# Polynomial-time classical simulation of quantum ferromagnets

Sergey Bravyi
David Gosset

IBM

PRL 119, 100503 (2017)
arXiv:1612.05602

**Quantum Monte Carlo:** a powerful suite of probabilistic classical simulation algorithms for quantum many-body systems.

Can simulate systems orders of magnitude larger than with exact diagonalization…

TABLE I. QMC results for the ground-state energy, the spin stiffness, and the squared magnetization per spin. The numbers within parenthesis indicate the statistical errors of the least significant digit of the results.

| $L$ | $-E_0$ | $\rho$ | $M_x^2$ |
|---|---|---|---|
| 4 | 0.562485(4) | 0.2769(1) | 0.13282(2) |
| 6 | 0.552696(4) | 0.2718(1) | 0.11885(4) |
| 8 | 0.550436(4) | 0.2705(2) | 0.1126(2) |
| 10 | 0.549643(4) | 0.2700(3) | 0.1087(2) |
| 12 | 0.549296(4) | 0.2698(4) | 0.1065(3) |
| 14 | 0.549118(4) | 0.2695(3) | |
| 16 | 0.549020(4) | 0.2699(4) | |

**[Sandvik, Hamer 1999]**

Ground state properties of 2D ferromagnetic XY model on $L \times L$ grid.

$$H = -\sum_{<ij>} X_i X_j + Y_i Y_j$$

They also perform finite-temperature numerics up to $L = 64$.

TABLE I. QMC results for the ground-state energy, the spin stiffness, and the squared magnetization per spin. The numbers within parenthesis indicate the statistical errors of the least significant digit of the results.

| $L$ | $-E_0$ | $\rho$ | $M_x^2$ |
|---|---|---|---|
| 4 | 0.562485(4) | 0.2769(1) | 0.13282(2) |
| 6 | 0.552696(4) | 0.2718(1) | 0.11885(4) |
| 8 | 0.550436(4) | 0.2705(2) | 0.1126(2) |
| 10 | 0.549643(4) | 0.2700(3) | 0.1087(2) |
| 12 | 0.549296(4) | 0.2698(4) | 0.1065(3) |
| 14 | 0.549118(4) | 0.2695(3) | |
| 16 | 0.549020(4) | 0.2699(4) | |

**[Sandvik, Hamer 1999]**
Ground state properties of 2D ferromagnetic XY model on $L \times L$ grid.

$$H = -\sum_{<ij>} X_i X_j + Y_i Y_j$$

They also perform finite-temperature numerics up to $L = 64$.

**This is a classical simulation of up to 4096 spins!**
**What's the catch?**
**How does it work?**

## What's the catch?

Quantum Monte Carlo can only be used to study **stoquastic Hamiltonians**

$$\langle x|H|x\rangle \in \mathbb{R} \qquad \langle y|H|x\rangle \leq 0 \quad x \neq y$$

**Stoquastic**
i.e., "sign-problem free"

# What's the catch?

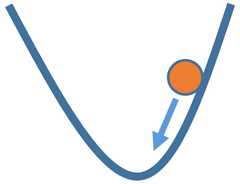
Quantum Monte Carlo can only be used to study **stoquastic Hamiltonians**

$$\langle x|H|x\rangle \in \mathbb{R} \qquad \langle y|H|x\rangle \leq 0 \qquad x \neq y$$

**Stoquastic**
i.e., "sign-problem free"

**Examples:**

Particle in a potential



$$H = \frac{p^2}{2m} + V(\vec{x})$$

# What's the catch?

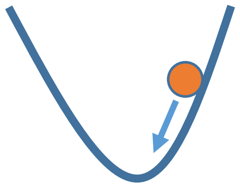
Quantum Monte Carlo can only be used to study **stoquastic Hamiltonians**

$$\langle x|H|x\rangle \in \mathbb{R} \qquad \langle y|H|x\rangle \leq 0 \qquad x \neq y$$

**Stoquastic**
i.e., "sign-problem free"

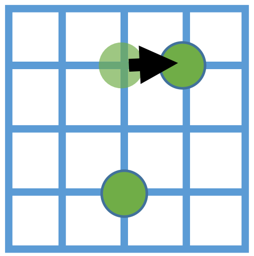
## Examples:

Particle in a potential



$$H = \frac{p^2}{2m} + V(\vec{x})$$

Hopping and interacting bosons



$$H = -\sum_{<ij>} (a_i^\dagger a_j + a_j^\dagger a_i) + V(\vec{n})$$

## What's the catch?

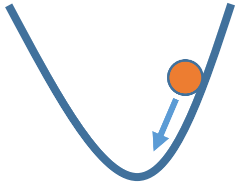
Quantum Monte Carlo can only be used to study **stoquastic Hamiltonians**

$$\langle x|H|x\rangle \in \mathbb{R} \qquad \langle y|H|x\rangle \leq 0 \qquad x \neq y$$

**Stoquastic**
i.e., "sign-problem free"

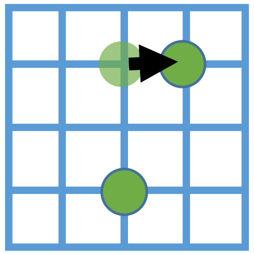
## Examples:

Particle in a potential



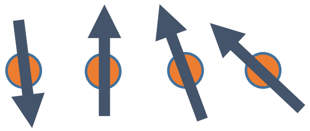
$$H = \frac{p^2}{2m} + V(\vec{x})$$

Hopping and interacting bosons



$$H = -\sum_{<ij>} (a_i^\dagger a_j + a_j^\dagger a_i) + V(\vec{n})$$

Quantum annealing Hamiltonians



$$H = -(1-s)\sum_i X_i + sV\left(\vec{Z}\right)$$

## How does it work?

QMC is based on a probabilistic representation of the Gibbs state

$$\rho = \frac{e^{-\beta H}}{Z(\beta)} \qquad Z(\beta) = \mathrm{Tr}(e^{-\beta H})$$

A collection of samples from a certain probability distribution associated with $\rho$ are sufficient to evaluate expectation values of observables.

