Efficient implementation of connectivity changing moves for dense polymers

Nathan Clisby
Swinburne University of Technology

CompPhys17
Leipzig University
November 30, 2017

Dense polymers

- Interested in both polymer melts and dense configurations of individual polymers.

[1] P. G. de Gennes: Scaling Concepts in Polymer Physics, 1979.

[2] J. P. Wittmer et al.: Why polymer chains in a melt are not random walks, in: EPL (Europhysics Letters) 77 (2007), p. 56003.

Dense polymers

- Interested in both polymer melts and dense configurations of individual polymers.
- In a melt each individual chain behaves like a simple random walk and obeys Gaussian statistics[1], although bond correlation function decays with a power law.[2]

[1] P. G. de Gennes: Scaling Concepts in Polymer Physics, 1979.

[2] J. P. Wittmer et al.: Why polymer chains in a melt are not random walks, in: EPL (Europhysics Letters) 77 (2007), p. 56003.

Dense polymers

- Interested in both polymer melts and dense configurations of individual polymers.
- In a melt each individual chain behaves like a simple random walk and obeys Gaussian statistics[1], although bond correlation function decays with a power law.[2]
- Dynamics (including molecular dynamics) for long polymers very slow due to entanglement.

[1] P. G. de Gennes: Scaling Concepts in Polymer Physics, 1979.

[2] J. P. Wittmer et al.: Why polymer chains in a melt are not random walks, in: EPL (Europhysics Letters) 77 (2007), p. 56003.

Dense polymers

- Interested in both polymer melts and dense configurations of individual polymers.
- In a melt each individual chain behaves like a simple random walk and obeys Gaussian statistics[1], although bond correlation function decays with a power law.[2]
- Dynamics (including molecular dynamics) for long polymers very slow due to entanglement.
- Monte Carlo algorithms may allow for dramatically better sampling rates for equilibrium systems.

[1] P. G. de Gennes: Scaling Concepts in Polymer Physics, 1979.

[2] J. P. Wittmer et al.: Why polymer chains in a melt are not random walks, in: EPL (Europhysics Letters) 77 (2007), p. 56003.

Fast global moves

- We will implement a Markov chain Monte Carlo algorithm.

Fast global moves

- We will implement a Markov chain Monte Carlo algorithm.
- For computational efficiency, want it to be:

Fast global moves

- We will implement a Markov chain Monte Carlo algorithm.
- For computational efficiency, want it to be:
  - Global, so that it moves rapidly around the state space.

Fast global moves

- We will implement a Markov chain Monte Carlo algorithm.
- For computational efficiency, want it to be:
  - Global, so that it moves rapidly around the state space.
  - Fast! Ideally, for system of size N want to be able to perform the move in CPU time $o(N)$.

Fast global moves

- We will implement a Markov chain Monte Carlo algorithm.
- For computational efficiency, want it to be:
  - Global, so that it moves rapidly around the state space.
  - Fast! Ideally, for system of size N want to be able to perform the move in CPU time $o(N)$.
- Ideally, want a move and implementation such that τ_{int} for a suitable global observable, in CPU time units, scales as less than the system size.

Fast global moves

- We will implement a Markov chain Monte Carlo algorithm.
- For computational efficiency, want it to be:
  - Global, so that it moves rapidly around the state space.
  - Fast! Ideally, for system of size N want to be able to perform the move in CPU time $o(N)$.
- Ideally, want a move and implementation such that τ_{int} for a suitable global observable, in CPU time units, scales as less than the system size.
- Will now give an example: the pivot algorithm for self-avoiding walks.

Self-avoiding walk model

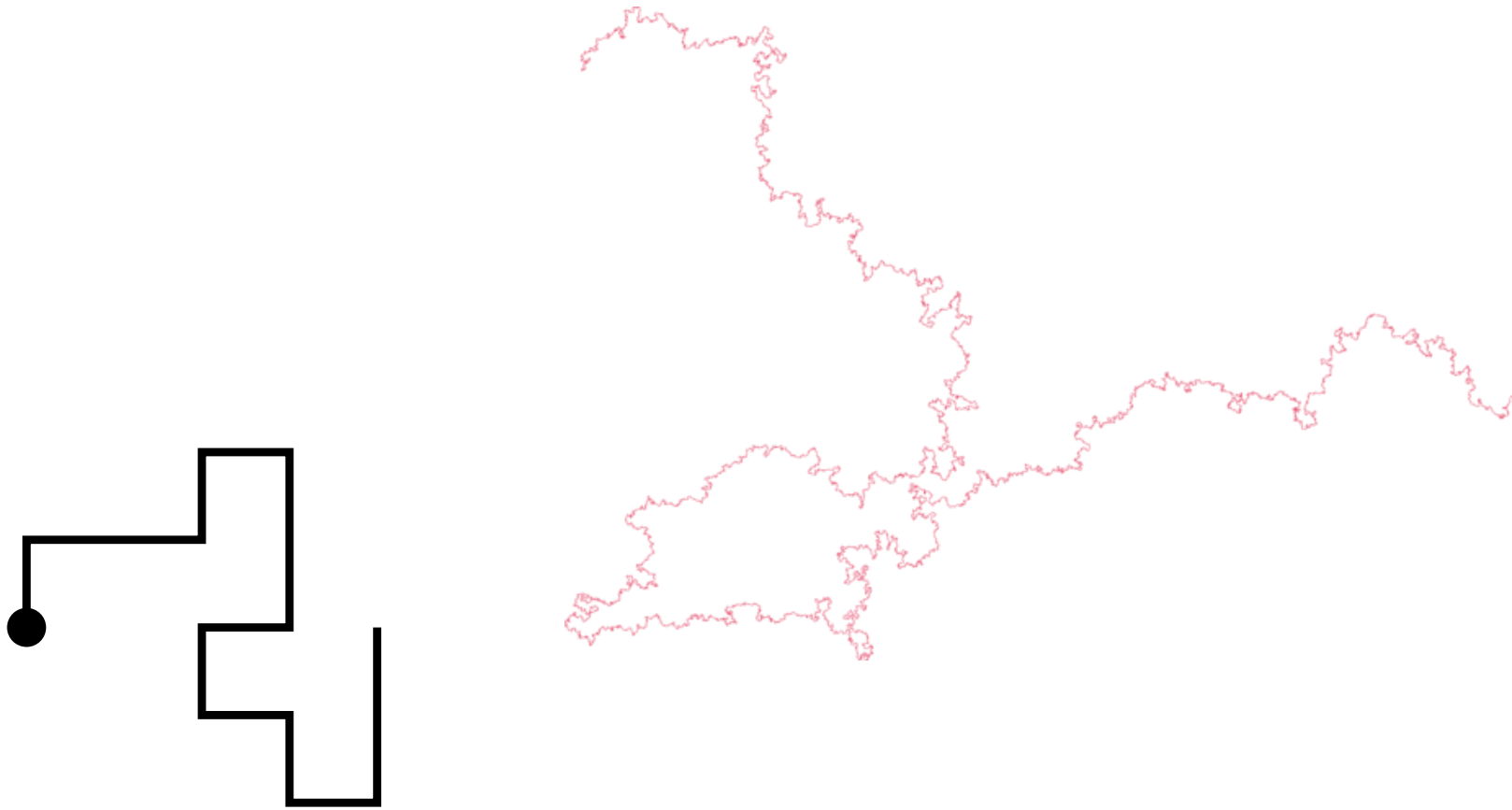
- Self-avoiding walk (SAW): a walk between nearest neighbours on a graph that avoids itself.

Self-avoiding walk model

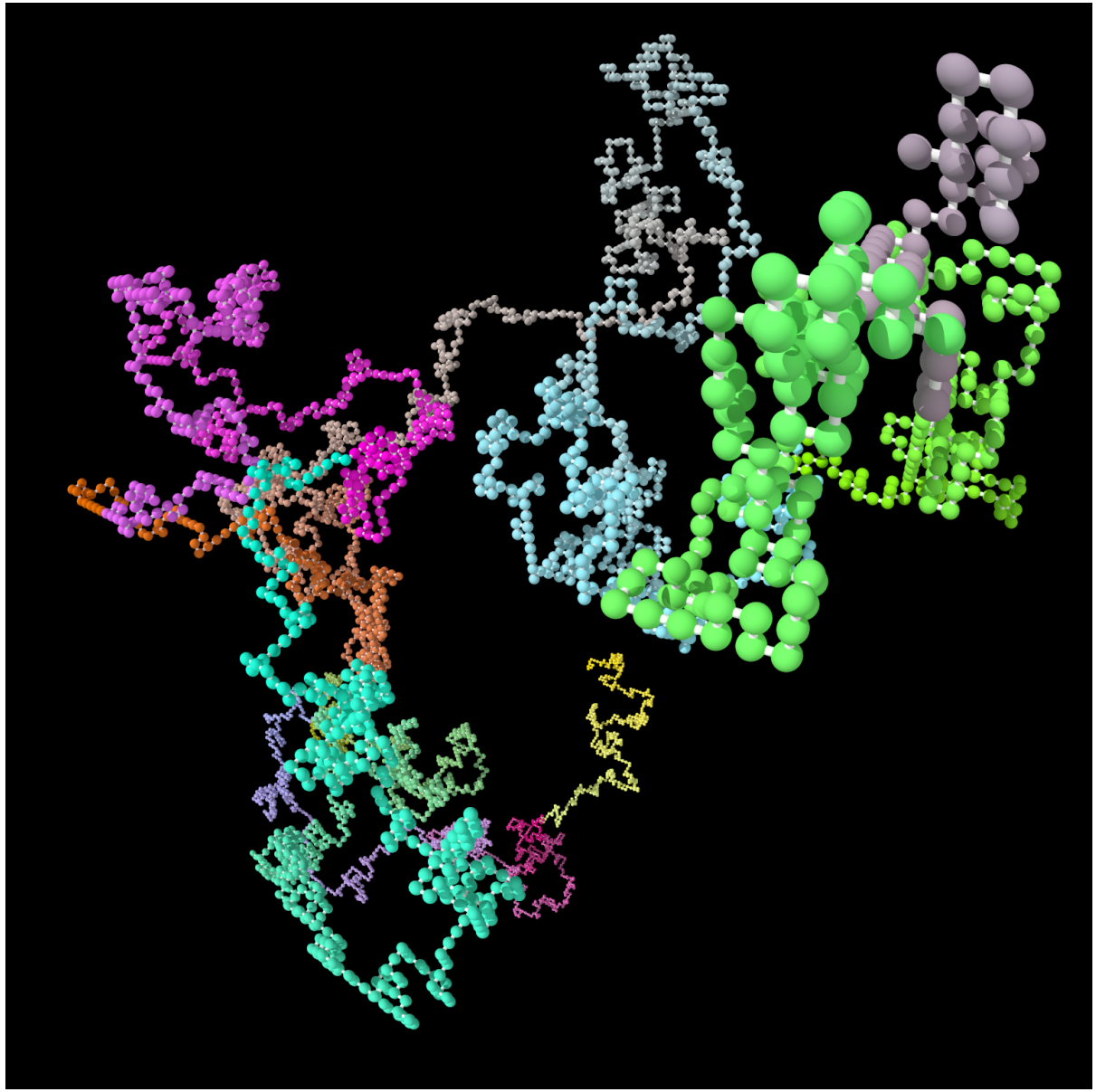
- Self-avoiding walk (SAW): a walk between nearest neighbours on a graph that avoids itself.
- Graph: typically $\mathbb{Z}^d$, but sometimes random networks, percolation clusters.

Self-avoiding walk model

- Self-avoiding walk (SAW): a walk between nearest neighbours on a graph that avoids itself.
- Graph: typically $\mathbb{Z}^d$, but sometimes random networks, percolation clusters.
- Models dilute solution of polymers in a good solvent.



Self-avoiding walks of 15 and 2^{25} steps.



Self-avoiding walk model

- Mean-squared end-to-end distance for SAWs of length N:

$$\langle R_{\mathrm{E}}^2 \rangle_N = D_{\mathrm{E}} N^{2\nu} \left(1 + O(N^{-\Delta_1})\right)$$

Self-avoiding walk model

- Mean-squared end-to-end distance for SAWs of length N:

$$\langle R_{\mathrm{E}}^2 \rangle_N = D_{\mathrm{E}} N^{2\nu} \left(1 + O(N^{-\Delta_1})\right)$$

- Flory exponent ν is universal; same for SAWs and real world polymers.

Pivot algorithm

- The pivot algorithm[3] is the fastest method for sampling SAWs.

[3] Neal Madras/Alan D. Sokal: The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk, in: J. Stat. Phys. 50 (1988), pp. 109–186.

Pivot algorithm

- The pivot algorithm[3] is the fastest method for sampling SAWs.
- Procedure:

[3] Neal Madras/Alan D. Sokal: The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk, in: J. Stat. Phys. 50 (1988), pp. 109–186.

Pivot algorithm

- The pivot algorithm[3] is the fastest method for sampling SAWs.
- Procedure:
  - Choose a pivot site at random

[3] Neal Madras/Alan D. Sokal: The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk, in: J. Stat. Phys. 50 (1988), pp. 109–186.

Pivot algorithm

- The pivot algorithm[3] is the fastest method for sampling SAWs.
- Procedure:
  - Choose a pivot site at random
  - Then rotate or reflect one of the two parts of the walk.

[3] Neal Madras/Alan D. Sokal: The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk, in: J. Stat. Phys. 50 (1988), pp. 109–186.

Pivot algorithm

- The pivot algorithm[3] is the fastest method for sampling SAWs.
- Procedure:
  - Choose a pivot site at random
  - Then rotate or reflect one of the two parts of the walk.
  - Retain new walk if it is self-avoiding, otherwise restore original walk.

[3] Neal Madras/Alan D. Sokal: The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk, in: J. Stat. Phys. 50 (1988), pp. 109–186.

Pivot algorithm

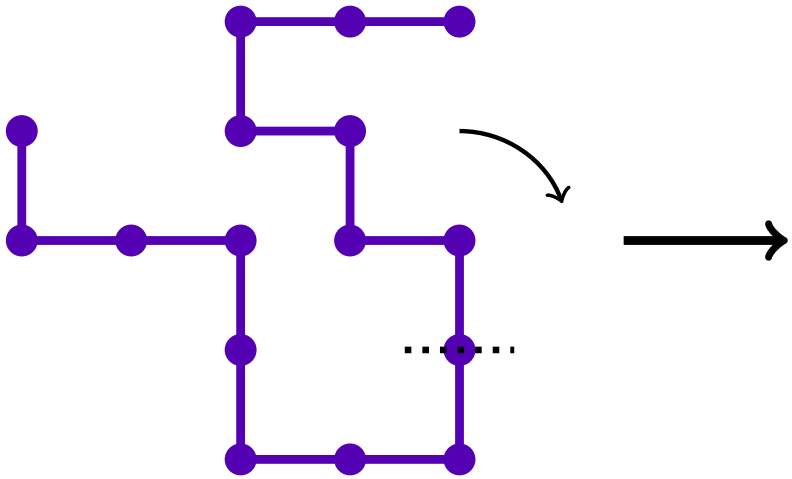
- The pivot algorithm[3] is the fastest method for sampling SAWs.
- Procedure:
  - Choose a pivot site at random
  - Then rotate or reflect one of the two parts of the walk.
  - Retain new walk if it is self-avoiding, otherwise restore original walk.
- "Global" because on average half of the monomers are moved.

[3] Neal Madras/Alan D. Sokal: The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk, in: J. Stat. Phys. 50 (1988), pp. 109–186.

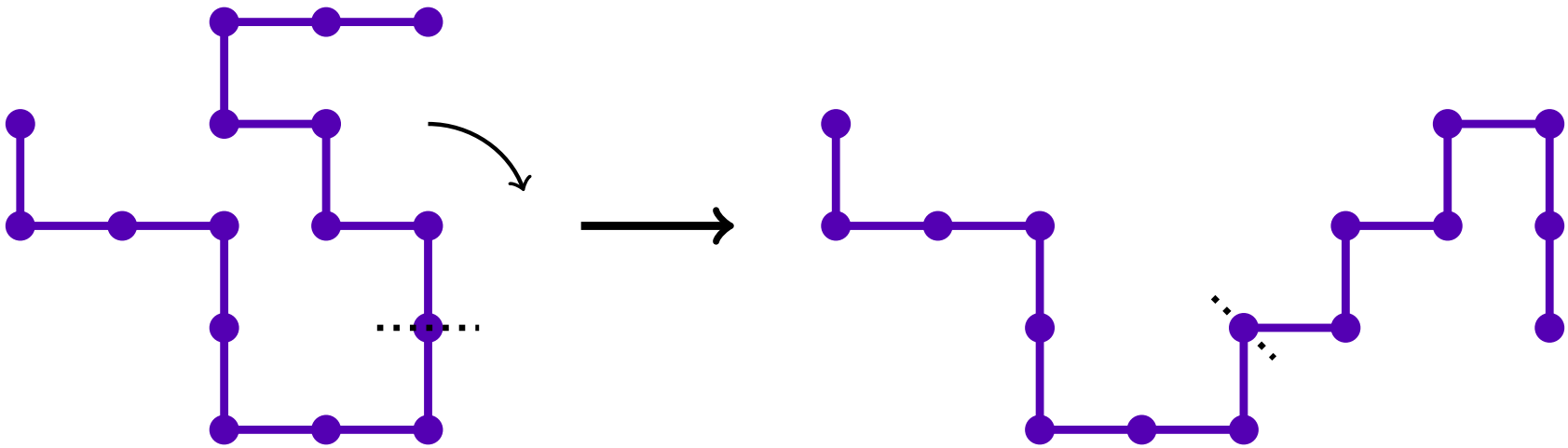
Pivot algorithm

- The pivot algorithm[3] is the fastest method for sampling SAWs.
- Procedure:
  - Choose a pivot site at random
  - Then rotate or reflect one of the two parts of the walk.
  - Retain new walk if it is self-avoiding, otherwise restore original walk.
- "Global" because on average half of the monomers are moved.
- Ergodic, samples SAWs uniformly at random.

[3] Neal Madras/Alan D. Sokal: The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk, in: J. Stat. Phys. 50 (1988), pp. 109–186.



Example pivot move



Example pivot move

- Time $O(N)$ to write down an N-step walk, so this must be best possible for pivot move?

[4] Tom Kennedy: A faster implementation of the pivot algorithm for self-avoiding walks, in: J. Stat. Phys. 106 (2002), pp. 407–429.

[5] Nathan Clisby: Accurate Estimate of the Critical Exponent ν for Self-Avoiding Walks via a Fast Implementation of the Pivot Algorithm, in: Phys. Rev. Lett. 104 (2010), p. 055702.

- Time $O(N)$ to write down an N-step walk, so this must be best possible for pivot move?
- In fact, don't need to write down! CPU time $O(N^{0.74})$[4].

[4] Tom Kennedy: A faster implementation of the pivot algorithm for self-avoiding walks, in: J. Stat. Phys. 106 (2002), pp. 407–429.

[5] Nathan Clisby: Accurate Estimate of the Critical Exponent ν for Self-Avoiding Walks via a Fast Implementation of the Pivot Algorithm, in: Phys. Rev. Lett. 104 (2010), p. 055702.

- Time $O(N)$ to write down an N-step walk, so this must be best possible for pivot move?
- In fact, don't need to write down! CPU time $O(N^{0.74})$[4].
- Just verify that walk is self-avoiding after a pivot move, update information about global observables.

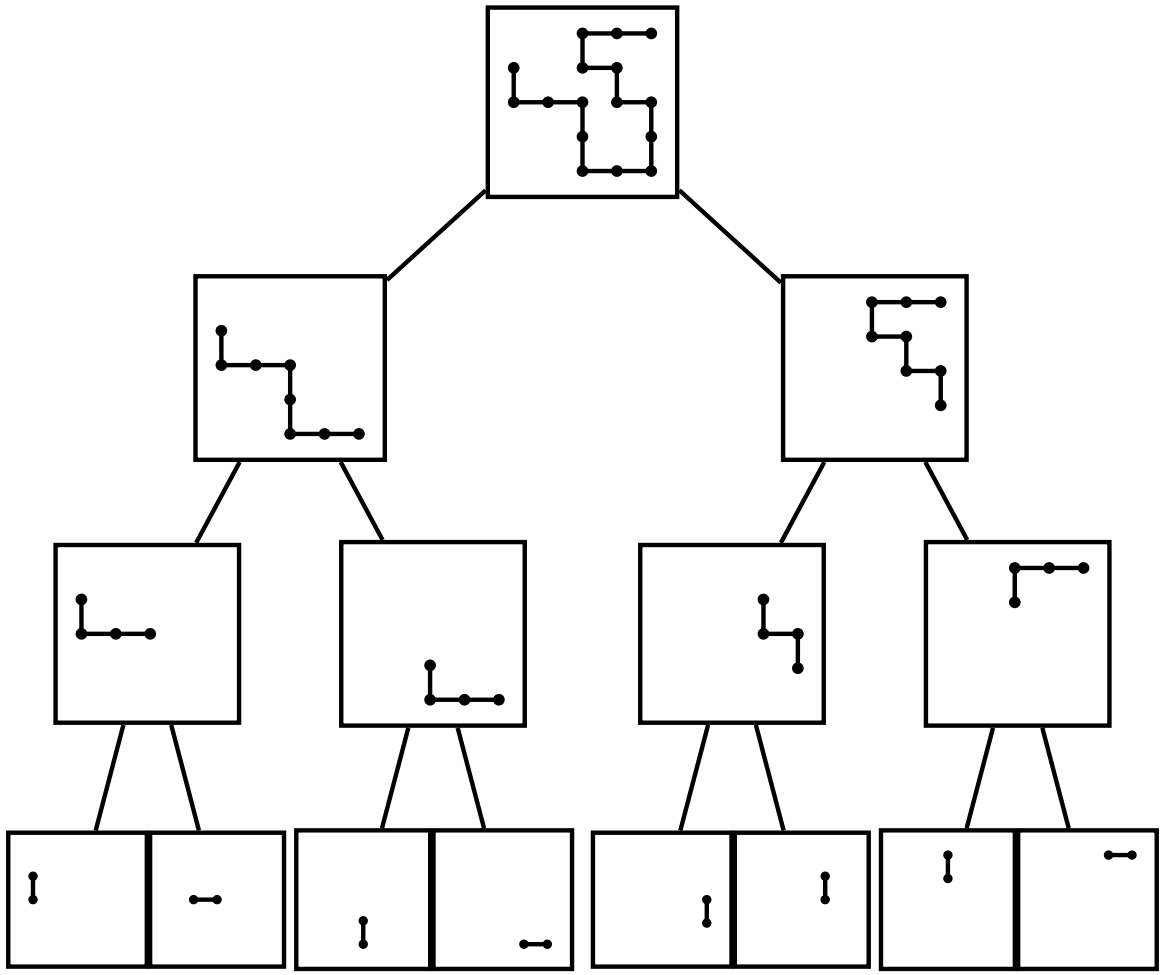
[4] Tom Kennedy: A faster implementation of the pivot algorithm for self-avoiding walks, in: J. Stat. Phys. 106 (2002), pp. 407–429.

[5] Nathan Clisby: Accurate Estimate of the Critical Exponent ν for Self-Avoiding Walks via a Fast Implementation of the Pivot Algorithm, in: Phys. Rev. Lett. 104 (2010), p. 055702.

- Time $O(N)$ to write down an N-step walk, so this must be best possible for pivot move?
- In fact, don't need to write down! CPU time $O(N^{0.74})$[4].
- Just verify that walk is self-avoiding after a pivot move, update information about global observables.
- Bookkeeping can be handled efficiently in binary tree structure $O(\log N)$[5].

[4] Tom Kennedy: A faster implementation of the pivot algorithm for self-avoiding walks, in: J. Stat. Phys. 106 (2002), pp. 407–429.

[5] Nathan Clisby: Accurate Estimate of the Critical Exponent ν for Self-Avoiding Walks via a Fast Implementation of the Pivot Algorithm, in: Phys. Rev. Lett. 104 (2010), p. 055702.



SAW-tree representation of a walk.

SAW-tree implementation

- Represent the SAW as binary trees, with "bounding box" information for sub-walks.

SAW-tree implementation

- Represent the SAW as binary trees, with "bounding box" information for sub-walks.
- Store symmetry operations in nodes of the tree.

SAW-tree implementation

- Represent the SAW as binary trees, with "bounding box" information for sub-walks.
- Store symmetry operations in nodes of the tree.
- Most operations take $O(\log N)$.

SAW-tree implementation

- Represent the SAW as binary trees, with "bounding box" information for sub-walks.
- Store symmetry operations in nodes of the tree.
- Most operations take $O(\log N)$.
- In CPU units, $\tau_{\mathrm{int}}(R_{\mathrm{E}}^2) = O(N^p \log N)$, with $p \approx 0.11$ for $d = 3$.

SAW-tree implementation

- Represent the SAW as binary trees, with "bounding box" information for sub-walks.
- Store symmetry operations in nodes of the tree.
- Most operations take $O(\log N)$.
- In CPU units, $\tau_{\mathrm{int}}(R_{\mathrm{E}}^2) = O(N^p \log N)$, with $p \approx 0.11$ for $d = 3$.
- *Global effect* for *local cost*.

