exploring higher-dimensional phase transitions

a brief odyssey

an overview

an overview

1. the Lenz-Ising model

an overview

1. the Lenz-Ising model
2. simulations and ~computational topology~

an overview

1. the Lenz-Ising model
2. simulations and ~computational topology~
3. speedbumps

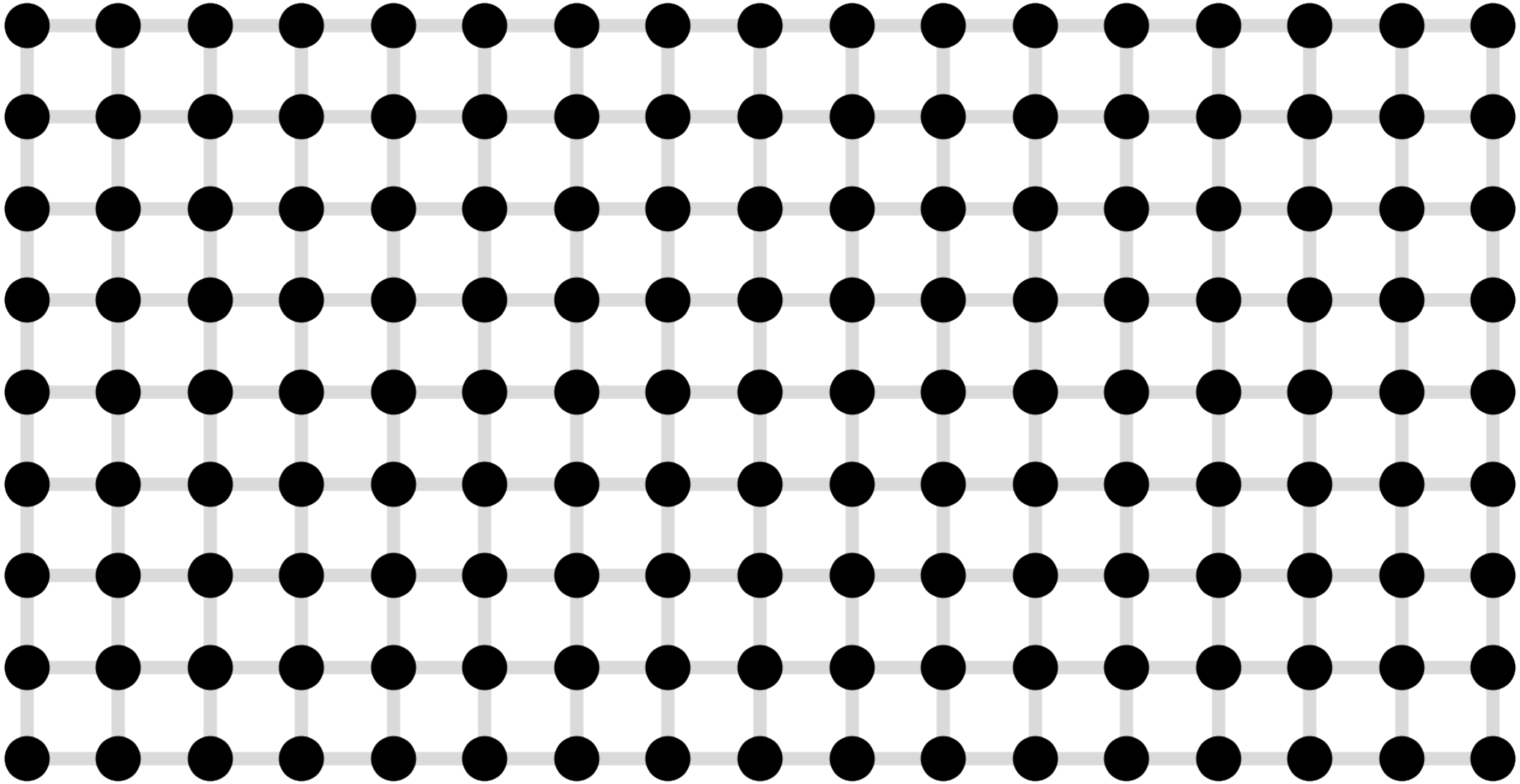
an overview

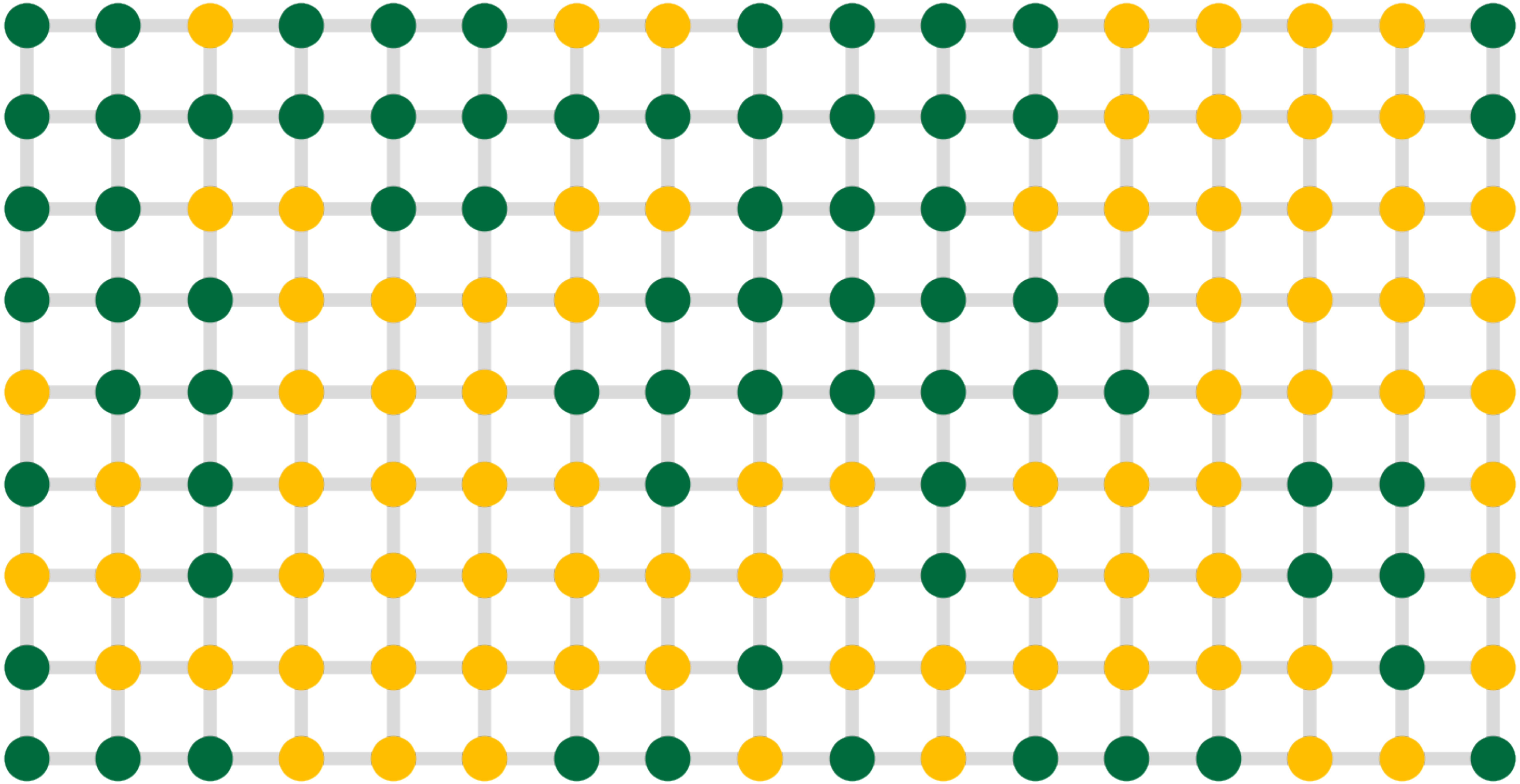
1. the Lenz-Ising model
2. simulations and ~computational topology~
3. speedbumps
4. future work

1. the Lenz-Ising model

how do we simulate *magnetism*?

how do we simulate magnetism *well*?

