The Quantum Metropolis Algorithm

An implementation of Metropolis' famous algorithm on a quantum computer.

Robert Rüger

Institut für **T**heoretische **P**hysik
Goethe-Universität Frankfurt

July 13$^{\text{th}}$, 2011

Contents

Contents

The Classical Metropolis Algorithm
... its origins

Exact calculation of expectation values in thermal equilibrium

$$\langle O \rangle = \frac{\int \mathrm{d}\Gamma \, O(\Gamma) \, \mathrm{e}^{-\beta \, H(\Gamma)}}{\int \mathrm{d}\Gamma \, \mathrm{e}^{-\beta \, H(\Gamma)}} \xrightarrow{\text{discretization}} \frac{\sum_\mu O(\mu) \, \mathrm{e}^{-\beta \, H(\mu)}}{\sum_\mu \mathrm{e}^{-\beta \, H(\mu)}}$$

Problem: Exact evaluation is numerically so absurdly laborious, that it is applicable only for the smallest systems!

Example: A system of 100 two-state particles has 2^{100} different states. The fastest computers do about 10^{16} simple operations per second.

$$\frac{2^{100}}{10^{16}\mathrm{s}^{-1}} = 1.27 \cdot 10^{14}\mathrm{s} = 4 \text{ million years!}$$

The Classical Metropolis Algorithm
... its origins

Exact calculation of expectation values in thermal equilibrium

$$\langle O \rangle = \frac{\int \mathrm{d}\Gamma \, O(\Gamma) \, \mathrm{e}^{-\beta \, H(\Gamma)}}{\int \mathrm{d}\Gamma \, \mathrm{e}^{-\beta \, H(\Gamma)}} \xrightarrow{\text{discretization}} \frac{\sum_\mu O(\mu) \, \mathrm{e}^{-\beta \, H(\mu)}}{\sum_\mu \mathrm{e}^{-\beta \, H(\mu)}}$$

Problem: Exact evaluation is numerically so absurdly laborious, that it is applicable only for the smallest systems!

Example: A system of 100 two-state particles has 2^{100} different states. The fastest computers do about 10^{16} simple operations per second.

$$\frac{2^{100}}{10^{16}\mathrm{s}^{-1}} = 1.27 \cdot 10^{14}\mathrm{s} = 4 \text{ million years!}$$

The Classical Metropolis Algorithm

... its origins

Exact calculation of expectation values in thermal equilibrium

$$\langle O \rangle = \frac{\int \mathrm{d}\Gamma \, O(\Gamma) \, \mathrm{e}^{-\beta \, H(\Gamma)}}{\int \mathrm{d}\Gamma \, \mathrm{e}^{-\beta \, H(\Gamma)}} \quad \xrightarrow{\text{discretization}} \quad \frac{\sum_\mu O(\mu) \, \mathrm{e}^{-\beta \, H(\mu)}}{\sum_\mu \mathrm{e}^{-\beta \, H(\mu)}}$$

Problem: Exact evaluation is numerically so absurdly laborious, that it is applicable only for the smallest systems!

Example: A system of 100 two-state particles has 2^{100} different states. The fastest computers do about 10^{16} simple operations per second.

$$\frac{2^{100}}{10^{16}\mathrm{s}^{-1}} = 1.27 \cdot 10^{14}\mathrm{s} = 4 \text{ million years!}$$

The Classical Metropolis Algorithm
... its origins

Simple Sampling Monte Carlo

$$\langle O \rangle \approx \langle O \rangle_{ss} = \frac{\sum_{\mu_{i=1}}^{\mu_N} O(\mu_i)\, \mathrm{e}^{-\beta\, H(\mu_i)}}{\sum_{\mu_{i=1}}^{\mu_N} \mathrm{e}^{-\beta\, H(\mu_i)}}$$

Problem: A lot of computer time is wasted sampling states that don't add significantly to the sum.

Example: Think of a system at very low temperature, where only a few states near the ground-state contribute to the sums and all the others are exponentially surpressed.

The Classical Metropolis Algorithm

... its origins

Simple Sampling Monte Carlo

$$\langle O \rangle \approx \langle O \rangle_{ss} = \frac{\sum_{\mu_{i=1}}^{\mu_N} O(\mu_i)\, \mathrm{e}^{-\beta\, H(\mu_i)}}{\sum_{\mu_{i=1}}^{\mu_N} \mathrm{e}^{-\beta\, H(\mu_i)}}$$

Problem: A lot of computer time is wasted sampling states that don't add significantly to the sum.

Example: Think of a system at very low temperature, where only a few states near the ground-state contribute to the sums and all the others are exponentially surpressed.