# How does it work?

$$Z(\beta) = \mathrm{Tr}(e^{-\beta H})$$

Partition function at temperature $T = \beta^{-1}$

# How does it work?

$$Z(\beta) = \mathrm{Tr}(e^{-\beta H})$$

Partition function at temperature $T = \beta^{-1}$

$$= \mathrm{Tr}((e^{-\Delta H})^{M})$$

$\Delta \ll 1$ $\quad M = \beta\Delta^{-1}$

# How does it work?

$$Z(\beta) = \mathrm{Tr}(e^{-\beta H})$$

Partition function at temperature $T = \beta^{-1}$

$$= \mathrm{Tr}((e^{-\Delta H})^M)$$

$\Delta \ll 1$ $\quad M = \beta\Delta^{-1}$

$$\approx \sum_{z_1, z_2, \ldots z_M \in \{0,1\}^n} \langle z_1 | I - \Delta H | z_2 \rangle \langle z_2 | I - \Delta H | z_3 \rangle \ldots \langle z_M | I - \Delta H | z_1 \rangle$$

Insert complete sets of states

# How does it work?

$$Z(\beta) = \mathrm{Tr}(e^{-\beta H})$$

Partition function at temperature $T = \beta^{-1}$

$$= \mathrm{Tr}((e^{-\Delta H})^M)$$

$\Delta \ll 1$ $\quad M = \beta \Delta^{-1}$

$$\approx \sum_{z_1, z_2, \dots z_M \in \{0,1\}^n} \underbrace{\langle z_1 | I - \Delta H | z_2 \rangle \langle z_2 | I - \Delta H | z_3 \rangle \dots \langle z_M | I - \Delta H | z_1 \rangle}_{}$$

Insert complete sets of states

$\boldsymbol{p(z_1, z_2, \dots z_M) \geq 0}$ (use stoquasticity)

# How does it work?

$$Z(\beta) = \mathrm{Tr}(e^{-\beta H})$$

Partition function at temperature $T = \beta^{-1}$

$$= \mathrm{Tr}((e^{-\Delta H})^M)$$

$\Delta \ll 1$ $\quad M = \beta \Delta^{-1}$

$$\approx \sum_{z_1, z_2, \ldots z_M \in \{0,1\}^n} \underbrace{\langle z_1 | I - \Delta H | z_2 \rangle \langle z_2 | I - \Delta H | z_3 \rangle \ldots \langle z_M | I - \Delta H | z_1 \rangle}_{\boldsymbol{p(z_1, z_2, \ldots z_M) \geq 0}}$$

Insert complete sets of states

$\boldsymbol{p(z_1, z_2, \ldots z_M) \geq 0}$ (use stoquasticity)

QMC uses a classical Markov chain to equilibrate to the probability distribution proportional to $p(z_1, z_2, \ldots, z_M)$.

## How does it work?

$$Z(\beta) = \mathrm{Tr}(e^{-\beta H})$$

Partition function at temperature $T = \beta^{-1}$

$$= \mathrm{Tr}((e^{-\Delta H})^M)$$

$\Delta \ll 1 \qquad M = \beta\Delta^{-1}$

$$\approx \sum_{z_1,z_2,\dots z_M \in \{0,1\}^n} \underbrace{\langle z_1|I-\Delta H|z_2\rangle\langle z_2|I-\Delta H|z_3\rangle \dots \langle z_M|I-\Delta H|z_1\rangle}_{\boldsymbol{p(z_1,z_2,\dots z_M) \geq 0}}$$

Insert complete sets of states

(use stoquasticity)

QMC uses a classical Markov chain to equilibrate to the probability distribution proportional to $p(z_1, z_2, \dots, z_M)$.

Physical properties are computed as expectation values.

## In additional to empirical evidence, there is also complexity-theoretic evidence suggesting that stoquastic Hamiltonians may be easier to simulate.

**Local Hamiltonian problem:** Given a local Hamiltonian $H$ and two numbers $a < b$, decide if the ground energy of $H$ is $\leq a$ or $\geq b$.
(promised that one case holds).

**In additional to empirical evidence, there is also complexity-theoretic evidence suggesting that stoquastic Hamiltonians may be easier to simulate.**

**Local Hamiltonian problem:** Given a local Hamiltonian $H$ and two numbers $a < b$, decide if the ground energy of $H$ is $\leq a$ or $\geq b$.
(promised that one case holds).

The local Hamiltonian problem is QMA-complete in general. [Kitaev 99]

**In additional to empirical evidence, there is also complexity-theoretic evidence suggesting that stoquastic Hamiltonians may be easier to simulate.**

**Local Hamiltonian problem:** Given a local Hamiltonian $H$ and two numbers $a < b$, decide if the ground energy of $H$ is $\leq a$ or $\geq b$.
(promised that one case holds).

The local Hamiltonian problem is QMA-complete in general. [Kitaev 99]

For stoquastic local Hamiltonians it is StoqMA-complete (MA⊆StoqMA⊆AM)
[Bravyi, Divincenzo, Oliveira, Terhal 2006]

**In additional to empirical evidence, there is also complexity-theoretic evidence suggesting that stoquastic Hamiltonians may be easier to simulate.**

**Local Hamiltonian problem:** Given a local Hamiltonian $H$ and two numbers $a < b$, decide if the ground energy of $H$ is $\leq a$ or $\geq b$.
(promised that one case holds).

The local Hamiltonian problem is QMA-complete in general. [Kitaev 99]

For stoquastic local Hamiltonians it is StoqMA-complete (MA⊆StoqMA⊆AM)
[Bravyi, Divincenzo, Oliveira, Terhal 2006]

For classical local Hamiltonians it is NP-complete.

# Examples:

$$H = \sum_{1 \le i < j \le n} h(i,j)$$

| Model | $h(i,j)$ | LH problem | |
|---|---|---|---|
| Ising model | $h(i,j) = \alpha_{ij} Z_i Z_j$ | NP-complete | (Classical) |
| Transverse-field Ising model | $h(i,j) = \alpha_{ij} X_i X_j - \gamma_i Z_i - \gamma_j Z_j$ | StoqMA-complete [Bravyi, Hastings 2014] | (Stoquastic) |
| XY model | $h(i,j) = \alpha_{ij}(X_i X_j + Y_i Y_j)$ | QMA-complete [Cubitt Montanaro 2013] | (Quantum) |