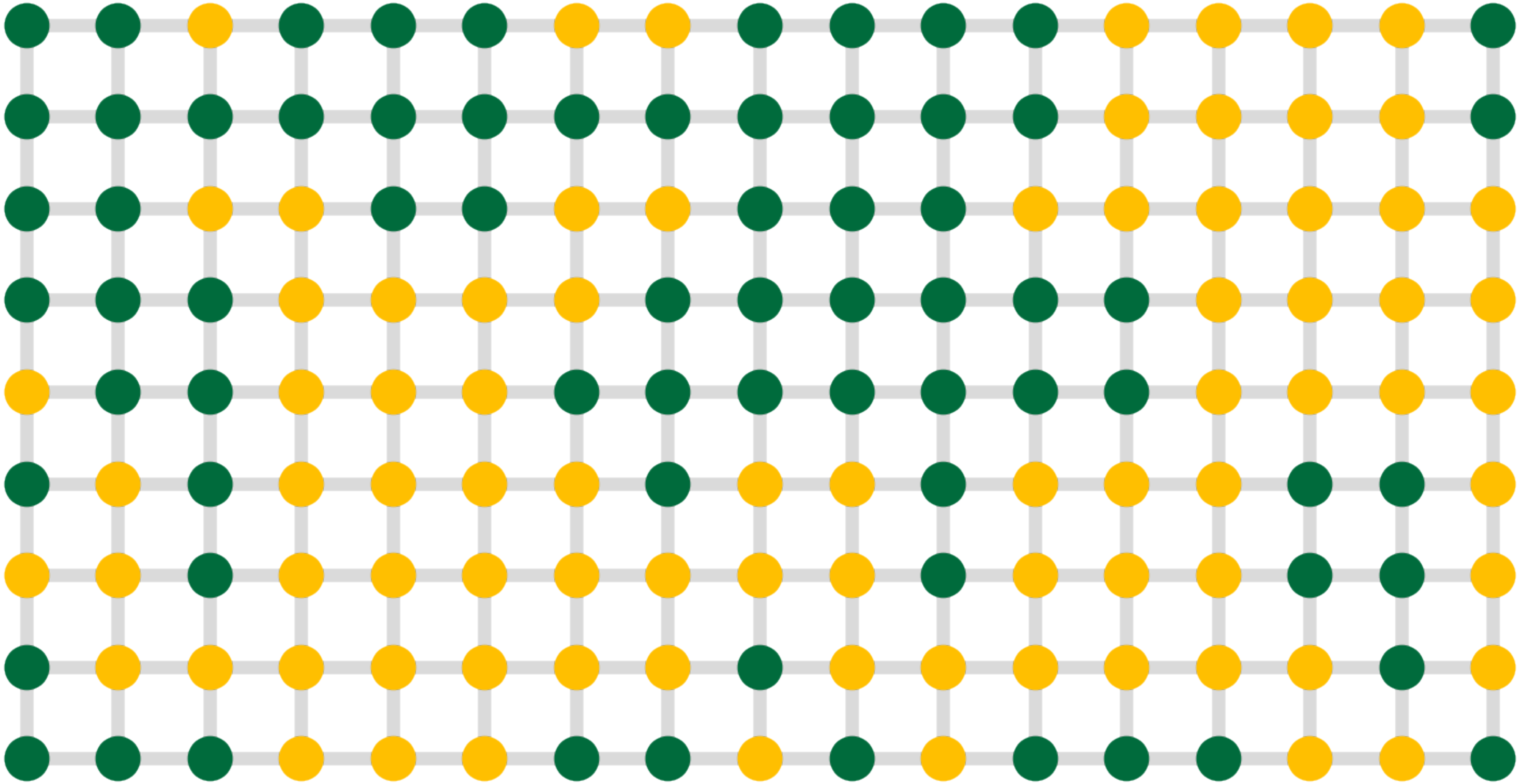


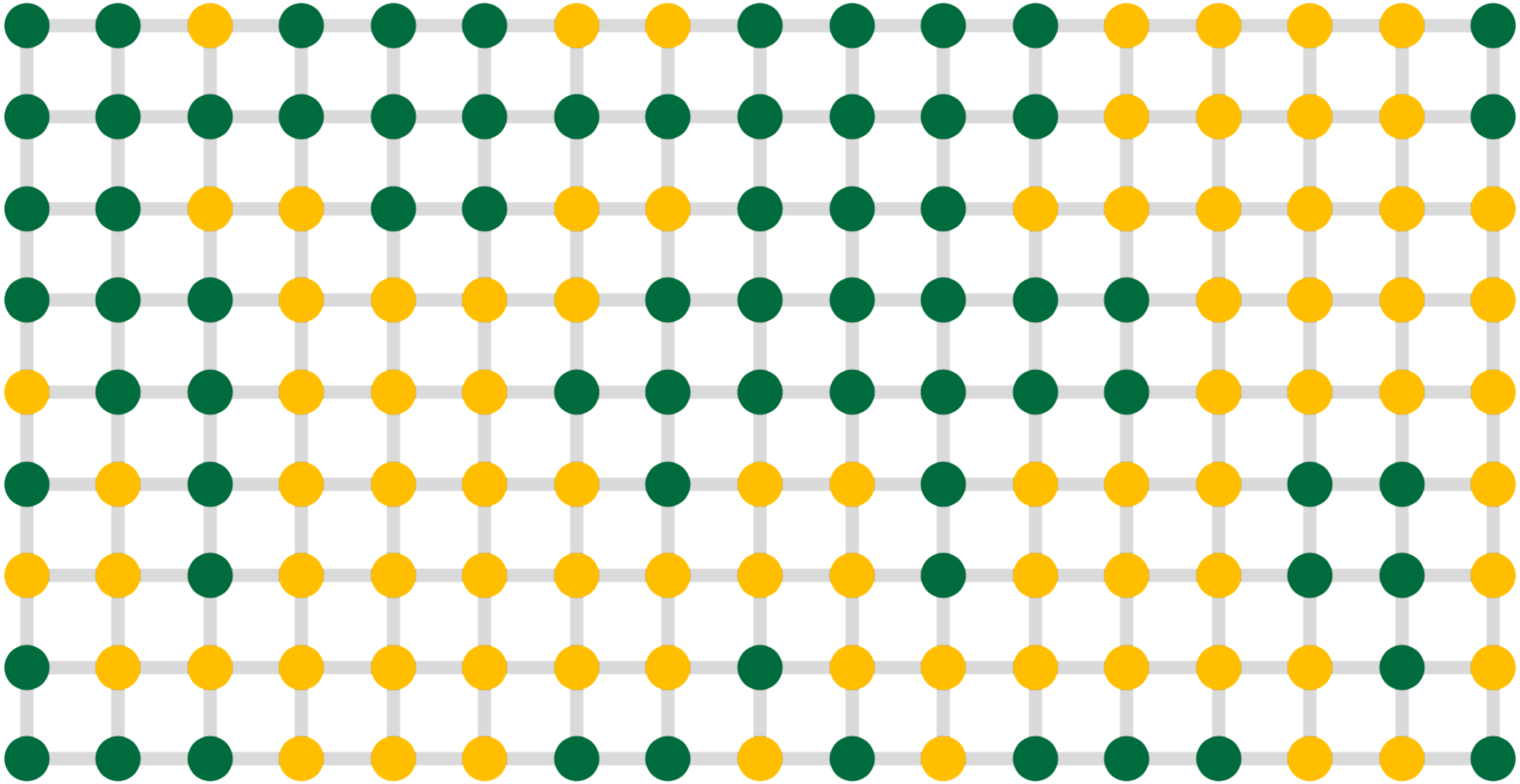


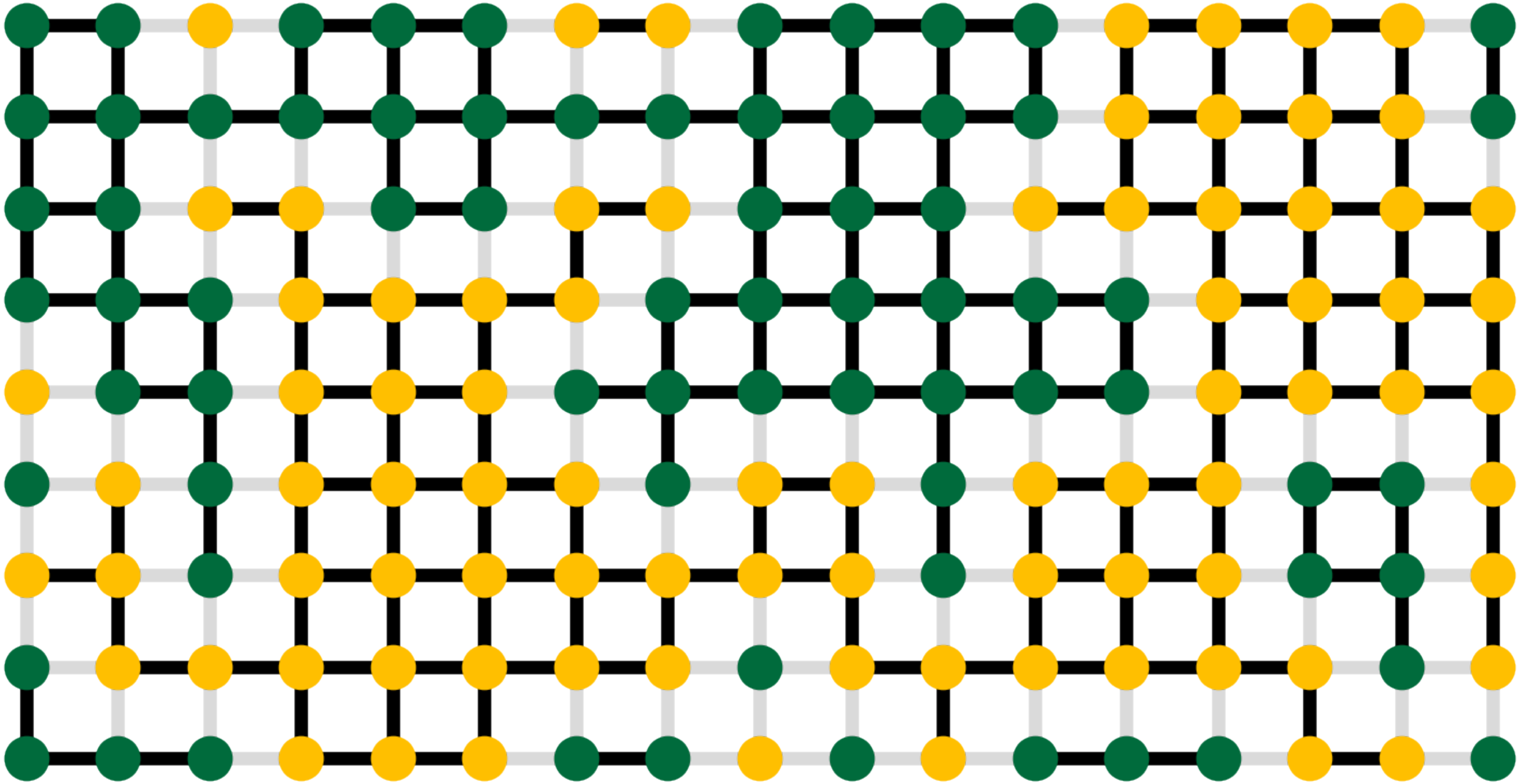
spin configuration

$$\sigma : V \to \mathbb{F}_2$$

(we like this to be linear)

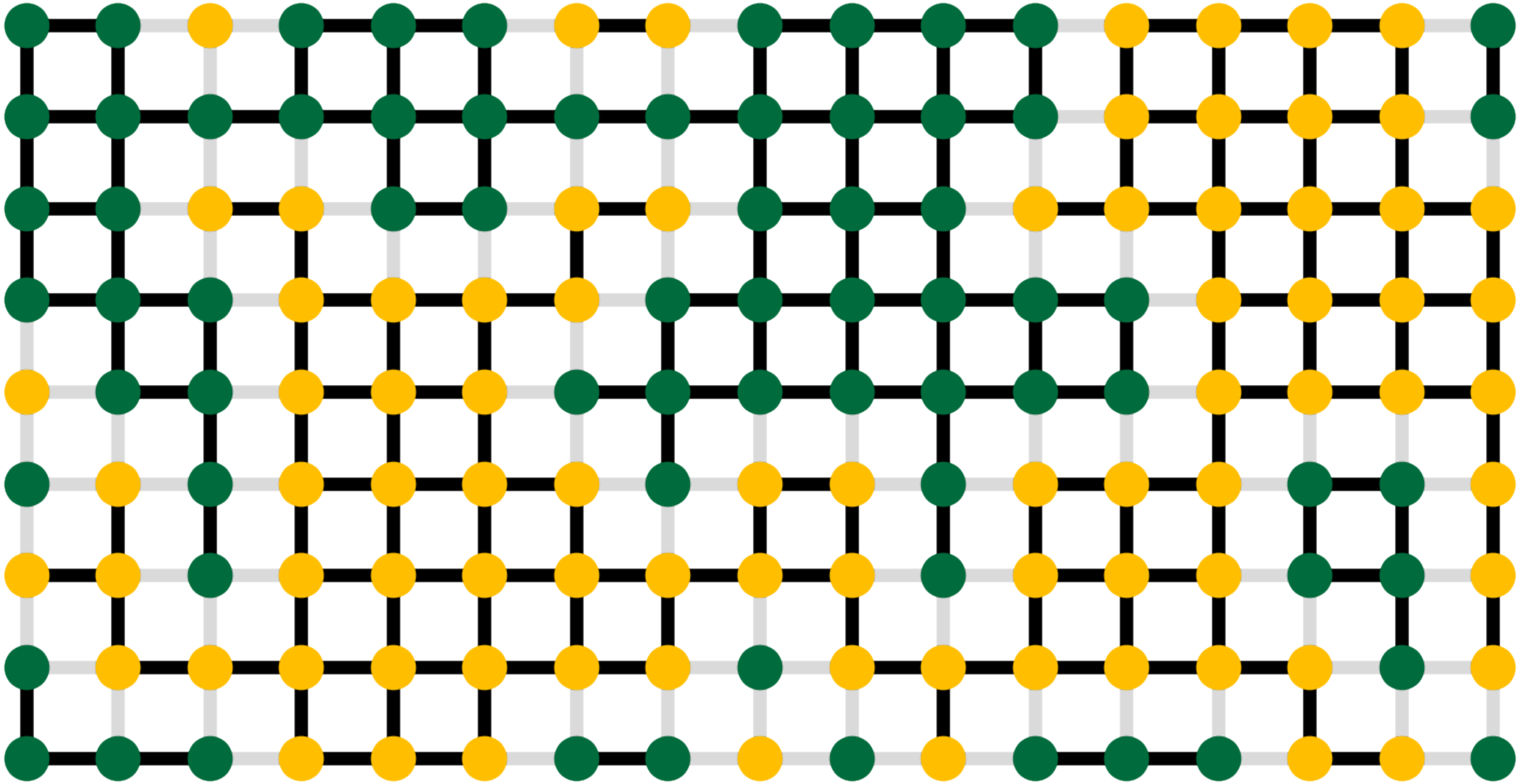






edge configuration

$\omega : E \to \{0,1\}$



we want to characterize *how likely* a given (σ, ω) pair is.

what we *know*:

what we *know*:

1. for each edge e in the graph,

what we *know*:

1. for each edge e in the graph,
   a. e is included in some cluster with probability p, or

what we *know*:

1. for each edge e in the graph,
   a. e is included in some cluster with probability p, or
   b. e is included in *no* cluster, with probability $(1-p)$;

what we *know*:

1. for each edge e in the graph,
    a. e is included in some cluster with probability p, or
    b. e is included in *no* cluster, with probability $(1-p)$;
2. there are *exactly* two states each cluster can take on.

$$\mu(\sigma, \omega) \propto \frac{1}{Z} \cdot \left(\prod_{e \in E_G} (1-p)^{1-\omega(e)} \cdot p^{\omega(e)} \right) \cdot 2^{\kappa(G)}$$

$$\mu(\sigma, \omega) \propto \frac{1}{Z} \cdot \left(\prod_{e \in E_G} (1-p)^{1-\omega(e)} \cdot p^{\omega(e)} \right) \cdot 2^{\kappa(G)}$$

