Worm Algorithms

Youjin Deng

Department of Modern Physics
University of Science and Technology of China
P.R.China

November 6
Hefei

AUSTRALIAN RESEARCH COUNCIL
Centre of Excellence for Mathematics
and Statistics of Complex Systems

References/Collaborators

1. Youjin Deng, Timothy M. Garoni, and Alan D. Sokal, *Dynamic Critical Behavior of the Worm Algorithm for the Ising Model*, Phys. Rev. Lett. 99, 110601 (2007).
2. Wei Zhang, Timothy M. Garoni, and Youjin Deng, *Simulating the fully-packed loop model on the honeycomb lattice with a worm algorithm*, preprint.

How do we efficiently simulate models near criticality?

- Problem: critical slowing-down

How do we efficiently simulate models near criticality?

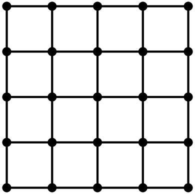
- Problem: critical slowing-down
- The current state-of-the-art: cluster algorithms
  - Swendsen & Wang PRL 1987
  - Use global moves in clever way

How do we efficiently simulate models near criticality?

- Problem: critical slowing-down
- The current state-of-the-art: cluster algorithms
  - Swendsen & Wang PRL 1987
  - Use global moves in clever way
- More recent idea: worm algorithms
  - Prokof'ev & Svistunov PRL 2001
  - Enlarge an Eulerian configuration space to include defects
  - Move the defects via random walk

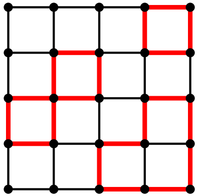
Eulerian subgraphs

- Fix a finite graph $G = (V, E)$



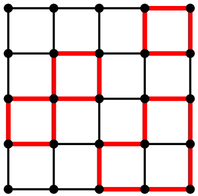
Eulerian subgraphs

- Fix a finite graph $G = (V, E)$
- $A \subseteq E$ is Eulerian if every vertex in (V, A) has even degree



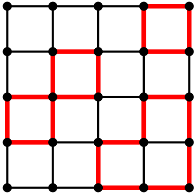
Eulerian subgraphs

- Fix a finite graph $G = (V, E)$
- $A \subseteq E$ is Eulerian if every vertex in (V, A) has even degree
- The cycle space $\mathcal{C}(G) = \{A \subseteq E : A \text{ is Eulerian}\}$



Eulerian subgraphs

- Fix a finite graph $G = (V, E)$
- $A \subseteq E$ is Eulerian if every vertex in (V, A) has even degree
- The cycle space $\mathcal{C}(G) = \{A \subseteq E : A \text{ is Eulerian}\}$

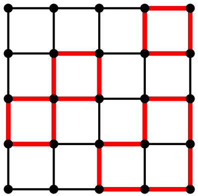


- Consider the Ising model on G

$$Z_{\text{Ising}} = \sum_{\sigma \in \{-1,+1\}^V} \prod_{ij \in E} e^{\beta \sigma_i \sigma_j}$$

Eulerian subgraphs

- Fix a finite graph $G = (V, E)$
- $A \subseteq E$ is Eulerian if every vertex in (V, A) has even degree
- The cycle space $\mathcal{C}(G) = \{A \subseteq E : A \text{ is Eulerian}\}$



- Consider the Ising model on G

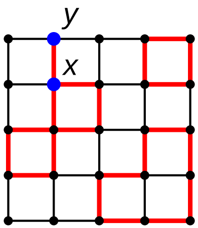
$$Z_{\text{Ising}} = \sum_{\sigma \in \{-1,+1\}^V} \prod_{ij \in E} e^{\beta \sigma_i \sigma_j}$$

- The high-temperature expansion is

$$Z_{\text{Ising}} = \left(2^{|V|} \cosh^{|E|} \beta\right) \sum_{A \in \mathcal{C}(G)} (\tanh \beta)^{|A|}$$

State space for worm dynamics

- Let ∂A be the set of all vertices with odd degree in (V, A)



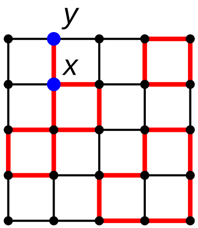
- For distinct $x, y \in V$ define

$$\mathcal{S}_{x,y} = \{A \subseteq E : \partial A = \{x, y\}\}$$
$$\mathcal{S}_{x,x} = \{A \subseteq E : \partial A = \emptyset\}$$

- In this notation $\mathcal{S}_{x,x} = \mathcal{C}(G)$

State space for worm dynamics

- Let ∂A be the set of all vertices with odd degree in (V, A)



- For distinct $x, y \in V$ define

$$\mathcal{S}_{x,y} = \{A \subseteq E : \partial A = \{x, y\}\}$$
$$\mathcal{S}_{x,x} = \{A \subseteq E : \partial A = \emptyset\}$$

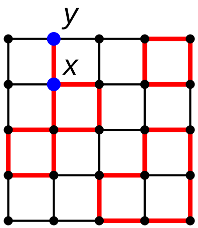
- In this notation $\mathcal{S}_{x,x} = \mathcal{C}(G)$

- State space of worm algorithm is

$$\mathcal{S} = \{(A, x, y) : x, y \in V \text{ and } A \in \mathcal{S}_{x,y}\}$$

State space for worm dynamics

- Let ∂A be the set of all vertices with odd degree in (V, A)



- For distinct $x, y \in V$ define

$$\mathcal{S}_{x,y} = \{A \subseteq E : \partial A = \{x, y\}\}$$
$$\mathcal{S}_{x,x} = \{A \subseteq E : \partial A = \emptyset\}$$

- In this notation $\mathcal{S}_{x,x} = \mathcal{C}(G)$

- State space of worm algorithm is

$$\mathcal{S} = \{(A, x, y) : x, y \in V \text{ and } A \in \mathcal{S}_{x,y}\}$$

- Assign $(A, x, y) \in \mathcal{S}$ probability

$$\pi(A, x, y) \propto d_x\, d_y\, (\tanh \beta)^{|A|}$$

Ising susceptibility

- If $Z := \sum_{A \in \mathcal{C}(G)} w^{|A|}$ and $w := \tanh \beta$

$$Z_{\text{Ising}} = \left(2^{|V|} \cosh^{|E|} \beta\right) Z \qquad \text{Partition function}$$

$$Z \left\langle \sigma_x \sigma_y \right\rangle_{\text{Ising}} = \sum_{A \in \mathcal{S}_{x,y}} w^{|A|} \qquad \text{Two-point function}$$

$$Z \left\langle \mathcal{M}^2 \right\rangle_{\text{Ising}} = \sum_{A \in \mathcal{S}} w^{|A|} \qquad \text{Magnetization}$$

