There are $7 \times 10^{26\,018\,276}$ self-avoiding walks of 38 797 311 steps on $\mathbb{Z}^3$

Nathan Clisby
MASCOS, The University of Melbourne

AustMS Annual Meeting
The University of Ballarat
September 25, 2012

Outline

- Self-avoiding walks
- Enumeration
- Direct sampling and weighted sampling (PERM)
- Ingredients for efficiently estimating c_N
  - Global move (pivot)
  - Efficient data structure (SAW-tree)
  - Clever choice of observable
  - Minimizing statistical error
- Results and conclusion

Outline

- Self-avoiding walks
- Enumeration
- Direct sampling and weighted sampling (PERM)
- Ingredients for efficiently estimating c_N
  - Global move (pivot)
  - Efficient data structure (SAW-tree)
  - Clever choice of observable
  - Minimizing statistical error
- Results and conclusion

Outline

- Self-avoiding walks
- Enumeration
- Direct sampling and weighted sampling (PERM)
- Ingredients for efficiently estimating c_N
  - Global move (pivot)
  - Efficient data structure (SAW-tree)
  - Clever choice of observable
  - Minimizing statistical error
- Results and conclusion

Outline

- Self-avoiding walks
- Enumeration
- Direct sampling and weighted sampling (PERM)
- Ingredients for efficiently estimating c_N
  - Global move (pivot)
  - Efficient data structure (SAW-tree)
  - Clever choice of observable
  - Minimizing statistical error
- Results and conclusion

Outline

- Self-avoiding walks
- Enumeration
- Direct sampling and weighted sampling (PERM)
- Ingredients for efficiently estimating c_N
  - Global move (pivot)
  - Efficient data structure (SAW-tree)
  - Clever choice of observable
  - Minimizing statistical error
- Results and conclusion

Outline

- Self-avoiding walks
- Enumeration
- Direct sampling and weighted sampling (PERM)
- Ingredients for efficiently estimating c_N
  - Global move (pivot)
  - Efficient data structure (SAW-tree)
  - Clever choice of observable
  - Minimizing statistical error
- Results and conclusion

Outline

- Self-avoiding walks
- Enumeration
- Direct sampling and weighted sampling (PERM)
- Ingredients for efficiently estimating c_N
  - Global move (pivot)
  - Efficient data structure (SAW-tree)
  - Clever choice of observable
  - Minimizing statistical error
- Results and conclusion

Outline

- Self-avoiding walks
- Enumeration
- Direct sampling and weighted sampling (PERM)
- Ingredients for efficiently estimating c_N
  - Global move (pivot)
  - Efficient data structure (SAW-tree)
  - Clever choice of observable
  - Minimizing statistical error
- Results and conclusion

Outline

- Self-avoiding walks
- Enumeration
- Direct sampling and weighted sampling (PERM)
- Ingredients for efficiently estimating c_N
    - Global move (pivot)
    - Efficient data structure (SAW-tree)
    - Clever choice of observable
    - Minimizing statistical error
- Results and conclusion

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.
- N-step SAW on $\mathbb{Z}^d$ is a mapping $\omega : \{0, 1, \ldots, N\} \to \mathbb{Z}^d$ with $|\omega(i+1) - \omega(i)| = 1$ for each i ($|x|$ denotes the Euclidean norm of x), and with $\omega(i) \neq \omega(j)$ for all $i \neq j$.
- For uniqueness, choose $\omega(0) = 0$.

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.
- N-step SAW on $\mathbb{Z}^d$ is a mapping $\omega : \{0, 1, \ldots, N\} \to \mathbb{Z}^d$ with $|\omega(i+1) - \omega(i)| = 1$ for each i ($|x|$ denotes the Euclidean norm of x), and with $\omega(i) \neq \omega(j)$ for all $i \neq j$.
- For uniqueness, choose $\omega(0) = 0$.

Self-avoiding walk model

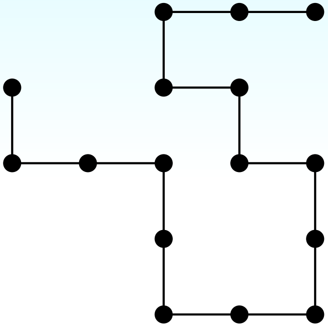
- A walk on a lattice, step to neighbouring site provided it has not already been visited.
- Models polymers in good solvent limit.
- Exactly captures universal properties such as critical exponents.
- N-step SAW on $\mathbb{Z}^d$ is a mapping $\omega : \{0, 1, \ldots, N\} \to \mathbb{Z}^d$ with $|\omega(i+1) - \omega(i)| = 1$ for each i ($|x|$ denotes the Euclidean norm of x), and with $\omega(i) \neq \omega(j)$ for all $i \neq j$.
- For uniqueness, choose $\omega(0) = 0$.

Self-avoiding walk model

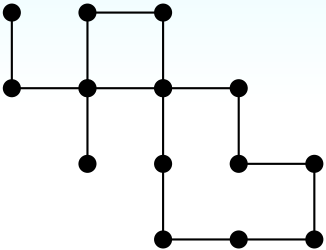
- A walk on a lattice, step to neighbouring site provided it has not already been visited.
- Models polymers in good solvent limit.
- Exactly captures universal properties such as critical exponents.
- N-step SAW on $\mathbb{Z}^d$ is a mapping $\omega : \{0, 1, \ldots, N\} \to \mathbb{Z}^d$ with $|\omega(i+1) - \omega(i)| = 1$ for each i ($|x|$ denotes the Euclidean norm of x), and with $\omega(i) \neq \omega(j)$ for all $i \neq j$.
- For uniqueness, choose $\omega(0) = 0$.

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.
- N-step SAW on $\mathbb{Z}^d$ is a mapping $\omega : \{0, 1, \ldots, N\} \to \mathbb{Z}^d$ with $|\omega(i+1) - \omega(i)| = 1$ for each i ($|x|$ denotes the Euclidean norm of x), and with $\omega(i) \neq \omega(j)$ for all $i \neq j$.
- For uniqueness, choose $\omega(0) = 0$.



SAW



Not a SAW

Asymptotic behaviour

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

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

- For $\mathbb{Z}^2$, $c_N = 1, 4, 12, 36, 100, 284, 780, 2172, \cdots$
- For $\mathbb{Z}^3$, $c_N = 1, 6, 30, 150, 726, 3534, \cdots$
- γ is a *universal* exponent.
- μ is the connective constant; lattice dependent.
- (Also interested in the mean size of a SAW)

Asymptotic behaviour

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

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

- For $\mathbb{Z}^2$, $c_N = 1, 4, 12, 36, 100, 284, 780, 2172, \cdots$
- For $\mathbb{Z}^3$, $c_N = 1, 6, 30, 150, 726, 3534, \cdots$
- γ is a *universal* exponent.
- μ is the connective constant; lattice dependent.
- (Also interested in the mean size of a SAW)

Asymptotic behaviour

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

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

- For $\mathbb{Z}^2$, $c_N = 1, 4, 12, 36, 100, 284, 780, 2172, \cdots$
- For $\mathbb{Z}^3$, $c_N = 1, 6, 30, 150, 726, 3534, \cdots$
- γ is a *universal* exponent.
- μ is the connective constant; lattice dependent.
- (Also interested in the mean size of a SAW)

Asymptotic behaviour

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

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

- For $\mathbb{Z}^2$, $c_N = 1, 4, 12, 36, 100, 284, 780, 2172, \cdots$
- For $\mathbb{Z}^3$, $c_N = 1, 6, 30, 150, 726, 3534, \cdots$
- γ is a *universal* exponent.
- μ is the connective constant; lattice dependent.
- (Also interested in the mean size of a SAW)

Asymptotic behaviour

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

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

- For $\mathbb{Z}^2$, $c_N = 1, 4, 12, 36, 100, 284, 780, 2172, \cdots$
- For $\mathbb{Z}^3$, $c_N = 1, 6, 30, 150, 726, 3534, \cdots$
- γ is a *universal* exponent.
- μ is the connective constant; lattice dependent.
- (Also interested in the mean size of a SAW)

Asymptotic behaviour

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

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

- For $\mathbb{Z}^2$, $c_N = 1, 4, 12, 36, 100, 284, 780, 2172, \cdots$
- For $\mathbb{Z}^3$, $c_N = 1, 6, 30, 150, 726, 3534, \cdots$
- γ is a *universal* exponent.
- μ is the connective constant; lattice dependent.
- (Also interested in the mean size of a SAW)

- Long history, has been studied by physicists and mathematicians for 60 years.
- Rich and active research area (more than 1800 articles in Web of Science with SAW in title / abstract).
- Hard! No immediate prospect of exact solution, although recent progress with exact results for $d = 2$.
- Has driven development of advanced algorithms for enumeration and Monte Carlo simulation.

- Long history, has been studied by physicists and mathematicians for 60 years.
- Rich and active research area (more than 1800 articles in Web of Science with SAW in title / abstract).
- Hard! No immediate prospect of exact solution, although recent progress with exact results for $d = 2$.
- Has driven development of advanced algorithms for enumeration and Monte Carlo simulation.

- Long history, has been studied by physicists and mathematicians for 60 years.
- Rich and active research area (more than 1800 articles in Web of Science with SAW in title / abstract).
- Hard! No immediate prospect of exact solution, although recent progress with exact results for $d = 2$.
- Has driven development of advanced algorithms for enumeration and Monte Carlo simulation.

