Efficient Monte Carlo simulation of polymers

Nathan Clisby
MASCOS, The University of Melbourne

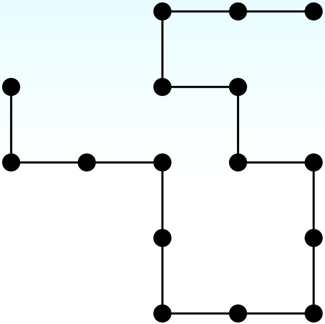
104th Statistical Mechanics Conference, Rutgers University
December 21, 2010

Self-avoiding walk model

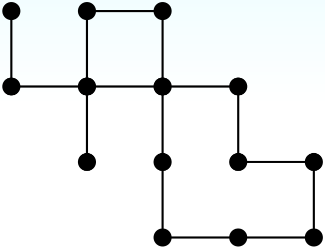
- Models polymers in good solvent limit.
- Exactly captures universal properties such as critical exponents.

Self-avoiding walk model

- Models polymers in good solvent limit.
- Exactly captures universal properties such as critical exponents.



SAW



Not a SAW

Critical phenomena

- The number of SAWs of length N, c_N, tells us about how many conformations are available to SAWs of a particular length:

$$c_N \sim A\ N^{\gamma-1}\mu^N\,[1 + \mathrm{corrections}]$$

- Mean square end to end distance tells us about the size of a typical SAW:

$$\langle R_{\mathrm{e}}^2\rangle_N \sim D_e N^{2\nu}\,[1 + \mathrm{corrections}]$$

- We wish to determine γ, ν, and μ as accurately as possible for SAWs on $\mathbb{Z}^3$.

Critical phenomena

- The number of SAWs of length N, c_N, tells us about how many conformations are available to SAWs of a particular length:

$$c_N \sim A\, N^{\gamma-1}\mu^N\,[1 + \mathrm{corrections}]$$

- Mean square end to end distance tells us about the size of a typical SAW:

$$\langle R_\mathrm{e}^2 \rangle_N \sim D_e N^{2\nu}\,[1 + \mathrm{corrections}]$$

- We wish to determine γ, ν, and μ as accurately as possible for SAWs on $\mathbb{Z}^3$.

Critical phenomena

- The number of SAWs of length N, c_N, tells us about how many conformations are available to SAWs of a particular length:

$$c_N \sim A\ N^{\gamma-1}\mu^N\,[1 + \mathrm{corrections}]$$

- Mean square end to end distance tells us about the size of a typical SAW:

$$\langle R_\mathrm{e}^2\rangle_N \sim D_e N^{2\nu}\,[1 + \mathrm{corrections}]$$

- We wish to determine γ, ν, and μ as accurately as possible for SAWs on $\mathbb{Z}^3$.

Monte Carlo

- Scheme: top-level method for sampling polymer configurations. Examples include: Markov chain Monte Carlo, umbrella sampling, Wang-Landau, PERM.
- Move set: operations which are used to generate new configurations from old ones.
- Implementation: polymer data structure.
- Focus here: moves and implementation.

Monte Carlo

- Scheme: top-level method for sampling polymer configurations. Examples include: Markov chain Monte Carlo, umbrella sampling, Wang-Landau, PERM.
- Move set: operations which are used to generate new configurations from old ones.
- Implementation: polymer data structure.
- Focus here: moves and implementation.

Monte Carlo

- Scheme: top-level method for sampling polymer configurations. Examples include: Markov chain Monte Carlo, umbrella sampling, Wang-Landau, PERM.
- Move set: operations which are used to generate new configurations from old ones.
- Implementation: polymer data structure.
- Focus here: moves and implementation.

Monte Carlo

- Scheme: top-level method for sampling polymer configurations. Examples include: Markov chain Monte Carlo, umbrella sampling, Wang-Landau, PERM.
- Move set: operations which are used to generate new configurations from old ones.
- Implementation: polymer data structure.
- Focus here: moves and implementation.

Markov chain Monte Carlo (MCMC)

- Sample from a probability distribution.
- Generate a new configuration from current one.
- Ensure that chain samples uniformly from whole set of configurations.
- Efficiency for calculating observable A determined by degree of correlation in the time series A_i. In particular, the integrated autocorrelation time τ_{int} of the chain.
- τ_{int} is the number of time steps necessary before an "essentially new" configuration is reached with respect to observable A.

Markov chain Monte Carlo (MCMC)

- Sample from a probability distribution.
- Generate a new configuration from current one.
- Ensure that chain samples uniformly from whole set of configurations.
- Efficiency for calculating observable A determined by degree of correlation in the time series A_i. In particular, the integrated autocorrelation time τ_{int} of the chain.
- τ_{int} is the number of time steps necessary before an "essentially new" configuration is reached with respect to observable A.

Markov chain Monte Carlo (MCMC)

- Sample from a probability distribution.
- Generate a new configuration from current one.
- Ensure that chain samples uniformly from whole set of configurations.
- Efficiency for calculating observable A determined by degree of correlation in the time series A_i. In particular, the integrated autocorrelation time τ_{int} of the chain.
- τ_{int} is the number of time steps necessary before an "essentially new" configuration is reached with respect to observable A.

Markov chain Monte Carlo (MCMC)

- Sample from a probability distribution.
- Generate a new configuration from current one.
- Ensure that chain samples uniformly from whole set of configurations.
- Efficiency for calculating observable A determined by degree of correlation in the time series A_i. In particular, the integrated autocorrelation time τ_{int} of the chain.
- τ_{int} is the number of time steps necessary before an "essentially new" configuration is reached with respect to observable A.

Markov chain Monte Carlo (MCMC)

- Sample from a probability distribution.
- Generate a new configuration from current one.
- Ensure that chain samples uniformly from whole set of configurations.
- Efficiency for calculating observable A determined by degree of correlation in the time series A_i. In particular, the integrated autocorrelation time τ_{int} of the chain.
- τ_{int} is the number of time steps necessary before an "essentially new" configuration is reached with respect to observable A.

Moves

- A move is a transformation of the SAW which may result in a new SAW.
- Local moves include one bead flips and the reptation or slithering snake move.
- Non-local moves include pivots and cut-and-permute moves.
- Many popular local and non-local moves can be defined in terms of "cut-and-paste" moves.

Moves

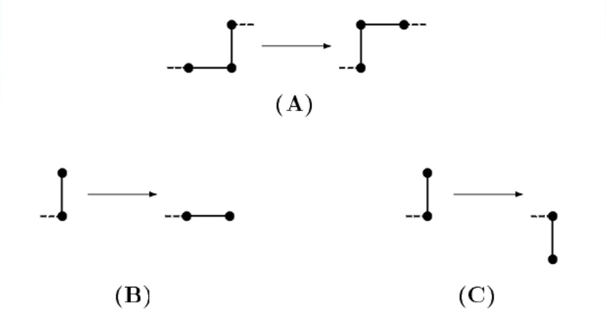
- A move is a transformation of the SAW which may result in a new SAW.
- Local moves include one bead flips and the reptation or slithering snake move.
- Non-local moves include pivots and cut-and-permute moves.
- Many popular local and non-local moves can be defined in terms of “cut-and-paste” moves.

Moves

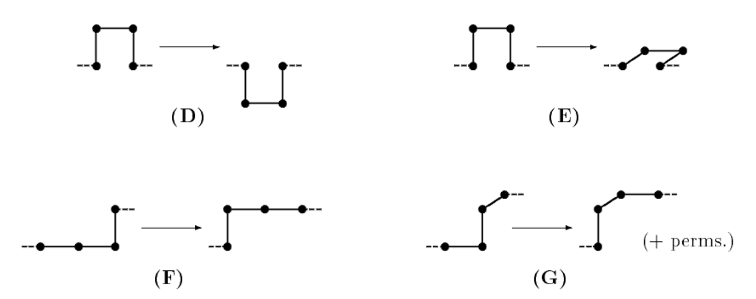
- A move is a transformation of the SAW which may result in a new SAW.
- Local moves include one bead flips and the reptation or slithering snake move.
- Non-local moves include pivots and cut-and-permute moves.
- Many popular local and non-local moves can be defined in terms of "cut-and-paste" moves.

Moves

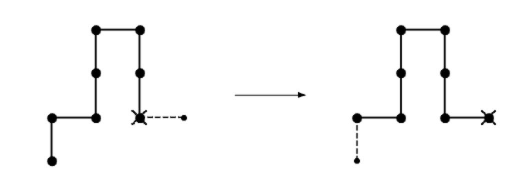
- A move is a transformation of the SAW which may result in a new SAW.
- Local moves include one bead flips and the reptation or slithering snake move.
- Non-local moves include pivots and cut-and-permute moves.
- Many popular local and non-local moves can be defined in terms of "cut-and-paste" moves.



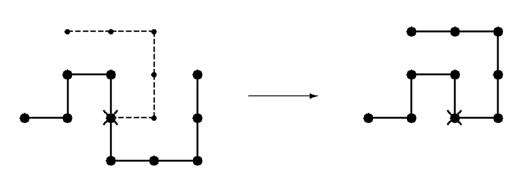
One bead moves (from Sokal, 1994 [Sok94])



Two bead moves (from Sokal, 1994 [Sok94])

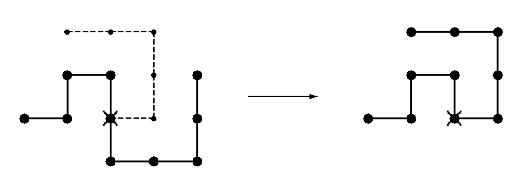


Slithering snake / reptation move (from Sokal, 1994 [Sok94])



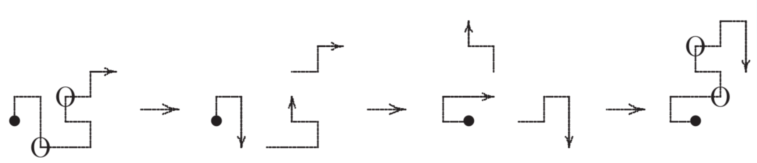
Pivot move (from Sokal, 1994 [Sok94])

Non-local move (unless close to end)



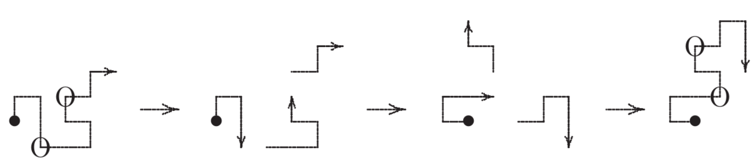
Pivot move (from Sokal, 1994 [Sok94])

Non-local move (unless close to end)



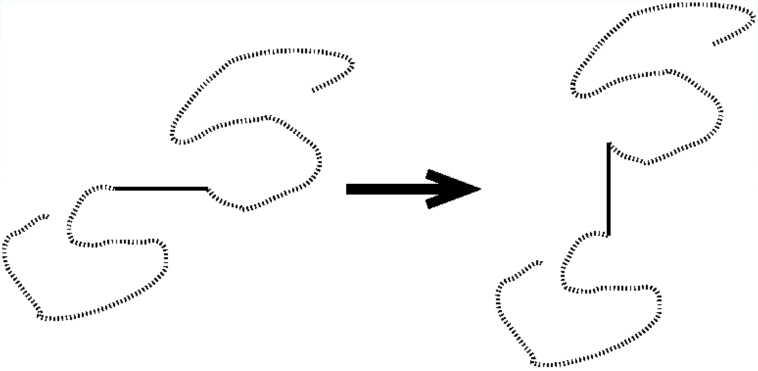
Cut-and-paste move, arbitrary symmetry, may permute sub-walks
(from Janse van Rensburg, 2009 [JvR09])

Scale of move depends on size of sub-walks – any length scale possible. Pivots are a special case of cut-and-paste moves.



Cut-and-paste move, arbitrary symmetry, may permute sub-walks (from Janse van Rensburg, 2009 [JvR09])

Scale of move depends on size of sub-walks – any length scale possible. Pivots are a special case of cut-and-paste moves.



Rotating a single bond in the middle of a SAW, $O(N)$ monomers move distance $O(1)$. Cut-and-paste moves generalise what are usually regarded as “local” moves.

Pivot algorithm

- Invented in 1969 by Lal.
- The power of the method only realised since influential paper by Madras and Sokal in 1988 (over 500 citations).
- Monte Carlo method of choice for studying SAWs and similar models when the length of the walk is fixed.
- Markov chain, pivot operations generate new SAWs. When resulting configuration is not a SAW, move is rejected.
- Will now show a sequence of *successful* pivots applied to an $n = 65536$ site SAW on the square lattice.

