# The importance of fast algorithms for simulating complex systems

Nathan Clisby
MASCOS, The University of Melbourne

Mathematics of Planet Earth Conference
July 11, 2013

# Statistical mechanics

- Study of physical systems with large numbers of particles.

# Statistical mechanics

- Study of physical systems with large numbers of particles.
- Model particle interactions using simple rules $\cdots$

# Statistical mechanics

- Study of physical systems with large numbers of particles.
- Model particle interactions using simple rules $\cdots$
- $\cdots$ but system may still have globally emergent changes called *phase transitions*.

# Statistical mechanics

- Study of physical systems with large numbers of particles.
- Model particle interactions using simple rules $\cdots$
- $\cdots$ but system may still have globally emergent changes called *phase transitions*.
- Collective, emergent behaviour is emblematic of *complex systems*.

# Two typical phase transitions are $\cdots$

Two typical phase transitions are $\cdots$

Liquid water boiling

(source: wikimedia commons)

Two typical phase transitions are · · ·

Liquid water boiling

(source: wikimedia commons)

Ice melting

source: Ian Joughin, released under Creative Commons license www.imaggeo.net/view/984

Two typical phase transitions are · · ·

Liquid water boiling

(source: wikimedia commons)

Ice melting

source: Ian Joughin, released under Creative Commons license www.imaggeo.net/view/984

Study of long chain molecules, or polymers, is another important example.

# Why study simple mathematical models?

# Why study simple mathematical models?

- By studying simple model systems we can get *exact* information about real-world systems.

# Why study simple mathematical models?

- By studying simple model systems we can get *exact* information about real-world systems.
- In particular, can understand *how* phase transitions occur.

# Why study simple mathematical models?

- By studying simple model systems we can get *exact* information about real-world systems.
- In particular, can understand *how* phase transitions occur.
- These common features are *universal*.

- Will focus on one model system (polymers).

- Will focus on one model system (polymers).
- Will show what a "good" computer algorithm looks like, and explain why good algorithms make a big difference.

# Our model: self-avoiding walk (SAW)

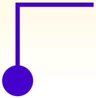
- A walk on a grid, with the rule that you can't revisit any grid points.

# Our model: self-avoiding walk (SAW)

- A walk on a grid, with the rule that you can't revisit any grid points.
- Can study SAW in any dimension, most commonly 3 dimensions due to physical relevance, or in 2 dimensions because this is a rich and important topic in mathematical physics.