## Examples:

$$H = \sum_{1 \le i < j \le n} h(i,j)$$

| Model | $h(i,j)$ | LH problem | |
|---|---|---|---|
| **Ising model** | $h(i,j) = \alpha_{ij} Z_i Z_j$ | **NP-complete** | **(Classical)** |
| **Transverse-field Ising model** | $h(i,j) = \alpha_{ij} X_i X_j - \gamma_i Z_i - \gamma_j Z_j$ | **StoqMA-complete** [Bravyi, Hastings 2014] | **(Stoquastic)** |
| **XY model** | $h(i,j) = \alpha_{ij} (X_i X_j + Y_i Y_j)$ | **QMA-complete** [Cubitt Montanaro 2013] | **(Quantum)** |

These examples illustrate three flavours of **intractable** constraint satisfaction problems.
(they represent all nontrivial possibilities within the framework of [Cubitt Montanaro 2013])

# Can we identify cases where Quantum Monte Carlo is provably efficient?

$$H = \sum_{1 \le i < j \le n} h(i,j)$$

| | | Approximate Ground energy | Approximate Partition Function |
|---|---|---|---|
| Ferromagnetic Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert Z_i Z_j$ | Trivial | |
| | | | |
| | | | |

## Can we identify cases where Quantum Monte Carlo is provably efficient?

$$H = \sum_{1 \le i < j \le n} h(i,j)$$

| | | Approximate Ground energy | Approximate Partition Function |
|---|---|---|---|
| Ferromagnetic Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert Z_i Z_j$ | Trivial | In BPP [Jerrum ,Sinclair 1989] |
| | | | |
| | | | |

# Can we identify cases where Quantum Monte Carlo is provably efficient?

$$H = \sum_{1 \le i < j \le n} h(i,j)$$

| | | Approximate Ground energy | Approximate Partition Function |
|---|---|---|---|
| Ferromagnetic Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert Z_i Z_j$ | Trivial | In BPP [Jerrum ,Sinclair 1989] |
| Ferromagnetic Transverse-field Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert X_i X_j - \gamma_i Z_i - \gamma_j Z_j$ | | |
| | | | |

# Can we identify cases where Quantum Monte Carlo is provably efficient?

$$H = \sum_{1 \le i < j \le n} h(i,j)$$

| | | Approximate Ground energy | Approximate Partition Function |
|---|---|---|---|
| Ferromagnetic Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert Z_i Z_j$ | **Trivial** | **In BPP** [Jerrum ,Sinclair 1989] |
| Ferromagnetic Transverse-field Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert X_i X_j - \gamma_i Z_i - \gamma_j Z_j$ | **In BPP** [Bravyi 2015] | **In BPP** [Bravyi 2015] |
| | | | |

# Can we identify cases where Quantum Monte Carlo is provably efficient?

$$H = \sum_{1 \le i < j \le n} h(i,j)$$

| | | Approximate Ground energy | Approximate Partition Function |
|---|---|---|---|
| Ferromagnetic Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert Z_i Z_j$ | Trivial | In BPP [Jerrum ,Sinclair 1989] |
| Ferromagnetic Transverse-field Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert X_i X_j - \gamma_i Z_i - \gamma_j Z_j$ | In BPP [Bravyi 2015] | In BPP [Bravyi 2015] |
| Ferromagnetic XY model | $h(i,j) = -\lvert\alpha_{ij}\rvert (X_i X_j + Y_i Y_j)$ | | |

# Can we identify cases where Quantum Monte Carlo is provably efficient?

$$H = \sum_{1 \le i < j \le n} h(i,j)$$

| | | Approximate Ground energy | Approximate Partition Function |
|---|---|---|---|
| Ferromagnetic Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert Z_i Z_j$ | Trivial | In BPP [Jerrum ,Sinclair 1989] |
| Ferromagnetic Transverse-field Ising model | $h(i,j) = -\lvert\alpha_{ij}\rvert X_i X_j - \gamma_i Z_i - \gamma_j Z_j$ | In BPP [Bravyi 2015] | In BPP [Bravyi 2015] |
| Ferromagnetic XY model | $h(i,j) = -\lvert\alpha_{ij}\rvert (X_i X_j + Y_i Y_j)$ | In BPP [This talk] | In BPP [This talk] |

**Open question :** Can QMC be used to efficiently simulate quantum adiabatic algorithms with stoquastic Hamiltonians?

$$H = -(1-s)\sum_i X_i + sV\left(\vec{Z}\right)$$

[Bravyi Terhal 2008]
[Hastings Freedman 2013]
[Crosson Harrow 2016]
[Jarret Jordan Lackey 2016]
…

# I. Results

# The Hamiltonian

**We consider Hamiltonians of the form**

$$H = \sum_{i<j} -b_{ij} X_i X_j + c_{ij} Y_i Y_j + \sum_{i=1}^{n} d_i (I + Z_i)$$

**Coefficients must satisfy**

$|b_{ij}|, |c_{ij}|, |d_i| \leq 1$ (sets energy scale)

$|c_{ij}| \leq b_{ij}$ (ensures stoquasticity)

# The Hamiltonian

**We consider Hamiltonians of the form**

$$H = \sum_{i<j} \underbrace{-b_{ij}X_iX_j + c_{ij}Y_iY_j}_{} + \sum_{i=1}^{n} d_i(I + Z_i)$$

$$\begin{pmatrix} 0 & 0 & 0 & -b_{ij} - c_{ij} \\ 0 & 0 & c_{ij} - b_{ij} & 0 \\ 0 & c_{ij} - b_{ij} & 0 & 0 \\ -b_{ij} - c_{ij} & 0 & 0 & 0 \end{pmatrix}$$

$$|c_{ij}| \leq b_{ij}$$

(ensures stoquasticity)

# The Hamiltonian

**We consider Hamiltonians of the form**

$$H = \sum_{i<j} \underbrace{-b_{ij} X_i X_j + c_{ij} Y_i Y_j}_{p_{ij}\left(Y_i Y_j - X_i X_j\right) + q_{ij}\left(-Y_i Y_j - X_i X_j\right)} + \sum_{i=1}^{n} d_i (I + Z_i)$$

$$p_{ij}, q_{ij} \geq 0$$

# The Hamiltonian

**We consider Hamiltonians of the form**

$$H = \sum_{i<j} -b_{ij} X_i X_j + c_{ij} Y_i Y_j + \sum_{i=1}^{n} d_i (I + Z_i)$$