Pivot algorithm

- Invented in 1969 by Lal.
- The power of the method only realised since influential paper by Madras and Sokal in 1988 (over 500 citations).
- Monte Carlo method of choice for studying SAWs and similar models when the length of the walk is fixed.
- Markov chain, pivot operations generate new SAWs. When resulting configuration is not a SAW, move is rejected.
- Will now show a sequence of *successful* pivots applied to an $n = 65536$ site SAW on the square lattice.

Pivot algorithm

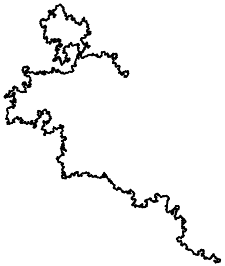
- Invented in 1969 by Lal.
- The power of the method only realised since influential paper by Madras and Sokal in 1988 (over 500 citations).
- Monte Carlo method of choice for studying SAWs and similar models when the length of the walk is fixed.
- Markov chain, pivot operations generate new SAWs. When resulting configuration is not a SAW, move is rejected.
- Will now show a sequence of *successful* pivots applied to an $n = 65536$ site SAW on the square lattice.

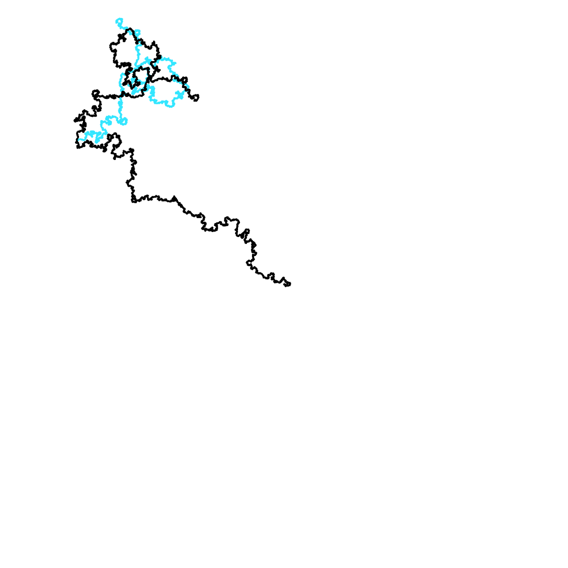
Pivot algorithm

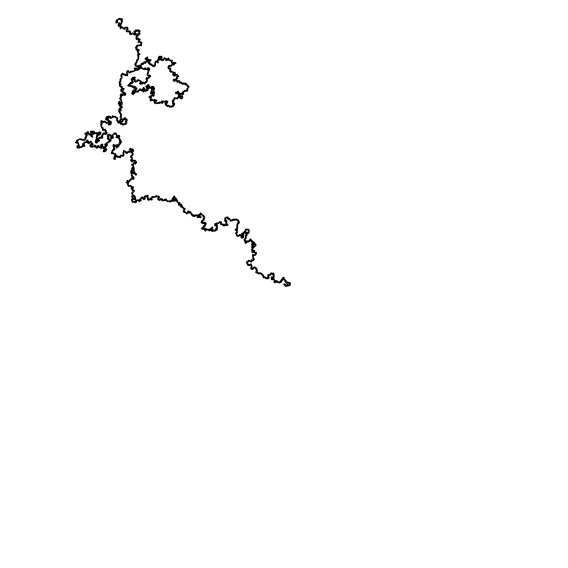
- Invented in 1969 by Lal.
- The power of the method only realised since influential paper by Madras and Sokal in 1988 (over 500 citations).
- Monte Carlo method of choice for studying SAWs and similar models when the length of the walk is fixed.
- Markov chain, pivot operations generate new SAWs. When resulting configuration is not a SAW, move is rejected.
- Will now show a sequence of *successful* pivots applied to an $n = 65536$ site SAW on the square lattice.

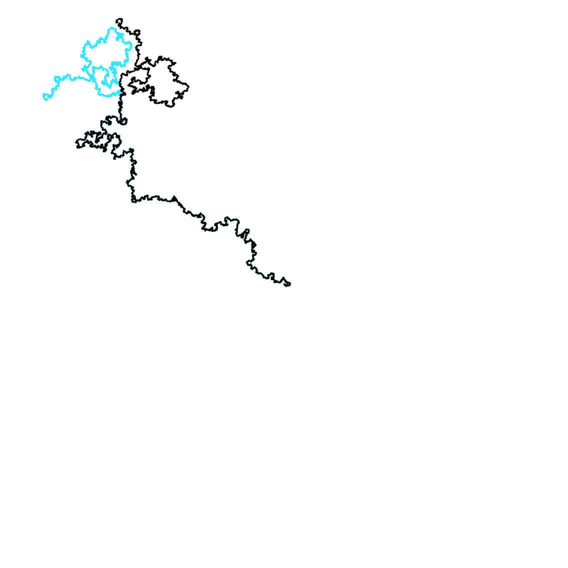
Pivot algorithm

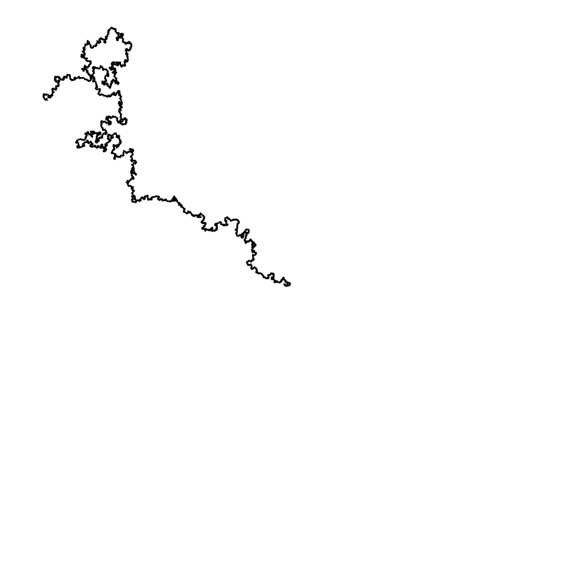
- Invented in 1969 by Lal.
- The power of the method only realised since influential paper by Madras and Sokal in 1988 (over 500 citations).
- Monte Carlo method of choice for studying SAWs and similar models when the length of the walk is fixed.
- Markov chain, pivot operations generate new SAWs. When resulting configuration is not a SAW, move is rejected.
- Will now show a sequence of *successful* pivots applied to an $n = 65536$ site SAW on the square lattice.

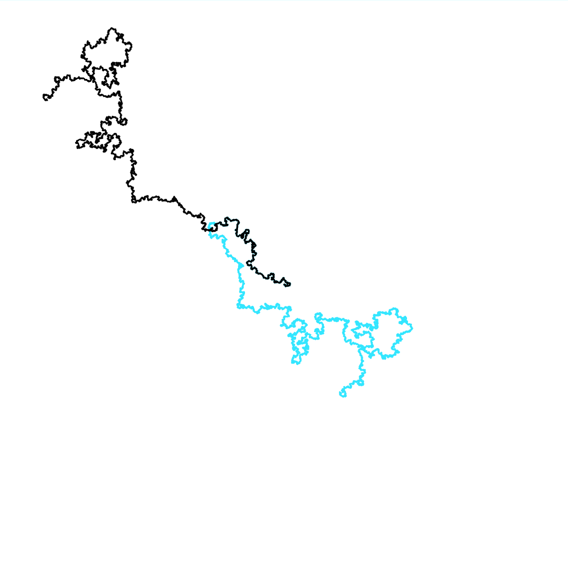


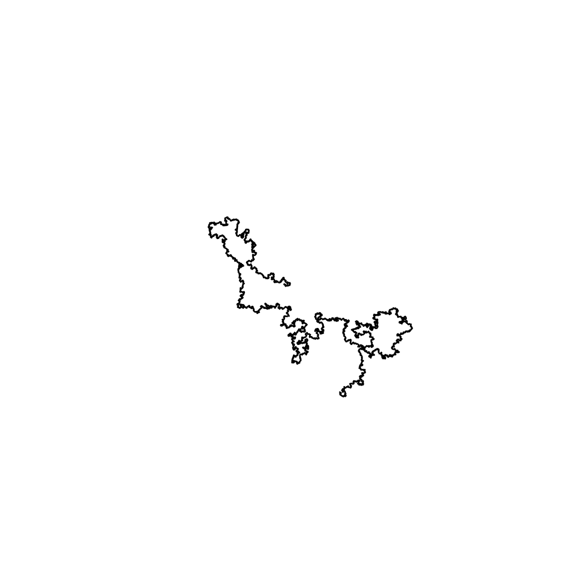


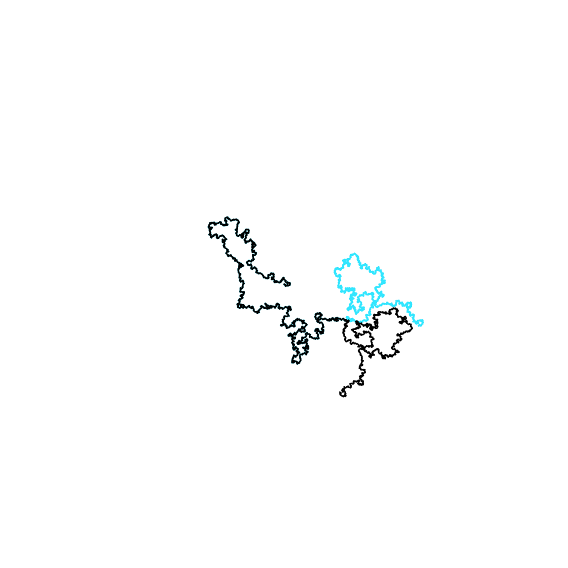












SAW-tree

- How do we implement pivot algorithm efficiently?
  - Represent SAW as a binary tree.
- Each node of tree contains global information about sub-walk, including bounding box and global observables.
- Ensures *all* cut-and-paste moves can be performed in time $O(\log N)$ via tree-rotations, as height of binary tree $O(\log N)$.
- Calculation of change of interaction energy is system dependent:
  - $O(\log N)$ for SAWs, even for global moves.
  - complicated dependence on N - should be worse than $\log N$.

SAW-tree

- How do we implement pivot algorithm efficiently?
  - Represent SAW as a binary tree.
- Each node of tree contains global information about sub-walk, including bounding box and global observables.
- Ensures *all* cut-and-paste moves can be performed in time $O(\log N)$ via tree-rotations, as height of binary tree $O(\log N)$.
- Calculation of change of interaction energy is system dependent:
  - $O(\log N)$ for SAWs, even for global moves.
  - [illegible] complicated dependence on N - should be worse than $\log N$.

SAW-tree

- How do we implement pivot algorithm efficiently?
  - Represent SAW as a binary tree.
- Each node of tree contains global information about sub-walk, including bounding box and global observables.
- Ensures *all* cut-and-paste moves can be performed in time $O(\log N)$ via tree-rotations, as height of binary tree $O(\log N)$.
- Calculation of change of interaction energy is system dependent:
  - $O(\log N)$ for SAWs, even for global moves
  - [illegible] complicated dependence on N - should be worse than $\log N$.

SAW-tree

- How do we implement pivot algorithm efficiently?
  - Represent SAW as a binary tree.
- Each node of tree contains global information about sub-walk, including bounding box and global observables.
- Ensures *all* cut-and-paste moves can be performed in time $O(\log N)$ via tree-rotations, as height of binary tree $O(\log N)$.
- Calculation of change of interaction energy is system dependent:
  - $O(\log N)$ for SAWs, even for global moves
  - complicated dependence on N - should be worse than $\log N$.

SAW-tree

- How do we implement pivot algorithm efficiently?
  - Represent SAW as a binary tree.
- Each node of tree contains global information about sub-walk, including bounding box and global observables.
- Ensures *all* cut-and-paste moves can be performed in time $O(\log N)$ via tree-rotations, as height of binary tree $O(\log N)$.
- Calculation of change of interaction energy is system dependent:
  - $O(\log N)$ for SAWs, even for global moves
  - unknown for polymers in θ or collapsed phases; may have complicated dependence on N - should be worse than $\log N$.

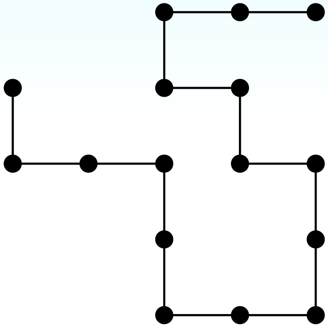
SAW-tree

- How do we implement pivot algorithm efficiently?
  - Represent SAW as a binary tree.
- Each node of tree contains global information about sub-walk, including bounding box and global observables.
- Ensures *all* cut-and-paste moves can be performed in time $O(\log N)$ via tree-rotations, as height of binary tree $O(\log N)$.
- Calculation of change of interaction energy is system dependent:
  - $O(\log N)$ for SAWs, even for global moves
  - unknown for polymers in θ or collapsed phases; may have complicated dependence on N - should be worse than $\log N$.