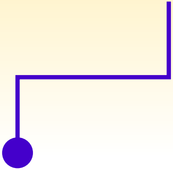
Simple random walk

Simple random walk

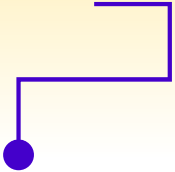


Simple random walk

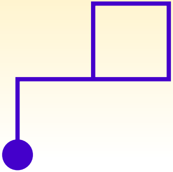
Simple random walk



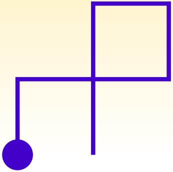
Simple random walk



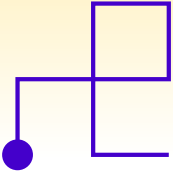
Simple random walk



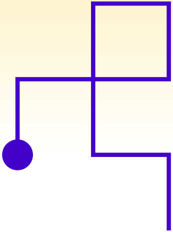
Simple random walk



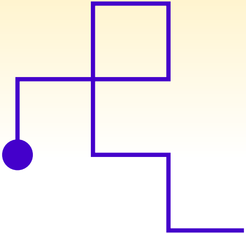
Simple random walk



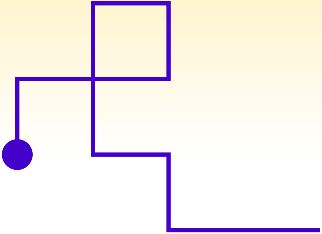
Simple random walk



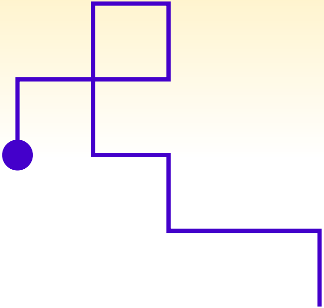
Simple random walk



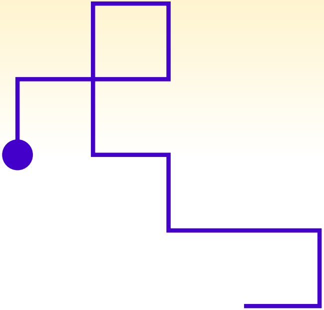
Simple random walk



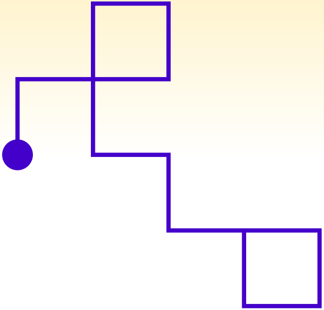
Simple random walk



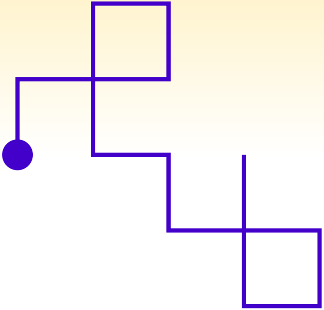
Simple random walk



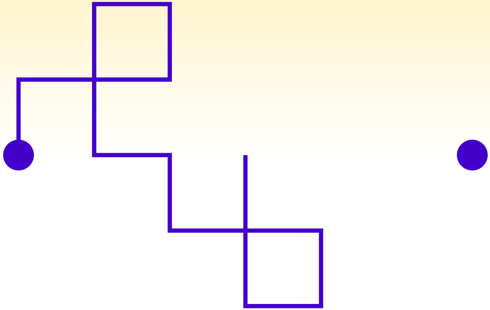
Simple random walk



Simple random walk

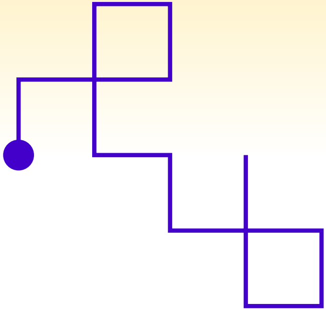


Simple random walk



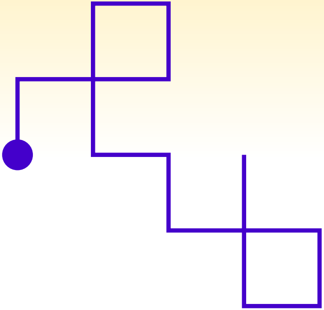
Simple random walk

Self-avoiding walk

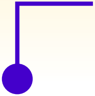


Simple random walk

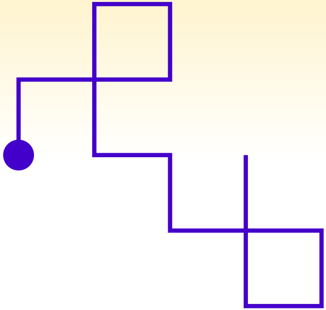
Self-avoiding walk



Simple random walk

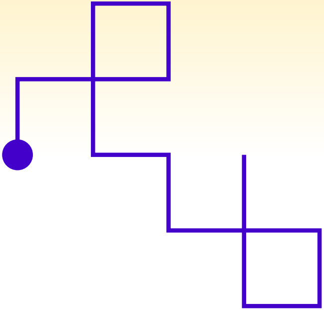


Self-avoiding walk

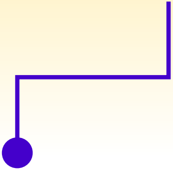


Simple random walk