**Special cases:**

| | | |
|---|---|---|
| $d_i = 0 \quad c_{ij} = 0$ | **Classical Ferromagnetic Ising model** |
| $c_{ij} = 0$ | **Ferromagnetic transverse-field Ising model** |
| $b_{ij} = 1 \quad c_{ij} = -1$ | **Ferromagnetic XY model** |
| $b_{ij} = 1 \quad c_{ij} = 1$ | **(name?)** |

# Approximating the partition function

**Definition ($\epsilon$-approximation)**

Write $a \approx^{\epsilon} b$ iff $e^{-\epsilon} b \leq a \leq e^{\epsilon} \mathrm{b}$

# Approximating the partition function

**Definition ($\epsilon$-approximation)**

Write $a \approx^{\epsilon} b$ iff $e^{-\epsilon} b \le a \le e^{\epsilon} \mathrm{b}$

An $\epsilon$-approximation of $Z(\beta)$ can be used to compute an estimate of the **free energy**

$$F = -\frac{1}{\beta} \log Z(\beta)$$

to within additive error $O(\epsilon\beta^{-1})$ and an estimate of the **ground energy** to within additive error $O((\epsilon + n)\beta^{-1})$.

# Polynomial-time approximation algorithm

**Theorem**

There exists a classical randomized algorithm which, given $H, \beta,$ and a precision parameter $\epsilon \in (0,1)$ outputs an estimate satisfying $Z \approx^{\epsilon} Z(\beta)$ with high probability.

The runtime of the algorithm is $poly(n, \beta, \epsilon^{-1})$

**As a corollary we obtain an efficient algorithm to approximate the free energy and the ground energy to a given additive error.**

# Polynomial-time approximation algorithm

**Theorem**
There exists a classical randomized algorithm which, given $H, \beta,$ and a precision parameter $\epsilon \in (0,1)$ outputs an estimate satisfying $Z \approx^{\epsilon} Z(\beta)$ with high probability.

The runtime of the algorithm is $poly(n, \beta, \epsilon^{-1}) = \tilde{O}(n^{115}(1+\beta^{46})\epsilon^{-25})$.

**As a corollary we obtain an efficient algorithm to approximate the free energy and the ground energy to a given additive error.**

# Polynomial-time approximation algorithm

**Theorem**
There exists a classical randomized algorithm which, given $H, \beta,$ and a precision parameter $\epsilon \in (0,1)$ outputs an estimate satisfying $Z \approx^{\epsilon} Z(\beta)$ with high probability.

The runtime of the algorithm is $poly(n, \beta, \epsilon^{-1}) = \tilde{O}(n^{115}(1 + \beta^{46})\epsilon^{-25})$.

**As a corollary we obtain an efficient algorithm to approximate the free energy and the ground energy to a given additive error.**

**The algorithm is not practical.**

# Polynomial-time approximation algorithm

**Theorem**
There exists a classical randomized algorithm which, given $H, \beta,$ and a precision parameter $\epsilon \in (0,1)$ outputs an estimate satisfying $Z \approx^{\epsilon} Z(\beta)$ with high probability.

The runtime of the algorithm is $poly(n, \beta, \epsilon^{-1}) = \tilde{O}(n^{115}(1 + \beta^{46})\epsilon^{-25})$.

**As a corollary we obtain an efficient algorithm to approximate the free energy and the ground energy to a given additive error.**

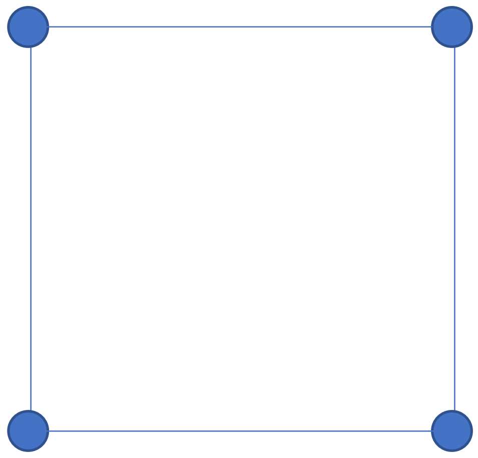
**The algorithm is not practical.**

**The proof is based on a reduction to counting perfect matchings…**

# II. Perfect matchings

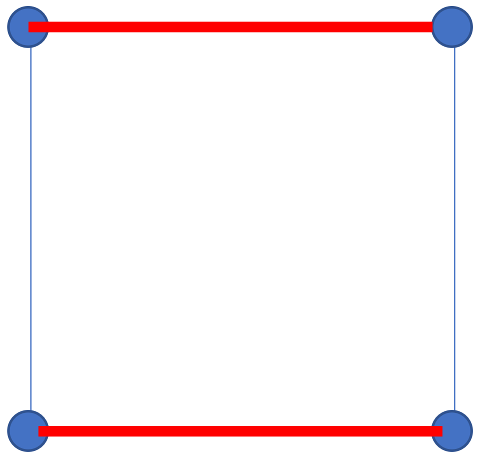
A **perfect matching** of a graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that every vertex is incident to exactly one edge in $M$

**Example:**



A **perfect matching** of a graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that every vertex is incident to exactly one edge in $M$

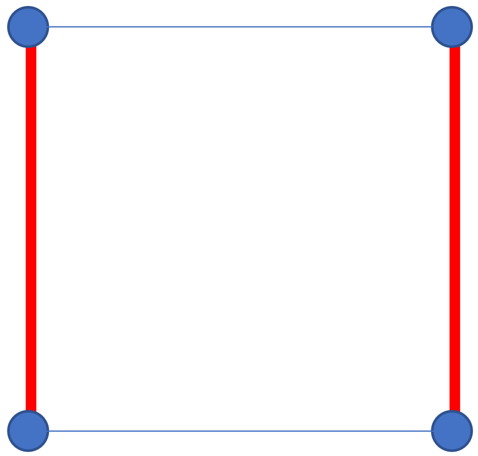
**Example:**



The red edges are a perfect matching

A **perfect matching** of a graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that every vertex is incident to exactly one edge in $M$