The Classical Metropolis Algorithm
... its origins

Simple Sampling Monte Carlo

$$\langle O \rangle \approx \langle O \rangle_{ss} = \frac{\sum_{\mu_{i=1}}^{\mu_N} O(\mu_i)\, \mathrm{e}^{-\beta\, H(\mu_i)}}{\sum_{\mu_{i=1}}^{\mu_N} \mathrm{e}^{-\beta\, H(\mu_i)}}$$

Problem: A lot of computer time is wasted sampling states that don't add significantly to the sum.

Example: Think of a system at very low temperature, where only a few states near the ground-state contribute to the sums and all the others are exponentially surpressed.

The Classical Metropolis Algorithm

... its origins

"Instead of choosing configurations randomly, then weighting them with $\exp(-E/kT)$*, we choose configurations with a probability* $\exp(-E/kT)$ *and weight them evenly."*

— Metropolis et al.

Importance Sampling Monte Carlo

$$\langle O \rangle \approx \langle O \rangle_{is} = \frac{1}{N} \sum_{\mu_{i=1}^{(\rho)}}^{\mu_N^{(\rho)}} O\left(\mu_i^{(\rho)}\right) \quad \text{with} \quad \rho = \frac{1}{Z}\, \mathrm{e}^{-\beta\, H\left(\mu_i^{(\rho)}\right)}$$

Note: This is exactly like an experimental measurement, with the exception that we have to generate the states ourselves, while nature does it for the experimental physicist.

The Classical Metropolis Algorithm

... its origins

"Instead of choosing configurations randomly, then weighting them with $\exp(-E/kT)$*, we choose configurations with a probability* $\exp(-E/kT)$ *and weight them evenly."*

— Metropolis et al.

Importance Sampling Monte Carlo

$$\langle O \rangle \approx \langle O \rangle_{is} = \frac{1}{N} \sum_{\mu_{i=1}^{(\rho)}}^{\mu_N^{(\rho)}} O\left(\mu_i^{(\rho)}\right) \quad \text{with} \quad \rho = \frac{1}{Z}\, \mathrm{e}^{-\beta\, H\left(\mu_i^{(\rho)}\right)}$$

Note: This is exactly like an experimental measurement, with the exception that we have to generate the states ourselves, while nature does it for the experimental physicist.

The Classical Metropolis Algorithm

... its origins

"Instead of choosing configurations randomly, then weighting them with $\exp(-E/kT)$*, we choose configurations with a probability* $\exp(-E/kT)$ *and weight them evenly."*

— Metropolis et al.

Importance Sampling Monte Carlo

$$\langle O \rangle \approx \langle O \rangle_{is} = \frac{1}{N} \sum_{\mu_{i=1}^{(\rho)}}^{\mu_N^{(\rho)}} O\left(\mu_i^{(\rho)}\right) \quad \text{with} \quad \rho = \frac{1}{Z}\, \mathrm{e}^{-\beta\, H\left(\mu_i^{(\rho)}\right)}$$

Note: This is exactly like an experimental measurement, with the exception that <u>we</u> have to generate the states ourselves, while <u>nature</u> does it for the experimental physicist.

The Classical Metropolis Algorithm

... a very quick derivation

Problem

It it is not trivial to generate states that follow the Boltzmann distribution!

Idea

1. Find one of the "important states".
2. Modify it a litte bit to get another important state.

Goal

Build a Markov process whose Markov chain contains Boltzmann distributed states.

The Classical Metropolis Algorithm

... a very quick derivation

Problem

It it is not trivial to generate states that follow the Boltzmann distribution!

Idea

1. Find one of the “important states”.
2. Modify it a litte bit to get another important state.

Goal

Build a Markov process whose Markov chain contains Boltzmann distributed states.

The Classical Metropolis Algorithm

... a very quick derivation

Problem

It it is not trivial to generate states that follow the Boltzmann distribution!

Idea

1. Find one of the "important states".
2. Modify it a litte bit to get another important state.

Goal

Build a Markov process whose Markov chain contains Boltzmann distributed states.

The Classical Metropolis Algorithm

... a very quick derivation

Master Equation

Look at the probability $P_k(\mu)$ of the system to be in the state μ after k Markov steps.

$$P_{k+1}(\mu) = P_k(\mu) + \sum_{\nu} \Big(P_k(\nu)\, T(\nu \to \mu) - P_k(\mu)\, T(\mu \to \nu) \Big)$$

Detailed Balance

In the end, we want $P_k(\mu)$ to be the states Boltzmann probability for all steps k. This implies $P_{k+1}(\mu) = P_k(\mu)$.

$$\Rightarrow \quad P_k(\nu)\, T(\nu \to \mu) = P_k(\mu)\, T(\mu \to \nu)$$

The Classical Metropolis Algorithm

... a very quick derivation

Master Equation

Look at the probability $P_k(\mu)$ of the system to be in the state μ after k Markov steps.