Ising susceptibility

- If $Z := \sum_{A \in \mathcal{C}(G)} w^{|A|}$ and $w := \tanh \beta$

$$Z_{\text{Ising}} = \left(2^{|V|} \cosh^{|E|} \beta\right) Z \qquad \text{Partition function}$$

$$Z \langle \sigma_x \sigma_y \rangle_{\text{Ising}} = \sum_{A \in \mathcal{S}_{x,y}} w^{|A|} \qquad \text{Two-point function}$$

$$Z \langle \mathcal{M}^2 \rangle_{\text{Ising}} = \sum_{A \in \mathcal{S}} w^{|A|} \qquad \text{Magnetization}$$

- If G is translationally invariant then

$$\pi(A, x, y) = w^{|A|} / Z \, V \, \chi$$

Ising susceptibility

- If $Z := \sum_{A\in\mathcal{C}(G)} w^{|A|}$ and $w := \tanh\beta$

$$Z_{\text{Ising}} = \left(2^{|V|}\cosh^{|E|}\beta\right) Z \qquad \text{Partition function}$$

$$Z\,\langle\sigma_x\sigma_y\rangle_{\text{Ising}} = \sum_{A\in\mathcal{S}_{x,y}} w^{|A|} \qquad \text{Two-point function}$$

$$Z\,\langle\mathcal{M}^2\rangle_{\text{Ising}} = \sum_{A\in\mathcal{S}} w^{|A|} \qquad \text{Magnetization}$$

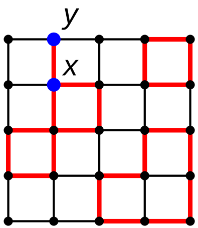
- If G is translationally invariant then

$$\pi(A,x,y) = w^{|A|}/Z\,V\,\chi$$

- Therefore the observable $\mathcal{D}_0(A,x,y) = \delta_{x,y}$ satisfies

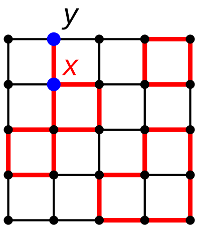
$$\langle\mathcal{D}_0\rangle_\pi = 1/\chi$$

Worm dynamics



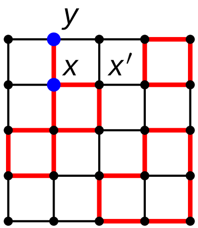
- Start in configuration (A, x, y)

Worm dynamics



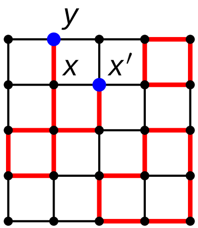
- Start in configuration (A, x, y)
- Pick x or y

Worm dynamics



- Start in configuration (A, x, y)
- Pick x or y
- Pick $x' \sim x$

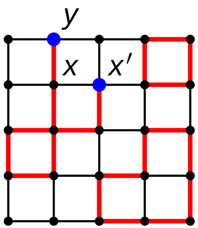
Worm dynamics



- Start in configuration (A, x, y)
- Pick x or y
- Pick $x' \sim x$
- Propose $(A, x, y) \to (A \triangle xx', x', y)$

- If transition removes an edge accept with probability 1

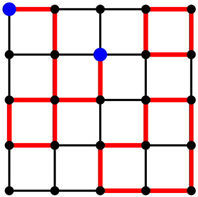
Worm dynamics



- Start in configuration (A, x, y)
- Pick x or y
- Pick $x' \sim x$
- Propose $(A, x, y) \to (A \triangle xx', x', y)$

- If transition removes an edge accept with probability 1
- If transition adds an edge accept with probability w

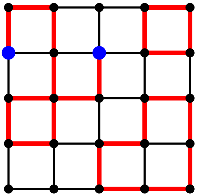
Worm dynamics



- Start in configuration (A, x, y)
- Pick x or y
- Pick $x' \sim x$
- Propose $(A, x, y) \to (A \triangle xx', x', y)$

- If transition removes an edge accept with probability 1
- If transition adds an edge accept with probability w

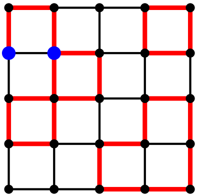
Worm dynamics



- Start in configuration (A, x, y)
- Pick x or y
- Pick $x' \sim x$
- Propose $(A, x, y) \to (A \triangle xx', x', y)$

- If transition removes an edge accept with probability 1
- If transition adds an edge accept with probability w

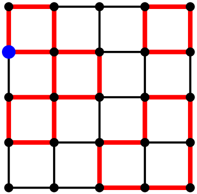
Worm dynamics



- Start in configuration (A, x, y)
- Pick x or y
- Pick $x' \sim x$
- Propose $(A, x, y) \to (A \triangle xx', x', y)$

- If transition removes an edge accept with probability 1
- If transition adds an edge accept with probability w

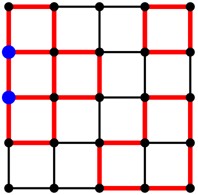
Worm dynamics



- Start in configuration (A, x, y)
- Pick x or y
- Pick $x' \sim x$
- Propose $(A, x, y) \to (A \triangle xx', x', y)$

- If transition removes an edge accept with probability 1
- If transition adds an edge accept with probability w

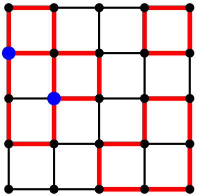
Worm dynamics



- Start in configuration (A, x, y)
- Pick x or y
- Pick $x' \sim x$
- Propose $(A, x, y) \rightarrow (A \triangle xx', x', y)$

- If transition removes an edge accept with probability 1
- If transition adds an edge accept with probability w

Worm dynamics



- Start in configuration (A, x, y)
- Pick x or y
- Pick $x' \sim x$
- Propose $(A, x, y) \rightarrow (A \triangle xx', x', y)$

- If transition removes an edge accept with probability 1
- If transition adds an edge accept with probability w

Transition matrix

- Let G be translationally invariant with degree z
- Worm dynamics corresponds to transition matrix P on $\mathcal{S}$

$$P[(A,x,y) \to (A \triangle xx', x', y)] = \frac{1}{2}\frac{1}{z}\begin{cases} 1, & xx' \in A, \\ w, & xx' \notin A, \end{cases}$$

  - And similarly for y moves. . .
  - All other non-diagonal elements of P are zero

Transition matrix

- Let G be translationally invariant with degree z
- Worm dynamics corresponds to transition matrix P on $\mathcal{S}$

$$P[(A,x,y) \to (A \triangle xx', x', y)] = \frac{1}{2}\frac{1}{z}\begin{cases} 1, & xx' \in A, \\ w, & xx' \notin A, \end{cases}$$

  - And similarly for y moves. . .
  - All other non-diagonal elements of P are zero

Lemma

P is in detailed balance with $\pi(A,x,y) = w^{|A|}/Z\, V\, \chi$

- Can estimate χ by running the worm dynamics

Efficiency

- Worm dynamics provide a valid way to compute χ

Efficiency

- Worm dynamics provide a valid way to compute χ
- But how efficient is the worm algorithm?