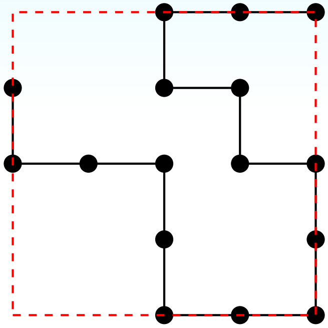
SAW-tree

- How do we implement pivot algorithm efficiently?
  - Represent SAW as a binary tree.
- Each node of tree contains global information about sub-walk, including bounding box and global observables.
- Ensures *all* cut-and-paste moves can be performed in time $O(\log N)$ via tree-rotations, as height of binary tree $O(\log N)$.
- Calculation of change of interaction energy is system dependent:
  - $O(\log N)$ for SAWs, even for global moves
  - unknown for polymers in θ or collapsed phases; may have complicated dependence on N - should be worse than $\log N$.

SAW-tree

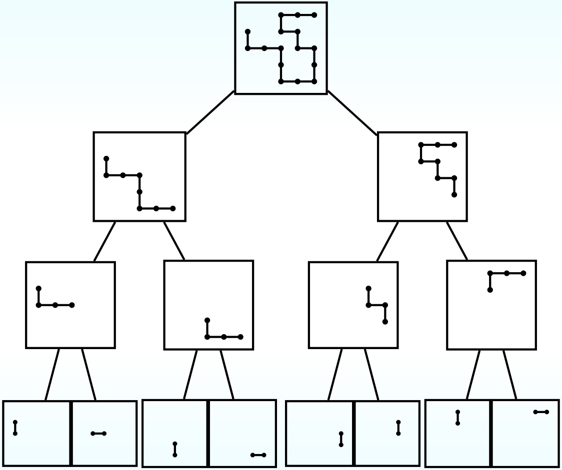


SAW-tree

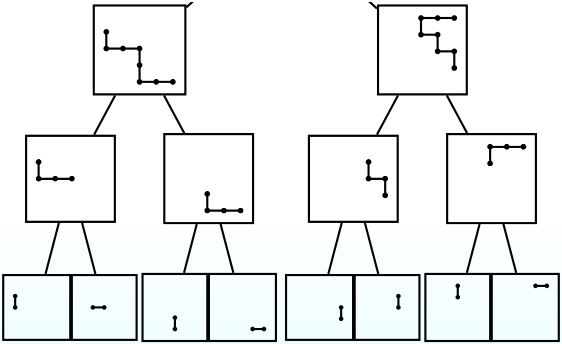


Bounding box

SAW-tree



SAW-tree



Pivot algorithm

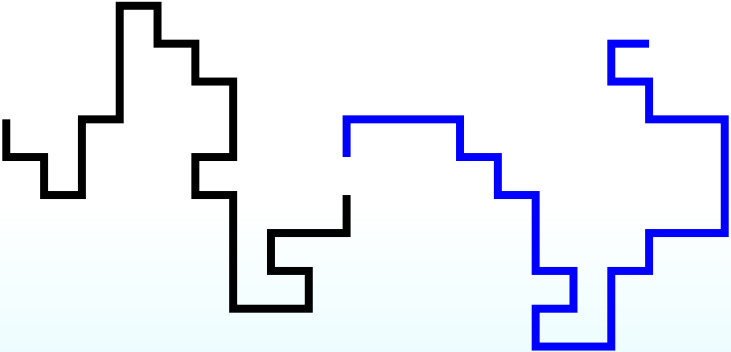
- After applying pivot to a SAW with 64 sites, will show algorithm to determine whether new configuration is self-avoiding.
- Can be easily adapted to ISAW and related models.
- Algorithm uses “depth-first search” in an attempt to find intersections, recursively applying the observation that when the bounding box of two sub-walks do not intersect, then the sub-walks themselves cannot intersect.

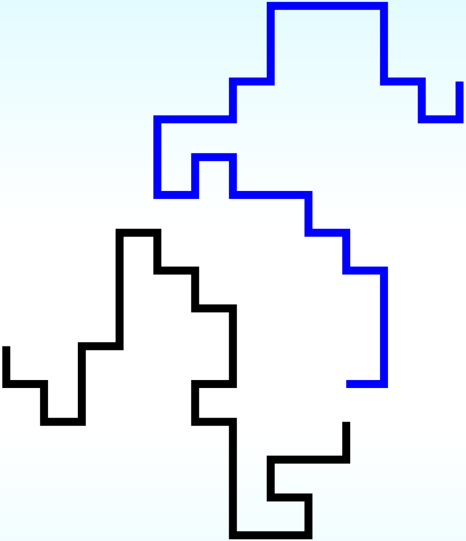
Pivot algorithm

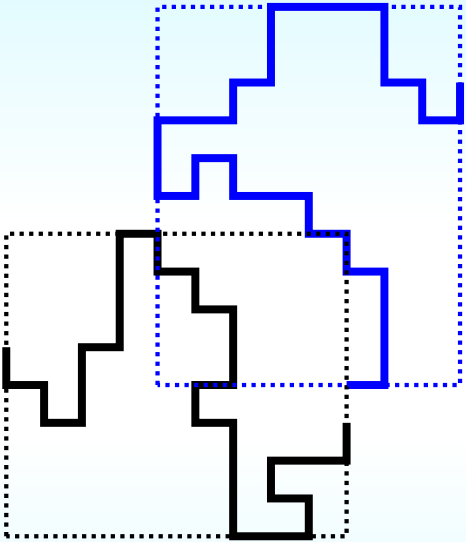
- After applying pivot to a SAW with 64 sites, will show algorithm to determine whether new configuration is self-avoiding.
- Can be easily adapted to ISAW and related models.
- Algorithm uses "depth-first search" in an attempt to find intersections, recursively applying the observation that when the bounding box of two sub-walks do not intersect, then the sub-walks themselves cannot intersect.

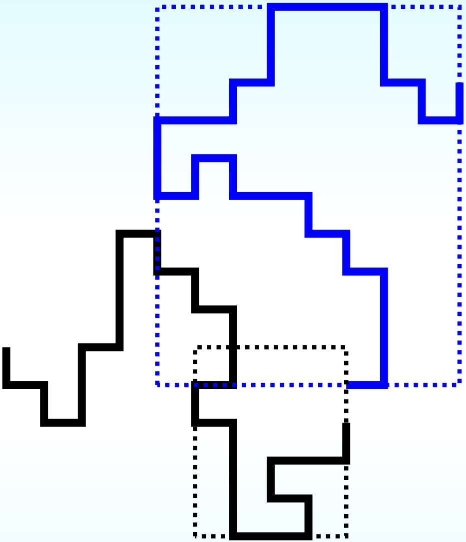
Pivot algorithm

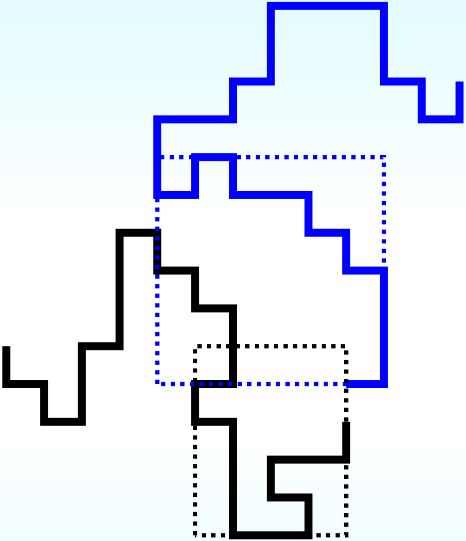
- After applying pivot to a SAW with 64 sites, will show algorithm to determine whether new configuration is self-avoiding.
- Can be easily adapted to ISAW and related models.
- Algorithm uses "depth-first search" in an attempt to find intersections, recursively applying the observation that when the bounding box of two sub-walks do not intersect, then the sub-walks themselves cannot intersect.