$$P_{k+1}(\mu) = P_k(\mu) + \sum_{\nu} \Big(P_k(\nu)\, T(\nu \to \mu) - P_k(\mu)\, T(\mu \to \nu) \Big)$$

Detailed Balance

In the end, we want $P_k(\mu)$ to be the states Boltzmann probability for all steps k. This implies $P_{k+1}(\mu) = P_k(\mu)$.

$$\Rightarrow \quad P_k(\nu)\, T(\nu \to \mu) = P_k(\mu)\, T(\mu \to \nu)$$

The Classical Metropolis Algorithm

... a very quick derivation

Transition probabilities

Detailed balance gives the required transition probabilities.

$$\frac{T(\mu \to \nu)}{T(\nu \to \mu)} = \frac{P(\nu)}{P(\mu)} = \frac{Z^{-1}\,\mathrm{e}^{-\beta\,H(\nu)}}{Z^{-1}\,\mathrm{e}^{-\beta\,H(\mu)}} = \mathrm{e}^{-\beta\left(H(\nu)-H(\mu)\right)}$$

Two step Markov process

$$\frac{T(\mu \to \nu)}{T(\nu \to \mu)} = \frac{S(\mu \to \nu)}{S(\nu \to \mu)}\frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \mathrm{e}^{-\beta\left(H(\nu)-H(\mu)\right)}$$

Where $S(\mu \to \nu)$ is the probability that a transition is suggested and $A(\mu \to \nu)$ is its probability to be accepted.

The Classical Metropolis Algorithm

... a very quick derivation

Transition probabilities

Detailed balance gives the required transition probabilities.

$$\frac{T(\mu \to \nu)}{T(\nu \to \mu)} = \frac{P(\nu)}{P(\mu)} = \frac{Z^{-1}\,\mathrm{e}^{-\beta\,H(\nu)}}{Z^{-1}\,\mathrm{e}^{-\beta\,H(\mu)}} = \mathrm{e}^{-\beta\big(H(\nu)-H(\mu)\big)}$$

Two step Markov process

$$\frac{T(\mu \to \nu)}{T(\nu \to \mu)} = \frac{S(\mu \to \nu)}{S(\nu \to \mu)}\,\frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \mathrm{e}^{-\beta\big(H(\nu)-H(\mu)\big)}$$

Where $S(\mu \to \nu)$ is the probability that a transition is suggested and $A(\mu \to \nu)$ is its probability to be accepted.

The Classical Metropolis Algorithm

... a very quick derivation

Choice of $S(\mu \to \nu)$ and $A(\mu \to \nu)$

Choose $S(\mu \to \nu) = S(\nu \to \mu)$ equal in both directions.

$$\Rightarrow \quad \frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \mathrm{e}^{-\beta\left(H(\nu)-H(\mu)\right)}$$

Choose the largest acceptance probabilities that still satisfy the equation.

$$A(\mu \to \nu) = \min\left(1,\ \mathrm{e}^{-\beta\left(H(\nu)-H(\mu)\right)}\right)$$

This particular choice of S and A defines the Metropolis Algorithm!

The Classical Metropolis Algorithm

... a very quick derivation

Choice of $S(\mu \to \nu)$ and $A(\mu \to \nu)$

Choose $S(\mu \to \nu) = S(\nu \to \mu)$ equal in both directions.

$$\Rightarrow \quad \frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \mathrm{e}^{-\beta\big(H(\nu)-H(\mu)\big)}$$

Choose the largest acceptance probabilities that still satisfy the equation.

$$A(\mu \to \nu) = \min\left(1,\ \mathrm{e}^{-\beta\big(H(\nu)-H(\mu)\big)}\right)$$

This particular choice of S and A defines the Metropolis Algorithm!

The Classical Metropolis Algorithm

... a very quick derivation

Choice of $S(\mu \to \nu)$ and $A(\mu \to \nu)$

Choose $S(\mu \to \nu) = S(\nu \to \mu)$ equal in both directions.

$$\Rightarrow \quad \frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \mathrm{e}^{-\beta\left(H(\nu)-H(\mu)\right)}$$

Choose the largest acceptance probabilities that still satisfy the equation.

$$A(\mu \to \nu) = \min\left(1,\ \mathrm{e}^{-\beta\left(H(\nu)-H(\mu)\right)}\right)$$

This particular choice of S and A defines the Metropolis Algorithm!

The Classical Metropolis Algorithm

... an example: 2d Ising-Model

The Ising-Model Hamiltonian

$$H = -J \sum_{\langle ij \rangle} S_i S_j - B \sum_i S_i$$

Implementation of the Metropolis Algorithm

1. Start in an arbitrary state.
2. E.g.: Randomly select a single spin to flip.
3. Calculate the resulting energy difference ΔE.
4. Flip it with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

(... show the video of the algorithm at work ...)