Efficiency

- Worm dynamics provide a valid way to compute χ
- But how efficient is the worm algorithm?
- How do we measure efficiency anyway?

Efficiency

- Worm dynamics provide a valid way to compute χ
- But how efficient is the worm algorithm?
- How do we measure efficiency anyway?
- Empirically – measuring autocorrelations

Markov-chain Monte Carlo

- Markov chain
  - State space S, with $|S| < \infty$
  - Transition matrix P
  - Stationary distribution π

Markov-chain Monte Carlo

- Markov chain
  - State space S, with $|S| < \infty$
  - Transition matrix P
  - Stationary distribution π
- Observables (random variables) $X, Y, \ldots$

Markov-chain Monte Carlo

- Markov chain
  - State space S, with $|S| < \infty$
  - Transition matrix P
  - Stationary distribution π
- Observables (random variables) $X, Y, \ldots$
- Simulate Markov chain $s_0 \xrightarrow{P} s_1 \xrightarrow{P} s_2 \xrightarrow{P} \ldots$ with $s_t \in S$

Markov-chain Monte Carlo

- Markov chain
  - State space S, with $|S| < \infty$
  - Transition matrix P
  - Stationary distribution π
- Observables (random variables) $X, Y, \ldots$
- Simulate Markov chain $s_0 \xrightarrow{P} s_1 \xrightarrow{P} s_2 \xrightarrow{P} \ldots$ with $s_t \in S$
- Get time series $X_0, X_1, X_2, \ldots$ with $X_t = X(s_t)$

Markov-chain Monte Carlo

- Markov chain
  - State space S, with $|S| < \infty$
  - Transition matrix P
  - Stationary distribution π
- Observables (random variables) $X, Y, \ldots$
- Simulate Markov chain $s_0 \xrightarrow{P} s_1 \xrightarrow{P} s_2 \xrightarrow{P} \ldots$ with $s_t \in S$
- Get time series $X_0, X_1, X_2, \ldots$ with $X_t = X(s_t)$
- Define the autocorrelation function

$$\rho_X(t) := \frac{\langle X_s X_{s+t} \rangle_\pi - \langle X \rangle_\pi^2}{\mathrm{var}_\pi(X)}$$

Markov-chain Monte Carlo

- Markov chain
  - State space S, with $|S| < \infty$
  - Transition matrix P
  - Stationary distribution π
- Observables (random variables) $X, Y, \ldots$
- Simulate Markov chain $s_0 \xrightarrow{P} s_1 \xrightarrow{P} s_2 \xrightarrow{P} \ldots$ with $s_t \in S$
- Get time series $X_0, X_1, X_2, \ldots$ with $X_t = X(s_t)$
- Define the autocorrelation function

$$\rho_X(t) := \frac{\langle X_s X_{s+t} \rangle_\pi - \langle X \rangle_\pi^2}{\mathrm{var}_\pi(X)}$$

- Stationary process – start “in equilibrium”

Integrated autocorrelation times

- The integrated autocorrelation time

$$\tau_{\mathrm{int},X} := \frac{1}{2} \sum_{t=-\infty}^{\infty} \rho_X(t)$$

Integrated autocorrelation times

- The integrated autocorrelation time

$$\tau_{\text{int},X} := \frac{1}{2} \sum_{t=-\infty}^{\infty} \rho_X(t)$$

- If $\widehat{X}$ is the sample mean of $\{X_t\}_{t=1}^{T}$ then we have

$$\text{var}(\widehat{X}) \sim 2\,\tau_{\text{int},X} \frac{\text{var}(X)}{T}, \qquad T \to \infty$$

Integrated autocorrelation times

- The integrated autocorrelation time

$$\tau_{\mathrm{int},X} := \frac{1}{2} \sum_{t=-\infty}^{\infty} \rho_X(t)$$

- If $\widehat{X}$ is the sample mean of $\{X_t\}_{t=1}^{T}$ then we have

$$\mathrm{var}(\widehat{X}) \sim 2\,\tau_{\mathrm{int},X} \frac{\mathrm{var}(X)}{T}, \qquad T \to \infty$$

- 1 “effectively independent” observation every $2\,\tau_{\mathrm{int},X}$ steps

Exponential autocorrelation times

- $\rho_X(t)$ typically decays exponentially as $t \to \infty$
- The exponential autocorrelation time

$$\tau_{\mathrm{exp},X} := \limsup_{t\to\infty} \frac{t}{-\log|\rho_X(t)|} \quad \text{and} \quad \tau_{\mathrm{exp}} := \sup_X \tau_{\mathrm{exp},X}$$

Exponential autocorrelation times

- $\rho_X(t)$ typically decays exponentially as $t \to \infty$
- The exponential autocorrelation time

$$\tau_{\exp,X} := \limsup_{t\to\infty} \frac{t}{-\log|\rho_X(t)|} \quad \text{and} \quad \tau_{\exp} := \sup_X \tau_{\exp,X}$$

  - Typically $\tau_{\exp,X} = \tau_{\exp} < \infty$ and $\tau_{\mathrm{int},X} \le \tau_{\exp}$ for all X

Exponential autocorrelation times

- $\rho_X(t)$ typically decays exponentially as $t \to \infty$
- The exponential autocorrelation time

$$\tau_{\mathrm{exp},X} := \limsup_{t\to\infty} \frac{t}{-\log|\rho_X(t)|} \quad \text{and} \quad \tau_{\mathrm{exp}} := \sup_X \tau_{\mathrm{exp},X}$$

  - Typically $\tau_{\mathrm{exp},X} = \tau_{\mathrm{exp}} < \infty$ and $\tau_{\mathrm{int},X} \leq \tau_{\mathrm{exp}}$ for all X
- Start the chain with arbitrary distribution α
- Distribution at time t is αP^t

Exponential autocorrelation times

- $\rho_X(t)$ typically decays exponentially as $t \to \infty$
- The exponential autocorrelation time

$$\tau_{\exp,X} := \limsup_{t\to\infty} \frac{t}{-\log|\rho_X(t)|} \quad \text{and} \quad \tau_{\exp} := \sup_X \tau_{\exp,X}$$

  - Typically $\tau_{\exp,X} = \tau_{\exp} < \infty$ and $\tau_{\text{int},X} \le \tau_{\exp}$ for all X
- Start the chain with arbitrary distribution α
- Distribution at time t is αP^t

Lemma

αP^t tends to π with rate bounded by $e^{-t/\tau_{\exp}}$

Critical slowing-down

- Near a critical point the autocorrelation times typically diverge like

$$\tau \sim \xi^z$$

Critical slowing-down

- Near a critical point the autocorrelation times typically diverge like

$$\tau \sim \xi^{z}$$

- More precisely, we have a family of exponents: z_{exp}, and $z_{\mathrm{int},X}$ for each observable X.

