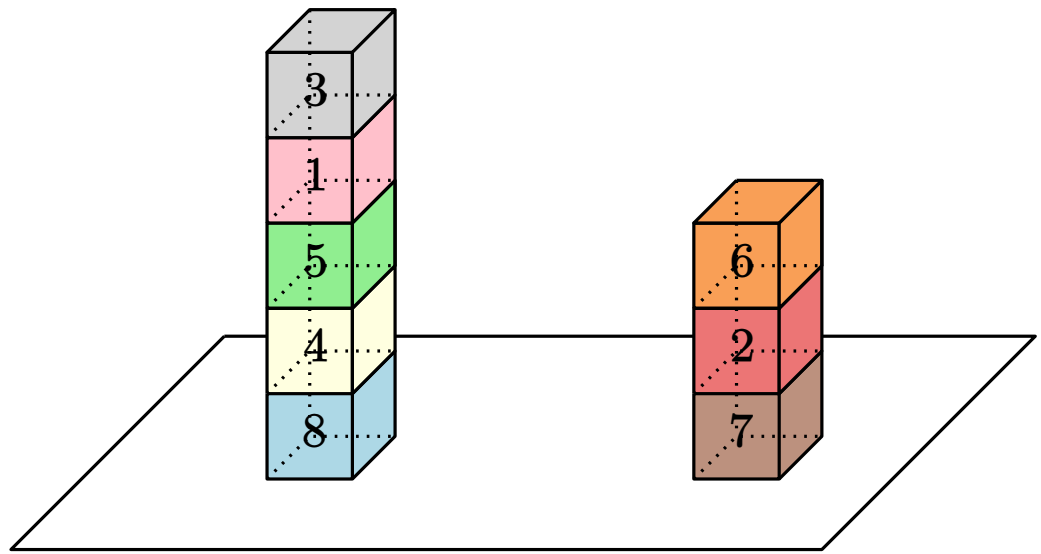
Our Contribution

We employ the assembly calculus to implement (through neurons and synapses) planning strategies in the blocks world (**Julia** programming language)

Our Contribution

We employ the assembly calculus to implement (through neurons and synapses) planning strategies in the blocks world (**Julia** programming language)

Blocks world: n unique blocks labelled in $\{1, \ldots, n\}$, organized as follows

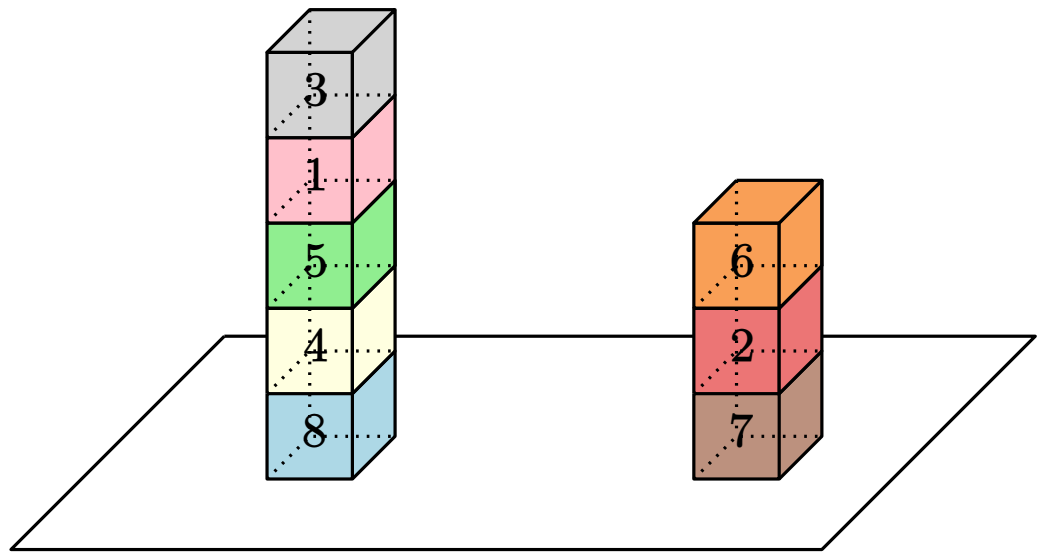


Our Contribution

We employ the assembly calculus to implement (through neurons and synapses) planning strategies in the blocks world (**Julia** programming language)

Blocks world: n unique blocks labelled in $\{1, \ldots, n\}$, organized as follows

- set of stacks $\{S_i\}_i$
- each stack is a sequence of blocks $S_i = (b_1^{(i)}, \ldots, b_{k_i}^{(i)})$
- $\cup_i \{b_1^{(i)}, \ldots, b_{k_i}^{(i)}\} = \{1, \ldots, n\}$



The Planning Task

Possible moves:

- move a block from the top of a stack to the table
- move a block from the top of a stack to the top of another stack
- move a block from the table to the top of a stack

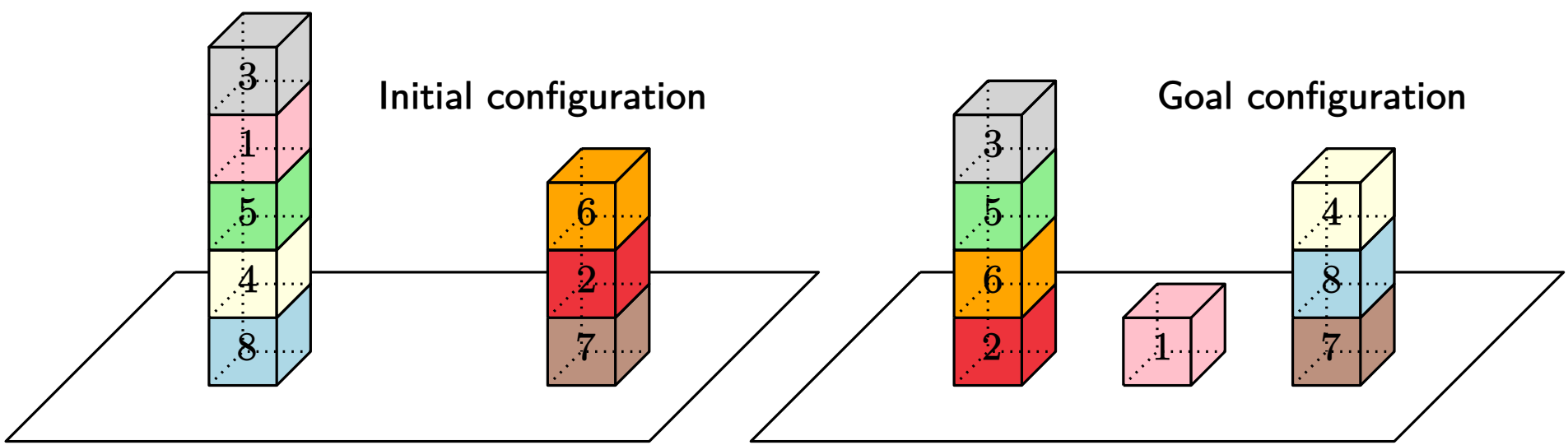
The Planning Task

Possible moves:

- move a block from the top of a stack to the table
- move a block from the top of a stack to the top of another stack
- move a block from the table to the top of a stack

Input:

- set of initial stacks $\{S_i^{(in)}\}_i$, $S_i^{(in)} = (b_1^{(i)}, \ldots, b_{k_i}^{(i)})$
- $\cup_i \{b_1^{(i)}, \ldots, b_{k_i}^{(i)}\} = \{1, \ldots, n\}$
- set of goal stacks $\{S_i^{(goal)}\}_i$, $S_i^{(goal)} = (c_1^{(i)}, \ldots, c_{k_i}^{(i)})$
- $\cup_i \{c_1^{(i)}, \ldots, c_{k_i}^{(i)}\} = \{1, \ldots, n\}$



The Planning Task

Possible moves:

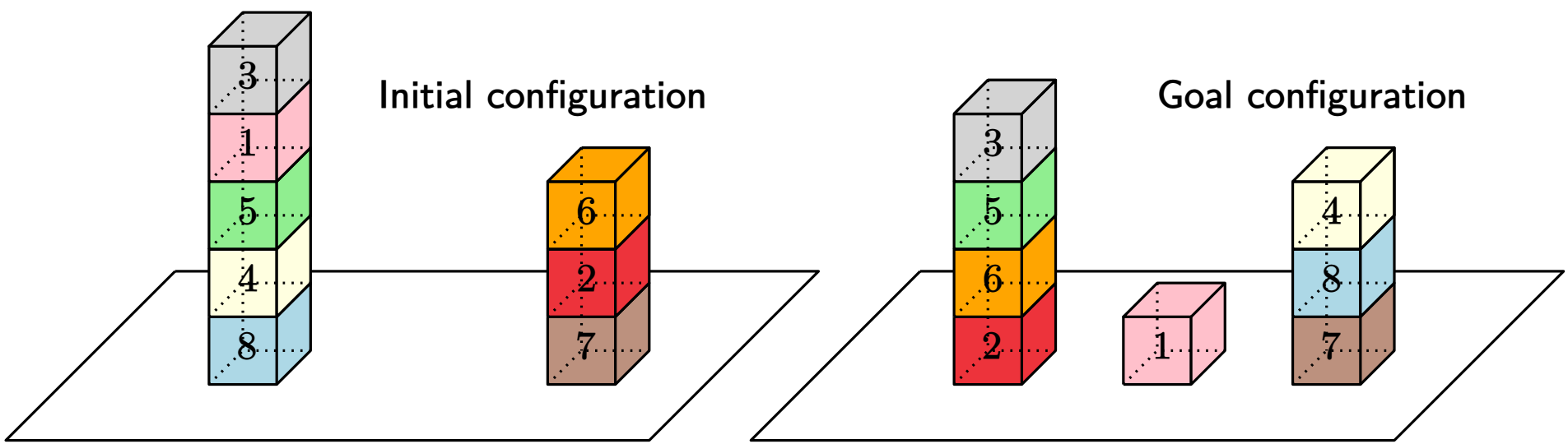
- move a block from the top of a stack to the table
- move a block from the top of a stack to the top of another stack
- move a block from the table to the top of a stack

Input:

- set of initial stacks $\{S_i^{(in)}\}_i$, $S_i^{(in)} = (b_1^{(i)}, \ldots, b_{k_i}^{(i)})$
- $\cup_i \{b_1^{(i)}, \ldots, b_{k_i}^{(i)}\} = \{1, \ldots, n\}$
- set of goal stacks $\{S_i^{(goal)}\}_i$, $S_i^{(goal)} = (c_1^{(i)}, \ldots, c_{k_i}^{(i)})$
- $\cup_i \{c_1^{(i)}, \ldots, c_{k_i}^{(i)}\} = \{1, \ldots, n\}$

Output:

- the sequence of moves



Strategies

Brute-force strategy: *move all blocks to the table and place them correctly, one by one*

Strategies

Brute-force strategy: *move all blocks to the table and place them correctly, one by one*

[Gupta et Nau, AI 1992] proves the problem of finding the optimum is NP-complete, and provides a 2-apx algorithm

> *Move each block which is not in its final position to the table, and then place the blocks (which are on the table) correctly one by one*

Strategies

Brute-force strategy: *move all blocks to the table and place them correctly, one by one*

[Gupta et Nau, AI 1992] proves the problem of finding the optimum is NP-complete, and provides a 2-apx algorithm

> *Move each block which is not in its final position to the table, and then place the blocks (which are on the table) correctly one by one*

We implement both of them...

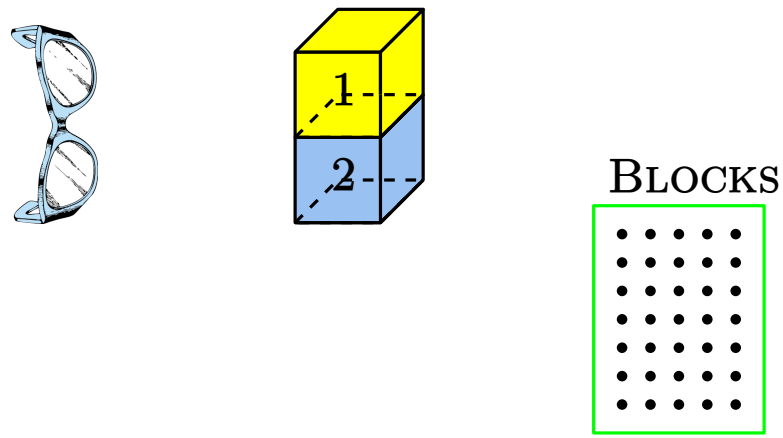
Back to the Brain: Assumptions

The individual "scans" the block world configuration by processing each stack from the top block to the bottom block, one at a time

Back to the Brain: Assumptions

The individual "scans" the block world configuration by processing each stack from the top block to the bottom block, one at a time

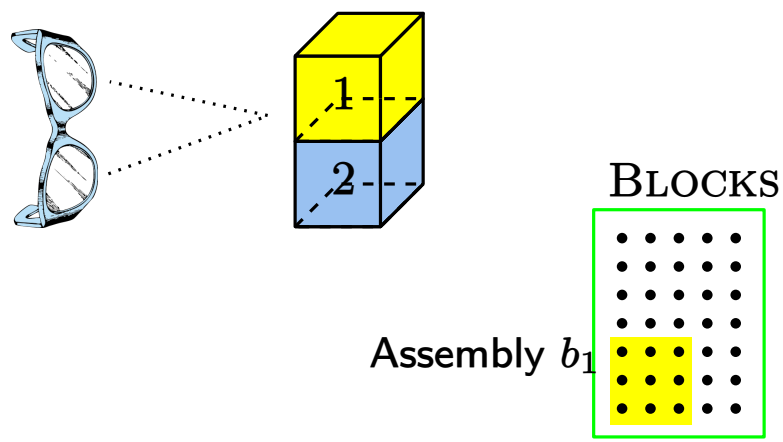
The scanning of each block activates an unique corresponding assembly in the brain area BLOCKS (if disinhibited)



Back to the Brain: Assumptions

The individual "scans" the block world configuration by processing each stack from the top block to the bottom block, one at a time

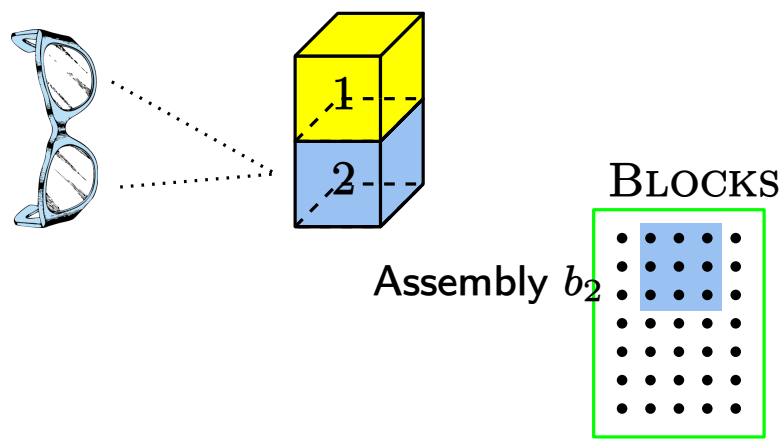
The scanning of each block activates an unique corresponding assembly in the brain area BLOCKS (if disinhibited)



Back to the Brain: Assumptions

The individual "scans" the block world configuration by processing each stack from the top block to the bottom block, one at a time

The scanning of each block activates an unique corresponding assembly in the brain area BLOCKS (if disinhibited)

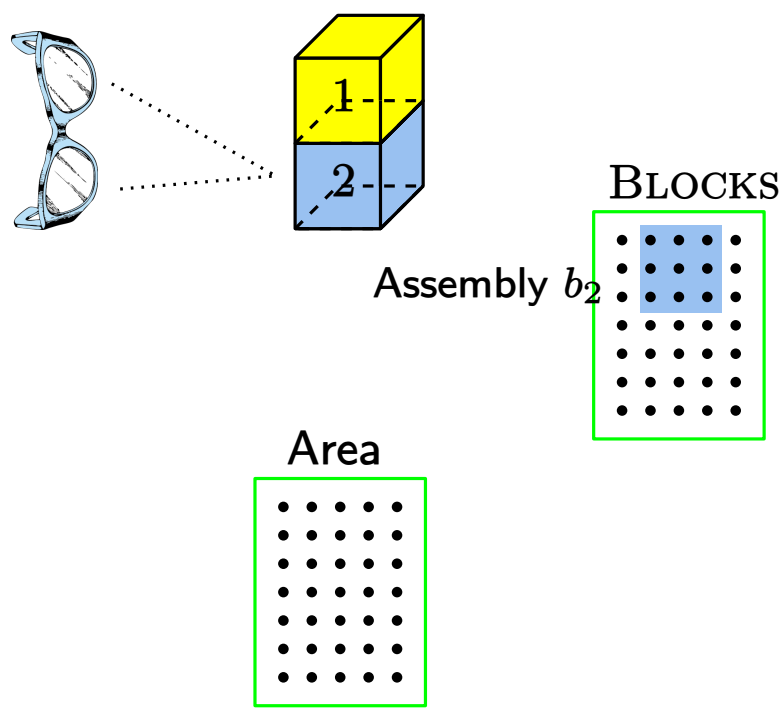


Back to the Brain: Assumptions

The individual "scans" the block world configuration by processing each stack from the top block to the bottom block, one at a time

The scanning of each block activates an unique corresponding assembly in the brain area BLOCKS (if disinhibited)

Before and after each activation, the brain makes some operations, such as projection, inhibition, disinhibition

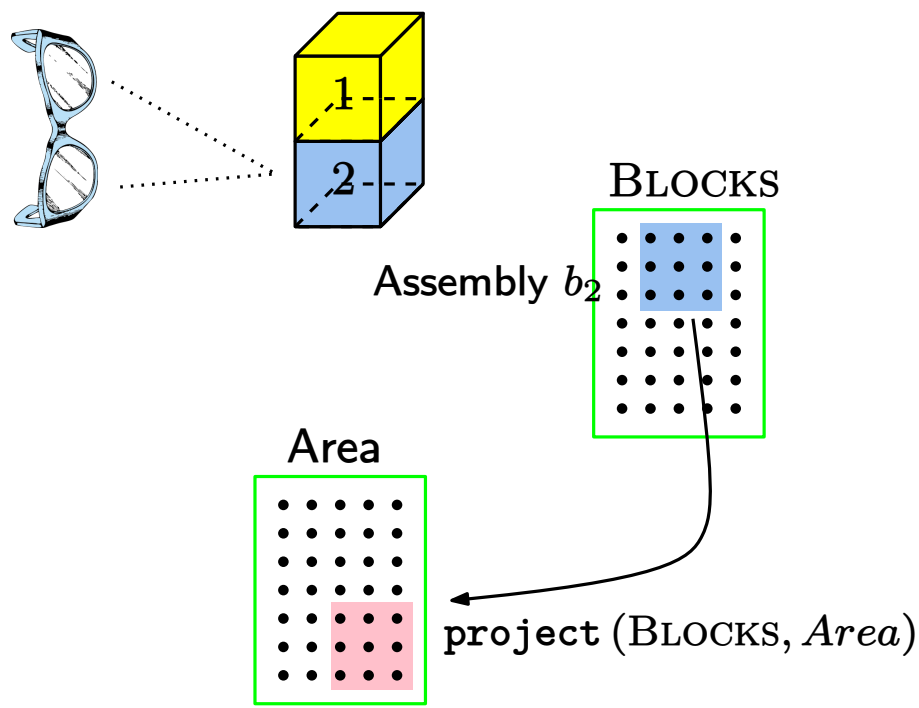


Back to the Brain: Assumptions

The individual "scans" the block world configuration by processing each stack from the top block to the bottom block, one at a time