**Example:**



The red edges are a perfect matching

Now suppose the graph has edge weights $\{w_e\}_{e \in E}$. Each perfect matching $M$ is assigned weight

$$\prod_{e \in M} w_e$$

Now suppose the graph has edge weights $\{w_e\}_{e\in E}$. Each perfect matching $M$ is assigned weight

$$\prod_{e\in M} w_e$$

**Perfect matching sum:**

$$\text{PerfMatch}(G) = \sum_{\text{Perfect matchings M}} \prod_{e\in M} w_e$$

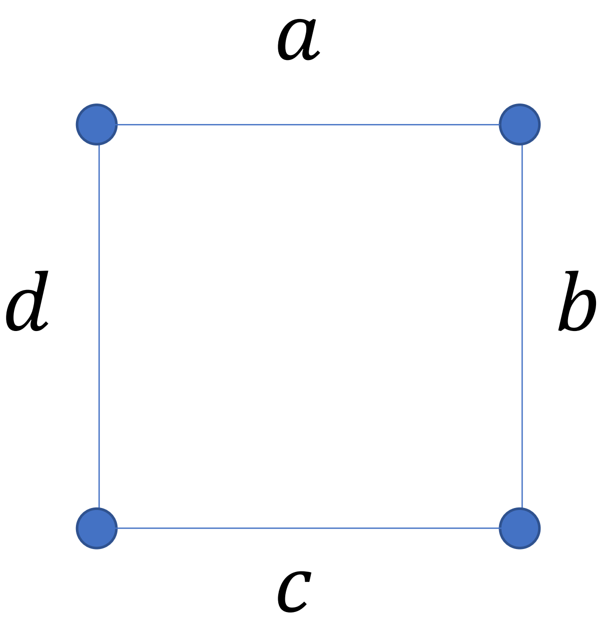
Now suppose the graph has edge weights $\{w_e\}_{e\in E}$. Each perfect matching $M$ is assigned weight

$$\prod_{e\in M} w_e$$

**Perfect matching sum:**

$$\text{PerfMatch}(G) = \sum_{\text{Perfect matchings M}} \prod_{e\in M} w_e$$

**Example:**




$$\text{PerfMatch}(G) = ac + bd$$

A **nearly perfect matching** of a graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that every vertex is incident to exactly one edge in $M$, **except for 2 vertices which are untouched.**

A **nearly perfect matching** of a graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that every vertex is incident to exactly one edge in $M$, **except for 2 vertices which are untouched.**

**Nearly perfect matching sum:**

$$\mathrm{NearPerfMatch}(G) = \sum_{\substack{\text{Nearly} \\ \text{Perfect matchings M}}} \prod_{e \in M} w_e$$

**Suppose *G* is a graph with nonnegative edge weights.**

| | **Exactly compute** $\mathrm{PerfMatch}(G)$ | **$\epsilon$-approximation to** $\mathrm{PerfMatch}(G)$ |
|---|---|---|
| **Planar graphs:** | **In P**<br>Fisher, Kasteleyn, Temperley algorithm | |
| **Bipartite graphs:**<br>(permanent of nonnegative matrix) | | |
| **General graphs:** | | |

Suppose $G$ is a graph with nonnegative edge weights.

| | Exactly compute $\mathrm{PerfMatch}(G)$ | $\epsilon$-approximation to $\mathrm{PerfMatch}(G)$ |
|---|---|---|
| Planar graphs: | **In P**<br>Fisher, Kasteleyn, Temperley algorithm | **In P** |
| Bipartite graphs: (permanent of nonnegative matrix) | | |
| General graphs: | | |

Suppose $G$ is a graph with nonnegative edge weights.

| | Exactly compute $\text{PerfMatch}(G)$ | $\epsilon$-approximation to $\text{PerfMatch}(G)$ |
|---|---|---|
| Planar graphs: | In P<br>Fisher, Kasteleyn, Temperley algorithm | In P |
| Bipartite graphs: (permanent of nonnegative matrix) | #*P*-hard<br>[Valiant 1979] | In BPP<br>[Jerrum, Sinclair, Vigoda 2004] |
| General graphs: | | |

**Suppose $G$ is a graph with nonnegative edge weights.**

| | **Exactly compute** PerfMatch$(G)$ | **$\epsilon$-approximation to** PerfMatch$(G)$ |
|---|---|---|
| **Planar graphs:** | **In P**<br>Fisher, Kasteleyn, Temperley algorithm | **In P** |
| **Bipartite graphs:** (permanent of nonnegative matrix) | **#*P*-hard**<br>[Valiant 1979] | **In BPP**<br>[Jerrum, Sinclair, Vigoda 2004] |
| **General graphs:** | **#*P*-hard** | |

**Suppose $G$ is a graph with nonnegative edge weights.**

| | **Exactly compute** PerfMatch$(G)$ | **$\epsilon$-approximation to** PerfMatch$(G)$ |
|---|---|---|
| **Planar graphs:** | **In P**<br>Fisher, Kasteleyn, Temperley algorithm | **In P** |
| **Bipartite graphs:**<br>(permanent of nonnegative matrix) | **#*P*-hard**<br>[Valiant 1979] | **In BPP**<br>[Jerrum, Sinclair, Vigoda 2004] |
| **General graphs:** | **#*P*-hard** | [Jerrum Sinclair 1989]<br>**Algorithm with runtime**<br>$\text{poly}(\lvert V \rvert, \epsilon,^{-1}\ R)$<br>$R = \dfrac{\text{NearPerfMatch}(G)}{\text{PerfMatch}(G)}$ |

# III. Algorithm

# Reduction to perfect matchings

**Theorem**

There is an (efficiently computable) graph $G$ with positive edge weights, such that

$$Z(\beta) \approx^{\epsilon} \text{PerfMatch}(G)$$

and

$$\frac{\text{NearPerfMatch}(G)}{\text{PerfMatch}(G)} = O(\text{poly}(\beta, n, \epsilon^{-1}))$$

**We then use [Jerrum, Sinclair 1989] which gives an efficient algorithm for approximating the perfect matching sum.**

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(G_J \dots G_2 G_1) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(G_J \dots G_2 G_1) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

Each $G_j$ satisfies

$$G_j = e^{-sh+O(s^2)} \qquad s > 0$$

where $h$ is one of the terms in the Hamiltonian

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(G_J \dots G_2 G_1) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

Each $G_j$ satisfies

$$G_j = e^{-sh+O(s^2)} \qquad s > 0$$

where $h$ is one of the terms in the Hamiltonian

$$h = Y_i Y_j - X_i X_j, \quad \text{or} \quad h = -Y_i Y_j - X_i X_j, \quad \text{or} \quad h = \pm(I + Z_i)$$

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(G_J \ldots G_2 G_1) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