for each edge...

chance this edge *isn't* in a cluster

$$\mu(\sigma, \omega) \propto \frac{1}{Z} \cdot \left(\prod_{e \in E_G} \boxed{(1-p)^{1-\omega(e)}} \cdot p^{\omega(e)} \right) \cdot 2^{\kappa(G)}$$

for each edge...

chance this edge *isn't* in a cluster

$$\mu(\sigma, \omega) \propto \frac{1}{Z} \cdot \left(\prod_{e \in E_G} \boxed{(1-p)^{1-\omega(e)}} \cdot \boxed{p^{\omega(e)}} \right) \cdot 2^{\kappa(G)}$$

for each edge...

chance this edge
is in a cluster

$$\mu(\sigma, \omega) \propto \frac{1}{Z} \cdot \left(\prod_{e \in E_G} \boxed{(1-p)^{1-\omega(e)}} \cdot \boxed{p^{\omega(e)}} \right) \cdot \boxed{2^{\kappa(G)}}$$

chance this edge *isn't* in a cluster

number of ways to assign spins to each cluster

for each edge...

chance this edge *is* in a cluster

$$\mu(\sigma, \omega) \propto \frac{1}{Z} \cdot \left(\prod_{e \in E_G} (1-p)^{1-\omega(e)} \cdot p^{\omega(e)} \right) \cdot 2^{\kappa(G)}$$

$$\mu(\sigma, \omega) \propto \frac{1}{Z} \cdot \left(\prod_{e \in E_G} (1 - p)^{1-\omega(e)} \cdot p^{\omega(e)} \right) \cdot q^{\kappa(G)}$$

$$\mu(\sigma, \omega) \propto \frac{1}{Z} \cdot \left(\prod_{e \in E_G} (1-p)^{1-\omega(e)} \cdot p^{\omega(e)} \right) \cdot q^{\kappa(G)}$$