The scanning of each block activates an unique corresponding assembly in the brain area BLOCKS (if disinhibited)

Before and after each activation, the brain makes some operations, such as projection, inhibition, disinhibition

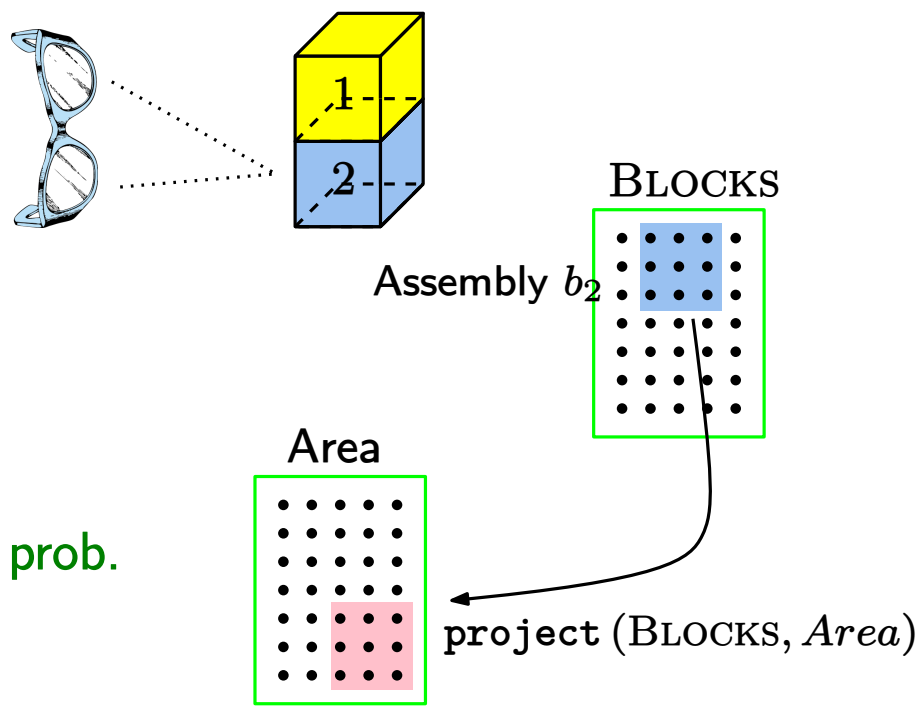


Back to the Brain: Assumptions

The individual "scans" the block world configuration by processing each stack from the top block to the bottom block, one at a time

The scanning of each block activates an unique corresponding assembly in the brain area BLOCKS (if disinhibited)

Before and after each activation, the brain makes some operations, such as projection, inhibition, disinhibition

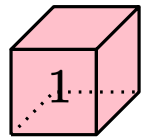


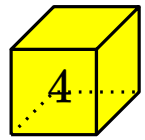
Each area has n neurons, the same Erdös-Renyi prob. p, the same k for the **k-CAP**

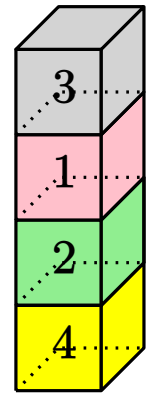
A Toy Example

Given a stack of blocks and two of its blocks a, b, is a **above** *b?*

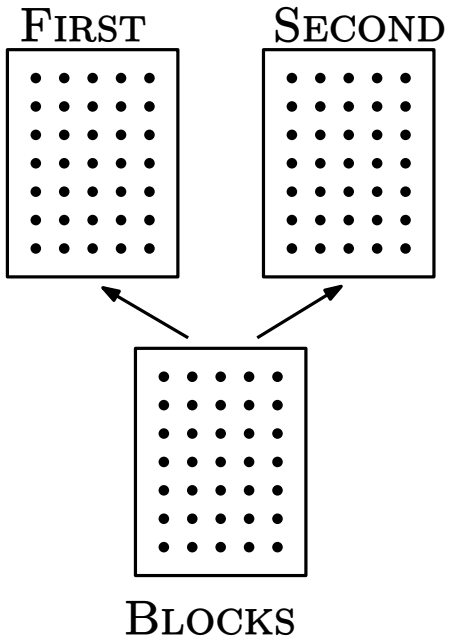
E.g., is block 1 above block 4?







Assumptions: 3 brain areas, BLOCKS, FIRST, SECOND



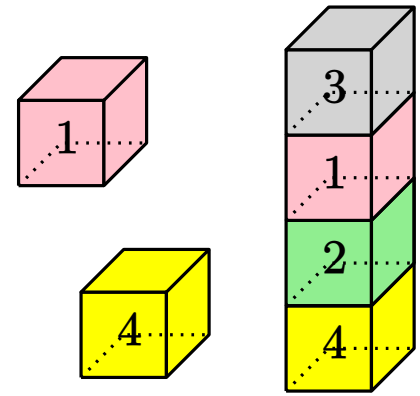
IsAbove Algorithm

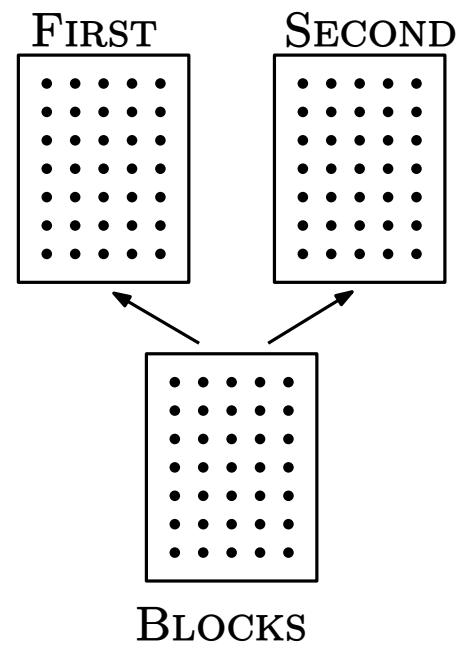
Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?





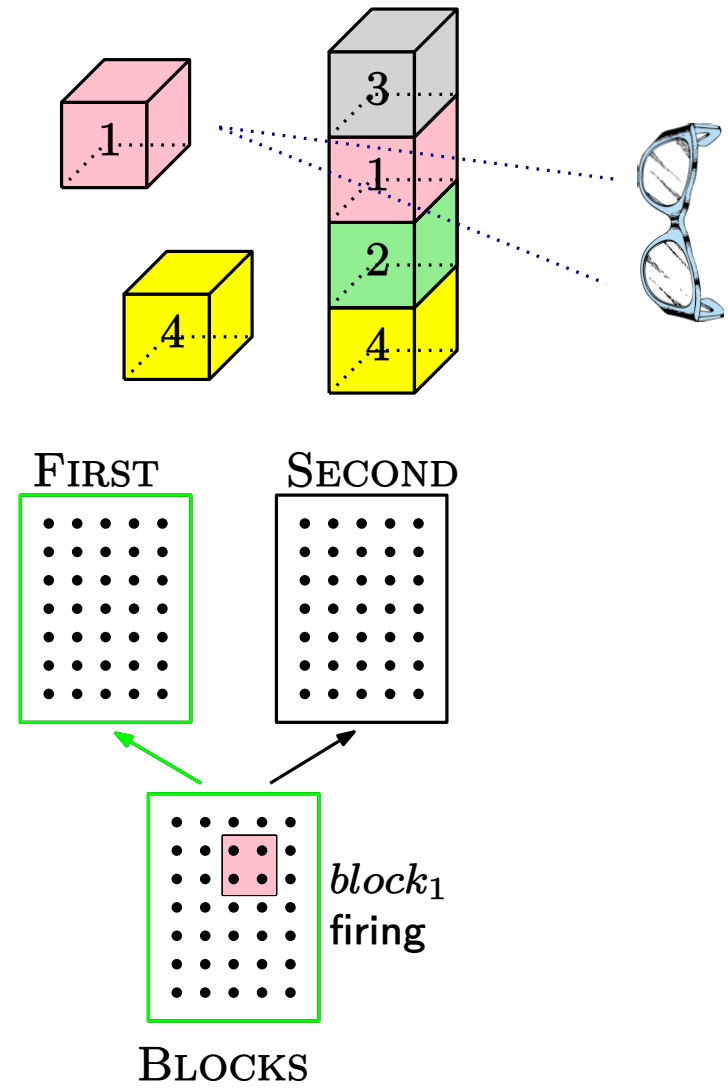
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



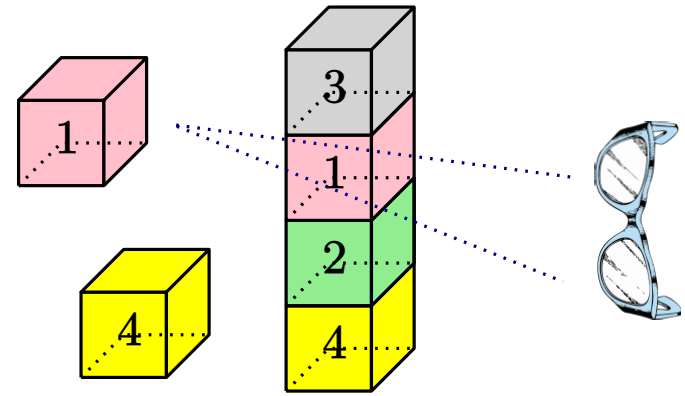
IsAbove Algorithm

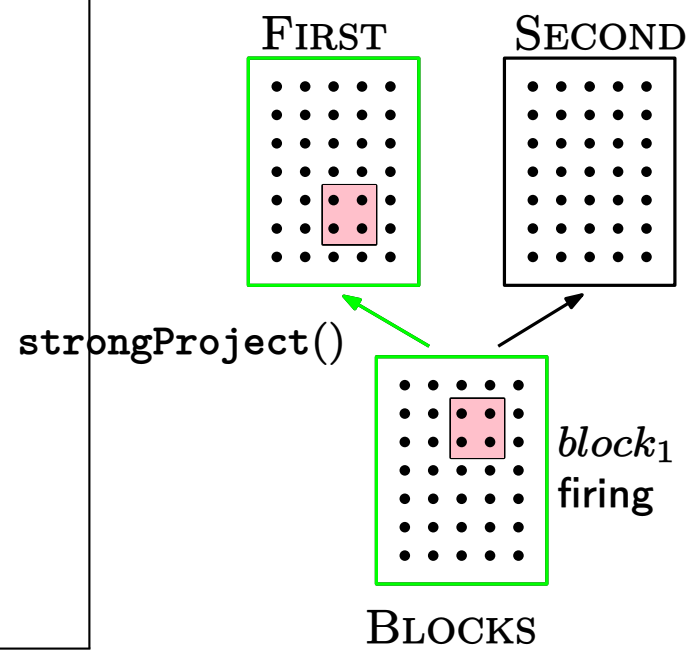
Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?





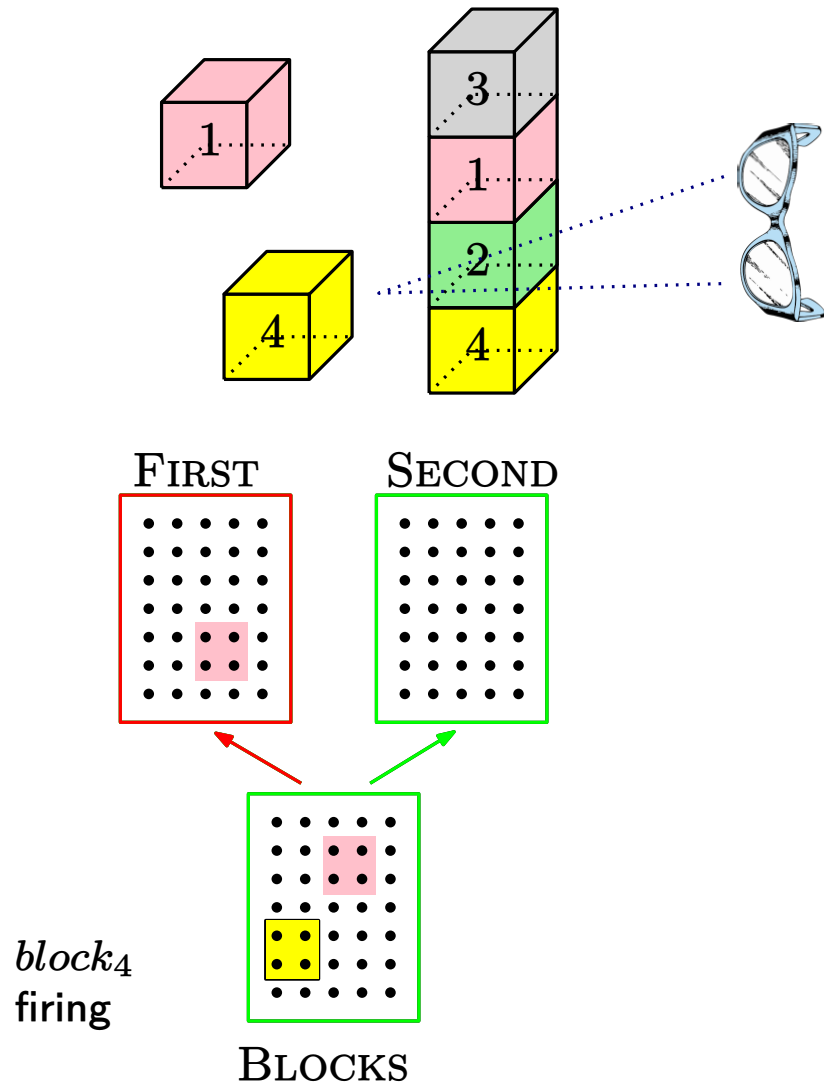
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



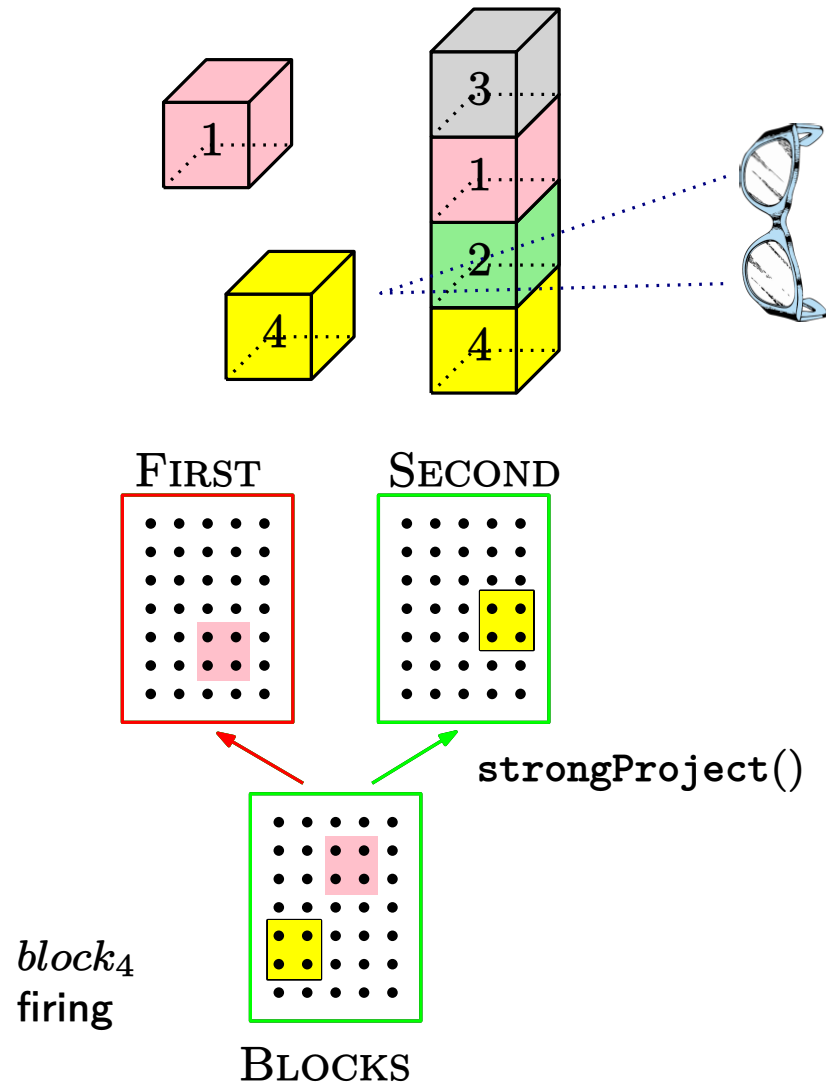
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea ({Blocks, First});
2  disinhibitFiber ({(Blocks, First)});
3  fire (x);
4  strongProject();
5  inhibitArea ({First});
6  disinhibitArea ({Second});
7  disinhibitFiber ({(Blocks, Second)});
8  fire (y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea ({Second});
12     disinhibitArea ({First});
13     fire (b_i);
14     if read (First) then return true ;
15     inhibitArea ({First});
16     disinhibitArea ({Second});
17     fire (b_i);
18     if read (Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



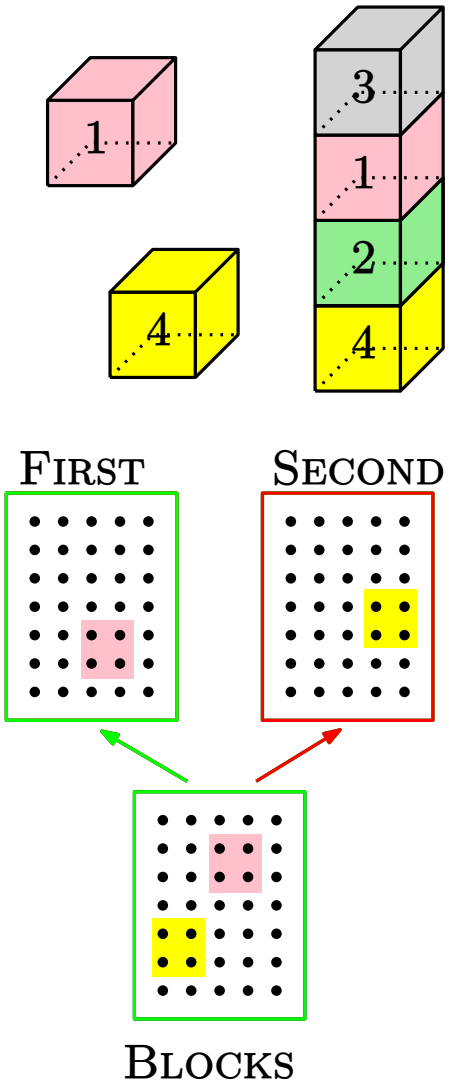
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