Each $G_j$ satisfies

$$G_j = e^{-sh+O(s^2)} \qquad s > 0$$

where $h$ is one of the terms in the Hamiltonian

$$h = Y_i Y_j - X_i X_j, \quad \text{or} \quad h = -Y_i Y_j - X_i X_j, \quad \text{or} \quad h = \pm(I + Z_i)$$

**The resulting $G_j$ are very special gates…**

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(G_J \ldots G_2 G_1) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

Each $G_J$ is from the gate set containing 1-qubit gates

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} t & 0 \\ 0 & 1 \end{pmatrix} \qquad t > 0$$

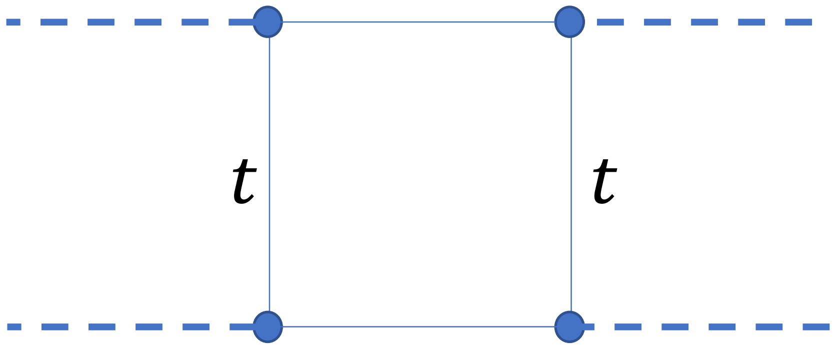
and two qubit gates

$$\begin{pmatrix} 1+t^2 & 0 & 0 & t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t & 0 & 0 & 1 \end{pmatrix} \qquad t > 0$$

**"Matchgates"**

Let $\Gamma$ be a a weighted graph with special input and output edges ($k$ of each, say)

**Example:**




**Input edges** **Output edges**

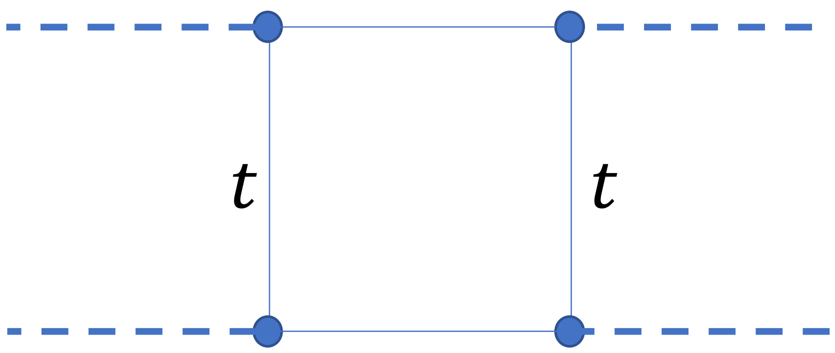
Let $\Gamma$ be a a weighted graph with special input and output edges ($k$ of each, say)

**We say $\Gamma$ implements a $k$-qubit operator $G$ if**

$$\langle y|G|x\rangle = \text{PerfMatch}(\Gamma_{xy})$$

$\Gamma_{xy}$ = remove input edges with $x_i = 0$ and output edges with $y_i = 0$. Require that a perfect matching includes the remaining external edges.

**Example:**




**Input edges**

**Output edges**

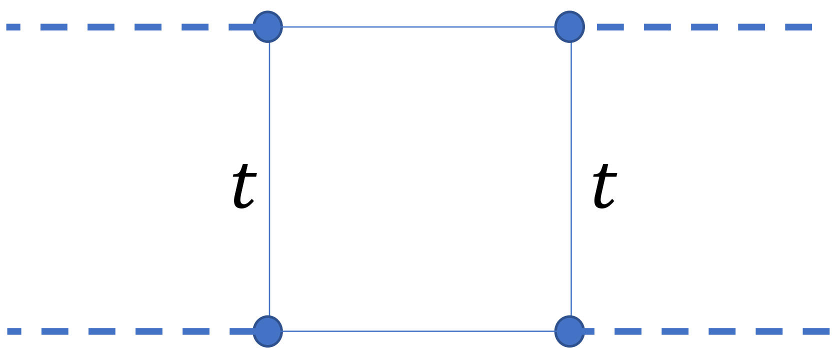
Let $\Gamma$ be a a weighted graph with special input and output edges ($k$ of each, say)

**We say $\Gamma$ implements a $k$-qubit operator $G$ if**

$$\langle y|G|x\rangle = \mathrm{PerfMatch}(\Gamma_{xy})$$

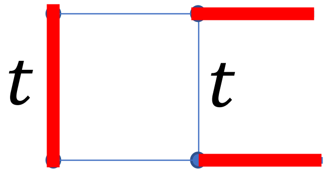
$\Gamma_{xy}$ = remove input edges with $x_i = 0$ and output edges with $y_i = 0$. Require that a perfect matching includes the remaining external edges.

**Example:**




**Input edges** **Output edges**

$\boldsymbol{x = 00, y = 11}$




$\boldsymbol{\langle 11|G|00\rangle = t}$

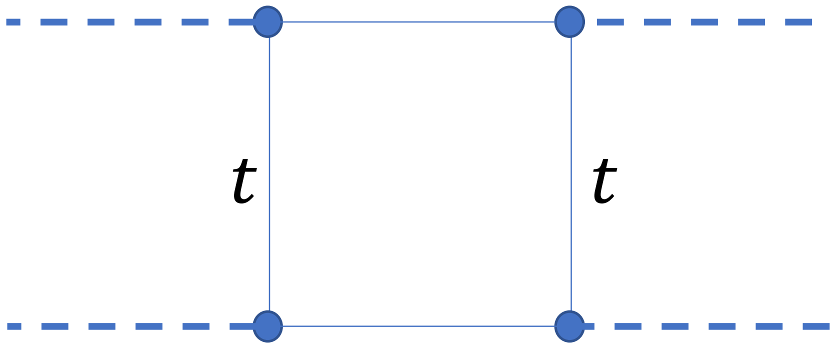
Let $\Gamma$ be a a weighted graph with special input and output edges ($k$ of each, say)

**We say $\Gamma$ implements a $k$-qubit operator $G$ if**

$$\langle y|G|x\rangle = \mathrm{PerfMatch}(\Gamma_{xy})$$

$\Gamma_{xy}$ = remove input edges with $x_i = 0$ and output edges with $y_i = 0$. Require that a perfect matching includes the remaining external edges.