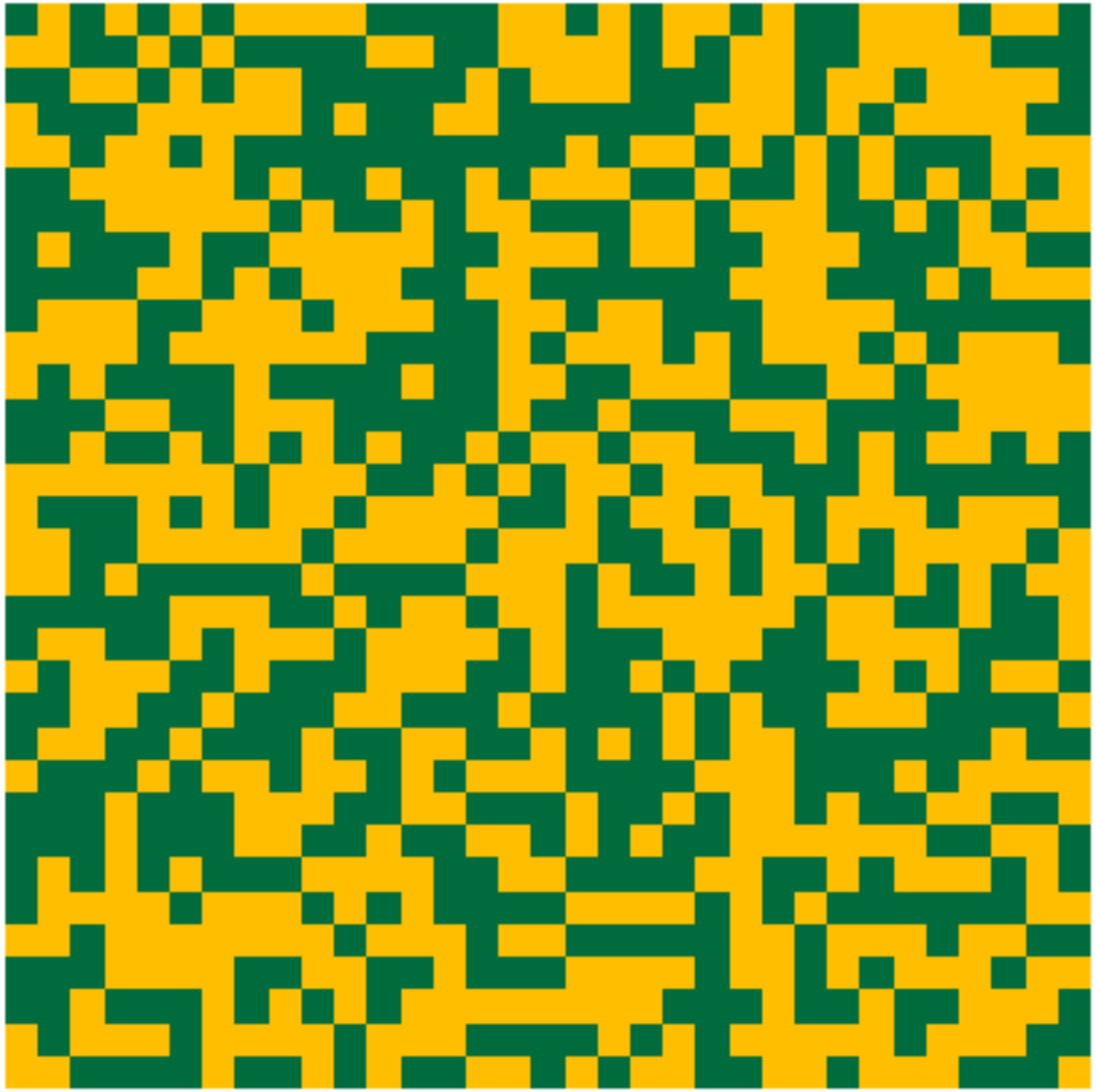
Random Cluster model

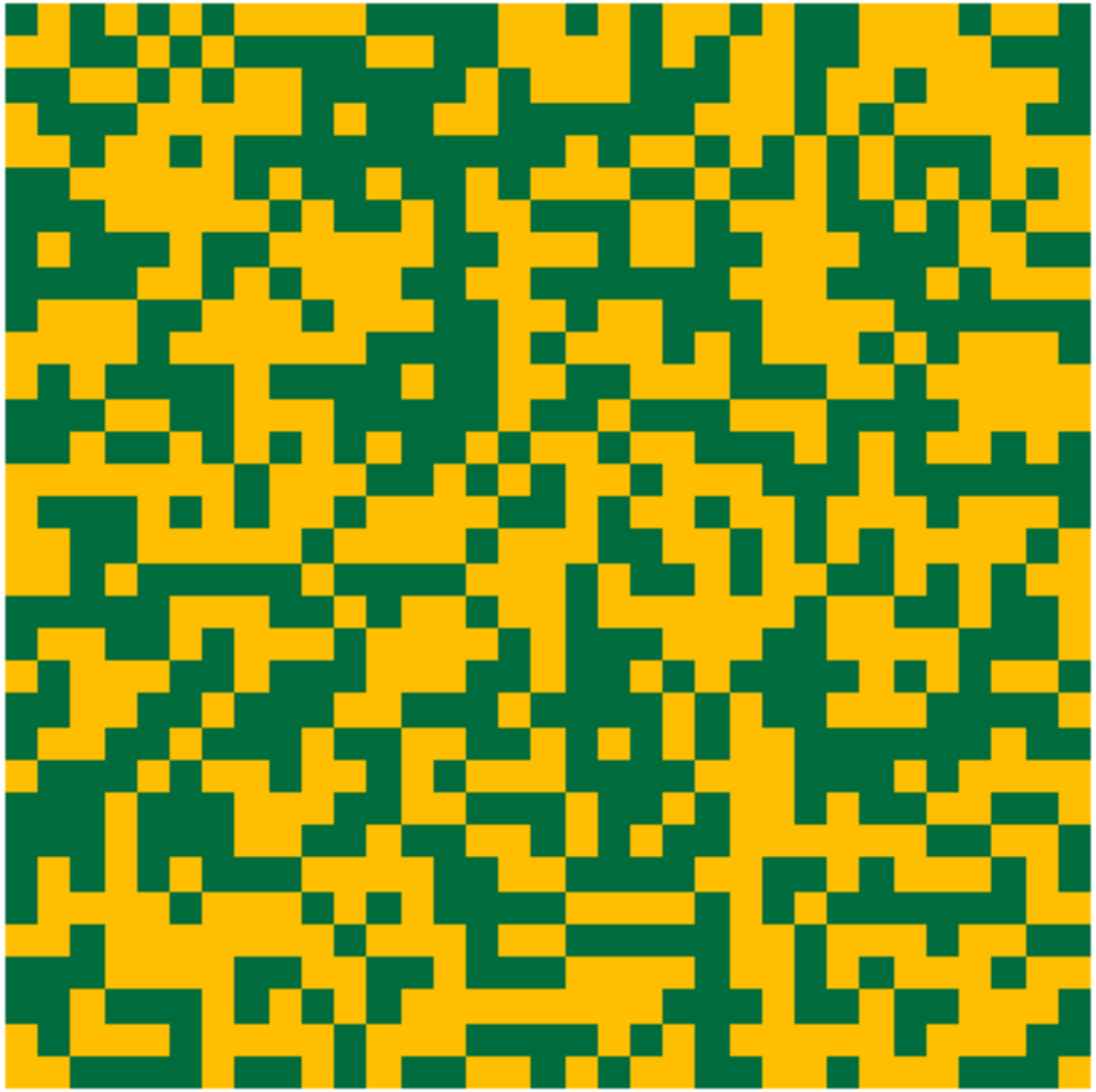
most general

2. simulations and ~computational topology~

a *random process* is a series of draws from a random variable.

a *Markov chain* is a random process where the outcome of the next draw depends *only* on the value of the previous draw traditional Metropolis-Hastings algorithm.





Swendsen-Wang algorithm.

Swendsen-Wang algorithm.

Given a lattice L, some number of iterations N, and an initial labeling σ_0,

Swendsen-Wang algorithm.

Given a lattice L, some number of iterations N, and an initial labeling σ_0,

1. for each time t in $\{1, ..., N\}$,

Swendsen-Wang algorithm.

Given a lattice L, some number of iterations N, and an initial labeling σ_0,

1. for each time t in $\{1, ..., N\}$,
    a. initialize a collection C of edges.

Swendsen-Wang algorithm.

Given a lattice L, some number of iterations N, and an initial labeling σ_0,

1. for each time t in $\{1, \ldots, N\}$,
   a. initialize a collection C of edges.
   b. for each edge $e = (u,v)$ in G,

Swendsen-Wang algorithm.