Results

- Simulations of SAWs with up to 1 billion monomers.

[6] Nathan Clisby/Burkhard Dünweg: High-precision estimate of the hydrodynamic radius for self-avoiding walks, in: Phys. Rev. E 94 (2016), p. 052102.

[7] Nathan Clisby: Scale-free Monte Carlo method for calculating the critical exponent of self-avoiding walks, in: J. Phys. A: Math. Theor. 50 (2017), p. 264003.

[8] Nathan Clisby: Monte Carlo study of four-dimensional self-avoiding walks of up to one billion steps, in: arXiv (2017), p. 1703.10557.

Results

- Simulations of SAWs with up to 1 billion monomers.
- $\nu = 0.587\,597\,0(4)$[6].

[6] Nathan Clisby/Burkhard Dünweg: High-precision estimate of the hydrodynamic radius for self-avoiding walks, in: Phys. Rev. E 94 (2016), p. 052102.

[7] Nathan Clisby: Scale-free Monte Carlo method for calculating the critical exponent of self-avoiding walks, in: J. Phys. A: Math. Theor. 50 (2017), p. 264003.

[8] Nathan Clisby: Monte Carlo study of four-dimensional self-avoiding walks of up to one billion steps, in: arXiv (2017), p. 1703.10557.

Results

- Simulations of SAWs with up to 1 billion monomers.
- $\nu = 0.587\,597\,0(4)$[6].
- $\gamma = 1.156\,953(1)$[7].

[6] Nathan Clisby/Burkhard Dünweg: High-precision estimate of the hydrodynamic radius for self-avoiding walks, in: Phys. Rev. E 94 (2016), p. 052102.

[7] Nathan Clisby: Scale-free Monte Carlo method for calculating the critical exponent of self-avoiding walks, in: J. Phys. A: Math. Theor. 50 (2017), p. 264003.

[8] Nathan Clisby: Monte Carlo study of four-dimensional self-avoiding walks of up to one billion steps, in: arXiv (2017), p. 1703.10557.

Results

- Simulations of SAWs with up to 1 billion monomers.
- $\nu = 0.587\,597\,0(4)$[6].
- $\gamma = 1.156\,953(1)$[7].
- For 4-dimensional SAWs, obtained power of log correction for $\langle R_{\mathrm{E}}^2 \rangle$ of $0.2516(14)$[8], versus prediction of $1/4$.

[6] Nathan Clisby/Burkhard Dünweg: High-precision estimate of the hydrodynamic radius for self-avoiding walks, in: Phys. Rev. E 94 (2016), p. 052102.

[7] Nathan Clisby: Scale-free Monte Carlo method for calculating the critical exponent of self-avoiding walks, in: J. Phys. A: Math. Theor. 50 (2017), p. 264003.

[8] Nathan Clisby: Monte Carlo study of four-dimensional self-avoiding walks of up to one billion steps, in: arXiv (2017), p. 1703.10557.

Results

- Simulations of SAWs with up to 1 billion monomers.
- $\nu = 0.587\,597\,0(4)$[6].
- $\gamma = 1.156\,953(1)$[7].
- For 4-dimensional SAWs, obtained power of log correction for $\langle R_{\mathrm{E}}^2 \rangle$ of $0.2516(14)$[8], versus prediction of $1/4$.
- Can we find a fast global move for dense polymers?

[6] Nathan Clisby/Burkhard Dünweg: High-precision estimate of the hydrodynamic radius for self-avoiding walks, in: Phys. Rev. E 94 (2016), p. 052102.

[7] Nathan Clisby: Scale-free Monte Carlo method for calculating the critical exponent of self-avoiding walks, in: J. Phys. A: Math. Theor. 50 (2017), p. 264003.

[8] Nathan Clisby: Monte Carlo study of four-dimensional self-avoiding walks of up to one billion steps, in: arXiv (2017), p. 1703.10557.

Hamiltonian paths

- Hamiltonian paths are self-avoiding walks which visit every vertex in a graph.

Hamiltonian paths

- Hamiltonian paths are self-avoiding walks which visit every vertex in a graph.
- Generally take the graph to be a region of the square or simple cubic lattices.

Hamiltonian paths

- Hamiltonian paths are self-avoiding walks which visit every vertex in a graph.
- Generally take the graph to be a region of the square or simple cubic lattices.
- Model of the crystal phase of polymers.

Hamiltonian paths

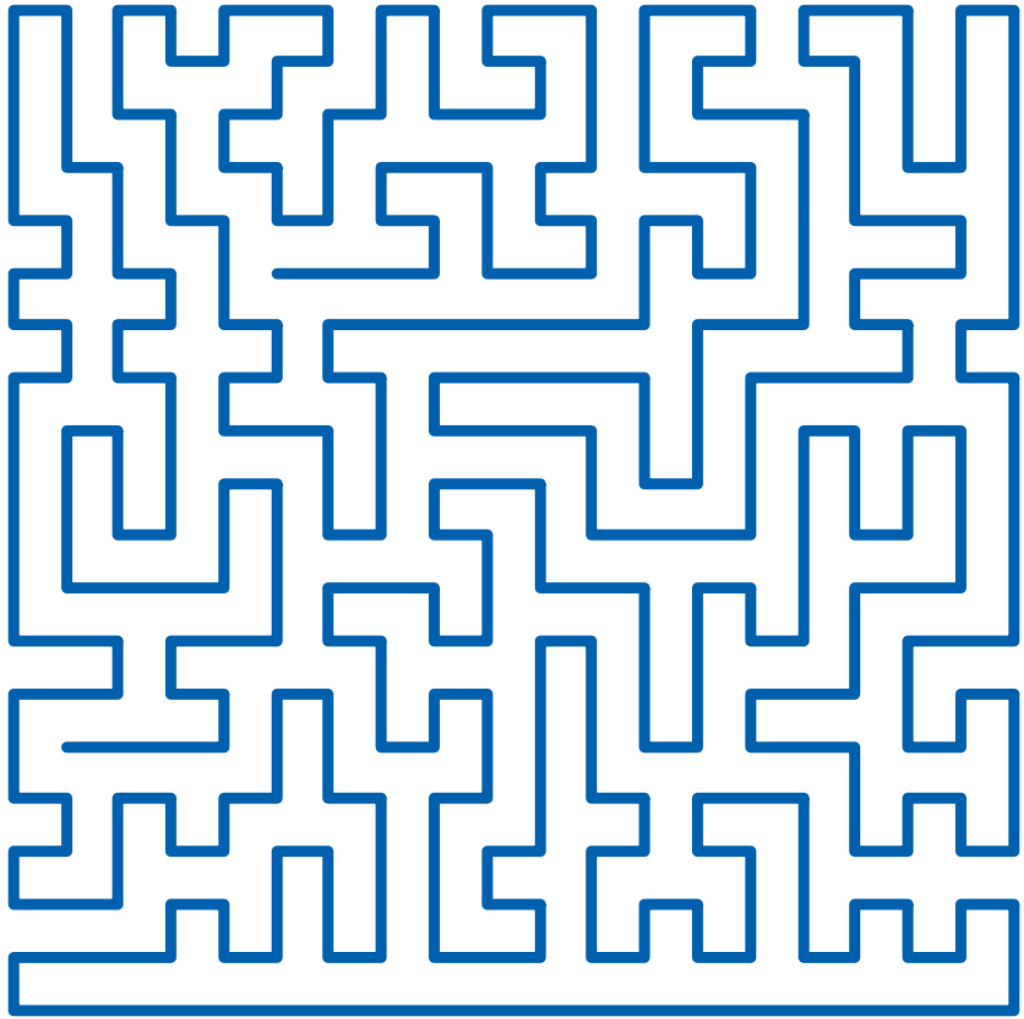
- Hamiltonian paths are self-avoiding walks which visit every vertex in a graph.
- Generally take the graph to be a region of the square or simple cubic lattices.
- Model of the crystal phase of polymers.
- Can extend to polymer melts which involve many paths.

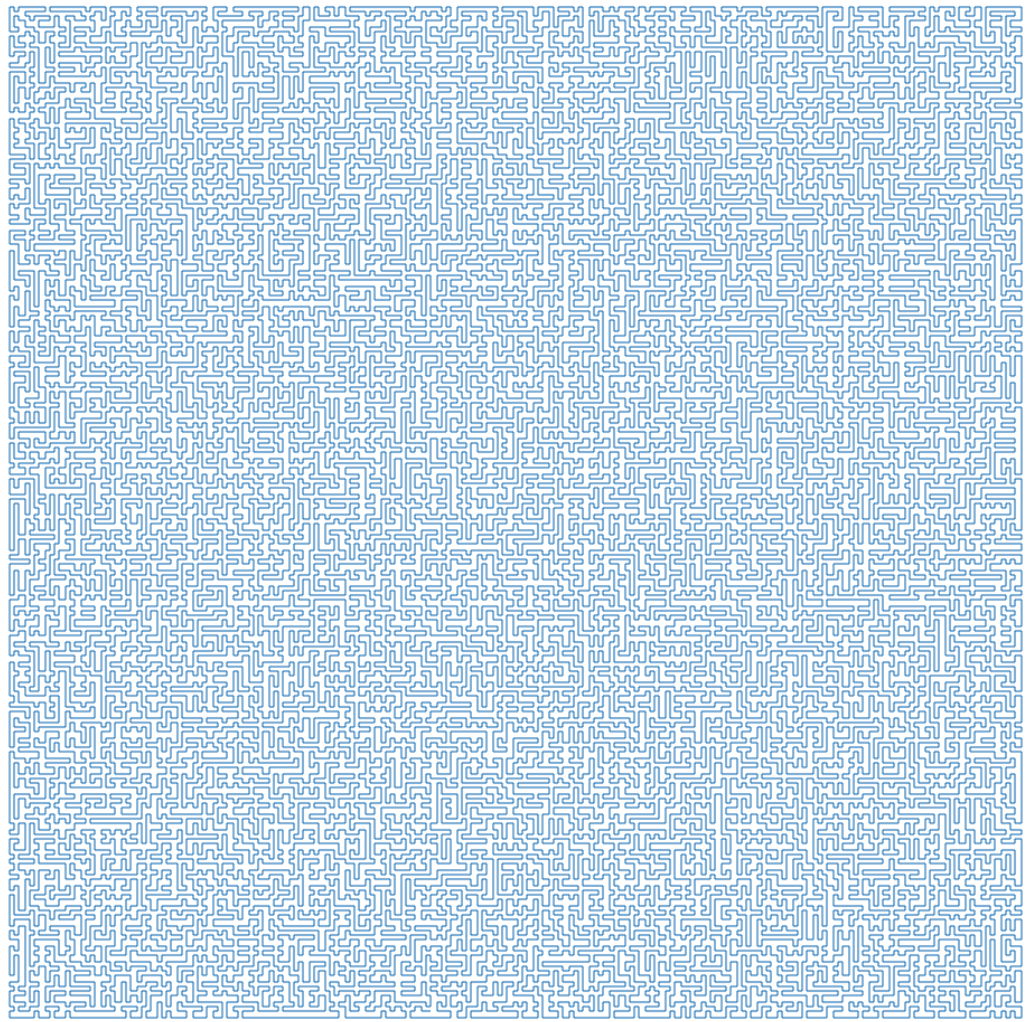
Hamiltonian paths

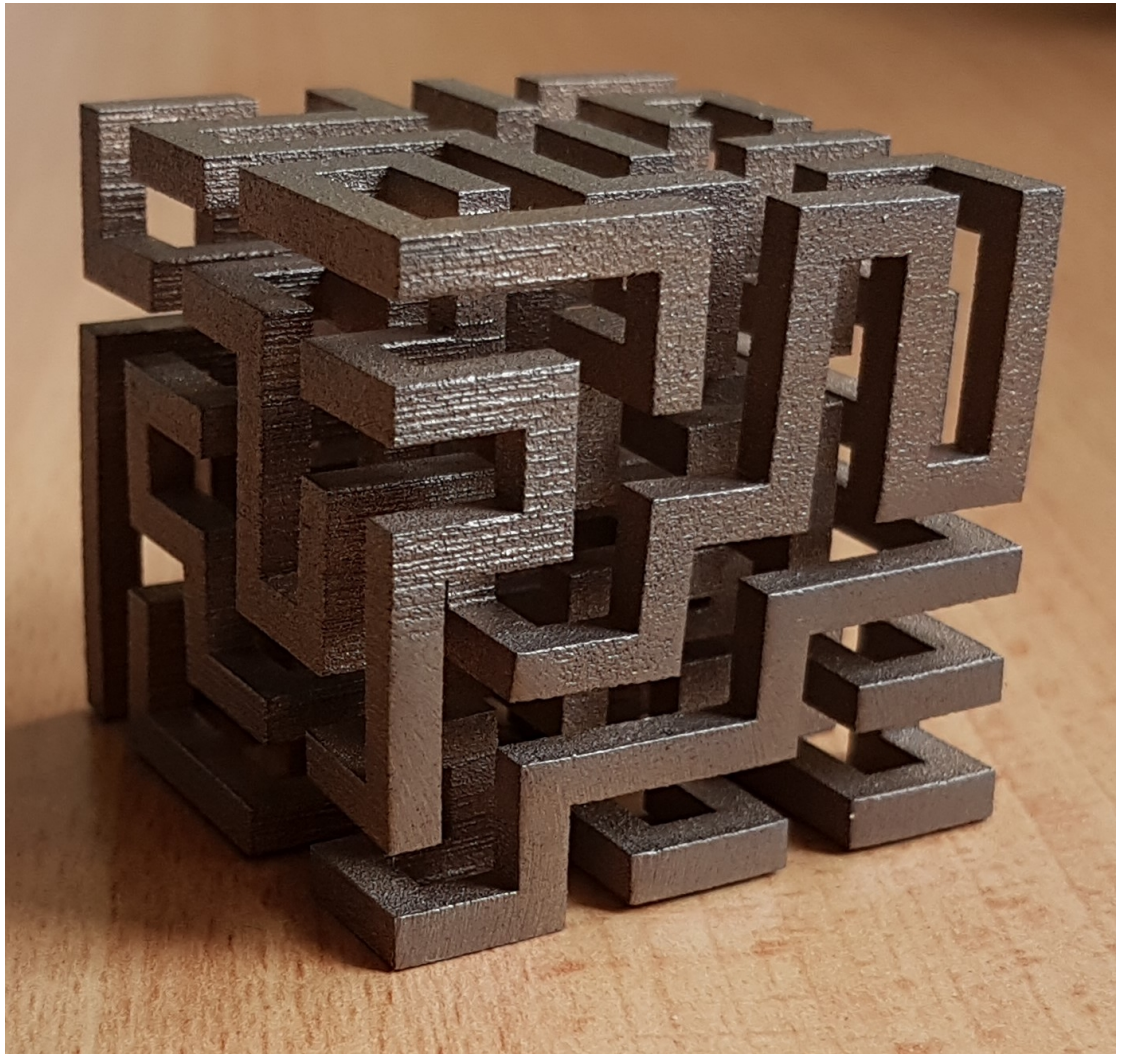
- Hamiltonian paths are self-avoiding walks which visit every vertex in a graph.
- Generally take the graph to be a region of the square or simple cubic lattices.
- Model of the crystal phase of polymers.
- Can extend to polymer melts which involve many paths.
- Universality implies that these lattice models capture essential physics of real dense polymer systems.

Hamiltonian paths

- Hamiltonian paths are self-avoiding walks which visit every vertex in a graph.
- Generally take the graph to be a region of the square or simple cubic lattices.
- Model of the crystal phase of polymers.
- Can extend to polymer melts which involve many paths.
- Universality implies that these lattice models capture essential physics of real dense polymer systems.
- What do they look like?







Backbite and bond rebridging moves

- One of the most effective methods for equilibrating dense polymer systems is via connectivity changing moves. (Another powerful technique is coarse graining.)

Backbite and bond rebridging moves

- One of the most effective methods for equilibrating dense polymer systems is via connectivity changing moves. (Another powerful technique is coarse graining.)
- Three important examples are:

Backbite and bond rebridging moves

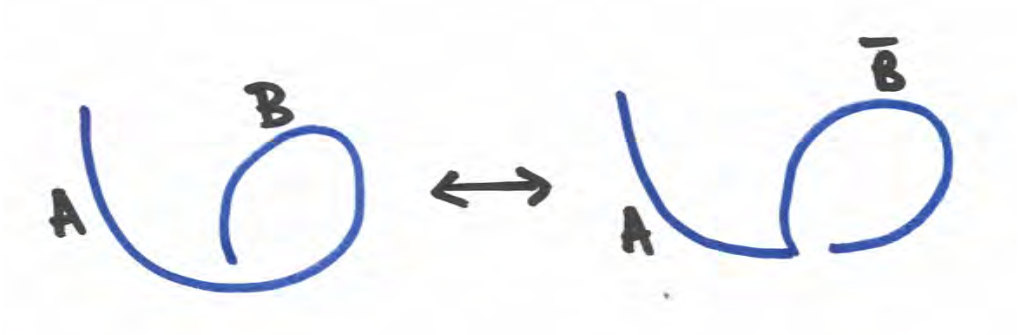
- One of the most effective methods for equilibrating dense polymer systems is via connectivity changing moves. (Another powerful technique is coarse graining.)
- Three important examples are:
  - Backbite: choose an end, extend by a single step which will form a loop, traverse loop in opposite direction and delete last bond.

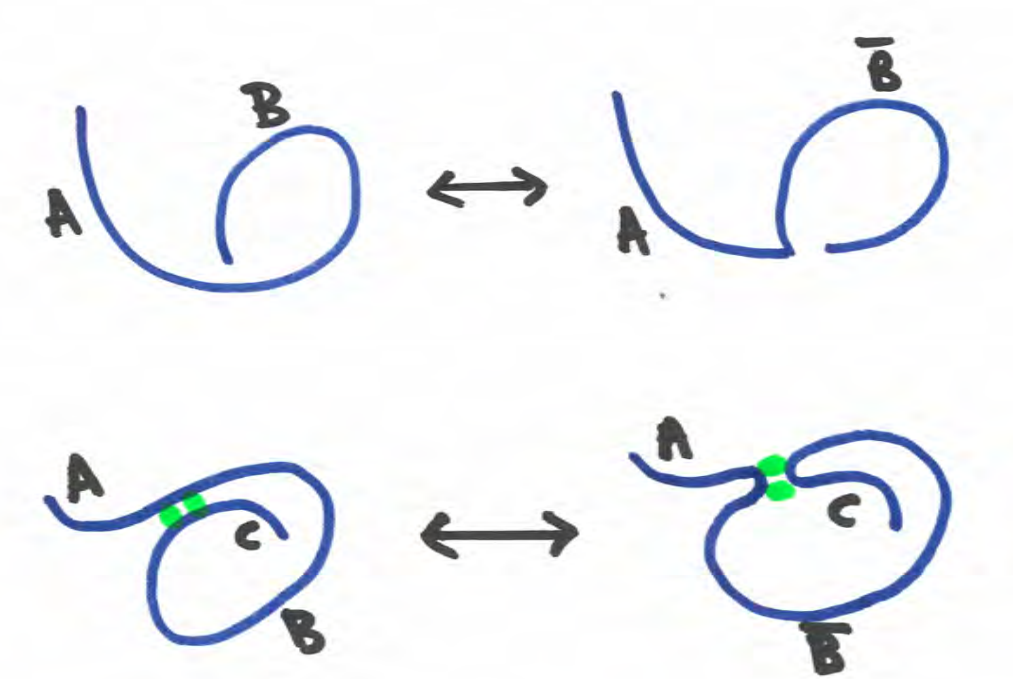
Backbite and bond rebridging moves

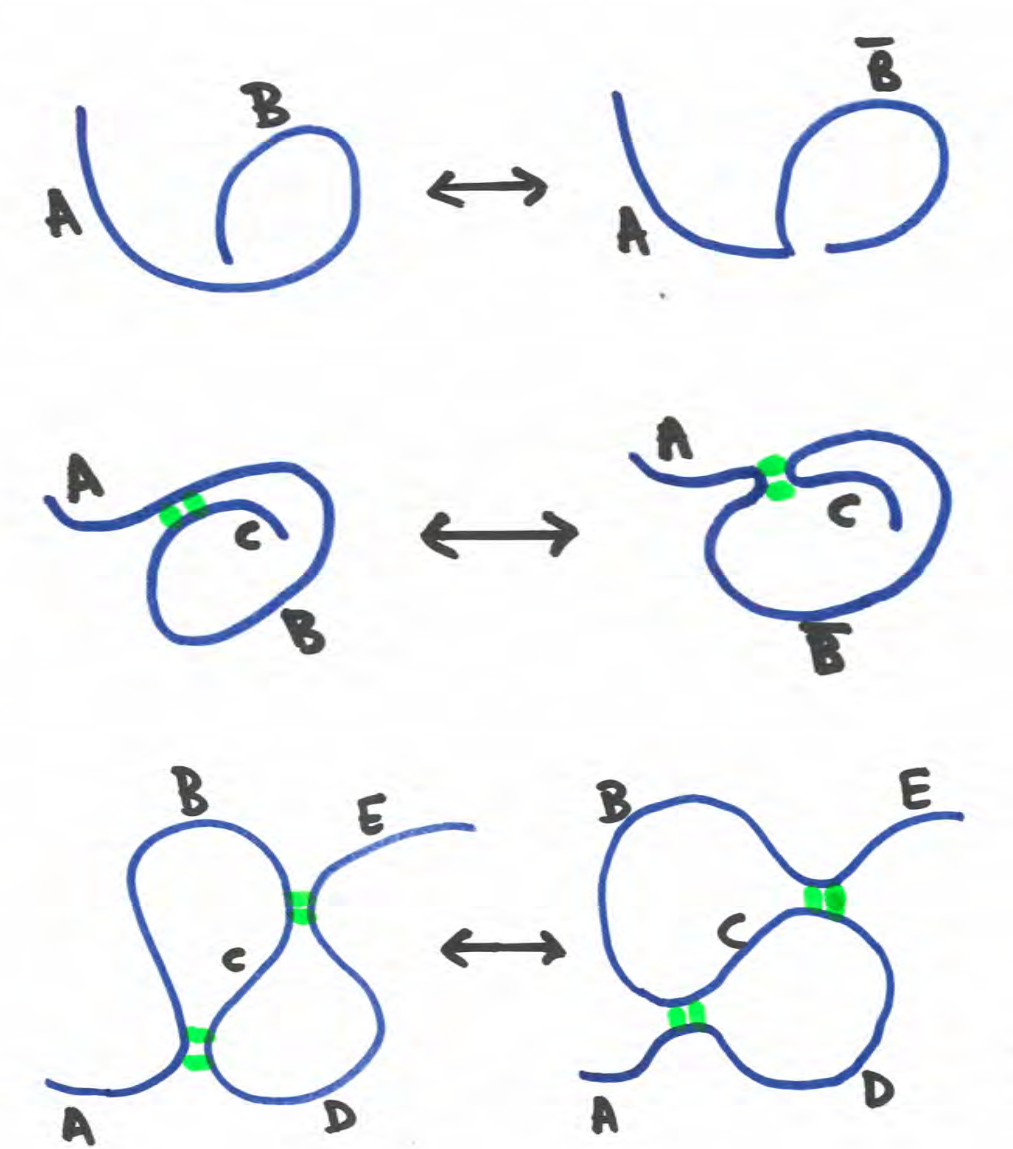
- One of the most effective methods for equilibrating dense polymer systems is via connectivity changing moves. (Another powerful technique is coarse graining.)
- Three important examples are:
  - Backbite: choose an end, extend by a single step which will form a loop, traverse loop in opposite direction and delete last bond.
  - Rebridging.

Backbite and bond rebridging moves

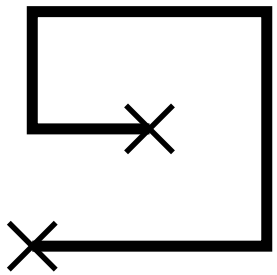
- One of the most effective methods for equilibrating dense polymer systems is via connectivity changing moves. (Another powerful technique is coarse graining.)
- Three important examples are:
  - Backbite: choose an end, extend by a single step which will form a loop, traverse loop in opposite direction and delete last bond.
  - Rebridging.
  - Double rebridging.