The Classical Metropolis Algorithm

... an example: 2d Ising-Model

The Ising-Model Hamiltonian

$$H = -J \sum_{\langle ij \rangle} S_i S_j - B \sum_i S_i$$

Implementation of the Metropolis Algorithm

1. Start in an arbitrary state.
2. E.g.: Randomly select a single spin to flip.
3. Calculate the resulting energy difference ΔE.
4. Flip it with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

(... show the video of the algorithm at work ...)

The Classical Metropolis Algorithm

... an example: 2d Ising-Model

The Ising-Model Hamiltonian

$$H = -J \sum_{\langle ij \rangle} S_i S_j - B \sum_i S_i$$

Implementation of the Metropolis Algorithm

1. Start in an arbitrary state.
2. E.g.: Randomly select a single spin to flip.
3. Calculate the resulting energy difference ΔE.
4. Flip it with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

(... show the video of the algorithm at work ...)

The Classical Metropolis Algorithm

... an example: 2d Ising-Model

The Ising-Model Hamiltonian

$$H = -J \sum_{\langle ij \rangle} S_i S_j - B \sum_i S_i$$

Implementation of the Metropolis Algorithm

1. Start in an arbitrary state.
2. E.g.: Randomly select a single spin to flip.
3. Calculate the resulting energy difference ΔE.
4. Flip it with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

(... show the video of the algorithm at work ...)

The Classical Metropolis Algorithm

... an example: 2d Ising-Model

The Ising-Model Hamiltonian

$$H = -J \sum_{\langle ij \rangle} S_i S_j - B \sum_i S_i$$

Implementation of the Metropolis Algorithm

1. Start in an arbitrary state.
2. E.g.: Randomly select a single spin to flip.
3. Calculate the resulting energy difference ΔE.
4. Flip it with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

(... show the video of the algorithm at work ...)

The Classical Metropolis Algorithm

... an example: 2d Ising-Model

The Ising-Model Hamiltonian

$$H = -J \sum_{\langle ij \rangle} S_i S_j - B \sum_i S_i$$

Implementation of the Metropolis Algorithm

1. Start in an arbitrary state.
2. E.g.: Randomly select a single spin to flip.
3. Calculate the resulting energy difference ΔE.
4. Flip it with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

(... show the video of the algorithm at work ...)

The Classical Metropolis Algorithm

... an example: 2d Ising-Model

The Ising-Model Hamiltonian

$$H = -J \sum_{\langle ij \rangle} S_i S_j - B \sum_i S_i$$

Implementation of the Metropolis Algorithm

1. Start in an arbitrary state.
2. E.g.: Randomly select a single spin to flip.
3. Calculate the resulting energy difference ΔE.
4. Flip it with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

(... show the video of the algorithm at work ...)

Contents

The Basics of Quantum Computing

... bits and qubits

Classical Bit

- bit = **b**inary dig**it** or **b**asic **i**ndissoluble **i**nformation **u**nit
- basic unit for measuring information
- has the two states 0 and 1

Quantum Bit

- qubit = **qu**antum **bit**
- basic unit of quantum information
- state in a two dimensional Hilbert space

$$|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle \quad \text{with} \quad |c_0|^2 + |c_1|^2 = 1$$

- $|0\rangle$ and $|1\rangle$ are called the computational basis

The Basics of Quantum Computing

... bits and qubits

Classical Bit

- bit = **b**inary dig**it** or **b**asic indissoluble **i**nformation **u**nit
- basic unit for measuring information
- has the two states 0 and 1

Quantum Bit

- qubit = **qu**antum **bit**
- basic unit of quantum information
- state in a two dimensional Hilbert space

$$|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle \quad \text{with} \quad |c_0|^2 + |c_1|^2 = 1$$

- $|0\rangle$ and $|1\rangle$ are called the computational basis

The Basics of Quantum Computing

... registers and quantum registers

Registers

- a collection of N classical bits, e.g. (01101010)
- has 2^N different states

Quantum Registers

- a tensor product of N qubits, e.g. $|\phi\rangle = |\psi\rangle_1 \otimes |\psi\rangle_2$
- state in a 2^N dimensional Hilbert space, e.g.

$$|\phi\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle$$

$$\text{with} \quad |c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1$$

The Basics of Quantum Computing

... registers and quantum registers

Registers

- a collection of N classical bits, e.g. (01101010)
- has 2^N different states

Quantum Registers

- a tensor product of N qubits, e.g. $|\phi\rangle = |\psi\rangle_1 \otimes |\psi\rangle_2$
- state in a 2^N dimensional Hilbert space, e.g.

$$|\phi\rangle = c_{00}\,|00\rangle + c_{01}\,|01\rangle + c_{10}\,|10\rangle + c_{11}\,|11\rangle$$
$$\text{with} \quad |c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1$$

Contents

The Quantum Metropolis Algorithm

... what is it?