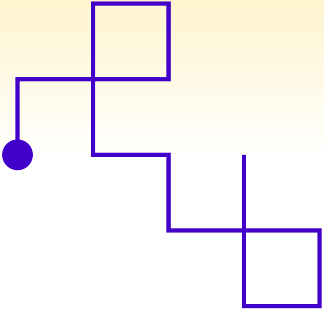
Self-avoiding walk



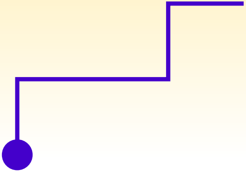
Simple random walk



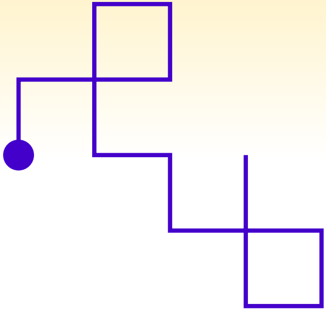
Self-avoiding walk



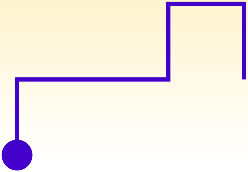
Simple random walk



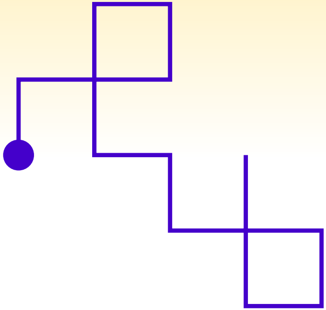
Self-avoiding walk



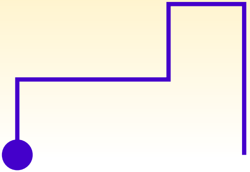
Simple random walk



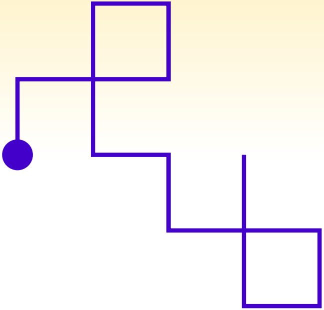
Self-avoiding walk



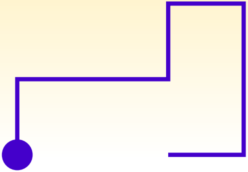
Simple random walk



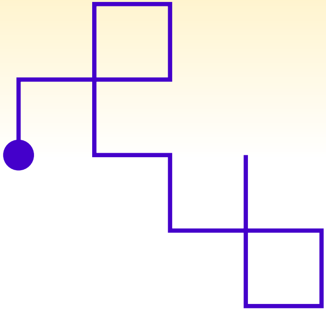
Self-avoiding walk



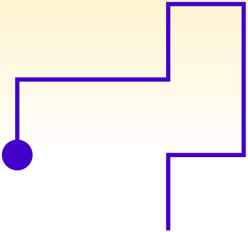
Simple random walk



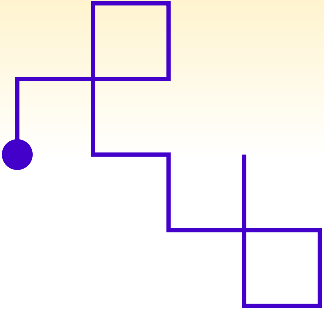
Self-avoiding walk



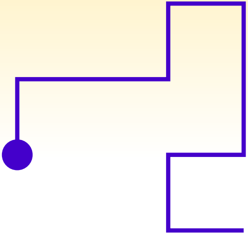
Simple random walk



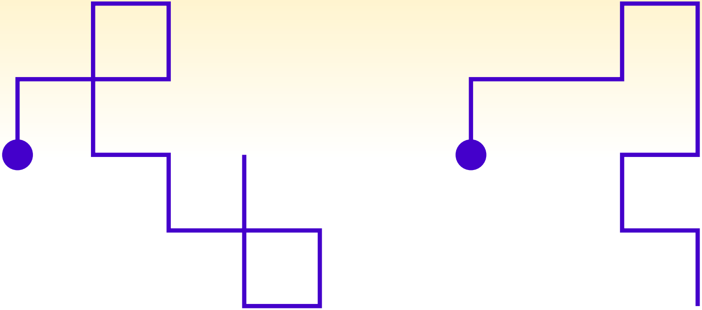
Self-avoiding walk



Simple random walk

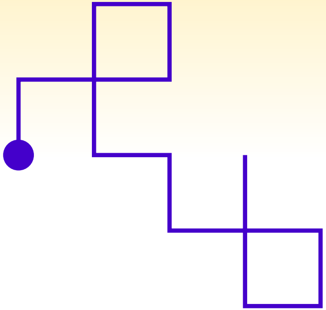


Self-avoiding walk

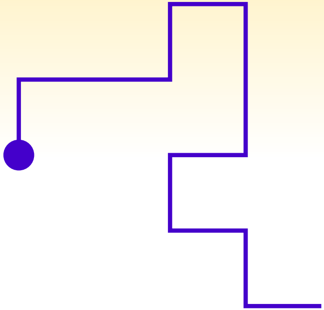


Simple random walk

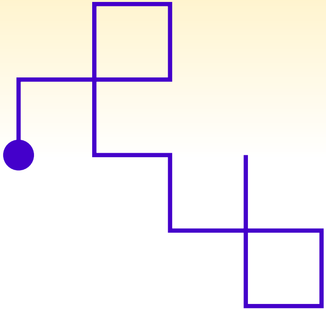
Self-avoiding walk



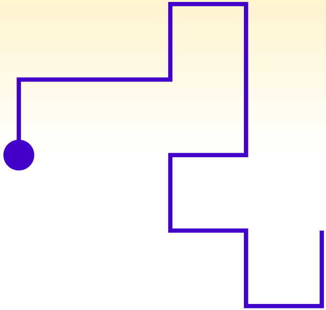
Simple random walk



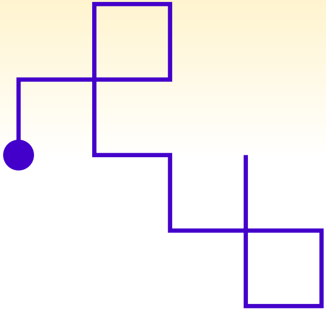
Self-avoiding walk



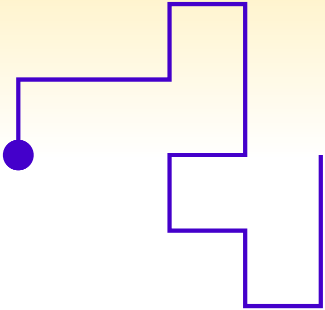
Simple random walk



Self-avoiding walk



Simple random walk



Self-avoiding walk

- Polymers are long chain molecules, which may have millions or even billions of atoms (monomers) in the chain.

- Polymers are long chain molecules, which may have millions or even billions of atoms (monomers) in the chain.
- Plastic, rubber, and DNA are all polymers.

- Polymers are long chain molecules, which may have millions or even billions of atoms (monomers) in the chain.
- Plastic, rubber, and DNA are all polymers.
- Clearly, can't have two atoms in the same place, and it turns out that this is the key property of polymers in a good solvent.

- Polymers are long chain molecules, which may have millions or even billions of atoms (monomers) in the chain.
- Plastic, rubber, and DNA are all polymers.
- Clearly, can't have two atoms in the same place, and it turns out that this is the key property of polymers in a good solvent.
- In fact SAW *exactly* captures universal properties of polymers, such as the way in which the typical size of a polymer grows with the number of atoms in the chain.

- Typical size of a SAW / polymer grows with the number of monomers, $N$, as:

$$\mathbf{R} = Dn^{\nu}$$

- Typical size of a SAW / polymer grows with the number of monomers, $N$, as:

$$\mathbf{R} = Dn^{\nu}$$

- $D$ is non-universal, different for every kind of polymer molecule, or every choice of grid for SAW.