B
A
B
A

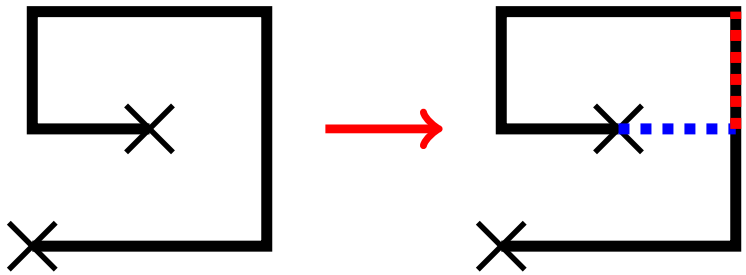

A
B
B
A
A
C
B
A
C
B


A
B
B
A
A
C
B
A
C
B
B
E
C
A
D
B
E
C
A
D

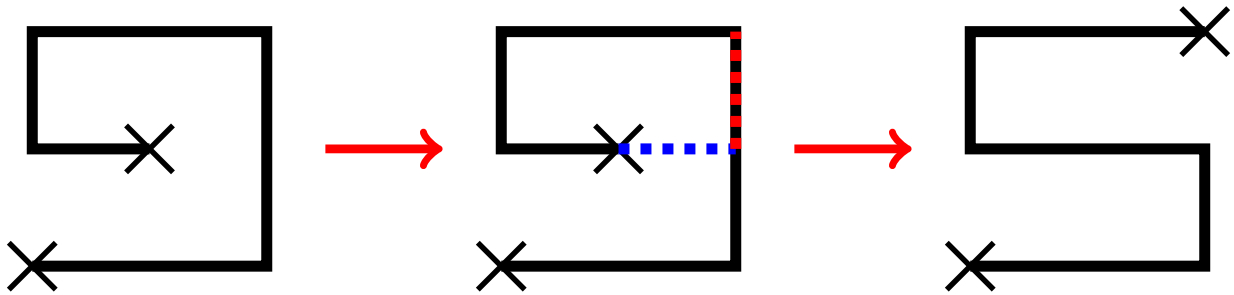
Backbite moves for sampling Hamiltonian paths.



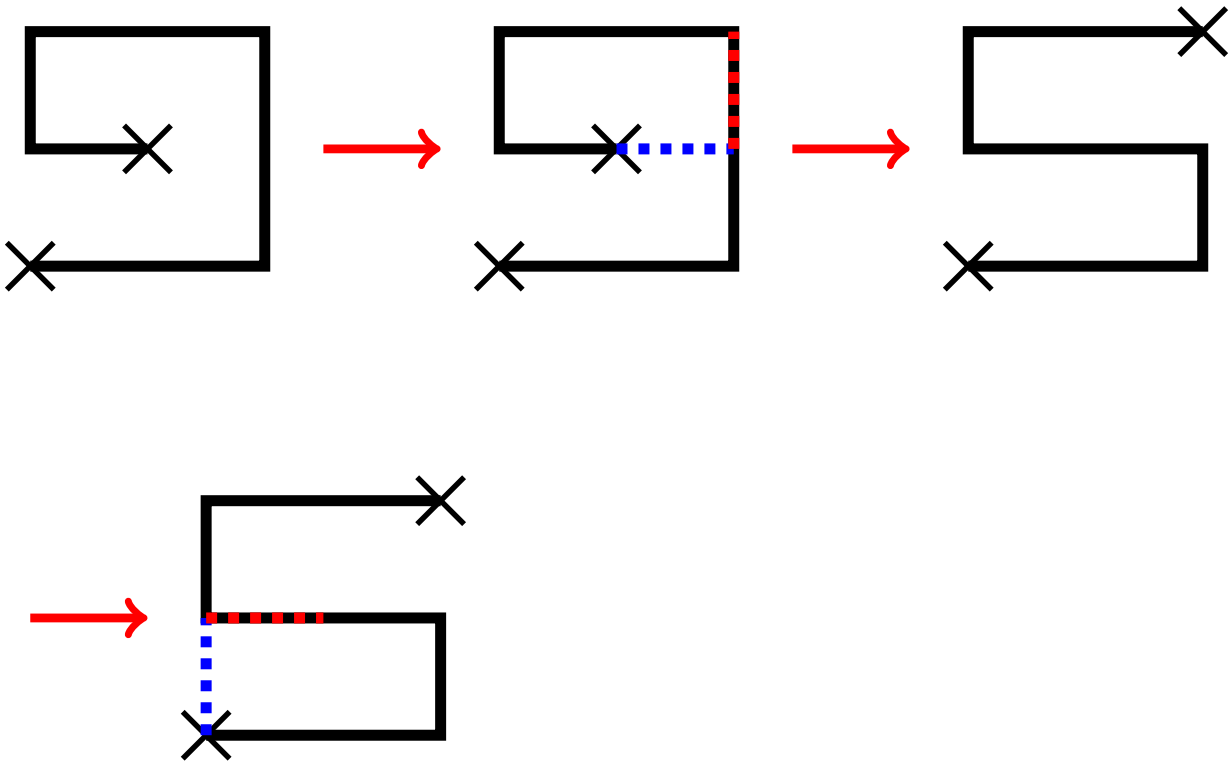
Backbite moves for sampling Hamiltonian paths.



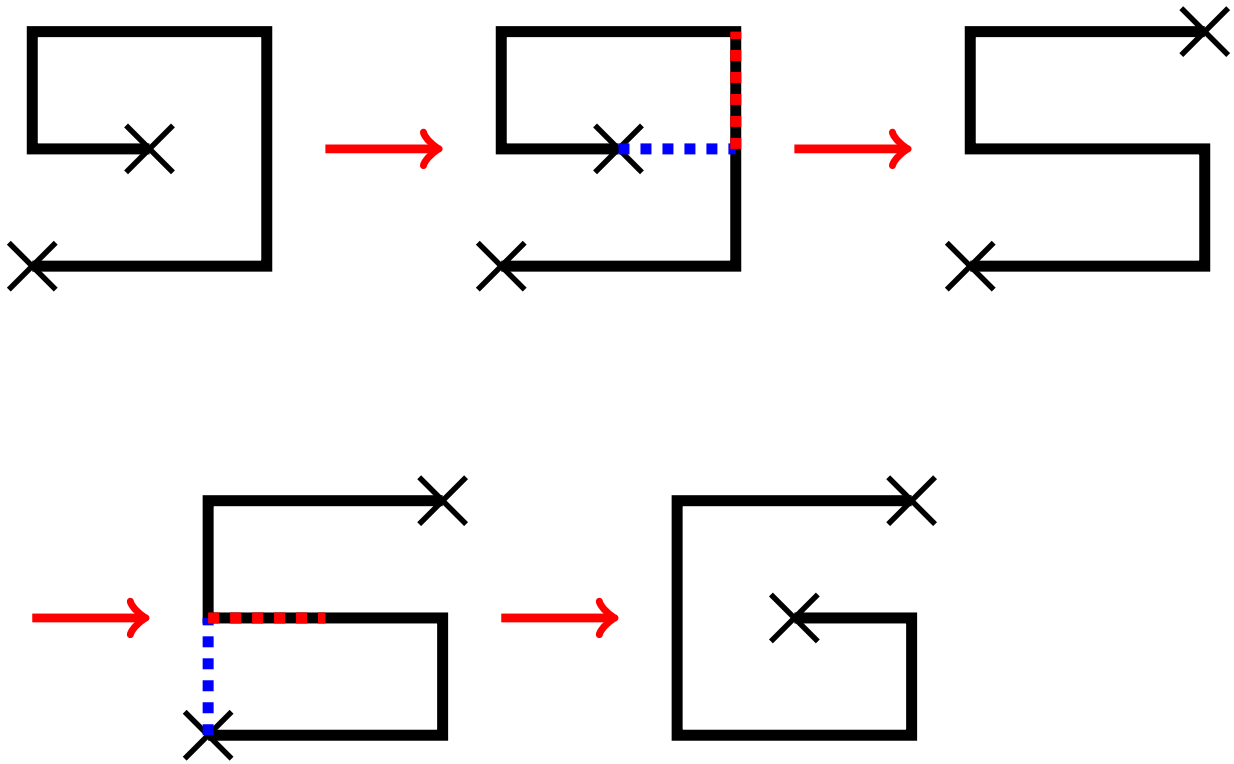
Backbite moves for sampling Hamiltonian paths.



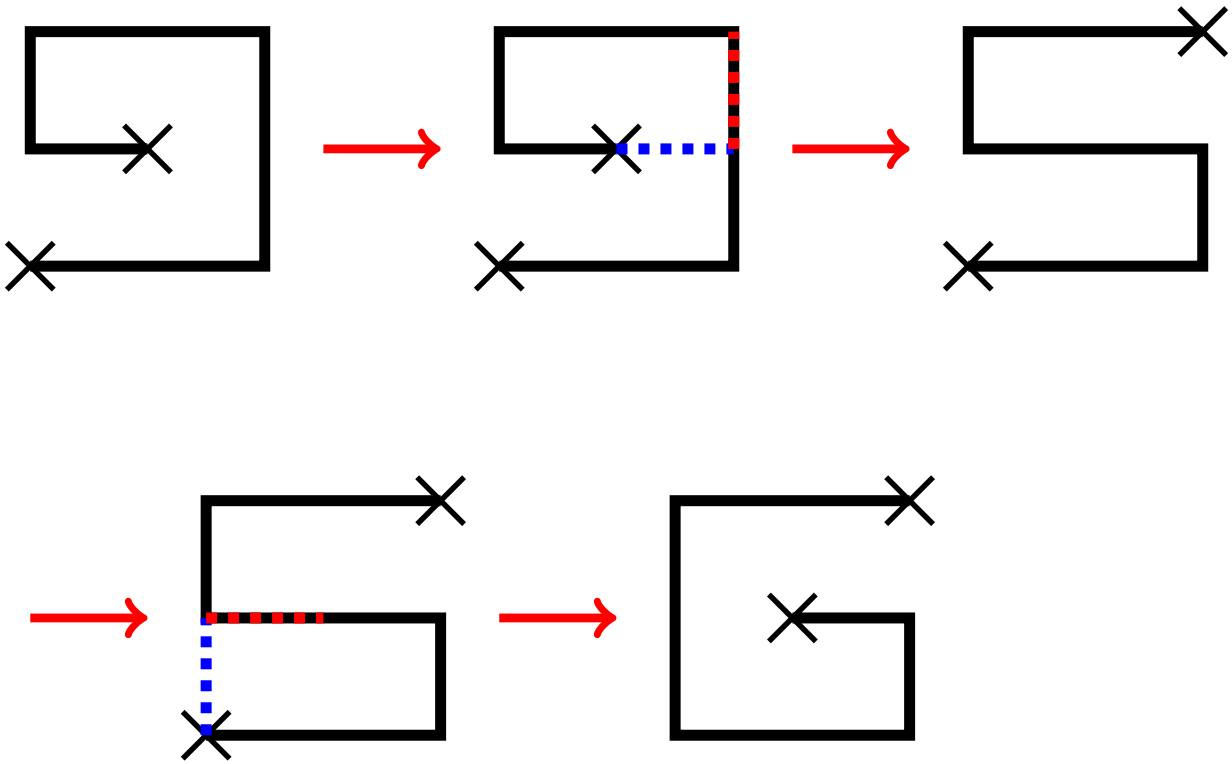
Backbite moves for sampling Hamiltonian paths.



Backbite moves for sampling Hamiltonian paths.



Backbite moves for sampling Hamiltonian paths.



Each time we make a backbite move we create a loop, delete the edge which completes the loop, and *reverse* the orientation of the remaining edges of the loop.

Backbite algorithm properties

- Is the algorithm correct?

Backbite algorithm properties

- Is the algorithm correct?
  - Yes, it satisfies detailed balance, and ergodicity has been verified for small lattices.

Backbite algorithm properties

- Is the algorithm correct?
  - Yes, it satisfies detailed balance, and ergodicity has been verified for small lattices.
  - (But no proof! If you have any ideas how to prove this let me know. See: http://clisby.net/projects/hamiltonian_path/)

Backbite algorithm properties

- Is the algorithm correct?
  - Yes, it satisfies detailed balance, and ergodicity has been verified for small lattices.
  - (But no proof! If you have any ideas how to prove this let me know. See: http://clisby.net/projects/hamiltonian_path/)
- Are the moves global?

Backbite algorithm properties

- Is the algorithm correct?
  - Yes, it satisfies detailed balance, and ergodicity has been verified for small lattices.
  - (But no proof! If you have any ideas how to prove this let me know. See: http://clisby.net/projects/hamiltonian_path/)
- Are the moves global?
  - Kind of! Global change to connectivity, but endpoint shifts by only two steps.

Backbite algorithm properties

- Is the algorithm correct?
  - Yes, it satisfies detailed balance, and ergodicity has been verified for small lattices.
  - (But no proof! If you have any ideas how to prove this let me know. See: http://clisby.net/projects/hamiltonian_path/)
- Are the moves global?
  - Kind of! Global change to connectivity, but endpoint shifts by only two steps.
- Are the moves fast?

Backbite algorithm properties

- Is the algorithm correct?
  - Yes, it satisfies detailed balance, and ergodicity has been verified for small lattices.
  - (But no proof! If you have any ideas how to prove this let me know. See: http://clisby.net/projects/hamiltonian_path/)
- Are the moves global?
  - Kind of! Global change to connectivity, but endpoint shifts by only two steps.
- Are the moves fast?
  - For a loop of length N, naive implementation of backbite move takes CPU time $O(N)$ as it requires reversal of N steps.

Backbite move

- For the simple cubic lattice, loops are of mean size $O(N)$.

Backbite move

- For the simple cubic lattice, loops are of mean size $O(N)$.
- $\Rightarrow$ mean CPU time per backbite move $O(N)$.

Backbite move

- For the simple cubic lattice, loops are of mean size $O(N)$.
- $\Rightarrow$ mean CPU time per backbite move $O(N)$.
- But, like with SAW, we don't need to explicitly "write down" each step of the walk, just need to be able to store information about structure, and find neighbours of walk ends.

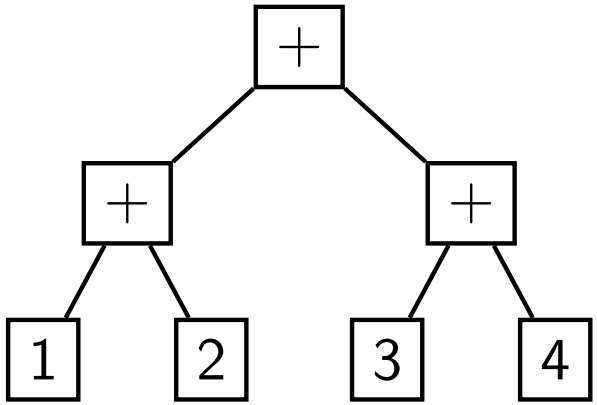
Backbite move

- For the simple cubic lattice, loops are of mean size $O(N)$.
- $\Rightarrow$ mean CPU time per backbite move $O(N)$.
- But, like with SAW, we don't need to explicitly "write down" each step of the walk, just need to be able to store information about structure, and find neighbours of walk ends.
- Use binary tree data structure with *time reversal* as our symmetry operation, with bookkeeping for determining neighbours.

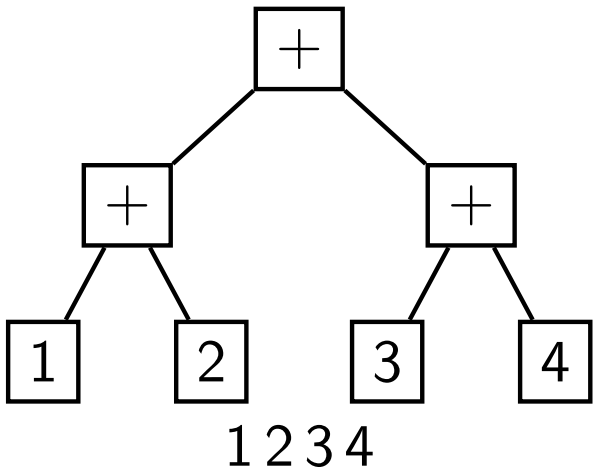
Backbite move

- For the simple cubic lattice, loops are of mean size $O(N)$.
- $\Rightarrow$ mean CPU time per backbite move $O(N)$.
- But, like with SAW, we don't need to explicitly "write down" each step of the walk, just need to be able to store information about structure, and find neighbours of walk ends.
- Use binary tree data structure with *time reversal* as our symmetry operation, with bookkeeping for determining neighbours.
- Can generalise to other moves for dense polymers, e.g. bond rebridging.

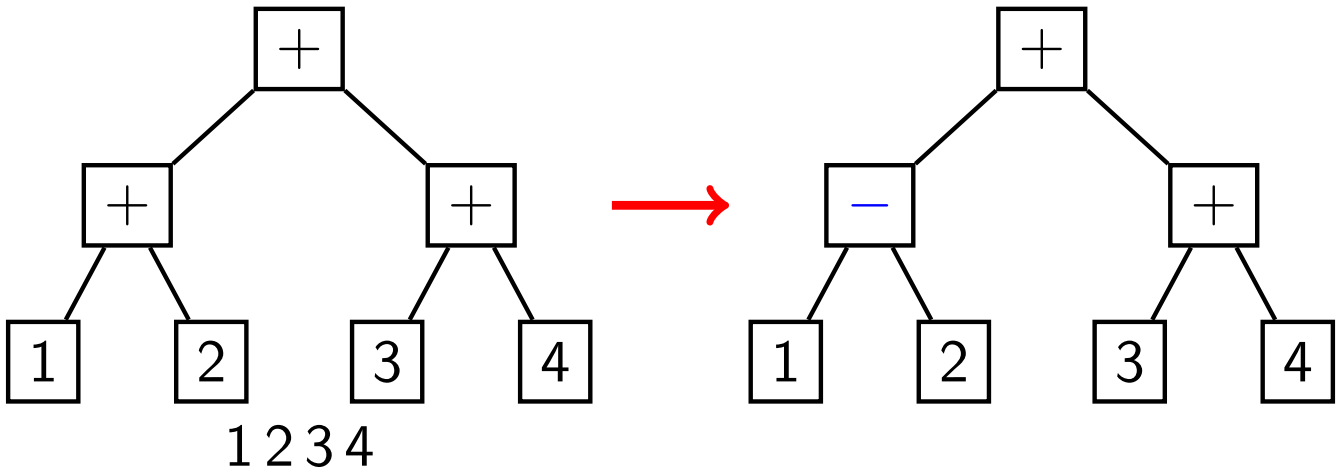
Time reversal symmetry elements in the binary tree nodes.



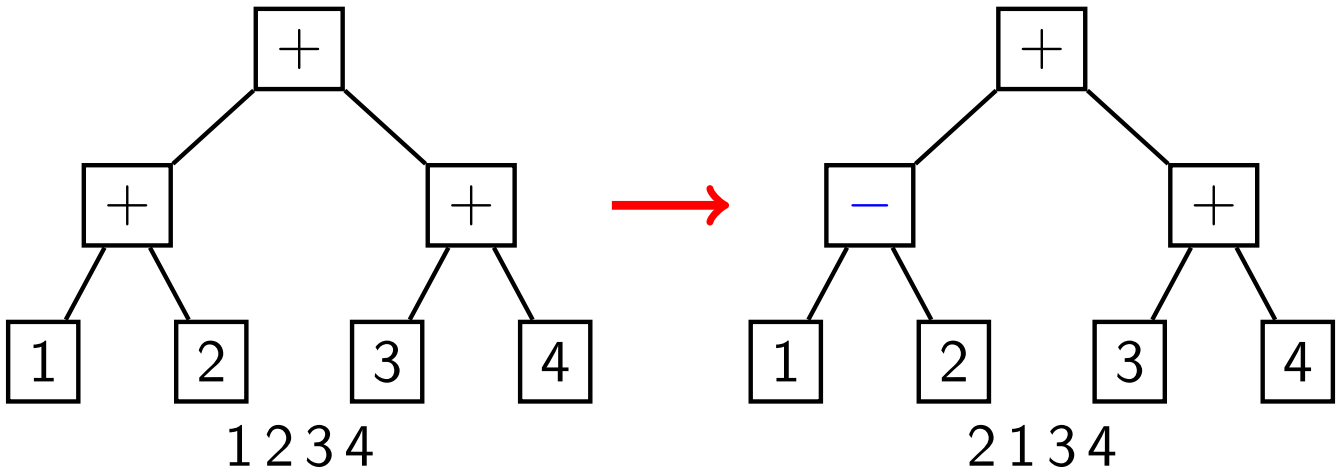
Time reversal symmetry elements in the binary tree nodes.



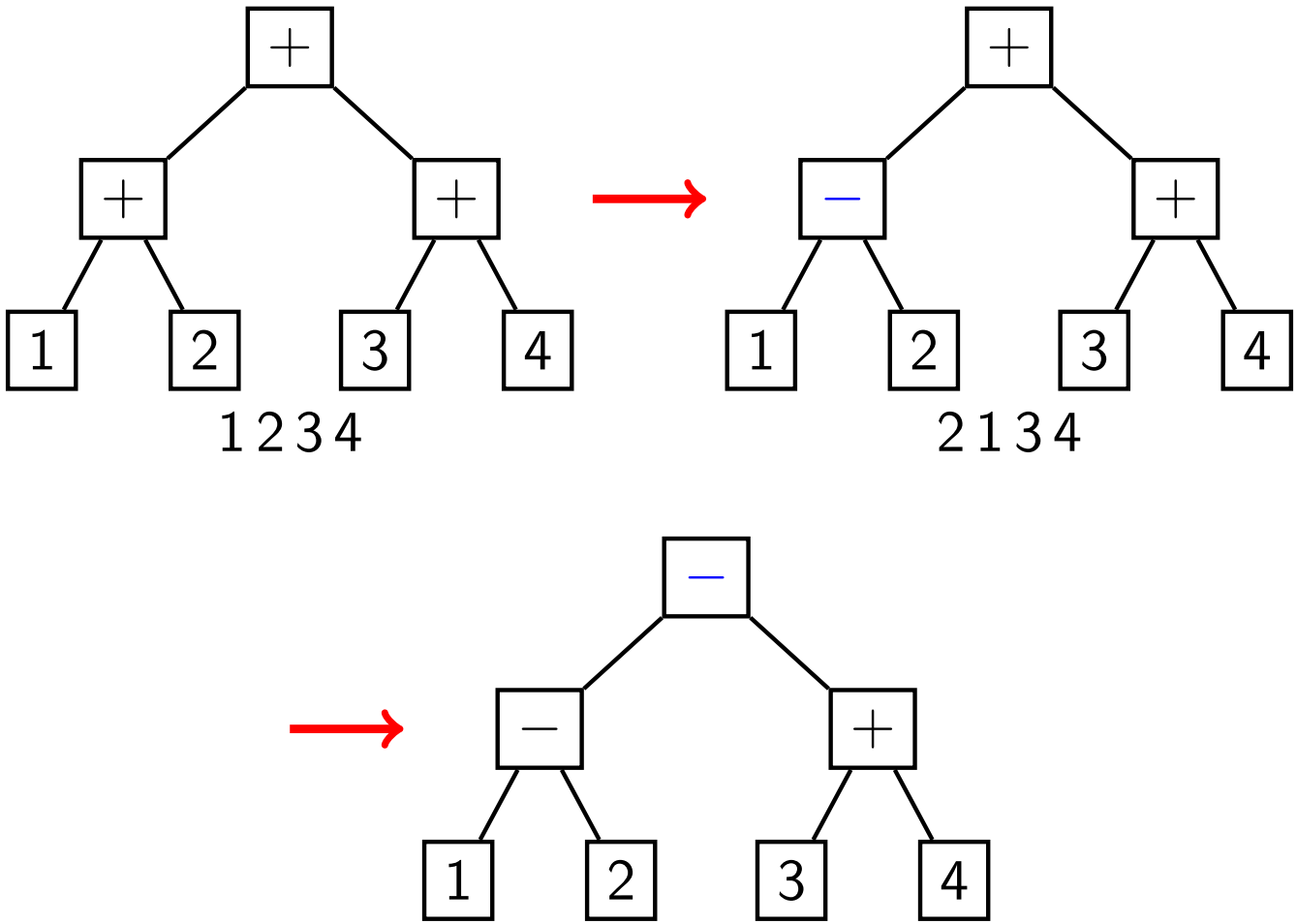
Time reversal symmetry elements in the binary tree nodes.



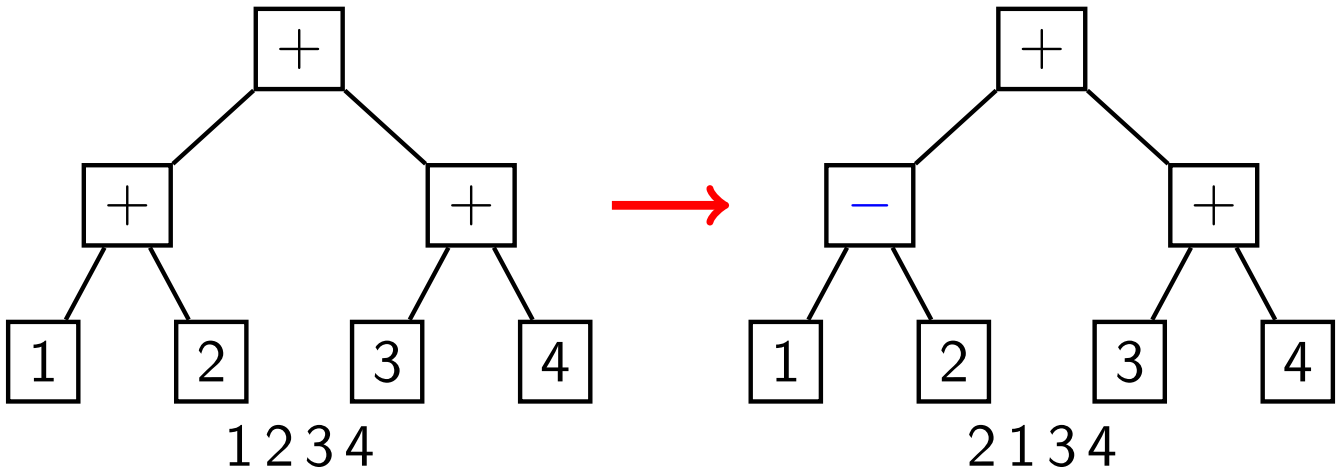
Time reversal symmetry elements in the binary tree nodes.