Goal

We want a quantum computer to sample the energy eigenstates $|\psi_i\rangle$ of a given Hamiltonian according to the Boltzmann distribution.

Straightforward translation into quantum mechanics

1. Start in a random energy eigenstate $|\psi_i\rangle$ with energy E_i.
2. Suggest a nearby energy eigenstate $|\psi_j\rangle$ with energy E_j.
3. Calculate their energy difference $\Delta E = E_j - E_i$.
4. Go to $|\psi_j\rangle$ with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

... unfortunately there are some problems with this ...

The Quantum Metropolis Algorithm

... what is it?

Goal

We want a quantum computer to sample the energy eigenstates $|\psi_i\rangle$ of a given Hamiltonian according to the Boltzmann distribution.

Straightforward translation into quantum mechanics

1. Start in a random energy eigenstate $|\psi_i\rangle$ with energy E_i.
2. Suggest a nearby energy eigenstate $|\psi_j\rangle$ with energy E_j.
3. Calculate their energy difference $\Delta E = E_j - E_i$.
4. Go to $|\psi_j\rangle$ with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

... unfortunately there are some problems with this ...

The Quantum Metropolis Algorithm

... what is it?

Goal

We want a quantum computer to sample the energy eigenstates $|\psi_i\rangle$ of a given Hamiltonian according to the Boltzmann distribution.

Straightforward translation into quantum mechanics

1. Start in a random energy eigenstate $|\psi_i\rangle$ with energy E_i.
2. Suggest a nearby energy eigenstate $|\psi_j\rangle$ with energy E_j.
3. Calculate their energy difference $\Delta E = E_j - E_i$.
4. Go to $|\psi_j\rangle$ with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

... unfortunately there are some problems with this ...

The Quantum Metropolis Algorithm

... what is it?

Goal

We want a quantum computer to sample the energy eigenstates $|\psi_i\rangle$ of a given Hamiltonian according to the Boltzmann distribution.

Straightforward translation into quantum mechanics

1. Start in a random energy eigenstate $|\psi_i\rangle$ with energy E_i.
2. Suggest a nearby energy eigenstate $|\psi_j\rangle$ with energy E_j.
3. Calculate their energy difference $\Delta E = E_j - E_i$.
4. Go to $|\psi_j\rangle$ with probability $\min\left(1,\ e^{-\beta\,\Delta E}\right)$.
5. Return to 2.

... unfortunately there are some problems with this ...

The Quantum Metropolis Algorithm

... what is it?

Goal

We want a quantum computer to sample the energy eigenstates $|\psi_i\rangle$ of a given Hamiltonian according to the Boltzmann distribution.

Straightforward translation into quantum mechanics

1. Start in a random energy eigenstate $|\psi_i\rangle$ with energy E_i.
2. Suggest a nearby energy eigenstate $|\psi_j\rangle$ with energy E_j.
3. Calculate their energy difference $\Delta E = E_j - E_i$.
4. Go to $|\psi_j\rangle$ with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

... unfortunately there are some problems with this ...

The Quantum Metropolis Algorithm

... what is it?

Goal

We want a quantum computer to sample the energy eigenstates $|\psi_i\rangle$ of a given Hamiltonian according to the Boltzmann distribution.

Straightforward translation into quantum mechanics

1. Start in a random energy eigenstate $|\psi_i\rangle$ with energy E_i.
2. Suggest a nearby energy eigenstate $|\psi_j\rangle$ with energy E_j.
3. Calculate their energy difference $\Delta E = E_j - E_i$.
4. Go to $|\psi_j\rangle$ with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

... unfortunately there are some problems with this ...

The Quantum Metropolis Algorithm

... what is it?

Goal

We want a quantum computer to sample the energy eigenstates $|\psi_i\rangle$ of a given Hamiltonian according to the Boltzmann distribution.

Straightforward translation into quantum mechanics

1. Start in a random energy eigenstate $|\psi_i\rangle$ with energy E_i.
2. Suggest a nearby energy eigenstate $|\psi_j\rangle$ with energy E_j.
3. Calculate their energy difference $\Delta E = E_j - E_i$.
4. Go to $|\psi_j\rangle$ with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

... unfortunately there are some problems with this ...

The Quantum Metropolis Algorithm

... what is it?

Goal

We want a quantum computer to sample the energy eigenstates $|\psi_i\rangle$ of a given Hamiltonian according to the Boltzmann distribution.

Straightforward translation into quantum mechanics

1. Start in a random energy eigenstate $|\psi_i\rangle$ with energy E_i.
2. Suggest a nearby energy eigenstate $|\psi_j\rangle$ with energy E_j.
3. Calculate their energy difference $\Delta E = E_j - E_i$.
4. Go to $|\psi_j\rangle$ with probability $\min\left(1,\ \mathrm{e}^{-\beta\,\Delta E}\right)$.
5. Return to 2.