- Long history, has been studied by physicists and mathematicians for 60 years.
- Rich and active research area (more than 1800 articles in Web of Science with SAW in title / abstract).
- Hard! No immediate prospect of exact solution, although recent progress with exact results for $d = 2$.
- Has driven development of advanced algorithms for enumeration and Monte Carlo simulation.

- Direct enumeration does not get far, $c_N \sim \mu^N$ with $\mu \approx 2.64$ for $\mathbb{Z}^2$ and $\mu \approx 4.68$ for $\mathbb{Z}^3$.
- So, transform problem and count something else.
- For 2d lattices: finite lattice method extremely powerful. Count boundary states instead of walks, $O(1.3^n)$ (unfortunately, still exponential). Recently, Iwan Jensen found $c_{79} = 10194710293557466193787900071923676$ for $\mathbb{Z}^2$!
- For 3d lattices: most powerful method "length-doubling algorithm", combines brute force enumeration with inclusion-exclusion. $O(\mu^n) \to O((\sqrt{2\mu})^n)$[1].
- I think there are strong prospects to apply length-doubling algorithm to other problems, and improve its efficiency.
- $c_{36} = 2941370856334701726560670$ for $\mathbb{Z}^3$.
- Series analysis used to extract information about asymptotic behaviour.

[1] R. Schram et al., J. Stat. Mech.: Theor. Exp., P060109 (2011)

- Direct enumeration does not get far, $c_N \sim \mu^N$ with $\mu \approx 2.64$ for $\mathbb{Z}^2$ and $\mu \approx 4.68$ for $\mathbb{Z}^3$.
- So, transform problem and count something else.
- For 2d lattices: finite lattice method extremely powerful. Count boundary states instead of walks, $O(1.3^n)$ (unfortunately, still exponential). Recently, Iwan Jensen found $c_{79} = 10194710293557466193787900071923676$ for $\mathbb{Z}^2$!
- For 3d lattices: most powerful method "length-doubling algorithm", combines brute force enumeration with inclusion-exclusion. $O(\mu^n) \to O((\sqrt{2\mu})^n)$[1].
- I think there are strong prospects to apply length-doubling algorithm to other problems, and improve its efficiency.
- $c_{36} = 2941370856334701726560670$ for $\mathbb{Z}^3$.
- Series analysis used to extract information about asymptotic behaviour.

[1] R. Schram et al., J. Stat. Mech.: Theor. Exp., P060109 (2011)

- Direct enumeration does not get far, $c_N \sim \mu^N$ with $\mu \approx 2.64$ for $\mathbb{Z}^2$ and $\mu \approx 4.68$ for $\mathbb{Z}^3$.
- So, transform problem and count something else.
- For 2d lattices: finite lattice method extremely powerful. Count boundary states instead of walks, $O(1.3^n)$ (unfortunately, still exponential). Recently, Iwan Jensen found $c_{79} = 10194710293557466193787900071923676$ for $\mathbb{Z}^2$!
- For 3d lattices: most powerful method “length-doubling algorithm”, combines brute force enumeration with inclusion-exclusion. $O(\mu^n) \to O((\sqrt{2\mu})^n)$[1].
- I think there are strong prospects to apply length-doubling algorithm to other problems, and improve its efficiency.
- $c_{36} = 2941370856334701726560670$ for $\mathbb{Z}^3$.
- Series analysis used to extract information about asymptotic behaviour.

[1] R. Schram et al., J. Stat. Mech.: Theor. Exp., P060109 (2011)

- Direct enumeration does not get far, $c_N \sim \mu^N$ with $\mu \approx 2.64$ for $\mathbb{Z}^2$ and $\mu \approx 4.68$ for $\mathbb{Z}^3$.
- So, transform problem and count something else.
- For 2d lattices: finite lattice method extremely powerful. Count boundary states instead of walks, $O(1.3^n)$ (unfortunately, still exponential). Recently, Iwan Jensen found $c_{79} = 10194710293557466193787900071923676$ for $\mathbb{Z}^2$!
- For 3d lattices: most powerful method "length-doubling algorithm", combines brute force enumeration with inclusion-exclusion. $O(\mu^n) \rightarrow O((\sqrt{2\mu})^n)$[1].
- I think there are strong prospects to apply length-doubling algorithm to other problems, and improve its efficiency.
- $c_{36} = 2941370856334701726560670$ for $\mathbb{Z}^3$.
- Series analysis used to extract information about asymptotic behaviour.

[1] R. Schram et al., J. Stat. Mech.: Theor. Exp., P060109 (2011)

- Direct enumeration does not get far, $c_N \sim \mu^N$ with $\mu \approx 2.64$ for $\mathbb{Z}^2$ and $\mu \approx 4.68$ for $\mathbb{Z}^3$.
- So, transform problem and count something else.
- For 2d lattices: finite lattice method extremely powerful. Count boundary states instead of walks, $O(1.3^n)$ (unfortunately, still exponential). Recently, Iwan Jensen found $c_{79} = 10194710293557466193787900071923676$ for $\mathbb{Z}^2$!
- For 3d lattices: most powerful method "length-doubling algorithm", combines brute force enumeration with inclusion-exclusion. $O(\mu^n) \to O((\sqrt{2\mu})^n)$[1].
- I think there are strong prospects to apply length-doubling algorithm to other problems, and improve its efficiency.
- $c_{36} = 2941370856334701726560670$ for $\mathbb{Z}^3$.
- Series analysis used to extract information about asymptotic behaviour.

[1] R. Schram et al., J. Stat. Mech.: Theor. Exp., P060109 (2011)

- Direct enumeration does not get far, $c_N \sim \mu^N$ with $\mu \approx 2.64$ for $\mathbb{Z}^2$ and $\mu \approx 4.68$ for $\mathbb{Z}^3$.
- So, transform problem and count something else.
- For 2d lattices: finite lattice method extremely powerful. Count boundary states instead of walks, $O(1.3^n)$ (unfortunately, still exponential). Recently, Iwan Jensen found $c_{79} = 10194710293557466193787900071923676$ for $\mathbb{Z}^2$!
- For 3d lattices: most powerful method "length-doubling algorithm", combines brute force enumeration with inclusion-exclusion. $O(\mu^n) \to O((\sqrt{2\mu})^n)$[1].
- I think there are strong prospects to apply length-doubling algorithm to other problems, and improve its efficiency.
- $c_{36} = 2941370856334701726560670$ for $\mathbb{Z}^3$.
- Series analysis used to extract information about asymptotic behaviour.

[1] R. Schram et al., J. Stat. Mech.: Theor. Exp., P060109 (2011)

- Direct enumeration does not get far, $c_N \sim \mu^N$ with $\mu \approx 2.64$ for $\mathbb{Z}^2$ and $\mu \approx 4.68$ for $\mathbb{Z}^3$.
- So, transform problem and count something else.
- For 2d lattices: finite lattice method extremely powerful. Count boundary states instead of walks, $O(1.3^n)$ (unfortunately, still exponential). Recently, Iwan Jensen found $c_{79} = 10194710293557466193787900071923676$ for $\mathbb{Z}^2$!
- For 3d lattices: most powerful method "length-doubling algorithm", combines brute force enumeration with inclusion-exclusion. $O(\mu^n) \to O((\sqrt{2\mu})^n)$[1].
- I think there are strong prospects to apply length-doubling algorithm to other problems, and improve its efficiency.
- $c_{36} = 2941370856334701726560670$ for $\mathbb{Z}^3$.
- Series analysis used to extract information about asymptotic behaviour.

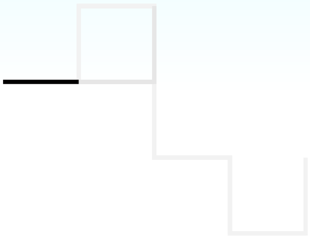
[1] R. Schram et al., J. Stat. Mech.: Theor. Exp., P060109 (2011)

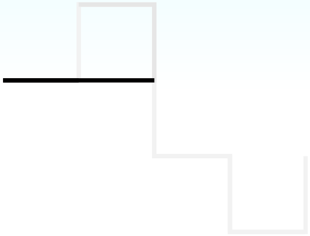
- Wish to estimate c_N beyond limits accessible to exact enumeration.
- Obvious approach: simple sampling. Generate a simple random walk of length N, calculate probability that RW is self-avoiding. Probability $= c_N/(2d)^N \approx 4.68^N/6^N$ for $\mathbb{Z}^3$.
- Can improve slightly: forbid immediate reversals in the walk. Probability $= c_n/2d/(2d-1)^{N-1} \approx 4.68^N/6/5^{N-1}$ for $\mathbb{Z}^3$.
- For $N = 100$ only 1 in 1000 random walks with no immediate reversals is a SAW. Cannot push this much further.

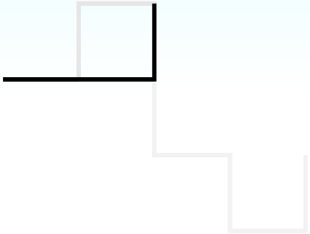
- Wish to estimate c_N beyond limits accessible to exact enumeration.
- Obvious approach: simple sampling. Generate a simple random walk of length N, calculate probability that RW is self-avoiding. Probability $= c_N/(2d)^N \approx 4.68^N/6^N$ for $\mathbb{Z}^3$.
- Can improve slightly: forbid immediate reversals in the walk. Probability $= c_n/2d/(2d-1)^{N-1} \approx 4.68^N/6/5^{N-1}$ for $\mathbb{Z}^3$.
- For $N = 100$ only 1 in 1000 random walks with no immediate reversals is a SAW. Cannot push this much further.