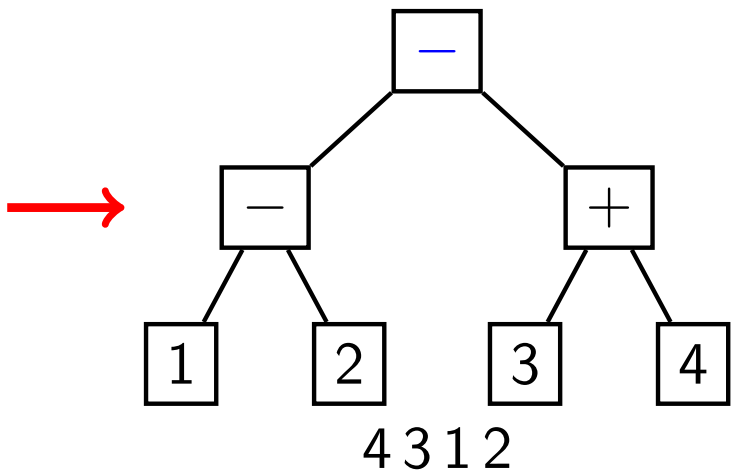


Time reversal symmetry elements in the binary tree nodes.

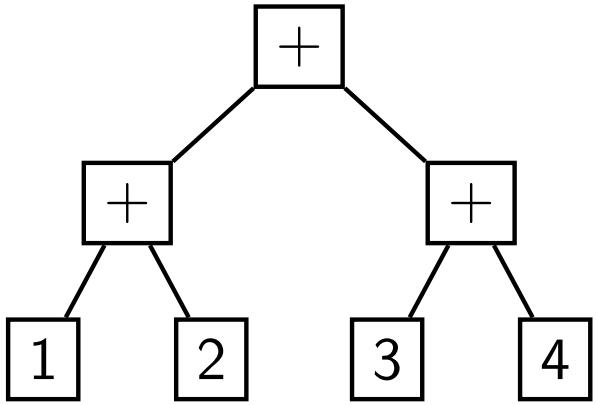


Time reversal symmetry elements in the binary tree nodes.

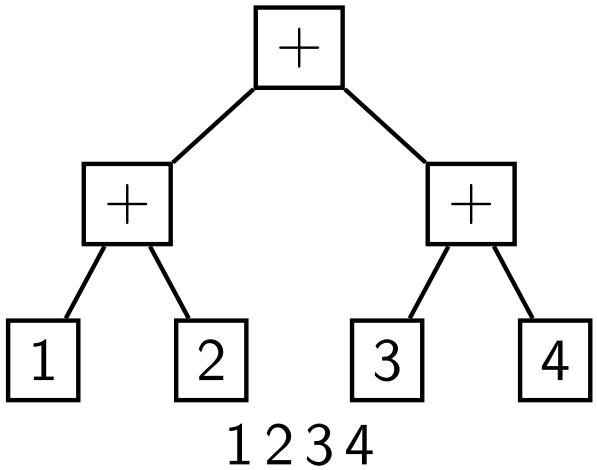




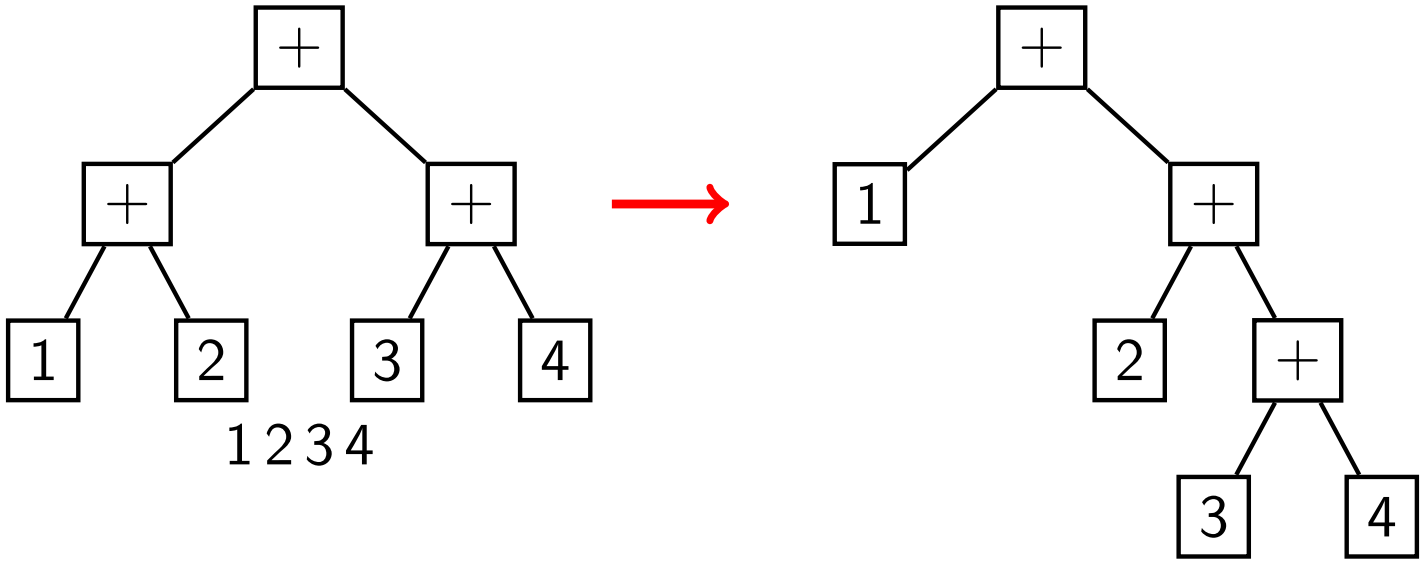
How do we reverse sequences of steps which don't align with the tree?



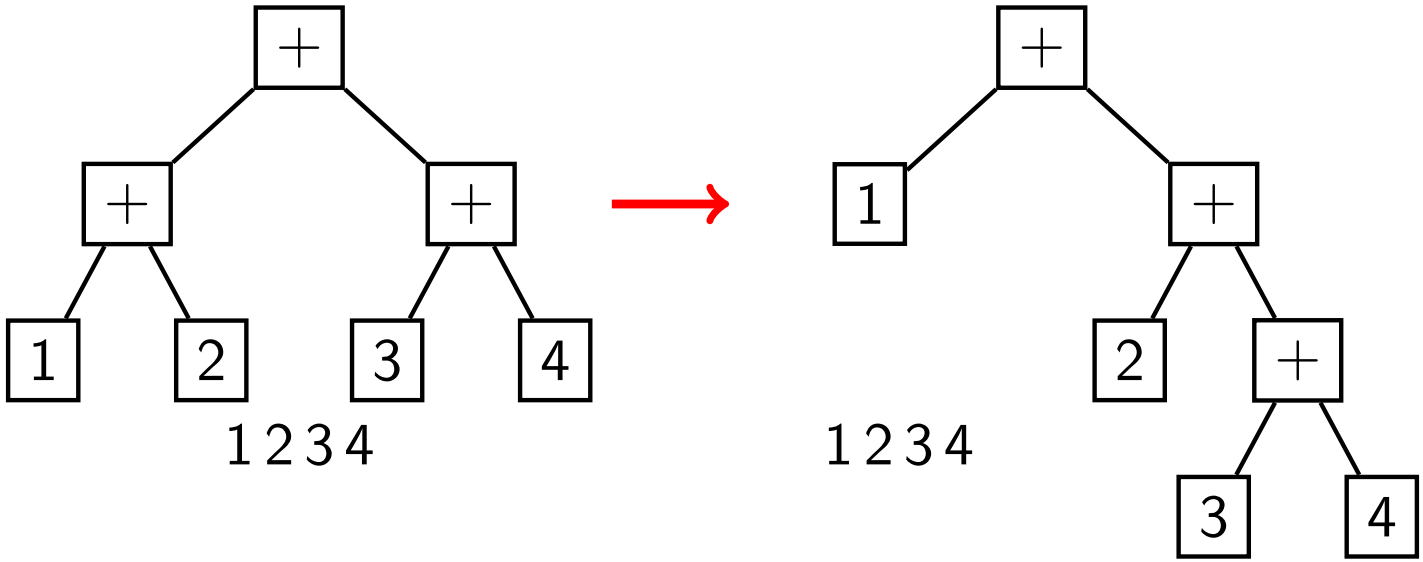
How do we reverse sequences of steps which don't align with the tree?



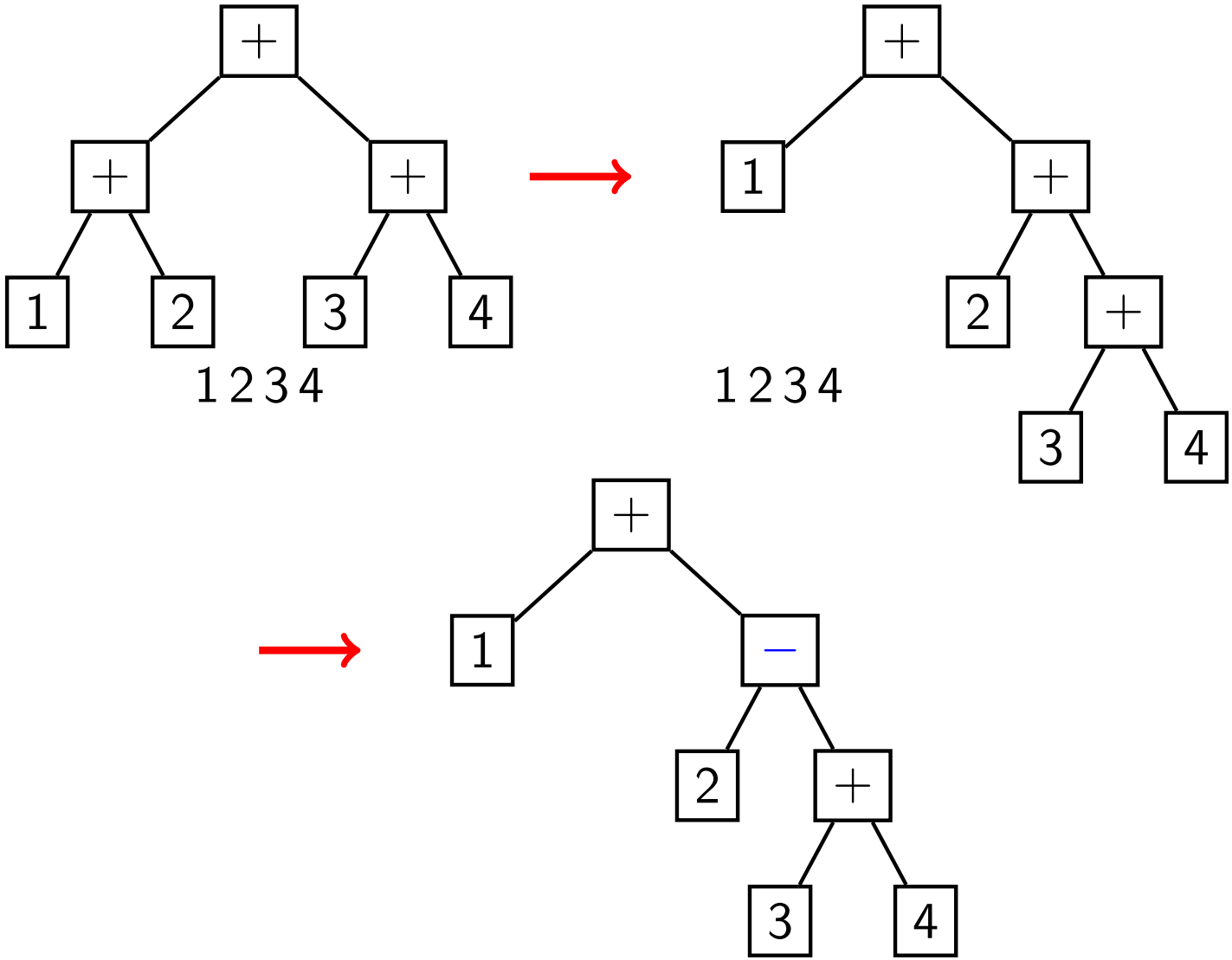
How do we reverse sequences of steps which don't align with the tree?



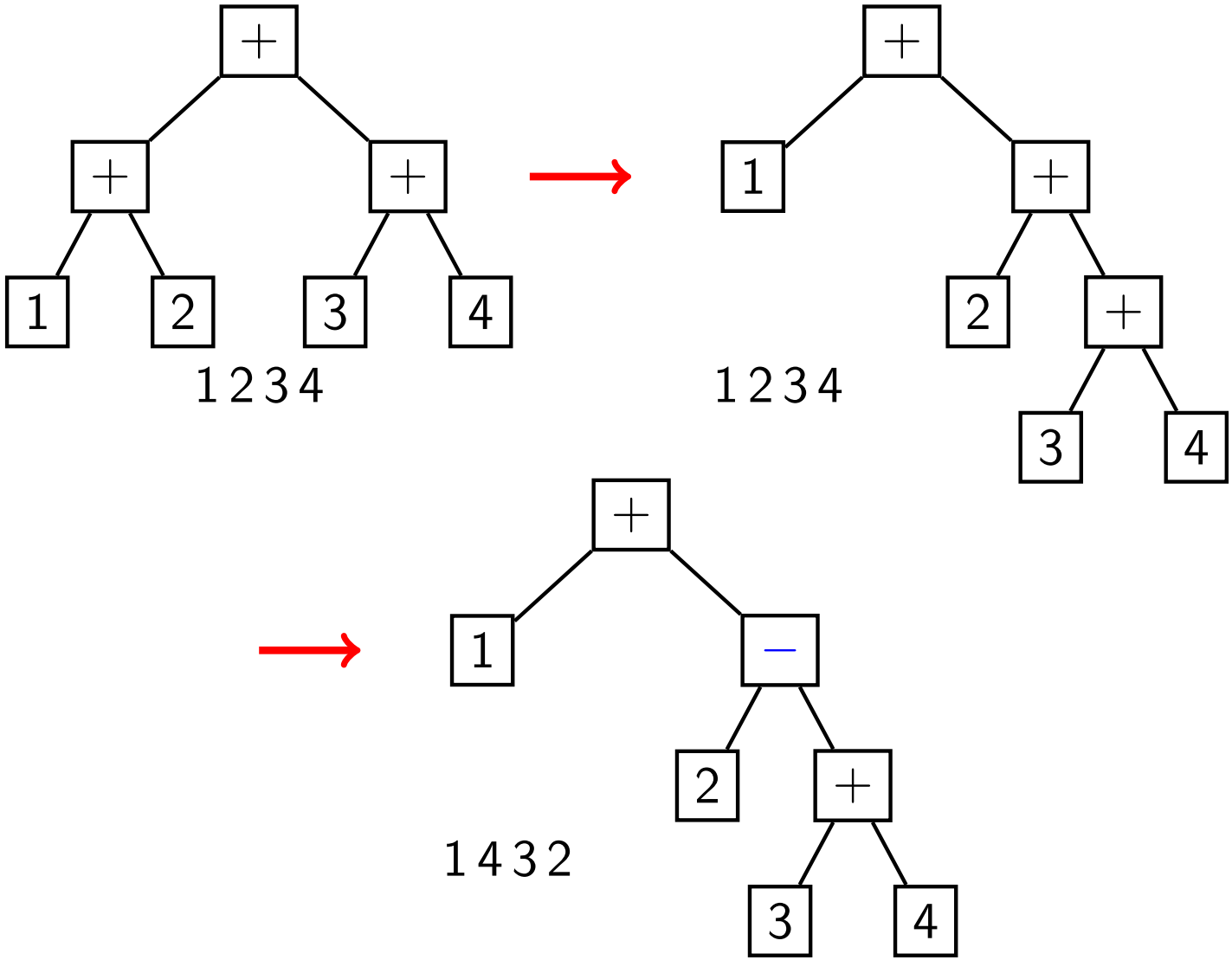
How do we reverse sequences of steps which don't align with the tree?



How do we reverse sequences of steps which don't align with the tree?



How do we reverse sequences of steps which don't align with the tree?



Backbite algorithm performance

- Binary tree implementation: to reverse $O(N)$ steps requires $O(\log N)$ tree rotations.

Backbite algorithm performance

- Binary tree implementation: to reverse $O(N)$ steps requires $O(\log N)$ tree rotations.
- For square lattice, mean size of loops is $O(N^{\approx 0.75})$.

Backbite algorithm performance

- Binary tree implementation: to reverse $O(N)$ steps requires $O(\log N)$ tree rotations.
- For square lattice, mean size of loops is $O(N^{\approx 0.75})$.
- $L = 8192$, $N \approx 67$ million: 360 thousand steps, versus 6.77 tree rotations.

Backbite algorithm performance

- Binary tree implementation: to reverse $O(N)$ steps requires $O(\log N)$ tree rotations.
- For square lattice, mean size of loops is $O(N^{\approx 0.75})$.
- $L = 8192$, $N \approx 67$ million: 360 thousand steps, versus 6.77 tree rotations.
- For simple cubic lattice, mean size of loops is $O(N)$.

Backbite algorithm performance

- Binary tree implementation: to reverse $O(N)$ steps requires $O(\log N)$ tree rotations.
- For square lattice, mean size of loops is $O(N^{\approx 0.75})$.
- $L = 8192$, $N \approx 67$ million: 360 thousand steps, versus 6.77 tree rotations.
- For simple cubic lattice, mean size of loops is $O(N)$.
- $L = 512$, $N \approx 134$ million:

Backbite algorithm performance

- Binary tree implementation: to reverse $O(N)$ steps requires $O(\log N)$ tree rotations.
- For square lattice, mean size of loops is $O(N^{\approx 0.75})$.
- $L = 8192$, $N \approx 67$ million: 360 thousand steps, versus 6.77 tree rotations.
- For simple cubic lattice, mean size of loops is $O(N)$.
- $L = 512$, $N \approx 134$ million:
  - Reverse 46 million steps, $\cdots$

Backbite algorithm performance

- Binary tree implementation: to reverse $O(N)$ steps requires $O(\log N)$ tree rotations.
- For square lattice, mean size of loops is $O(N^{\approx 0.75})$.
- $L = 8192$, $N \approx 67$ million: 360 thousand steps, versus 6.77 tree rotations.
- For simple cubic lattice, mean size of loops is $O(N)$.
- $L = 512$, $N \approx 134$ million:
  - Reverse 46 million steps, $\cdots$
  - $\cdots$ or perform 20.6 tree rotations.

Is the backbite move global?

- When sampling Hamiltonian path configurations via backbite moves, the endpoints only move by two steps.

Is the backbite move global?

- When sampling Hamiltonian path configurations via backbite moves, the endpoints only move by two steps.
- Would like a truly global move.

Is the backbite move global?

- When sampling Hamiltonian path configurations via backbite moves, the endpoints only move by two steps.
- Would like a truly global move.
- Also, for efficiency, would like to study polymer melt with a single long polymer. (Avoids necessity of taking two limits, polymer length N, number of polymers k, both $N, k \to \infty$.)

Is the backbite move global?

- When sampling Hamiltonian path configurations via backbite moves, the endpoints only move by two steps.
- Would like a truly global move.
- Also, for efficiency, would like to study polymer melt with a single long polymer. (Avoids necessity of taking two limits, polymer length N, number of polymers k, both $N, k \to \infty$.)
- What if we use periodic boundary conditions? (PBC)

A single dense polymer with PBC

- Can explicitly see random walk behaviour for $d = 3$ when we unwind it.

A single dense polymer with PBC

- Can explicitly see random walk behaviour for $d = 3$ when we unwind it.
- Multiple copies are space filling, because in reduced coordinates the walk must fill the original grid.

A single dense polymer with PBC

- Can explicitly see random walk behaviour for $d = 3$ when we unwind it.
- Multiple copies are space filling, because in reduced coordinates the walk must fill the original grid.
- In the large N limit, do we get the same behaviour as the polymer melt?

A single dense polymer with PBC

- Can explicitly see random walk behaviour for $d = 3$ when we unwind it.
- Multiple copies are space filling, because in reduced coordinates the walk must fill the original grid.
- In the large N limit, do we get the same behaviour as the polymer melt?
- For a segment of length k to wind around box we expect that:

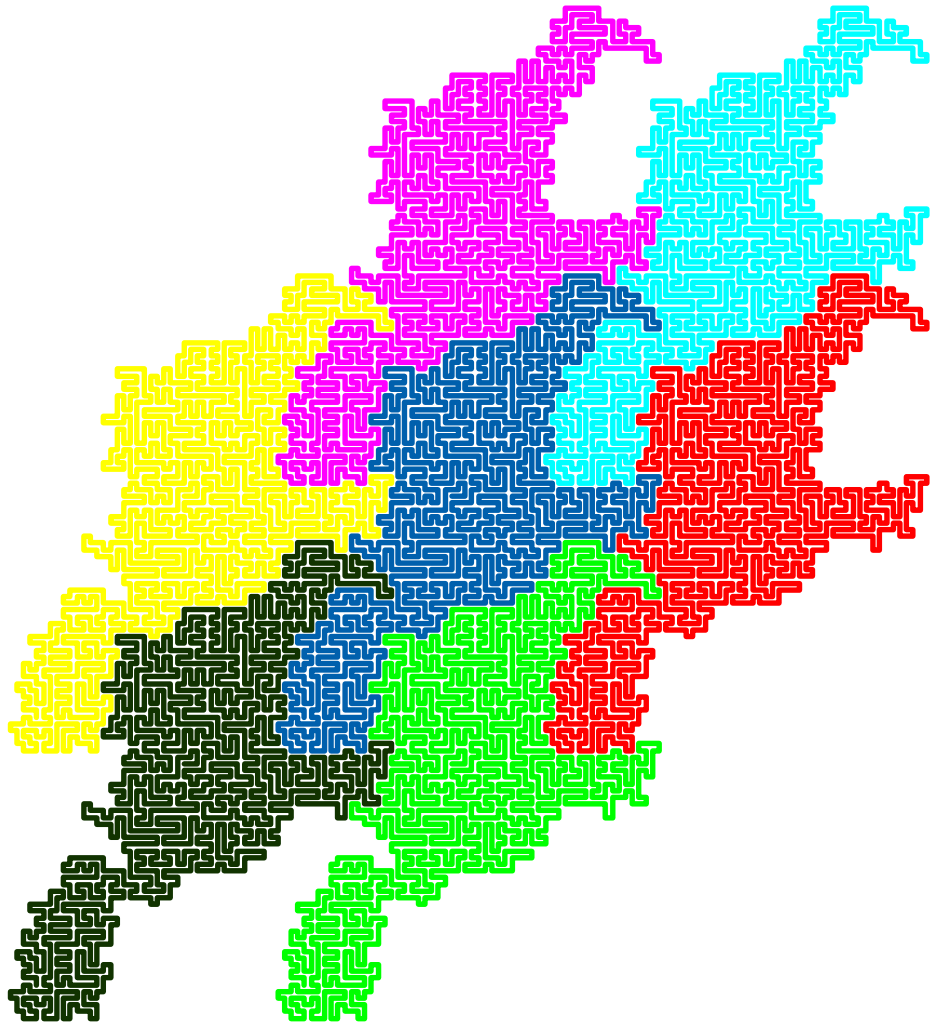
$$k^{1/2} = L = N^{1/d};$$
$$\Rightarrow k = O(N^{2/d}).$$