- Typical size of a SAW / polymer grows with the number of monomers, $N$, as:

$$\mathbf{R} = Dn^{\nu}$$

- $D$ is non-universal, different for every kind of polymer molecule, or every choice of grid for SAW.
- $\nu$ is universal! *Exactly* the same in each case.

- Typical size of a SAW / polymer grows with the number of monomers, $N$, as:

$$\mathbf{R} = Dn^{\nu}$$

- $D$ is non-universal, different for every kind of polymer molecule, or every choice of grid for SAW.
- $\nu$ is universal! *Exactly* the same in each case.
- SAW are *fractal* objects, and $1/\nu$ is their fractal dimension.

- Self-avoidance forces self-avoiding walks to be much more spread out than walks which are allowed to intersect themselves.

- Self-avoidance forces self-avoiding walks to be much more spread out than walks which are allowed to intersect themselves.
- More spread out in 2 dimensions than 3, because in 3 dimensions walks have more freedom and so it's easier to remain self-avoiding.

- Self-avoidance forces self-avoiding walks to be much more spread out than walks which are allowed to intersect themselves.
- More spread out in 2 dimensions than 3, because in 3 dimensions walks have more freedom and so it's easier to remain self-avoiding.
- Because they are fractals they are self-similar.

- Self-avoidance forces self-avoiding walks to be much more spread out than walks which are allowed to intersect themselves.
- More spread out in 2 dimensions than 3, because in 3 dimensions walks have more freedom and so it's easier to remain self-avoiding.
- Because they are fractals they are self-similar.

Typical simple random walks and self-avoiding walks

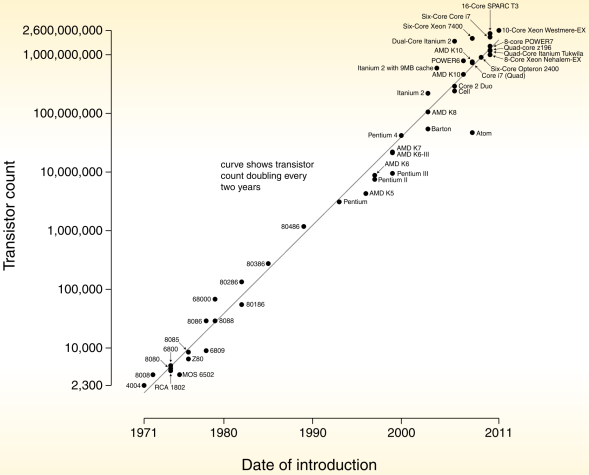
# Moore's Law

- In 1965 Moore observed that the number of transistors on an integrated circuit appeared to increase exponentially with time, doubling every 2 years.

# Moore's Law

- In 1965 Moore observed that the number of transistors on an integrated circuit appeared to increase exponentially with time, doubling every 2 years.
- This rule of thumb has held to a good approximation ever since, i.e. available computing power per $ has increased exponentially.

Microprocessor Transistor Counts 1971-2011 & Moore's Law




Line shows doubling every 2 years. Source: Wikipedia.

- Since 1985, transistor count has increased from approximately 300 thousand, to 3 billion.

- Since 1985, transistor count has increased from approximately 300 thousand, to 3 billion.
- Computer hardware also much cheaper.

- Since 1985, transistor count has increased from approximately 300 thousand, to 3 billion.
- Computer hardware also much cheaper.
- Translates to rough increase of computing power available per dollar of the order of 10 million. (Neglecting electricity and maintenance, which would reduce this factor.)

- Since 1985, transistor count has increased from approximately 300 thousand, to 3 billion.
- Computer hardware also much cheaper.
- Translates to rough increase of computing power available per dollar of the order of 10 million. (Neglecting electricity and maintenance, which would reduce this factor.)
- This makes a *big* difference to the quality of computer simulations, and the size / resolution of the systems that can be simulated.