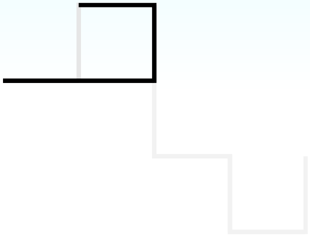
- Wish to estimate c_N beyond limits accessible to exact enumeration.
- Obvious approach: simple sampling. Generate a simple random walk of length N, calculate probability that RW is self-avoiding. Probability $= c_N/(2d)^N \approx 4.68^N/6^N$ for $\mathbb{Z}^3$.
- Can improve slightly: forbid immediate reversals in the walk. Probability $= c_n/2d/(2d-1)^{N-1} \approx 4.68^N/6/5^{N-1}$ for $\mathbb{Z}^3$.
- For $N = 100$ only 1 in 1000 random walks with no immediate reversals is a SAW. Cannot push this much further.

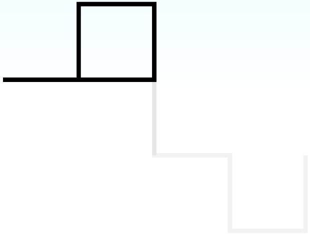
- Wish to estimate c_N beyond limits accessible to exact enumeration.
- Obvious approach: simple sampling. Generate a simple random walk of length N, calculate probability that RW is self-avoiding. Probability $= c_N/(2d)^N \approx 4.68^N/6^N$ for $\mathbb{Z}^3$.
- Can improve slightly: forbid immediate reversals in the walk. Probability $= c_n/2d/(2d-1)^{N-1} \approx 4.68^N/6/5^{N-1}$ for $\mathbb{Z}^3$.
- For $N = 100$ only 1 in 1000 random walks with no immediate reversals is a SAW. Cannot push this much further.

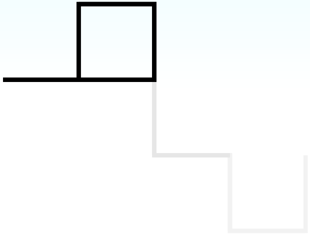


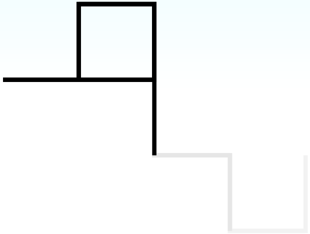


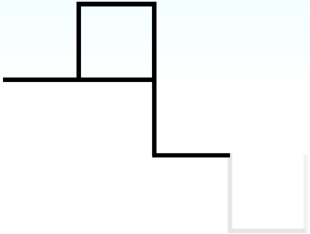


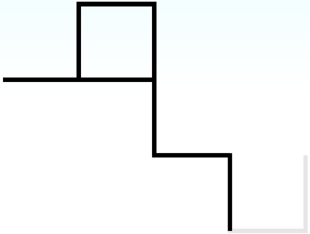


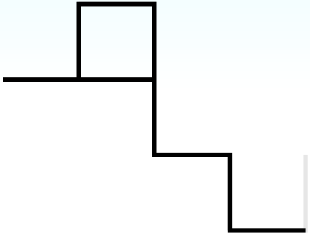


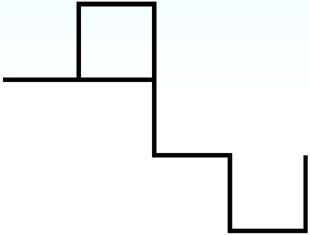












- Rosenbluth sampling: only choose free edges.
- This introduces bias: compact walks which have few choices available are preferred.
- Correct bias by weighting walks.
- Weights provide an estimator of c_N, $c_N = \langle W_N \rangle$.
- Two issues:
  - High variance (poor estimator of c_N)
  - Attrition still occurs, since walks can become trapped. Can't sample truly long walks (ok up to N of the order of hundreds)

- Rosenbluth sampling: only choose free edges.
- This introduces bias: compact walks which have few choices available are preferred.
- Correct bias by weighting walks.
- Weights provide an estimator of c_N, $c_N = \langle W_N \rangle$.
- Two issues:
  - High variance (poor estimator of c_N)
  - Attrition still occurs, since walks can become trapped. Can't sample truly long walks (ok up to N of the order of hundreds)

- Rosenbluth sampling: only choose free edges.
- This introduces bias: compact walks which have few choices available are preferred.
- Correct bias by weighting walks.
- Weights provide an estimator of c_N, $c_N = \langle W_N \rangle$.
- Two issues:
  - High variance (poor estimator of c_N)
  - Attrition still occurs, since walks can become trapped. Can't sample truly long walks (ok up to N of the order of hundreds).

- Rosenbluth sampling: only choose free edges.
- This introduces bias: compact walks which have few choices available are preferred.
- Correct bias by weighting walks.
- Weights provide an estimator of c_N, $c_N = \langle W_N \rangle$.
- Two issues:
  - High variance (poor estimator of c_N)
  - Attrition still occurs, since walks can become trapped: hard to sample truly long walks (ok up to N of the order of hundreds)

- Rosenbluth sampling: only choose free edges.
- This introduces bias: compact walks which have few choices available are preferred.
- Correct bias by weighting walks.
- Weights provide an estimator of c_N, $c_N = \langle W_N \rangle$.
- Two issues:
  - High variance (poor estimator of c_N)
  - Attrition still occurs, since walks can become trapped. Can't sample truly long walks (ok up to N of the order of hundreds).

- Rosenbluth sampling: only choose free edges.
- This introduces bias: compact walks which have few choices available are preferred.
- Correct bias by weighting walks.
- Weights provide an estimator of c_N, $c_N = \langle W_N \rangle$.
- Two issues:
  - High variance (poor estimator of c_N)
  - Attrition still occurs, since walks can become trapped. Can't sample truly long walks (ok up to N of the order of hundreds).

- Rosenbluth sampling: only choose free edges.
- This introduces bias: compact walks which have few choices available are preferred.
- Correct bias by weighting walks.
- Weights provide an estimator of c_N, $c_N = \langle W_N \rangle$.
- Two issues:
  - High variance (poor estimator of c_N)
  - Attrition still occurs, since walks can become trapped. Can't sample truly long walks (ok up to N of the order of hundreds).

- PERM: Pruned Enriched Rosenbluth Sampling, a variant of sequential importance sampling.
- Prune: low weight walks, either discard with $P = 0.5$ or double weight.
- Enrich: high weight walks, make copies, ensure total weight remains the same.
- PERM: sensible choices for enrichment ensure attrition is eliminated, variance reduced.
- Dramatically better than Rosenbluth sampling, arbitrarily large N achievable.
- Sophisticated choices for pruning and enrichment algorithms can reduce correlations and variance.

- PERM: Pruned Enriched Rosenbluth Sampling, a variant of sequential importance sampling.
- Prune: low weight walks, either discard with $P = 0.5$ or double weight.
- Enrich: high weight walks, make copies, ensure total weight remains the same.
- PERM: sensible choices for enrichment ensure attrition is eliminated, variance reduced.
- Dramatically better than Rosenbluth sampling, arbitrarily large N achievable.
- Sophisticated choices for pruning and enrichment algorithms can reduce correlations and variance.

- PERM: Pruned Enriched Rosenbluth Sampling, a variant of sequential importance sampling.
- Prune: low weight walks, either discard with $P = 0.5$ or double weight.
- Enrich: high weight walks, make copies, ensure total weight remains the same.
- PERM: sensible choices for enrichment ensure attrition is eliminated, variance reduced.
- Dramatically better than Rosenbluth sampling, arbitrarily large N achievable.
- Sophisticated choices for pruning and enrichment algorithms can reduce correlations and variance.

- PERM: Pruned Enriched Rosenbluth Sampling, a variant of sequential importance sampling.
- Prune: low weight walks, either discard with $P = 0.5$ or double weight.
- Enrich: high weight walks, make copies, ensure total weight remains the same.
- PERM: sensible choices for enrichment ensure attrition is eliminated, variance reduced.
- Dramatically better than Rosenbluth sampling, arbitrarily large N achievable.
- Sophisticated choices for pruning and enrichment algorithms can reduce correlations and variance.

- PERM: Pruned Enriched Rosenbluth Sampling, a variant of sequential importance sampling.
- Prune: low weight walks, either discard with $P = 0.5$ or double weight.
- Enrich: high weight walks, make copies, ensure total weight remains the same.
- PERM: sensible choices for enrichment ensure attrition is eliminated, variance reduced.
- Dramatically better than Rosenbluth sampling, arbitrarily large N achievable.
- Sophisticated choices for pruning and enrichment algorithms can reduce correlations and variance.

- PERM: Pruned Enriched Rosenbluth Sampling, a variant of sequential importance sampling.
- Prune: low weight walks, either discard with $P = 0.5$ or double weight.
- Enrich: high weight walks, make copies, ensure total weight remains the same.
- PERM: sensible choices for enrichment ensure attrition is eliminated, variance reduced.
- Dramatically better than Rosenbluth sampling, arbitrarily large N achievable.
- Sophisticated choices for pruning and enrichment algorithms can reduce correlations and variance.

Factors limiting the efficiency of PERM.

- Correlations introduced by enrichment.
- Variance of sample is reduced, but not eliminated. (In practice, variance can be essentially eliminated, at the expense of stronger correlation.)
- Intrinsic limit: CPU time $O(N)$ to produce a single walk. (Prohibitive for truly large N).
- Will now describe a method that overcomes each of these deficiencies.