A single dense polymer with PBC

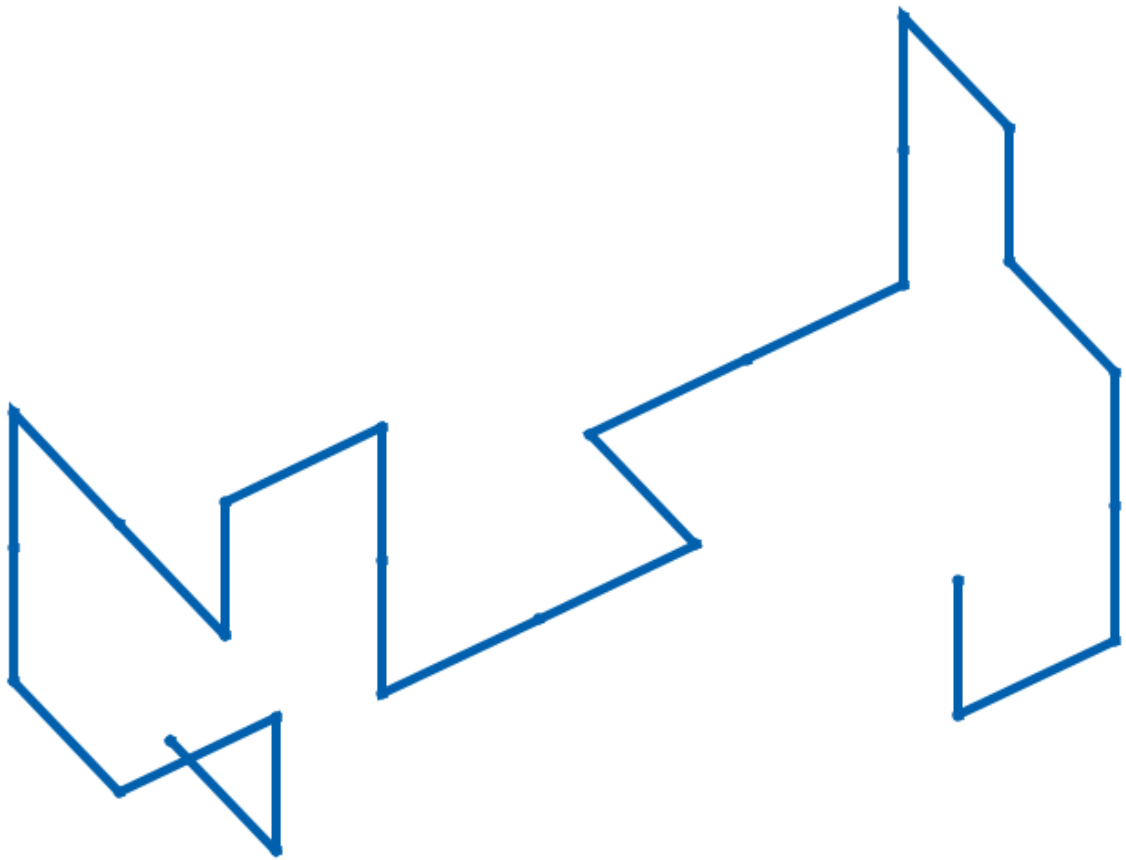
- Can explicitly see random walk behaviour for $d = 3$ when we unwind it.
- Multiple copies are space filling, because in reduced coordinates the walk must fill the original grid.
- In the large N limit, do we get the same behaviour as the polymer melt?
- For a segment of length k to wind around box we expect that:

$$k^{1/2} = L = N^{1/d};$$
$$\Rightarrow k = O(N^{2/d}).$$

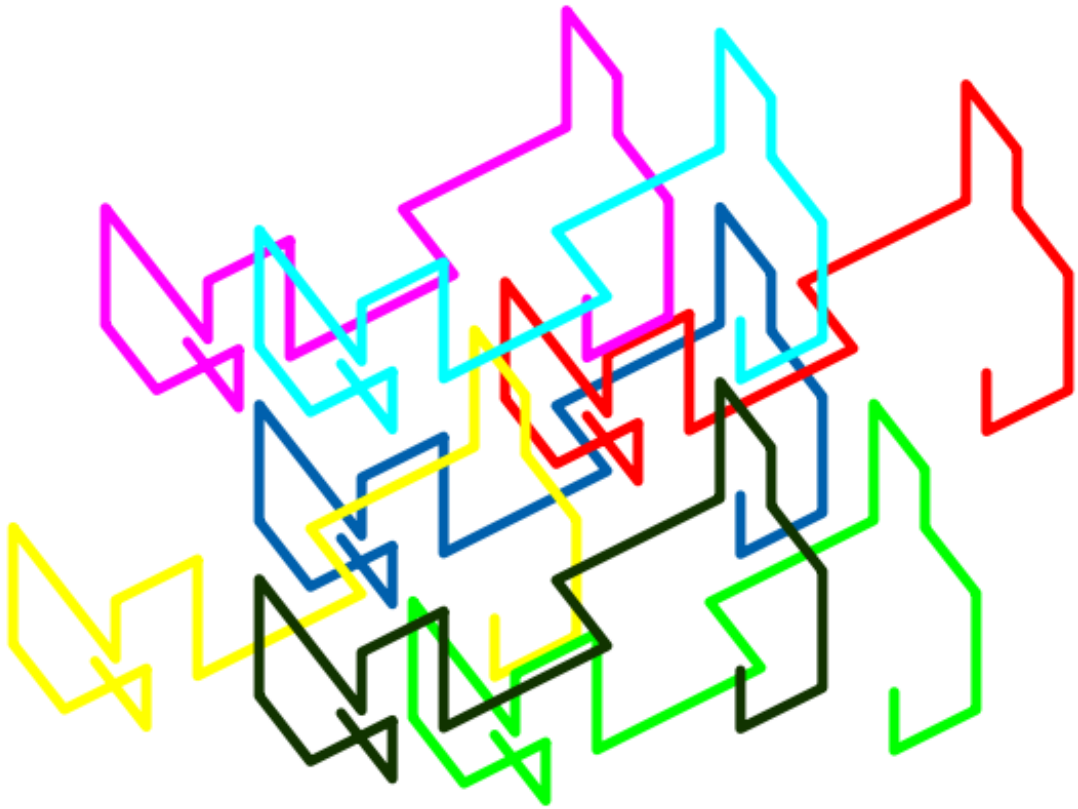
- In 2d, $k = O(N)$ and walks wind around $O(1)$ times. In 3d, $k = O(N^{2/3})$ and walks wind around $O(N^{1/3})$ times.



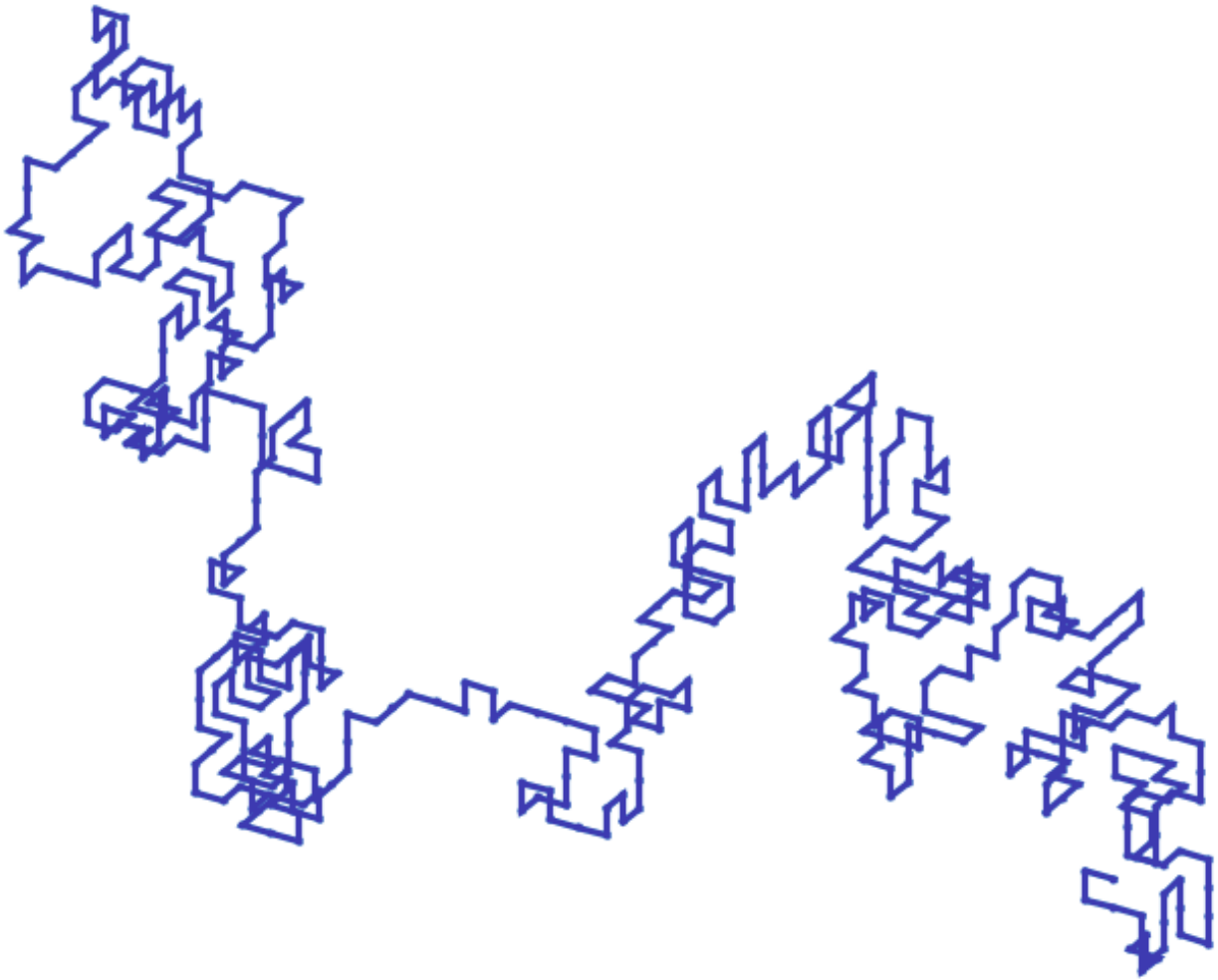
40×40 grid, with copies from winding. Random walk behaviour competing with space-filling nature (remains disc like).



$3 \times 3 \times 3$ grid.

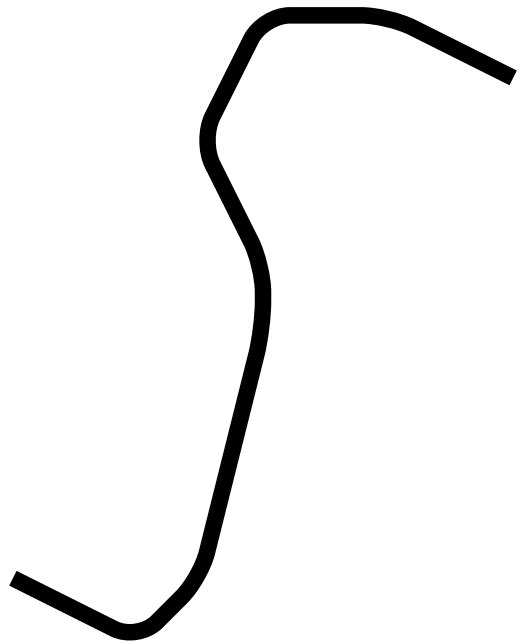


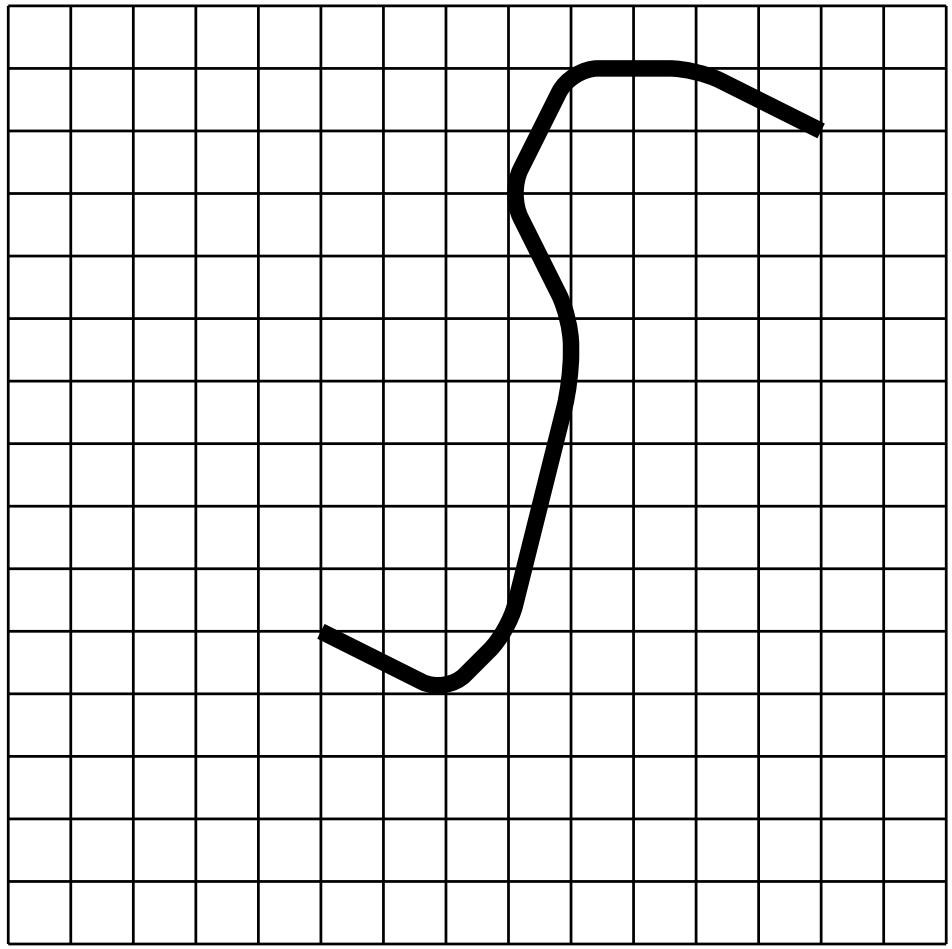
$3 \times 3 \times 3$ grid, with copies from winding around.



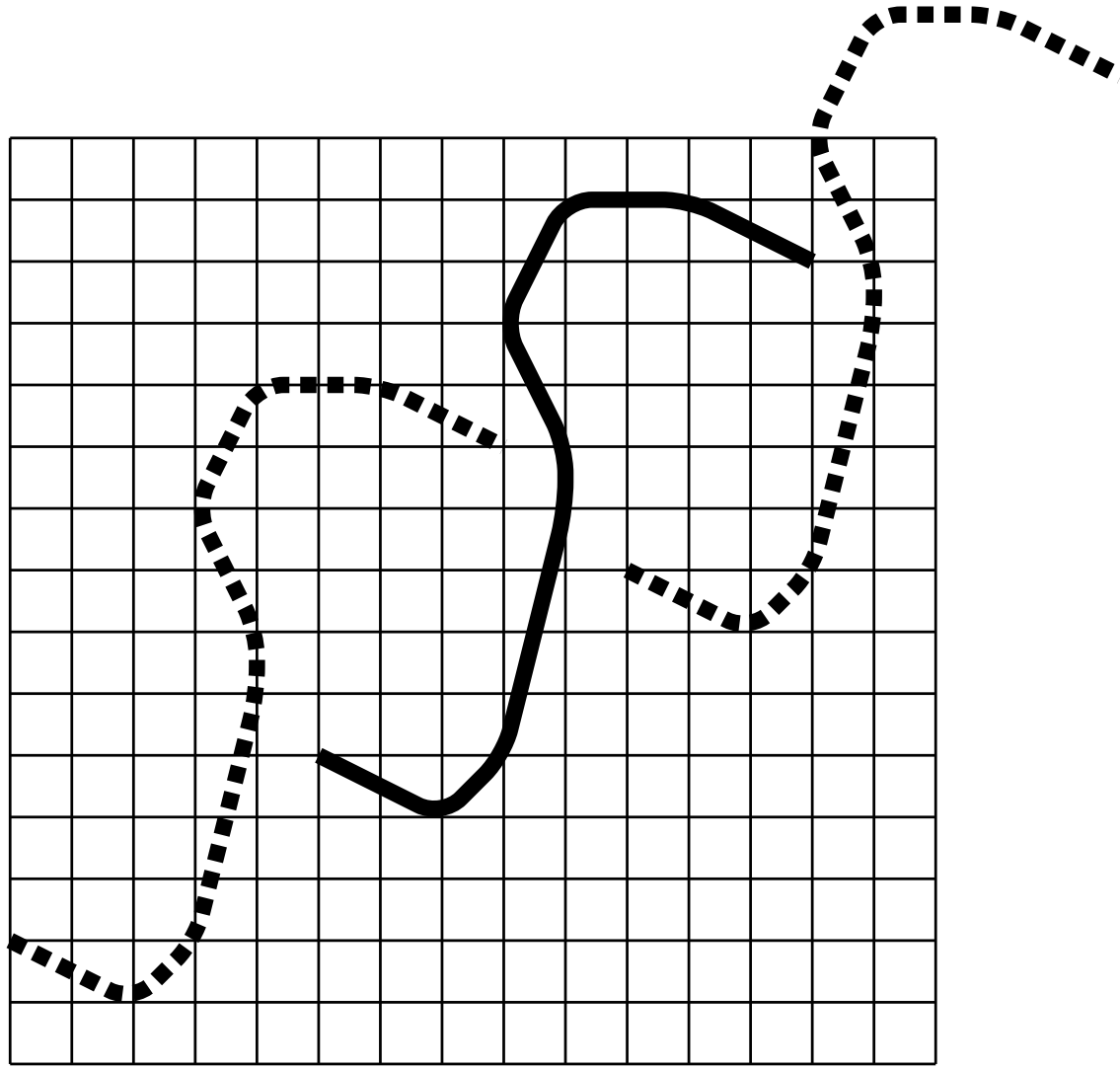
$8 \times 8 \times 8$ grid; random walk behaviour clearly observable.

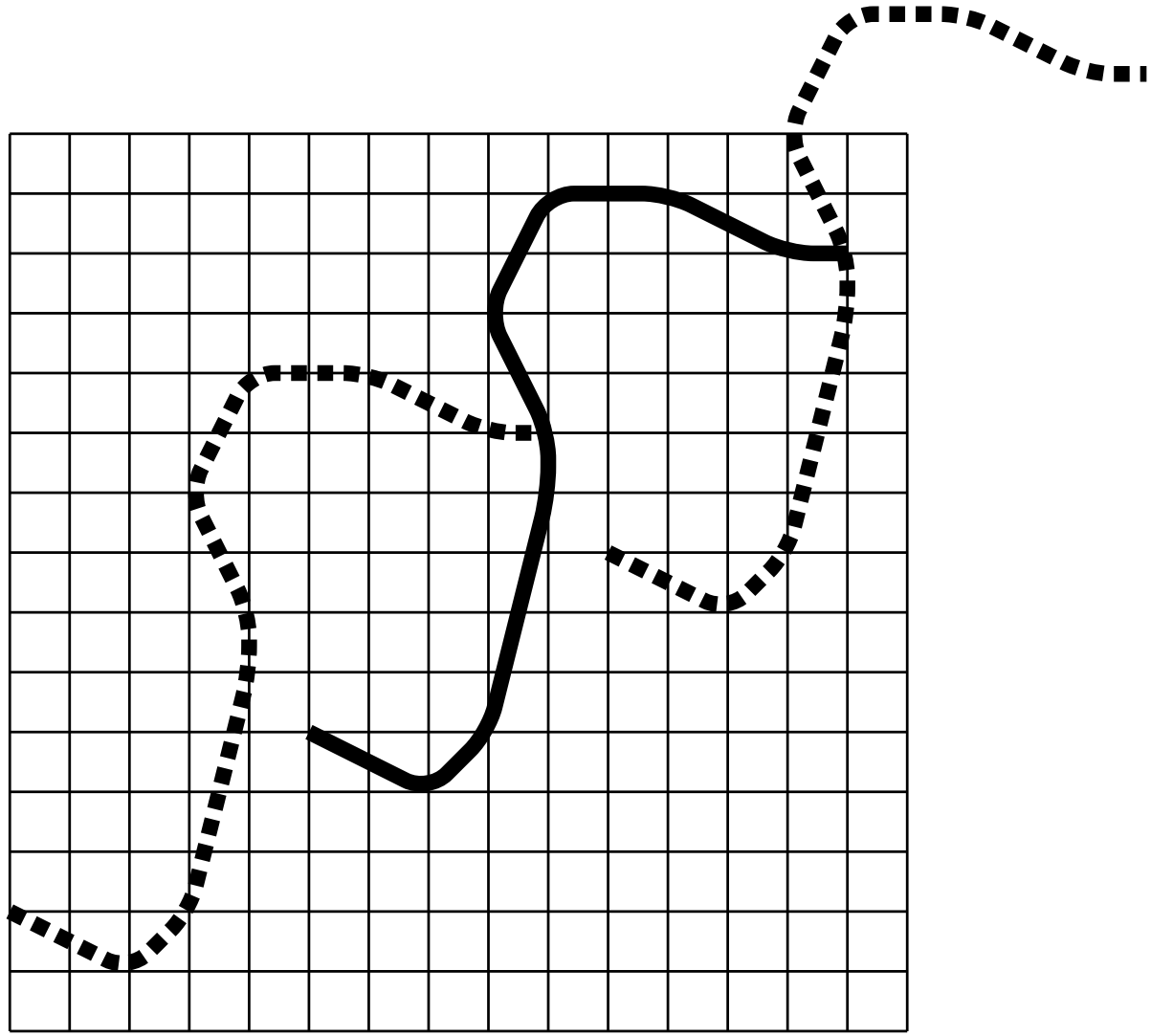
What are backbite moves like when we use PBC?

