Monte Carlo simulation of self-avoiding walks

Nathan Clisby
MASCOS, The University of Melbourne

36$^{\text{th}}$ Conference on Stochastic Processes and Their Applications
University of Colorado Boulder
July 31, 2013

Introduction

My research focus is on designing fast computer algorithms for

Introduction

My research focus is on designing fast computer algorithms for

- enumeration of self-avoiding walks (SAW) and other lattice objects;

Introduction

My research focus is on designing fast computer algorithms for

- enumeration of self-avoiding walks (SAW) and other lattice objects;
- Monte Carlo sampling of self-avoiding walks and polymers.

Introduction

My research focus is on designing fast computer algorithms for

- enumeration of self-avoiding walks (SAW) and other lattice objects;
- Monte Carlo sampling of self-avoiding walks and polymers.
- In fact, key insight described here led initially to a bad enumeration algorithm, then a good Monte Carlo algorithm.

Self-avoiding walk model

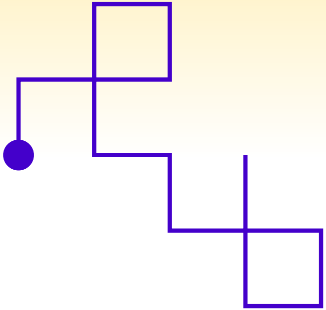
- A walk on a lattice, step to neighbouring site provided it has not already been visited.

Self-avoiding walk model

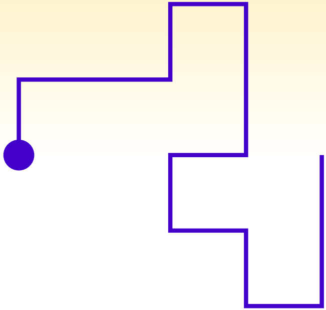
- A walk on a lattice, step to neighbouring site provided it has not already been visited.
- Models polymers in good solvent limit.

Self-avoiding walk model

- A walk on a lattice, step to neighbouring site provided it has not already been visited.
- Models polymers in good solvent limit.
- Exactly captures universal properties such as critical exponents.



Simple random walk



Self-avoiding walk

Typical self-avoiding walks

Asymptotic behaviour

We are interested in

Asymptotic behaviour

We are interested in

- the number of SAW of length N, c_N:

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

Asymptotic behaviour

We are interested in

- the number of SAW of length N, c_N:

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

- and the mean square end to end distance:

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

Asymptotic behaviour

We are interested in

- the number of SAW of length N, c_N:

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

- and the mean square end to end distance:

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

- γ and ν are *universal* exponents.

Asymptotic behaviour

We are interested in

- the number of SAW of length N, c_N:

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

- and the mean square end to end distance:

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

- γ and ν are *universal* exponents.
- μ is the connective constant; lattice dependent.

Why do we want to sample SAW rapidly?

Why do we want to sample SAW rapidly?

- Estimate γ, ν, and μ.

Why do we want to sample SAW rapidly?

- Estimate γ, ν, and μ.
- $\mathrm{SLE}_{8/3}$: confirm correspondence, study properties.

Why do we want to sample SAW rapidly?

- Estimate γ, ν, and μ.
- $\mathrm{SLE}_{8/3}$: confirm correspondence, study properties.
- Confirm predictions of CFT.

Why do we want to sample SAW rapidly?

- Estimate γ, ν, and μ.
- $\mathrm{SLE}_{8/3}$: confirm correspondence, study properties.
- Confirm predictions of CFT.
- Polymer knotting, perhaps protein folding.

Why do we want to sample SAW rapidly?

- Estimate γ, ν, and μ.
- $\mathrm{SLE}_{8/3}$: confirm correspondence, study properties.
- Confirm predictions of CFT.
- Polymer knotting, perhaps protein folding.
- Algorithms can be applied to realistic polymer systems, perhaps combining Monte Carlo with Molecular Dynamics.

Why are fast simulations of SAW possible?

Why are fast simulations of SAW possible?

SAW have a natural one-dimensional order $\cdots$

Why are fast simulations of SAW possible?

SAW have a natural one-dimensional order $\cdots$
$\cdots$ and segments of the walk which are well separated on the chain are (typically) well separated in space.

Markov chain Monte Carlo (MCMC)

- Want to estimate physical properties of system by sampling from all possible configurations (state space).

Markov chain Monte Carlo (MCMC)

- Want to estimate physical properties of system by sampling from all possible configurations (state space).
- Basic idea: generate new configurations by deforming current configuration via a “move”.

Markov chain Monte Carlo (MCMC)

- Want to estimate physical properties of system by sampling from all possible configurations (state space).
- Basic idea: generate new configurations by deforming current configuration via a "move".
- Trade-offs:

Markov chain Monte Carlo (MCMC)

- Want to estimate physical properties of system by sampling from all possible configurations (state space).
- Basic idea: generate new configurations by deforming current configuration via a "move".
- Trade-offs:
  - small deformations are fast, but many moves required for configuration to change significantly

Markov chain Monte Carlo (MCMC)

- Want to estimate physical properties of system by sampling from all possible configurations (state space).
- Basic idea: generate new configurations by deforming current configuration via a "move".
- Trade-offs:
  - small deformations are fast, but many moves required for configuration to change significantly
  - large deformations are slow, but move rapidly around state space

Pivot algorithm

- How can we sample self-avoiding walks?

Pivot algorithm

- How can we sample self-avoiding walks?
- Local moves make a small deformation, e.g. adding or removing a step, and take CPU time $O(N^2)$ to get an "essentially new" configuration.

Pivot algorithm

- How can we sample self-avoiding walks?
- Local moves make a small deformation, e.g. adding or removing a step, and take CPU time $O(N^2)$ to get an "essentially new" configuration.
- Global moves can do better.

Pivot algorithm

- How can we sample self-avoiding walks?
- Local moves make a small deformation, e.g. adding or removing a step, and take CPU time $O(N^2)$ to get an "essentially new" configuration.
- Global moves can do better.
- Pivot algorithm, invented in 1969 by Lal but studied in depth by Madras and Sokal in 1988.

Pivot algorithm

- How can we sample self-avoiding walks?
- Local moves make a small deformation, e.g. adding or removing a step, and take CPU time $O(N^2)$ to get an "essentially new" configuration.
- Global moves can do better.
- Pivot algorithm, invented in 1969 by Lal but studied in depth by Madras and Sokal in 1988.
- More CPU time per move, but still dramatically more efficient.

Pivot algorithm

- How can we sample self-avoiding walks?
- Local moves make a small deformation, e.g. adding or removing a step, and take CPU time $O(N^2)$ to get an "essentially new" configuration.
- Global moves can do better.
- Pivot algorithm, invented in 1969 by Lal but studied in depth by Madras and Sokal in 1988.
- More CPU time per move, but still dramatically more efficient.
- Made it possible to study dramatically longer SAW (from hundreds of steps to tens of thousands).

Pivot algorithm

- Procedure:

Pivot algorithm

- Procedure:
  - Choose a pivot site at random.

Pivot algorithm

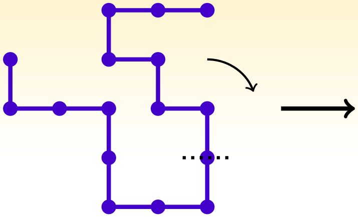
- Procedure:
  - Choose a pivot site at random.
  - Then rotate or reflect one of the two parts of the walk.

Pivot algorithm

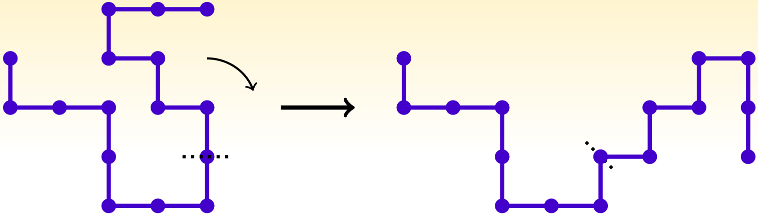
- Procedure:
  - Choose a pivot site at random.
  - Then rotate or reflect one of the two parts of the walk.
  - Retain new walk if it is self-avoiding, otherwise restore original walk.

Pivot algorithm

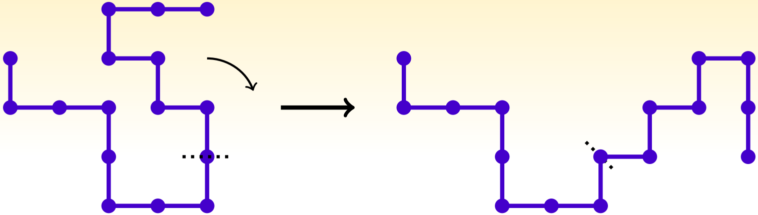
- Procedure:
  - Choose a pivot site at random.
  - Then rotate or reflect one of the two parts of the walk.
  - Retain new walk if it is self-avoiding, otherwise restore original walk.
- "Global" because on average one quarter of the monomers are moved.



Example pivot move



Example pivot move



Example pivot move

Run simulation

Why is it so effective?

- Pivots are rarely successful.

Why is it so effective?

- Pivots are rarely successful.
- Every time a pivot attempt *is* successful there is a large change in global observables.

Why is it so effective?

- Pivots are rarely successful.
- Every time a pivot attempt *is* successful there is a large change in global observables.
- After each successful pivot, the successive values of global properties e.g. $\langle R_e^2 \rangle_N$ are (almost) uncorrelated.

Why is it so effective?

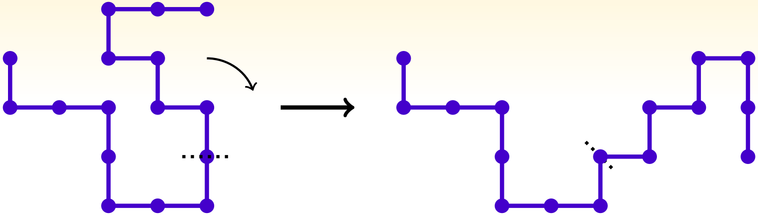
- Pivots are rarely successful.
- Every time a pivot attempt *is* successful there is a large change in global observables.
- After each successful pivot, the successive values of global properties e.g. $\langle R_e^2 \rangle_N$ are (almost) uncorrelated.
- Integrated autocorrelation time for global observables $O(N^p)$, $p \approx 0.19$ for square lattice, $p \approx 0.11$ for cubic.