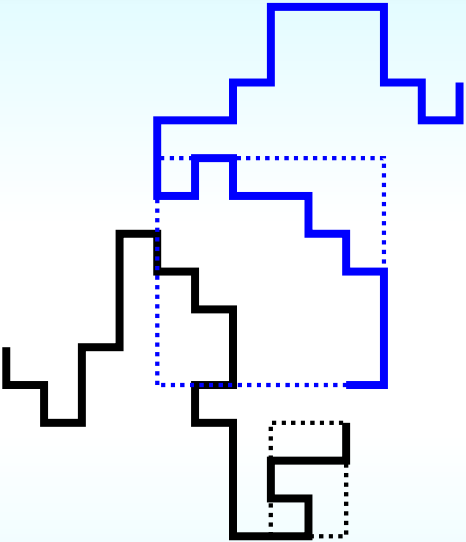


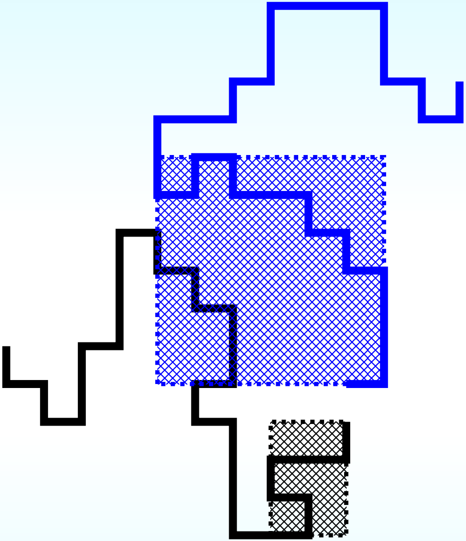


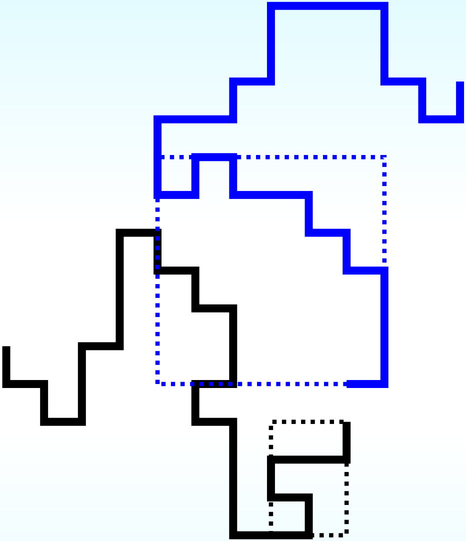


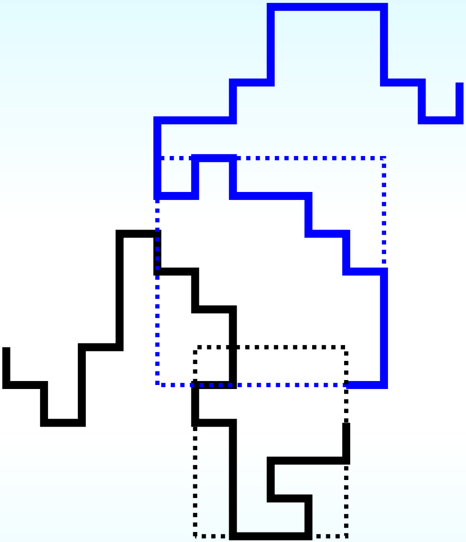


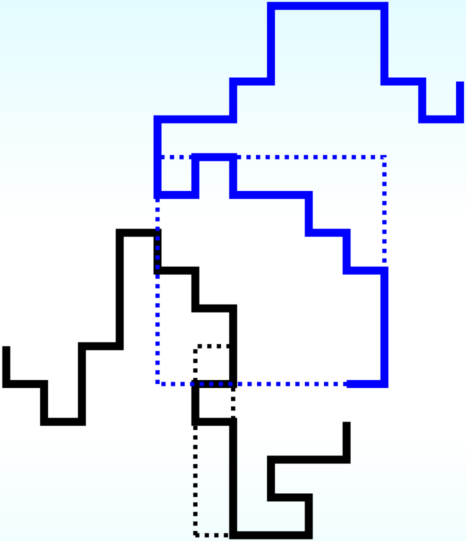


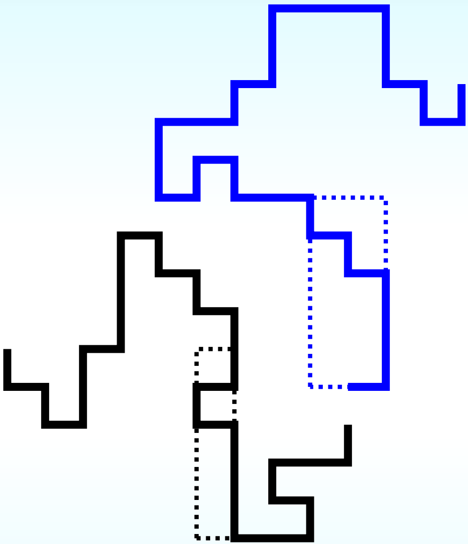


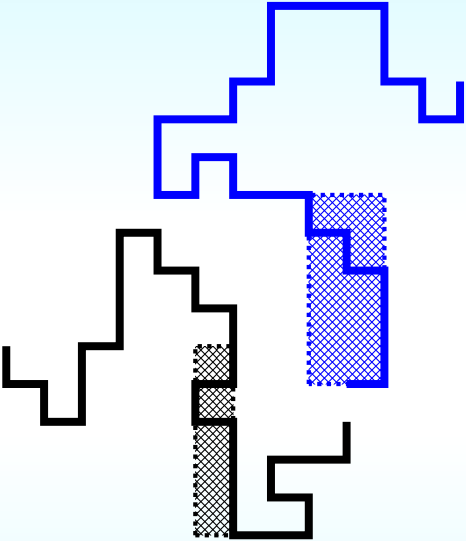


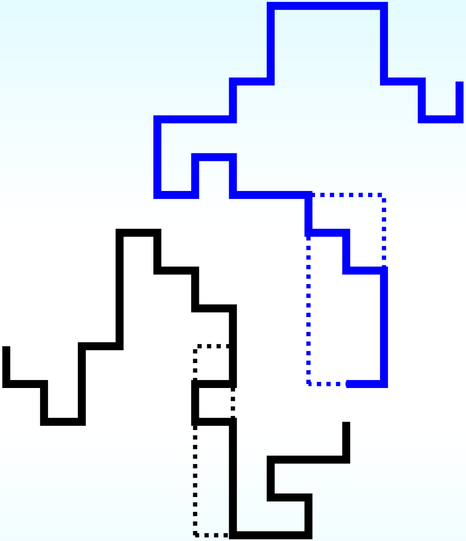


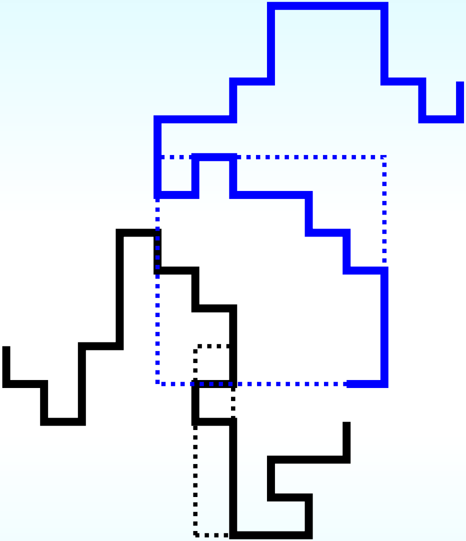


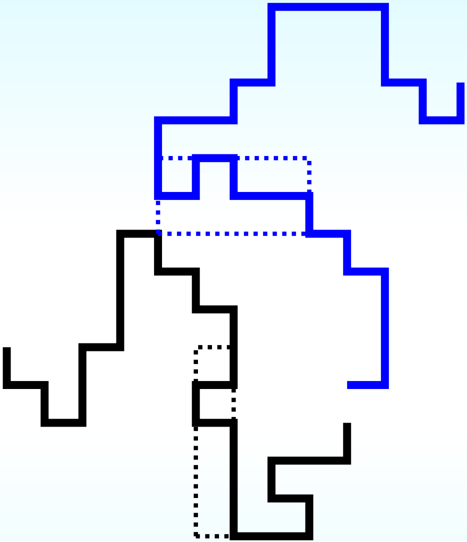


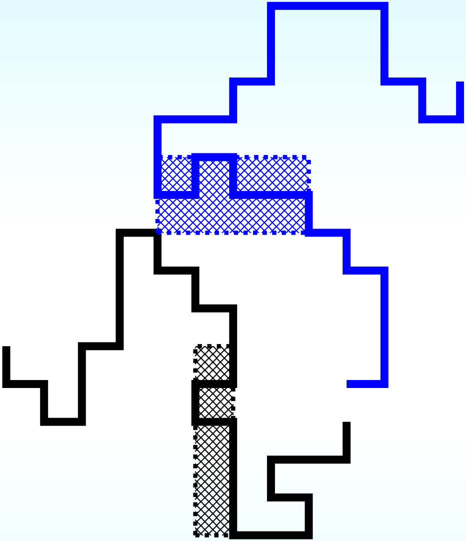


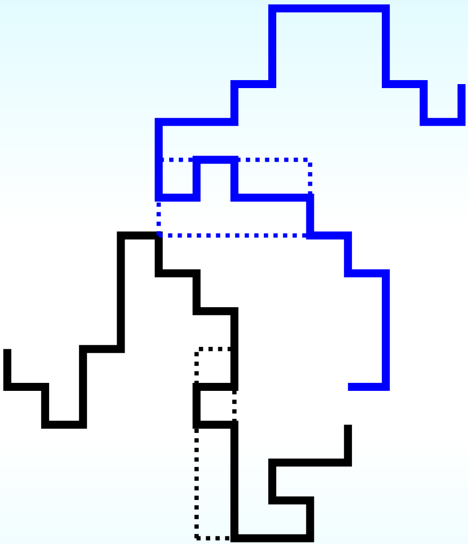


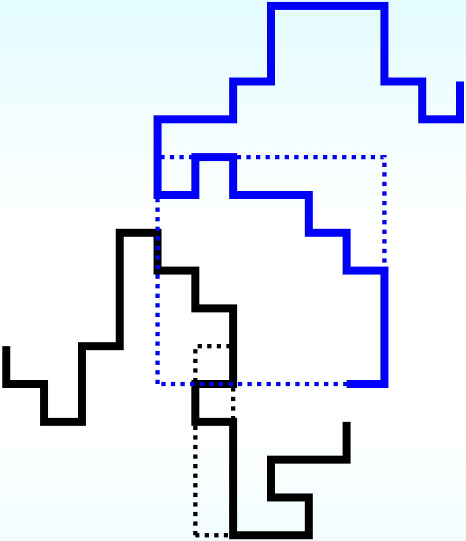


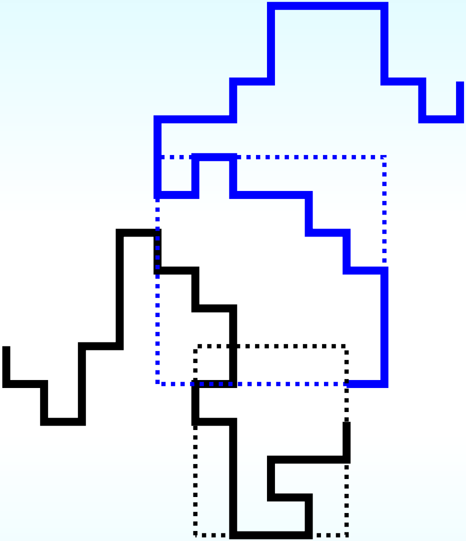


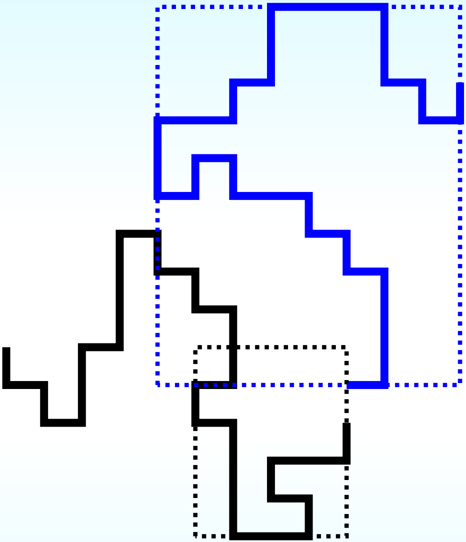


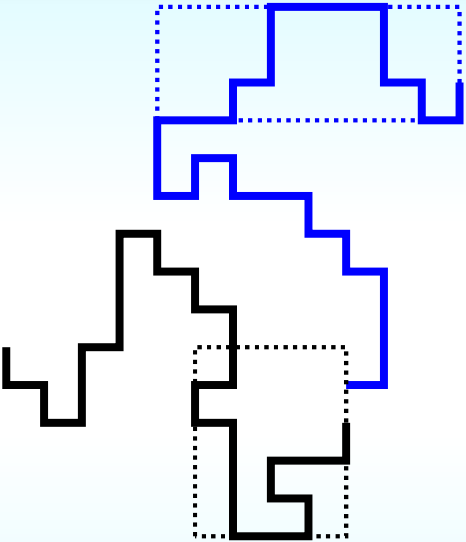


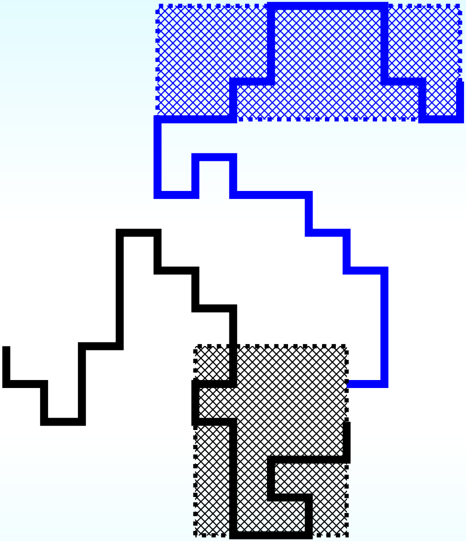


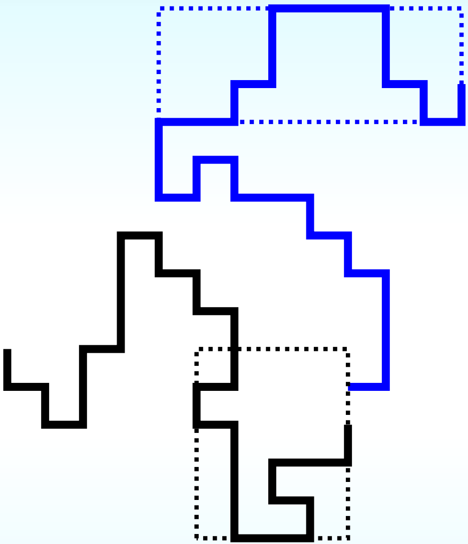


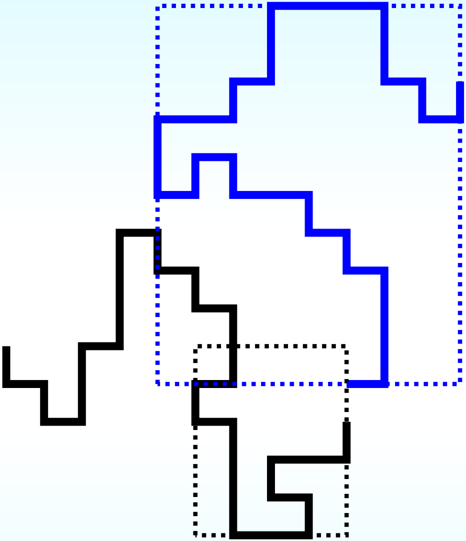