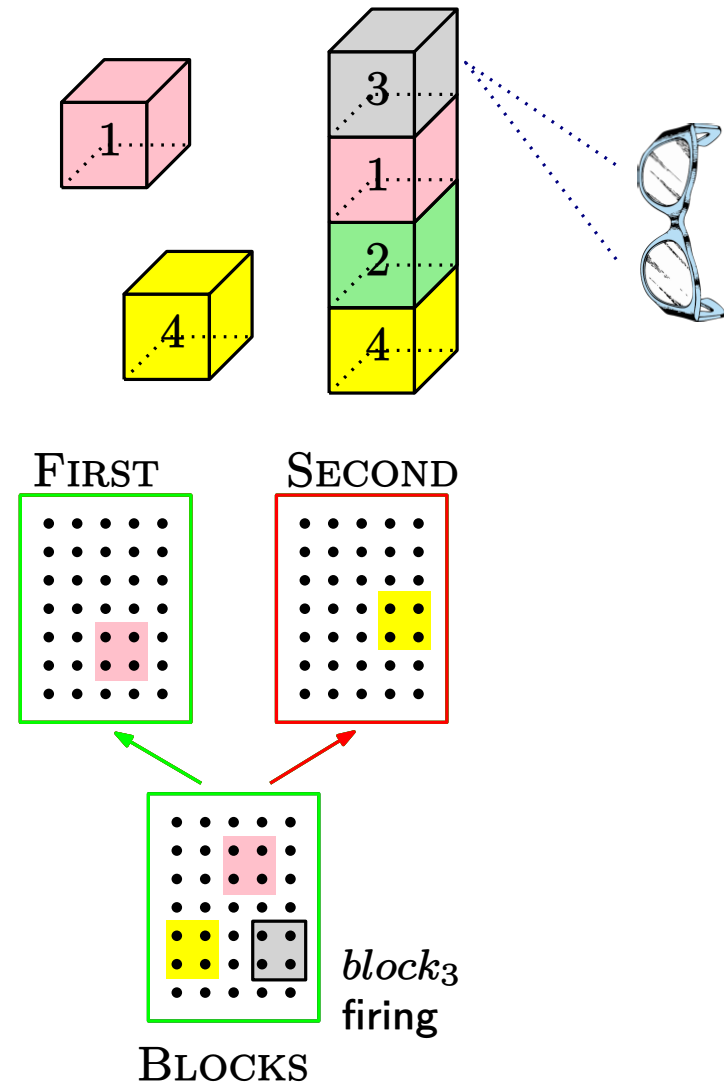
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



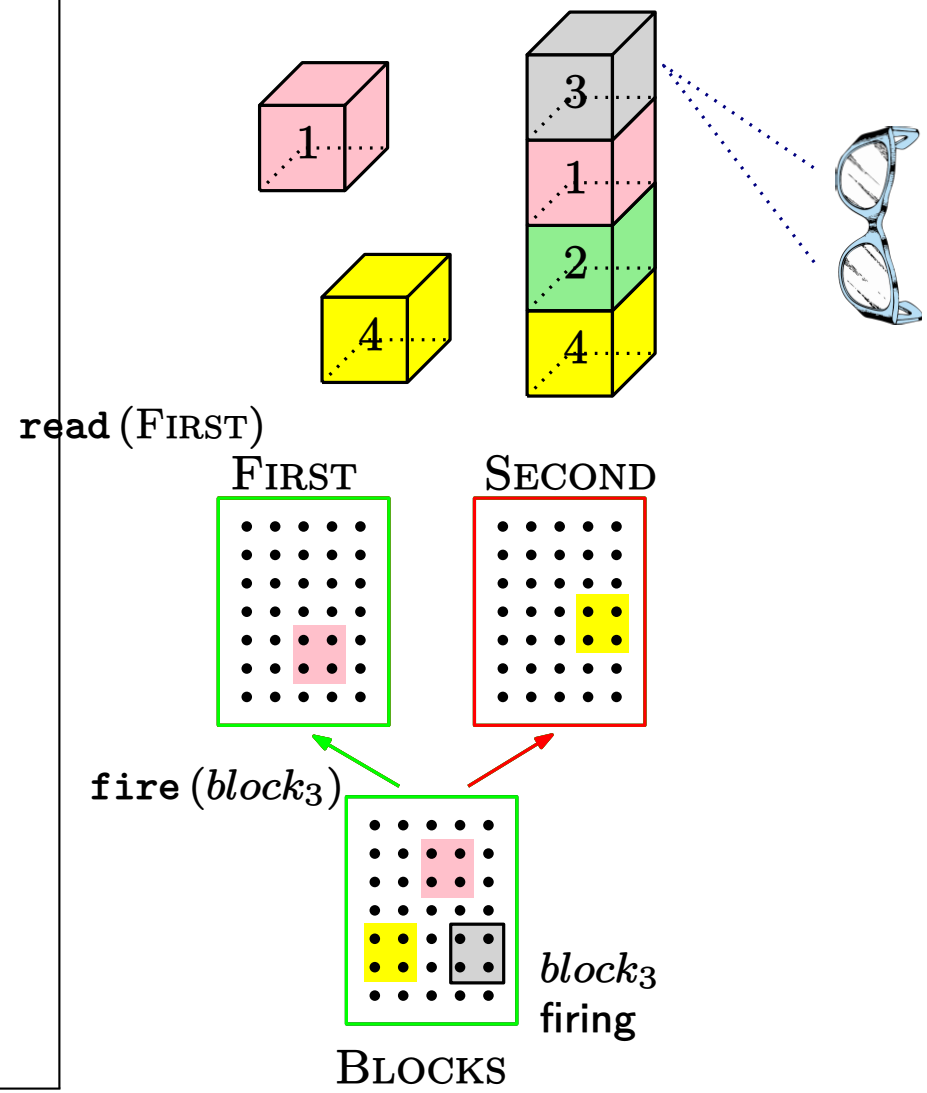
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea ({Blocks, First});
2  disinhibitFiber ({(Blocks, First)});
3  fire (x);
4  strongProject();
5  inhibitArea ({First});
6  disinhibitArea ({Second});
7  disinhibitFiber ({(Blocks, Second)});
8  fire (y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea ({Second});
12     disinhibitArea ({First});
13     fire (b_i);
14     if read (First) then return true ;
15     inhibitArea ({First});
16     disinhibitArea ({Second});
17     fire (b_i);
18     if read (Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



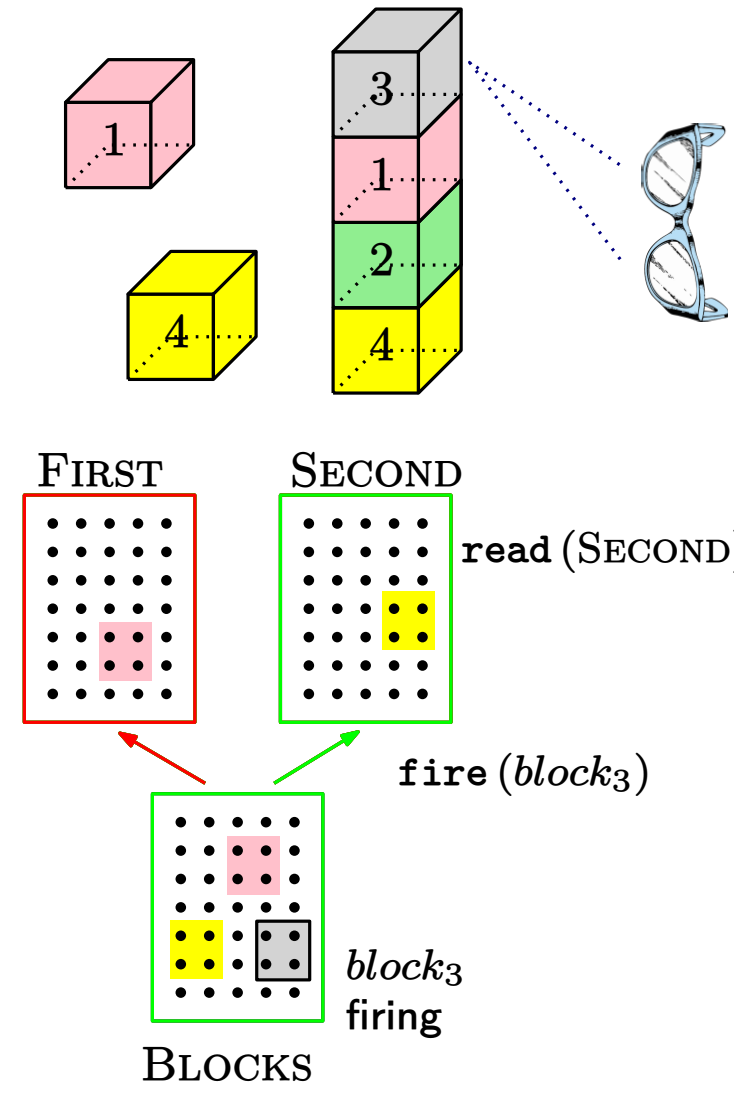
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



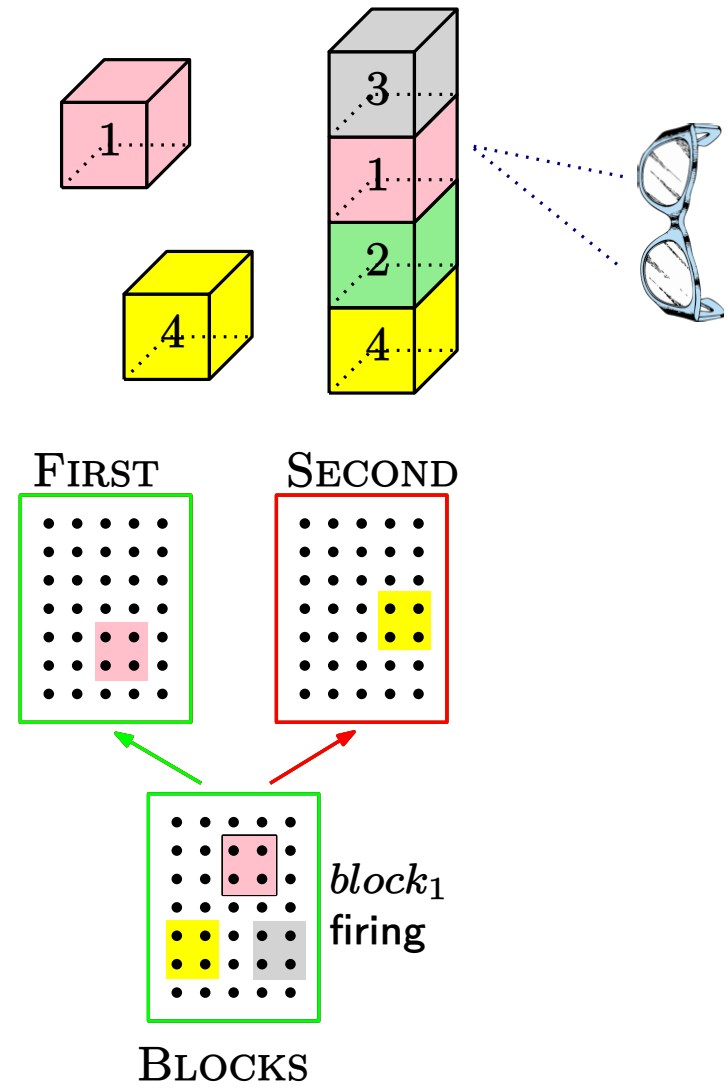
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



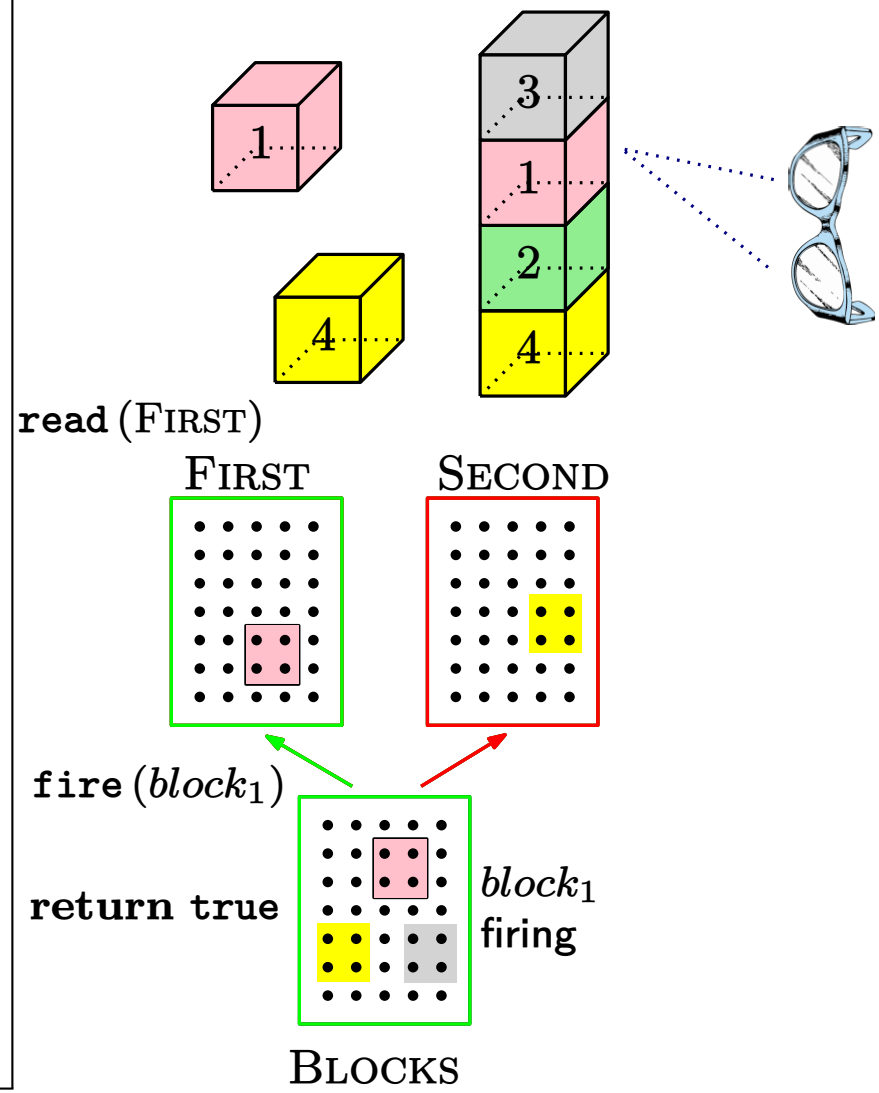
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



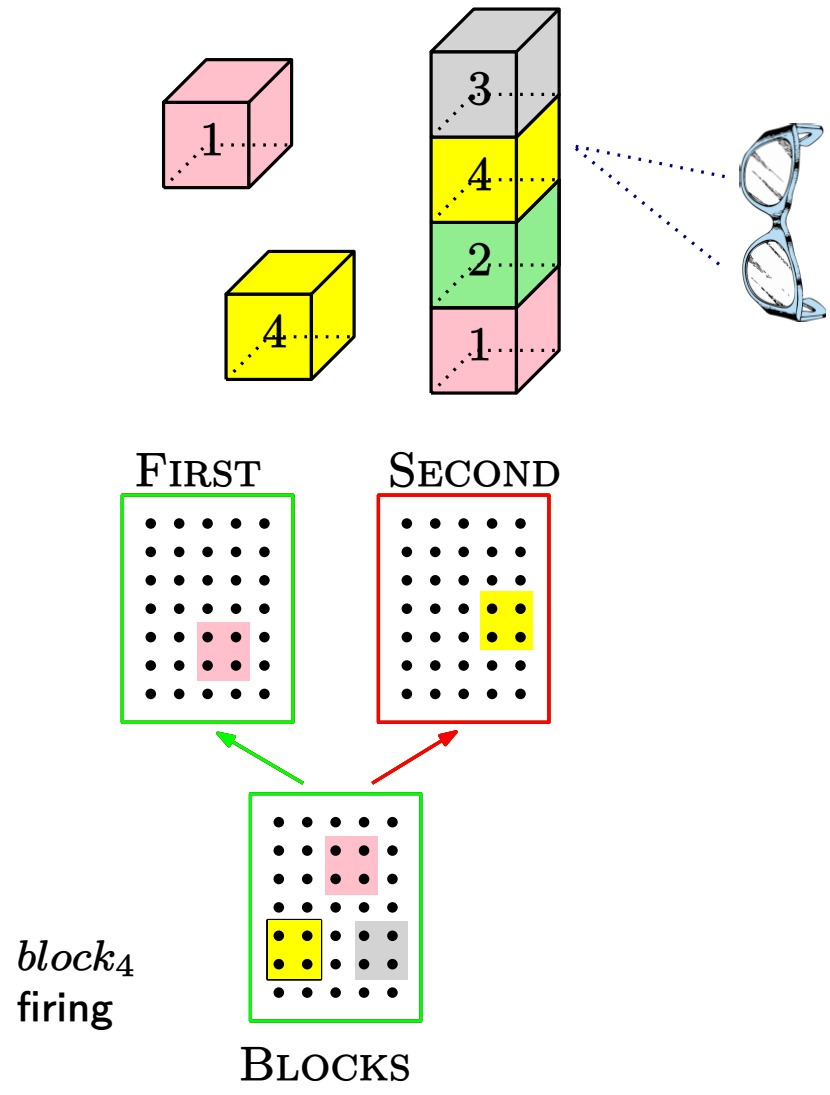
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



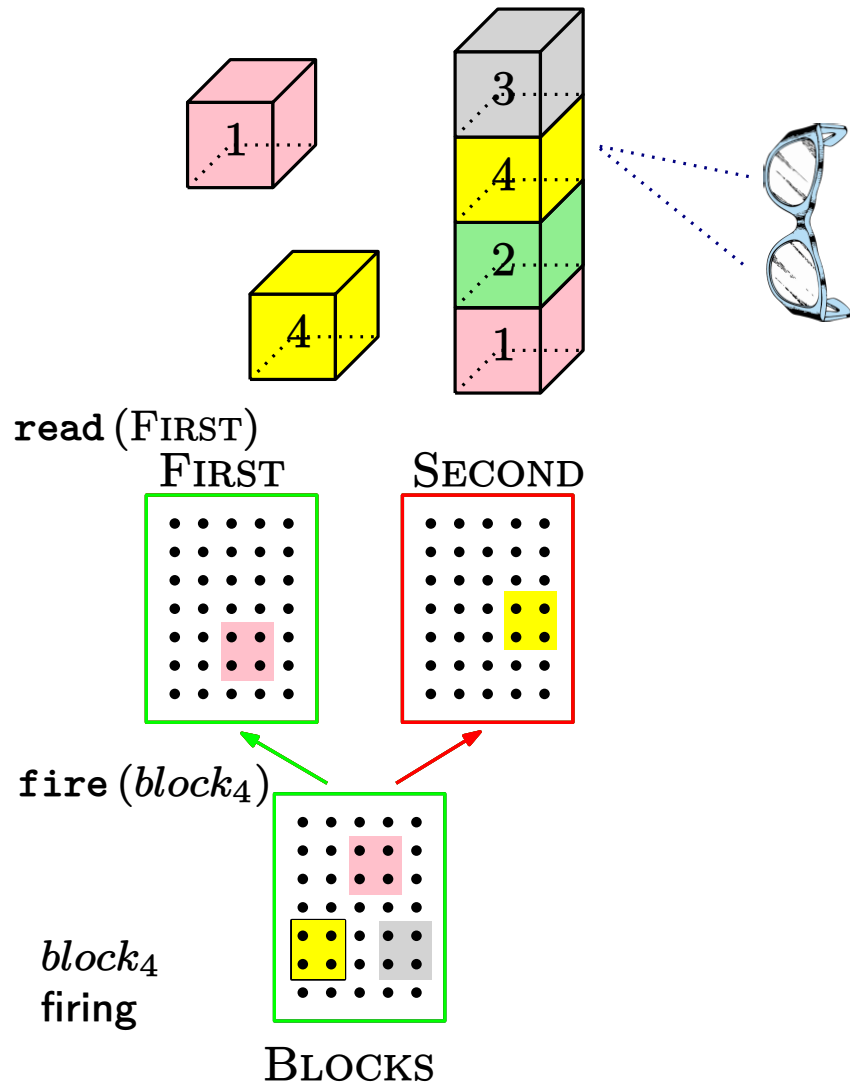
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