Why is it so effective?

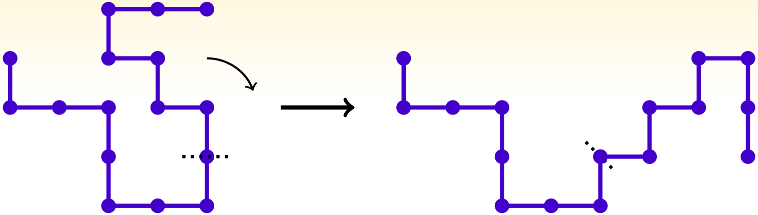
- Pivots are rarely successful.
- Every time a pivot attempt *is* successful there is a large change in global observables.
- After each successful pivot, the successive values of global properties e.g. $\langle R_e^2 \rangle_N$ are (almost) uncorrelated.
- Integrated autocorrelation time for global observables $O(N^p)$, $p \approx 0.19$ for square lattice, $p \approx 0.11$ for cubic.
- Usual implementation: CPU time $O(N)$ for an "essentially new" configuration.

More to the story: pivot moves are in fact *local* moves if you think of them the right way.

More to the story: pivot moves are in fact *local* moves if you think of them the right way.

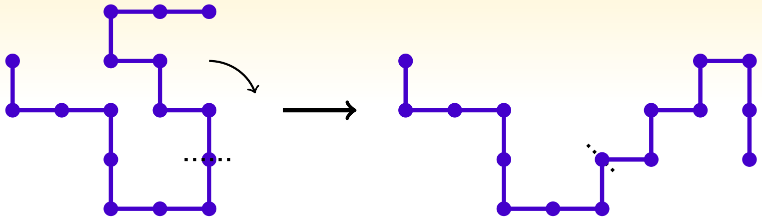


More to the story: pivot moves are in fact *local* moves if you think of them the right way.



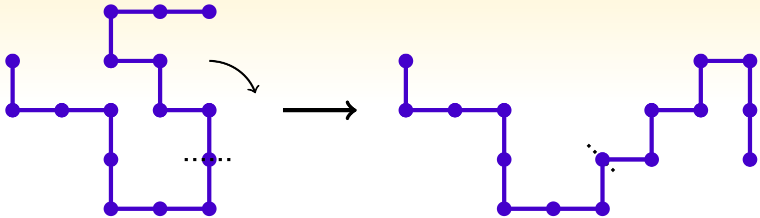
LSRSLSL**S**LRLRRS

More to the story: pivot moves are in fact *local* moves if you think of them the right way.



LSRSLSL**S**LRLRRS $\longrightarrow$ LSRSLSL**R**LRLRRS

More to the story: pivot moves are in fact *local* moves if you think of them the right way.



LSRSLSL**S**LRLRRS $\longrightarrow$ LSRSLSL**R**LRLRRS

With the right implementation, pivot move has *global effect* for the cost of a *local* move. Tricky part is checking that after subwalk is rotated the walk remains self-avoiding.

SAW-tree

Ingredients for a fast implementation of the pivot algorithm:

SAW-tree

Ingredients for a fast implementation of the pivot algorithm:

- a fast test for intersections for a proposed pivot move;

SAW-tree

Ingredients for a fast implementation of the pivot algorithm:

- a fast test for intersections for a proposed pivot move;
- a fast update operation to change walk if pivot is accepted.

SAW-tree

Ingredients for a fast implementation of the pivot algorithm:

- a fast test for intersections for a proposed pivot move;
- a fast update operation to change walk if pivot is accepted.

Key observation

SAW-tree

Ingredients for a fast implementation of the pivot algorithm:

- a fast test for intersections for a proposed pivot move;
- a fast update operation to change walk if pivot is accepted.

Key observation

SAW can be decomposed into two (equal) subwalks, with a symmetry operation concatenating the two subwalks.

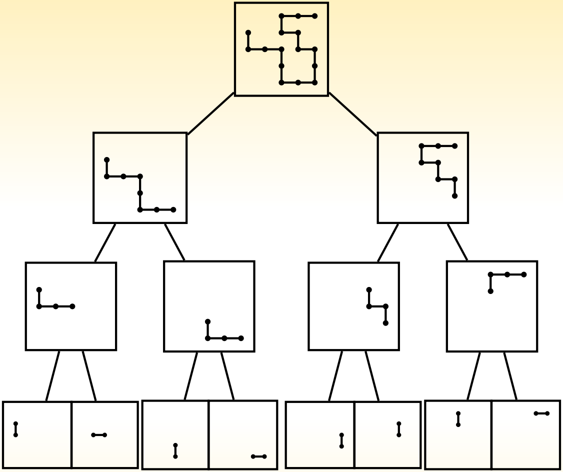
SAW-tree

Ingredients for a fast implementation of the pivot algorithm:

- a fast test for intersections for a proposed pivot move;
- a fast update operation to change walk if pivot is accepted.

Key observation

SAW can be decomposed into two (equal) subwalks, with a symmetry operation concatenating the two subwalks.
This results in a natural *binary tree* structure.



SAW-tree representation of a walk.

SAW-tree

- State of a walk, which includes information on global observables such as R_e^2, only depends on the states of its two subwalks.

SAW-tree

- State of a walk, which includes information on global observables such as R_e^2, only depends on the states of its two subwalks.
- R_e^2, R_g^2, R_m^2 can all be calculated in this way.

SAW-tree

- State of a walk, which includes information on global observables such as R_e^2, only depends on the states of its two subwalks.
- R_e^2, R_g^2, R_m^2 can all be calculated in this way.
- Problem of keeping track of changes after a pivot move is bookkeeping, can be done in time $O(\log N)$.

Intersection testing

- Mean number of *contacts* in an N-step SAW is $O(N)$, but almost all of these are between neighbouring segments of the chain.

Intersection testing

- Mean number of *contacts* in an N-step SAW is $O(N)$, but almost all of these are between neighbouring segments of the chain.
- Mean number of contacts between two *halves* of an N-step SAW is constant (Müller and Schäfer, 1998).

Intersection testing

How much information do we need about a walk in order to decide whether it is self-avoiding?

Intersection testing

How much information do we need about a walk in order to decide whether it is self-avoiding?

- Imagine examining a walk with a magnifying glass in hand.

Intersection testing

How much information do we need about a walk in order to decide whether it is self-avoiding?

- Imagine examining a walk with a magnifying glass in hand.
    - For parts of the walk which are far apart we can easily see that there are no intersections.

Intersection testing

How much information do we need about a walk in order to decide whether it is self-avoiding?

- Imagine examining a walk with a magnifying glass in hand.
    - For parts of the walk which are far apart we can easily see that there are no intersections.
    - Whenever parts of the walk approach each other we need to examine the walk closely using magnifying glass.

Intersection testing

How much information do we need about a walk in order to decide whether it is self-avoiding?

- Imagine examining a walk with a magnifying glass in hand.
  - For parts of the walk which are far apart we can easily see that there are no intersections.
  - Whenever parts of the walk approach each other we need to examine the walk closely using magnifying glass.
  - Degree of magnification depends on how closely they approach.

Intersection testing

How much information do we need about a walk in order to decide whether it is self-avoiding?

- Imagine examining a walk with a magnifying glass in hand.
  - For parts of the walk which are far apart we can easily see that there are no intersections.
  - Whenever parts of the walk approach each other we need to examine the walk closely using magnifying glass.
  - Degree of magnification depends on how closely they approach.
- This idea captured by storing *bounding box* information in a binary tree.

Pivot algorithm

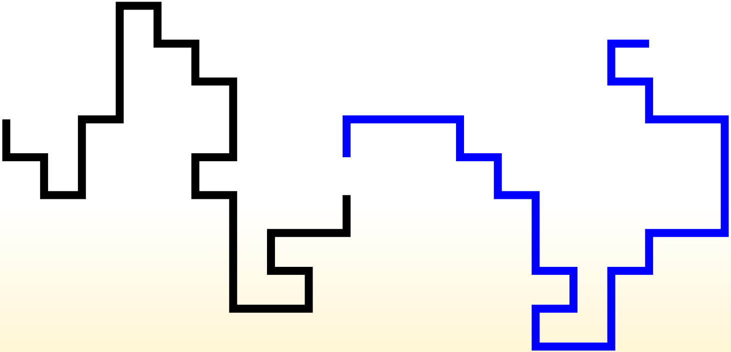
- After applying pivot to a SAW with 64 sites, will show algorithm to determine whether new configuration is self-avoiding.