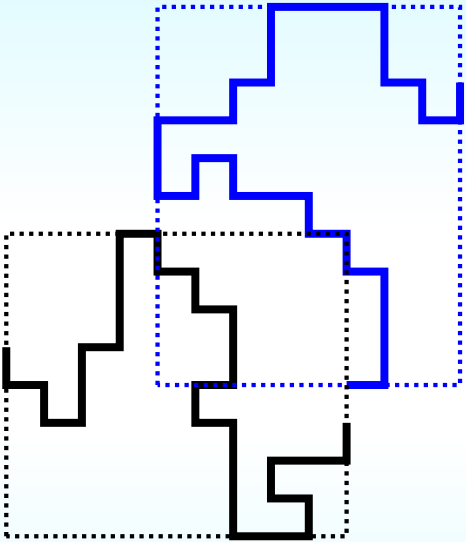


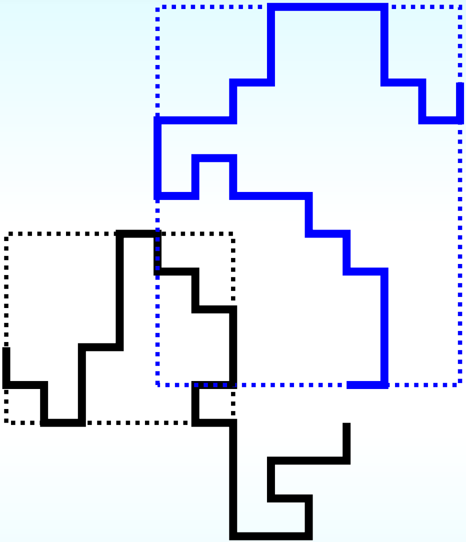


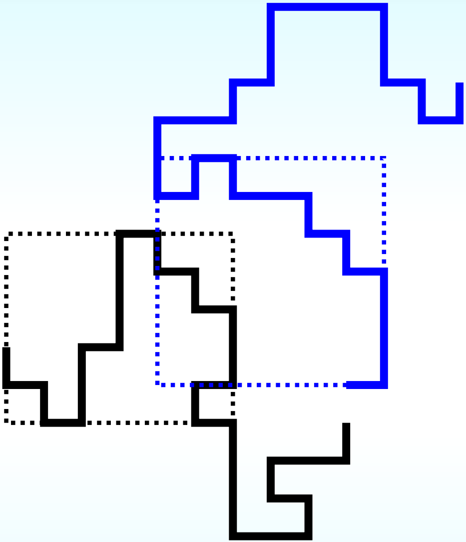


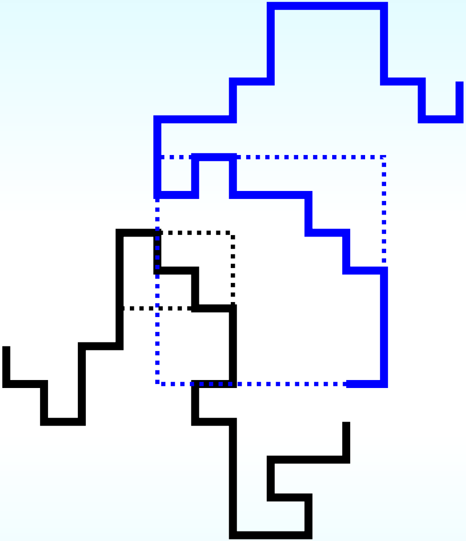


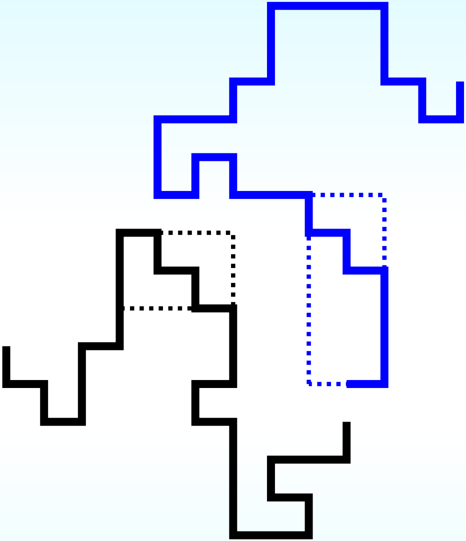


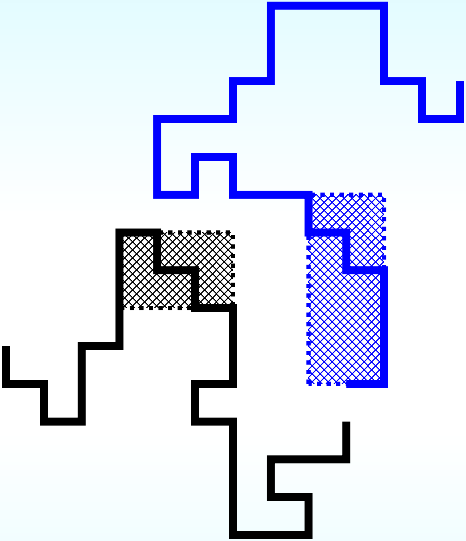


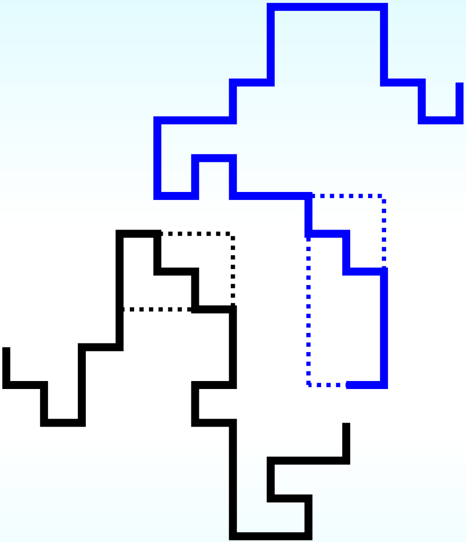


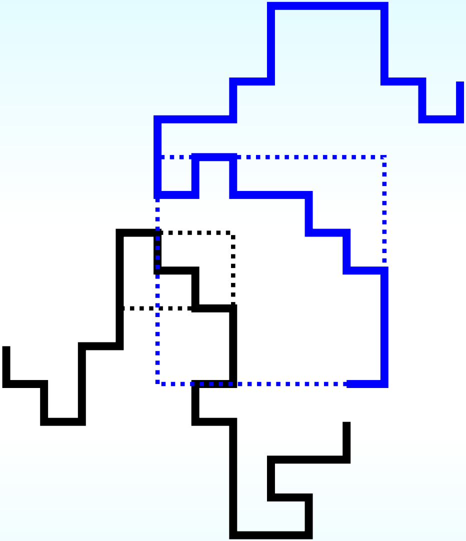


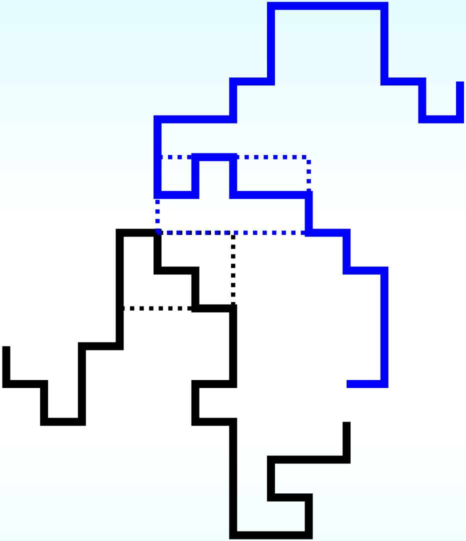


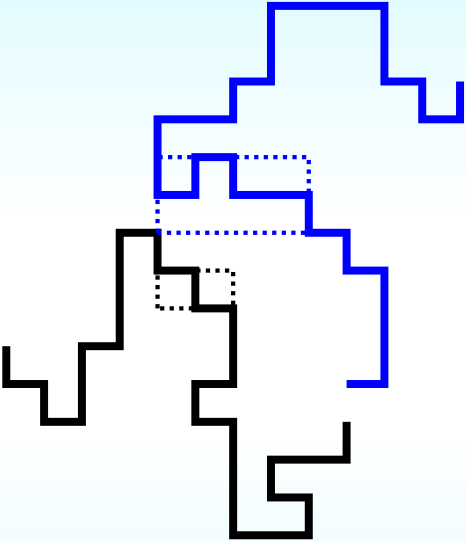


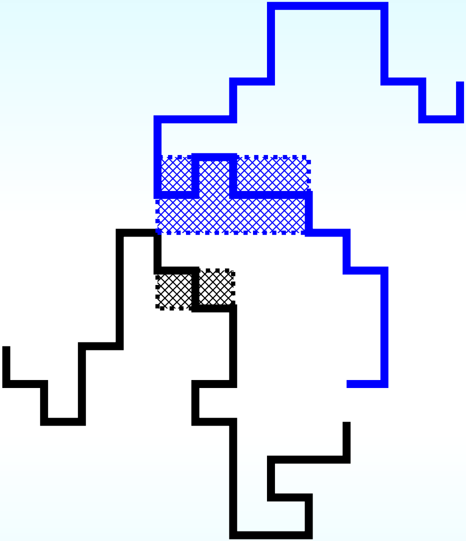


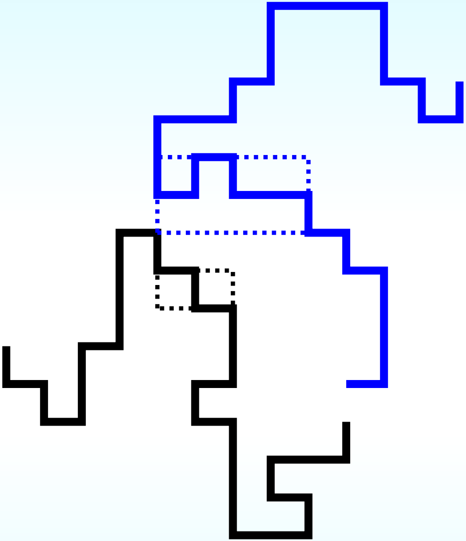


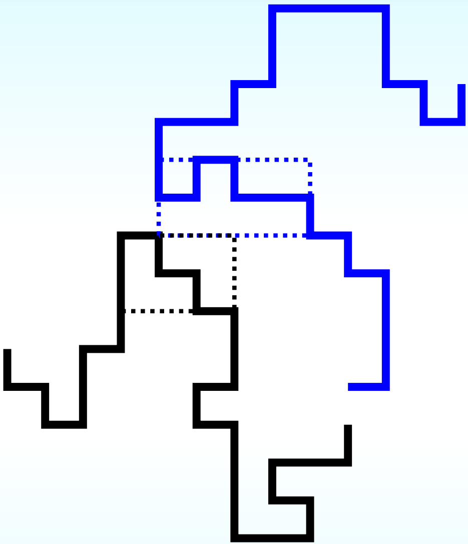


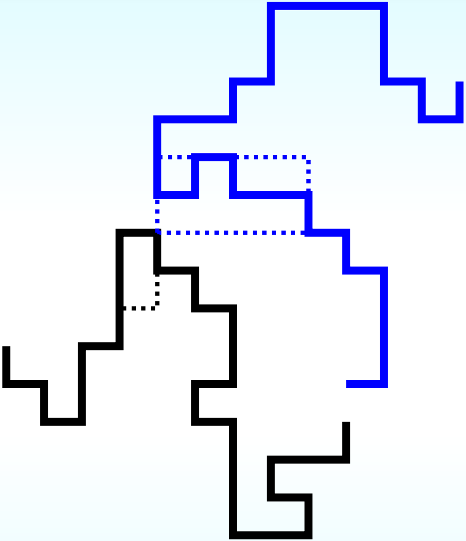


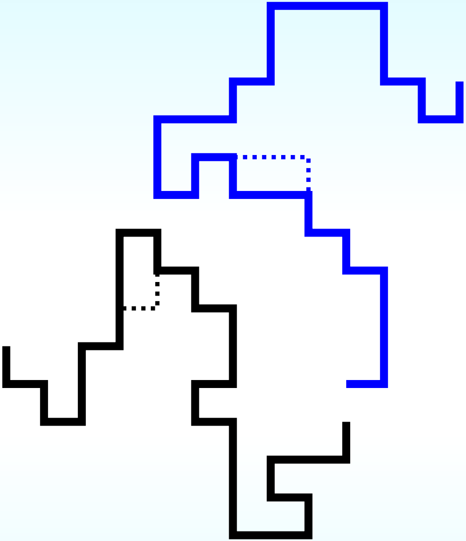


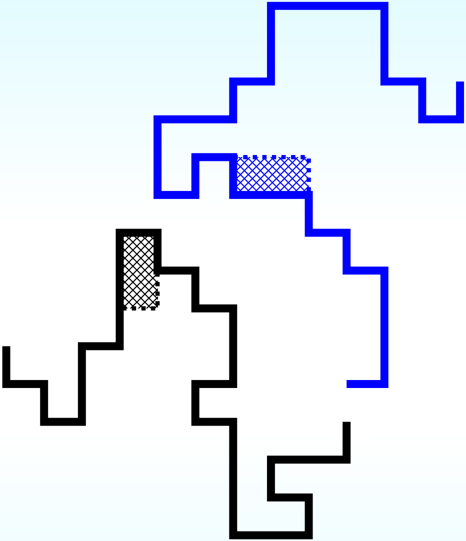


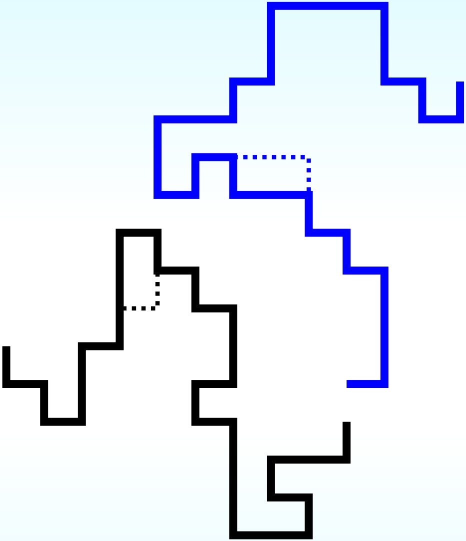


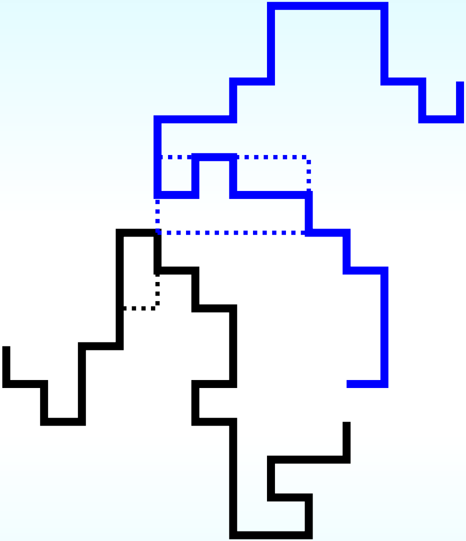


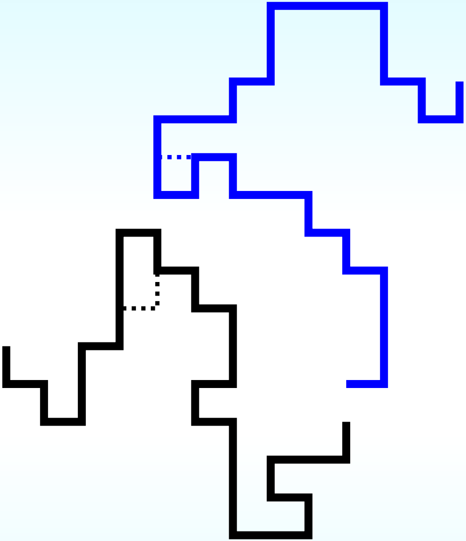