Critical slowing-down

- Near a critical point the autocorrelation times typically diverge like
$$\tau \sim \xi^z$$
- More precisely, we have a family of exponents: z_{exp}, and $z_{\text{int},X}$ for each observable X.
- Different algorithms for the same model can have very different z
- E.g. $d = 2$ Ising model
  - Glauber (Metropolis) algorithm $z \approx 2$
  - Swendsen-Wang algorithm $z \approx 0.2$

Worm simulations

- Simulated the critical square-lattice Ising model

Worm simulations

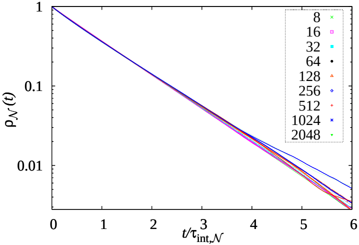
- Simulated the critical square-lattice Ising model
- Focus on two observables:
  - $\mathcal{N}(A, x, y) = |A|$
  - $\mathcal{D}_0(A, x, y) = \delta_{x,y}$
- $\langle \mathcal{N} \rangle$ is "energy-like"
- $\langle \mathcal{D}_0 \rangle = 1/\chi$

Worm simulations

- Simulated the critical square-lattice Ising model
- Focus on two observables:
  - $\mathcal{N}(A,x,y) = |A|$
  - $\mathcal{D}_0(A,x,y) = \delta_{x,y}$
- $\langle\mathcal{N}\rangle$ is “energy-like”
- $\langle\mathcal{D}_0\rangle = 1/\chi$
- Measured observables after every hit (worm update)
- Natural unit of time is one sweep (L^d hits)

Dynamics of $\mathcal{N}$

- $\rho_{\mathcal{N}}(t)$ is almost a perfect exponential

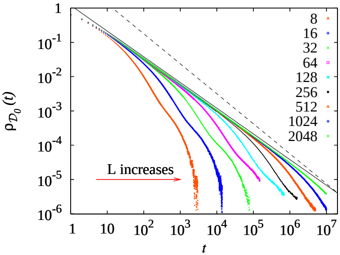


- Scaled time by $\tau_{\mathrm{int},\mathcal{N}}$
- Good data collapse suggests $z_{\mathrm{exp}} \approx z_{\mathrm{int},\mathcal{N}}$
- Fitting $\tau_{\mathrm{int},\mathcal{N}}$ gives $z_{\mathrm{int},\mathcal{N}} \approx 0.379$

- Li-Sokal bound $z_{\mathrm{exp}}, z_{\mathrm{int},\mathcal{N}} \geq \alpha/\nu$ applies to worm dynamics

Dynamics of $\mathcal{D}_0$

- $\rho_{\mathcal{D}_0}(t)$ decays significantly in $O(1)$ hits!



- $\rho_{\mathcal{D}_0}(t) \sim t^{-s}$ with $s \approx 0.75$

- $\mathcal{D}_0$ decorrelates on a totally different time scale to $\mathcal{N}$

Three dimensions

- Qualitatively similar behavior when $d = 3$:

Three dimensions

- Qualitatively similar behavior when $d = 3$:
- $\rho_{\mathcal{D}_0}(t) \sim t^{-s}$
- $s \approx 0.66$

Three dimensions

- Qualitatively similar behavior when $d = 3$:
- $\rho_{\mathcal{D}_0}(t) \sim t^{-s}$
- $s \approx 0.66$
- $\rho_{\mathcal{N}}(t)$ roughly exponential
- $z_{\text{exp}} \approx z_{\text{int},\mathcal{N}} \approx \alpha/\nu \approx 0.174$
- Li-Sokal bound may be sharp for $d = 3$ worm algorithm

Three dimensions

- Qualitatively similar behavior when $d = 3$:
- $\rho_{\mathcal{D}_0}(t) \sim t^{-s}$
- $s \approx 0.66$
- $\rho_{\mathcal{N}}(t)$ roughly exponential
- $z_{\mathrm{exp}} \approx z_{\mathrm{int},\mathcal{N}} \approx \alpha/\nu \approx 0.174$
- Li-Sokal bound may be sharp for $d = 3$ worm algorithm
- Compare Swendsen-Wang $z_{SW} \approx 0.46$

Practical efficiency for square/cubic lattice critical Ising

- Swendsen-Wang seems to outperform worm when $d = 2$

Practical efficiency for square/cubic lattice critical Ising

- Swendsen-Wang seems to outperform worm when $d = 2$
- Efficiency depends on observable, X

Practical efficiency for square/cubic lattice critical Ising

- Swendsen-Wang seems to outperform worm when $d = 2$
- Efficiency depends on observable, X
- A simple way to compare worm and SW is to compute $\kappa = T_{\mathrm{CPU}}\,\mathrm{var}\widehat{X}$ for both algorithms

Practical efficiency for square/cubic lattice critical Ising

- Swendsen-Wang seems to outperform worm when $d = 2$
- Efficiency depends on observable, X
- A simple way to compare worm and SW is to compute $\kappa = T_{\mathrm{CPU}}\,\mathrm{var}\widehat{X}$ for both algorithms
- When $d = 3$ and $X = \mathcal{D}_0$ we find $\kappa_{worm}/\kappa_{SW} \approx L^{-0.33}$
  - With the crossover $\kappa_{worm}/\kappa_{SW} \approx 1$ at around $L \approx 20$

Practical efficiency for square/cubic lattice critical Ising

- Swendsen-Wang seems to outperform worm when $d = 2$
- Efficiency depends on observable, X
- A simple way to compare worm and SW is to compute $\kappa = T_{\mathrm{CPU}}\, \mathrm{var}\widehat{X}$ for both algorithms
- When $d = 3$ and $X = \mathcal{D}_0$ we find $\kappa_{worm}/\kappa_{SW} \approx L^{-0.33}$
  - With the crossover $\kappa_{worm}/\kappa_{SW} \approx 1$ at around $L \approx 20$
- There is also a natural worm estimator for ξ

Practical efficiency for square/cubic lattice critical Ising

- Swendsen-Wang seems to outperform worm when $d = 2$
- Efficiency depends on observable, X
- A simple way to compare worm and SW is to compute $\kappa = T_{\mathrm{CPU}}\,\mathrm{var}\widehat{X}$ for both algorithms
- When $d = 3$ and $X = \mathcal{D}_0$ we find $\kappa_{worm}/\kappa_{SW} \approx L^{-0.33}$
  - With the crossover $\kappa_{worm}/\kappa_{SW} \approx 1$ at around $L \approx 20$
- There is also a natural worm estimator for ξ
- Again SW outperforms worm when $d = 2$