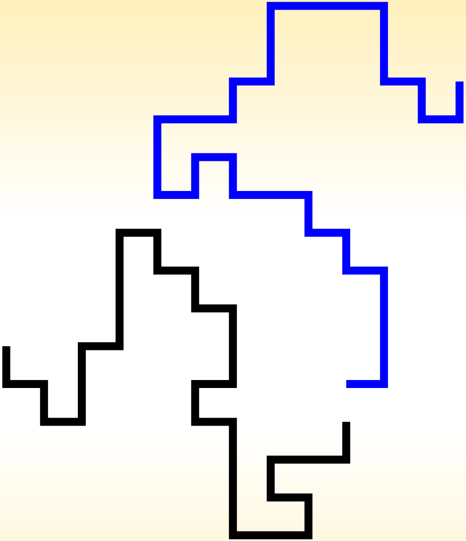
Pivot algorithm

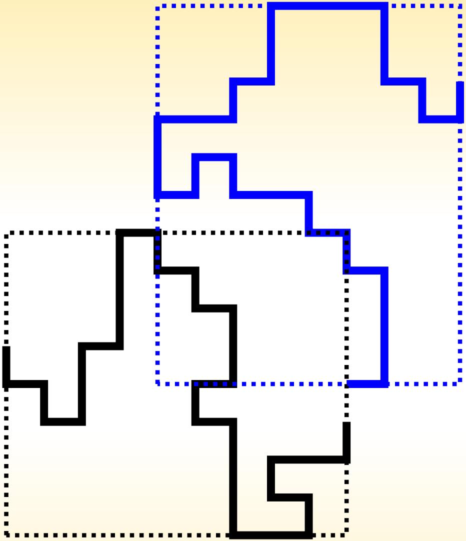
- After applying pivot to a SAW with 64 sites, will show algorithm to determine whether new configuration is self-avoiding.
- Algorithm uses "depth-first search" in an attempt to find intersections, recursively applying the observation that when the bounding box of two subwalks do not intersect, then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

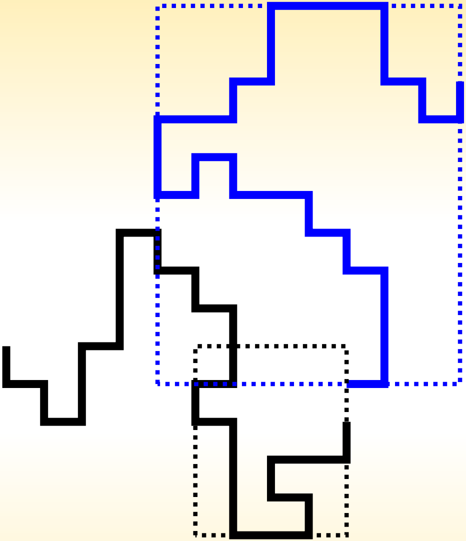
Pivot algorithm

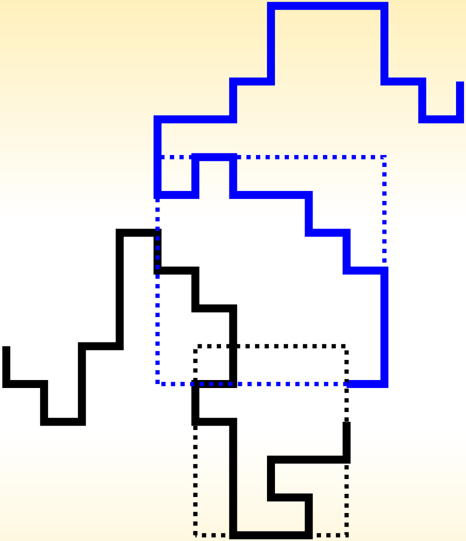
- After applying pivot to a SAW with 64 sites, will show algorithm to determine whether new configuration is self-avoiding.
- Algorithm uses "depth-first search" in an attempt to find intersections, recursively applying the observation that when the bounding box of two subwalks do not intersect, then the subwalks themselves cannot intersect.
- $O(1)$ contacts $\Longrightarrow$ mean CPU time $O(\log N)$ to complete test.