- Exact solution (very hard for SAW)

- Exact solution (very hard for SAW)
- Theory / mathematical techniques (extremely useful, especially for $d = 2$ and $d \geq 4$)

- Exact solution (very hard for SAW)
- Theory / mathematical techniques (extremely useful, especially for $d = 2$ and $d \geq 4$)
- Molecular dynamics (very important for understanding polymer dynamics)

- Exact solution (very hard for SAW)
- Theory / mathematical techniques (extremely useful, especially for $d = 2$ and $d \geq 4$)
- Molecular dynamics (very important for understanding polymer dynamics)
- Computer enumeration (very powerful for $d = 2$, useful for $d = 3$)

- Exact solution (very hard for SAW)
- Theory / mathematical techniques (extremely useful, especially for $d = 2$ and $d \geq 4$)
- Molecular dynamics (very important for understanding polymer dynamics)
- Computer enumeration (very powerful for $d = 2$, useful for $d = 3$)
- Markov chain Monte Carlo (MCMC) sampling, and other Monte Carlo methods.

# MCMC

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

# 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".

# 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:

# 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

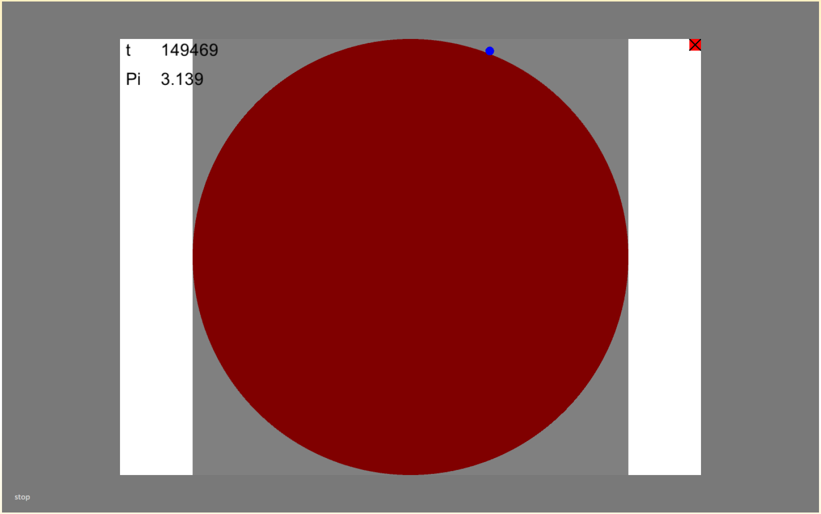
# MCMC

- 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

# 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

MCMC estimation of $\pi$


t 149469
Pi 3.139
stop

# 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 monomer, 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 monomer, 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 monomer, 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 monomer, 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 monomer, 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.
- CPU time $O(N)$ for 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 monomer, 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.
- CPU time $O(N)$ for an essentially new configuration.
- 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

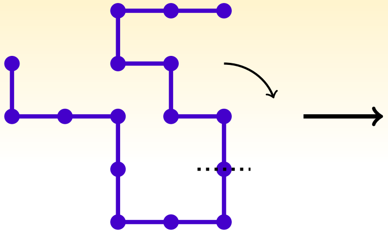
- Procedure:
  - Choose a pivot site at random
  - Then rotate or reflect one of the two parts of the walk.

# Pivot algorithm

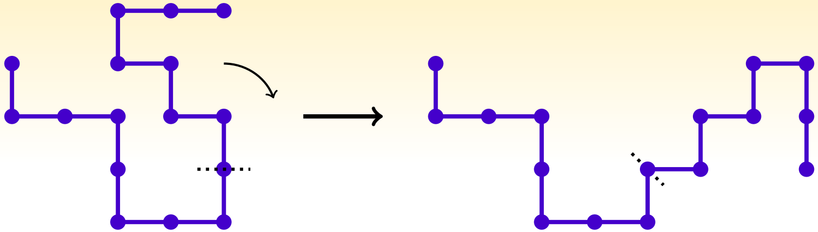
- Procedure:
  - Choose a pivot site at random
  - Then rotate or reflect one of the two parts of the walk.
  - Retain new walk if it is self-avoiding, otherwise restore original walk.