Given a lattice L, some number of iterations N, and an initial labeling σ_0,

1. for each time t in $\{1, \ldots, N\}$,
   a. initialize a collection C of edges.
   b. for each edge $e = (u,v)$ in G,
      i. include e in C with probability p if $0 = \sigma_{t-1}(v) - \sigma_{t-1}(u) = \sigma_{t-1}(u-v)$;

Swendsen-Wang algorithm.

Given a lattice L, some number of iterations N, and an initial labeling σ_0,

1. for each time t in $\{1, \ldots, N\}$,
   a. initialize a collection C of edges.
   b. for each edge $e = (u,v)$ in G,
      i. include e in C with probability p if $0 = \sigma_{t-1}(v) - \sigma_{t-1}(u) = \sigma_{t-1}(u-v)$;
      ii. ignore e otherwise.

Swendsen-Wang algorithm.

Given a lattice L, some number of iterations N, and an initial labeling σ_0,

1. for each time t in $\{1, \ldots, N\}$,
    a. initialize a collection C of edges.
    b. for each edge $e = (u,v)$ in G,
        i. include e in C with probability p if $0 = \sigma_{t-1}(v) - \sigma_{t-1}(u) = \sigma_{t-1}(u-v)$;
        ii. ignore e otherwise.
    c. get the *induced subgraph* S from G by ignoring all edges not in C.

Swendsen-Wang algorithm.

Given a lattice L, some number of iterations N, and an initial labeling σ_0,

1. for each time t in $\{1, \ldots, N\}$,
   a. initialize a collection C of edges.
   b. for each edge $e = (u,v)$ in G,
      i. include e in C with probability p if $0 = \sigma_{t-1}(v) - \sigma_{t-1}(u) = \sigma_{t-1}(u-v)$;
      ii. ignore e otherwise.
   c. get the *induced subgraph* S from G by ignoring all edges not in C.
   d. for each connected component k in $\kappa(S)$,

Swendsen-Wang algorithm.

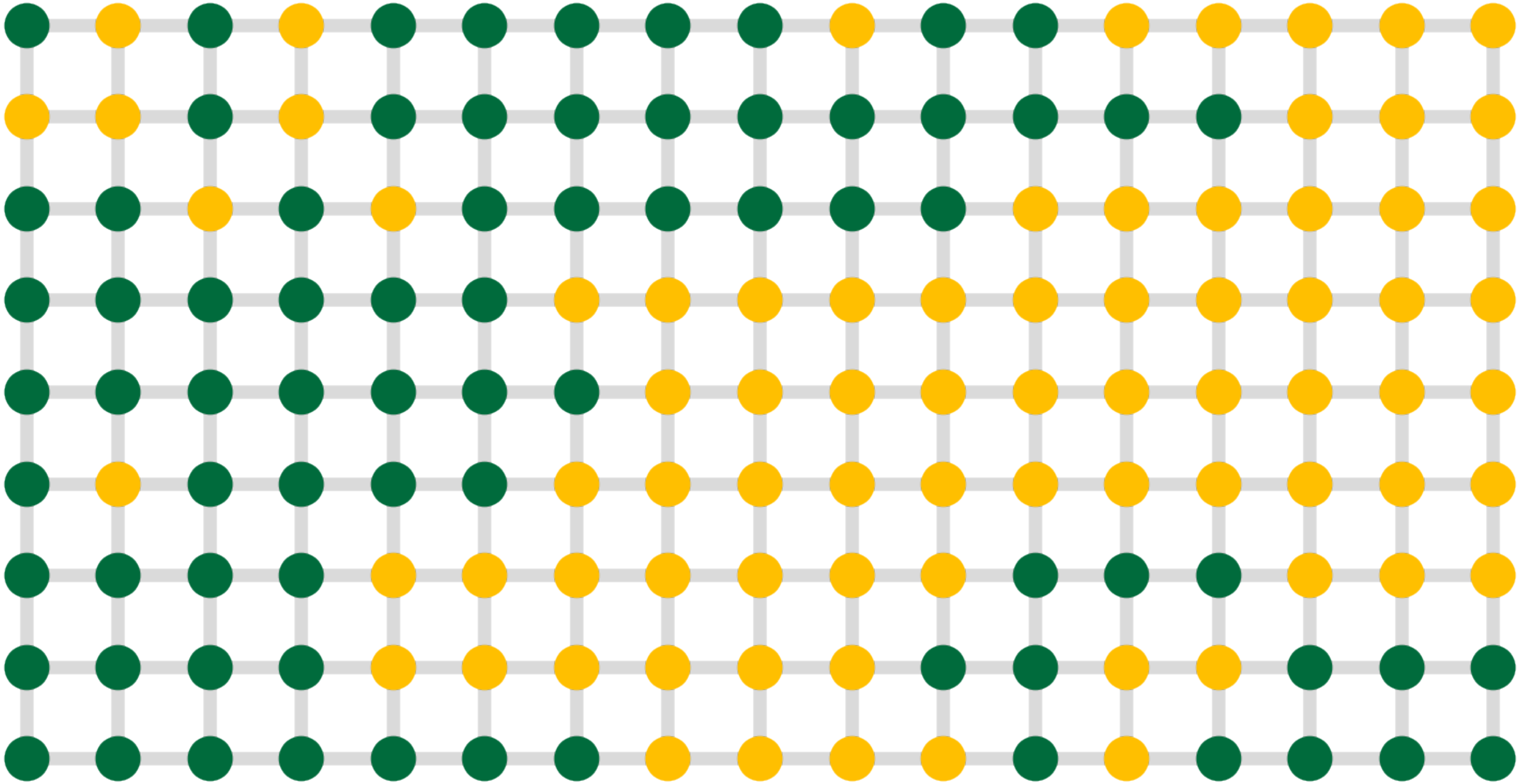
Given a lattice L, some number of iterations N, and an initial labeling σ_0,

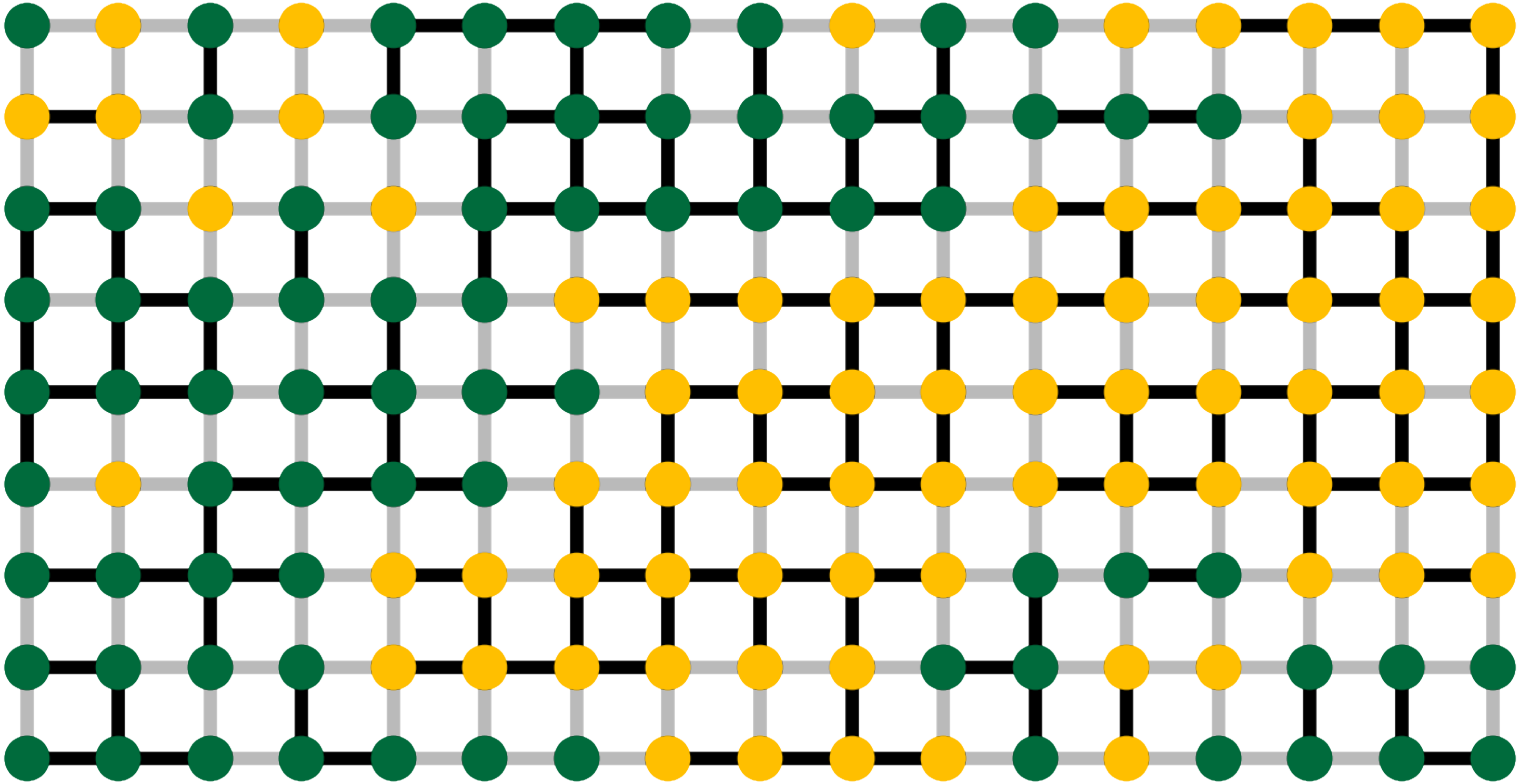
1. for each time t in $\{1, ..., N\}$,
   a. initialize a collection C of edges.
   b. for each edge $e = (u,v)$ in G,
      i. include e in C with probability p if $0 = \sigma_{t-1}(v) - \sigma_{t-1}(u) = \sigma_{t-1}(u-v)$;
      ii. ignore e otherwise.
   c. get the *induced subgraph* S from G by ignoring all edges not in C.
   d. for each connected component k in $\kappa(S)$,
      i. uniformly randomly choose a spin value *m;*