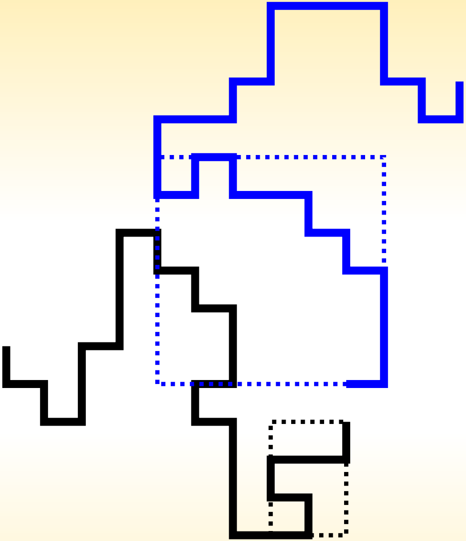


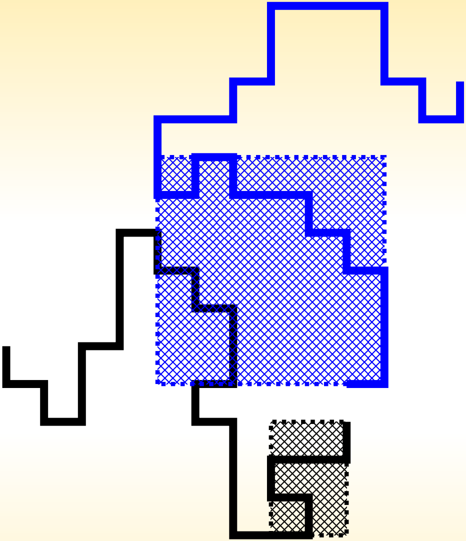


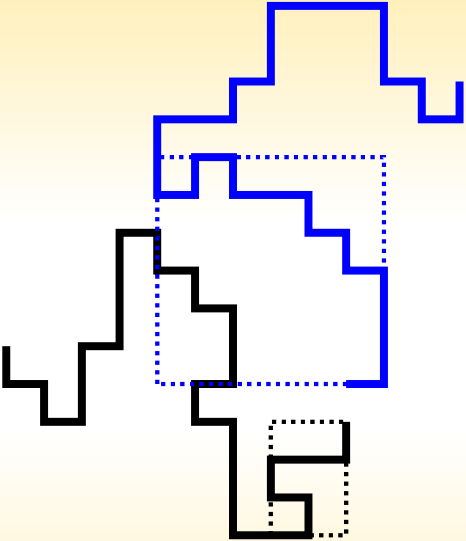


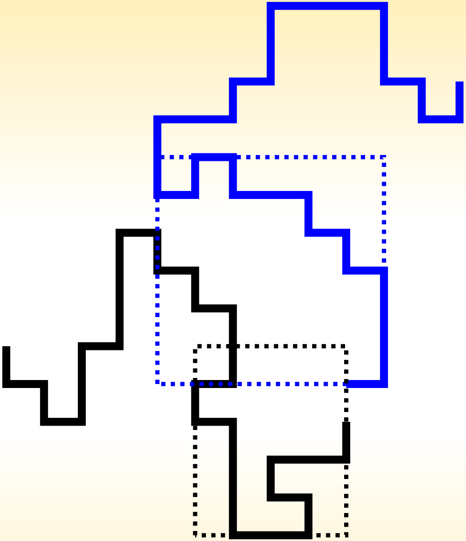


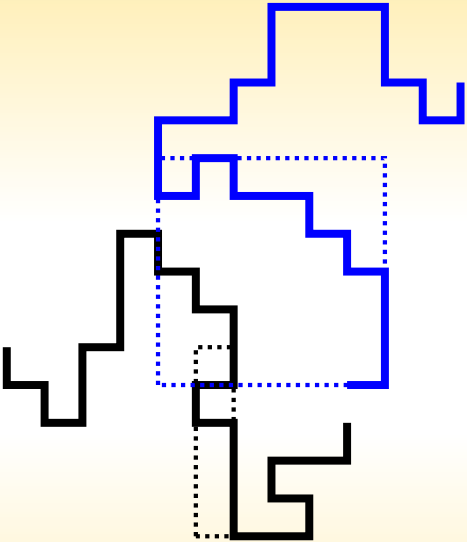


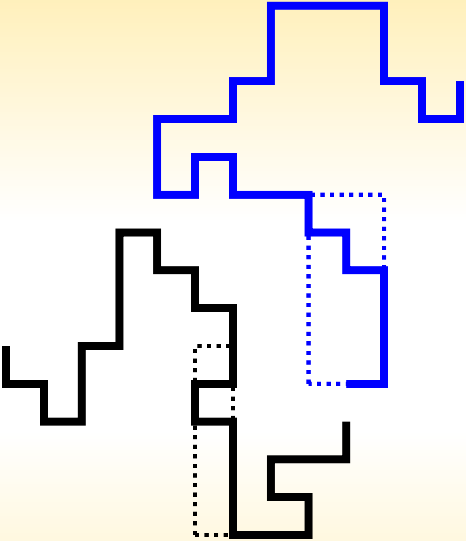


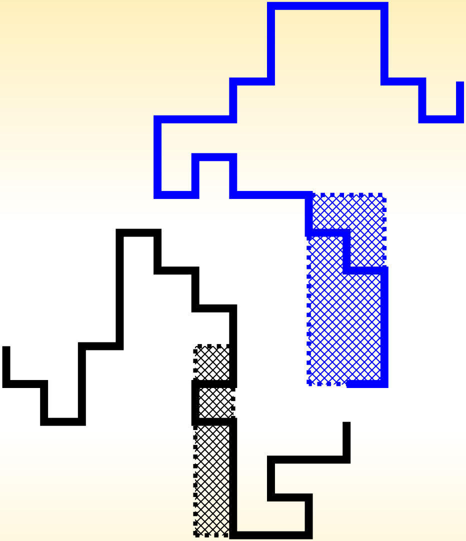


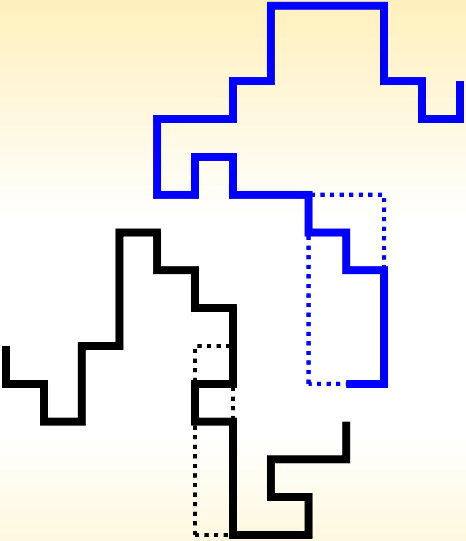


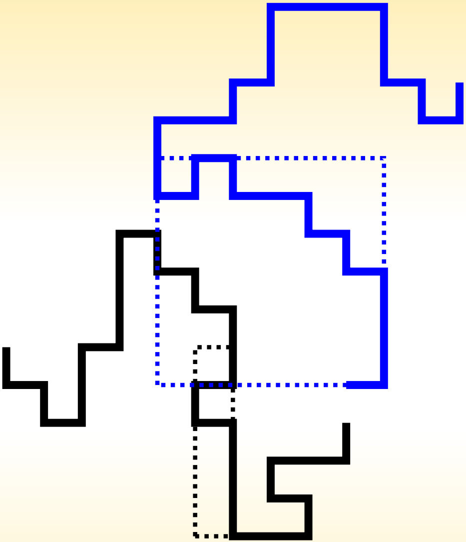


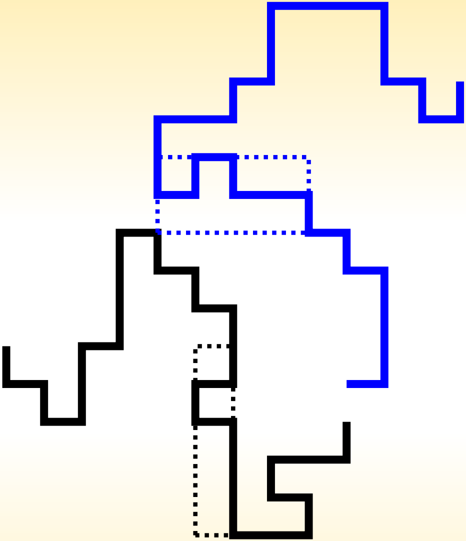


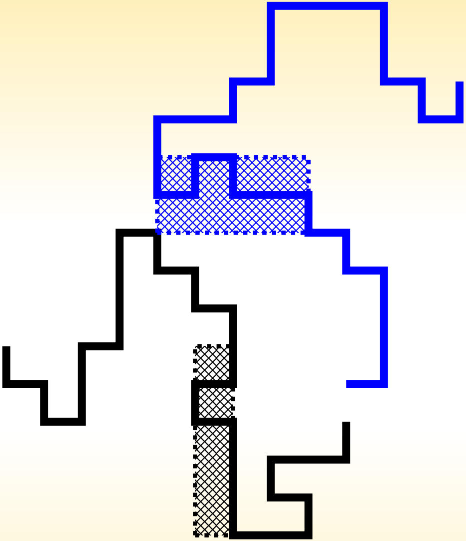


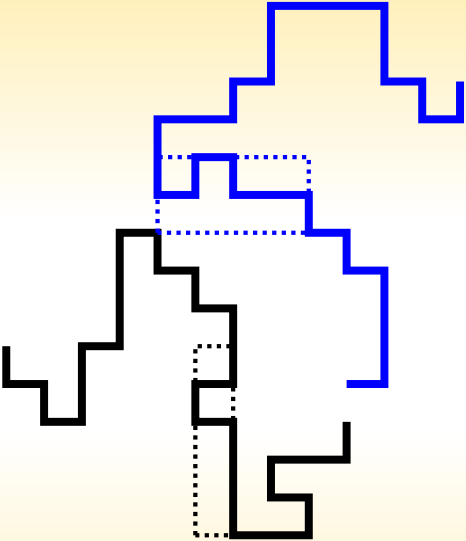


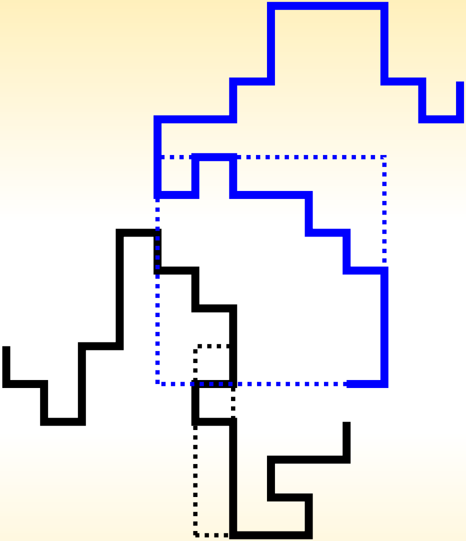


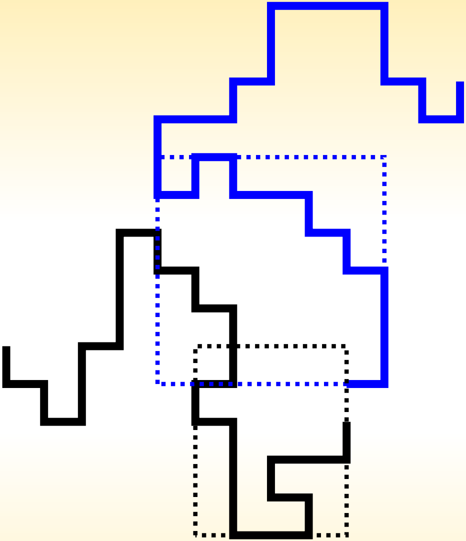


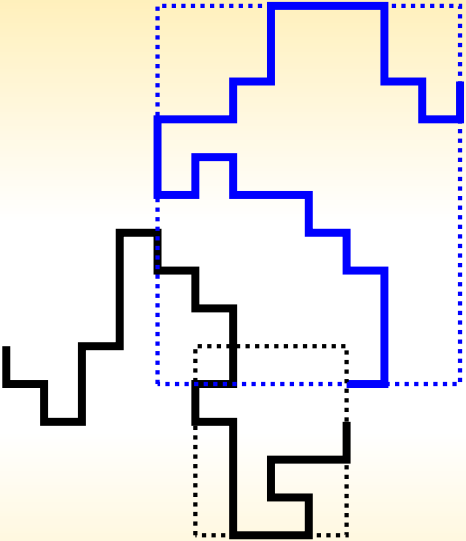


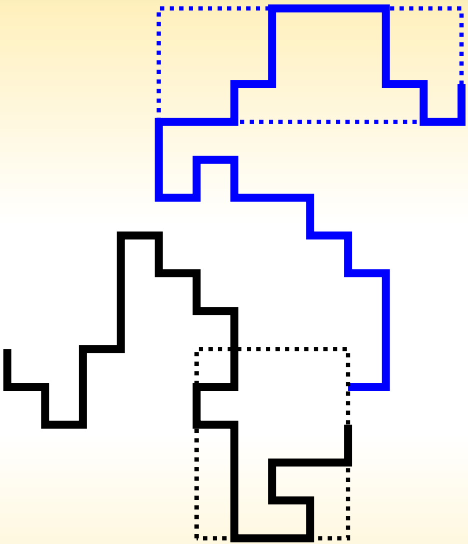


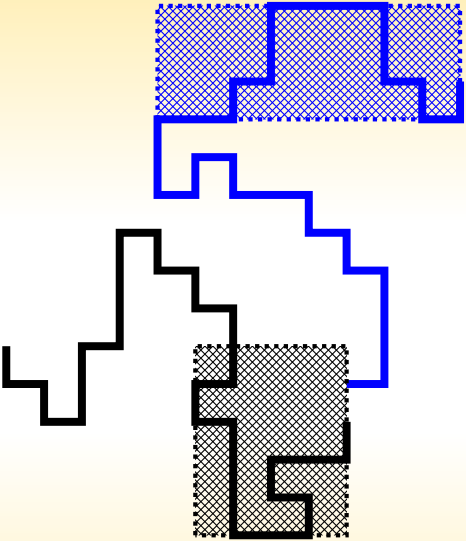


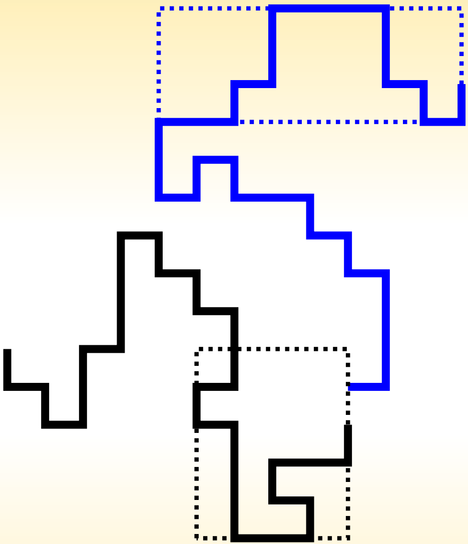


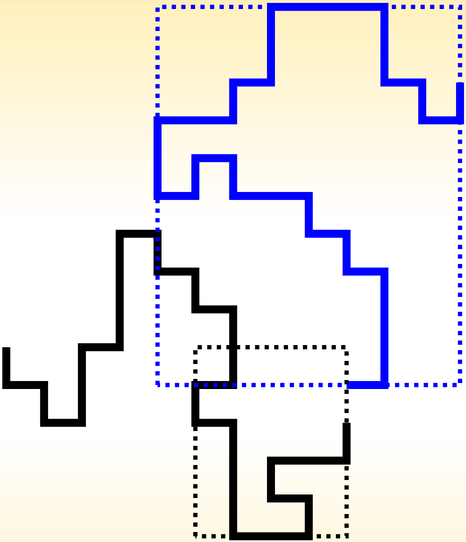


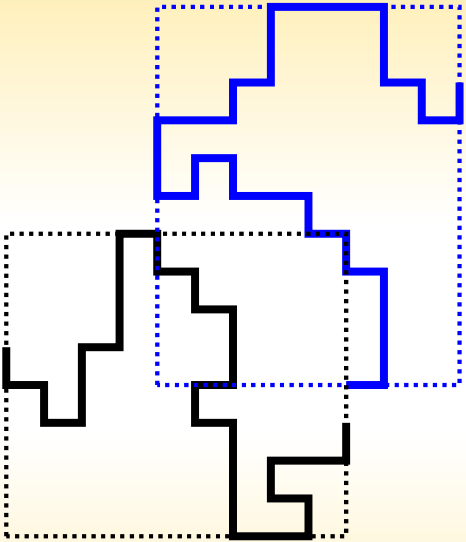


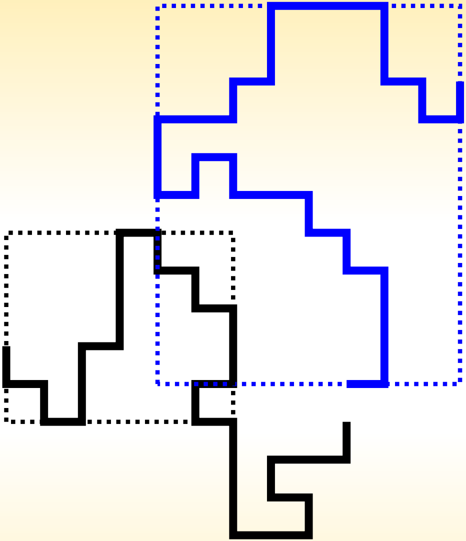


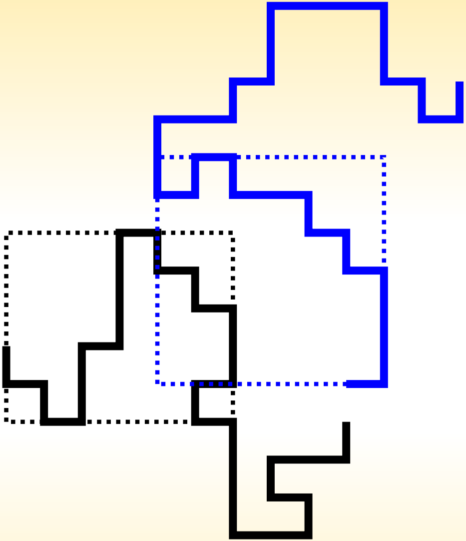


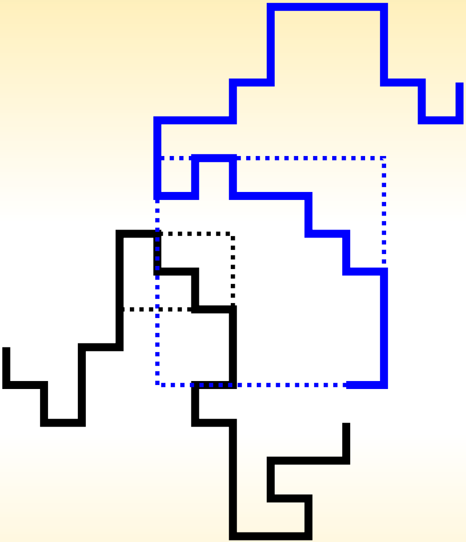


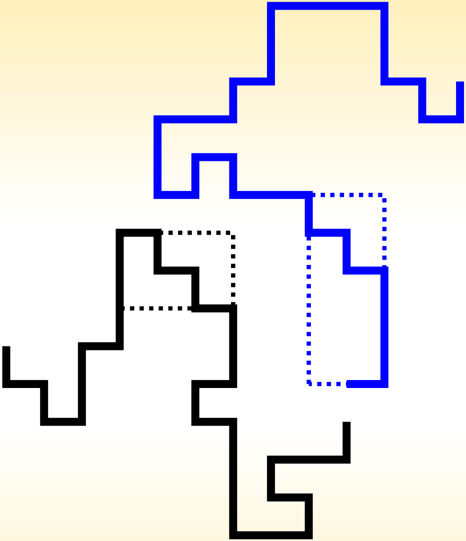


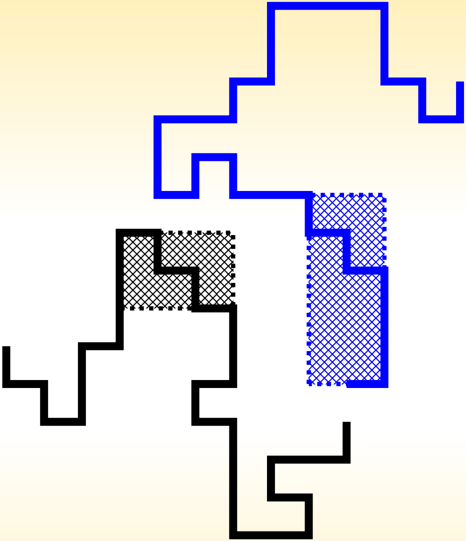


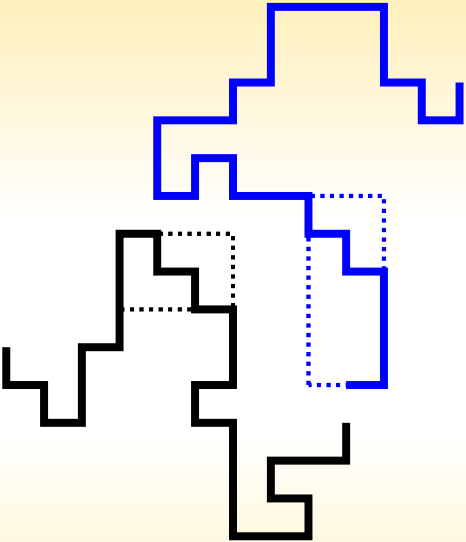


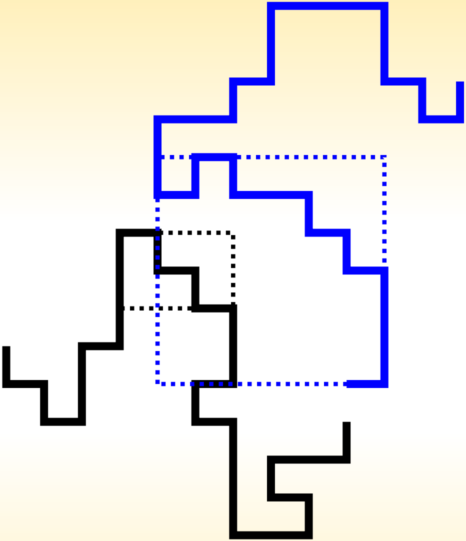


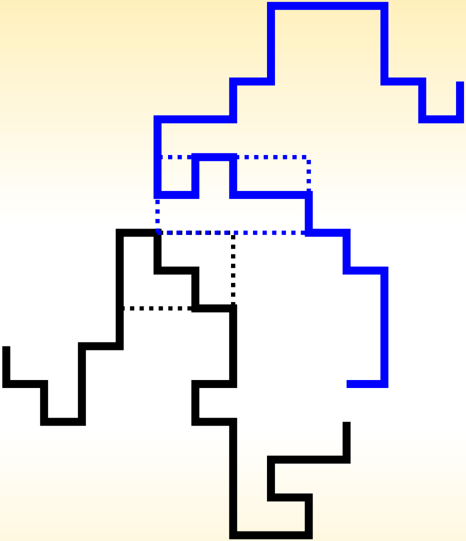


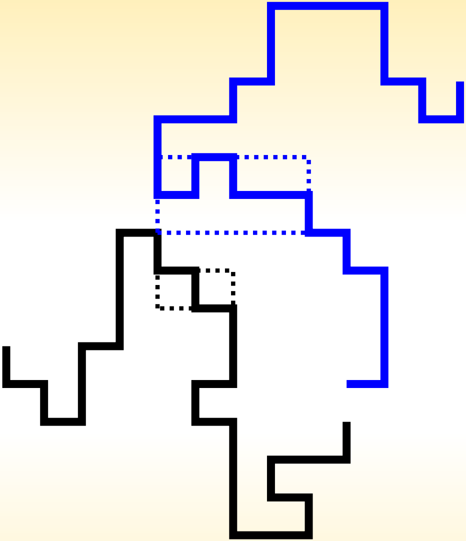


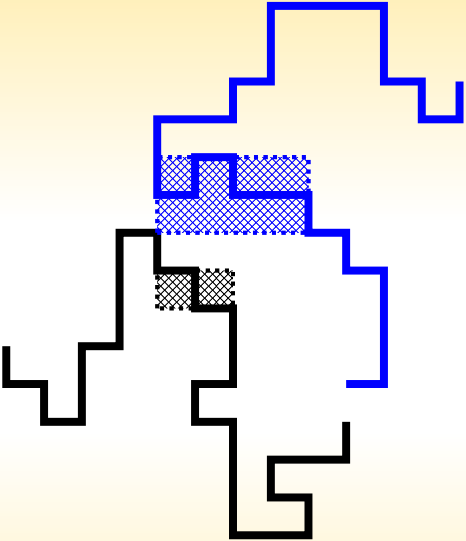


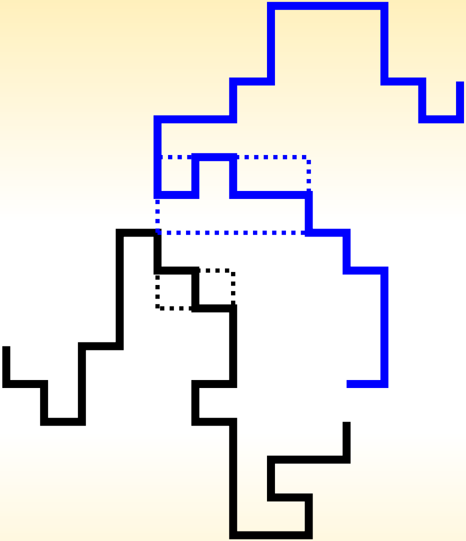