Practical efficiency for square/cubic lattice critical Ising

- Swendsen-Wang seems to outperform worm when $d = 2$
- Efficiency depends on observable, X
- A simple way to compare worm and SW is to compute $\kappa = T_{\text{CPU}}\, \text{var}\widehat{X}$ for both algorithms
- When $d = 3$ and $X = \mathcal{D}_0$ we find $\kappa_{worm}/\kappa_{SW} \approx L^{-0.33}$
  - With the crossover $\kappa_{worm}/\kappa_{SW} \approx 1$ at around $L \approx 20$
- There is also a natural worm estimator for ξ
- Again SW outperforms worm when $d = 2$
- For $d = 3$ we find $\kappa_{worm}/\kappa_{SW} \approx L^{-0.32}$
  - With the crossover $\kappa_{worm}/\kappa_{SW} \approx 1$ at around $L \approx 45$

An alternative perspective on worm dynamics

- Our perspective so far:

  Worm algorithm $\iff$ simulate Ising high-temperature graphs

An alternative perspective on worm dynamics

- Our perspective so far:

  Worm algorithm $\iff$ simulate Ising high-temperature graphs
- Eulerian-subgraph model

An alternative perspective on worm dynamics

- Our perspective so far:

  Worm algorithm $\iff$ simulate Ising high-temperature graphs

- Eulerian-subgraph model
  - State space $\mathcal{C}(G)$

An alternative perspective on worm dynamics

- Our perspective so far:

  Worm algorithm $\iff$ simulate Ising high-temperature graphs
- Eulerian-subgraph model
  - State space $\mathcal{C}(G)$
  - $\mathbb{P}(A) \propto w^{|A|}$

An alternative perspective on worm dynamics

- Our perspective so far:

  Worm algorithm $\iff$ simulate Ising high-temperature graphs

- Eulerian-subgraph model
  - State space $\mathcal{C}(G)$
  - $\mathbb{P}(A) \propto w^{|A|}$
- Run a worm simulation

An alternative perspective on worm dynamics

- Our perspective so far:

  Worm algorithm $\iff$ simulate Ising high-temperature graphs
- Eulerian-subgraph model
  - State space $\mathcal{C}(G)$
  - $\mathbb{P}(A) \propto w^{|A|}$
- Run a worm simulation
- Only observe the chain when $A \in \mathcal{C}(G)$, i.e. when $x = y$

An alternative perspective on worm dynamics

- Our perspective so far:

  Worm algorithm $\iff$ simulate Ising high-temperature graphs
- Eulerian-subgraph model
  - State space $\mathcal{C}(G)$
  - $\mathbb{P}(A) \propto w^{|A|}$
- Run a worm simulation
- Only observe the chain when $A \in \mathcal{C}(G)$, i.e. when $x = y$
- Obtain a new Markov chain $\overline{P}$
- Stationary distribution $\overline{\pi}(A, x, x) \propto f(x)\, w^{|A|}$

An alternative perspective on worm dynamics

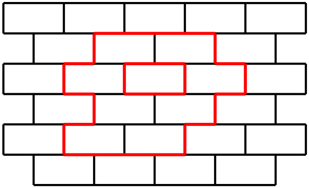
- Our perspective so far:

  Worm algorithm $\iff$ simulate Ising high-temperature graphs
- Eulerian-subgraph model
  - State space $\mathcal{C}(G)$
  - $\mathbb{P}(A) \propto w^{|A|}$
- Run a worm simulation
- Only observe the chain when $A \in \mathcal{C}(G)$, i.e. when $x = y$
- Obtain a new Markov chain $\overline{P}$
- Stationary distribution $\overline{\pi}(A, x, x) \propto f(x)\, w^{|A|}$
- New perspective:

  Worm algorithm $\iff$ simulate Eulerian-subgraph model

Loop models

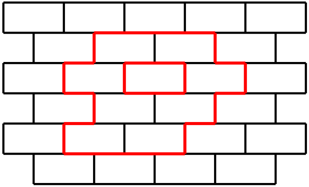
- Honeycomb lattice Eulerian subgraphs = disjoint cycles
- Nienhuis loop model



Loop models

- Honeycomb lattice Eulerian subgraphs = disjoint cycles
- Nienhuis loop model

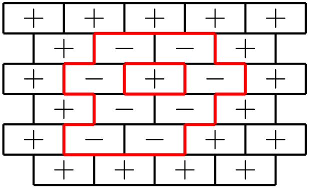
$0 < w < w_c$	Disordered phase
$w_c < w < \infty$	Critical densely-packed phase
$w = +\infty$	Critical fully-packed phase



Loop models

- Honeycomb lattice Eulerian subgraphs = disjoint cycles
- Nienhuis loop model

$$0 < w < w_c \quad \text{Disordered phase}$$
$$w_c < w < \infty \quad \text{Critical densely-packed phase}$$
$$w = +\infty \quad \text{Critical fully-packed phase}$$

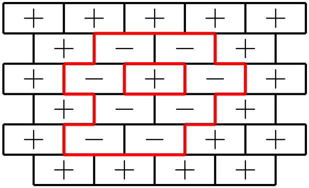


- 1 loop config $\leftrightarrow$ 2 dual spin configs
- Ising spin domains

Loop models

- Honeycomb lattice Eulerian subgraphs = disjoint cycles
- Nienhuis loop model

$$0 < w < w_c \quad \text{Disordered phase}$$
$$w_c < w < \infty \quad \text{Critical densely-packed phase}$$
$$w = +\infty \quad \text{Critical fully-packed phase}$$

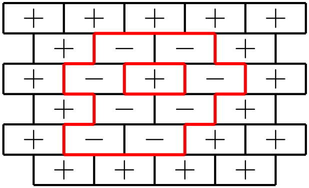


- 1 loop config $\leftrightarrow$ 2 dual spin configs
- Ising spin domains
- $w = e^{-2\beta}$

Loop models

- Honeycomb lattice Eulerian subgraphs = disjoint cycles
- Nienhuis loop model

$$
\begin{aligned}
0 < w < w_c &\quad \text{Disordered phase} \\
w_c < w < \infty &\quad \text{Critical densely-packed phase} \\
w = +\infty &\quad \text{Critical fully-packed phase}
\end{aligned}
$$

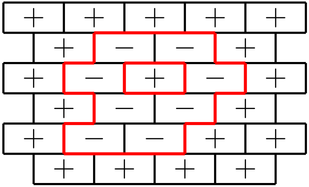


- 1 loop config $\leftrightarrow$ 2 dual spin configs
- Ising spin domains
- $w = e^{-2\beta}$
- $w > 1 \implies$ antiferromagnetic β