Factors limiting the efficiency of PERM.

- Correlations introduced by enrichment.
- Variance of sample is reduced, but not eliminated. (In practice, variance can be essentially eliminated, at the expense of stronger correlation.)
- Intrinsic limit: CPU time $O(N)$ to produce a single walk. (Prohibitive for truly large N).
- Will now describe a method that overcomes each of these deficiencies.

Factors limiting the efficiency of PERM.

- Correlations introduced by enrichment.
- Variance of sample is reduced, but not eliminated. (In practice, variance can be essentially eliminated, at the expense of stronger correlation.)
- Intrinsic limit: CPU time $O(N)$ to produce a single walk. (Prohibitive for truly large N).
- Will now describe a method that overcomes each of these deficiencies.

Factors limiting the efficiency of PERM.

- Correlations introduced by enrichment.
- Variance of sample is reduced, but not eliminated. (In practice, variance can be essentially eliminated, at the expense of stronger correlation.)
- Intrinsic limit: CPU time $O(N)$ to produce a single walk. (Prohibitive for truly large N).
- Will now describe a method that overcomes each of these deficiencies.

To calculate c_N efficiently we need to

- Utilise most efficient sampling method, rapidly move around state space.
- Utilise efficient data structures.
- Find a suitable observable, with low variance.
- Design computer experiment to minimise statistical error.
- Will see that working with *fixed length* walks confers dramatic advantage over growth algorithms.

To calculate c_N efficiently we need to

- Utilise most efficient sampling method, rapidly move around state space.
- Utilise efficient data structures.
- Find a suitable observable, with low variance.
- Design computer experiment to minimise statistical error.
- Will see that working with *fixed length* walks confers dramatic advantage over growth algorithms.

To calculate c_N efficiently we need to

- Utilise most efficient sampling method, rapidly move around state space.
- Utilise efficient data structures.
- Find a suitable observable, with low variance.
- Design computer experiment to minimise statistical error.
- Will see that working with *fixed length* walks confers dramatic advantage over growth algorithms.

To calculate c_N efficiently we need to

- Utilise most efficient sampling method, rapidly move around state space.
- Utilise efficient data structures.
- Find a suitable observable, with low variance.
- Design computer experiment to minimise statistical error.
- Will see that working with *fixed length* walks confers dramatic advantage over growth algorithms.

To calculate c_N efficiently we need to

- Utilise most efficient sampling method, rapidly move around state space.
- Utilise efficient data structures.
- Find a suitable observable, with low variance.
- Design computer experiment to minimise statistical error.
- Will see that working with *fixed length* walks confers dramatic advantage over growth algorithms.

Pivot algorithm

- Sample from the set of SAWs of a particular length.
- Markov chain:
  - Select a pivot site *uniformly at random.*
  - Randomly choose a lattice symmetry g (rotation or reflection).
  - Apply this symmetry to one of the two sub-walks created by splitting the walk at the pivot site.
  - If walk is self-avoiding: *accept* the pivot and update the configuration.
  - If walk is not self-avoiding: *reject* the pivot and keep the old configuration.
- Ergodic, samples SAWs uniformly at random.

Pivot algorithm

- Sample from the set of SAWs of a particular length.
- Markov chain:
  - Select a pivot site *uniformly at random*.
  - Randomly choose a lattice symmetry q (rotation or reflection).
  - Apply this symmetry to one of the two sub-walks created by splitting the walk at the pivot site.
  - If walk is self-avoiding: *accept* the pivot and update the configuration.
  - If walk is not self-avoiding: *reject* the pivot and keep the old configuration.
- Ergodic, samples SAWs uniformly at random.

Pivot algorithm

- Sample from the set of SAWs of a particular length.
- Markov chain:
  - Select a pivot site *uniformly at random*.
  - Randomly choose a lattice symmetry q (rotation or reflection).
  - Apply this symmetry to one of the two sub-walks created by splitting the walk at the pivot site.
  - If walk is self-avoiding: *accept* the pivot and update the configuration.
  - If walk is not self-avoiding: *reject* the pivot and keep the old configuration.
- Ergodic, samples SAWs uniformly at random.

Pivot algorithm

- Sample from the set of SAWs of a particular length.
- Markov chain:
  - Select a pivot site *uniformly at random*.
  - Randomly choose a lattice symmetry q (rotation or reflection).
  - Apply this symmetry to one of the two sub-walks created by splitting the walk at the pivot site.
  - If walk is self-avoiding: *accept* the pivot and update the configuration.
  - If walk is not self-avoiding: *reject* the pivot and keep the old configuration.
- Ergodic, samples SAWs uniformly at random.

Pivot algorithm

- Sample from the set of SAWs of a particular length.
- Markov chain:
  - Select a pivot site *uniformly at random*.
  - Randomly choose a lattice symmetry q (rotation or reflection).
  - Apply this symmetry to one of the two sub-walks created by splitting the walk at the pivot site.
  - If walk is self-avoiding: *accept* the pivot and update the configuration.
  - If walk is not self-avoiding: *reject* the pivot and keep the old configuration.
- Ergodic, samples SAWs uniformly at random.

Pivot algorithm

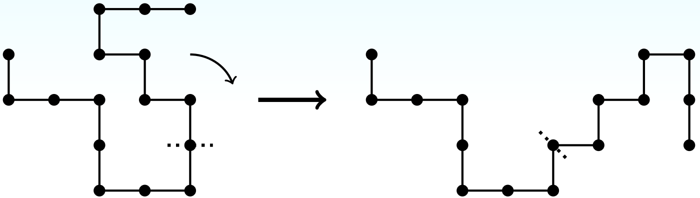
- Sample from the set of SAWs of a particular length.
- Markov chain:
  - Select a pivot site *uniformly at random*.
  - Randomly choose a lattice symmetry q (rotation or reflection).
  - Apply this symmetry to one of the two sub-walks created by splitting the walk at the pivot site.
  - If walk is self-avoiding: *accept* the pivot and update the configuration.
  - If walk is not self-avoiding: *reject* the pivot and keep the old configuration.
- Ergodic, samples SAWs uniformly at random.

Pivot algorithm

- Sample from the set of SAWs of a particular length.
- Markov chain:
  - Select a pivot site *uniformly at random*.
  - Randomly choose a lattice symmetry q (rotation or reflection).
  - Apply this symmetry to one of the two sub-walks created by splitting the walk at the pivot site.
  - If walk is self-avoiding: *accept* the pivot and update the configuration.
  - If walk is not self-avoiding: *reject* the pivot and keep the old configuration.
- Ergodic, samples SAWs uniformly at random.

Pivot algorithm

- Sample from the set of SAWs of a particular length.
- Markov chain:
  - Select a pivot site *uniformly at random*.
  - Randomly choose a lattice symmetry q (rotation or reflection).
  - Apply this symmetry to one of the two sub-walks created by splitting the walk at the pivot site.
  - If walk is self-avoiding: *accept* the pivot and update the configuration.
  - If walk is not self-avoiding: *reject* the pivot and keep the old configuration.
- Ergodic, samples SAWs uniformly at random.



Example pivot move

Why is it so effective?

- Pivots are rarely successful, $\Pr = O(N^{-p})$, $p \approx 0.11$ for $\mathbb{Z}^3$.
- Every time a pivot attempt *is* successful there is a large change in global observables.
- Only need $O(1)$ successful pivots before we have an *essentially new* configuration with respect to observables measuring size.
- $\Rightarrow \tau_{\mathrm{int}} = O(N^p)$.

Why is it so effective?

- Pivots are rarely successful, $\Pr = O(N^{-p})$, $p \approx 0.11$ for $\mathbb{Z}^3$.
- Every time a pivot attempt *is* successful there is a large change in global observables.
- Only need $O(1)$ successful pivots before we have an *essentially new* configuration with respect to observables measuring size.
- $\Rightarrow \tau_{\mathrm{int}} = O(N^p)$.

Why is it so effective?

- Pivots are rarely successful, $\Pr = O(N^{-p})$, $p \approx 0.11$ for $\mathbb{Z}^3$.
- Every time a pivot attempt *is* successful there is a large change in global observables.
- Only need $O(1)$ successful pivots before we have an *essentially new* configuration with respect to observables measuring size.
- $\Rightarrow \tau_{\mathrm{int}} = O(N^p)$.

Why is it so effective?

- Pivots are rarely successful, $\Pr = O(N^{-p})$, $p \approx 0.11$ for $\mathbb{Z}^3$.
- Every time a pivot attempt *is* successful there is a large change in global observables.
- Only need $O(1)$ successful pivots before we have an *essentially new* configuration with respect to observables measuring size.
- $\Rightarrow \tau_{\text{int}} = O(N^p)$.

An efficient data structure for SAW

- Represent SAW as a binary tree.
- Enables global moves like pivots to be performed in CPU time $T(N) = O(\log N)$.
- c.f. $O(N^{1-p})$ for hash table implementation[2].
- Dramatic improvement for large N.

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

An efficient data structure for SAW

- Represent SAW as a binary tree.
- Enables global moves like pivots to be performed in CPU time $T(N) = O(\log N)$.
- c.f. $O(N^{1-p})$ for hash table implementation[2].
- Dramatic improvement for large N.