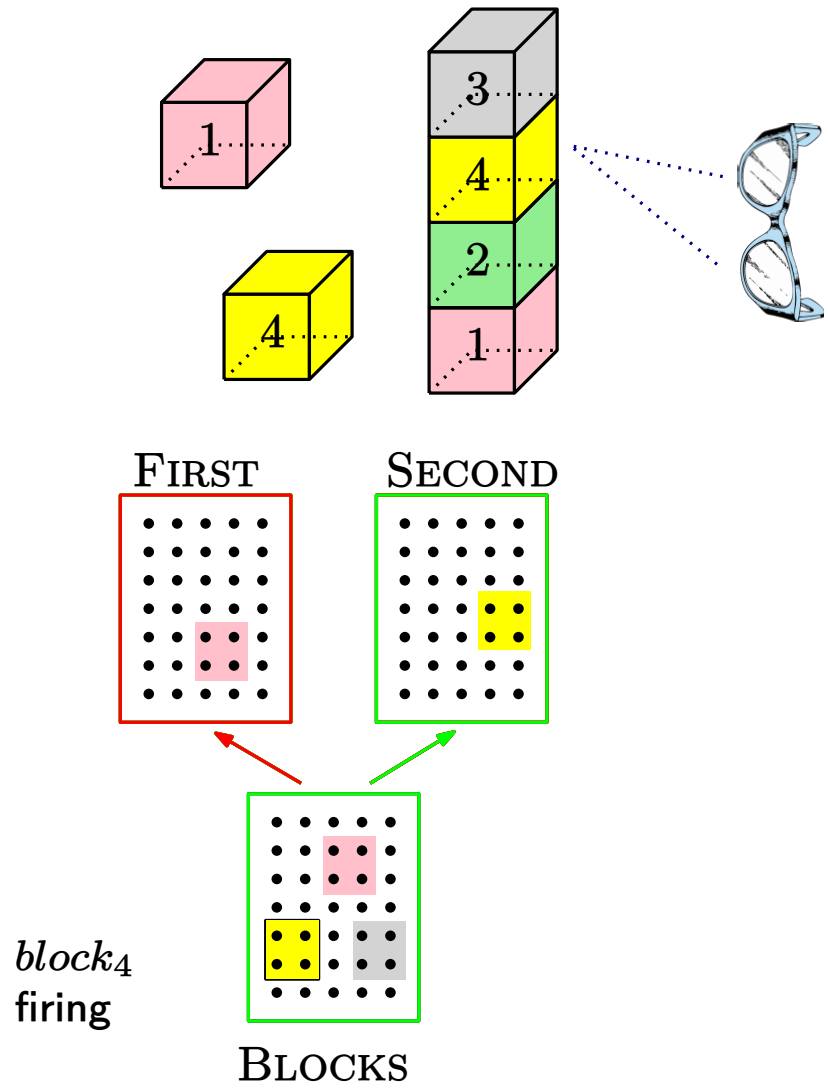
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



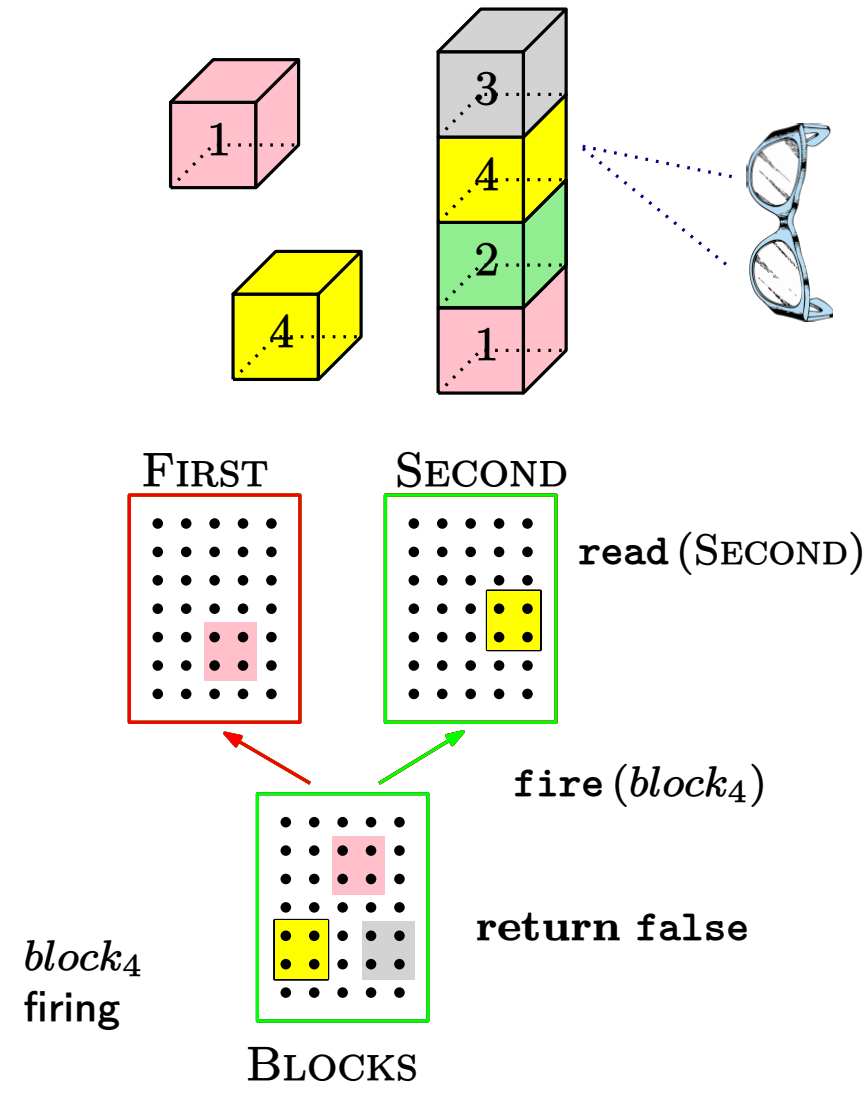
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea ({Blocks, First});
2  disinhibitFiber ({(Blocks, First)});
3  fire (x);
4  strongProject();
5  inhibitArea ({First});
6  disinhibitArea ({Second});
7  disinhibitFiber ({(Blocks, Second)});
8  fire (y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea ({Second});
12     disinhibitArea ({First});
13     fire (b_i);
14     if read (First) then return true ;
15     inhibitArea ({First});
16     disinhibitArea ({Second});
17     fire (b_i);
18     if read (Second) then return false ;
19 end
```

E.g., is block 1 above block 4?



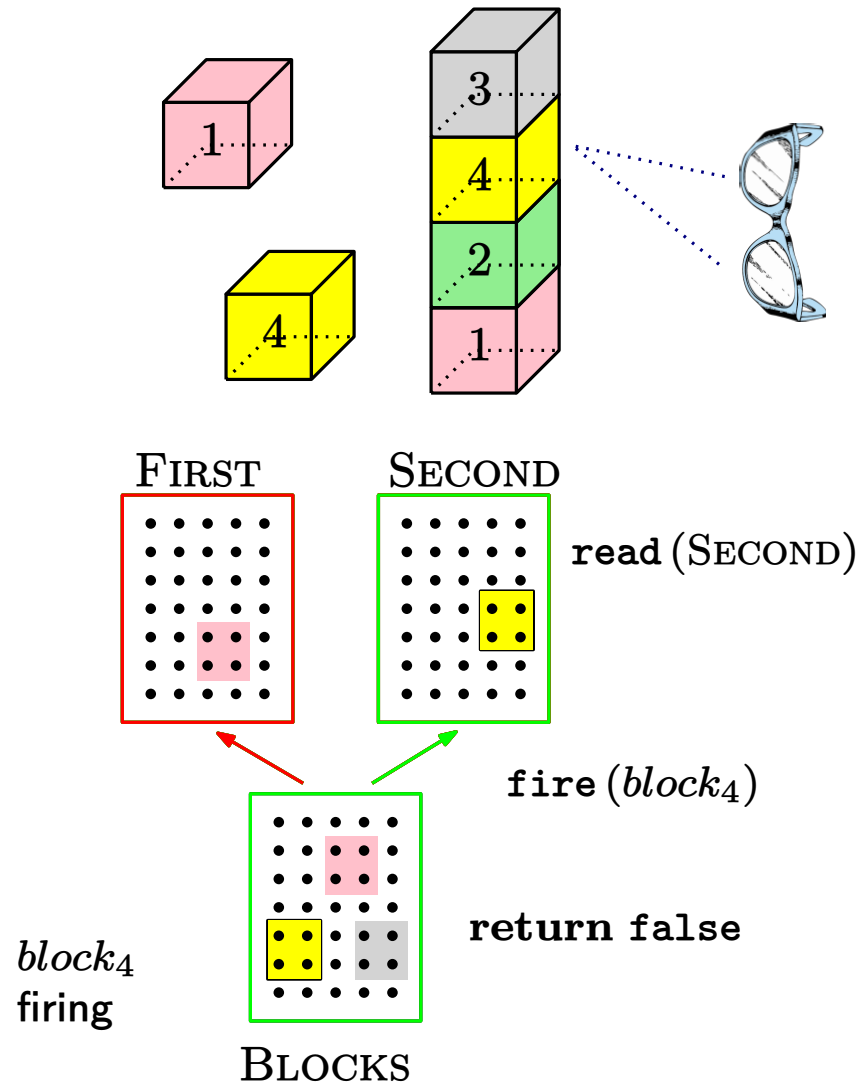
IsAbove Algorithm

Algorithm 1: IsAbove (S, x, y)

input: a stack S of blocks $b_1, b_2, \ldots, b_s$;
two blocks x, y with $x, y \in S$.