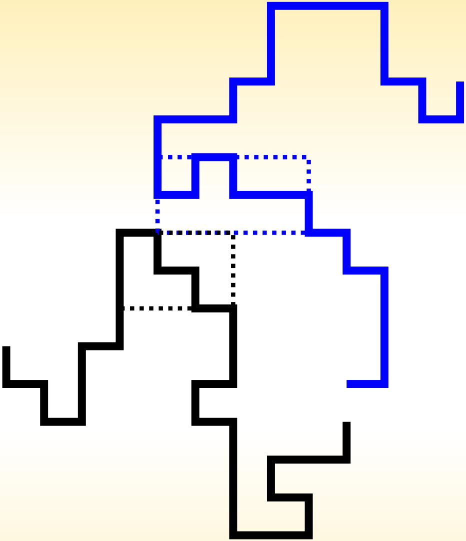


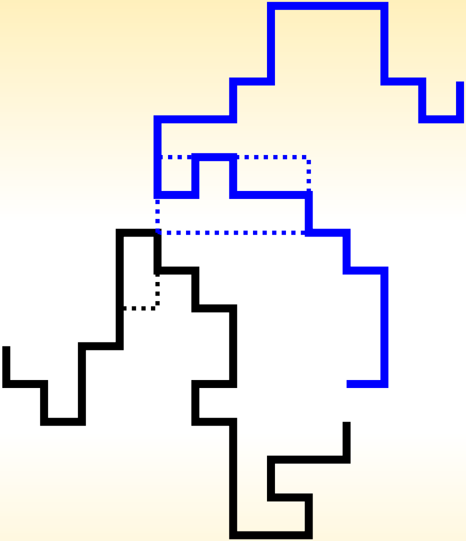


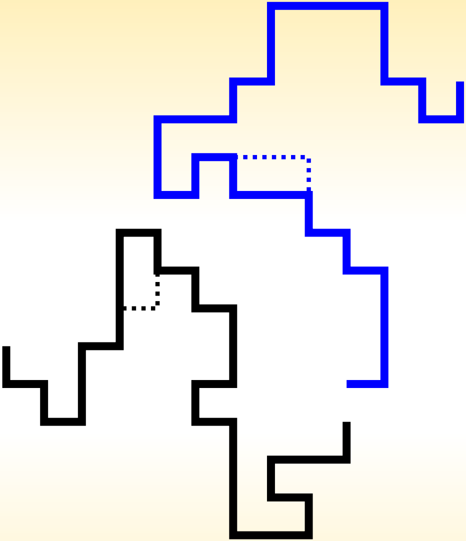


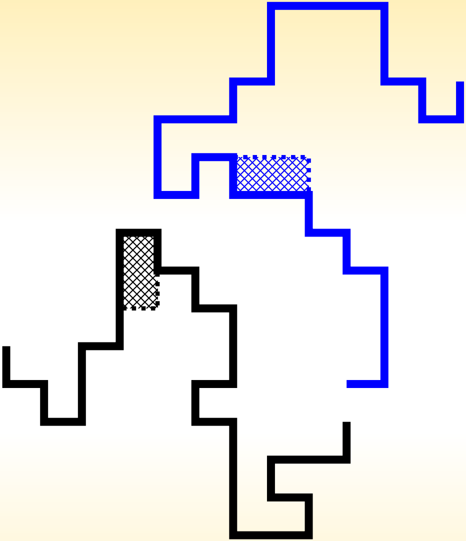


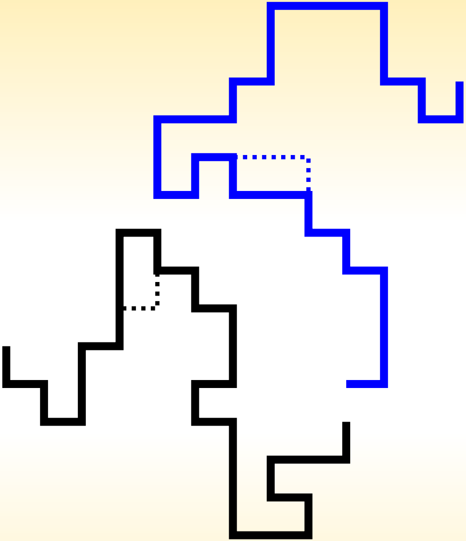


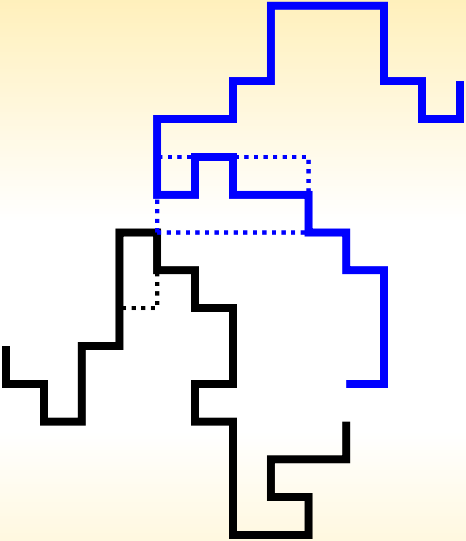


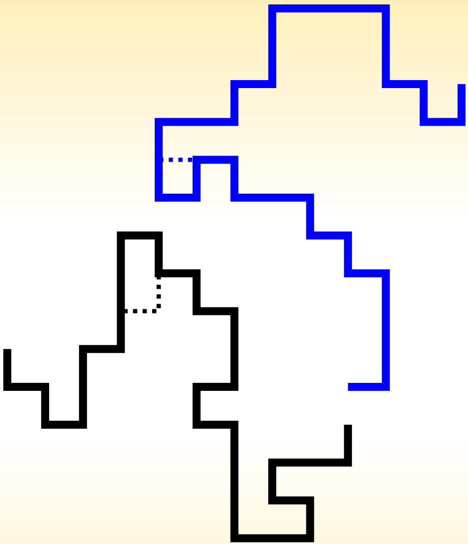


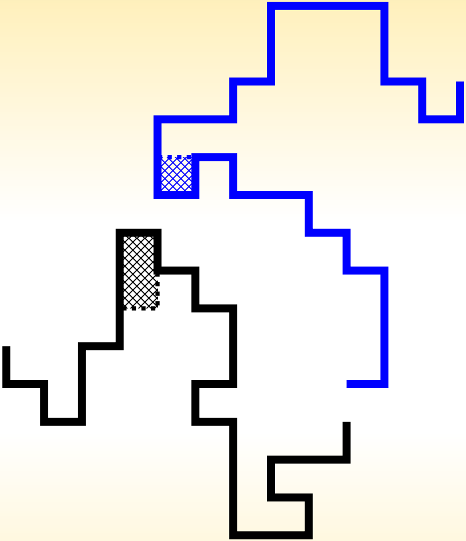


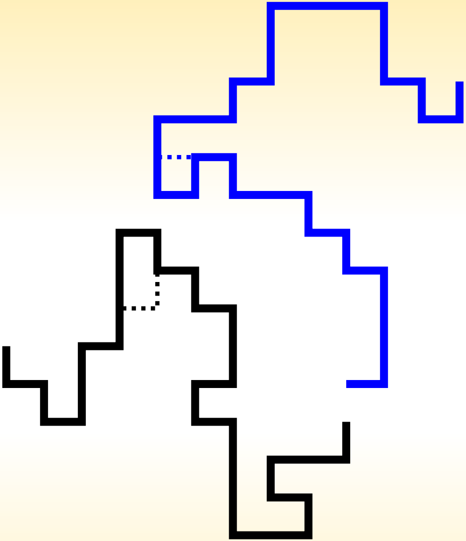


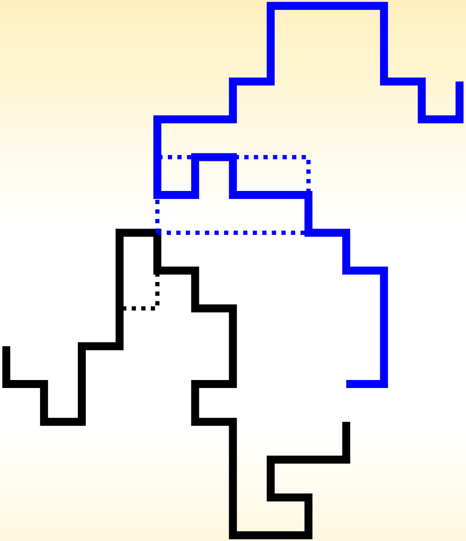


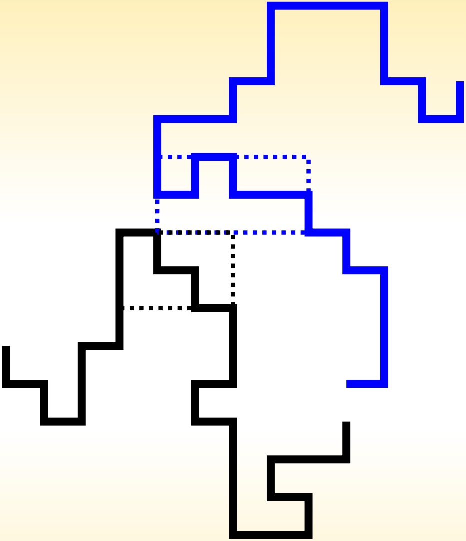


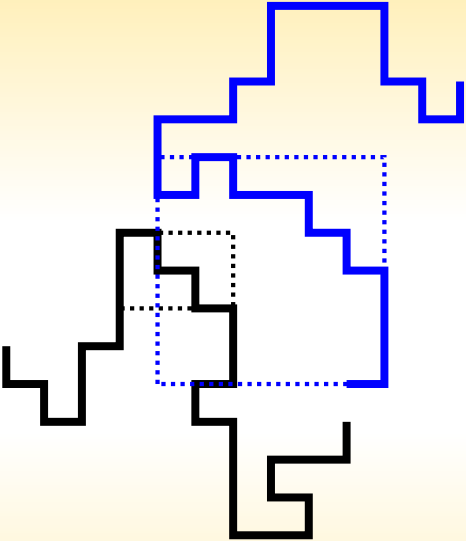


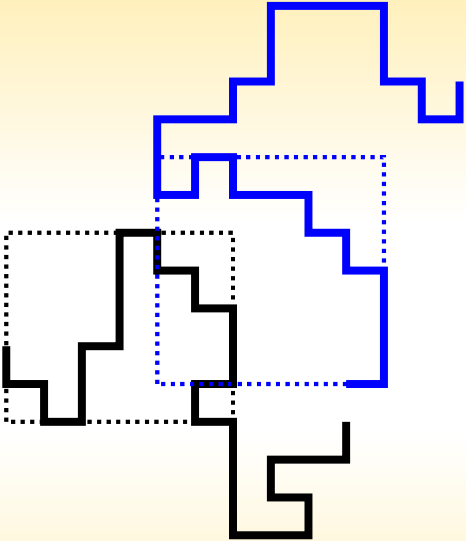


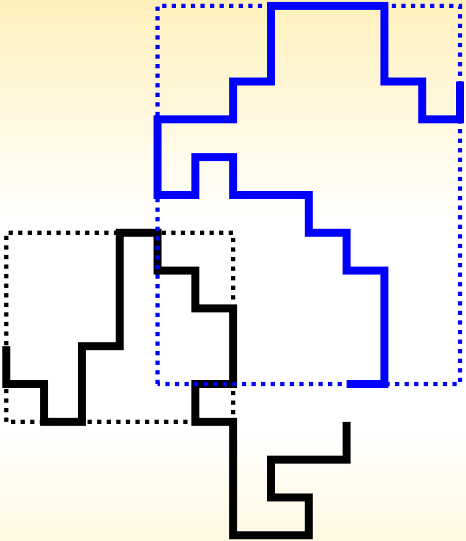


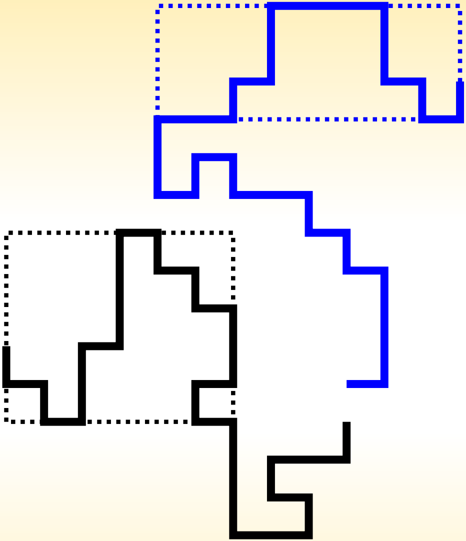


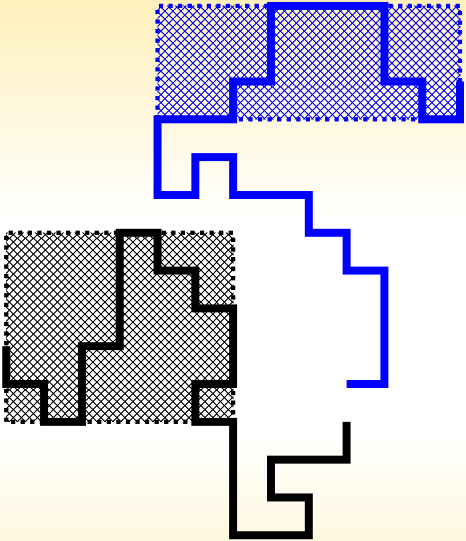


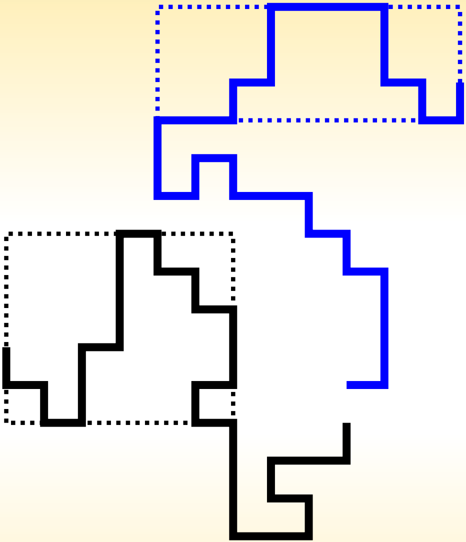


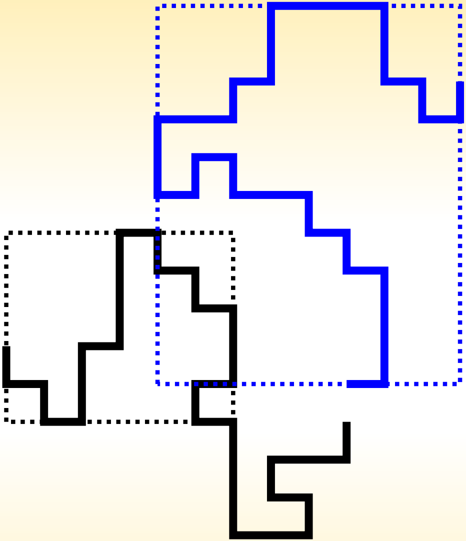


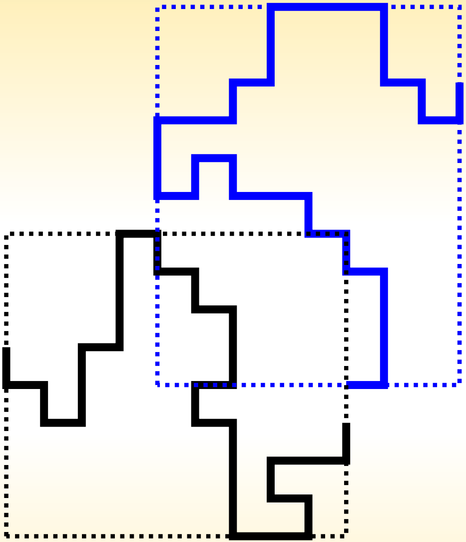


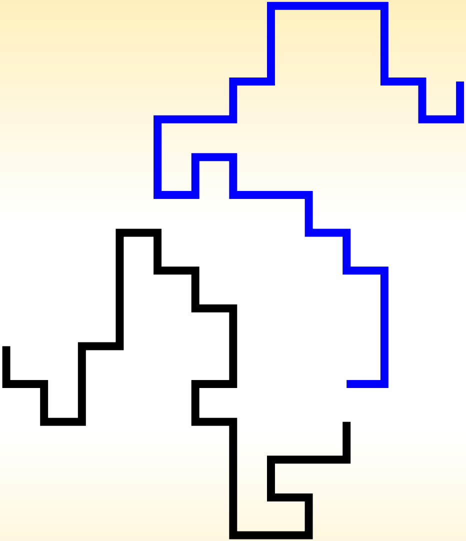












- Madras and Sokal (1988): $O(N)$ algorithm for a successful pivot.

- Madras and Sokal (1988): $O(N)$ algorithm for a successful pivot.
- Tom Kennedy (2002): broke $O(N)$ barrier by exploiting spatial separation.

- Madras and Sokal (1988): $O(N)$ algorithm for a successful pivot.
- Tom Kennedy (2002): broke $O(N)$ barrier by exploiting spatial separation.
- Until then it was thought that $O(N)$ was best possible since it takes time $O(N)$ to merely write down the walk.