... unfortunately there are some problems with this ...

The Quantum Metropolis Algorithm

... the implementation

Quantum Phase Estimation Algorithm

Attaches the binary representation of the states energy as a quantum register.

$$\sum_i c_i \left|\psi_i\right\rangle \longmapsto \sum_i c_i \left|\psi_i\right\rangle \otimes \left|E_i\right\rangle$$

Reading of the energy register collapses the state to the corresponding energy eigenstate.

This has two applications:

1. Preparing a random energy eigenstate.
2. Measuring the energy of a given energy eigenstate.

Quantum Phase Estimation Algorithm

Attaches the binary representation of the states energy as a quantum register.

$$\sum_i c_i |\psi_i\rangle \longmapsto \sum_i c_i |\psi_i\rangle \otimes |E_i\rangle$$

Reading of the energy register collapses the state to the corresponding energy eigenstate.

This has two applications:

1. Preparing a random energy eigenstate.
2. Measuring the energy of a given energy eigenstate.

Quantum Phase Estimation Algorithm

Attaches the binary representation of the states energy as a quantum register.

$$\sum_i c_i |\psi_i\rangle \longmapsto \sum_i c_i |\psi_i\rangle \otimes |E_i\rangle$$

Reading of the energy register collapses the state to the corresponding energy eigenstate.

This has two applications:

1. Preparing a random energy eigenstate.
2. Measuring the energy of a given energy eigenstate.

The Quantum Metropolis Algorithm
... the implementation

Quantum Phase Estimation Algorithm

Attaches the binary representation of the states energy as a quantum register.

$$\sum_i c_i |\psi_i\rangle \longmapsto \sum_i c_i |\psi_i\rangle \otimes |E_i\rangle$$

Reading of the energy register collapses the state to the corresponding energy eigenstate.

This has two applications:

1. Preparing a random energy eigenstate.
2. Measuring the energy of a given energy eigenstate.

The Quantum Metropolis Algorithm
... the implementation

There is no way to go from one eigenstate to the other directly!

Generation of new eigenstates

Apply a random local unitary transformation C.

$$C : |\psi_i\rangle \longmapsto \sum_j c_j |\psi_j\rangle \text{ where } E_j \approx E_i$$

Next step? Use QPE to collapse to a new eigenstate $|\psi_j\rangle$ and learn its energy E_j?

$$\sum_j c_j |\psi_j\rangle \longmapsto \sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \longmapsto |\psi_j\rangle \otimes |E_j\rangle$$

The Quantum Metropolis Algorithm
... the implementation

There is no way to go from one eigenstate to the other directly!

Generation of new eigenstates

Apply a random <u>local</u> unitary transformation C.

$$C : |\psi_i\rangle \longmapsto \sum_j c_j |\psi_j\rangle \text{ where } E_j \approx E_i$$

Next step? Use QPE to collapse to a new eigenstate $|\psi_j\rangle$ and learn its energy E_j?

$$\sum_j c_j |\psi_j\rangle \longmapsto \sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \longmapsto |\psi_j\rangle \otimes |E_j\rangle$$

The Quantum Metropolis Algorithm
... the implementation

There is no way to go from one eigenstate to the other directly!

Generation of new eigenstates

Apply a random <u>local</u> unitary transformation C.

$$C : |\psi_i\rangle \longmapsto \sum_j c_j |\psi_j\rangle \text{ where } E_j \approx E_i$$

Next step? Use QPE to collapse to a new eigenstate $|\psi_j\rangle$ and learn its energy E_j?

$$\sum_j c_j |\psi_j\rangle \longmapsto \sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \longmapsto |\psi_j\rangle \otimes |E_j\rangle$$

The Quantum Metropolis Algorithm
... the implementation

Measuring E_j irreversibly collapses into state $|\psi_j\rangle$!
How would we go back to $|\psi_i\rangle$ if we reject the move?

A less destructive measurement

Idea: An energy measurement reveals more information than we actually need! One bit (accept/reject) would be enough ...

$$\sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \longmapsto \sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \otimes \left(\sqrt{w_{ij}}\,|1\rangle + \sqrt{1-w_{ij}}\,|0\rangle\right)$$

If we measure the last qubit, only one bit of information is revealed and less damage is done to the state.

Better chance to undo it!

The Quantum Metropolis Algorithm

... the implementation

Measuring E_j irreversibly collapses into state $|\psi_j\rangle$!
How would we go back to $|\psi_i\rangle$ if we reject the move?