**Example:**

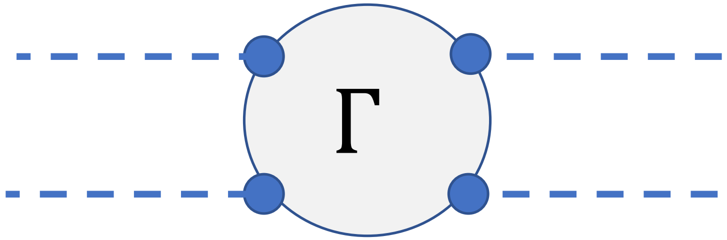



**Input edges**

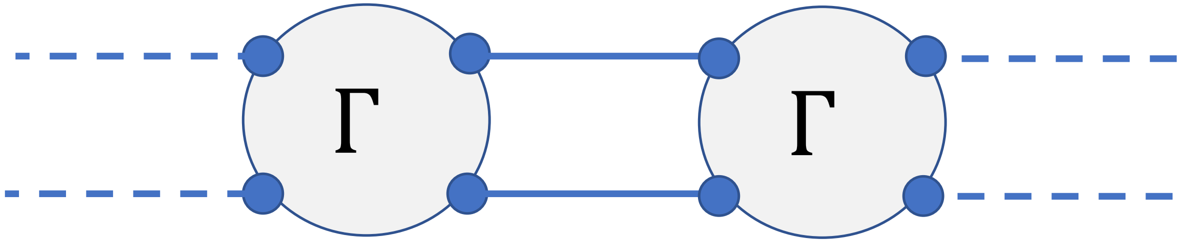
**Output edges**

$$G = \begin{pmatrix} 1+t^2 & 0 & 0 & t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t & 0 & 0 & 1 \end{pmatrix}$$

# Matchgates compose nicely

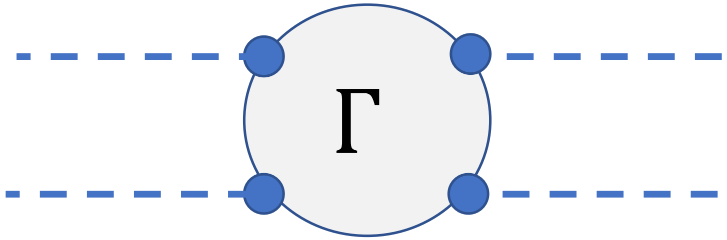



**Implements a 2 qubit gate $\boldsymbol{G}$**

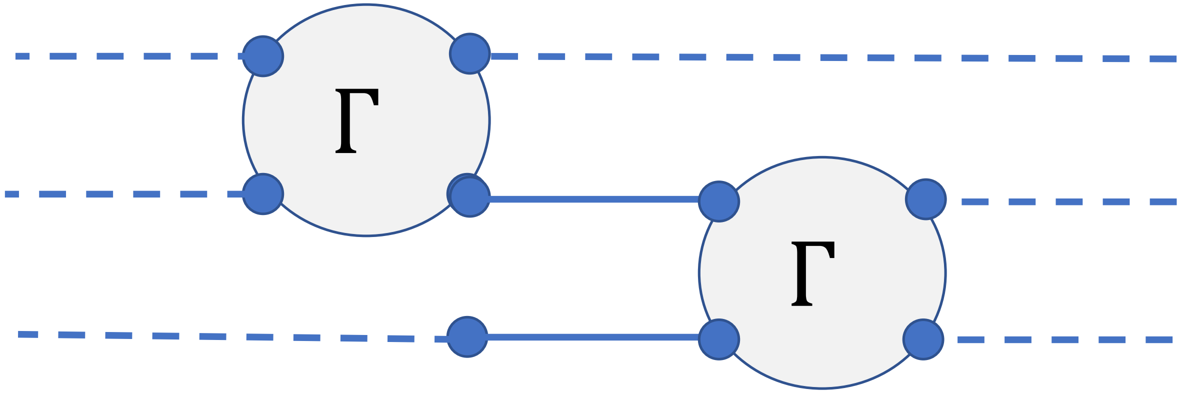



**Implements $\boldsymbol{G}^2$**

# Matchgates compose nicely

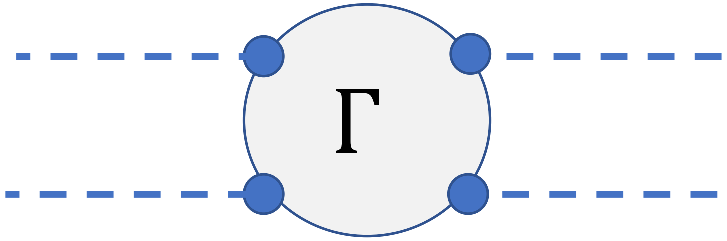



**Implements a 2 qubit gate $\boldsymbol{G}$**

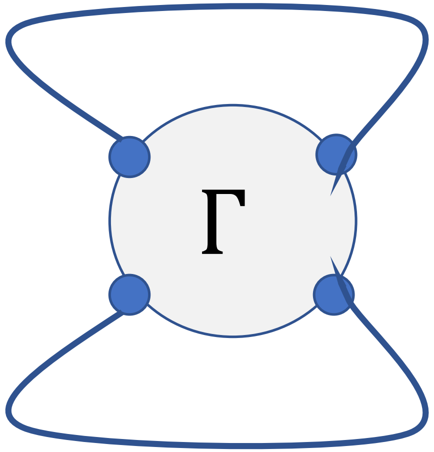



**Implements $\boldsymbol{G_{12} G_{23}}$**

# Matchgates compose nicely




**Implements a 2 qubit gate $\boldsymbol{G}$**




**Implements $\mathbf{Tr}(\boldsymbol{G})$**

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(G_J \dots G_2 G_1) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

Each $G_J$ is a matchgate.

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(\underbrace{G_J \ldots G_2 G_1}) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

Matchgate implementable with $n$ input edges and $n$ output edges

Each $G_J$ is a matchgate.

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(\underbrace{G_J \dots G_2 G_1}) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

Matchgate implementable with $n$ input edges and $n$ output edges

Each $G_J$ is a matchgate.

Trace is a matchgate with no external edges, i.e., a perfect matching sum.