Loop models

- Honeycomb lattice Eulerian subgraphs = disjoint cycles
- Nienhuis loop model

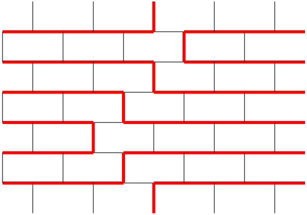
$$\begin{aligned} 0 < w < w_c & \quad \text{Disordered phase} \\ w_c < w < \infty & \quad \text{Critical densely-packed phase} \\ w = +\infty & \quad \text{Critical fully-packed phase} \end{aligned}$$



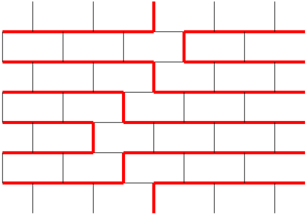
- 1 loop config $\leftrightarrow$ 2 dual spin configs
- Ising spin domains
- $w = e^{-2\beta}$
- $w > 1 \implies$ antiferromagnetic β

- Worm $\iff$ simulate dual Ising domain boundaries

Fully-packed loop model (FPL)

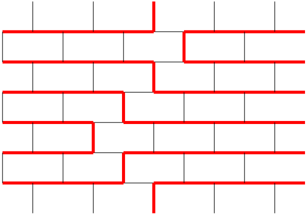


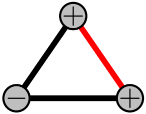
Fully-packed loop model (FPL)



- FPL $\leftrightarrow$ triangular-lattice Ising antiferromagnet at $T = 0$

Fully-packed loop model (FPL)

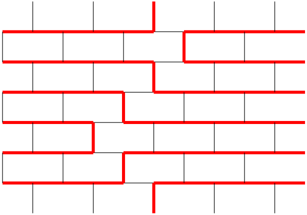


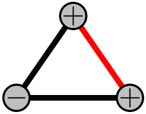


Frustration

- FPL $\leftrightarrow$ triangular-lattice Ising antiferromagnet at $T = 0$
- Frustrated systems hard to simulate

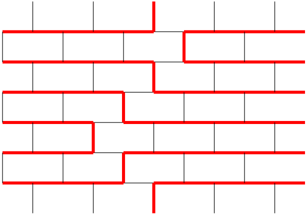
Fully-packed loop model (FPL)

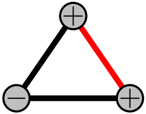




- FPL $\leftrightarrow$ triangular-lattice Ising antiferromagnet at $T = 0$
- Frustrated systems hard to simulate
- Cluster algorithms for frustrated Ising models thought to be non-ergodic at $T = 0$

Fully-packed loop model (FPL)





Frustration

- FPL $\leftrightarrow$ triangular-lattice Ising antiferromagnet at $T = 0$
- Frustrated systems hard to simulate
- Cluster algorithms for frustrated Ising models thought to be non-ergodic at $T = 0$
- Can we use worm instead?

P_∞ has absorbing states

- Standard worm transitions for $w \geq 1$ on z-regular graph:

$$P_w[(A,x,y) \to (A\triangle xx', x', y)] = \frac{1}{2z}\begin{cases} 1/w & xx' \in A \\ 1 & xx' \notin A \end{cases}$$

P_∞ has absorbing states

- Standard worm transitions for $w \geq 1$ on z-regular graph:

$$P_w[(A,x,y) \to (A \triangle xx', x', y)] = \frac{1}{2z} \begin{cases} 1/w & xx' \in A \\ 1 & xx' \notin A \end{cases}$$

- Identity transitions are fixed by normalization:

$$P_w[(A,x,y) \to (A,x,y)] = (1 - 1/w) \frac{d_x(A) + d_y(A)}{2z}$$

P_∞ has absorbing states

- Standard worm transitions for $w \geq 1$ on z-regular graph:

$$P_w[(A,x,y) \to (A \triangle xx', x', y)] = \frac{1}{2z} \begin{cases} 1/w & xx' \in A \\ 1 & xx' \notin A \end{cases}$$

- Identity transitions are fixed by normalization:

$$P_w[(A,x,y) \to (A,x,y)] = (1 - 1/w) \frac{d_x(A) + d_y(A)}{2z}$$

- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$

P_∞ has absorbing states

- Standard worm transitions for $w \geq 1$ on z-regular graph:

$$P_w[(A,x,y) \to (A \triangle xx', x', y)] = \frac{1}{2z} \begin{cases} 1/w & xx' \in A \\ 1 & xx' \notin A \end{cases}$$

- Identity transitions are fixed by normalization:

$$P_w[(A,x,y) \to (A,x,y)] = (1 - 1/w) \frac{d_x(A) + d_y(A)}{2z}$$

- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$
- Cannot use standard worm algorithm when $w = +\infty$

How do we solve the problem?

- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$

How do we solve the problem?

- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$
- But on the honeycomb lattice $z = 3$

How do we solve the problem?

- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$
- But on the honeycomb lattice $z = 3$
- So states with Eulerian A never get stuck

How do we solve the problem?

- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$
- But on the honeycomb lattice $z = 3$
- So states with Eulerian A never get stuck
- To simulate loop models, we only observe the chain when A is Eulerian...

How do we solve the problem?

- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$
- But on the honeycomb lattice $z = 3$
- So states with Eulerian A never get stuck
- To simulate loop models, we only observe the chain when A is Eulerian...
- Try defining new transition matrix P'_∞ such that:

How do we solve the problem?

- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$
- But on the honeycomb lattice $z = 3$
- So states with Eulerian A never get stuck
- To simulate loop models, we only observe the chain when A is Eulerian...
- Try defining new transition matrix P'_∞ such that:
  - $P'_\infty[(A,x,x) \to \cdot] = \lim_{w\to\infty} P_w[(A,x,x) \to \cdot]$

How do we solve the problem?

- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$
- But on the honeycomb lattice $z = 3$
- So states with Eulerian A never get stuck
- To simulate loop models, we only observe the chain when A is Eulerian...
- Try defining new transition matrix P'_∞ such that:
  - $P'_\infty[(A,x,x) \to \cdot] = \lim_{w\to\infty} P_w[(A,x,x) \to \cdot]$
  - $P'_\infty[(A,x,y) \to (A,x,y)] = 0$ when $x \neq y$

How do we solve the problem?

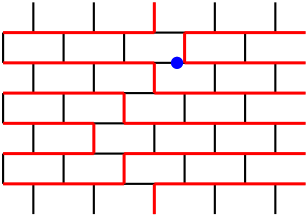
- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$
- But on the honeycomb lattice $z = 3$
- So states with Eulerian A never get stuck
- To simulate loop models, we only observe the chain when A is Eulerian...
- Try defining new transition matrix P'_∞ such that:
  - $P'_\infty[(A,x,x) \to \cdot] = \lim_{w\to\infty} P_w[(A,x,x) \to \cdot]$
  - $P'_\infty[(A,x,y) \to (A,x,y)] = 0$ when $x \neq y$
- This will get rid of the absorbing states...