```
1  disinhibitArea({Blocks, First});
2  disinhibitFiber({(Blocks, First)});
3  fire(x);
4  strongProject();
5  inhibitArea({First});
6  disinhibitArea({Second});
7  disinhibitFiber({(Blocks, Second)});
8  fire(y);
9  strongProject();
10 foreach i with 2 ≤ i ≤ s do
11     inhibitArea({Second});
12     disinhibitArea({First});
13     fire(b_i);
14     if read(First) then return true ;
15     inhibitArea({First});
16     disinhibitArea({Second});
17     fire(b_i);
18     if read(Second) then return false ;
19 end
```

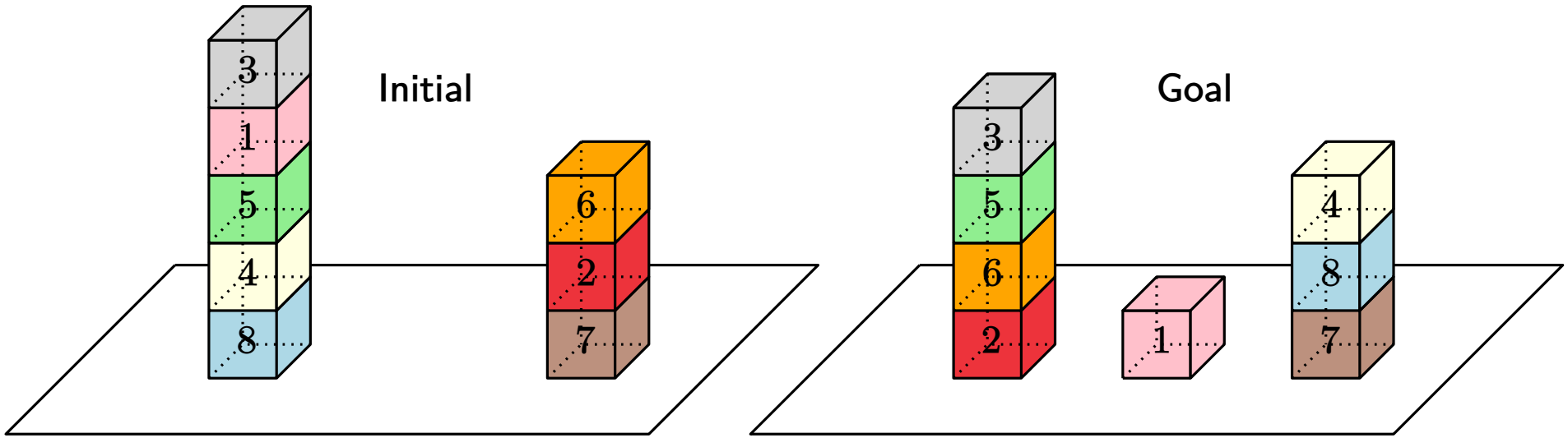
E.g., is block 1 above block 4?



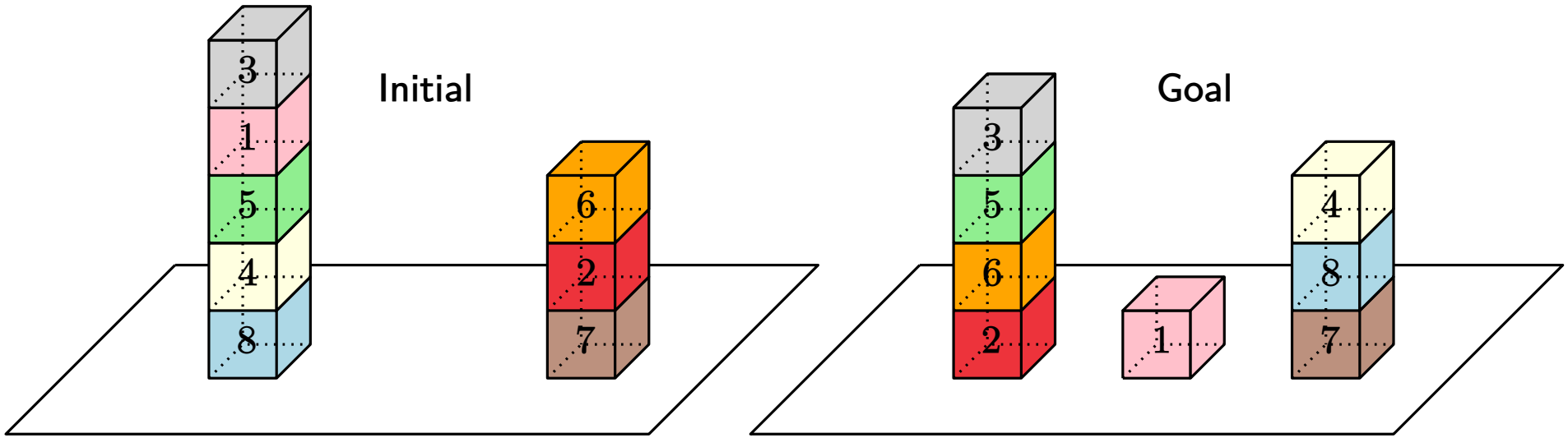
How does the brain understand a positive (negative) answer?

The answer corresponds to an assembly activating in a particular brain area

Back to the Planning Problem

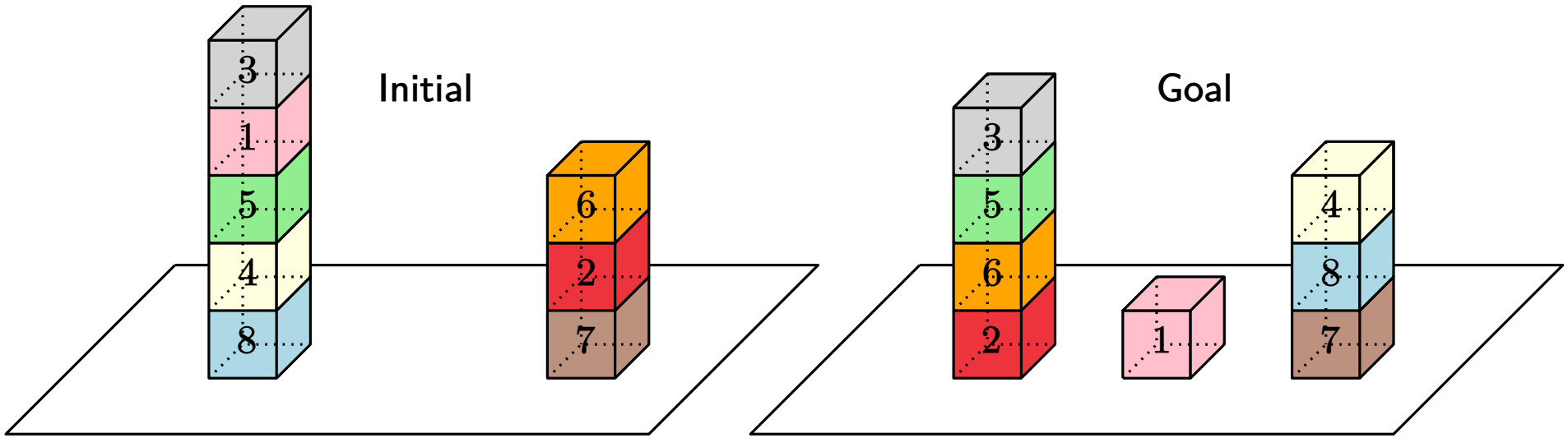


Back to the Planning Problem



- **First**: way to represent the initial and the goal stacks in the brain

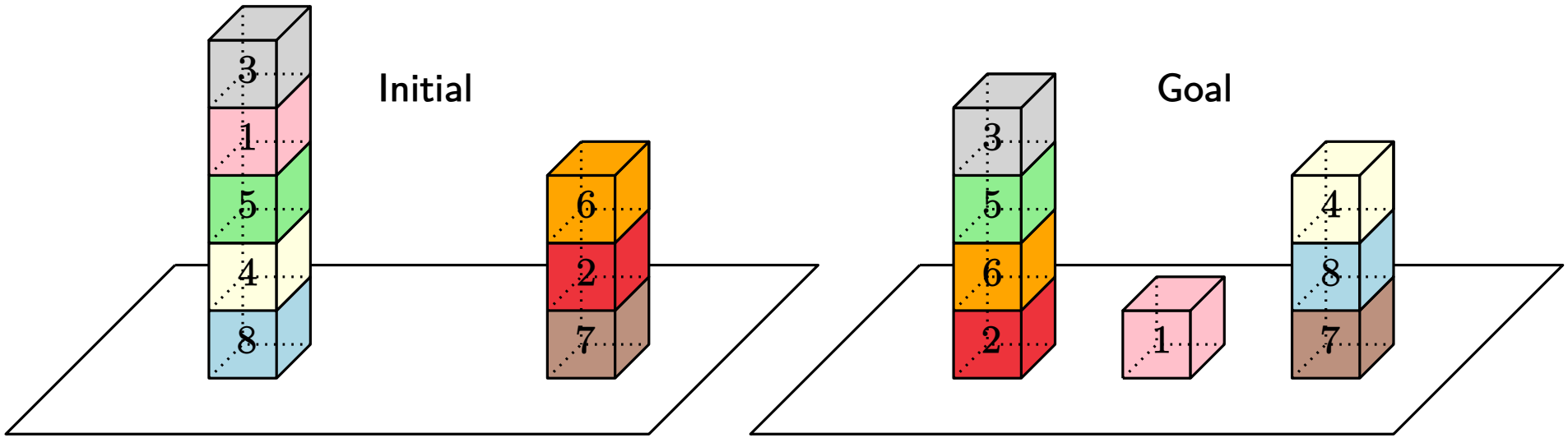
Back to the Planning Problem



- **First**: way to represent the initial and the goal stacks in the brain

A Parser

Back to the Planning Problem



- **First**: way to represent the initial and the goal stacks in the brain

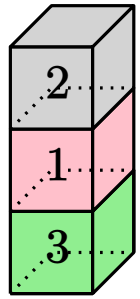
  A Parser

- **Second**: way to implement actions and represent them

The Parser

Assumptions:

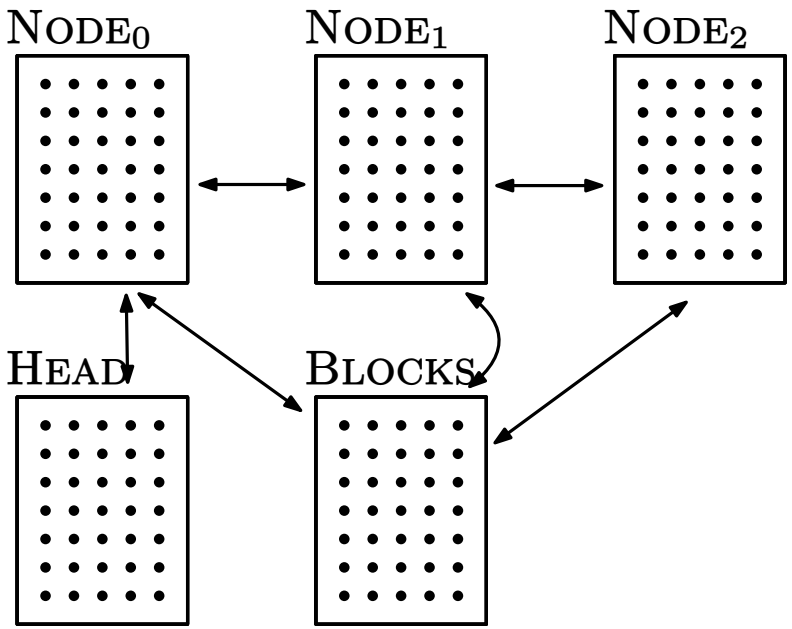
- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE_0, NODE_1, NODE_2, HEAD

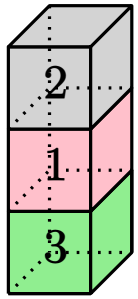


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

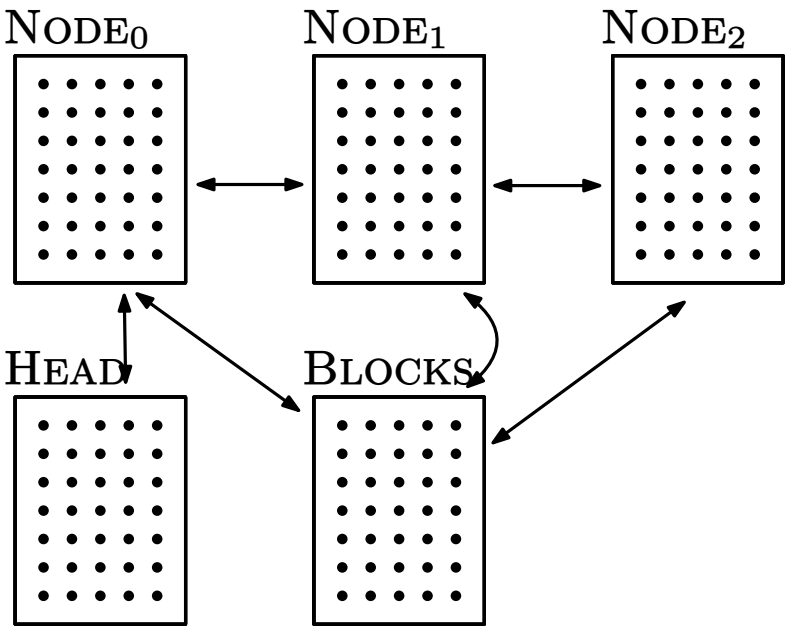


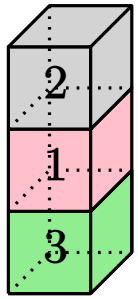


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

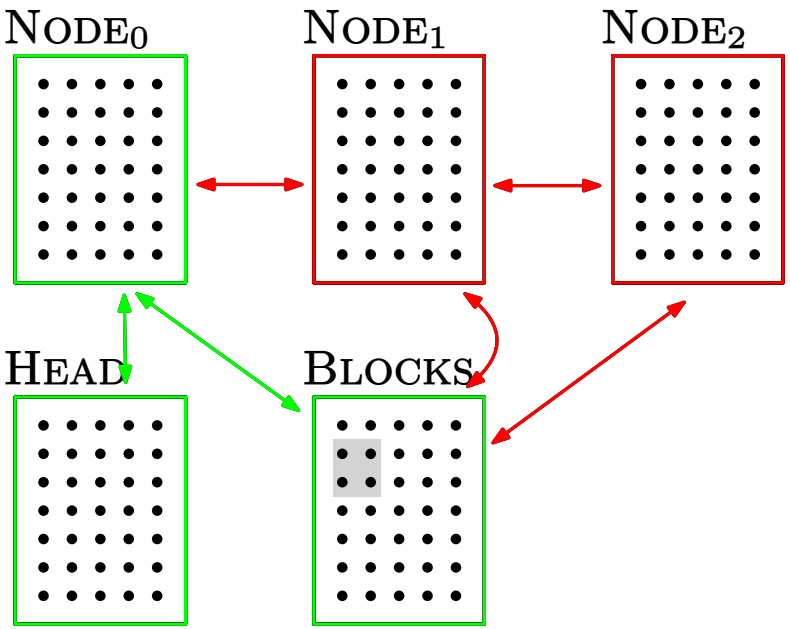