A less destructive measurement

Idea: An energy measurement reveals more information than we actually need! One bit (accept/reject) would be enough ...

$$\sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \longmapsto \sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \otimes \left(\sqrt{w_{ij}}\,|1\rangle + \sqrt{1-w_{ij}}\,|0\rangle\right)$$

If we measure the last qubit, only one bit of information is revealed and less damage is done to the state.

Better chance to undo it!

The Quantum Metropolis Algorithm

... the implementation

Measuring E_j irreversibly collapses into state $|\psi_j\rangle$!
How would we go back to $|\psi_i\rangle$ if we reject the move?

A less destructive measurement

Idea: An energy measurement reveals more information than we actually need! One bit (accept/reject) would be enough ...

$$\sum_j c_j \, |\psi_j\rangle \otimes |E_j\rangle \;\longmapsto\; \sum_j c_j \, |\psi_j\rangle \otimes |E_j\rangle \otimes \left(\sqrt{w_{ij}}\, |1\rangle + \sqrt{1 - w_{ij}}\, |0\rangle \right)$$

If we measure the last qubit, only one bit of information is revealed and less damage is done to the state.

Better chance to undo it!

Two different binary measurements

We can use QPE to determine if we are back in state $|\psi_i\rangle$.
Formally, we can define a projector on $|\psi_i\rangle$.

$$P = 1\,|\psi_i\rangle\,\langle\psi_i| + 0\,|\psi_i^\perp\rangle\,\langle\psi_i^\perp| = |\psi_i\rangle\,\langle\psi_i|$$

Use accept/reject as another binary measurement.

$$Q^{\|} = 1\,|1\rangle\,\langle 1| + 0\,|0\rangle\,\langle 0| = |1\rangle\,\langle 1|$$
$$Q^{\perp} = 0\,|1\rangle\,\langle 1| + 1\,|0\rangle\,\langle 0| = |0\rangle\,\langle 0|$$
$$\Rightarrow \quad Q = Q^{\|} + Q^{\perp} = \mathbb{1}$$

Two different binary measurements

We can use QPE to determine if we are back in state $|\psi_i\rangle$.
Formally, we can define a projector on $|\psi_i\rangle$.

$$P = 1\,|\psi_i\rangle\langle\psi_i| + 0\,|\psi_i^\perp\rangle\langle\psi_i^\perp| = |\psi_i\rangle\langle\psi_i|$$

Use accept/reject as another binary measurement.

$$\begin{aligned} Q^{\parallel} &= 1\,|1\rangle\langle 1| + 0\,|0\rangle\langle 0| = |1\rangle\langle 1| \\ Q^{\perp} &= 0\,|1\rangle\langle 1| + 1\,|0\rangle\langle 0| = |0\rangle\langle 0| \\ \Rightarrow \quad Q &= Q^{\parallel} + Q^{\perp} = \mathbb{1} \end{aligned}$$

Damage to the system's state

A measurement of the accept/reject qubit also has an effect on the first qubit register, that holds the system's state.

$$\sum_j c_j \, |\psi_j\rangle \xrightarrow{Q^{\|}} \begin{cases} |\phi_Q^{\|}\rangle & \text{if accepted} \\ |\phi_Q^{\perp}\rangle & \text{if rejected} \end{cases}$$

We can express the initial state $|\psi_i\rangle$ as a superposition of the states $|\phi_Q^{\perp}\rangle$ and $|\phi_Q^{\|}\rangle$.

$$|\psi_i\rangle = \left(Q^{\|} + Q^{\perp}\right) |\psi_i\rangle = \sqrt{q}|\phi_Q^{\|}\rangle + \sqrt{1-q}|\phi_Q^{\perp}\rangle$$

The Quantum Metropolis Algorithm
... the implementation

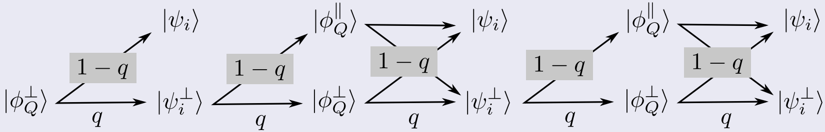
$Q^{\|}$ measurement

$$|\psi_i\rangle = \sqrt{q}|\phi_Q^{\|}\rangle + \sqrt{1-q}|\phi_Q^{\perp}\rangle$$
$$|\psi_i^{\perp}\rangle = \sqrt{1-q}|\phi_Q^{\|}\rangle - \sqrt{q}|\phi_Q^{\perp}\rangle$$

P measurement

$$|\phi_Q^{\|}\rangle = \sqrt{q}\,|\psi_i\rangle + \sqrt{1-q}|\psi_i^{\perp}\rangle$$
$$|\phi_Q^{\perp}\rangle = \sqrt{1-q}\,|\psi_i\rangle - \sqrt{q}|\psi_i^{\perp}\rangle$$

Iteration of $P\,Q^{\|}$ measurements



Probability never to hit $|\psi_i\rangle$ goes down exponentially with the number of iterations.