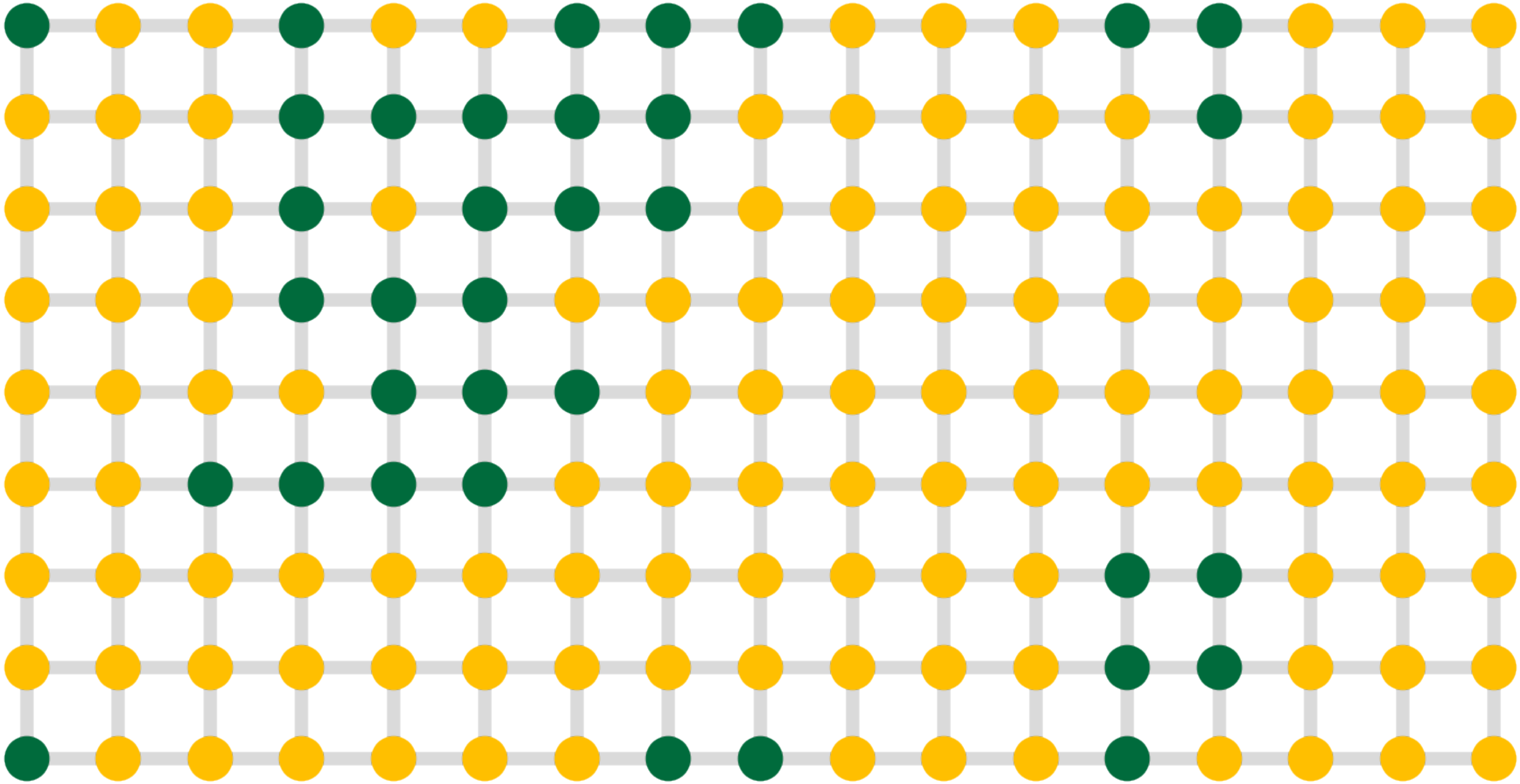
Swendsen-Wang algorithm.

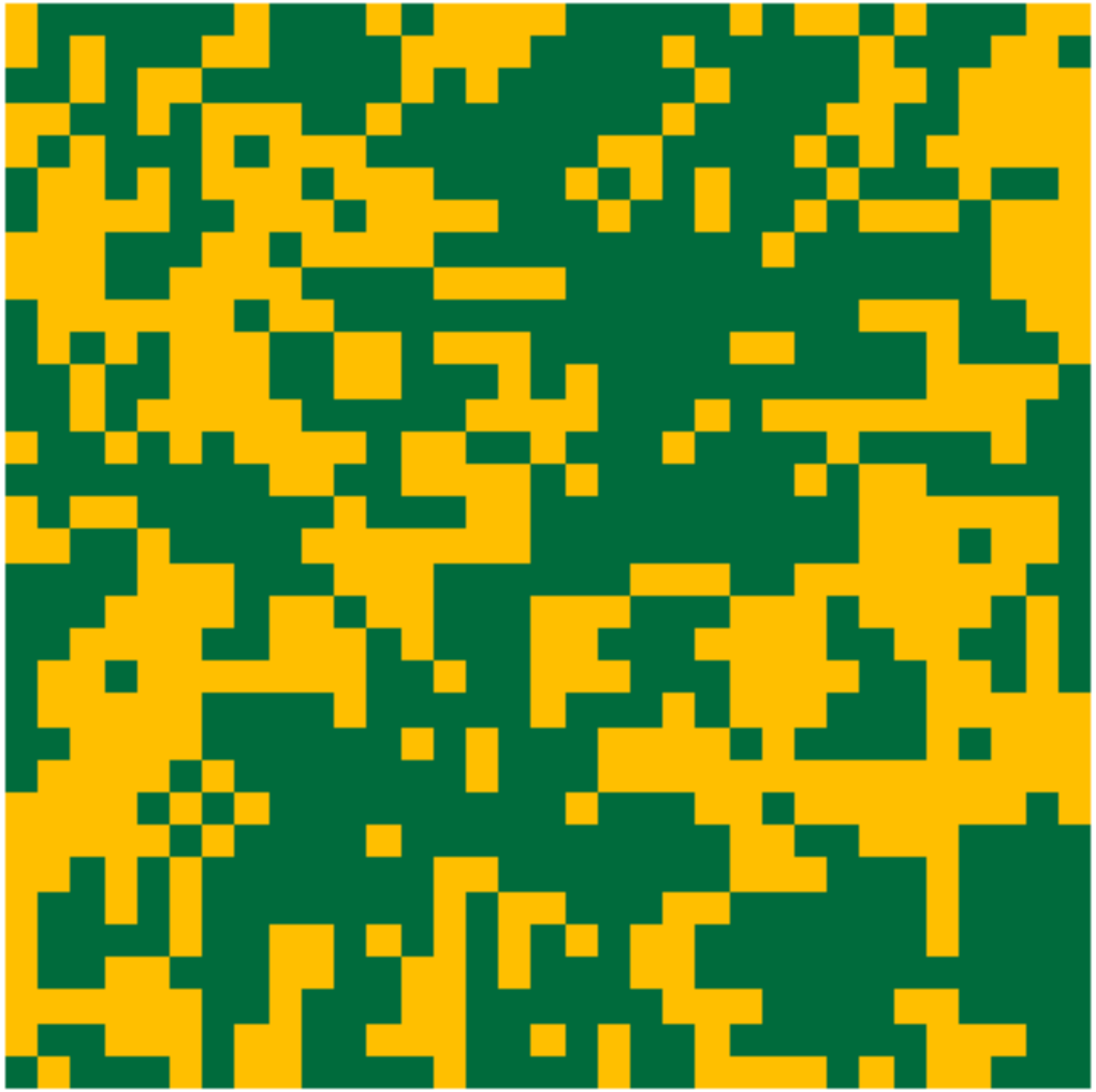
Given a lattice L, some number of iterations N, and an initial labeling σ_0,