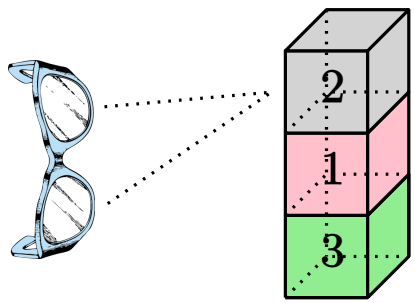


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

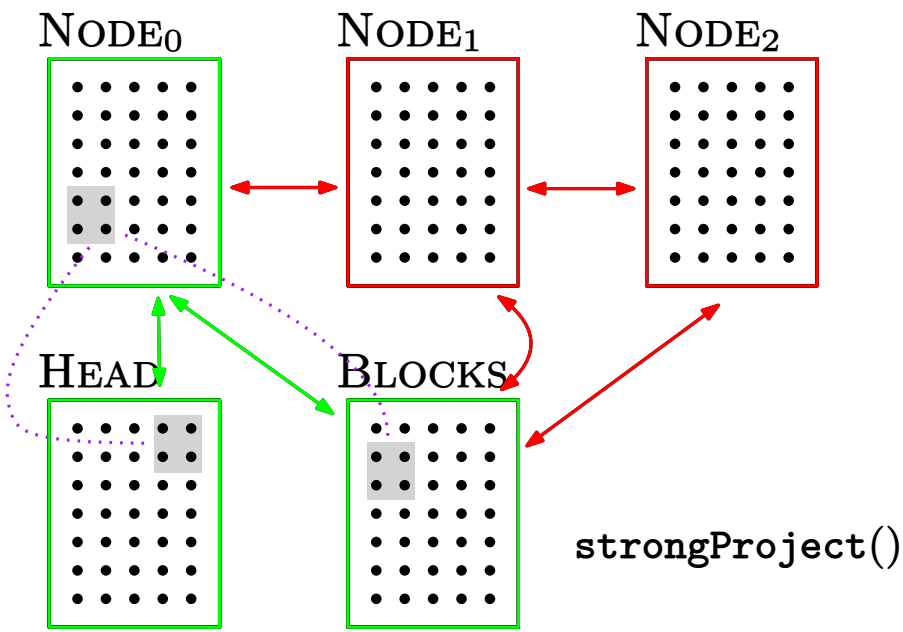


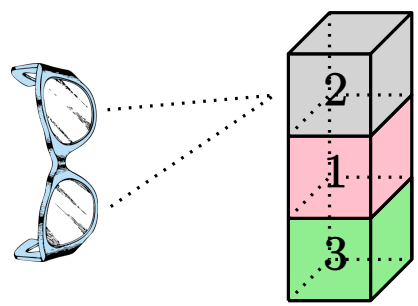


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

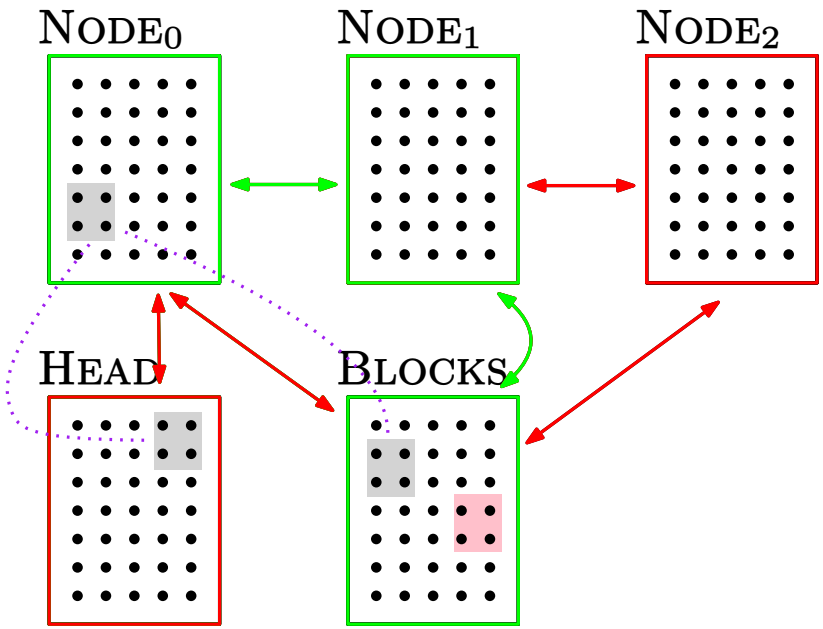


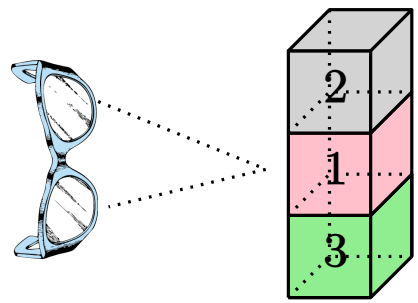


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

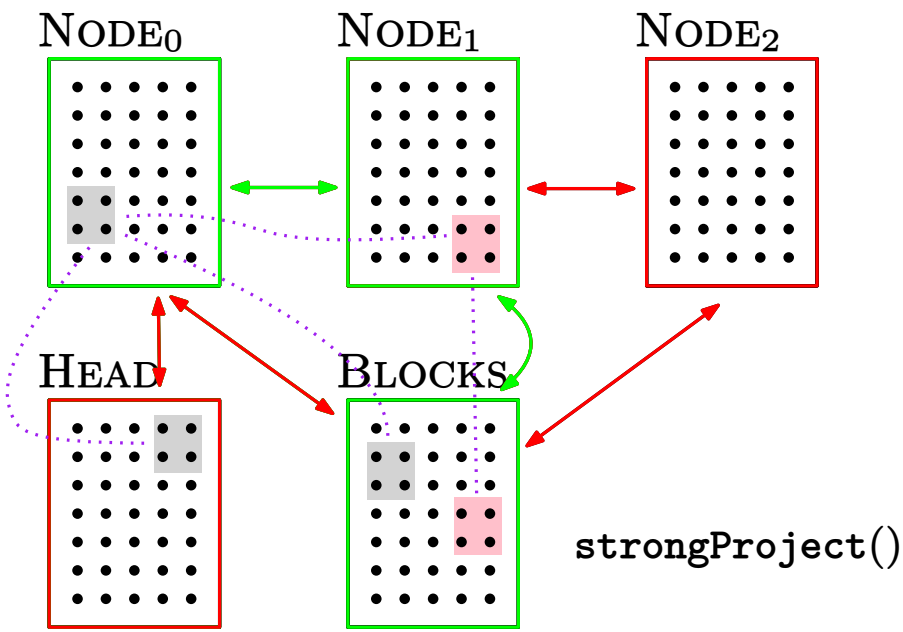


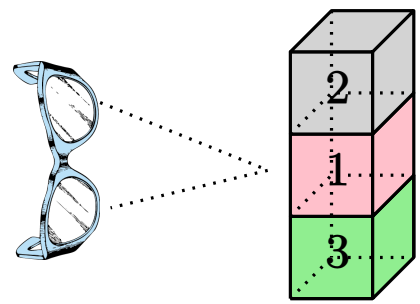


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

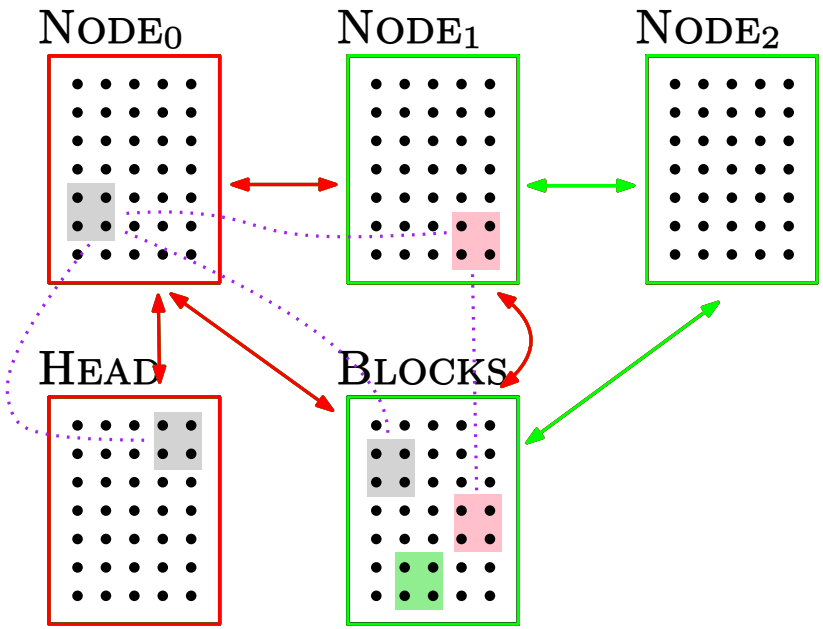


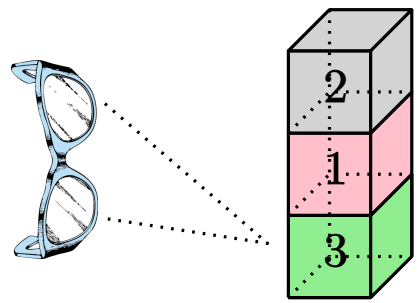


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

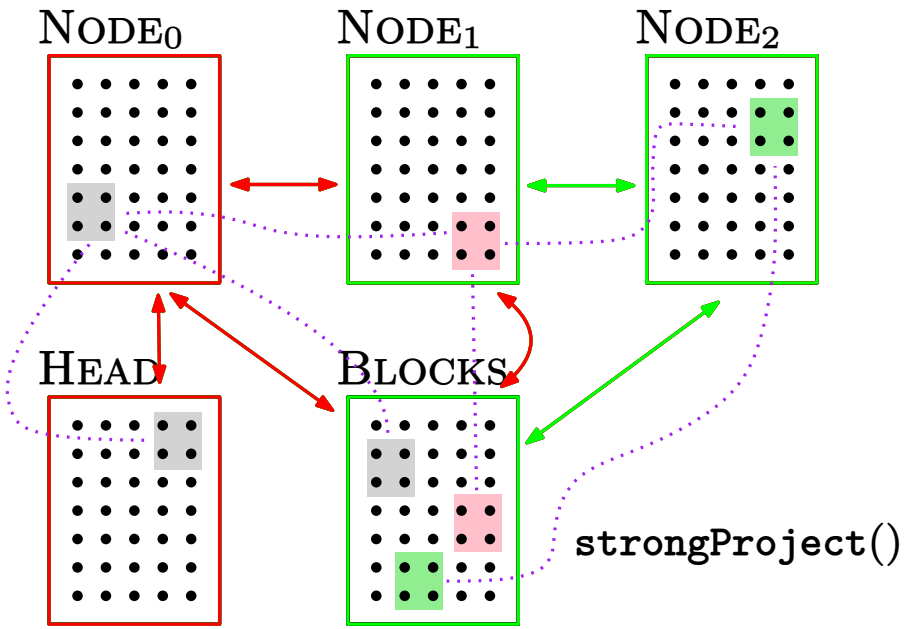


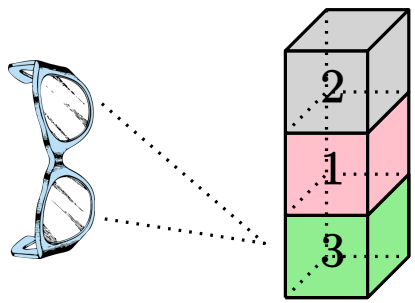


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

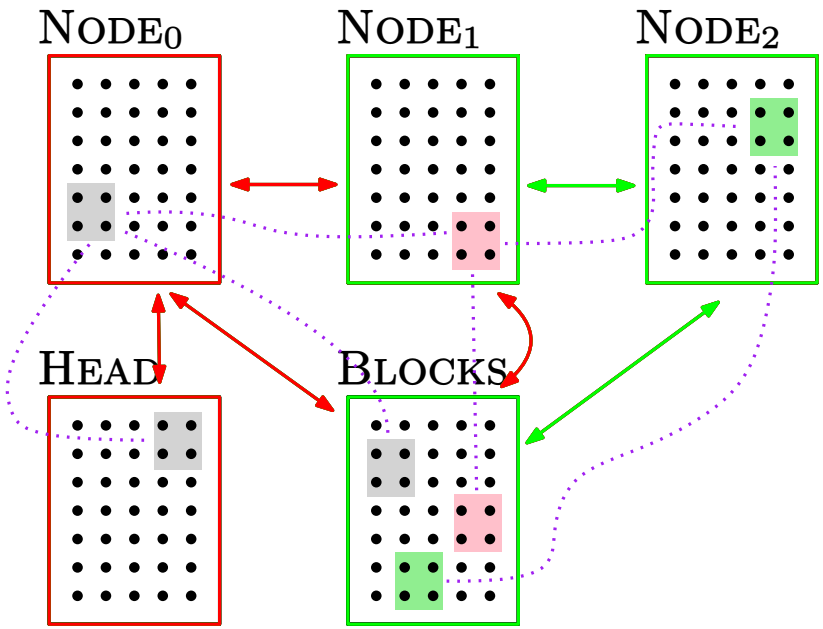


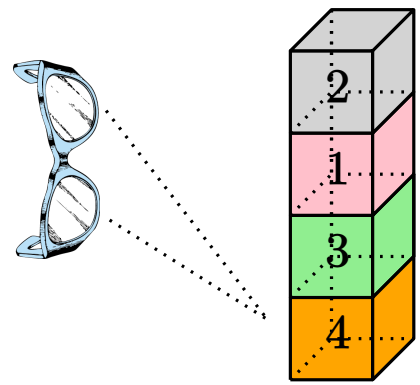


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

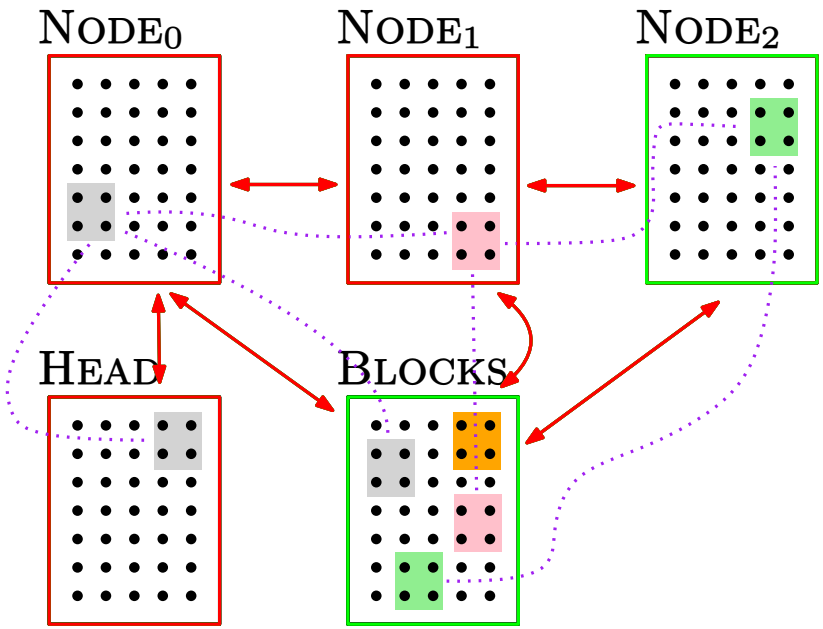


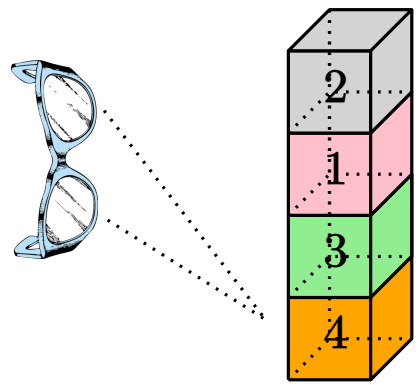


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

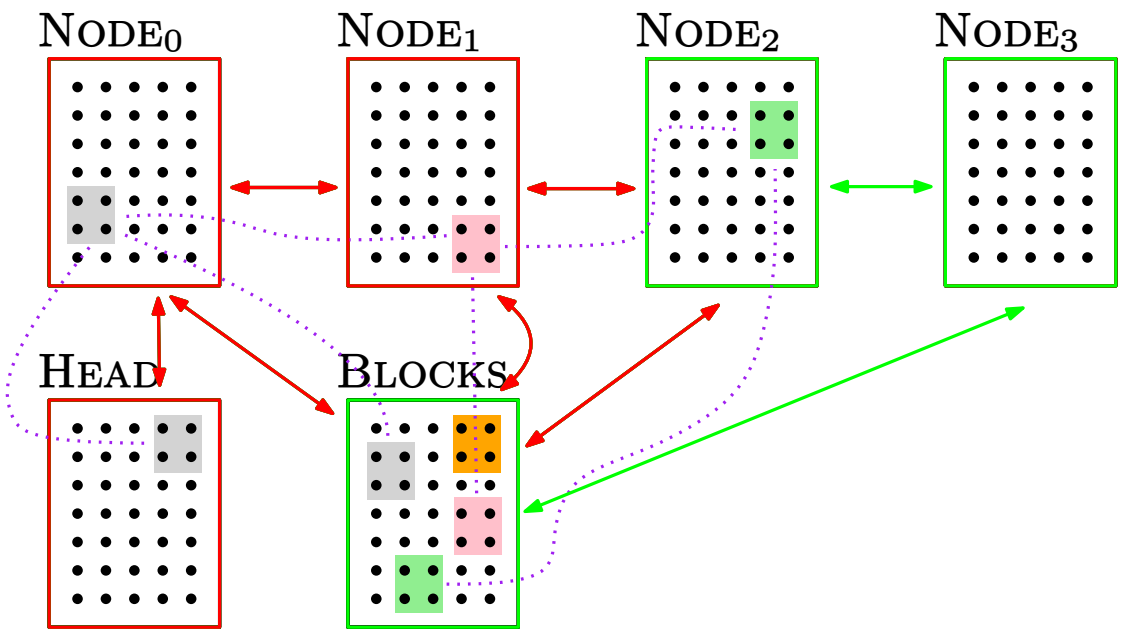


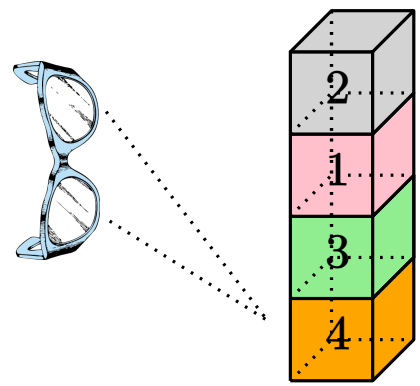


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD



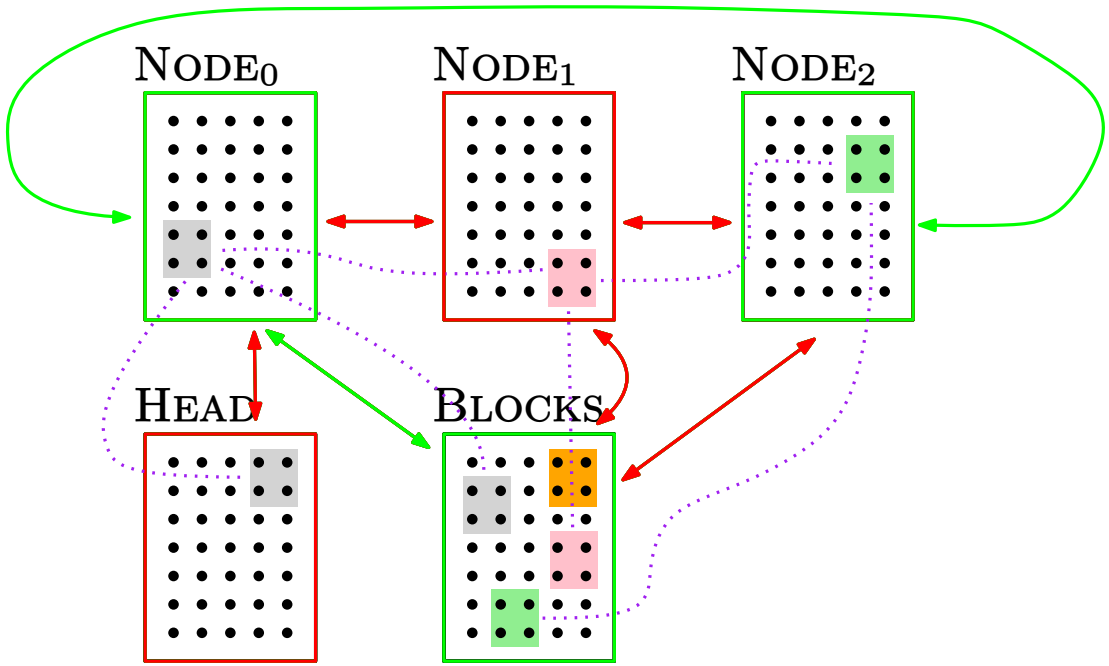


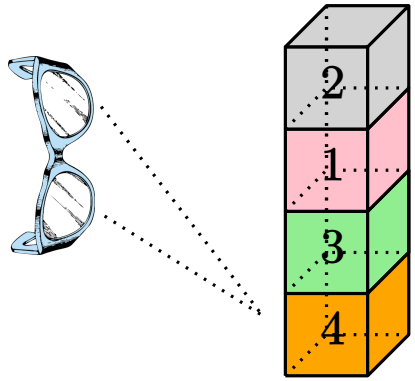
As many areas as the number of blocks... :(

The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

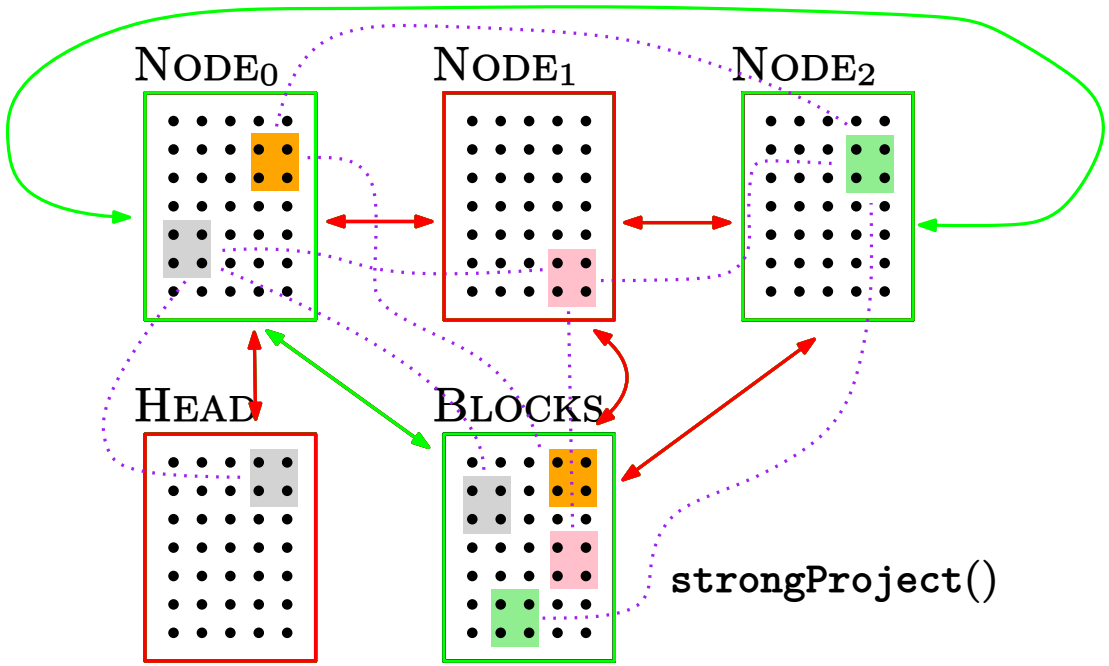


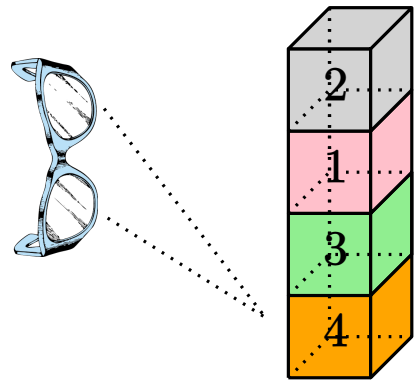


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD

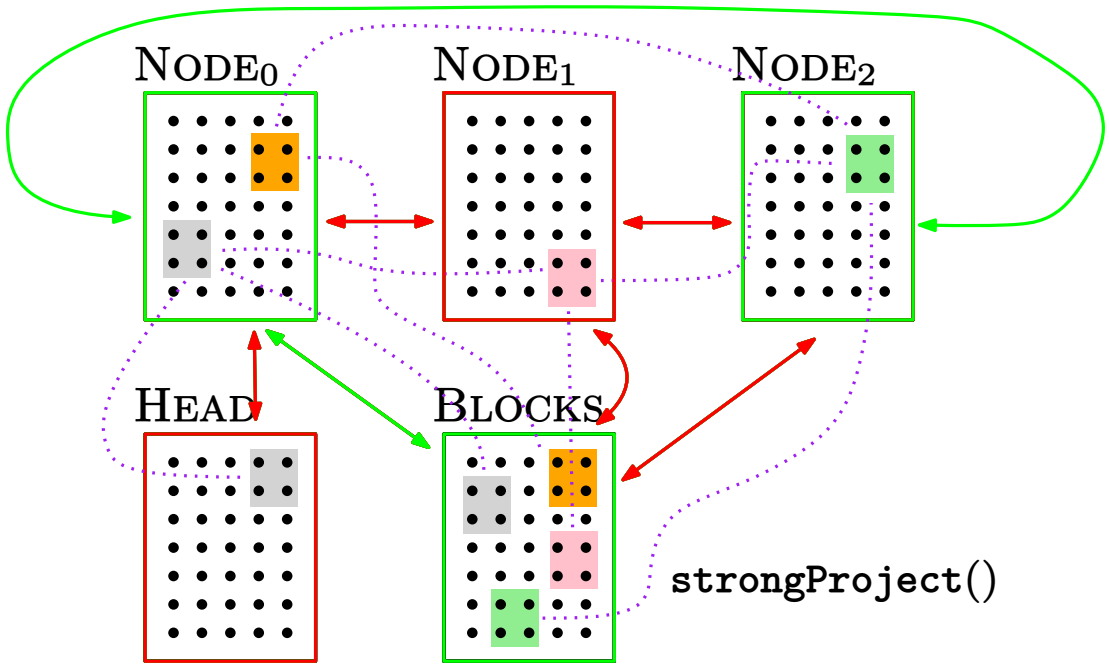


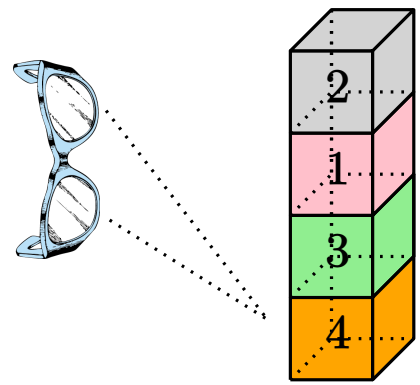


The Parser

Assumptions:

- single stack
- brain as undirected graph
- 5 brain areas, BLOCKS, NODE$_0$, NODE$_1$, NODE$_2$, HEAD