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(G_J \ldots G_2 G_1) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

Matchgate implementable with $n$ input edges and $n$ output edges

Each $G_J$ is a matchgate.

Trace is a matchgate with no external edges, i.e., a perfect matching sum. **(nonplanar and nonbipartite)**

**Proof sketch:**

Start with a Trotter-Suzuki style approximation

$$\mathrm{Tr}\left(e^{-\beta H}\right) \approx^{\epsilon} \mathrm{Tr}(G_J \dots G_2 G_1) \qquad J = \mathrm{poly}(n, \beta, \epsilon^{-1})$$

Matchgate implementable with $n$ input edges and $n$ output edges

Each $G_J$ is a matchgate.

Trace is a matchgate with no external edges, i.e., a perfect matching sum. **(nonplanar and nonbipartite)**

This gives first part of theorem:

$$Z(\beta) \approx^{\epsilon} \mathrm{PerfMatch}(G)$$

# Bounding $\text{NearPerfMatch}(G)$

We need to show:

$$\frac{\text{NearPerfMatch}(G)}{\text{PerfMatch}(G)} = O(\text{poly}(\beta, n, \epsilon^{-1}))$$

Recall that a nearly perfect matching is like a perfect matching but with 2 vertices unmatched.

# Bounding $\text{NearPerfMatch}(G)$

We need to show:

$$\frac{\text{NearPerfMatch}(G)}{\text{PerfMatch}(G)} = O(\text{poly}(\beta, n, \epsilon^{-1}))$$

Recall that a nearly perfect matching is like a perfect matching but with 2 vertices unmatched.

$$\text{NearPerfMatch}(G) = \sum_{u,v \in G} \Omega_{u,v}$$

$\Omega_{u,v} =$ sum of nearly perfect matchings with $u, v$ unmatched.

# Bounding $\text{NearPerfMatch}(G)$

We need to show: $$\frac{\text{NearPerfMatch}(G)}{\text{PerfMatch}(G)} = O(\text{poly}(\beta, n, \epsilon^{-1}))$$

Recall that a nearly perfect matching is like a perfect matching but with 2 vertices unmatched.

$$\text{NearPerfMatch}(G) = \sum_{u,v\in G} \Omega_{u,v}$$

$\Omega_{u,v} =$ sum of nearly perfect matchings with $u, v$ unmatched.

**To complete the proof we show that**

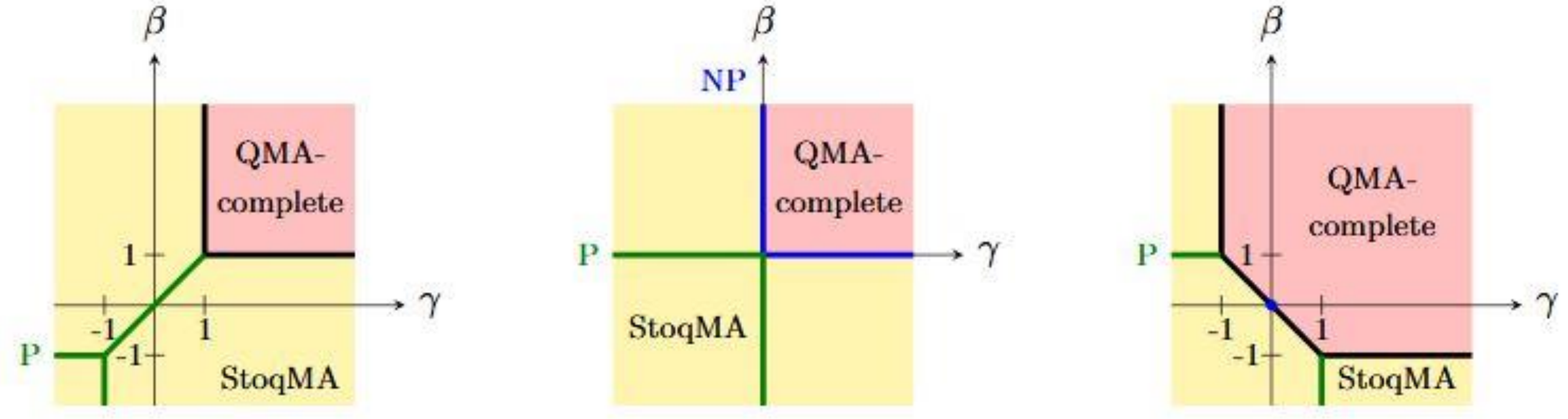
$$\frac{\Omega_{u,v}}{\text{PerfMatch}(G)} \approx \frac{\text{Tr}\left(G_J G_{J-1} \dots G_j O G_{j-1} G_{j-2} \dots G_i P G_{i-1} G_{i-2} \dots G_2 G_1\right)}{\text{Tr}\left(G_J \dots G_2 G_1\right)} = O(1)$$

**Imaginary time spin-spin correlation function**

# Open questions

Can QMC be used to efficiently simulate quantum adiabatic algorithms with stoquastic Hamiltonians?

Other models? See e.g., **[Piddock Montanaro 2015]**:




(a) $H = -XX + \beta YY + \gamma ZZ$ (b) $H = \beta YY + \gamma ZZ$ (c) $H = XX + \beta YY + \gamma ZZ$

**IBM is hiring for postdoctoral and research staff member positions in the theory of quantum computing.**

**Job ads:**
https://quantumexperience.ng.bluemix.net/qx/community/question?questionId=4ee83621979d8391db8c95523e36ebd6&channel=news

**Email:** dngosset@us.ibm.com

# IBM Q Awards: Att: Students, Profs & Developers

- **IBM Teach Me Quantum Award** – $10,000: Best university-level course-materials for a lecture series incorporating the IBM Q Experience and QISKit. *(Submissions close 15 November 2018)*

- **IBM Q Best Paper Award –** $2,500 and IBM lab invite: Highest-impact scientific paper by a master's degree or PhD student or postdoctoral researcher that uses the IBM Q Experience and QISKit as a tool to achieve the presented results. *(Submissions close 15 July 2018)*

- **Teach Me QISKit Award** – $1,000: Best interactive self-paced tutorial using QISKit and the IBM Q Experience. *(Submissions close 31 March 2018)*

- **QISKit Developer Challenge** – $5,000: Best solution to a specific challenge called: "Optimize to the Max." *(Submissions close 15 May 2018)*

**Submissions Open**
**15 January**

**bit.ly/ibmqawards**