- Madras and Sokal (1988): $O(N)$ algorithm for a successful pivot.
- Tom Kennedy (2002): broke $O(N)$ barrier by exploiting spatial separation.
- Until then it was thought that $O(N)$ was best possible since it takes time $O(N)$ to merely write down the walk.
- C. (2010): SAW-tree improved things further.

- Madras and Sokal (1988): $O(N)$ algorithm for a successful pivot.
- Tom Kennedy (2002): broke $O(N)$ barrier by exploiting spatial separation.
- Until then it was thought that $O(N)$ was best possible since it takes time $O(N)$ to merely write down the walk.
- C. (2010): SAW-tree improved things further.

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

Lattice	Madras and Sokal	Kennedy	SAW-tree
Square	$O(N^{0.81})$	$O(N^{0.38})$	$o(\log N)$
Cubic	$O(N^{0.89})$	$O(N^{0.74})$	$O(\log N)$

Polymer knotting

- To quantitatively understand knotting of DNA we will need to simulate extremely long polymers.

Polymer knotting

- To quantitatively understand knotting of DNA we will need to simulate extremely long polymers.
- For $N = 1$ billion, since 1985 speed-up factors of:

Polymer knotting

- To quantitatively understand knotting of DNA we will need to simulate extremely long polymers.
- For $N = 1$ billion, since 1985 speed-up factors of:
  - 10 million from computer improvements

Polymer knotting

- To quantitatively understand knotting of DNA we will need to simulate extremely long polymers.
- For $N = 1$ billion, since 1985 speed-up factors of:
  - 10 million from computer improvements
  - 100 million from invention of pivot algorithm (Madras and Sokal, 1988)

Polymer knotting

- To quantitatively understand knotting of DNA we will need to simulate extremely long polymers.
- For $N = 1$ billion, since 1985 speed-up factors of:
  - 10 million from computer improvements
  - 100 million from invention of pivot algorithm (Madras and Sokal, 1988)
  - 1 million from efficiently implementing the pivot algorithm (Kennedy in 2002, C. in 2010)

Can we apply method to other models?

- Hard problem!

Can we apply method to other models?

- Hard problem!
- Could, e.g., flip spins in special square / arbitrary rectangular regions for Ising model in time $O(\log N)$ / $O(N^{1/2})$:

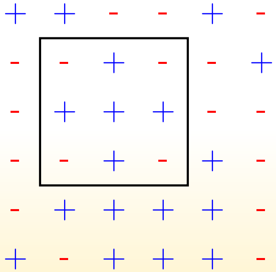
Can we apply method to other models?

- Hard problem!
- Could, e.g., flip spins in special square / arbitrary rectangular regions for Ising model in time $O(\log N)$ / $O(N^{1/2})$:

+	+	-	-	+	-
-	-	+	-	-	+
-	+	+	+	-	-
-	-	+	-	+	-
-	+	+	+	+	-
+	-	+	+	+	-

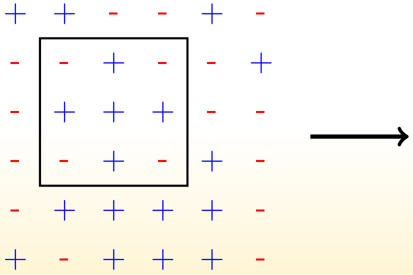
Can we apply method to other models?