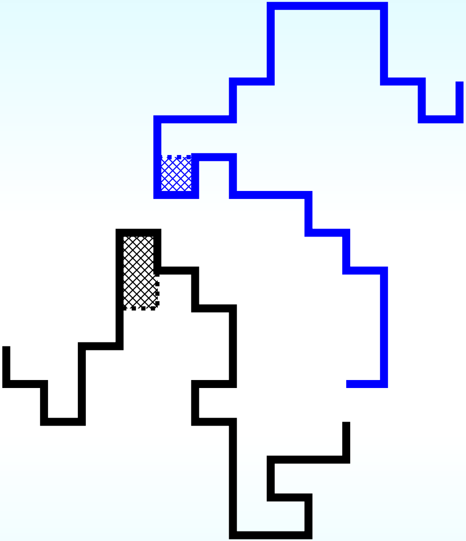


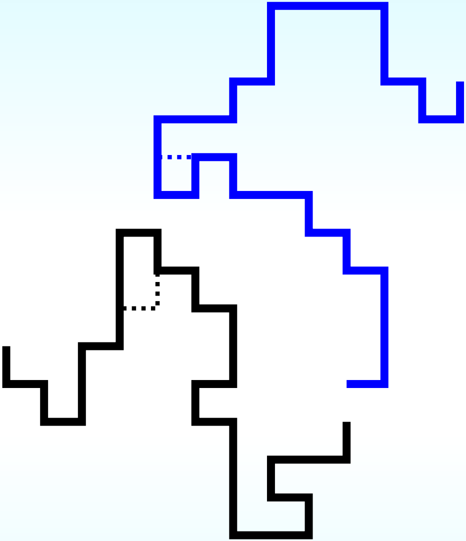


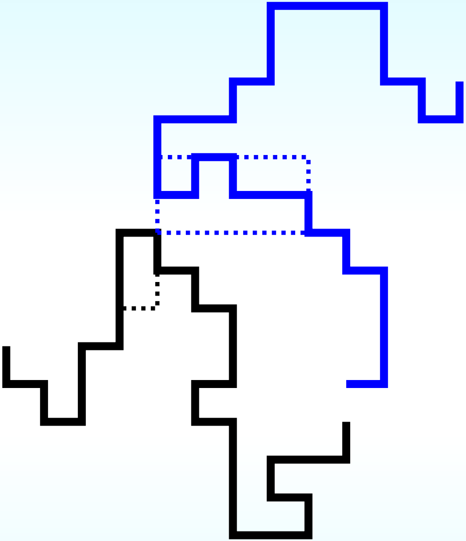


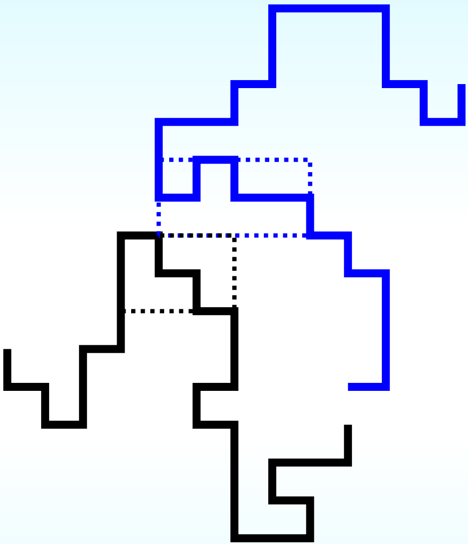


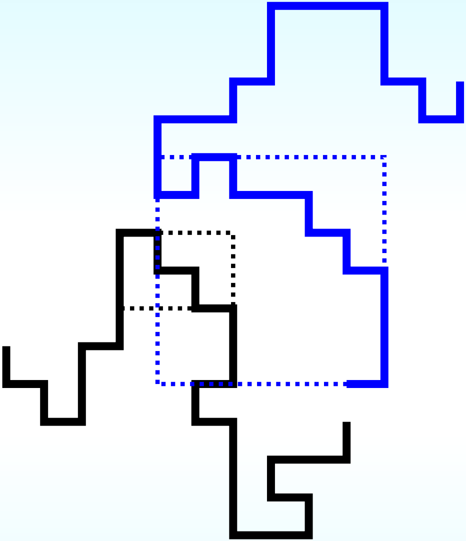


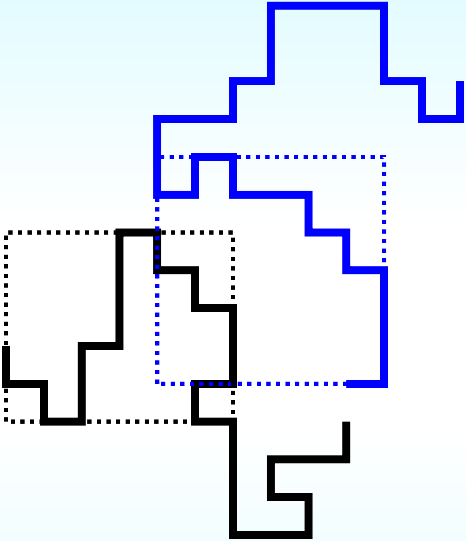


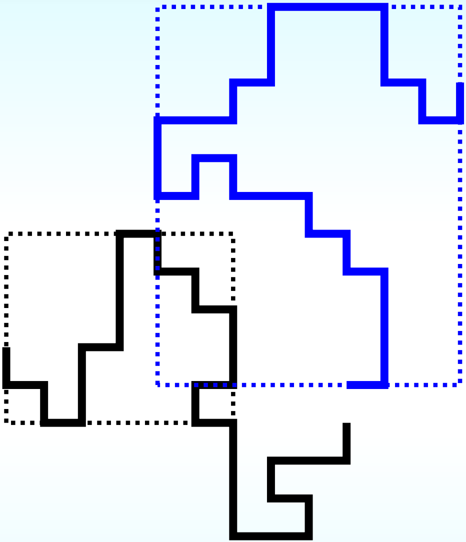


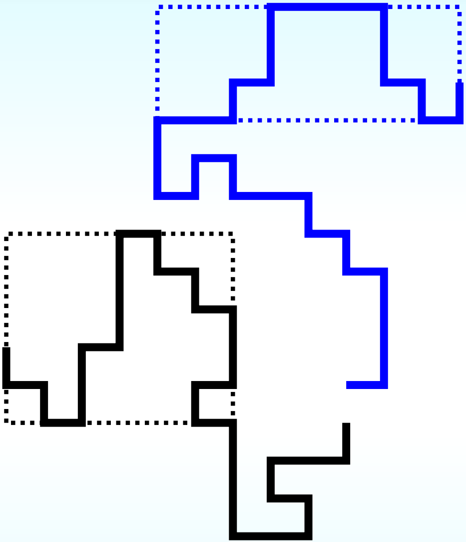


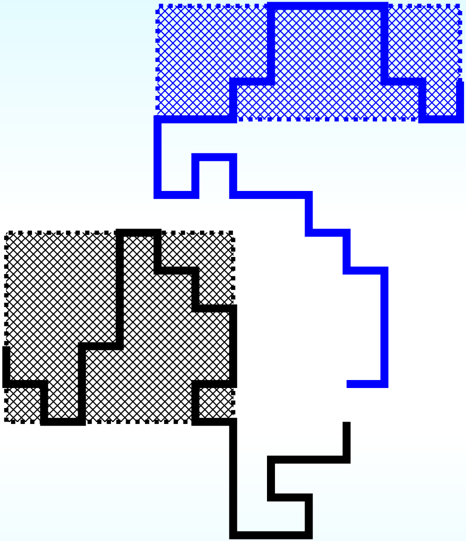


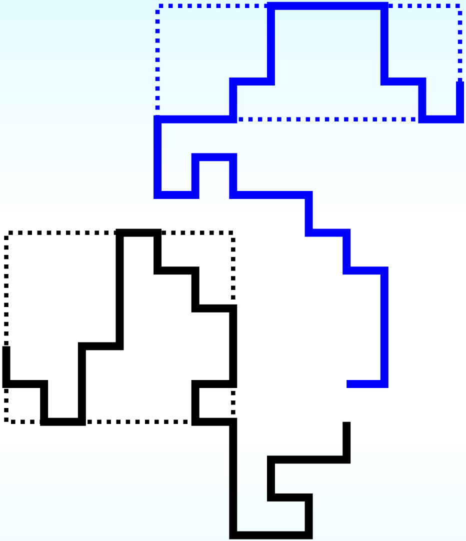


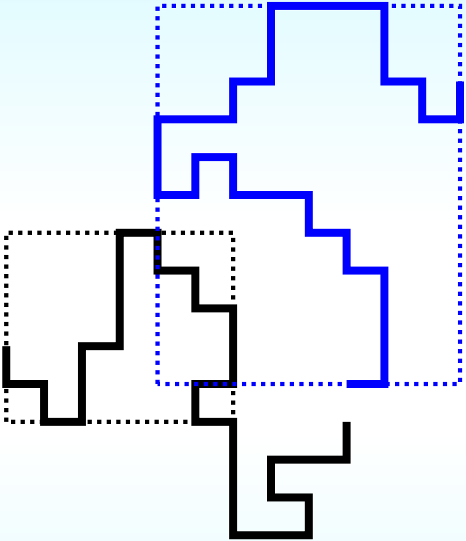


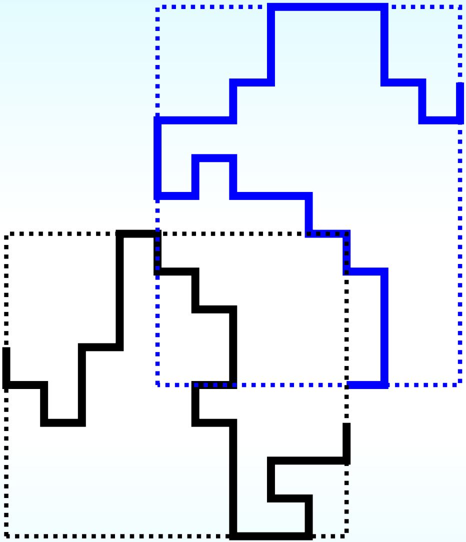


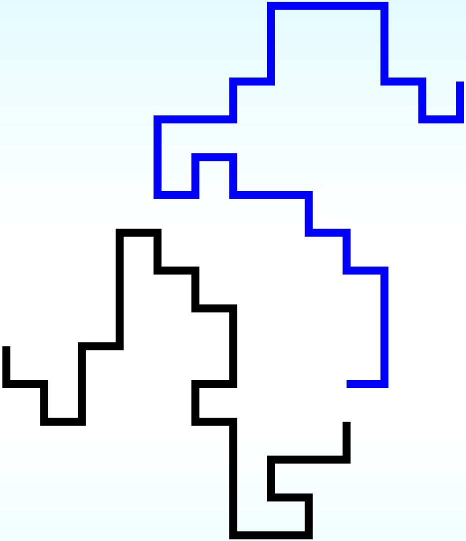












Implementing cut-and-paste moves via the SAW-tree:

- Split polymer into pieces, $O(\log N)$.
- Apply move(s) to sub-walk(s), $O(1)$.
- Calculate change in interaction energy between sub-walks. For SAWs, $O(\log N)$.
- Accept / reject move.
- Merge chains, $O(\log N)$.