# Pivot algorithm

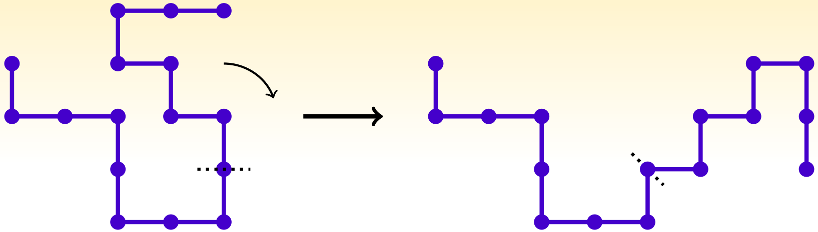
- Procedure:
  - Choose a pivot site at random
  - Then rotate or reflect one of the two parts of the walk.
  - Retain new walk if it is self-avoiding, otherwise restore original walk.
- "Global" because on average half of the monomers are moved.



Example pivot move



Example pivot move



Example pivot move

Run simulation

# Why is it so effective?

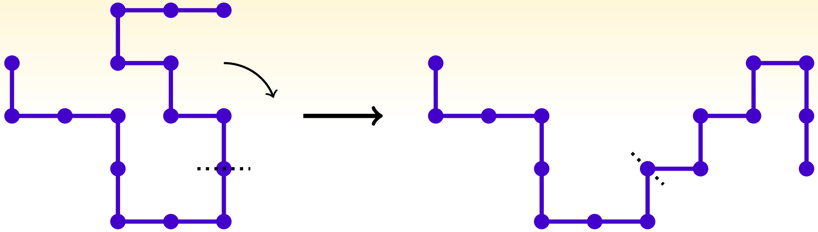
- Every time a pivot attempt *is* successful there is a large change in global observables.

# Why is it so effective?

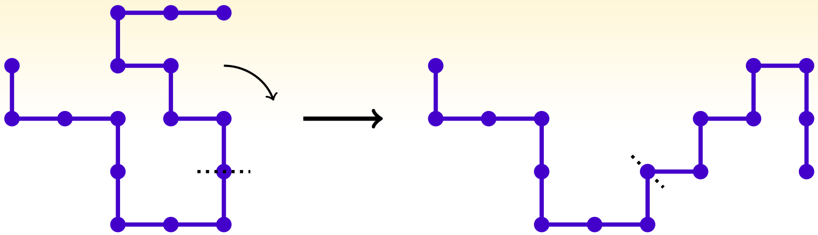
- Every time a pivot attempt *is* successful there is a large change in global observables.
- Only need $O(1)$ successful pivots before we have an *essentially new* configuration.

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.

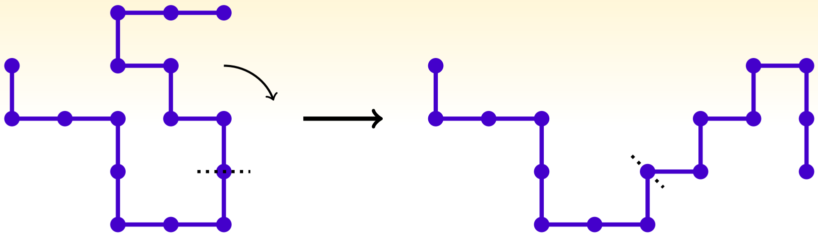


More to the story: pivot moves are in fact *local* moves if you think of them the right way.