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

An efficient data structure for SAW

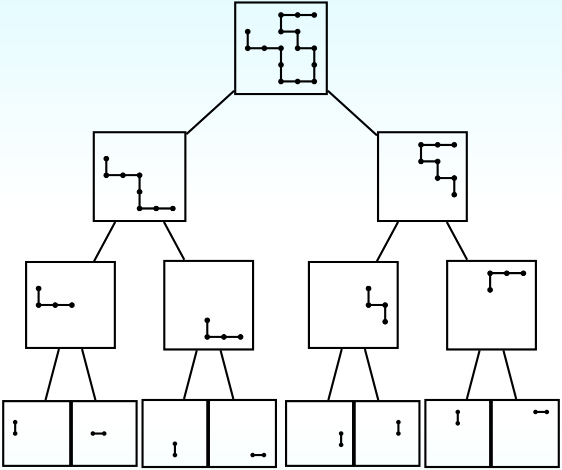
- Represent SAW as a binary tree.
- Enables global moves like pivots to be performed in CPU time $T(N) = O(\log N)$.
- c.f. $O(N^{1-p})$ for hash table implementation[2].
- Dramatic improvement for large N.

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

An efficient data structure for SAW

- Represent SAW as a binary tree.
- Enables global moves like pivots to be performed in CPU time $T(N) = O(\log N)$.
- c.f. $O(N^{1-p})$ for hash table implementation[2].
- Dramatic improvement for large N.

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



SAW-tree representation of a walk.

How to calculate c_N?

- Would like to apply pivot algorithm in canonical ensemble.
- Approach: measure probability that object from larger set is a SAW, $|S| = P(x \in S | x \in T)|T|$, with $|T|$ known.
- Obvious choice: concatenating pairs of SAWs. Every $M + N$-step walk can be split into M and N step subwalks $\Rightarrow c_{M+N} \leq c_M c_N$ for all M, N.
- 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

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

How to calculate c_N?

- Would like to apply pivot algorithm in canonical ensemble.
- Approach: measure probability that object from larger set is a SAW, $|S| = P(x \in S | x \in T)|T|$, with $|T|$ known.
- Obvious choice: concatenating pairs of SAWs. Every $M + N$-step walk can be split into M and N step subwalks $\Rightarrow c_{M+N} \leq c_M c_N$ for all M, N.
- 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

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

How to calculate c_N?

- Would like to apply pivot algorithm in canonical ensemble.
- Approach: measure probability that object from larger set is a SAW, $|S| = P(x \in S | x \in T)|T|$, with $|T|$ known.
- Obvious choice: concatenating pairs of SAWs. Every $M + N$-step walk can be split into M and N step subwalks $\Rightarrow c_{M+N} \leq c_M c_N$ for all M, N.
- 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

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

How to calculate c_N?

- Would like to apply pivot algorithm in canonical ensemble.
- Approach: measure probability that object from larger set is a SAW, $|S| = P(x \in S | x \in T)|T|$, with $|T|$ known.
- Obvious choice: concatenating pairs of SAWs. Every $M + N$-step walk can be split into M and N step subwalks $\Rightarrow c_{M+N} \leq c_M c_N$ for all M, N.
- 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

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

How to calculate c_N?

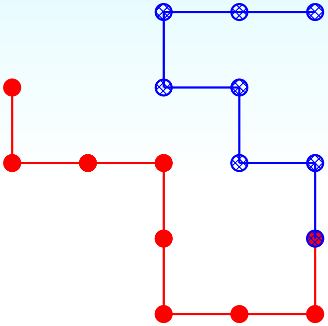
- Would like to apply pivot algorithm in canonical ensemble.
- Approach: measure probability that object from larger set is a SAW, $|S| = P(x \in S | x \in T)|T|$, with $|T|$ known.
- Obvious choice: concatenating pairs of SAWs. Every $M+N$-step walk can be split into M and N step subwalks $\Rightarrow c_{M+N} \leq c_M c_N$ for all M, N.
- 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

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

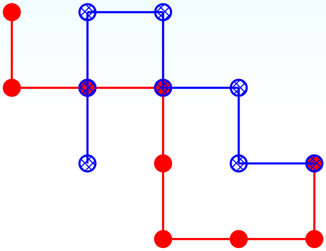
How to calculate c_N?

- Would like to apply pivot algorithm in canonical ensemble.
- Approach: measure probability that object from larger set is a SAW, $|S| = P(x \in S | x \in T)|T|$, with $|T|$ known.
- Obvious choice: concatenating pairs of SAWs. Every $M+N$-step walk can be split into M and N step subwalks $\Rightarrow c_{M+N} \leq c_M c_N$ for all M, N.
- 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