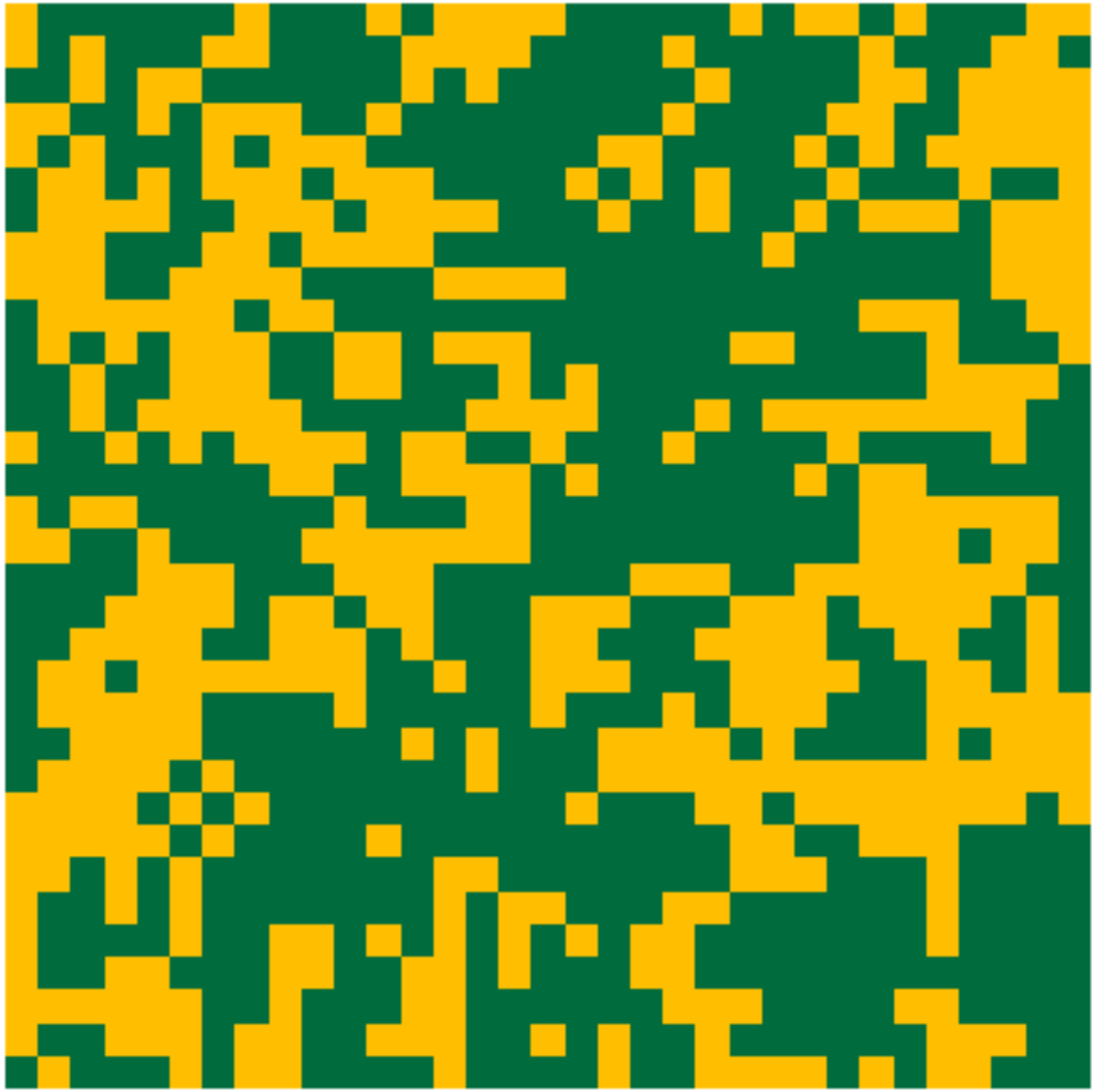
1. for each time t in $\{1, \ldots, N\}$,
   a. initialize a collection C of edges.
   b. for each edge $e = (u,v)$ in G,
      i. include e in C with probability p if $0 = \sigma_{t-1}(v) - \sigma_{t-1}(u) = \sigma_{t-1}(u-v)$;
      ii. ignore e otherwise.
   c. get the *induced subgraph* S from G by ignoring all edges not in C.
   d. for each connected component k in $\kappa(S)$,
      i. uniformly randomly choose a spin value m;
      ii. set $\sigma_t(v) = m$ for all vertices v in k; σ_t is the new spin configuration.











… but can we *generalize* this?

yes — we'll use *topology* to do it.

homology lets us characterize *holes*.

sequences of edges

$$\text{sequences of edges} \xrightarrow[\textit{boundary operator}]{\partial} \text{sequences of vertices}$$

*co*homology lets us characterize *co*holes.

labelings on vertices

$$\text{labelings on vertices} \xrightarrow[\textit{coboundary operator}]{\delta} \textit{sums} \text{ of labelings on edges}$$

labelings on
n-dimensional things

$$\xrightarrow[\text{coboundary operator}]{\delta}$$

sums of labelings on
$(n+1)$-dimensional things

can be written as a matrix!!!!!!!!!!!!!!!!!!!

labelings on
n-dimensional things

$$\xrightarrow[\text{coboundary operator}]{\delta}$$

sums of labelings on
(*n*+1)-dimensional things

Generalized Swendsen-Wang algorithm.

Generalized Swendsen-Wang algorithm.

Given a *cubical complex* L, some number of iterations N, and an initial labeling σ_0,

Generalized Swendsen-Wang algorithm.

Given a *cubical complex* L, some number of iterations N, and an initial labeling σ_0,

1. for teach time t in $\{1, ..., N\}$,

Generalized Swendsen-Wang algorithm.

Given a *cubical complex* L, some number of iterations N, and an initial labeling σ_0,

1. for teach time t in $\{1, ..., N\}$,
   a. initialize an empty collection C of *plaquettes* (i.e. $(n+1)$-dimensional objects)

Generalized Swendsen-Wang algorithm.

Given a *cubical complex* L, some number of iterations N, and an initial labeling σ_0,

1. for teach time t in $\{1, ..., N\}$,
   a. initialize an empty collection C of *plaquettes* (i.e. $(n+1)$-dimensional objects)
   b. for each plaquette $\square$:

Generalized Swendsen-Wang algorithm.

Given a *cubical complex* L, some number of iterations N, and an initial labeling σ_0,

1. for teach time t in $\{1, ..., N\}$,
   a. initialize an empty collection C of *plaquettes* (i.e. $(n+1)$-dimensional objects)
   b. for each plaquette $\square$:
      i. include $\square$ in C with probability p if $\delta(\sigma_{t-1})(\square) = (\sigma_{t-1} \circ \partial)(\square) = 0$

Generalized Swendsen-Wang algorithm.

Given a *cubical complex* L, some number of iterations N, and an initial labeling σ_0,

1. for teach time t in $\{1, \ldots, N\}$,
   a. initialize an empty collection C of *plaquettes* (i.e. $(n+1)$-dimensional objects)
   b. for each plaquette $\square$:
      i. include $\square$ in C with probability p if $\delta(\sigma_{t-1})(\square) = (\sigma_{t-1} \circ \partial)(\square) = 0$
      ii. ignore $\square$ otherwise.

Generalized Swendsen-Wang algorithm.

Given a *cubical complex* L, some number of iterations N, and an initial labeling σ_0,

1. for teach time t in $\{1, \ldots, N\}$,
   a. initialize an empty collection C of *plaquettes* (i.e. $(n+1)$-dimensional objects)
   b. for each plaquette $\square$:
      i. include $\square$ in C with probability p if $\delta(\sigma_{t-1})(\square) = (\sigma_{t-1} \circ \partial)(\square) = 0$
      ii. ignore $\square$ otherwise.
   c. obtain the induced cubical complex S by ignoring all plaquettes in C.

Generalized Swendsen-Wang algorithm.

Given a *cubical complex* L, some number of iterations N, and an initial labeling σ_0,

1. for teach time t in $\{1, \ldots, N\}$,
   a. initialize an empty collection C of *plaquettes* (i.e. $(n+1)$-dimensional objects)
   b. for each plaquette $\square$:
      i. include $\square$ in C with probability p if $\delta(\sigma_{t-1})(\square) = (\sigma_{t-1} \circ \partial)(\square) = 0$
      ii. ignore $\square$ otherwise.
   c. obtain the induced cubical complex S by ignoring all plaquettes in C.
   d. construct the *coboundary matrix* M_δ on S.