Implementing cut-and-paste moves via the SAW-tree:

- Split polymer into pieces, $O(\log N)$.
- Apply move(s) to sub-walk(s), $O(1)$.
- Calculate change in interaction energy between sub-walks. For SAWs, $O(\log N)$.
- Accept / reject move.
- Merge chains, $O(\log N)$.

Implementing cut-and-paste moves via the SAW-tree:

- Split polymer into pieces, $O(\log N)$.
- Apply move(s) to sub-walk(s), $O(1)$.
- Calculate change in interaction energy between sub-walks. For SAWs, $O(\log N)$.
- Accept / reject move.
- Merge chains, $O(\log N)$.

Implementing cut-and-paste moves via the SAW-tree:

- Split polymer into pieces, $O(\log N)$.
- Apply move(s) to sub-walk(s), $O(1)$.
- Calculate change in interaction energy between sub-walks. For SAWs, $O(\log N)$.
- Accept / reject move.
- Merge chains, $O(\log N)$.

Implementing cut-and-paste moves via the SAW-tree:

- Split polymer into pieces, $O(\log N)$.
- Apply move(s) to sub-walk(s), $O(1)$.
- Calculate change in interaction energy between sub-walks. For SAWs, $O(\log N)$.
- Accept / reject move.
- Merge chains, $O(\log N)$.

- Madras and Sokal (1988): implementation CPU time of approximately $O(N^{0.89})$ per attempted pivot for $\mathbb{Z}^3$.
- Kennedy (2002): implementation which is approximately $O(N^{0.74})$.
- SAW-tree implementation of the pivot algorithm, average case $O(\log N)$ for N-step SAWs [Cli10a]. Enables simulation of much longer SAWs, so far up to 265 million steps.
- Estimated $\nu = 0.587597(7)$ [Cli10b].
- Previous estimates are 0.5874(2) (MC, Prellberg, 2001) and 0.58756(5) (MCRG, Belohorec, 1997).

- Madras and Sokal (1988): implementation CPU time of approximately $O(N^{0.89})$ per attempted pivot for $\mathbb{Z}^3$.
- Kennedy (2002): implementation which is approximately $O(N^{0.74})$.
- SAW-tree implementation of the pivot algorithm, average case $O(\log N)$ for N-step SAWs [Cli10a]. Enables simulation of much longer SAWs, so far up to 265 million steps.
- Estimated $\nu = 0.587597(7)$ [Cli10b].
- Previous estimates are 0.5874(2) (MC, Prellberg, 2001) and 0.58756(5) (MCRG, Belohorec, 1997).

- Madras and Sokal (1988): implementation CPU time of approximately $O(N^{0.89})$ per attempted pivot for $\mathbb{Z}^3$.
- Kennedy (2002): implementation which is approximately $O(N^{0.74})$.
- SAW-tree implementation of the pivot algorithm, average case $O(\log N)$ for N-step SAWs [Cli10a]. Enables simulation of much longer SAWs, so far up to 265 million steps.
- Estimated $\nu = 0.587597(7)$ [Cli10b].
- Previous estimates are 0.5874(2) (MC, Prellberg, 2001) and 0.58756(5) (MCRG, Belohorec, 1997).

- Madras and Sokal (1988): implementation CPU time of approximately $O(N^{0.89})$ per attempted pivot for $\mathbb{Z}^3$.
- Kennedy (2002): implementation which is approximately $O(N^{0.74})$.
- SAW-tree implementation of the pivot algorithm, average case $O(\log N)$ for N-step SAWs [Cli10a]. Enables simulation of much longer SAWs, so far up to 265 million steps.
- Estimated $\nu = 0.587597(7)$ [Cli10b].
- Previous estimates are 0.5874(2) (MC, Prellberg, 2001) and 0.58756(5) (MCRG, Belohorec, 1997).

- Madras and Sokal (1988): implementation CPU time of approximately $O(N^{0.89})$ per attempted pivot for $\mathbb{Z}^3$.
- Kennedy (2002): implementation which is approximately $O(N^{0.74})$.
- SAW-tree implementation of the pivot algorithm, average case $O(\log N)$ for N-step SAWs [Cli10a]. Enables simulation of much longer SAWs, so far up to 265 million steps.
- Estimated $\nu = 0.587597(7)$ [Cli10b].
- Previous estimates are 0.5874(2) (MC, Prellberg, 2001) and 0.58756(5) (MCRG, Belohorec, 1997).

CPU time per attempted pivot, for SAWs of length N:

	$\mathbb{Z}^2$			$\mathbb{Z}^3$		
N	S-t (μs)	M&S/S-t	K/S-t	S-t (μs)	M&S/S-t	K/S-t
31	0.41	0.894	1.06	0.59	0.981	1.37
1023	0.87	5.15	1.90	1.71	6.31	3.75
32767	1.27	68.6	4.92	3.36	79.2	21.5
1048575	2.91	2510	32.2	7.53	3830	385
33554431	4.57	35200	134	12.58	61700	7130

Dimerization

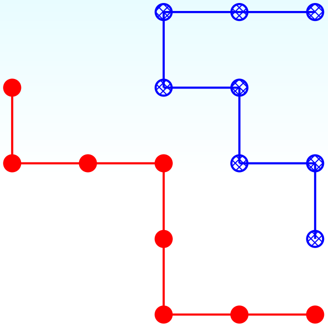
- Most straightforward method to calculate γ: dimerization, i.e. concatenating two SAWs to see if they form a longer SAW.
- Indicator function for successful concatenation is our observable, and

$$B(\omega_1, \omega_2) = \begin{cases} 0 & \text{if } \omega_1 \circ \omega_2 \text{ not self-avoiding} \\ 1 & \text{if } \omega_1 \circ \omega_2 \text{ self-avoiding} \end{cases}$$

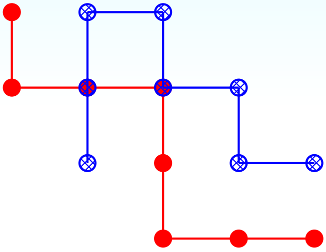
Dimerization

- Most straightforward method to calculate γ: dimerization, i.e. concatenating two SAWs to see if they form a longer SAW.
- Indicator function for successful concatenation is our observable, and

$$B(\omega_1, \omega_2) = \begin{cases} 0 & \text{if } \omega_1 \circ \omega_2 \text{ not self-avoiding} \\ 1 & \text{if } \omega_1 \circ \omega_2 \text{ self-avoiding} \end{cases}$$



$B(\omega_1, \omega_2) = 1$



$B(\omega_1, \omega_2) = 0$

Dimerization

- $$\langle B \rangle = \frac{\text{Number of } 2N-1 \text{ step SAWs}}{\text{Number of pairs of } N-1 \text{ step SAWs}}$$
  $$= \frac{c_{2N-1}}{c_{N-1}^2}$$
  $$\sim \frac{A\mu^{2N-1}(2N-1)^{\gamma-1}}{A^2\mu^{2N-2}(N-1)^{2\gamma-2}}$$
  $$\sim \frac{2^{\gamma-1}\mu}{A} N^{1-\gamma}\,[1 + \text{corrections}]$$
- Generate two Markov chains of $N-1$ step SAWs using pivot algorithm.
- Sample $B(\omega_1, \omega_2)$ for every time step.

Dimerization

- $$\begin{aligned}\langle B \rangle &= \frac{\text{Number of } 2N-1 \text{ step SAWs}}{\text{Number of pairs of } N-1 \text{ step SAWs}} \\ &= \frac{c_{2N-1}}{c_{N-1}^2} \\ &\sim \frac{A\mu^{2N-1}(2N-1)^{\gamma-1}}{A^2\mu^{2N-2}(N-1)^{2\gamma-2}} \\ &\sim \frac{2^{\gamma-1}\mu}{A} N^{1-\gamma}\,[1 + \text{corrections}]\end{aligned}$$
- Generate two Markov chains of $N-1$ step SAWs using pivot algorithm.
- Sample $B(\omega_1, \omega_2)$ for every time step.

Dimerization

- $$\begin{aligned}\langle B\rangle &= \frac{\text{Number of } 2N-1 \text{ step SAWs}}{\text{Number of pairs of } N-1 \text{ step SAWs}} \\ &= \frac{c_{2N-1}}{c_{N-1}^2} \\ &\sim \frac{A\mu^{2N-1}(2N-1)^{\gamma-1}}{A^2\mu^{2N-2}(N-1)^{2\gamma-2}} \\ &\sim \frac{2^{\gamma-1}\mu}{A} N^{1-\gamma}\left[1+\mathrm{corrections}\right]\end{aligned}$$
- Generate two Markov chains of $N-1$ step SAWs using pivot algorithm.
- Sample $B(\omega_1, \omega_2)$ for every time step.

- How many pivots must be completed before two walks are essentially new configurations with respect to observable B?
- Shape of walks close to the joint clearly important.
- Simple argument suggests mean distance from joint where first intersection occurs is $O(N^{2-\gamma})$.
- For uniform sampling, probability that pivot site is before $O(N^{2-\gamma})$ is $O(N^{1-\gamma})$. Is $\tau_{\mathrm{int}} = O(N^{\gamma-1})$?
- No. Require full probability distribution, not mean, to determine τ_{int}; in fact $\tau_{\mathrm{int}} = \Omega(N)$ for uniform sampling.

- How many pivots must be completed before two walks are essentially new configurations with respect to observable B?
- Shape of walks close to the joint clearly important.
- Simple argument suggests mean distance from joint where first intersection occurs is $O(N^{2-\gamma})$.
- For uniform sampling, probability that pivot site is before $O(N^{2-\gamma})$ is $O(N^{1-\gamma})$. Is $\tau_{\mathrm{int}} = O(N^{\gamma-1})$?
- No. Require full probability distribution, not mean, to determine τ_{int}; in fact $\tau_{\mathrm{int}} = \Omega(N)$ for uniform sampling.

- How many pivots must be completed before two walks are essentially new configurations with respect to observable B?
- Shape of walks close to the joint clearly important.
- Simple argument suggests mean distance from joint where first intersection occurs is $O(N^{2-\gamma})$.
- For uniform sampling, probability that pivot site is before $O(N^{2-\gamma})$ is $O(N^{1-\gamma})$. Is $\tau_{\mathrm{int}} = O(N^{\gamma-1})$?
- No. Require full probability distribution, not mean, to determine τ_{int}; in fact $\tau_{\mathrm{int}} = \Omega(N)$ for uniform sampling.

- How many pivots must be completed before two walks are essentially new configurations with respect to observable B?
- Shape of walks close to the joint clearly important.
- Simple argument suggests mean distance from joint where first intersection occurs is $O(N^{2-\gamma})$.
- For uniform sampling, probability that pivot site is before $O(N^{2-\gamma})$ is $O(N^{1-\gamma})$. Is $\tau_{\mathrm{int}} = O(N^{\gamma-1})$?
- No. Require full probability distribution, not mean, to determine τ_{int}; in fact $\tau_{\mathrm{int}} = \Omega(N)$ for uniform sampling.

- How many pivots must be completed before two walks are essentially new configurations with respect to observable B?
- Shape of walks close to the joint clearly important.
- Simple argument suggests mean distance from joint where first intersection occurs is $O(N^{2-\gamma})$.
- For uniform sampling, probability that pivot site is before $O(N^{2-\gamma})$ is $O(N^{1-\gamma})$. Is $\tau_{\text{int}} = O(N^{\gamma-1})$?
- No. Require full probability distribution, not mean, to determine τ_{int}; in fact $\tau_{\text{int}} = \Omega(N)$ for uniform sampling.

- Choose pivot sites preferentially close to the joint.
- Natural to choose using a power law.
- Robust choice: $\Pr(d) \propto \frac{1}{d}$, d is distance from joint.
- Sites chosen at all length scales with equal probability. Probability that $i \in [L, 2L]$ is $O(1/\log N)$; "scale-free" pivot moves.
- Lower bound for τ_{int}: the time necessary to achieve a pivot before the first intersection.
- Plausible upper bound for τ_{int}: time necessary for successful pivots to be achieved at all possible length scales, i.e. $O(\log N)$ successful pivots.
- $\tau_{\mathrm{int}} = O(N^p \log N)$.