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

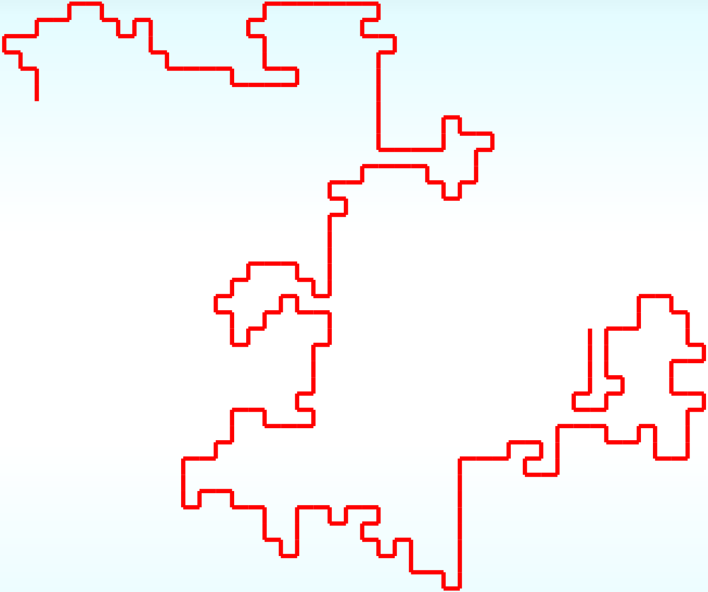


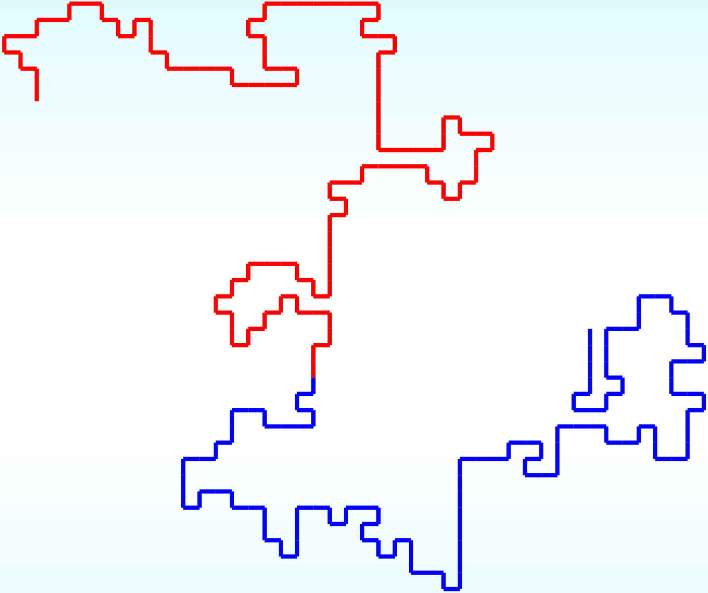
$B(\omega_1, \omega_2) = 1$

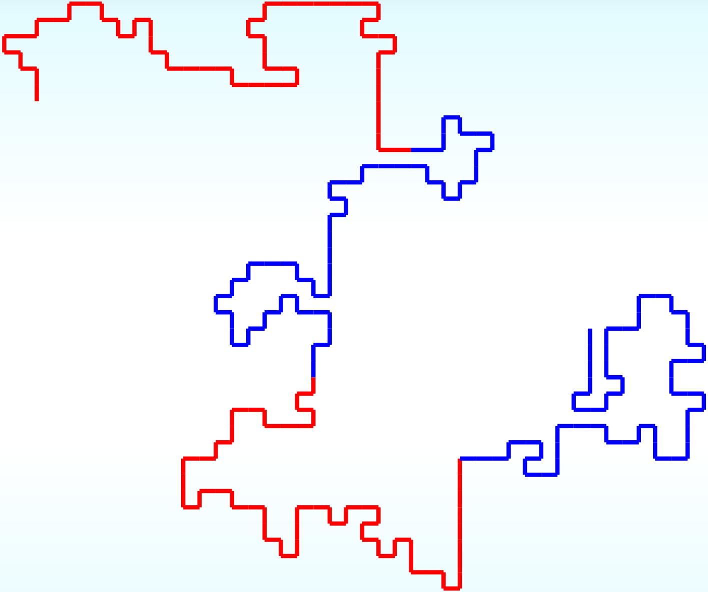


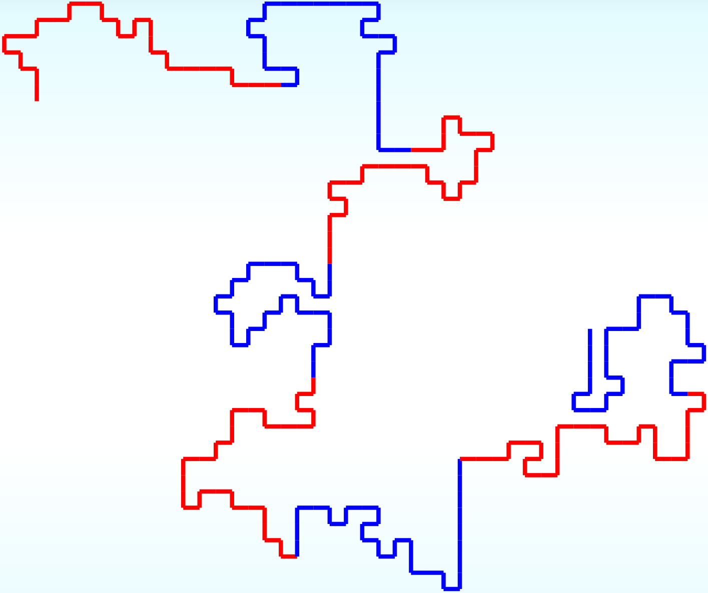
$B(\omega_1, \omega_2) = 0$

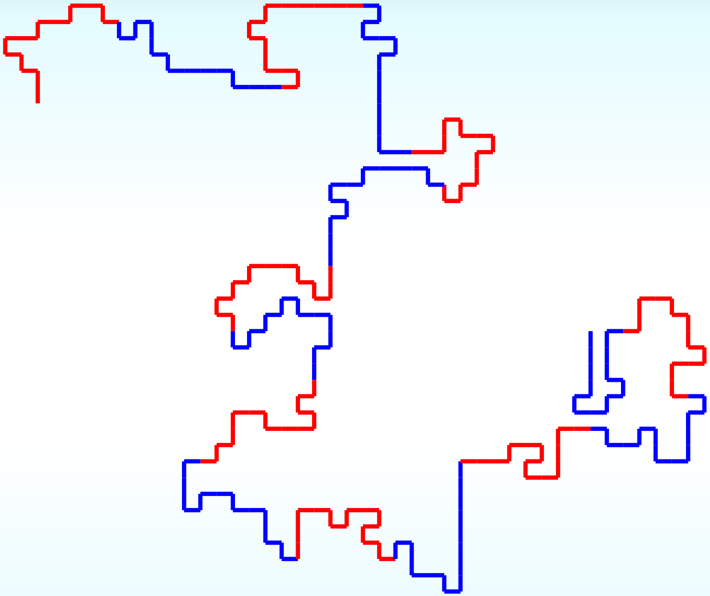
A long N step walk can be successively subdivided into smaller pieces.

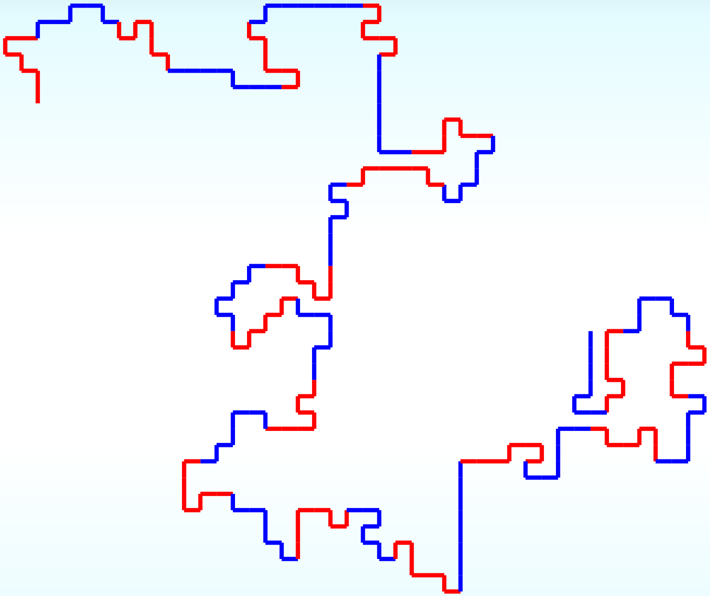


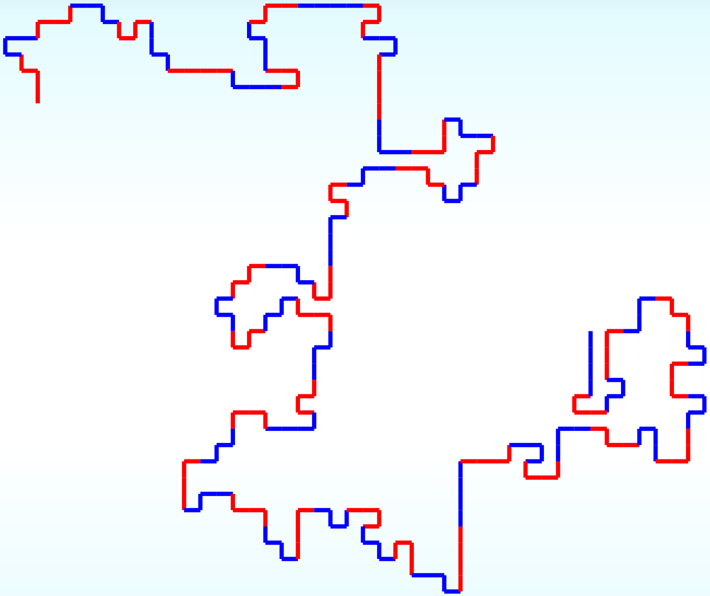


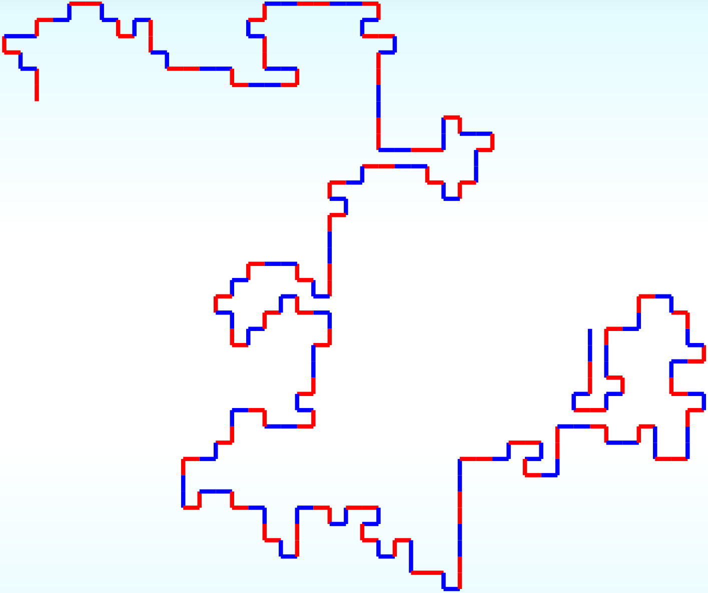


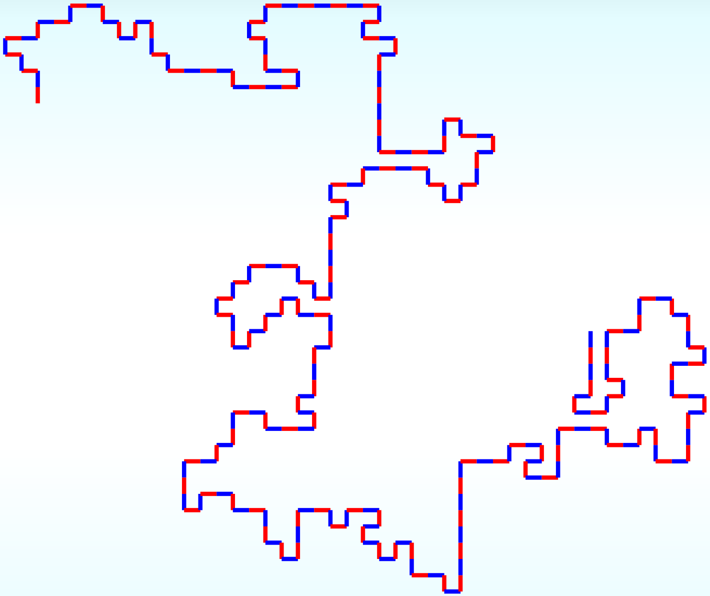












- Could choose $m, n = 36$ (longest known for $\mathbb{Z}^3$):

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

- Iterate to obtain estimates for c_N for longer walks.

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

where c_k is known.

- Telescoping, with length doubling at each iteration.
- C.f. sequential growth, N steps, product of N factors.

- 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.

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

where c_k is known.

- Telescoping, with length doubling at each iteration.
- C.f. sequential growth, N steps, product of N factors.

- 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.

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

where c_k is known.

- Telescoping, with length doubling at each iteration.
- C.f. sequential growth, N steps, product of N factors.

- 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.

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

where c_k is known.

- Telescoping, with length doubling at each iteration.
- C.f. sequential growth, N steps, product of N factors.

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

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

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

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

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

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

Scale free moves

- Need to calculate $\langle B_{k,k} \rangle$, $\langle B_{2k,2k} \rangle$, $\cdots$
- Use pivot algorithm / SAW-tree.
- 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.
- Uniform pivot sites: $\widetilde{\tau}_{\mathrm{int}} = \Omega(N)$.
- Choose distance from joint uniformly from all distance scales, i.e. $u = \log(\text{distance})$ chosen uniformly at random.
- Now: $\widetilde{\tau}_{\mathrm{int}} = N^p \log^2 N$.

Scale free moves

- Need to calculate $\langle B_{k,k} \rangle$, $\langle B_{2k,2k} \rangle$, $\cdots$
- Use pivot algorithm / SAW-tree.
- 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.
- Uniform pivot sites: $\widetilde{\tau}_{\text{int}} = \Omega(N)$.
- Choose distance from joint uniformly from all distance scales, i.e. $u = \log(\text{distance})$ chosen uniformly at random.
- Now: $\widetilde{\tau}_{\text{int}} = N^p \log^2 N$.