How do we solve the problem?

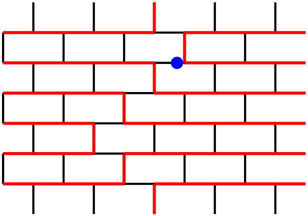
- $P_\infty[(A,x,y) \to (A,x,y)] = 1$ whenever $d_x(A) = d_y(A) = z$
- But on the honeycomb lattice $z = 3$
- So states with Eulerian A never get stuck
- To simulate loop models, we only observe the chain when A is Eulerian...
- Try defining new transition matrix P'_∞ such that:
  - $P'_\infty[(A,x,x) \to \cdot] = \lim_{w\to\infty} P_w[(A,x,x) \to \cdot]$
  - $P'_\infty[(A,x,y) \to (A,x,y)] = 0$ when $x \neq y$
- This will get rid of the absorbing states...
- If we are lucky P'_∞ will simulate the FPL...

Worm algorithm for honeycomb-lattice FPL

- Start in configuration (A, x, y)

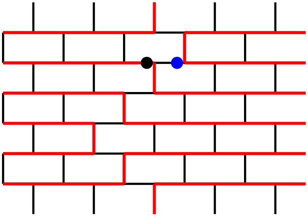


Worm algorithm for honeycomb-lattice FPL



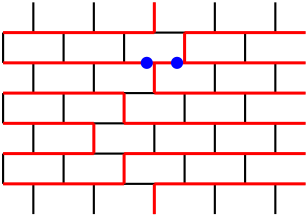
- Start in configuration (A, x, y)
- If $x = y$

Worm algorithm for honeycomb-lattice FPL



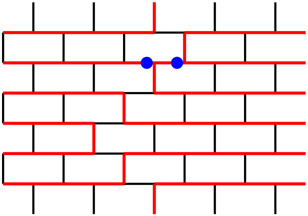
- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$

Worm algorithm for honeycomb-lattice FPL



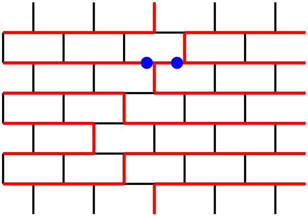
- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \to (A \cup xx', x', x)$ or $(A, x, x) \to (A \cup xx', x, x')$

Worm algorithm for honeycomb-lattice FPL



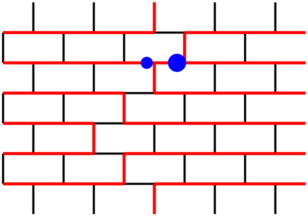
- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \to (A \cup xx', x', x)$ or $(A, x, x) \to (A \cup xx', x, x')$
  - Else $(A, x, x) \to (A, x, x)$

Worm algorithm for honeycomb-lattice FPL



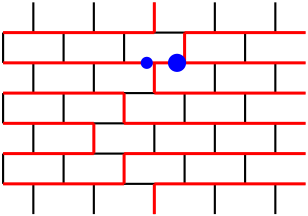
- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \to (A \cup xx', x', x)$ or $(A, x, x) \to (A \cup xx', x, x')$
  - Else $(A, x, x) \to (A, x, x)$
- Else if $x \neq y$

Worm algorithm for honeycomb-lattice FPL



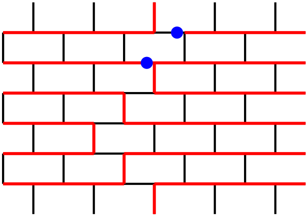
- Start in configuration (A,x,y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A,x,x) \to (A \cup xx', x', x)$ or $(A,x,x) \to (A \cup xx', x, x')$
  - Else $(A,x,x) \to (A,x,x)$
- Else if $x \neq y$
  - Pick x or y (say x)

Worm algorithm for honeycomb-lattice FPL



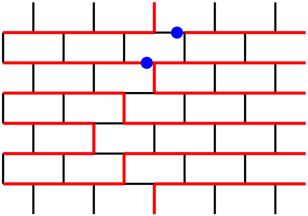
- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \to (A \cup xx', x', x)$ or $(A, x, x) \to (A \cup xx', x, x')$
  - Else $(A, x, x) \to (A, x, x)$
- Else if $x \neq y$
  - Pick x or y (say x)
  - If $d_x(A) = 3$
    - Pick one of the three $xx' \in A$
    - $(A, x, y) \to (A \setminus xx', x', y)$

Worm algorithm for honeycomb-lattice FPL



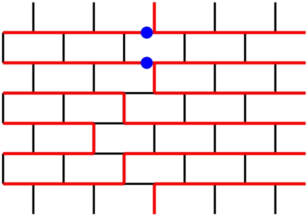
- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \rightarrow (A \cup xx', x', x)$ or $(A, x, x) \rightarrow (A \cup xx', x, x')$
  - Else $(A, x, x) \rightarrow (A, x, x)$
- Else if $x \neq y$
  - Pick x or y (say x)
  - If $d_x(A) = 3$
    - Pick one of the three $xx' \in A$
    - $(A, x, y) \rightarrow (A \setminus xx', x', y)$

Worm algorithm for honeycomb-lattice FPL



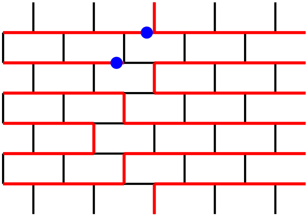
- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \to (A \cup xx', x', x)$ or $(A, x, x) \to (A \cup xx', x, x')$
  - Else $(A, x, x) \to (A, x, x)$
- Else if $x \neq y$
  - Pick x or y (say x)
  - If $d_x(A) = 3$
    - Pick one of the three $xx' \in A$
    - $(A, x, y) \to (A \setminus xx', x', y)$
  - Else if $d_x(A) = 1$
    - Pick one of the two $xx' \notin A$
    - $(A, x, y) \to (A \cup xx', x', y)$

Worm algorithm for honeycomb-lattice FPL



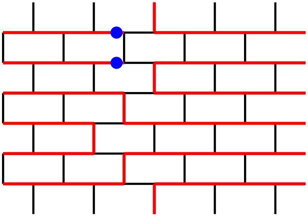
- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \to (A \cup xx', x', x)$ or $(A, x, x) \to (A \cup xx', x, x')$
  - Else $(A, x, x) \to (A, x, x)$
- Else if $x \neq y$
  - Pick x or y (say x)
  - If $d_x(A) = 3$
    - Pick one of the three $xx' \in A$
    - $(A, x, y) \to (A \setminus xx', x', y)$
  - Else if $d_x(A) = 1$
    - Pick one of the two $xx' \notin A$
    - $(A, x, y) \to (A \cup xx', x', y)$