- Hard problem!
- Could, e.g., flip spins in special square / arbitrary rectangular regions for Ising model in time $O(\log N)$ / $O(N^{1/2})$:



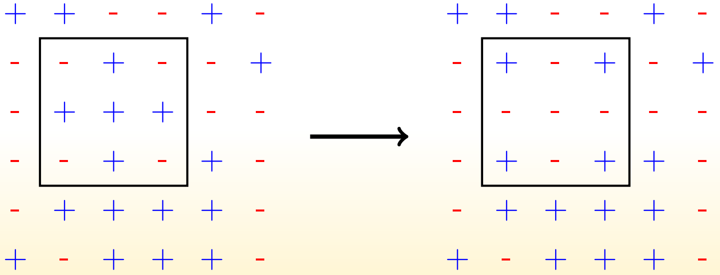
Can we apply method to other models?

- Hard problem!
- Could, e.g., flip spins in special square / arbitrary rectangular regions for Ising model in time $O(\log N)$ / $O(N^{1/2})$:



Can we apply method to other models?

- Hard problem!
- Could, e.g., flip spins in special square / arbitrary rectangular regions for Ising model in time $O(\log N)$ / $O(N^{1/2})$:

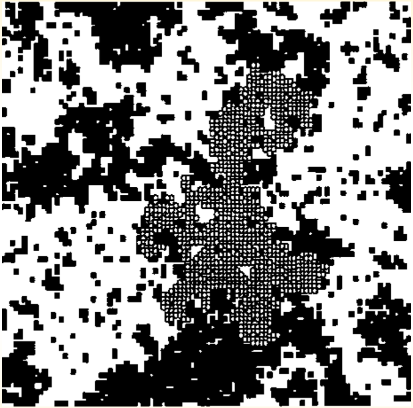


Can we apply method to other models?

- Hard problem!
- Could, e.g., flip spins in special square / arbitrary rectangular regions for Ising model in time $O(\log N)$ / $O(N^{1/2})$:

$$\begin{matrix} + & + & - & - & + & - \\ - & - & + & - & - & + \\ - & + & + & + & - & - \\ - & - & + & - & + & - \\ - & + & + & + & + & - \\ + & - & + & + & + & - \end{matrix} \quad \longrightarrow \quad \begin{matrix} + & + & - & - & + & - \\ - & + & - & + & - & + \\ - & - & - & - & - & - \\ - & + & - & + & + & - \\ - & + & + & + & + & - \\ + & - & + & + & + & - \end{matrix}$$

Whereas these are the kinds of clusters the system "wants" to flip at criticality:



(Image of a Wolff algorithm update due to Wolfhard Janke)

Other polymer models

- Could use SAW-tree for other polymer models, e.g. dense polymers, θ-polymers.

Other polymer models

- Could use SAW-tree for other polymer models, e.g. dense polymers, θ-polymers.
- Does improve things, but no algorithm as spectacularly as successful as the pivot algorithm is available.

How to calculate c_N and μ?

- Would like to apply pivot algorithm in canonical ensemble.

How to calculate c_N and μ?

- Would like to apply pivot algorithm in canonical ensemble.
- Appproach: concatenate pairs of SAW chosen uniformly at random.

How to calculate c_N and μ?

- Would like to apply pivot algorithm in canonical ensemble.
- Appproach: concatenate pairs of SAW chosen uniformly at random.
- S_N set of walks of length N.

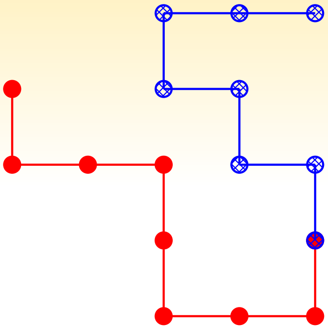
How to calculate c_N and μ?

- Would like to apply pivot algorithm in canonical ensemble.
- Appproach: concatenate pairs of SAW chosen uniformly at random.
- S_N set of walks of length N.
- $|S_{M+N}| = P(\omega_1 \circ \omega_2 \in S_{M+N} | (\omega_1, \omega_2) \in S_M \times S_N) |S_M| |S_N|$

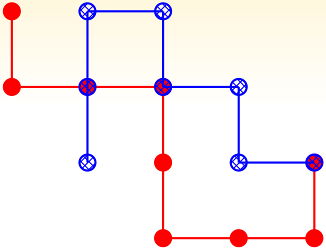
How to calculate c_N and μ?

- Would like to apply pivot algorithm in canonical ensemble.
- Appproach: concatenate pairs of SAW chosen uniformly at random.
- S_N set of walks of length N.
- $|S_{M+N}| = P(\omega_1 \circ \omega_2 \in S_{M+N} | (\omega_1, \omega_2) \in S_M \times S_N) |S_M||S_N|$
- Indicator function for successful concatenation is our observable, and

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



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



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

- Could choose $m, n = 36$ (longest known for $\mathbb{Z}^3$):

$$\langle B_{36,36} \rangle = \frac{c_{72}}{c_{36} c_{36}}$$

- Could choose $m, n = 36$ (longest known for $\mathbb{Z}^3$):

$$\langle B_{36,36} \rangle = \frac{c_{72}}{c_{36} c_{36}}$$

- Iterate to obtain estimates for c_N for longer walks.

$$\begin{aligned} c_N &= \frac{c_N}{c_{N/2}^2} \cdot \frac{c_{N/2}^2}{c_{N/4}^4} \cdots \frac{c_{2k}^{N/2k}}{c_k^{N/k}} c_k^{N/k} \\ &= \langle B_{N/2,N/2} \rangle \langle B_{N/4,N/4} \rangle^2 \cdots \langle B_{k,k} \rangle^{N/2k} c_k^{N/k} \end{aligned}$$

where c_k is known.

- Can then use $c_N \sim A\mu^N N^{\gamma-1}$ to estimate μ:

$$
\begin{aligned}
\log \mu_N &\equiv \frac{1}{N} \log c_N \\
&= \frac{1}{k} \log c_k + \frac{1}{2k} \log\langle B_{k,k}\rangle + \frac{1}{4k} \log\langle B_{2k,2k}\rangle + \cdots \\
&\qquad \cdots + \frac{1}{N} \log\langle B_{N/2,N/2}\rangle \\
&= \log \mu + \frac{(\gamma - 1)\log N}{N} + \frac{\log A}{N} + \text{corrections}
\end{aligned}
$$

- Can then use $c_N \sim A\mu^N N^{\gamma-1}$ to estimate μ:

$$\begin{aligned}\log \mu_N &\equiv \frac{1}{N}\log c_N \\ &= \frac{1}{k}\log c_k + \frac{1}{2k}\log\langle B_{k,k}\rangle + \frac{1}{4k}\log\langle B_{2k,2k}\rangle + \cdots \\ &\qquad \cdots + \frac{1}{N}\log\langle B_{N/2,N/2}\rangle \\ &= \log\mu + \frac{(\gamma-1)\log N}{N} + \frac{\log A}{N} + \text{corrections}\end{aligned}$$

- Corrections vanish with increasing N! In limit of large N systematic error of estimator $\to 0$.

Results for c_N and μ on $\mathbb{Z}^3$

- Computer experiment of 60000 CPU hours.

Results for c_N and μ on $\mathbb{Z}^3$

- Computer experiment of 60000 CPU hours.
- $c_{9471} = 1.43323(8) \times 10^{6352}$.

Results for c_N and μ on $\mathbb{Z}^3$

- Computer experiment of 60000 CPU hours.
- $c_{9471} = 1.43323(8) \times 10^{6352}$.
- $c_{38797311} = 7 \times 10^{26018276}$, $(6.6, 8.2)$.

Results for c_N and μ on $\mathbb{Z}^3$

- Computer experiment of 60000 CPU hours.
- $c_{9471} = 1.43323(8) \times 10^{6352}$.
- $c_{38797311} = 7 \times 10^{26018276}$, $(6.6, 8.2)$.
- $\mu = 4.684039931(27)$

Results for c_N and μ on $\mathbb{Z}^3$

- Computer experiment of 60000 CPU hours.
- $c_{9471} = 1.43323(8) \times 10^{6352}$.
- $c_{38797311} = 7 \times 10^{26018276}$, $(6.6, 8.2)$.
- $\mu = 4.684039931(27)$
- best previous estimate via "PERM" algorithm: $\mu = 4.6840386(11)$ (Grassberger, 2005).

Conclusion

- Newly developed algorithms have radically improved our ability to sample SAW.

Conclusion

- Newly developed algorithms have radically improved our ability to sample SAW.
- Asymptotically faster, large N limit now accessible.

Conclusion

- Newly developed algorithms have radically improved our ability to sample SAW.
- Asymptotically faster, large N limit now accessible.
- Extend to dense polymers, off-lattice walks, θ-polymers.

Conclusion

- Newly developed algorithms have radically improved our ability to sample SAW.
- Asymptotically faster, large N limit now accessible.
- Extend to dense polymers, off-lattice walks, θ-polymers.
- How many of these ideas can be transferred to sampling other combinatorial objects?

Conclusion

- Newly developed algorithms have radically improved our ability to sample SAW.
- Asymptotically faster, large N limit now accessible.
- Extend to dense polymers, off-lattice walks, θ-polymers.
- How many of these ideas can be transferred to sampling other combinatorial objects?
- Linear order and spatial separation seems to make SAW a special case.

Conclusion

- Newly developed algorithms have radically improved our ability to sample SAW.
- Asymptotically faster, large N limit now accessible.
- Extend to dense polymers, off-lattice walks, θ-polymers.
- How many of these ideas can be transferred to sampling other combinatorial objects?
- Linear order and spatial separation seems to make SAW a special case.
- Challenge: find efficient algorithms and computer implementations for other systems. (Very active research area, e.g. cluster algorithms, Wang-Landau method, PERM, worm algorithms.)