Scale free moves

- Need to calculate $\langle B_{k,k} \rangle$, $\langle B_{2k,2k} \rangle$, $\cdots$
- Use pivot algorithm / SAW-tree.
- 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.
- Uniform pivot sites: $\widetilde{\tau}_{\mathrm{int}} = \Omega(N)$.
- Choose distance from joint uniformly from all distance scales, i.e. $u = \log(\text{distance})$ chosen uniformly at random.
- Now: $\widetilde{\tau}_{\mathrm{int}} = N^p \log^2 N$.

Scale free moves

- Need to calculate $\langle B_{k,k} \rangle$, $\langle B_{2k,2k} \rangle$, $\cdots$
- Use pivot algorithm / SAW-tree.
- 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.
- Uniform pivot sites: $\widetilde{\tau}_{\mathrm{int}} = \Omega(N)$.
- Choose distance from joint uniformly from all distance scales, i.e. $u = \log(\text{distance})$ chosen uniformly at random.
- Now: $\widetilde{\tau}_{\mathrm{int}} = N^p \log^2 N$.

Scale free moves

- Need to calculate $\langle B_{k,k} \rangle$, $\langle B_{2k,2k} \rangle$, $\cdots$
- Use pivot algorithm / SAW-tree.
- 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.
- Uniform pivot sites: $\widetilde{\tau}_{\text{int}} = \Omega(N)$.
- Choose distance from joint uniformly from all distance scales, i.e. $u = \log(\text{distance})$ chosen uniformly at random.
- Now: $\widetilde{\tau}_{\text{int}} = N^p \log^2 N$.

Scale free moves

- Need to calculate $\langle B_{k,k} \rangle$, $\langle B_{2k,2k} \rangle$, $\cdots$
- Use pivot algorithm / SAW-tree.
- 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.
- Uniform pivot sites: $\widetilde{\tau}_{\mathrm{int}} = \Omega(N)$.
- Choose distance from joint uniformly from all distance scales, i.e. $u = \log(\text{distance})$ chosen uniformly at random.
- Now: $\widetilde{\tau}_{\mathrm{int}} = N^p \log^2 N$.

Scale free moves

- Need to calculate $\langle B_{k,k} \rangle$, $\langle B_{2k,2k} \rangle$, $\cdots$
- Use pivot algorithm / SAW-tree.
- 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.
- Uniform pivot sites: $\widetilde{\tau}_{\mathrm{int}} = \Omega(N)$.
- Choose distance from joint uniformly from all distance scales, i.e. $u = \log(\text{distance})$ chosen uniformly at random.
- Now: $\widetilde{\tau}_{\mathrm{int}} = N^p \log^2 N$.

Error estimate

- Expected error, for same CPU time, diminishes as a power law for higher order terms in the sum!

$$\log c_N = \frac{N}{k} \log c_k + \frac{N}{2k} \log\langle B_{k,k}\rangle + \cdots + \log\langle B_{N/2,N/2}\rangle$$

- Partition CPU time amongst different terms to minimize overall statistical error (short test run).

$$\sigma^2 = \sum \frac{a_i^2}{t_i} \qquad \text{Total time } t = \sum t_i$$

$$\Rightarrow t_i = \frac{a_i}{\sum a_i} t, \qquad \sigma = \frac{\sum a_i}{\sqrt{t}}$$

- Can accurately predict error on estimate for c_N prior to start of computer experiment.
- Dominated by low k contribution, appropriate partitioning of effort reduced error by $O(\sqrt{\log N})$. Relative error in c_N proportional to $1/k$.

Error estimate

- Expected error, for same CPU time, diminishes as a power law for higher order terms in the sum!

$$\log c_N = \frac{N}{k} \log c_k + \frac{N}{2k} \log \langle B_{k,k} \rangle + \cdots + \log \langle B_{N/2,N/2} \rangle$$

- Partition CPU time amongst different terms to minimize overall statistical error (short test run).

$$\sigma^2 = \sum \frac{a_i^2}{t_i} \qquad \text{Total time } t = \sum t_i$$

$$\Rightarrow t_i = \frac{a_i}{\sum a_i} t, \qquad \sigma = \frac{\sum a_i}{\sqrt{t}}$$

- Can accurately predict error on estimate for c_N prior to start of computer experiment.
- Dominated by low k contribution, appropriate partitioning of effort reduced error by $O(\sqrt{\log N})$. Relative error in c_N proportional to $1/k$.

Error estimate

- Expected error, for same CPU time, diminishes as a power law for higher order terms in the sum!

$$\log c_N = \frac{N}{k} \log c_k + \frac{N}{2k} \log \langle B_{k,k} \rangle + \cdots + \log \langle B_{N/2,N/2} \rangle$$

- Partition CPU time amongst different terms to minimize overall statistical error (short test run).

$$\sigma^2 = \sum \frac{a_i^2}{t_i} \qquad \text{Total time } t = \sum t_i$$

$$\Rightarrow t_i = \frac{a_i}{\sum a_i} t, \qquad \sigma = \frac{\sum a_i}{\sqrt{t}}$$

- Can accurately predict error on estimate for c_N prior to start of computer experiment.
- Dominated by low k contribution, appropriate partitioning of effort reduced error by $O(\sqrt{\log N})$. Relative error in c_N proportional to $1/k$.

Error estimate

- Expected error, for same CPU time, diminishes as a power law for higher order terms in the sum!

$$\log c_N = \frac{N}{k} \log c_k + \frac{N}{2k} \log\langle B_{k,k}\rangle + \cdots + \log\langle B_{N/2,N/2}\rangle$$

- Partition CPU time amongst different terms to minimize overall statistical error (short test run).

$$\sigma^2 = \sum \frac{a_i^2}{t_i} \qquad \text{Total time } t = \sum t_i$$

$$\Rightarrow t_i = \frac{a_i}{\sum a_i} t, \qquad \sigma = \frac{\sum a_i}{\sqrt{t}}$$

- Can accurately predict error on estimate for c_N prior to start of computer experiment.
- Dominated by low k contribution, appropriate partitioning of effort reduced error by $O(\sqrt{\log N})$. Relative error in c_N proportional to $1/k$.

- Can make unbiased estimates of c_N, for N up to 10^9 or so.
- Can push calculation to sufficiently large N s.t. asymptotic corrections for μ completely eliminated.
- $\Rightarrow$ Systematic error for μ negligible, error purely statistical.

- Can make unbiased estimates of c_N, for N up to 10^9 or so.
- Can push calculation to sufficiently large N s.t. asymptotic corrections for μ completely eliminated.
- $\Rightarrow$ Systematic error for μ negligible, error purely statistical.

- Can make unbiased estimates of c_N, for N up to 10^9 or so.
- Can push calculation to sufficiently large N s.t. asymptotic corrections for μ completely eliminated.
- $\Rightarrow$ Systematic error for μ negligible, error purely statistical.

Results

- Calculated $\log c_N$ with relative error of approximately 4×10^{-9} up to $N = 38797311$ (about 60000 CPU hours).
- Concentrated on $\mathbb{Z}^3$ because asymptotic behaviour for $\mathbb{Z}^2$ well understood from series.
- $c_{9471} = 1.43323(8) \times 10^{6352}$
- $c_{38797311} = 7 \times 10^{26018276}$. Confidence interval of mantissa is $(6.6, 8.2)$.
- For comparison, see[3]. Relative error from PERM and related algorithms of the order of 10^{-3} for short walks of 100 steps. Not a fair comparison:
  - Not much CPU time used, i.e. not serious computer experiments.
  - Estimates would degrade for large N. Best case: error increasing as $O(\sqrt{N})$.

[3] E. J. Janse van Rensburg. "Approximate Enumeration of Self-Avoiding Walks". In: *Algorithmic Probability and Combinatorics* 520 (2010), pp. 127–151.

Results

- Calculated $\log c_N$ with relative error of approximately 4×10^{-9} up to $N = 38797311$ (about 60000 CPU hours).
- Concentrated on $\mathbb{Z}^3$ because asymptotic behaviour for $\mathbb{Z}^2$ well understood from series.
- $c_{9471} = 1.43323(8) \times 10^{6352}$
- $c_{38797311} = 7 \times 10^{26018276}$. Confidence interval of mantissa is $(6.6, 8.2)$.
- For comparison, see[3]. Relative error from PERM and related algorithms of the order of 10^{-3} for short walks of 100 steps. Not a fair comparison:
  - Not much CPU time used, i.e. not serious computer experiments.
  - Estimates would degrade for large N. Best case: error increasing as $O(\sqrt{N})$.

[3] E. J. Janse van Rensburg. "Approximate Enumeration of Self-Avoiding Walks". In: *Algorithmic Probability and Combinatorics* 520 (2010), pp. 127–151.

Results

- Calculated $\log c_N$ with relative error of approximately 4×10^{-9} up to $N = 38797311$ (about 60000 CPU hours).
- Concentrated on $\mathbb{Z}^3$ because asymptotic behaviour for $\mathbb{Z}^2$ well understood from series.
- $c_{9471} = 1.43323(8) \times 10^{6352}$
- $c_{38797311} = 7 \times 10^{26018276}$. Confidence interval of mantissa is $(6.6, 8.2)$.
- For comparison, see[3]. Relative error from PERM and related algorithms of the order of 10^{-3} for short walks of 100 steps. Not a fair comparison:
  - Not much CPU time used, i.e. not serious computer experiments.
  - Estimates would degrade for large N. Best case: error increasing as $O(\sqrt{N})$.