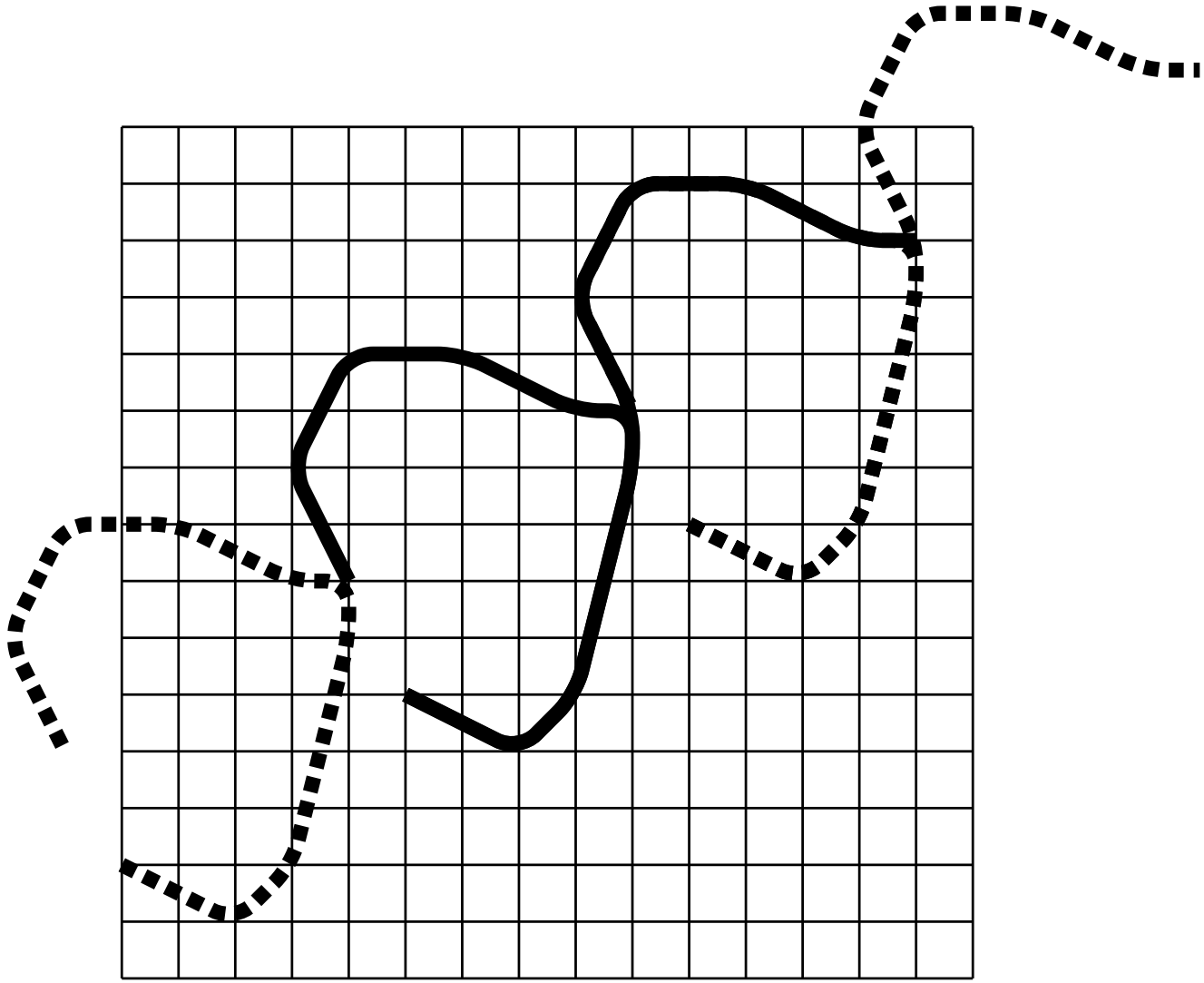


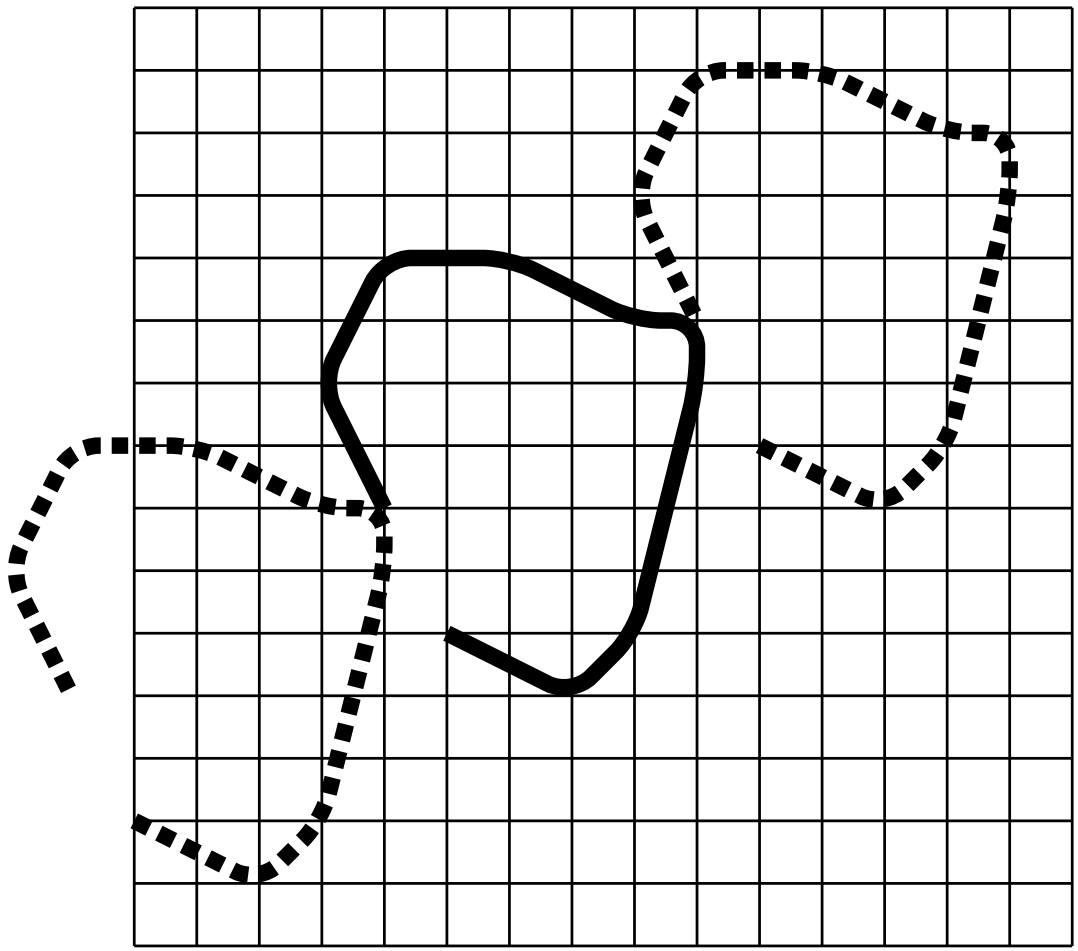


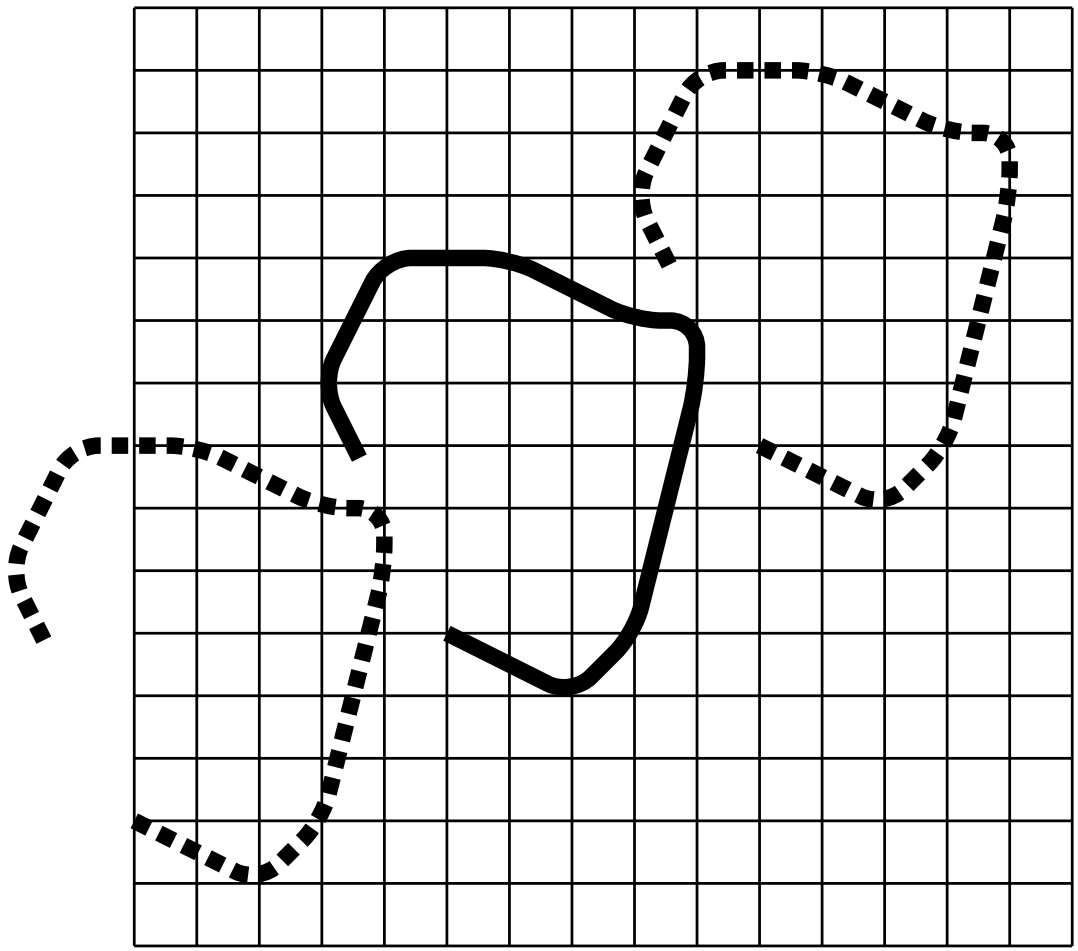
N.B.: 2d representation is a bit misleading, because phenomenon of random walk diffusion winding around multiple times occurs for $d > 2$.

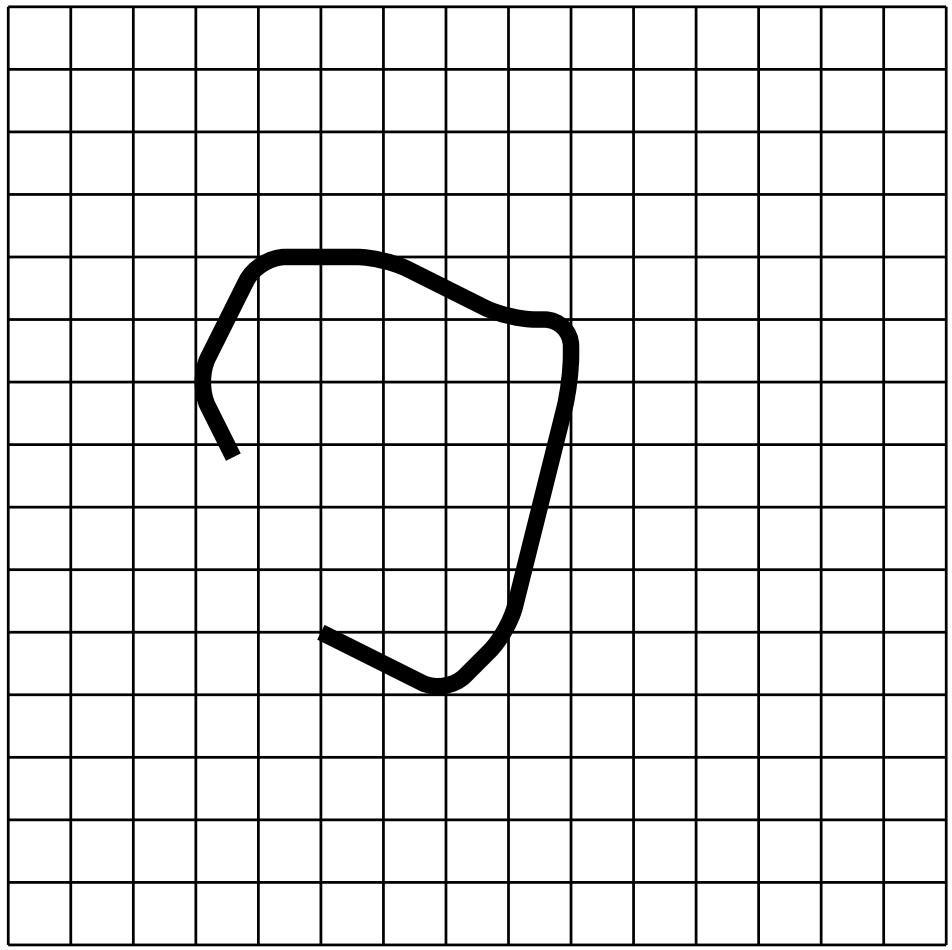


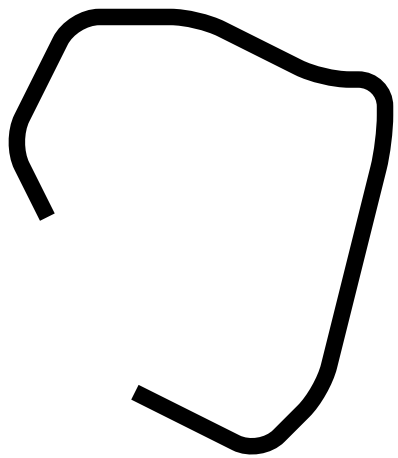


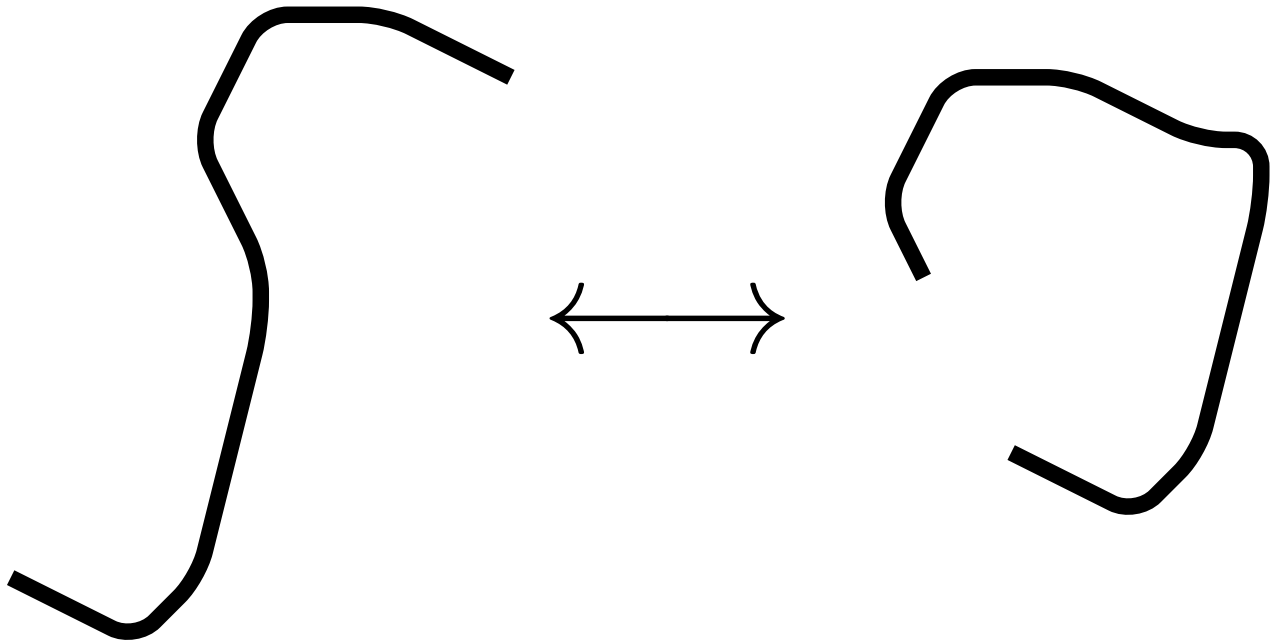






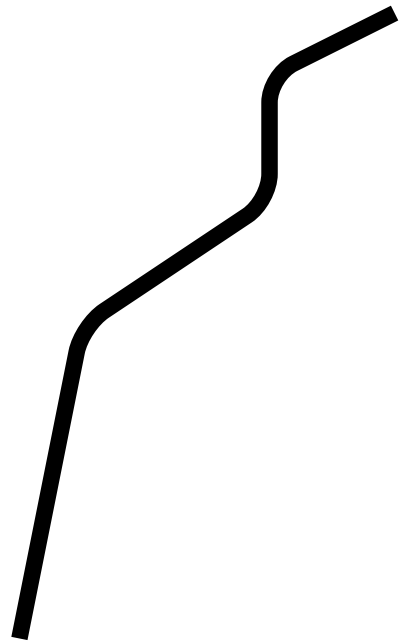


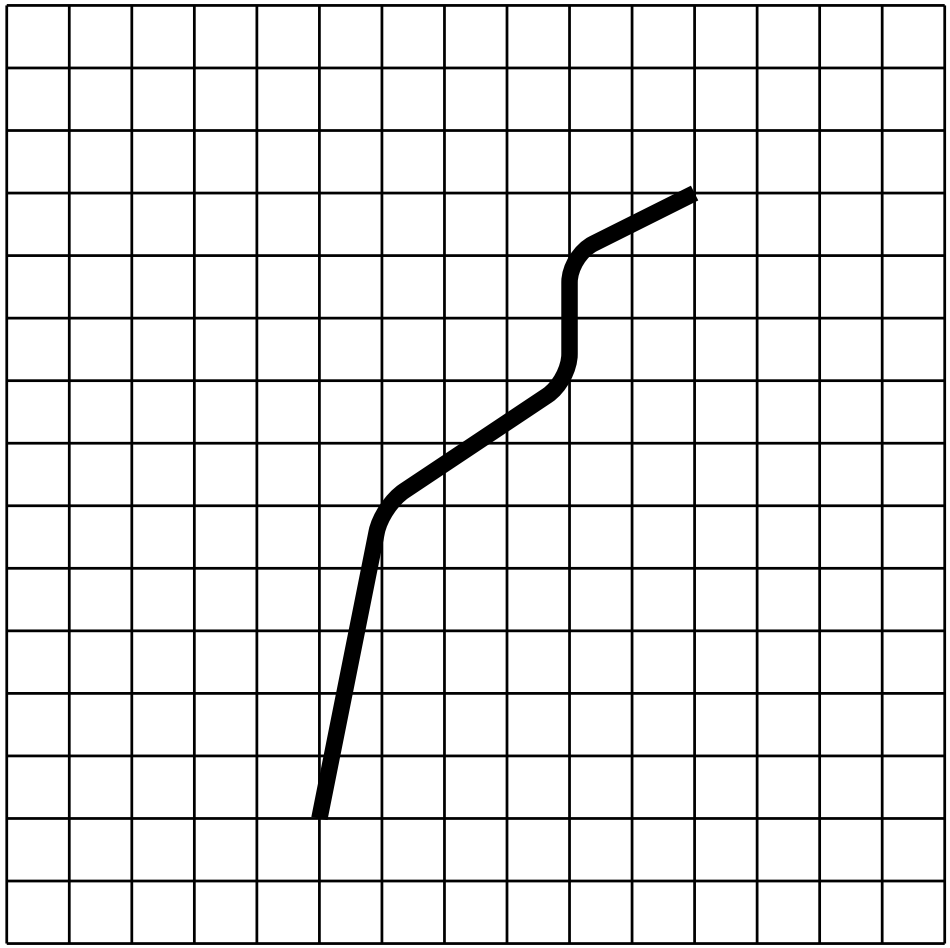


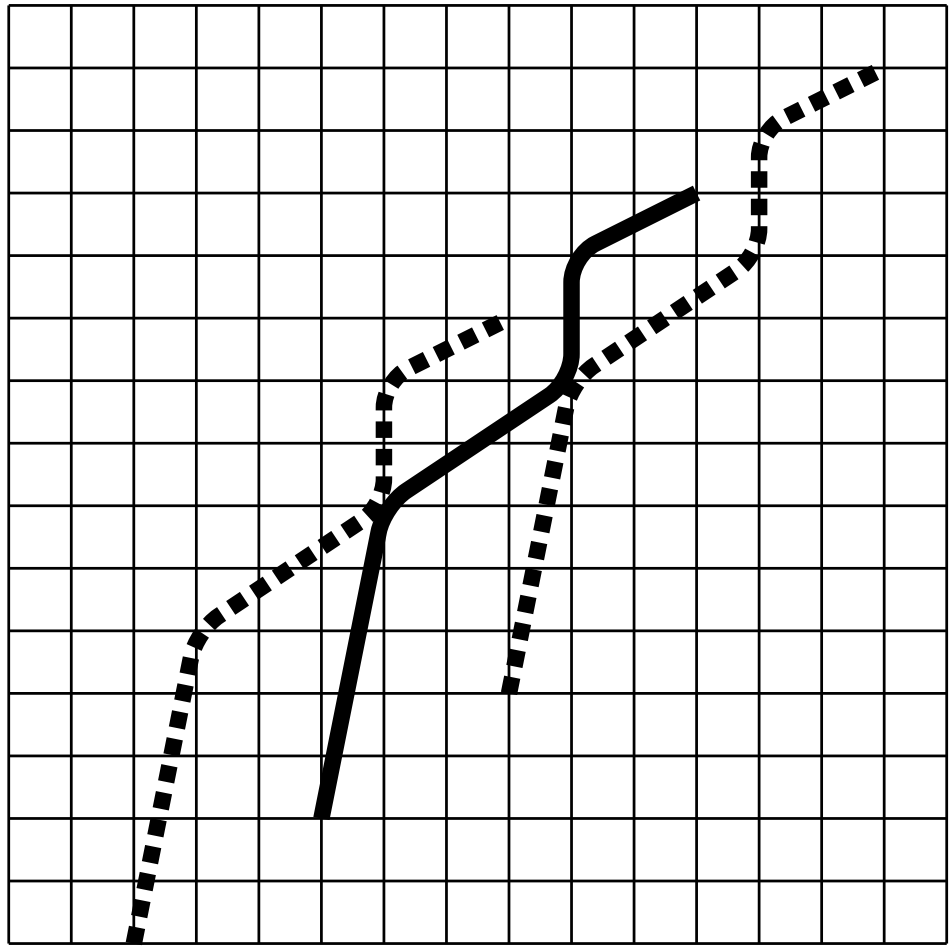


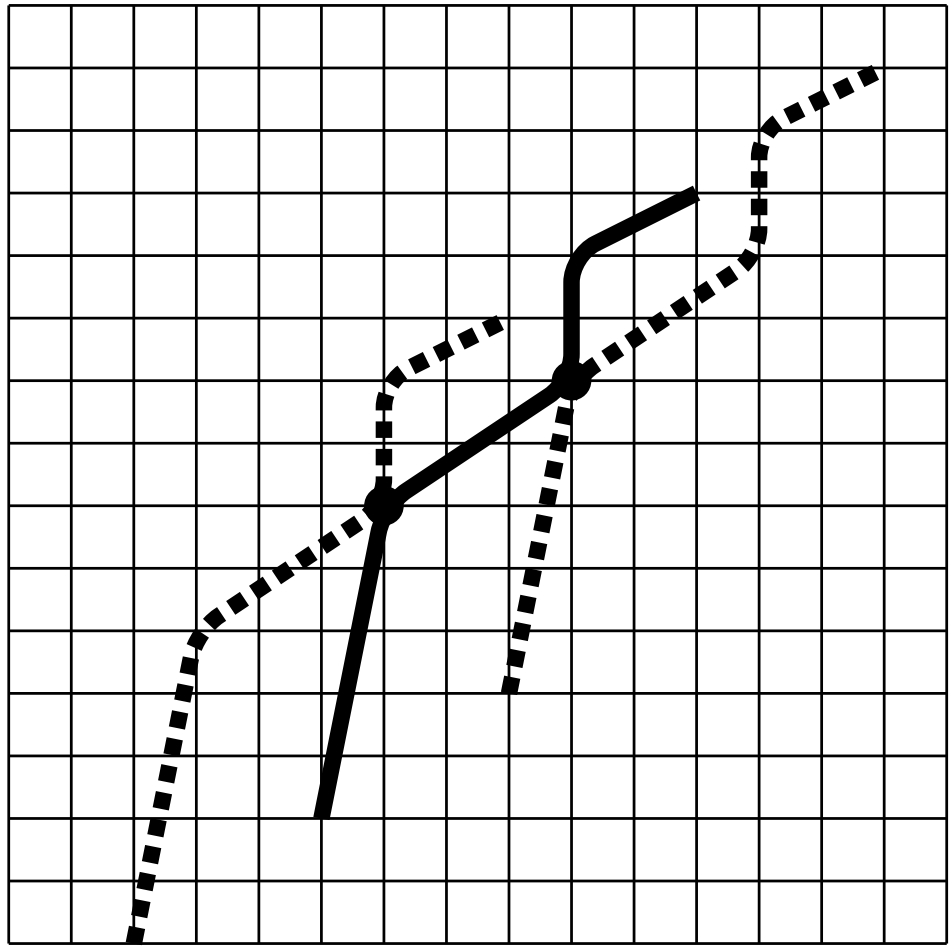
With PBC, endpoints can shift by even multiples of the boundary length with each backbite move.

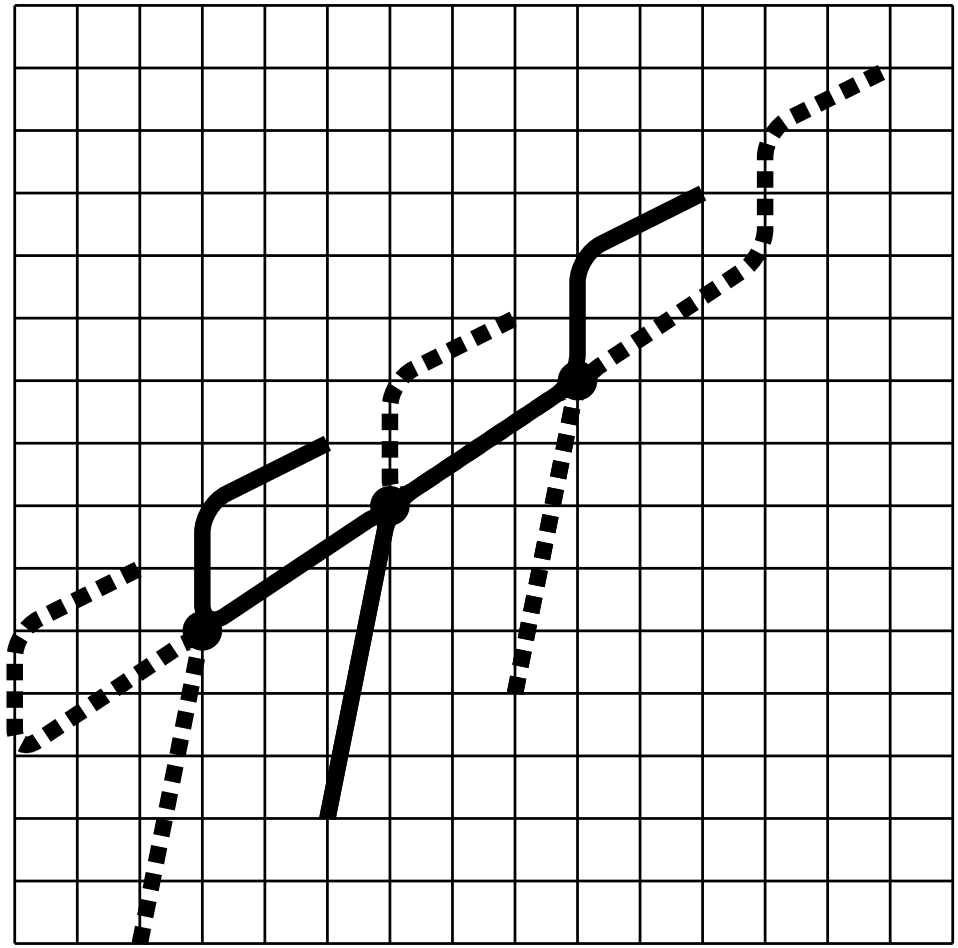
What are rebridging moves like when we use PBC?