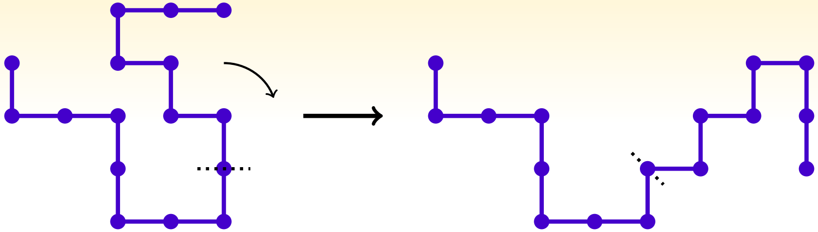
LSRSLSL**S**LRLRRS

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 *local cost*. Tricky part is checking that after subwalk is rotated the walk remains self-avoiding.

Following simulation based on timings from 3 years ago. $1s$ in animation corresponds to $10\mu s$ of computer time.

Comparison of hash table and SAW-tree implementations

Following simulation based on timings from 3 years ago. $1s$ in animation corresponds to $10\mu s$ of computer time.

## Comparison of hash table and SAW-tree implementations

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

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

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

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

# Polymers

- A key scientific question regarding polymers is their likelihood to form knots.

# Polymers

- A key scientific question regarding polymers is their likelihood to form knots.
- For "large" $N$ almost all polymers are knotted.

# Polymers

- A key scientific question regarding polymers is their likelihood to form knots.
- For "large" $N$ almost all polymers are knotted.
- But half-life of the unknot is 250 thousand!

# Polymers

- A key scientific question regarding polymers is their likelihood to form knots.
- For "large" $N$ almost all polymers are knotted.
- But half-life of the unknot is 250 thousand!
- To quantitatively understand knotting of DNA need to be able to simulate polymers with hundreds of millions or billions of monomers.

# Polymers

- A key scientific question regarding polymers is their likelihood to form knots.
- For "large" $N$ almost all polymers are knotted.
- But half-life of the unknot is 250 thousand!
- To quantitatively understand knotting of DNA need to be able to simulate polymers with hundreds of millions or billions of monomers.
- For $N = 1$ billion, since 1985 speed-up factors of:

# Polymers

- A key scientific question regarding polymers is their likelihood to form knots.
- For "large" $N$ almost all polymers are knotted.
- But half-life of the unknot is 250 thousand!
- To quantitatively understand knotting of DNA need to be able to simulate polymers with hundreds of millions or billions of monomers.
- For $N = 1$ billion, since 1985 speed-up factors of:
    - 10 million from computer improvements

# Polymers

- A key scientific question regarding polymers is their likelihood to form knots.
- For "large" $N$ almost all polymers are knotted.
- But half-life of the unknot is 250 thousand!
- To quantitatively understand knotting of DNA need to be able to simulate polymers with hundreds of millions or billions of monomers.
- 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)

# Polymers

- A key scientific question regarding polymers is their likelihood to form knots.
- For "large" $N$ almost all polymers are knotted.
- But half-life of the unknot is 250 thousand!
- To quantitatively understand knotting of DNA need to be able to simulate polymers with hundreds of millions or billions of monomers.
- 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, Clisby in 2010)

# Polymers

- A key scientific question regarding polymers is their likelihood to form knots.
- For "large" $N$ almost all polymers are knotted.
- But half-life of the unknot is 250 thousand!
- To quantitatively understand knotting of DNA need to be able to simulate polymers with hundreds of millions or billions of monomers.
- 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, Clisby in 2010)
- This is not the end of the story for polymer knotting - other hard problems must also be solved.

- New computers: improve all calculations by a (large) constant factor.

- New computers: improve all calculations by a (large) constant factor.
- New algorithms: factor depends on size of system, may make new regimes accessible.

- New computers: improve all calculations by a (large) constant factor.
- New algorithms: factor depends on size of system, may make new regimes accessible.
- Challenge: find efficient algorithms and computer implementations for other systems. (Very active research area, e.g. cluster algorithms, Wang-Landau method, PERM, worm algorithms.)