Chaining: novelty. Number of blocks represented depends only on n (number of neurons per area) and k (the **CAP**)

Parser Code

Algorithm 2: $\textsc{Parser}\,(S)$

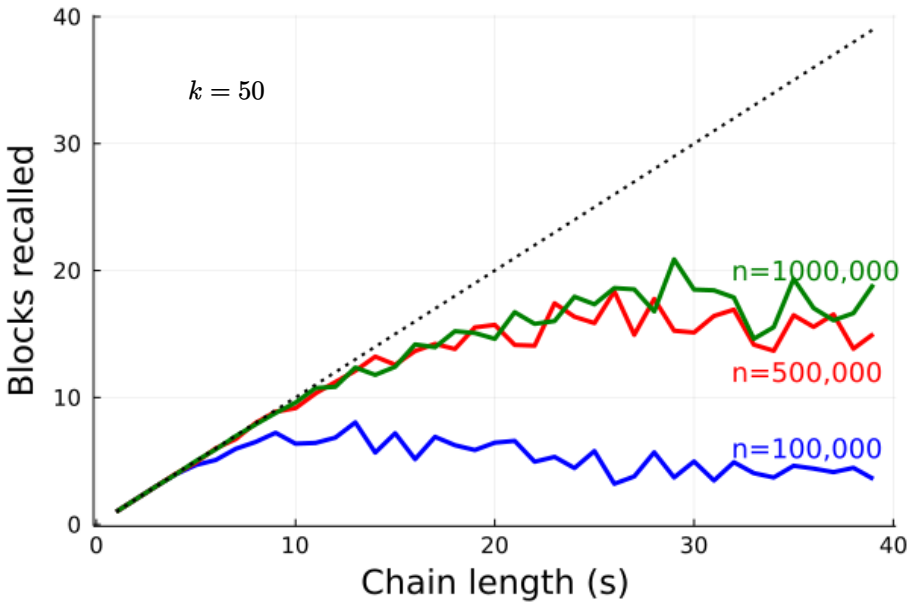
input: a stack S of blocks $b_1, b_2, \ldots, b_s$.

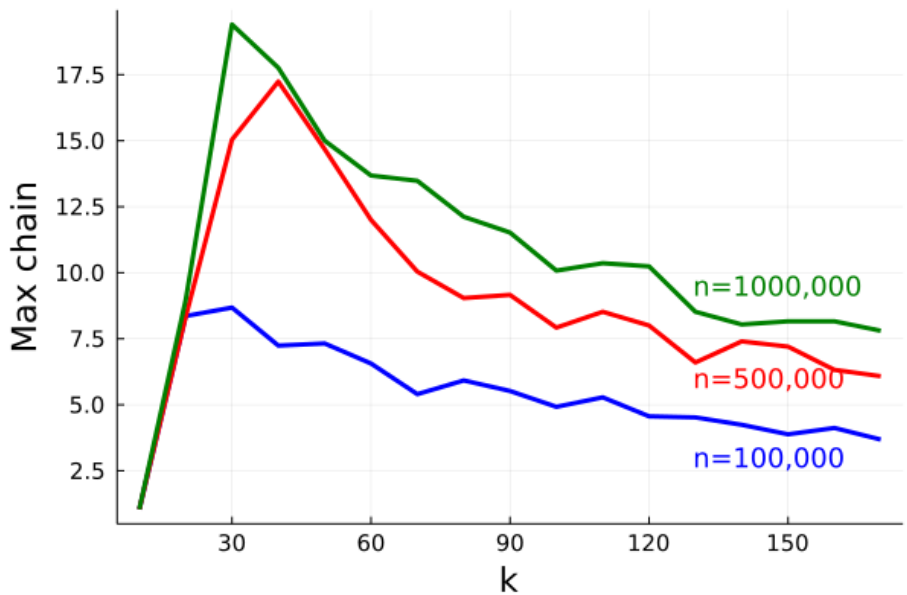
```
1  disinhibitArea ({Blocks, Head, Node_0});
2  disinhibitFiber ({(Head, Node_0) , (Node_0, Blocks)});
3  fire (b_1);
4  strongProject();
5  inhibitArea ({Head});
6  inhibitFiber ({(Head, Node_0) , (Node_0, Blocks)});
7  foreach i with 2 ≤ i ≤ s do
8  |   p = (i − 2) mod 3; c = (i − 1) mod 3;
9  |   disinhibitArea ({Node_c});
10 |   disinhibitFiber ({(Node_p, Node_c) , (Node_c, Blocks)});
11 |   fire (b_i);
12 |   strongProject();
13 |   inhibitArea ({Node_p});
14 |   inhibitFiber ({(Node_p, Node_c) , (Node_c, Blocks)});
15 end
16 inhibitArea ({Blocks, Node_(s−1) mod 3});
```

Chaining Experiments

Due to overlap of assemblies, chaining may fail

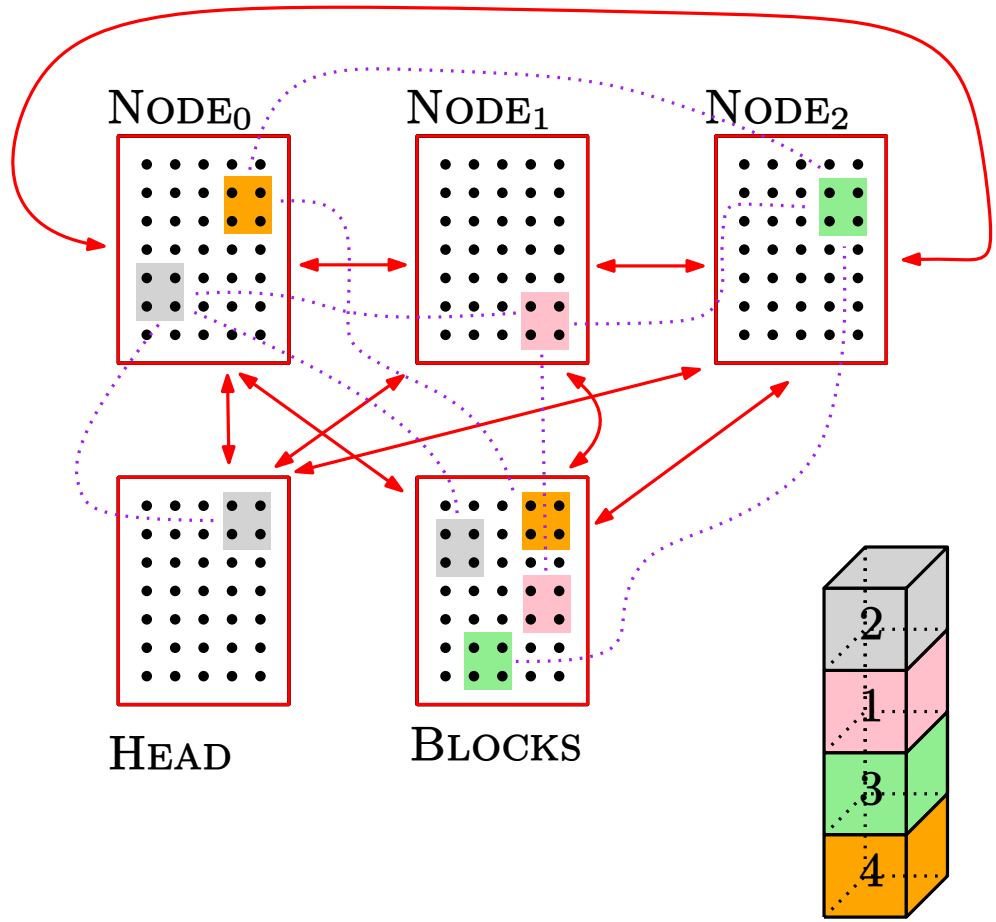
Experiments conducted averaging over 50 trials





Actions

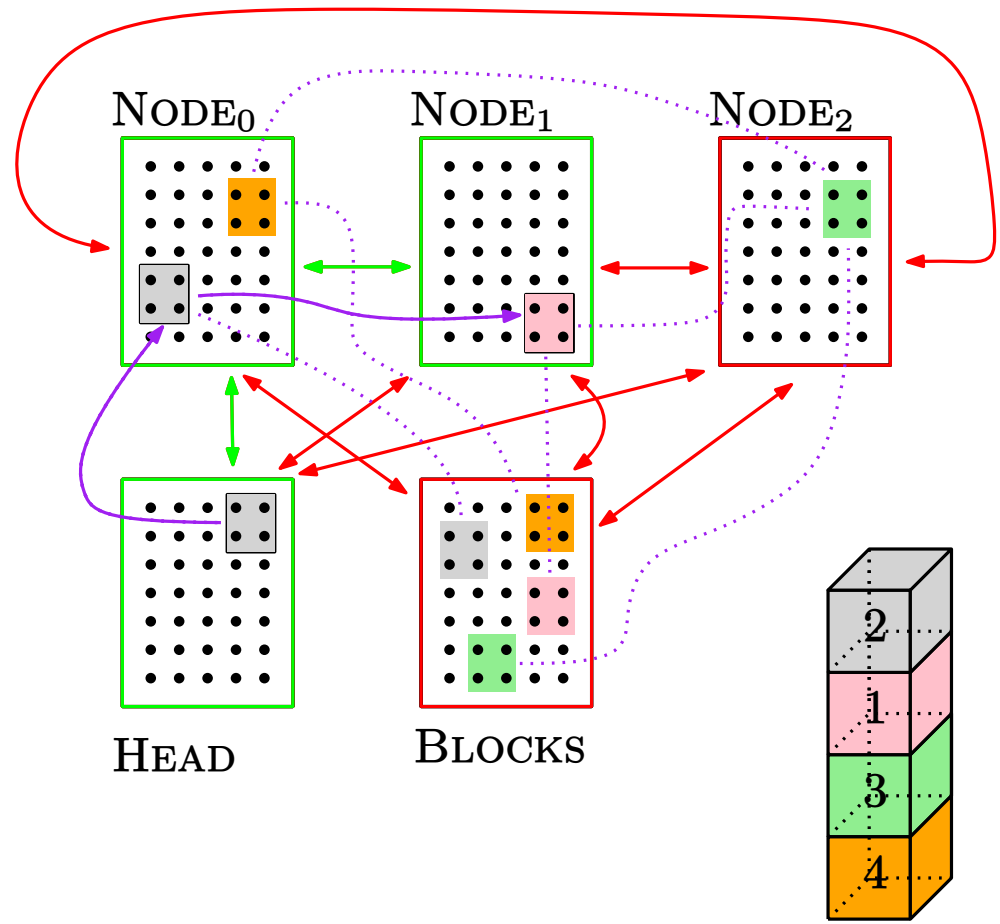
- *Removing the top block*



Actions

- *Removing the top block*

The block in HEAD fires



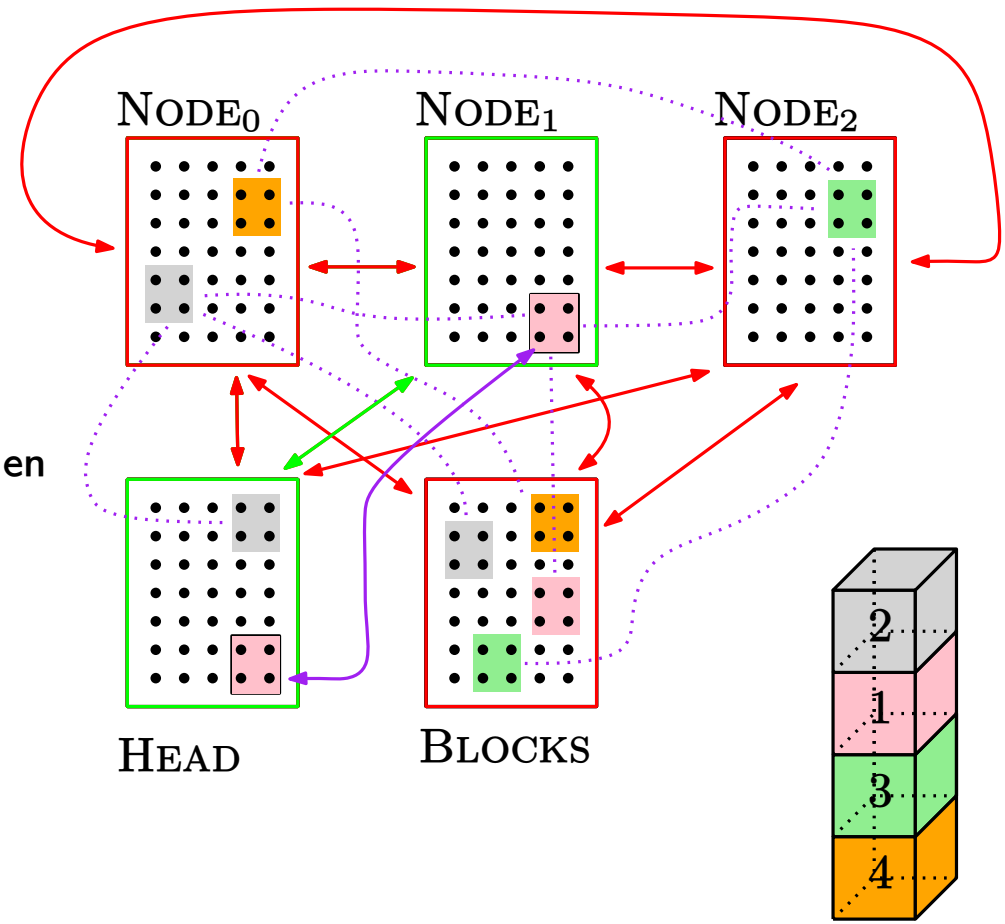
Actions

- *Removing the top block*

The last active assembly in HEAD will point to the new top

The model doesn't forget old connections, it simply doesn't use them again

Now, `project` (NODE$_1$, HEAD) and then `project` (HEAD, NODE$_1$) to reinforce



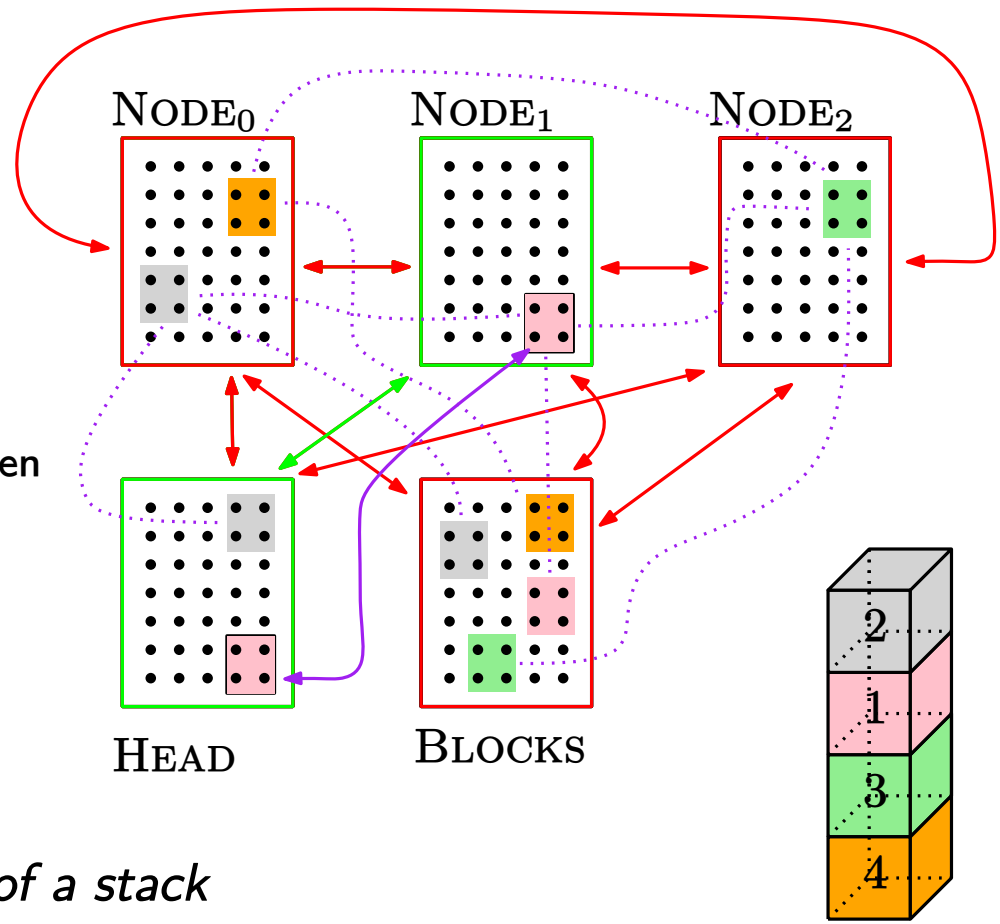
Actions

- *Removing the top block*

The last active assembly in HEAD will point to the new top

The model doesn't forget old connections, it simply doesn't use them again

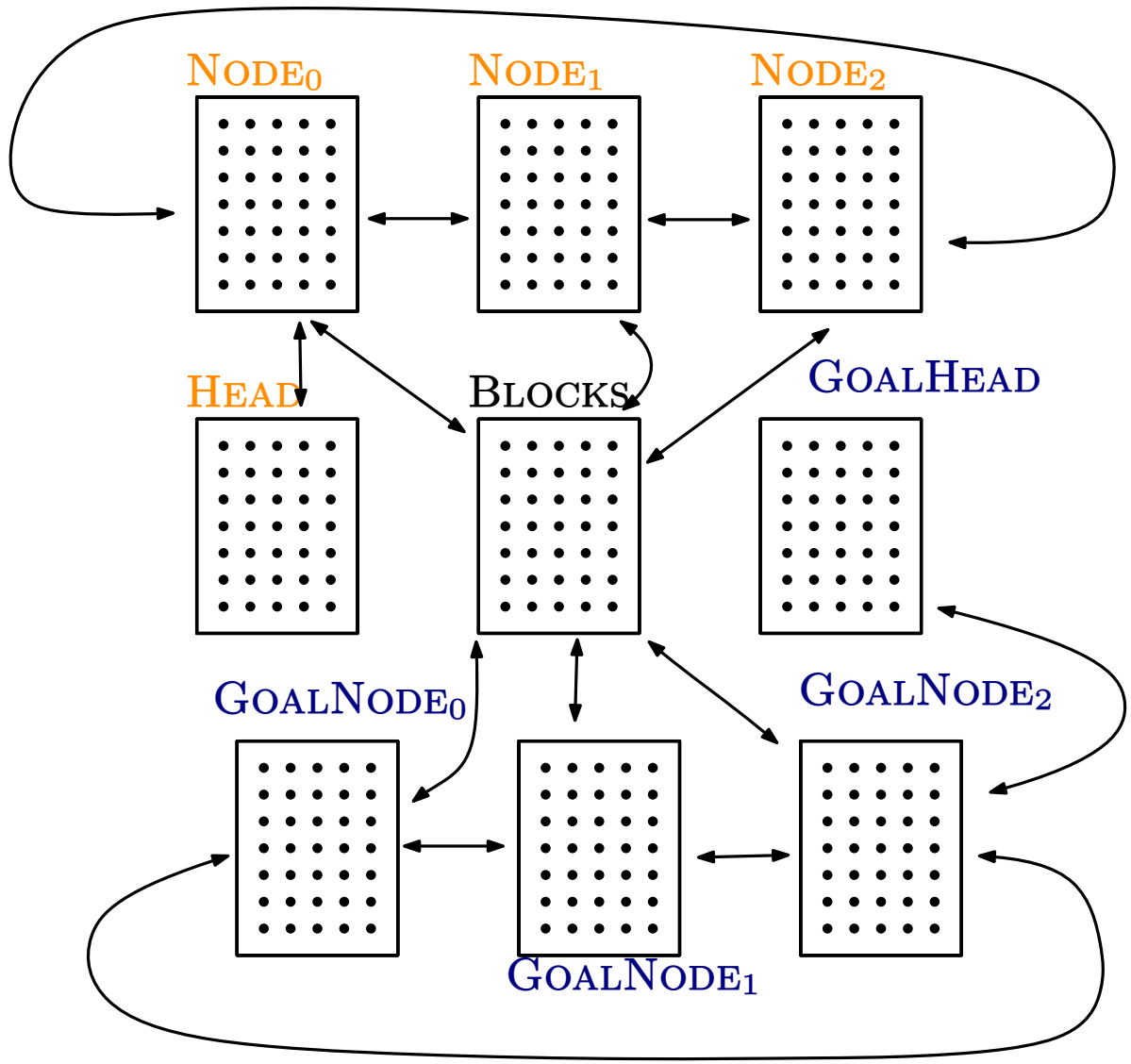
Now, `project` (NODE$_1$, HEAD) and then `project` (HEAD, NODE$_1$) to reinforce