- Choose pivot sites preferentially close to the joint.
- Natural to choose using a power law.
- Robust choice: $\Pr(d) \propto \frac{1}{d}$, d is distance from joint.
- Sites chosen at all length scales with equal probability. Probability that $i \in [L, 2L]$ is $O(1/\log N)$; "scale-free" pivot moves.
- Lower bound for τ_{int}: the time necessary to achieve a pivot before the first intersection.
- Plausible upper bound for τ_{int}: time necessary for successful pivots to be achieved at all possible length scales, i.e. $O(\log N)$ successful pivots.
- $\tau_{\text{int}} = O(N^p \log N)$.

- Choose pivot sites preferentially close to the joint.
- Natural to choose using a power law.
- Robust choice: $\Pr(d) \propto \frac{1}{d}$, d is distance from joint.
- Sites chosen at all length scales with equal probability. Probability that $i \in [L, 2L]$ is $O(1/\log N)$; "scale-free" pivot moves.
- Lower bound for τ_{int}: the time necessary to achieve a pivot before the first intersection.
- Plausible upper bound for τ_{int}: time necessary for successful pivots to be achieved at all possible length scales, i.e. $O(\log N)$ successful pivots.
- $\tau_{\mathrm{int}} = O(N^p \log N)$.

- Choose pivot sites preferentially close to the joint.
- Natural to choose using a power law.
- Robust choice: $\Pr(d) \propto \frac{1}{d}$, d is distance from joint.
- Sites chosen at all length scales with equal probability. Probability that $i \in [L, 2L]$ is $O(1/\log N)$; “scale-free” pivot moves.
- Lower bound for τ_{int}: the time necessary to achieve a pivot before the first intersection.
- Plausible upper bound for τ_{int}: time necessary for successful pivots to be achieved at all possible length scales, i.e. $O(\log N)$ successful pivots.
- $\tau_{\text{int}} = O(N^p \log N)$.

- Choose pivot sites preferentially close to the joint.
- Natural to choose using a power law.
- Robust choice: $\Pr(d) \propto \frac{1}{d}$, d is distance from joint.
- Sites chosen at all length scales with equal probability. Probability that $i \in [L, 2L]$ is $O(1/\log N)$; "scale-free" pivot moves.
- Lower bound for τ_{int}: the time necessary to achieve a pivot before the first intersection.
- Plausible upper bound for τ_{int}: time necessary for successful pivots to be achieved at all possible length scales, i.e. $O(\log N)$ successful pivots.
- $\tau_{\text{int}} = O(N^p \log N)$.

- Choose pivot sites preferentially close to the joint.
- Natural to choose using a power law.
- Robust choice: $\Pr(d) \propto \frac{1}{d}$, d is distance from joint.
- Sites chosen at all length scales with equal probability. Probability that $i \in [L, 2L]$ is $O(1/\log N)$; "scale-free" pivot moves.
- Lower bound for τ_{int}: the time necessary to achieve a pivot before the first intersection.
- Plausible upper bound for τ_{int}: time necessary for successful pivots to be achieved at all possible length scales, i.e. $O(\log N)$ successful pivots.
- $\tau_{\mathrm{int}} = O(N^p \log N)$.

- Choose pivot sites preferentially close to the joint.
- Natural to choose using a power law.
- Robust choice: $\Pr(d) \propto \frac{1}{d}$, d is distance from joint.
- Sites chosen at all length scales with equal probability. Probability that $i \in [L, 2L]$ is $O(1/\log N)$; "scale-free" pivot moves.
- Lower bound for τ_{int}: the time necessary to achieve a pivot before the first intersection.
- Plausible upper bound for τ_{int}: time necessary for successful pivots to be achieved at all possible length scales, i.e. $O(\log N)$ successful pivots.
- $\tau_{\mathrm{int}} = O(N^p \log N)$.

Comparison of parameter estimates.

- From $\langle B \rangle$, accurate calculation of the critical exponent γ for $d = 3$, $\gamma = 1.156957(9)$.
- Compare with 1.1573(2) (MC, Hsu & Grassberger, 2004), 1.1575(6) (MC, Caracciolo et al. 1998), and 1.1569(6) (enumeration, Clisby et al. 2007).

Comparison of parameter estimates.

- From $\langle B \rangle$, accurate calculation of the critical exponent γ for $d = 3$, $\gamma = 1.156957(9)$.
- Compare with 1.1573(2) (MC, Hsu & Grassberger, 2004), 1.1575(6) (MC, Caracciolo et al. 1998), and 1.1569(6) (enumeration, Clisby et al. 2007).

- Similar trick for the calculation of μ.
- Estimated $\mu = 4.6840395(5)$ in about 15000 CPU hours.
- Compare with 4.684038(6) (MC, Hsu & Grassberger, 2004), 4.684043(12) (enumeration, Clisby et al., 2007).

- Similar trick for the calculation of μ.
- Estimated $\mu = 4.6840395(5)$ in about 15000 CPU hours.
- Compare with 4.684038(6) (MC, Hsu & Grassberger, 2004), 4.684043(12) (enumeration, Clisby et al., 2007).

- Similar trick for the calculation of μ.
- Estimated $\mu = 4.6840395(5)$ in about 15000 CPU hours.
- Compare with 4.684038(6) (MC, Hsu & Grassberger, 2004), 4.684043(12) (enumeration, Clisby et al., 2007).

Scale-free moves

- If polymer system has internal length scales $N^{\alpha_1}, N^{\alpha_2}, \cdots$, choose moves uniformly from all possible length scales.
- Moves at smallest length scales will be accepted with high probability but result in little change.
- Larger length scales: low probability, large change.
- Algorithm self-selects the length scales which do the most work, with a $\log N$ penalty for τ_{int}.

Scale-free moves

- If polymer system has internal length scales $N^{\alpha_1}, N^{\alpha_2}, \cdots$, choose moves uniformly from all possible length scales.
- Moves at smallest length scales will be accepted with high probability but result in little change.
- Larger length scales: low probability, large change.
- Algorithm self-selects the length scales which do the most work, with a $\log N$ penalty for τ_{int}.

Scale-free moves

- If polymer system has internal length scales $N^{\alpha_1}, N^{\alpha_2}, \cdots$, choose moves uniformly from all possible length scales.
- Moves at smallest length scales will be accepted with high probability but result in little change.
- Larger length scales: low probability, large change.
- Algorithm self-selects the length scales which do the most work, with a $\log N$ penalty for τ_{int}.

Scale-free moves

- If polymer system has internal length scales $N^{\alpha_1}, N^{\alpha_2}, \cdots$, choose moves uniformly from all possible length scales.
- Moves at smallest length scales will be accepted with high probability but result in little change.
- Larger length scales: low probability, large change.
- Algorithm self-selects the length scales which do the most work, with a $\log N$ penalty for τ_{int}.

Confined polymers

- Length scales introduced by putting polymer in a confined region, e.g. between two parallel plates, or in a tube.
- Perform cut-and-paste moves on polymer.
- If we select endpoints of sub-walks uniformly from log(distance), we guarantee that all length scales will be accounted for.
- Cut-and-paste moves (including pivots).
- Moves may have different characteristic length scales, e.g. for polymer confined between parallel plates, rotations in x-y plane global, restricted for other planes.
- Automatic tuning by selecting site separation uniformly from log(distance).

Confined polymers

- Length scales introduced by putting polymer in a confined region, e.g. between two parallel plates, or in a tube.
- Perform cut-and-paste moves on polymer.
- If we select endpoints of sub-walks uniformly from log(distance), we guarantee that all length scales will be accounted for.
- Cut-and-paste moves (including pivots).
- Moves may have different characteristic length scales, e.g. for polymer confined between parallel plates, rotations in x-y plane global, restricted for other planes.
- Automatic tuning by selecting site separation uniformly from log(distance).

Confined polymers

- Length scales introduced by putting polymer in a confined region, e.g. between two parallel plates, or in a tube.
- Perform cut-and-paste moves on polymer.
- If we select endpoints of sub-walks uniformly from log(distance), we guarantee that all length scales will be accounted for.
- Cut-and-paste moves (including pivots).
- Moves may have different characteristic length scales, e.g. for polymer confined between parallel plates, rotations in x-y plane global, restricted for other planes.
- Automatic tuning by selecting site separation uniformly from log(distance).

Confined polymers

- Length scales introduced by putting polymer in a confined region, e.g. between two parallel plates, or in a tube.
- Perform cut-and-paste moves on polymer.
- If we select endpoints of sub-walks uniformly from log(distance), we guarantee that all length scales will be accounted for.
- Cut-and-paste moves (including pivots).
- Moves may have different characteristic length scales, e.g. for polymer confined between parallel plates, rotations in x-y plane global, restricted for other planes.
- Automatic tuning by selecting site separation uniformly from log(distance).

Confined polymers

- Length scales introduced by putting polymer in a confined region, e.g. between two parallel plates, or in a tube.
- Perform cut-and-paste moves on polymer.
- If we select endpoints of sub-walks uniformly from log(distance), we guarantee that all length scales will be accounted for.
- Cut-and-paste moves (including pivots).
- Moves may have different characteristic length scales, e.g. for polymer confined between parallel plates, rotations in x-y plane global, restricted for other planes.
- Automatic tuning by selecting site separation uniformly from log(distance).

Confined polymers

- Length scales introduced by putting polymer in a confined region, e.g. between two parallel plates, or in a tube.
- Perform cut-and-paste moves on polymer.
- If we select endpoints of sub-walks uniformly from log(distance), we guarantee that all length scales will be accounted for.
- Cut-and-paste moves (including pivots).
- Moves may have different characteristic length scales, e.g. for polymer confined between parallel plates, rotations in x-y plane global, restricted for other planes.
- Automatic tuning by selecting site separation uniformly from log(distance).

Other systems with intermediate length scales

- Polymer knotting (knots are localized on self-avoiding polygons).
- Polymers near θ (collapse) transition.
- Star polymers / branched polymers (distance to branch point).
- Polymers tethered to a surface.

Other systems with intermediate length scales

- Polymer knotting (knots are localized on self-avoiding polygons).
- Polymers near θ (collapse) transition.
- Star polymers / branched polymers (distance to branch point).
- Polymers tethered to a surface.

Other systems with intermediate length scales

- Polymer knotting (knots are localized on self-avoiding polygons).
- Polymers near θ (collapse) transition.
- Star polymers / branched polymers (distance to branch point).
- Polymers tethered to a surface.

Other systems with intermediate length scales

- Polymer knotting (knots are localized on self-avoiding polygons).
- Polymers near θ (collapse) transition.
- Star polymers / branched polymers (distance to branch point).
- Polymers tethered to a surface.

Summary

- SAW-tree data structure has resulted in much faster implementation of pivot algorithm, and other global moves.
- Cut-and-paste moves can be tuned to arbitrary length scales.
- Where there are intermediate length scales, which may be poorly understood, using scale-free moves for polymer simulation ensures that all length scales of the system are probed for a modest $\log N$ penalty.
- System "selects" moves which do the most work, no need to choose length scales by hand. Robust, simple, efficient.

Summary

- SAW-tree data structure has resulted in much faster implementation of pivot algorithm, and other global moves.
- Cut-and-paste moves can be tuned to arbitrary length scales.
- Where there are intermediate length scales, which may be poorly understood, using scale-free moves for polymer simulation ensures that all length scales of the system are probed for a modest log N penalty.
- System "selects" moves which do the most work, no need to choose length scales by hand. Robust, simple, efficient.

Summary

- SAW-tree data structure has resulted in much faster implementation of pivot algorithm, and other global moves.
- Cut-and-paste moves can be tuned to arbitrary length scales.
- Where there are intermediate length scales, which may be poorly understood, using scale-free moves for polymer simulation ensures that all length scales of the system are probed for a modest $\log N$ penalty.
- System "selects" moves which do the most work, no need to choose length scales by hand. Robust, simple, efficient.

Summary

- SAW-tree data structure has resulted in much faster implementation of pivot algorithm, and other global moves.
- Cut-and-paste moves can be tuned to arbitrary length scales.
- Where there are intermediate length scales, which may be poorly understood, using scale-free moves for polymer simulation ensures that all length scales of the system are probed for a modest $\log N$ penalty.
- System "selects" moves which do the most work, no need to choose length scales by hand. Robust, simple, efficient.

- N. Clisby, *Efficient implementation of the pivot algorithm for self-avoiding walks*, J. Stat. Phys. **140** (2010), 349–392.
- Nathan Clisby, *Accurate estimate of the critical exponent ν for self-avoiding walks via a fast implementation of the pivot algorithm*, Phys. Rev. Lett. **104** (2010), 055702.
- E. J. Janse van Rensburg, *Monte Carlo methods for the self-avoiding walk*, J. Phys. A: Math. Theor. **42** (2009), 323001 (97pp).
- Alan D. Sokal, *Monte Carlo methods for the self-avoiding walk*, arXiv:`hep-lat/9405016`, 1994.