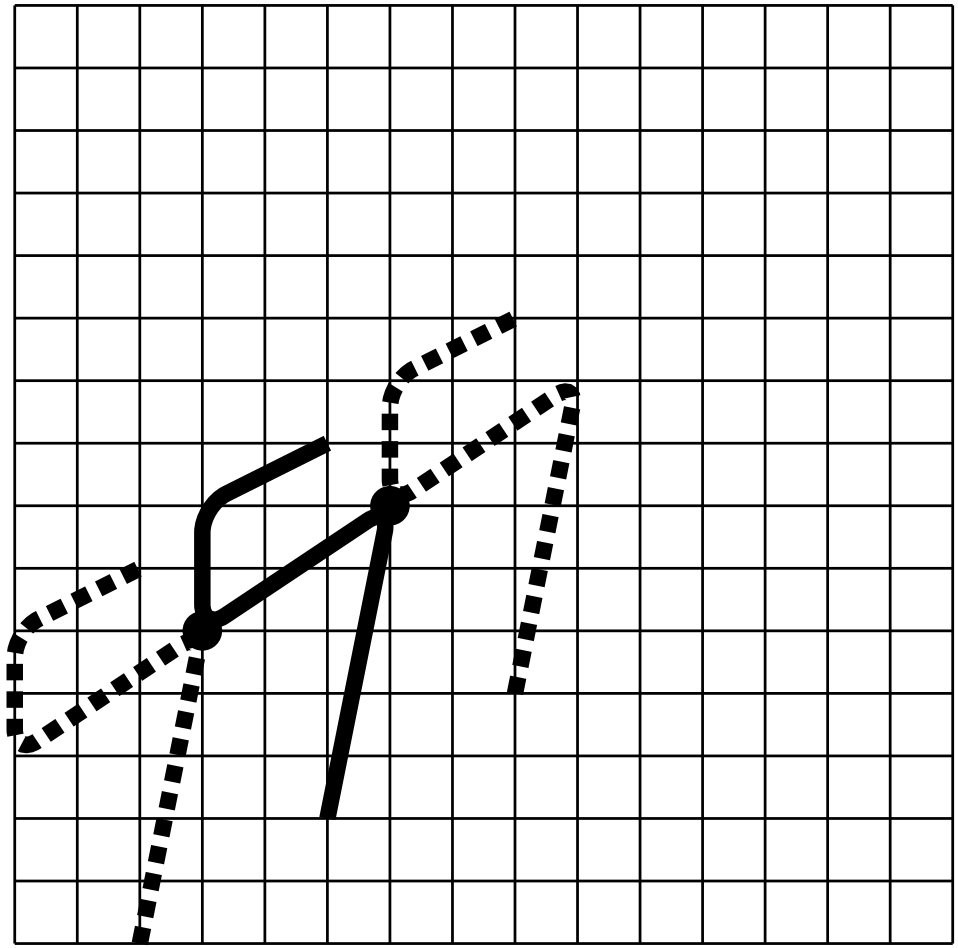


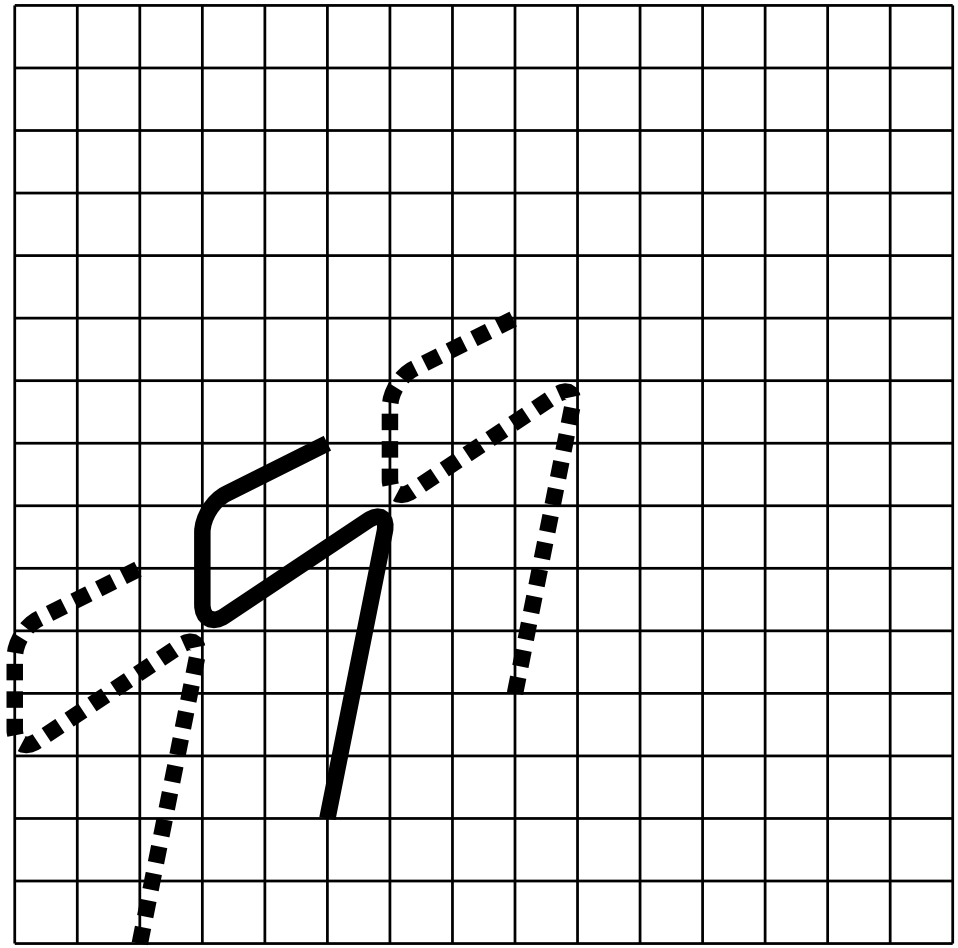


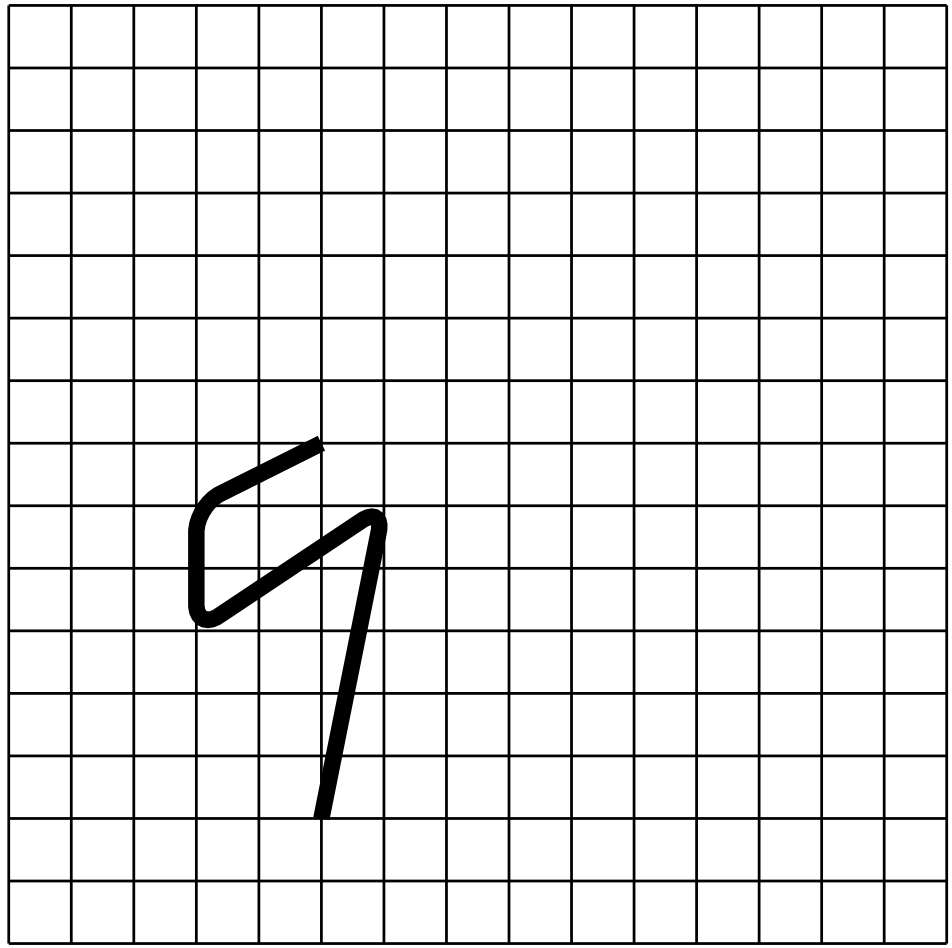


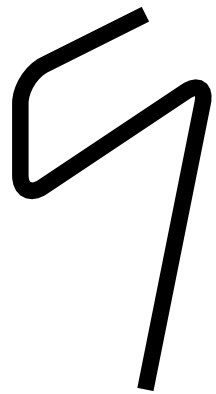


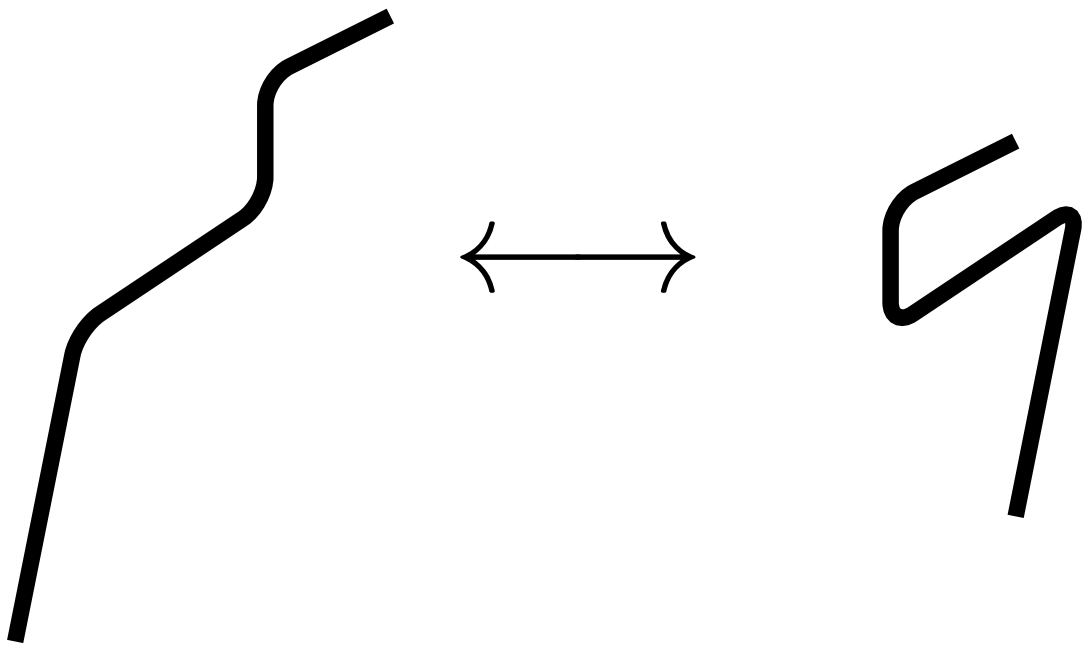




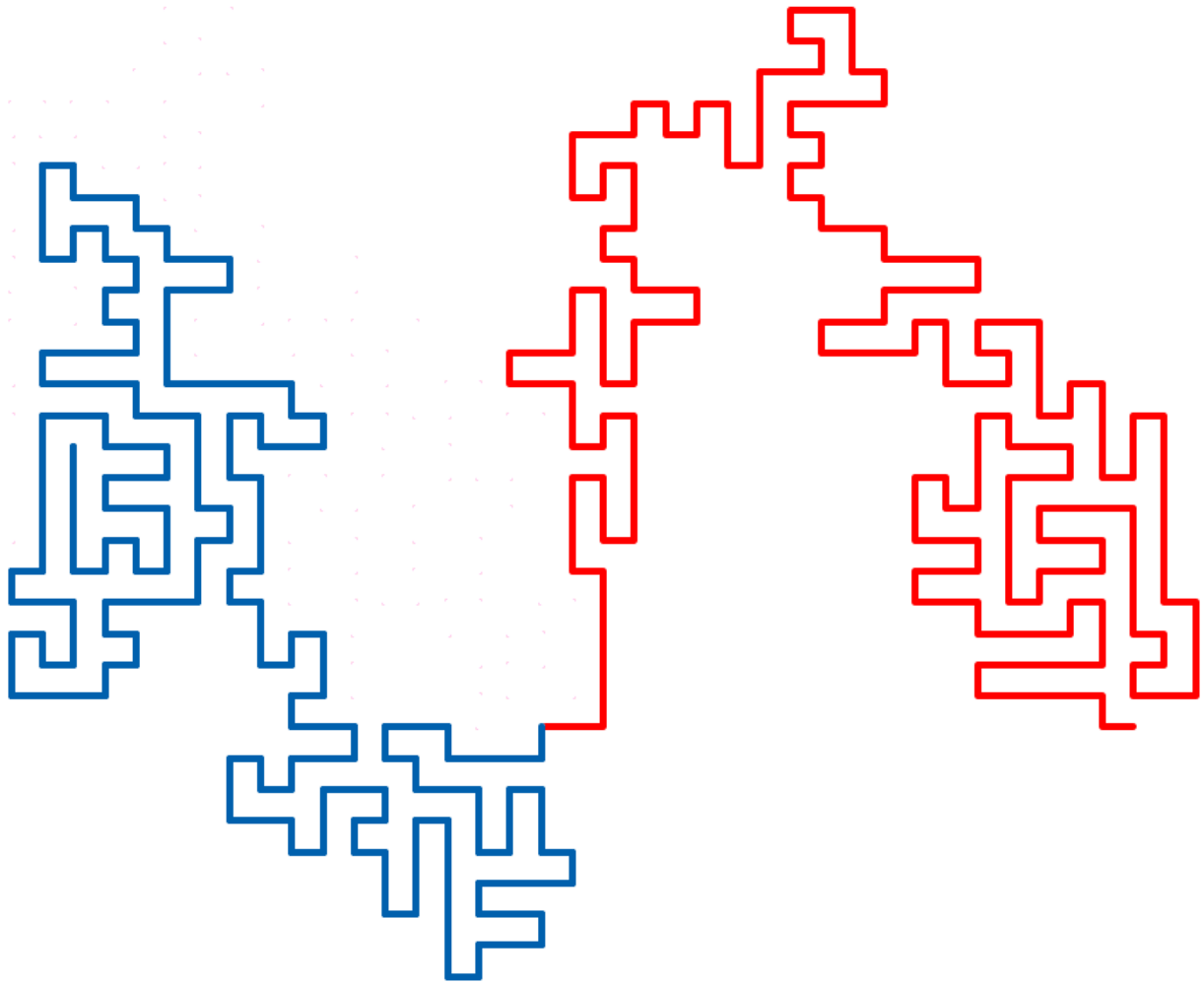


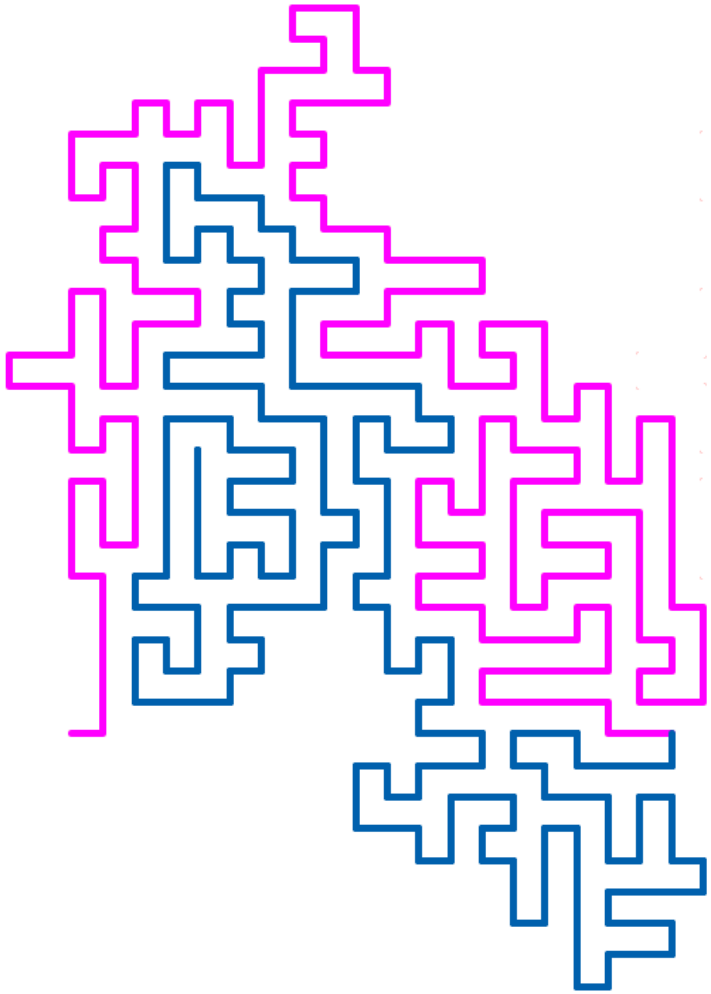


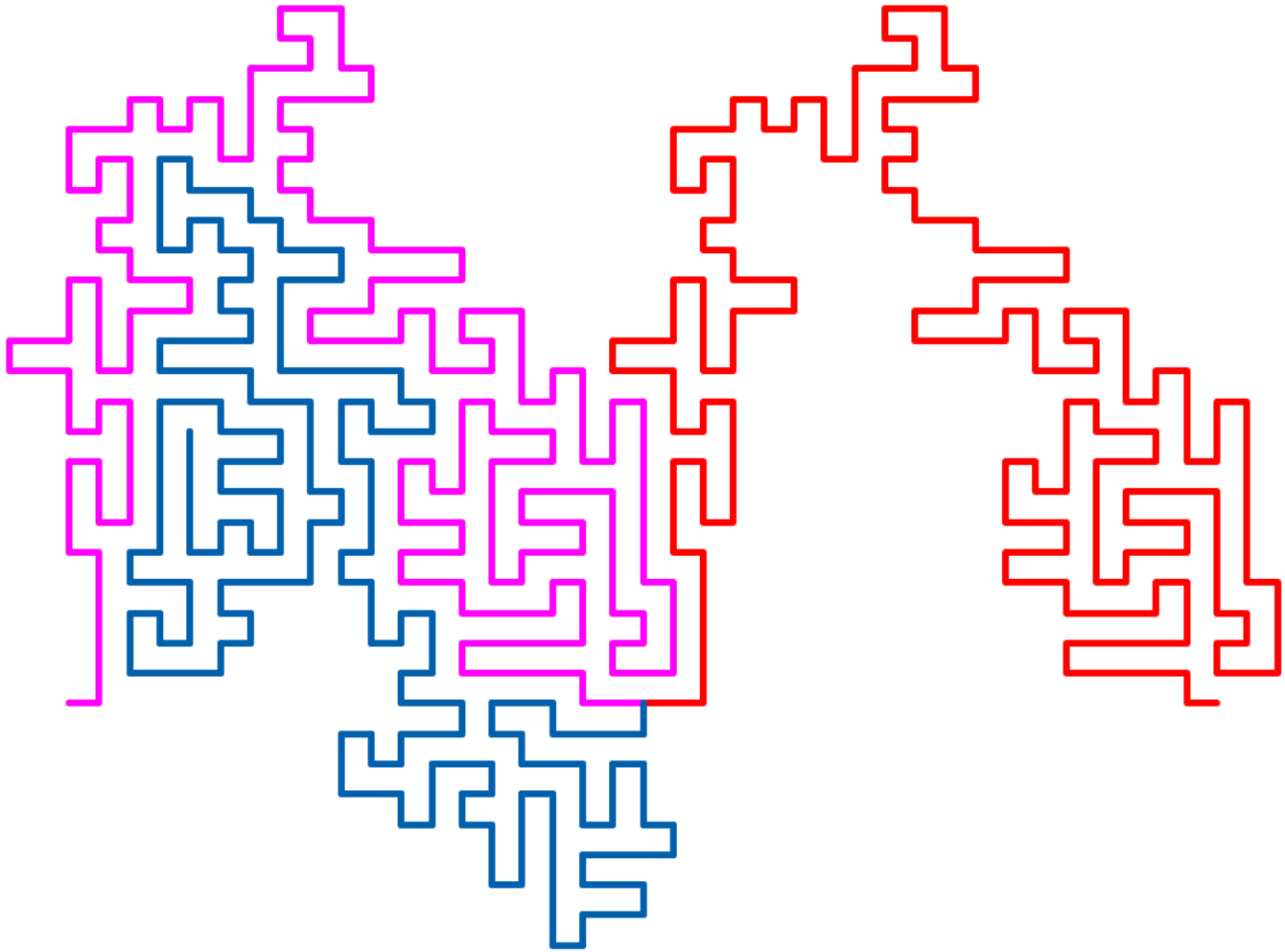




With PBC, endpoints can shift by even multiples of the boundary length.







Conclusion

- Backbite and other connectivity changing moves can be implemented in CPU time $O(\log N)$.

Conclusion

- Backbite and other connectivity changing moves can be implemented in CPU time $O(\log N)$.
- Allows for rapid sampling of dense polymers with millions of monomers.

Conclusion

- Backbite and other connectivity changing moves can be implemented in CPU time $O(\log N)$.
- Allows for rapid sampling of dense polymers with millions of monomers.
- With PBC, backbite moves are global, in the sense that they make macroscopic changes to position of endpoints.

Conclusion

- Backbite and other connectivity changing moves can be implemented in CPU time $O(\log N)$.
- Allows for rapid sampling of dense polymers with millions of monomers.
- With PBC, backbite moves are global, in the sense that they make macroscopic changes to position of endpoints.
- What is the interplay between different timescales? Large jumps in even multiples of the box dimensions, but slower exploration of the box in reduced coordinates.

Conclusion

- Backbite and other connectivity changing moves can be implemented in CPU time $O(\log N)$.
- Allows for rapid sampling of dense polymers with millions of monomers.
- With PBC, backbite moves are global, in the sense that they make macroscopic changes to position of endpoints.
- What is the interplay between different timescales? Large jumps in even multiples of the box dimensions, but slower exploration of the box in reduced coordinates.
- Key question: what is $\tau_{\mathrm{int}}(R_{\mathrm{E}}^2)$?

Conclusion

- Backbite and other connectivity changing moves can be implemented in CPU time $O(\log N)$.
- Allows for rapid sampling of dense polymers with millions of monomers.
- With PBC, backbite moves are global, in the sense that they make macroscopic changes to position of endpoints.
- What is the interplay between different timescales? Large jumps in even multiples of the box dimensions, but slower exploration of the box in reduced coordinates.
- Key question: what is $\tau_{\mathrm{int}}(R_{\mathrm{E}}^2)$?
- In large N limit, do periodic boundary conditions for a single polymer correspond to a single polymer in the midst of a melt?

Future work

- Study bond correlations within chain. (Use periodic boundary conditions?)
- Combine Monte Carlo with molecular dynamics to study dynamics. E.g., melts of rings are quite poorly understood, and their dynamics are even slower than for linear polymers as reptation is not possible.
- Develop efficient algorithms for other cases, e.g. θ-transition? Semi-dilute regime? Explicit solvent?
- Continuum models.
- Copolymers, star polymers, and many more applications.

Alternate representation of connectivity changing moves

- Can we find additional useful moves?

[9] S. Lin/B. W. Kernighan: An Effective Heuristic Algorithm for the Traveling-Salesman Problem, in: Oper. Res. 21 (1973), pp. 498–516.

[10] Pouya Baniasadi et al.: Deterministic Snakes and Ladders Heuristic for the Hamiltonian cycle problem, in: Mathematical Programming Computation 6 (2014), pp. 55–75.

Alternate representation of connectivity changing moves

- Can we find additional useful moves?
- Long history of algorithms to find optimal paths for the traveling salesman problem.

[9] S. Lin/B. W. Kernighan: An Effective Heuristic Algorithm for the Traveling-Salesman Problem, in: Oper. Res. 21 (1973), pp. 498–516.

[10] Pouya Baniasadi et al.: Deterministic Snakes and Ladders Heuristic for the Hamiltonian cycle problem, in: Mathematical Programming Computation 6 (2014), pp. 55–75.

Alternate representation of connectivity changing moves

- Can we find additional useful moves?
- Long history of algorithms to find optimal paths for the traveling salesman problem.
- Lin-Kernighan heuristic[9] involves 2-opt and 3-opt connectivity changing moves. No lattice, so these moves are represented differently.

[9] S. Lin/B. W. Kernighan: An Effective Heuristic Algorithm for the Traveling-Salesman Problem, in: Oper. Res. 21 (1973), pp. 498–516.

[10] Pouya Baniasadi et al.: Deterministic Snakes and Ladders Heuristic for the Hamiltonian cycle problem, in: Mathematical Programming Computation 6 (2014), pp. 55–75.

Alternate representation of connectivity changing moves

- Can we find additional useful moves?
- Long history of algorithms to find optimal paths for the traveling salesman problem.
- Lin-Kernighan heuristic[9] involves 2-opt and 3-opt connectivity changing moves. No lattice, so these moves are represented differently.
- Hamiltonian path search problem for a graph is a special case of the traveling salesman problem.[10]

[9] S. Lin/B. W. Kernighan: An Effective Heuristic Algorithm for the Traveling-Salesman Problem, in: Oper. Res. 21 (1973), pp. 498–516.