Quantum Metropolis Algorithm (1/3)

1. Measure the energy of an arbitrary initial state using QPE to prepare a random energy eigenstate $|\psi_i\rangle$.

$$|\psi\rangle = \sum_i c_i |\psi_i\rangle \longmapsto \sum_i c_i |\psi_i\rangle \otimes |E_i\rangle \longmapsto |\psi_i\rangle$$

2. Apply a random local unitary transformation.

$$|\psi_i\rangle \longmapsto \sum_j c_j |\psi_j\rangle \text{ where } E_j \approx E_i$$

Quantum Metropolis Algorithm (1/3)

1. Measure the energy of an arbitrary initial state using QPE to prepare a random energy eigenstate $|\psi_i\rangle$.

$$|\psi\rangle = \sum_i c_i |\psi_i\rangle \longmapsto \sum_i c_i |\psi_i\rangle \otimes |E_i\rangle \longmapsto |\psi_i\rangle$$

2. Apply a random local unitary transformation.

$$|\psi_i\rangle \longmapsto \sum_j c_j |\psi_j\rangle \text{ where } E_j \approx E_i$$

Quantum Metropolis Algorithm (2/3)

3. Attach the energy register $|E_j\rangle$ using QPE and attach the accept/reject qubit.

$$\sum_j c_j |\psi_j\rangle \longmapsto \sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \otimes \left(\sqrt{w_{ij}}\,|1\rangle + \sqrt{1-w_{ij}}\,|0\rangle\right)$$

4. Measure the accept/reject qubit.
    - Accepted! Use QPE to collapse to a new eigenstate $|\psi_j\rangle$ and learn its energy.
    - Rejected! Continue to step 5.

Quantum Metropolis Algorithm (2/3)

3. Attach the energy register $|E_j\rangle$ using QPE and attach the accept/reject qubit.

$$\sum_j c_j |\psi_j\rangle \longmapsto \sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \otimes \left(\sqrt{w_{ij}}\,|1\rangle + \sqrt{1-w_{ij}}\,|0\rangle\right)$$

4. Measure the accept/reject qubit.
   - Accepted! Use QPE to collapse to a new eigenstate $|\psi_j\rangle$ and learn its energy.
   - Rejected! Continue to step 5.

The Quantum Metropolis Algorithm

... the implementation

Quantum Metropolis Algorithm (2/3)

3. Attach the energy register $|E_j\rangle$ using QPE and attach the accept/reject qubit.

$$\sum_j c_j |\psi_j\rangle \longmapsto \sum_j c_j |\psi_j\rangle \otimes |E_j\rangle \otimes \left(\sqrt{w_{ij}}\,|1\rangle + \sqrt{1-w_{ij}}\,|0\rangle\right)$$

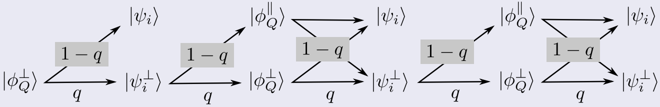
4. Measure the accept/reject qubit.
   - Accepted! Use QPE to collapse to a new eigenstate $|\psi_j\rangle$ and learn its energy.
   - Rejected! Continue to step 5.

The Quantum Metropolis Algorithm

... the implementation

Quantum Metropolis Algorithm (3/3)

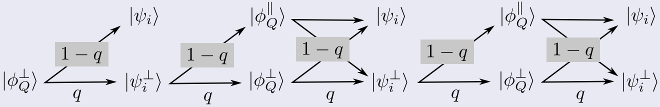
5. Being in state $|\phi_Q^\perp\rangle$ we iterate the $P\,Q^\parallel$ measurements in a two dimensional subspace.



- $|\psi_i\rangle$ hit! Rejection successful, return to step 2.
- $|\psi_i\rangle$ missed! Rejection failed, stop the algorithm.

Quantum Metropolis Algorithm (3/3)

5. Being in state $|\phi_Q^\perp\rangle$ we iterate the $P\ Q^\parallel$ measurements in a two dimensional subspace.



- $|\psi_i\rangle$ hit! Rejection successful, return to step 2.
- $|\psi_i\rangle$ missed! Rejection failed, stop the algorithm.

Sources

- Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.
- M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Oxford University Press, 1999.
- http://en.wikipedia.org/wiki/Quantum_computer
- K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, F. Verstraete. *Quantum Metropolis Sampling*. arXiv:0911.3635v2 [quant-ph], October 2010
- D. Poulin. *Quantum Metropolis Sampling: An algorithm to simulate thermal systems with a quantum computer*. Presentation at the Perimeter Institute for Theoretical Physics, March 2010. http://www.physique.usherbrooke.ca/~dpoulin/Documents/PI2010.pdf

Thank you for your attention!