[3] E. J. Janse van Rensburg. "Approximate Enumeration of Self-Avoiding Walks". In: *Algorithmic Probability and Combinatorics* 520 (2010), pp. 127–151.

Results

- Calculated $\log c_N$ with relative error of approximately 4×10^{-9} up to $N = 38797311$ (about 60000 CPU hours).
- Concentrated on $\mathbb{Z}^3$ because asymptotic behaviour for $\mathbb{Z}^2$ well understood from series.
- $c_{9471} = 1.43323(8) \times 10^{6352}$
- $c_{38797311} = 7 \times 10^{26018276}$. Confidence interval of mantissa is $(6.6, 8.2)$.
- For comparison, see[3]. Relative error from PERM and related algorithms of the order of 10^{-3} for short walks of 100 steps. Not a fair comparison:
  - Not much CPU time used, i.e. not serious computer experiments.
  - Estimates would degrade for large N. Best case: error increasing as $O(\sqrt{N})$.

[3] E. J. Janse van Rensburg. "Approximate Enumeration of Self-Avoiding Walks". In: *Algorithmic Probability and Combinatorics* 520 (2010), pp. 127–151.

Results

- Calculated $\log c_N$ with relative error of approximately 4×10^{-9} up to $N = 38797311$ (about 60000 CPU hours).
- Concentrated on $\mathbb{Z}^3$ because asymptotic behaviour for $\mathbb{Z}^2$ well understood from series.
- $c_{9471} = 1.43323(8) \times 10^{6352}$
- $c_{38797311} = 7 \times 10^{26018276}$. Confidence interval of mantissa is $(6.6, 8.2)$.
- For comparison, see[3]. Relative error from PERM and related algorithms of the order of 10^{-3} for short walks of 100 steps. Not a fair comparison:
  - Not much CPU time used, i.e. not serious computer experiments.
  - Estimates would degrade for large N. Best case: error increasing as $O(\sqrt{N})$.

[3] E. J. Janse van Rensburg. "Approximate Enumeration of Self-Avoiding Walks". In: *Algorithmic Probability and Combinatorics* 520 (2010), pp. 127–151.

Results

- Calculated $\log c_N$ with relative error of approximately 4×10^{-9} up to $N = 38797311$ (about 60000 CPU hours).
- Concentrated on $\mathbb{Z}^3$ because asymptotic behaviour for $\mathbb{Z}^2$ well understood from series.
- $c_{9471} = 1.43323(8) \times 10^{6352}$
- $c_{38797311} = 7 \times 10^{26018276}$. Confidence interval of mantissa is $(6.6, 8.2)$.
- For comparison, see[3]. Relative error from PERM and related algorithms of the order of 10^{-3} for short walks of 100 steps. Not a fair comparison:
  - Not much CPU time used, i.e. not serious computer experiments.
  - Estimates would degrade for large N. Best case: error increasing as $O(\sqrt{N})$.

[3] E. J. Janse van Rensburg. "Approximate Enumeration of Self-Avoiding Walks". In: *Algorithmic Probability and Combinatorics* 520 (2010), pp. 127–151.

Results

- Calculated $\log c_N$ with relative error of approximately 4×10^{-9} up to $N = 38797311$ (about 60000 CPU hours).
- Concentrated on $\mathbb{Z}^3$ because asymptotic behaviour for $\mathbb{Z}^2$ well understood from series.
- $c_{9471} = 1.43323(8) \times 10^{6352}$
- $c_{38797311} = 7 \times 10^{26018276}$. Confidence interval of mantissa is $(6.6, 8.2)$.
- For comparison, see[3]. Relative error from PERM and related algorithms of the order of 10^{-3} for short walks of 100 steps. Not a fair comparison:
  - Not much CPU time used, i.e. not serious computer experiments.
  - Estimates would degrade for large N. Best case: error increasing as $O(\sqrt{N})$.

[3] E. J. Janse van Rensburg. "Approximate Enumeration of Self-Avoiding Walks". In: *Algorithmic Probability and Combinatorics* 520 (2010), pp. 127–151.

Results

- For $\mathbb{Z}^3$ we have:
- PERM: $\mu = 4.684038(6)$ (Hsu and Grassberger, "Polymers confined between two parallel plane walls")
- Series: $\mu = 4.68404(1)$ (Clisby, Liang, and Slade, "Self-avoiding walk enumeration via the lace expansion")
- Series: $\mu = 4.684040(5)$ (Schram, Barkema, and Bisseling, "Exact enumeration of self-avoiding walks")
- Pivot: $\mu = 4.68403993(3)$, almost 200 times more accurate than previous best ($\sigma = 2.7 \times 10^{-8}$).

Results

- For $\mathbb{Z}^3$ we have:
- PERM: $\mu = 4.684038(6)$ (Hsu and Grassberger, "Polymers confined between two parallel plane walls")
- Series: $\mu = 4.68404(1)$ (Clisby, Liang, and Slade, "Self-avoiding walk enumeration via the lace expansion")
- Series: $\mu = 4.684040(5)$ (Schram, Barkema, and Bisseling, "Exact enumeration of self-avoiding walks")
- Pivot: $\mu = 4.68403993(3)$, almost 200 times more accurate than previous best ($\sigma = 2.7 \times 10^{-8}$).

Results

- For $\mathbb{Z}^3$ we have:
- PERM: $\mu = 4.684038(6)$ (Hsu and Grassberger, "Polymers confined between two parallel plane walls")
- Series: $\mu = 4.68404(1)$ (Clisby, Liang, and Slade, "Self-avoiding walk enumeration via the lace expansion")
- Series: $\mu = 4.684040(5)$ (Schram, Barkema, and Bisseling, "Exact enumeration of self-avoiding walks")
- Pivot: $\mu = 4.68403993(3)$, almost 200 times more accurate than previous best ($\sigma = 2.7 \times 10^{-8}$).

Results

- For $\mathbb{Z}^3$ we have:
- PERM: $\mu = 4.684038(6)$ (Hsu and Grassberger, "Polymers confined between two parallel plane walls")
- Series: $\mu = 4.68404(1)$ (Clisby, Liang, and Slade, "Self-avoiding walk enumeration via the lace expansion")
- Series: $\mu = 4.684040(5)$ (Schram, Barkema, and Bisseling, "Exact enumeration of self-avoiding walks")
- Pivot: $\mu = 4.68403993(3)$, almost 200 times more accurate than previous best ($\sigma = 2.7 \times 10^{-8}$).

Results

- For $\mathbb{Z}^3$ we have:
- PERM: $\mu = 4.684038(6)$ (Hsu and Grassberger, "Polymers confined between two parallel plane walls")
- Series: $\mu = 4.68404(1)$ (Clisby, Liang, and Slade, "Self-avoiding walk enumeration via the lace expansion")
- Series: $\mu = 4.684040(5)$ (Schram, Barkema, and Bisseling, "Exact enumeration of self-avoiding walks")
- Pivot: $\mu = 4.68403993(3)$, almost 200 times more accurate than previous best ($\sigma = 2.7 \times 10^{-8}$).

Conclusion

- Simple computer experiment.
- Different ingredients fit together to produce extremely accurate estimates.
- Choose a Monte Carlo scheme which enables efficient sampling (large jumps in state space)
- Efficient data structures help.
- Can you do better than incremental growth? (fusing objects and doubling size, or splitting in two)
- Is the self-avoiding walk model uniquely favourable, or can these ideas be applied elsewhere?

Conclusion

- Simple computer experiment.
- Different ingredients fit together to produce extremely accurate estimates.
- Choose a Monte Carlo scheme which enables efficient sampling (large jumps in state space)
- Efficient data structures help.
- Can you do better than incremental growth? (fusing objects and doubling size, or splitting in two)
- Is the self-avoiding walk model uniquely favourable, or can these ideas be applied elsewhere?

Conclusion

- Simple computer experiment.
- Different ingredients fit together to produce extremely accurate estimates.
- Choose a Monte Carlo scheme which enables efficient sampling (large jumps in state space)
- Efficient data structures help.
- Can you do better than incremental growth? (fusing objects and doubling size, or splitting in two)
- Is the self-avoiding walk model uniquely favourable, or can these ideas be applied elsewhere?

Conclusion

- Simple computer experiment.
- Different ingredients fit together to produce extremely accurate estimates.
- Choose a Monte Carlo scheme which enables efficient sampling (large jumps in state space)
- Efficient data structures help.
- Can you do better than incremental growth? (fusing objects and doubling size, or splitting in two)
- Is the self-avoiding walk model uniquely favourable, or can these ideas be applied elsewhere?

Conclusion

- Simple computer experiment.
- Different ingredients fit together to produce extremely accurate estimates.
- Choose a Monte Carlo scheme which enables efficient sampling (large jumps in state space)
- Efficient data structures help.
- Can you do better than incremental growth? (fusing objects and doubling size, or splitting in two)
- Is the self-avoiding walk model uniquely favourable, or can these ideas be applied elsewhere?

Conclusion

- Simple computer experiment.
- Different ingredients fit together to produce extremely accurate estimates.
- Choose a Monte Carlo scheme which enables efficient sampling (large jumps in state space)
- Efficient data structures help.
- Can you do better than incremental growth? (fusing objects and doubling size, or splitting in two)
- Is the self-avoiding walk model uniquely favourable, or can these ideas be applied elsewhere?