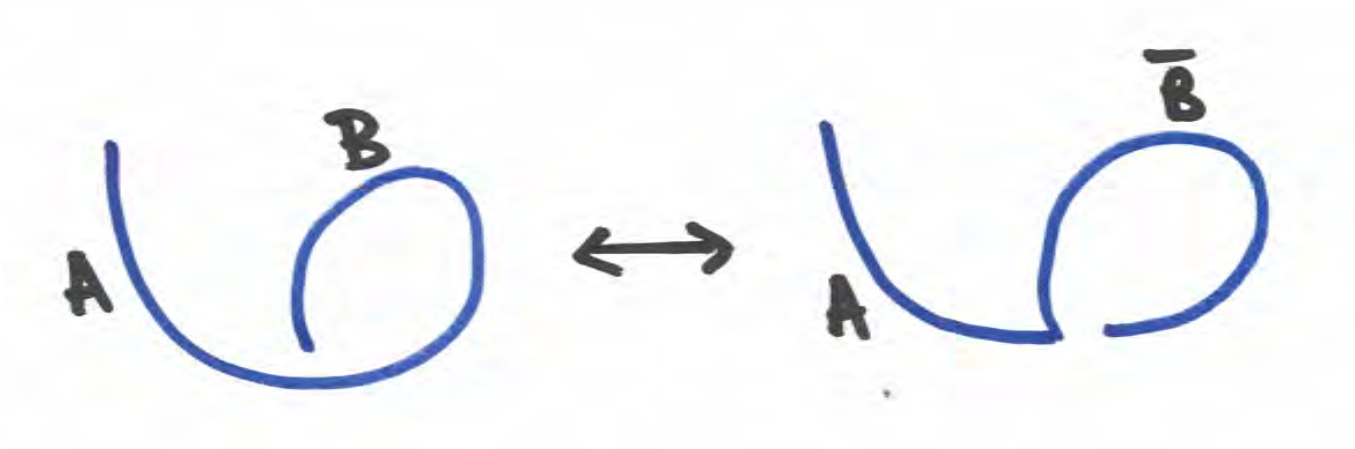
[10] Pouya Baniasadi et al.: Deterministic Snakes and Ladders Heuristic for the Hamiltonian cycle problem, in: Mathematical Programming Computation 6 (2014), pp. 55–75.

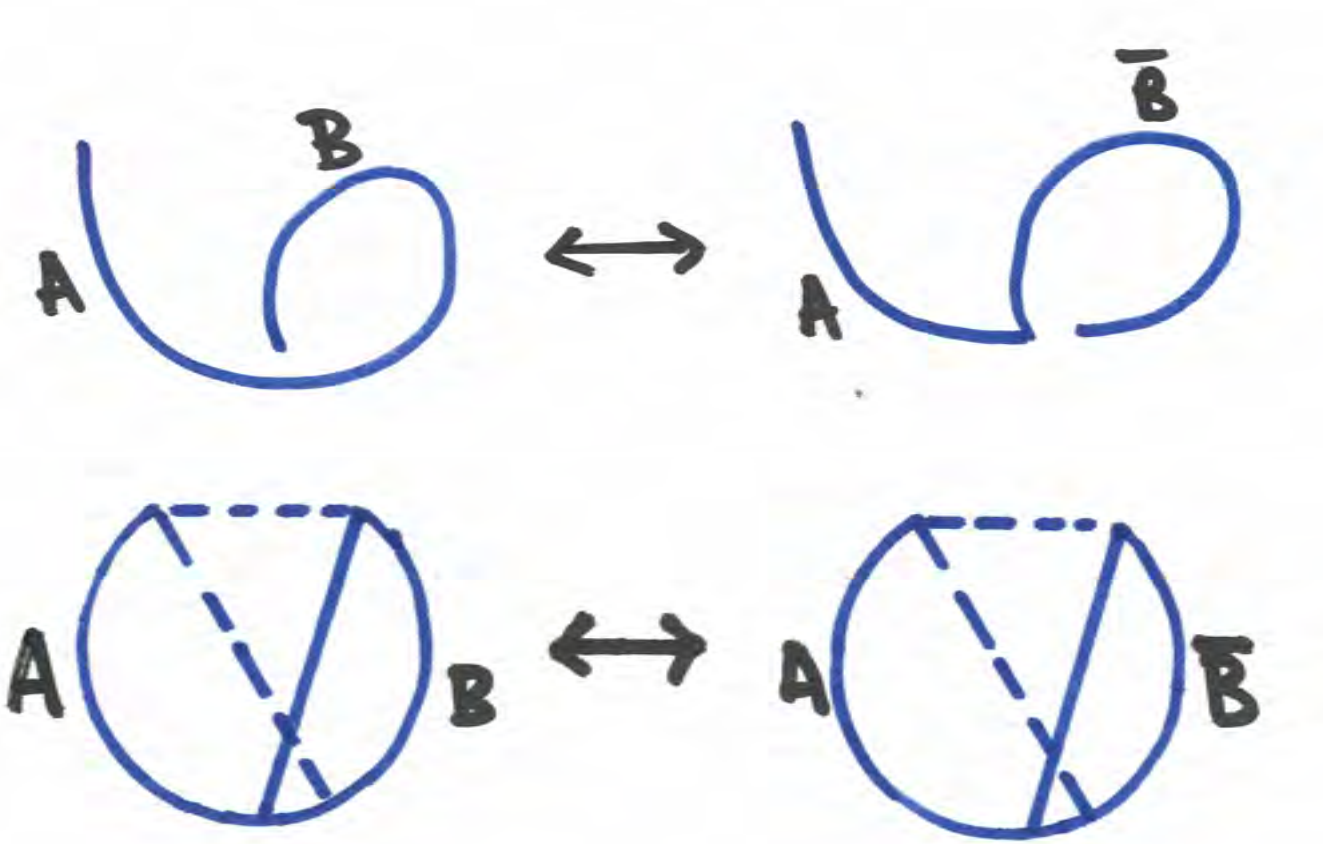
Alternate representation of connectivity changing moves

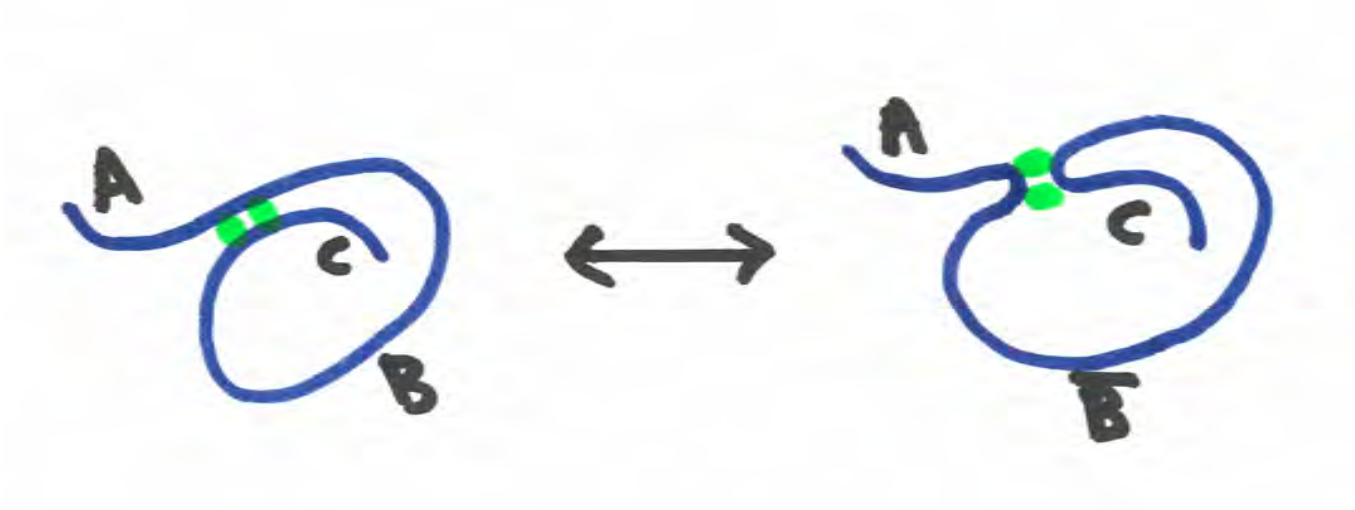
- Can we find additional useful moves?
- Long history of algorithms to find optimal paths for the traveling salesman problem.
- Lin-Kernighan heuristic[9] involves 2-opt and 3-opt connectivity changing moves. No lattice, so these moves are represented differently.
- Hamiltonian path search problem for a graph is a special case of the traveling salesman problem.[10]
- Any further insight if we abstract away the underlying lattice?

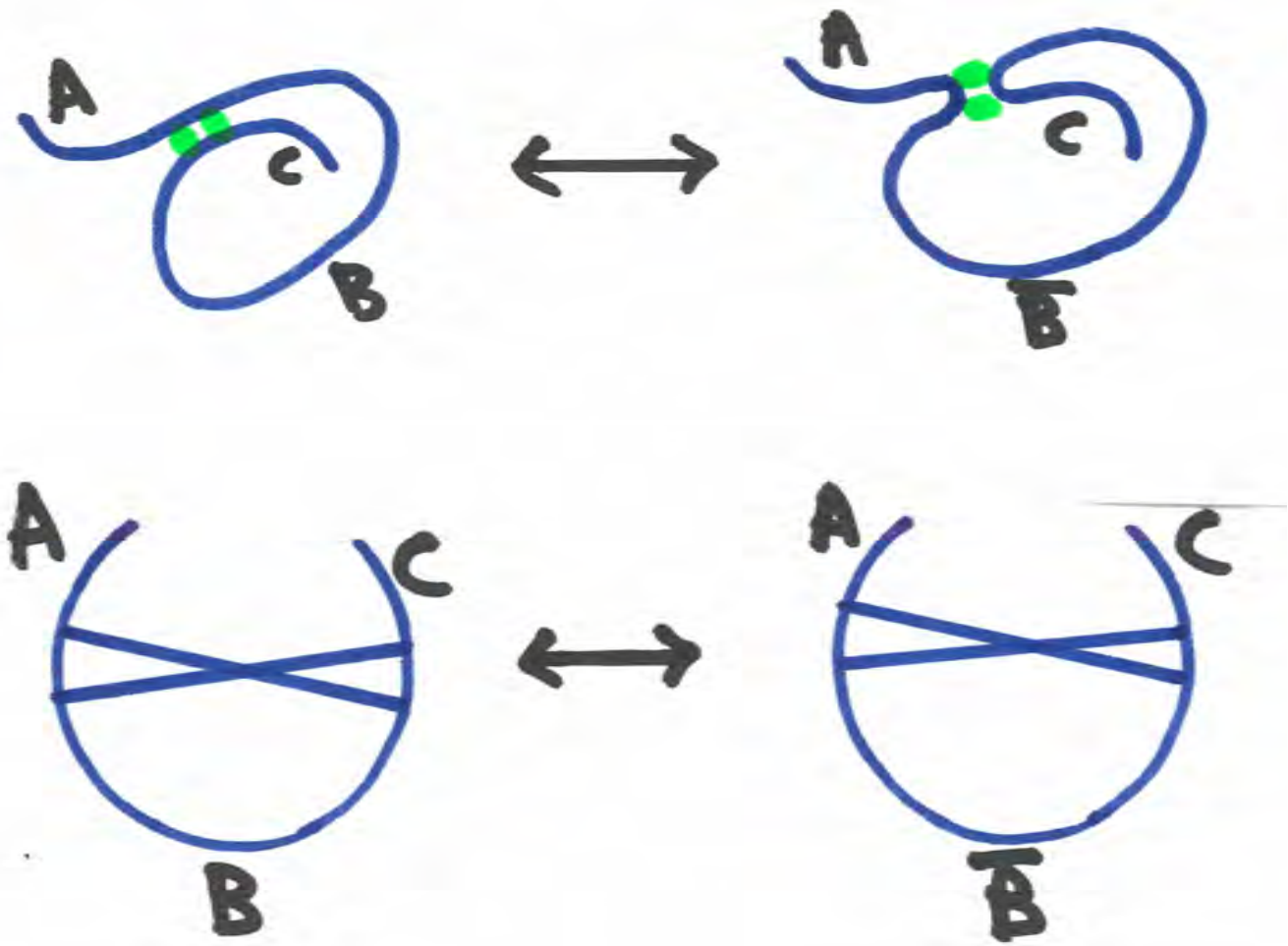
[9] S. Lin/B. W. Kernighan: An Effective Heuristic Algorithm for the Traveling-Salesman Problem, in: Oper. Res. 21 (1973), pp. 498–516.

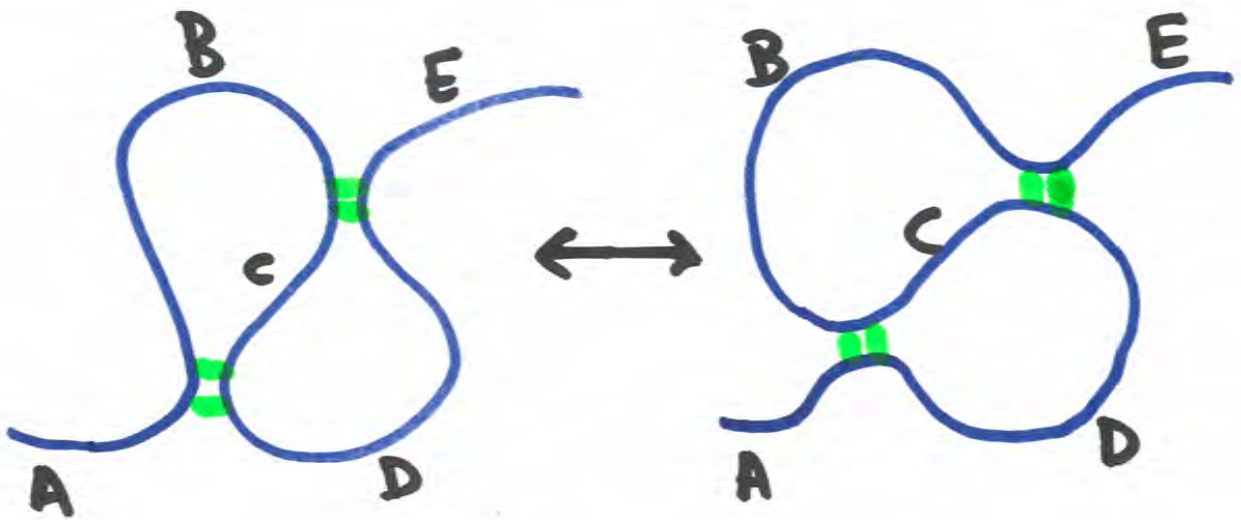
[10] Pouya Baniasadi et al.: Deterministic Snakes and Ladders Heuristic for the Hamiltonian cycle problem, in: Mathematical Programming Computation 6 (2014), pp. 55–75.

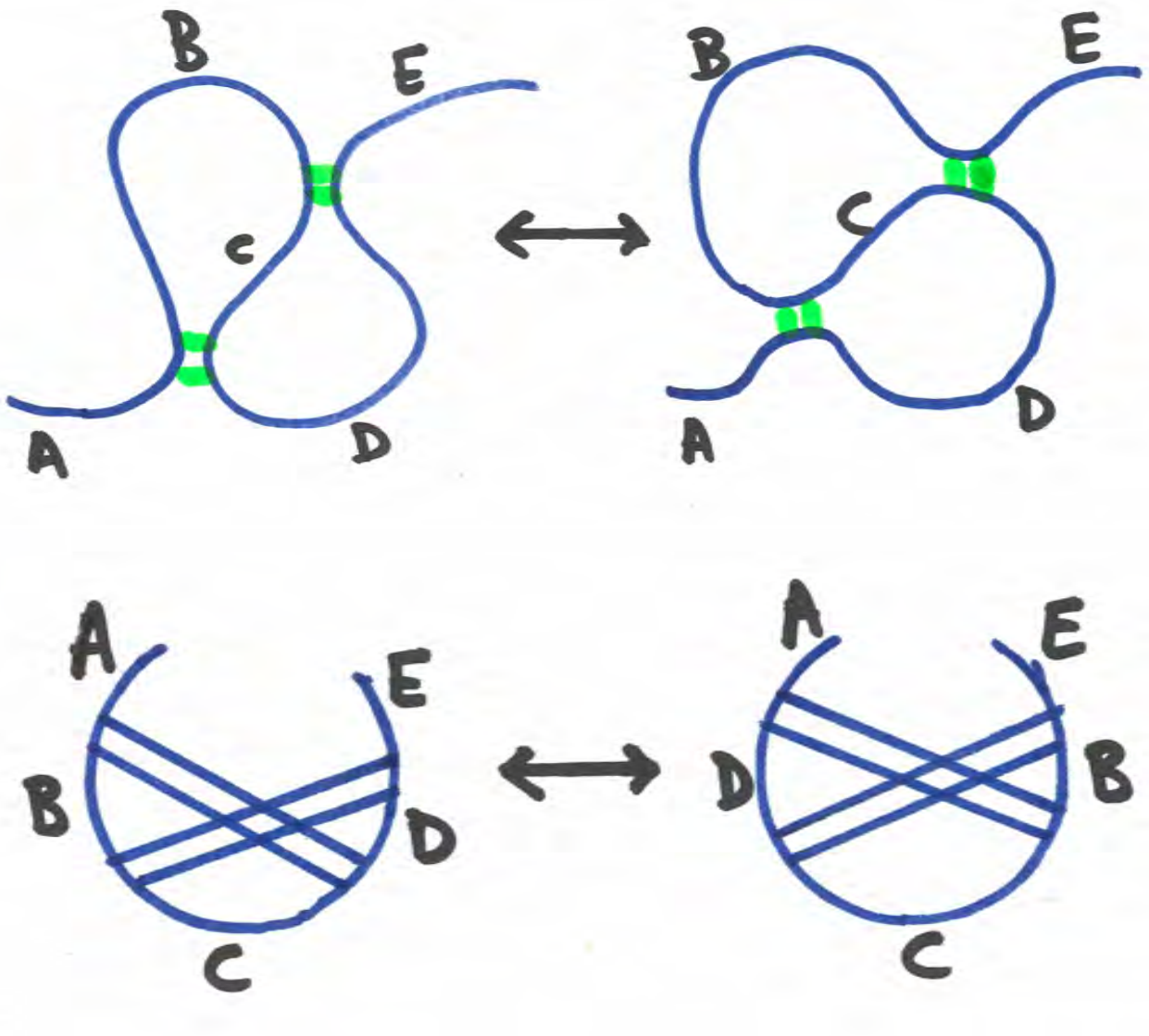

B
A
B
A

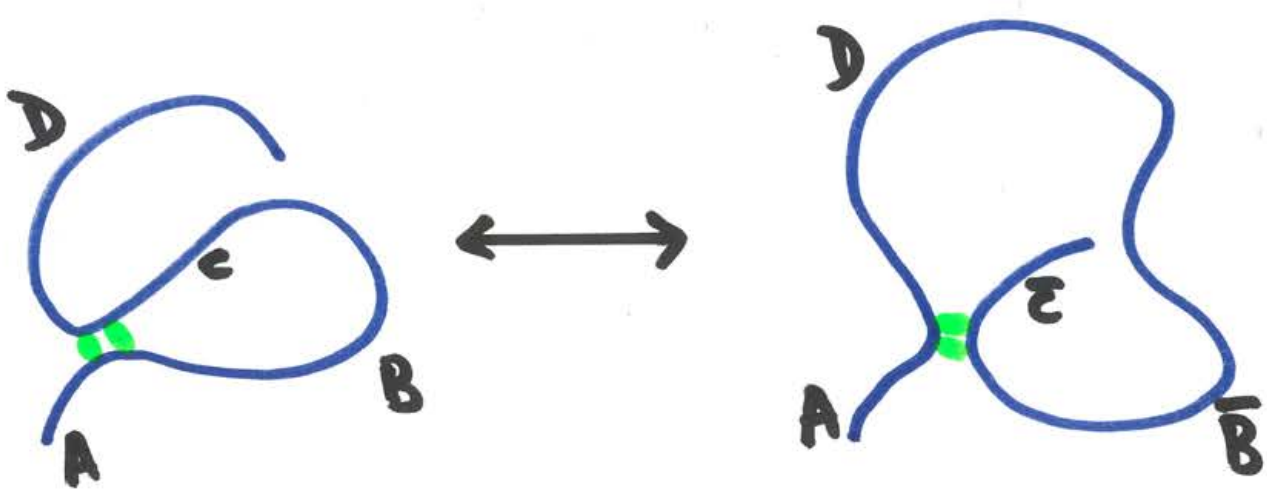

A
B
A
B
A
B
A
B

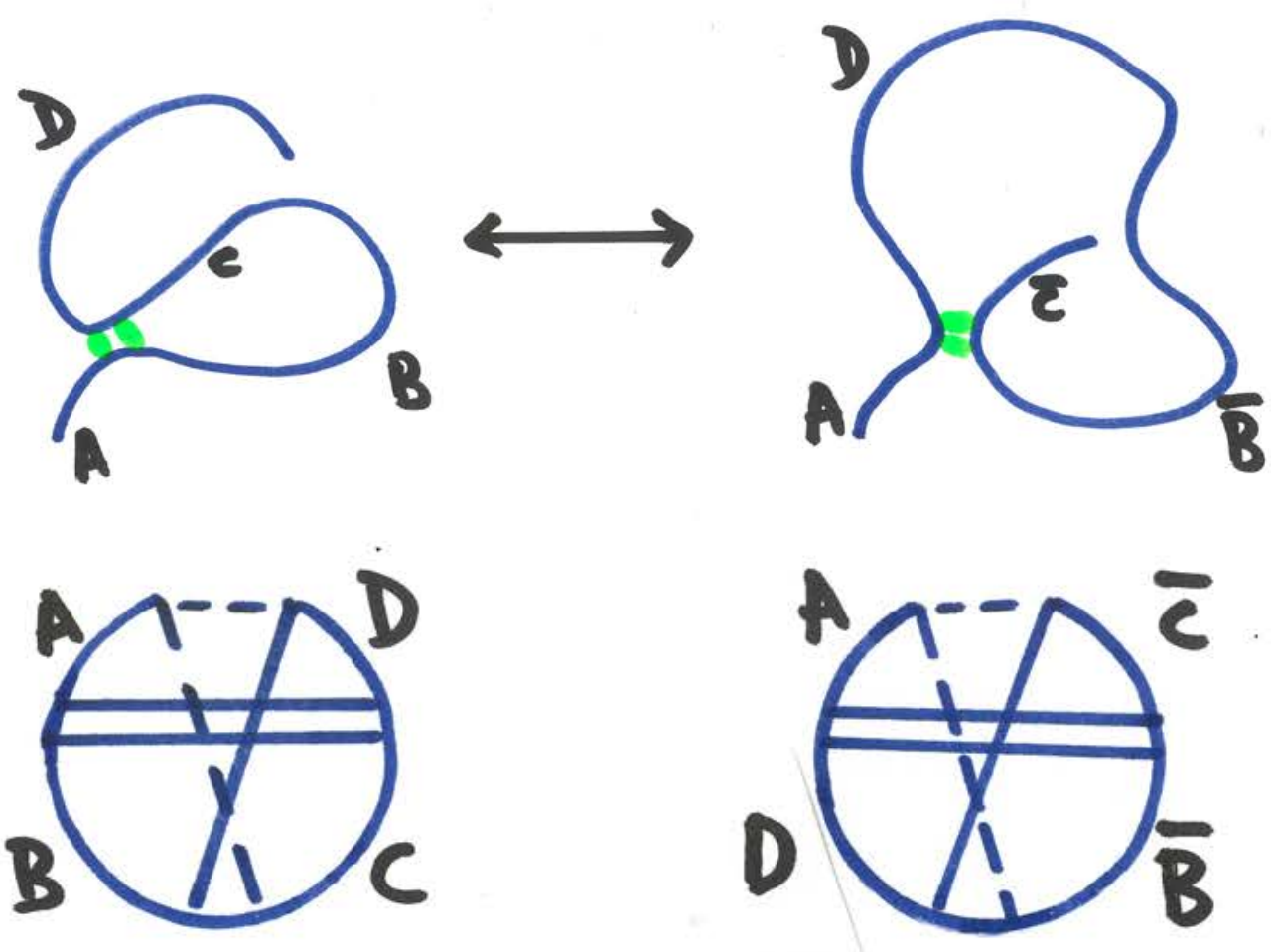

A
C
B
A
C
B


A
C
B
A
C
B
A
C
B
A
C
B


B
E
C
A
D
B
E
C
A
D


B
E
C
A
D
B
E
C
A
D
A
E
B
D
C
A
E
D
B
C


D
C
B
A
D
C
A
B


D
c
B
A
D
c
A
B
A
D
B
C
A
c
D
B

Alternate representation of connectivity-changing moves

- Abstracts away the lattice. Instead, think about transformations, and then determine which lattices are commensurate with those moves.

Alternate representation of connectivity-changing moves

- Abstracts away the lattice. Instead, think about transformations, and then determine which lattices are commensurate with those moves.
- One insight from this point of view so far: combination of backbite and rebridging may allow preferential sampling of moves close to the end. (Desirable for efficiency in 2d.)

Alternate representation of connectivity-changing moves

- Abstracts away the lattice. Instead, think about transformations, and then determine which lattices are commensurate with those moves.
- One insight from this point of view so far: combination of backbite and rebridging may allow preferential sampling of moves close to the end. (Desirable for efficiency in 2d.)
- Also, makes it easier to understand effect of moves, and how to implement them, as tree representation encodes linear information.

Computational efficiency

$d = 2$, $N = 10^9$

- Pr(success) $= 0.0189$
- 4-5 microseconds per pivot attempt
- 250 microseconds per pivot success

$d = 3$, $N = 10^9$

- Pr(success) $= 0.098$
- 9 microseconds per pivot attempt
- 90 microseconds per pivot success

$d = 4$, $N = 10^9$

- Pr(success) $= 0.50$
- 30 microseconds per pivot attempt
- 60 microseconds per pivot success