- *Put a block from the table to the top of a stack*

Comparisons

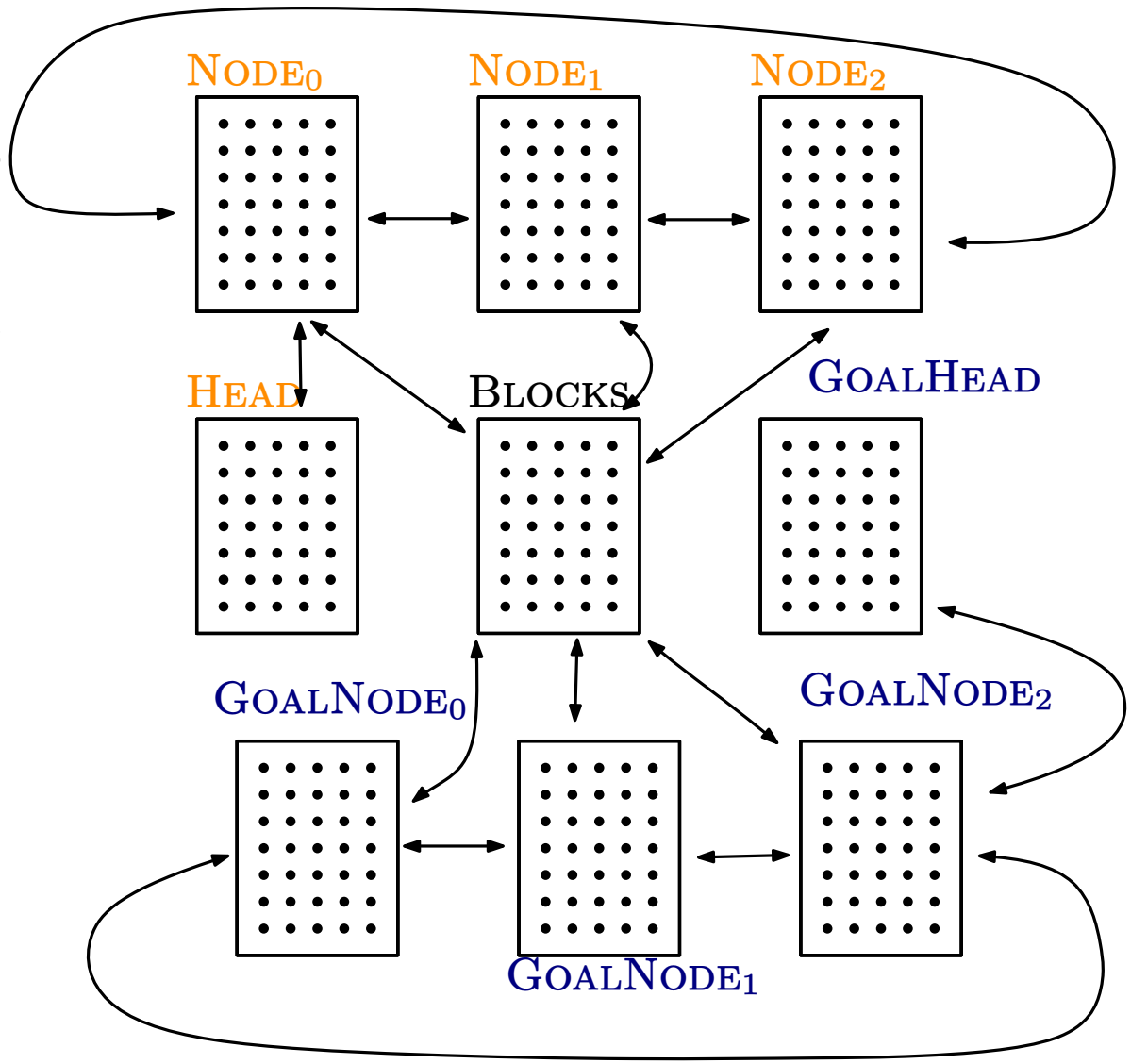
We have to save both the initial configuration and the goal configuration



Comparisons

We have to save both the initial configuration and the goal configuration

- *Intersect*: scrolls down the two stacks until reaching the last block, then starts comparing each pair of blocks by scrolling up

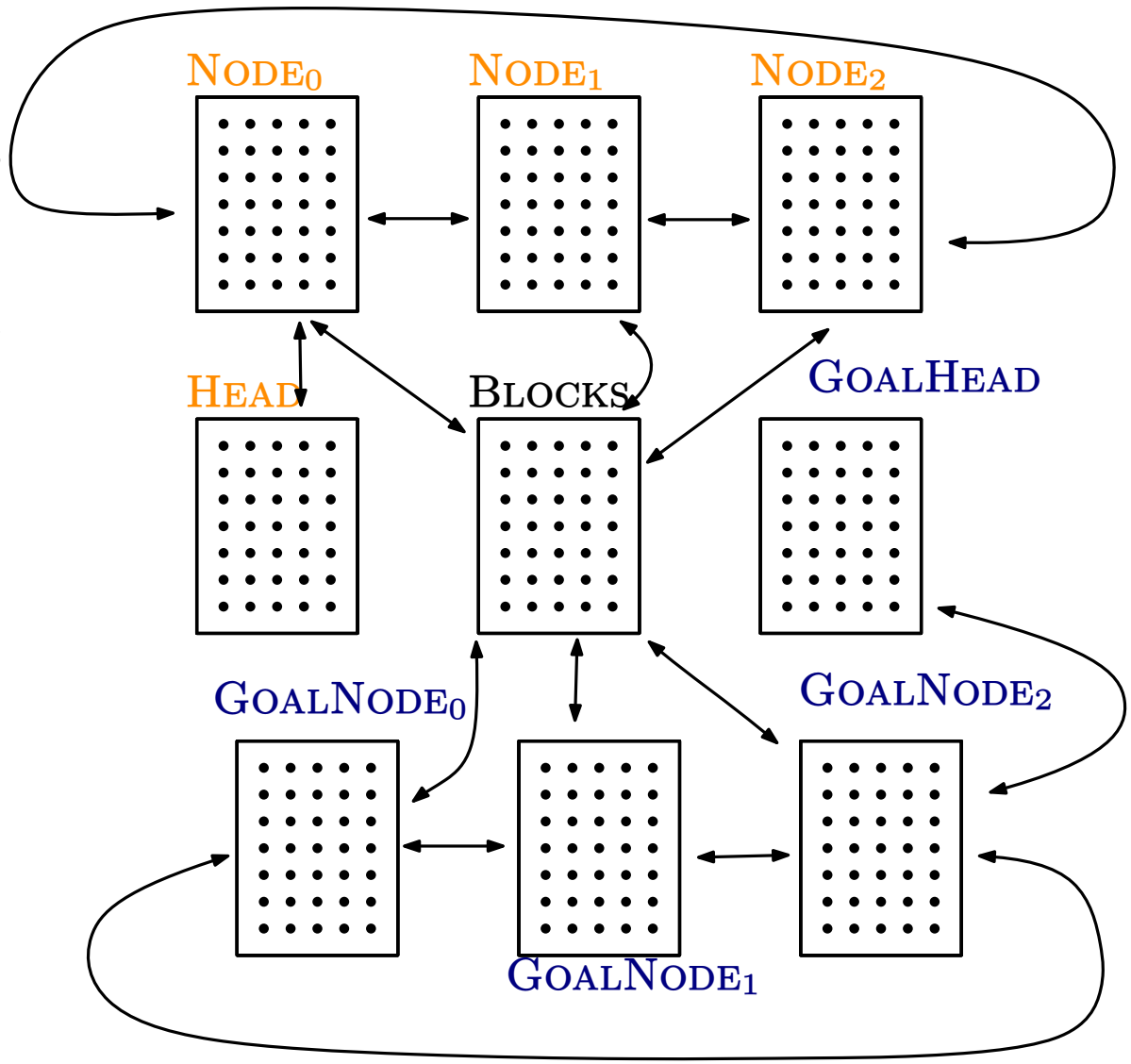


Comparisons

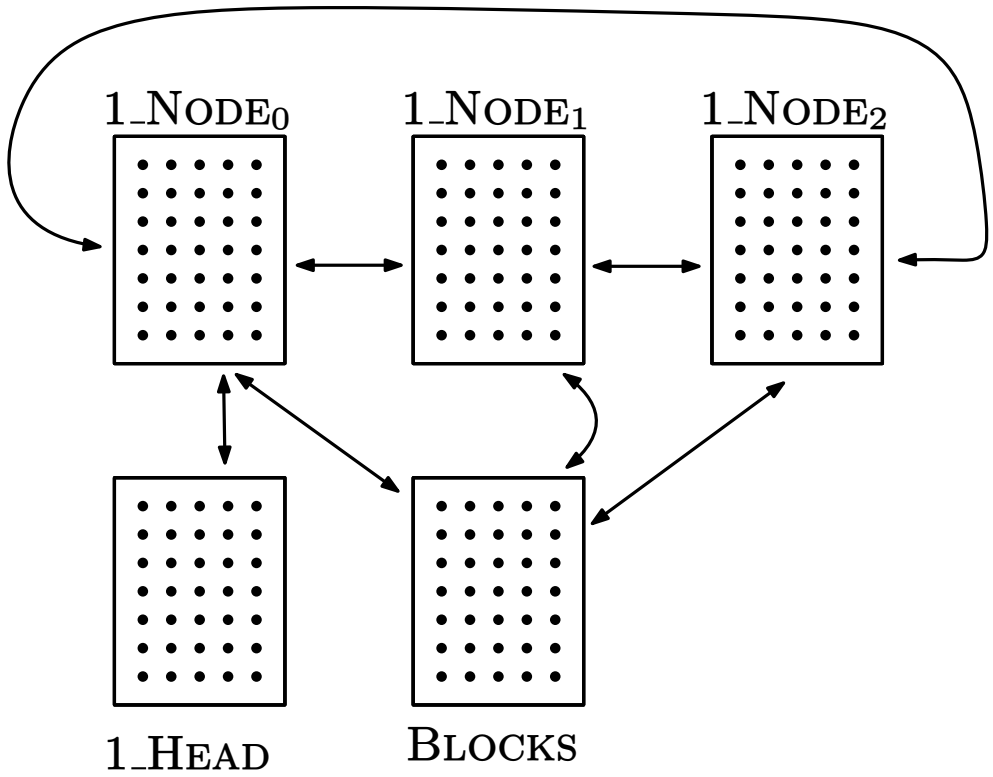
We have to save both the initial configuration and the goal configuration

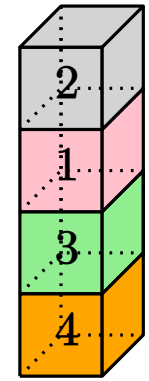
- *Intersect*: scrolls down the two stacks until reaching the last block, then starts comparing each pair of blocks by scrolling up

- By combining these operations we implement the planning strategies

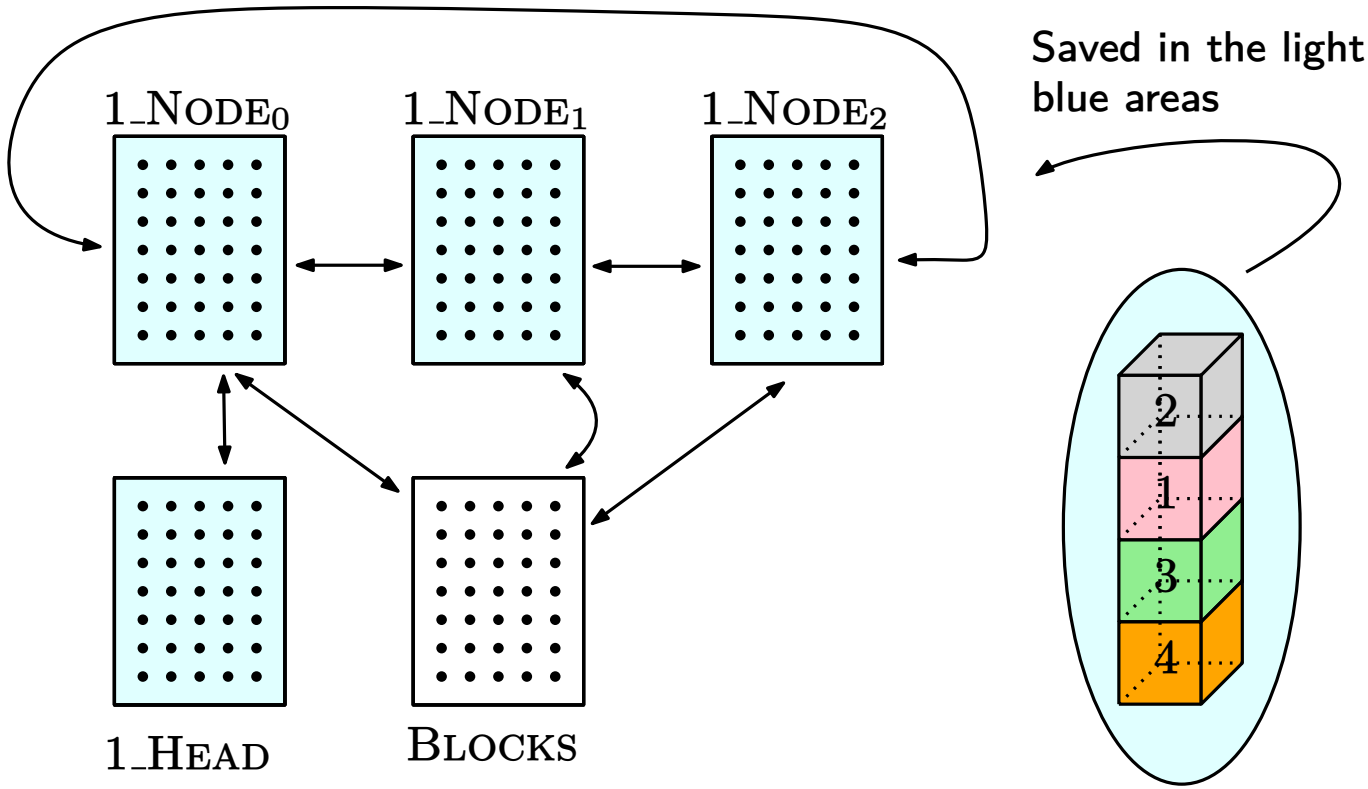


Multiple Stacks Case

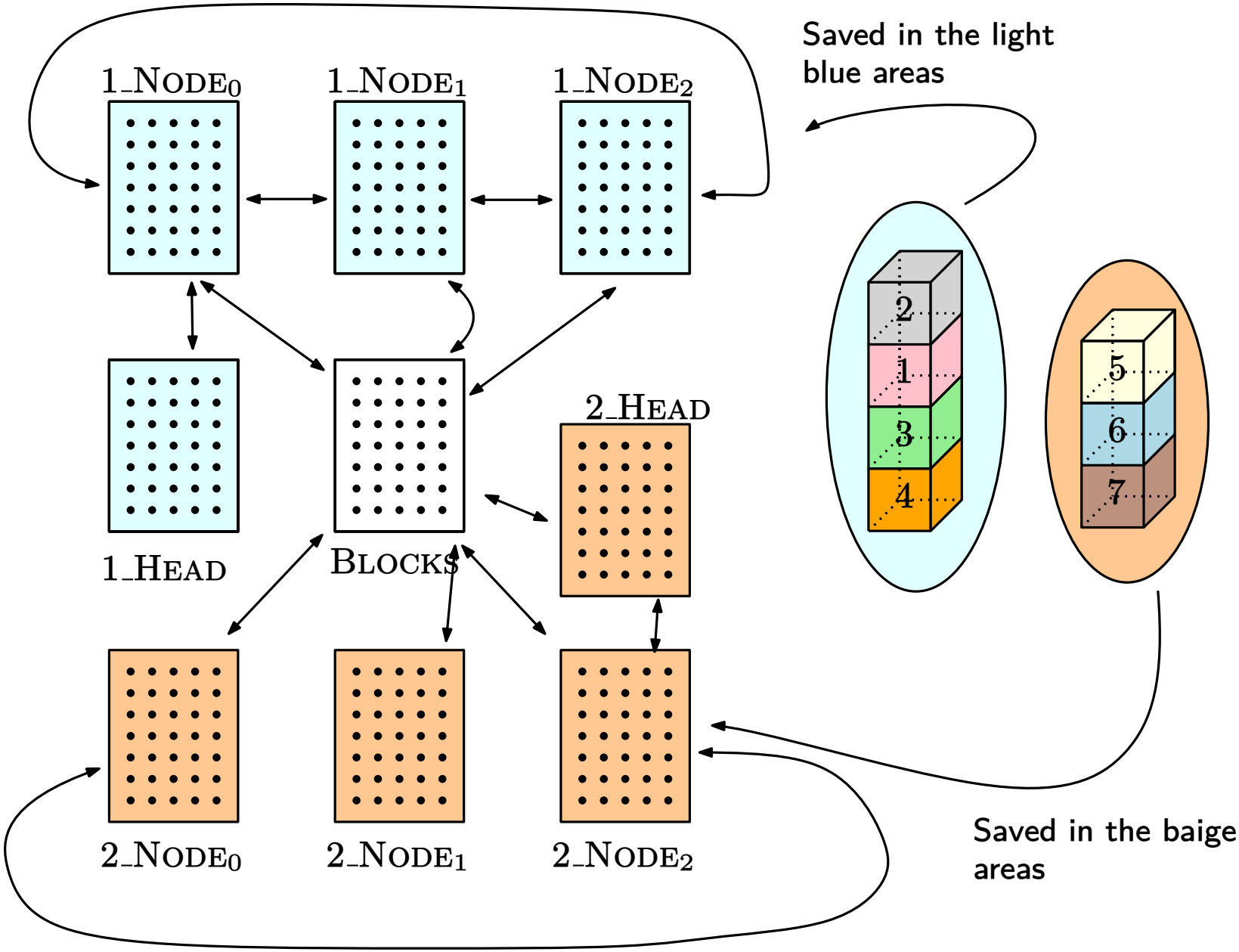




Multiple Stacks Case

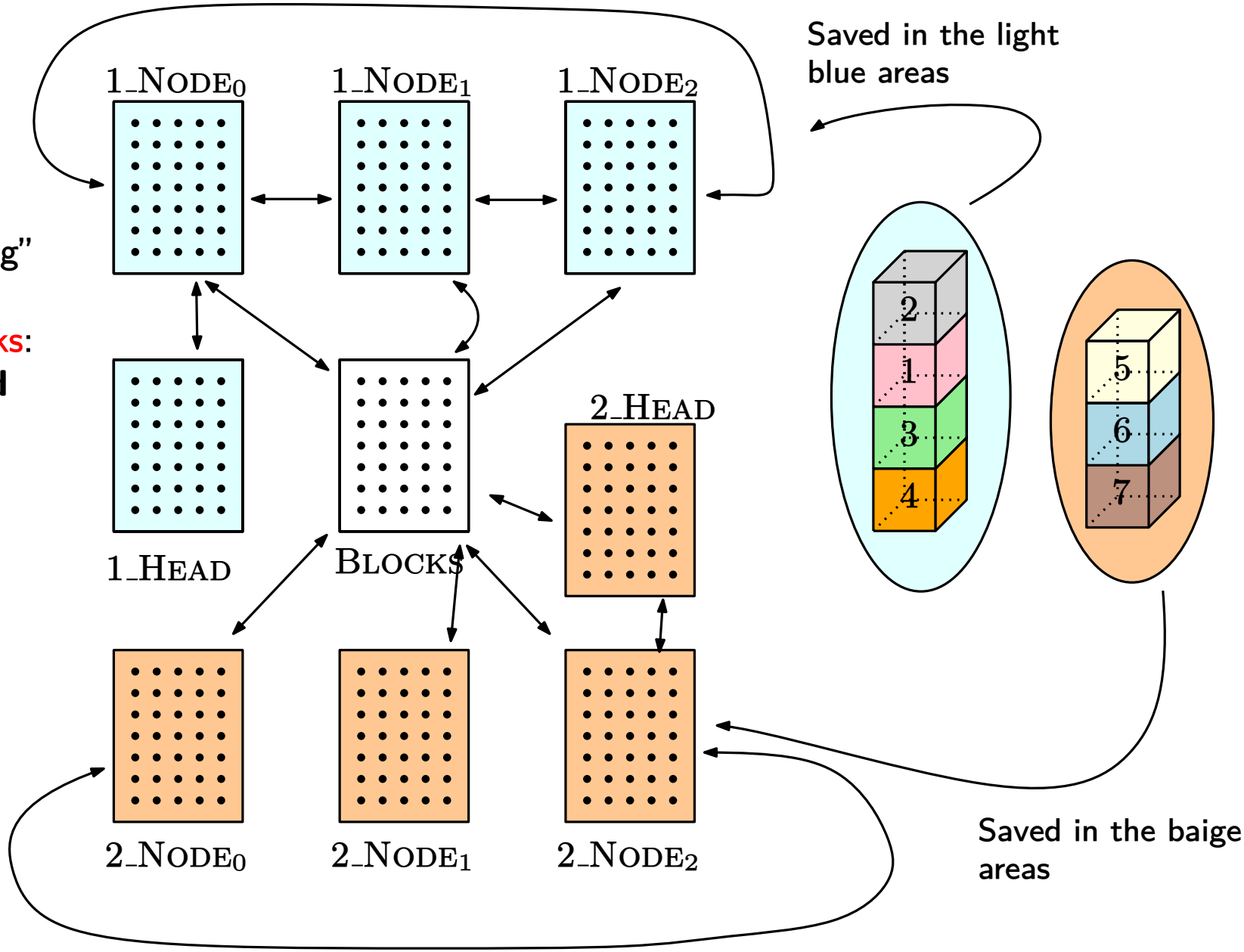


Multiple Stacks Case



Multiple Stacks Case

As many "saving" regions as the number of stacks: **to be improved** (like chaining)



Discussion

- Demonstrated experimentally that reasonably large and complex programs in the assembly calculus can **execute** correctly and reliably

- Shown the realization of a *list-like data structure* which makes use of a constant number of brain regions

- Shown how simple manipulations of the data structure (removing or putting a block) can be realized by making use of a constant number of brain regions

- **Bottleneck**: the parsing. Its reliability depends on the ratio between the number of neurons and the size of the assemblies in each region

    - Must be the object of *further investigation*