Worm algorithm for honeycomb-lattice FPL



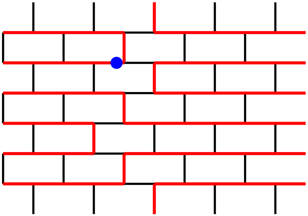
- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \to (A \cup xx', x', x)$ or $(A, x, x) \to (A \cup xx', x, x')$
  - Else $(A, x, x) \to (A, x, x)$
- Else if $x \neq y$
  - Pick x or y (say x)
  - If $d_x(A) = 3$
    - Pick one of the three $xx' \in A$
    - $(A, x, y) \to (A \setminus xx', x', y)$
  - Else if $d_x(A) = 1$
    - Pick one of the two $xx' \notin A$
    - $(A, x, y) \to (A \cup xx', x', y)$

Worm algorithm for honeycomb-lattice FPL



- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \to (A \cup xx', x', x)$ or $(A, x, x) \to (A \cup xx', x, x')$
  - Else $(A, x, x) \to (A, x, x)$
- Else if $x \neq y$
  - Pick x or y (say x)
  - If $d_x(A) = 3$
    - Pick one of the three $xx' \in A$
    - $(A, x, y) \to (A \setminus xx', x', y)$
  - Else if $d_x(A) = 1$
    - Pick one of the two $xx' \notin A$
    - $(A, x, y) \to (A \cup xx', x', y)$

Worm algorithm for honeycomb-lattice FPL



- Start in configuration (A, x, y)
- If $x = y$
  - Pick $x' \sim x$
  - If $xx' \notin A$
    - $(A, x, x) \to (A \cup xx', x', x)$ or $(A, x, x) \to (A \cup xx', x, x')$
  - Else $(A, x, x) \to (A, x, x)$
- Else if $x \neq y$
  - Pick x or y (say x)
  - If $d_x(A) = 3$
    - Pick one of the three $xx' \in A$
    - $(A, x, y) \to (A \setminus xx', x', y)$
  - Else if $d_x(A) = 1$
    - Pick one of the two $xx' \notin A$
    - $(A, x, y) \to (A \cup xx', x', y)$

Transition matrix

- $x \neq y$

$$P'_\infty[(A,x,y) \to (A \triangle xx', x', y)] = \begin{cases} 1/6 & d_x(A) = 3 \\ 1/4 & xx' \notin A \end{cases}$$

- $x = y$

$$P'_\infty[(A,x,x) \to (A,x,x)] = \frac{d_x(A)}{3}$$

$$P'_\infty[(A,x,x) \to (A \cup xx', x', x)] = \frac{1}{6}$$

Transition matrix

- $x \neq y$

$$P'_\infty[(A,x,y) \to (A \triangle xx', x', y)] = \begin{cases} 1/6 & d_x(A) = 3 \\ 1/4 & xx' \notin A \end{cases}$$

- $x = y$

$$P'_\infty[(A,x,x) \to (A,x,x)] = \frac{d_x(A)}{3}$$

$$P'_\infty[(A,x,x) \to (A \cup xx', x', x)] = \frac{1}{6}$$

Theorem

The set of states in $\mathcal{S}$ with no isolated vertices is recurrent and irreducible, its complement is transient, and the stationary distribution of $\overline{P'_\infty}$ is uniform on the fully-packed configurations

Transition matrix

- $x \neq y$

$$P'_\infty[(A,x,y) \to (A \triangle xx', x', y)] = \begin{cases} 1/6 & d_x(A) = 3 \\ 1/4 & xx' \notin A \end{cases}$$

- $x = y$

$$P'_\infty[(A,x,x) \to (A,x,x)] = \frac{d_x(A)}{3}$$

$$P'_\infty[(A,x,x) \to (A \cup xx', x', x)] = \frac{1}{6}$$

Theorem

The set of states in $\mathcal{S}$ with no isolated vertices is recurrent and irreducible, its complement is transient, and the stationary distribution of $\overline{P'_\infty}$ is uniform on the fully-packed configurations

- Therefore P'_∞ correctly simulates the FPL

Worm simulations of honeycomb-lattice FPL

- Simulated the honeycomb-lattice FPL

Worm simulations of honeycomb-lattice FPL

- Simulated the honeycomb-lattice FPL
- Again observed multiple time scales

Worm simulations of honeycomb-lattice FPL

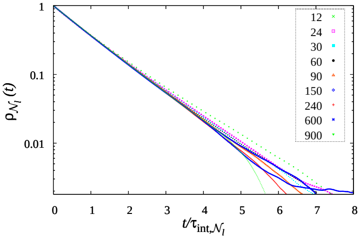
- Simulated the honeycomb-lattice FPL
- Again observed multiple time scales
- $\mathcal{N}_l(A, x, y) =$ number of loops (cyclomatic number)
- $\langle \mathcal{N}_l \rangle$ is "energy-like"

Worm simulations of honeycomb-lattice FPL

- Simulated the honeycomb-lattice FPL
- Again observed multiple time scales
- $\mathcal{N}_l(A,x,y) =$ number of loops (cyclomatic number)
- $\langle \mathcal{N}_l \rangle$ is "energy-like"
- $\mathcal{N}_l$ is the slowest mode observed

Dynamics of $\mathcal{N}_l$

- $\rho_{\mathcal{N}_l}(t)$ is almost a perfect exponential



- Scaled time by $\tau_{\mathrm{int},\mathcal{N}_l}$
- Good data collapse suggests $z_{\mathrm{exp}} \approx z_{\mathrm{int},\mathcal{N}_l}$
- Fitting $\tau_{\mathrm{int},\mathcal{N}_l}$ gives $z_{\mathrm{int},\mathcal{N}_l} = 0.515(8)$

Summary

- Standard worm algorithm for Ising high-temperature graphs outperforms SW in three dimensions

Summary

- Standard worm algorithm for Ising high-temperature graphs outperforms SW in three dimensions
- Worm decorrelates on different time scales – depends on observable

Summary

- Standard worm algorithm for Ising high-temperature graphs outperforms SW in three dimensions
- Worm decorrelates on different time scales – depends on observable
- Modified worm algorithm is demonstrated to be valid for honeycomb lattice FPL
- Dynamic exponent $z \approx 0.5$

Summary

- Standard worm algorithm for Ising high-temperature graphs outperforms SW in three dimensions
- Worm decorrelates on different time scales – depends on observable
- Modified worm algorithm is demonstrated to be valid for honeycomb lattice FPL
- Dynamic exponent $z \approx 0.5$
- By contrast, even the best cluster algorithms for frustrated Ising models are thought to be non-ergodic at $T = 0$