Generalized Swendsen-Wang algorithm.

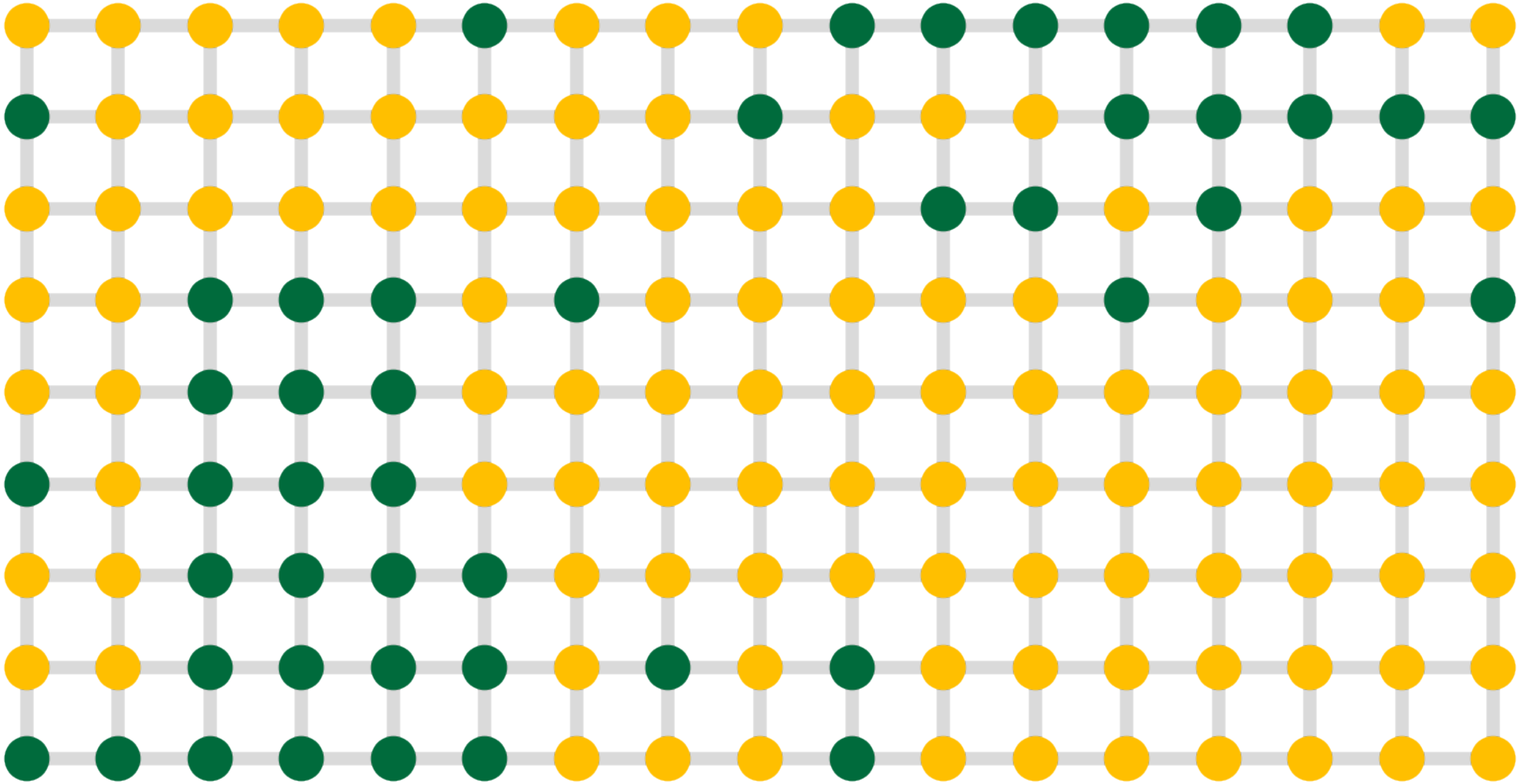
Given a *cubical complex* L, some number of iterations N, and an initial labeling σ_0,

1. for teach time t in $\{1, \ldots, N\}$,
   a. initialize an empty collection C of *plaquettes* (i.e. $(n+1)$-dimensional objects)
   b. for each plaquette $\square$:
      i. include $\square$ in C with probability p if $\delta(\sigma_{t-1})(\square) = (\sigma_{t-1} \circ \partial)(\square) = 0$
      ii. ignore $\square$ otherwise.
   c. obtain the induced cubical complex S by ignoring all plaquettes in C.
   d. construct the *coboundary matrix* M_δ on S.
   e. uniformly randomly sample a new labeling σ_t from the *kernel* of M_δ.

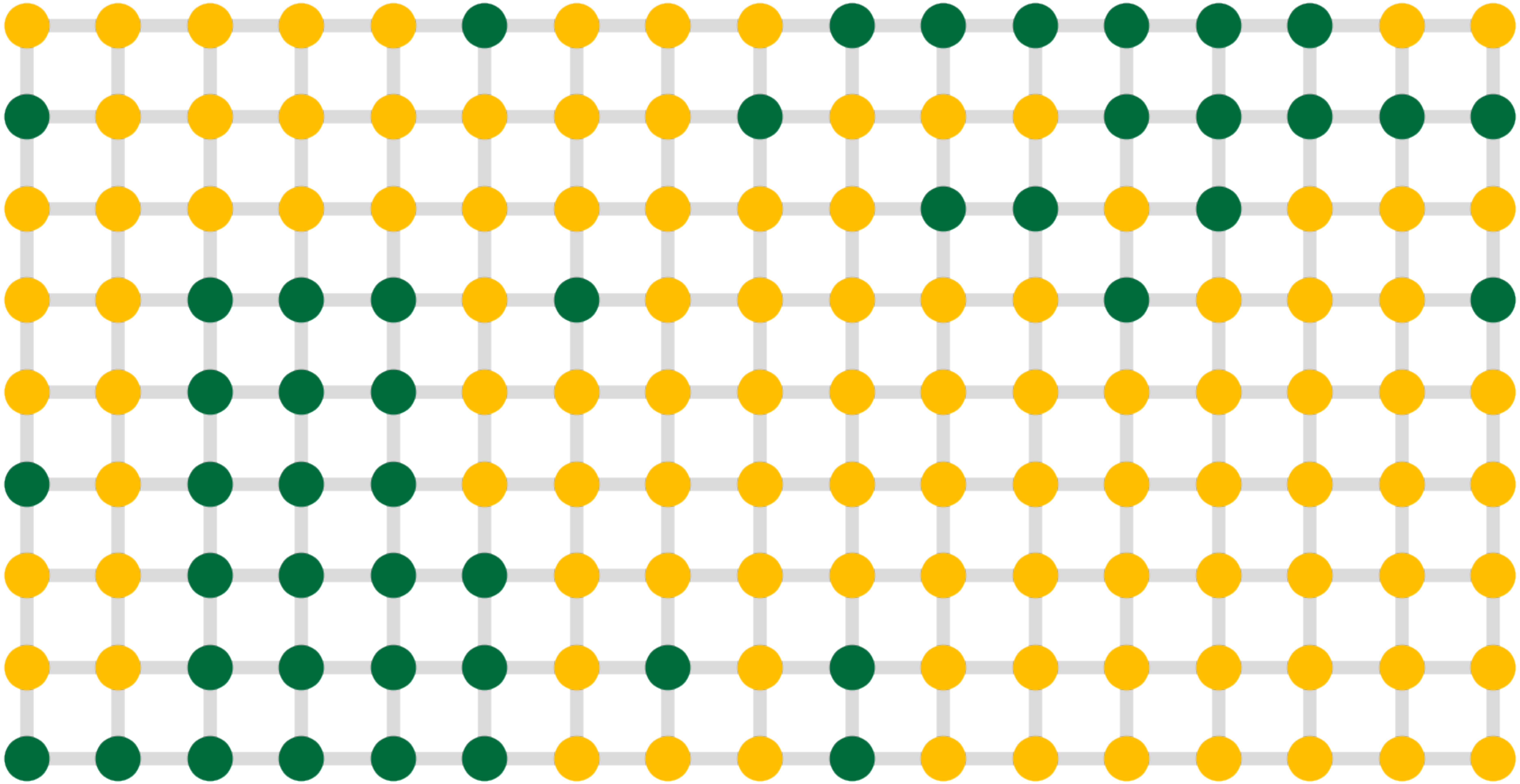
3. speedbumps

M_δ is *very, very* large.

M_δ is *very, very* large… so sampling from its null space is *intense.*



$M_\delta \in \mathrm{Mat}_{153\times 380}(\mathbb{Z})$



M_δ is *very, very* large, and *very very* sparse.

M_δ is *very, very* large, and *very very* sparse... so *storage* is an issue too.

4. future work

thank you!