

Quantum Fourier transforms and hidden subgroup problems

Sonya Berg

UC Davis

February 27, 2008

Outline

- 1 Quantum computation preliminaries
- 2 Quantum Fourier transforms
- 3 Hidden subgroup problems

Qubits

A **qubit** is the basic unit of information in quantum computing.

Definition: n -qubit state

An n -qubit state is a normalized vector in $(\mathbb{C}^2)^{\otimes n} \cong \mathbb{C}^{2^n}$.

An orthonormal basis for \mathbb{C}^{2^n} is given by n -bit strings, thought of as the numbers 0 through $2^n - 1$ in binary. This is called the **computational basis**.

Example: two-qubit state

$$|\psi\rangle = \frac{1}{\sqrt{2}} |00\rangle - \frac{1}{\sqrt{6}} |01\rangle + \frac{1}{\sqrt{3}} |11\rangle$$

Qubits

A **qubit** is the basic unit of information in quantum computing.

Definition: n -qubit state

An n -qubit state is a normalized vector in $(\mathbb{C}^2)^{\otimes n} \cong \mathbb{C}^{2^n}$.

An orthonormal basis for \mathbb{C}^{2^n} is given by n -bit strings, thought of as the numbers 0 through $2^n - 1$ in binary. This is called the **computational basis**.

Example: two-qubit state

$$|\psi\rangle = \frac{1}{\sqrt{2}} |00\rangle - \frac{1}{\sqrt{6}} |01\rangle + \frac{1}{\sqrt{3}} |11\rangle$$

Quantum gate

Definition: quantum gate

A **quantum gate** acting on an n -qubit state is a unitary matrix $U \in \mathcal{U}(2^n, \mathbb{C})$.

Example: Hadamard and bit flip gates

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Definition: gate approximation

A gate U' approximates U with error less than ϵ if

$$\|U - U'\| < \epsilon.$$

Quantum gate

Definition: quantum gate

A **quantum gate** acting on an n -qubit state is a unitary matrix $U \in \mathcal{U}(2^n, \mathbb{C})$.

Example: Hadamard and bit flip gates

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Definition: gate approximation

A gate U' approximates U with error less than ϵ if

$$\|U - U'\| < \epsilon.$$

Controlled gates

Definition: controlled- U gate

Let $U \in \mathcal{U}(2^k, \mathbb{C})$ be a quantum gate acting on k -qubit states. The **controlled- U** gate $cU \in \mathcal{U}(2^{k+1}, \mathbb{C})$ is defined by

$$\begin{aligned} cU|0\rangle|\psi\rangle &= |0\rangle|\psi\rangle, \\ cU|1\rangle|\psi\rangle &= |1\rangle U|\psi\rangle. \end{aligned}$$

Example: controlled-NOT gate

$$cX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Universal gate set

Definition: universal gate set

A set S of gates is **universal** for quantum computation if for any unitary U and any $\epsilon > 0$, there exists a sequence $U_1, \dots, U_{k(\epsilon)}$ of elements of S so that

$$\|U_1 \cdots U_{k(\epsilon)} - U\| < \epsilon.$$

Theorem: universal gate set

The set of all single qubit gates and the controlled gate cX is a universal set for quantum computation. These gates are called **elementary gates**.

Universal gate set

Definition: universal gate set

A set S of gates is **universal** for quantum computation if for any unitary U and any $\epsilon > 0$, there exists a sequence $U_1, \dots, U_{k(\epsilon)}$ of elements of S so that

$$\|U_1 \cdots U_{k(\epsilon)} - U\| < \epsilon.$$

Theorem: universal gate set

The set of all single qubit gates and the controlled gate cX is a universal set for quantum computation. These gates are called **elementary gates**.

Quantum measurement

Let

$$|\psi\rangle = \sum_{x=0}^{2^n-1} c_x |x\rangle = \sum_{v=0}^{2^k-1} |\alpha_v\rangle |v\rangle$$

be an n -qubit state.

k -qubit measurement

A k -qubit **measurement** of $|\psi\rangle$ is the map

$$|\psi\rangle \mapsto \frac{|\alpha_v\rangle |v\rangle}{\sqrt{\langle \alpha_v | \alpha_v \rangle}}$$

with probability $\langle \alpha_v | \alpha_v \rangle$. The name of v is said to be **observed**.

Example: measurement

Suppose we start with the two-qubit state

$$|\psi\rangle = \frac{1}{\sqrt{2}} |00\rangle - \frac{1}{\sqrt{6}} |01\rangle + \frac{1}{\sqrt{3}} |11\rangle.$$

Measuring both qubits, our outcome is

$|00\rangle$ with probability $1/2$,
 $|01\rangle$ with probability $1/6$,
 or $|11\rangle$ with probability $1/3$.

Measuring only the first qubit, our outcome is

$|0\rangle \left(\frac{\sqrt{3}}{2} |0\rangle - \frac{1}{2} |1\rangle \right)$ with probability $1/2 + 1/6 = 2/3$,
 or $|1\rangle$ with probability $1/3$.

Quantum algorithms and efficiency

- Let B_n denote the set of all n -qubit states.
- A **computational problem** $A = \{A_n\}$ is a collection of maps $A_n: P_n \subseteq B_n \rightarrow \{0, 1\}^n$.
- A **quantum algorithm** is a family of quantum gates $\{U_{2^n}\}$.
- A quantum algorithm **solves** A if for all $p \in P_n$, with probability at least $2/3$, $A_n(p)$ is equal to $U_{2^n}(p)$ followed by quantum measurement.
- A quantum algorithm is **efficient** if each U_{2^n} factors as $\text{poly}(n)$ elementary gates. Moreover, we need a technical uniformity condition requiring that the elementary gates are themselves classically computable in polynomial time.

Quantum algorithms and efficiency

- Let B_n denote the set of all n -qubit states.
- A **computational problem** $A = \{A_n\}$ is a collection of maps $A_n: P_n \subseteq B_n \rightarrow \{0, 1\}^n$.
- A **quantum algorithm** is a family of quantum gates $\{U_{2^n}\}$.
- A quantum algorithm **solves** A if for all $p \in P_n$, with probability at least $2/3$, $A_n(p)$ is equal to $U_{2^n}(p)$ followed by quantum measurement.
- A quantum algorithm is **efficient** if each U_{2^n} factors as $\text{poly}(n)$ elementary gates. Moreover, we need a technical uniformity condition requiring that the elementary gates are themselves classically computable in polynomial time.

Quantum algorithms and efficiency

- Let B_n denote the set of all n -qubit states.
- A **computational problem** $A = \{A_n\}$ is a collection of maps $A_n: P_n \subseteq B_n \rightarrow \{0, 1\}^n$.
- A **quantum algorithm** is a family of quantum gates $\{U_{2^n}\}$.
- A quantum algorithm **solves** A if for all $p \in P_n$, with probability at least $2/3$, $A_n(p)$ is equal to $U_{2^n}(p)$ followed by quantum measurement.
- A quantum algorithm is **efficient** if each U_{2^n} factors as $\text{poly}(n)$ elementary gates. Moreover, we need a technical uniformity condition requiring that the elementary gates are themselves classically computable in polynomial time.

Quantum algorithms and efficiency

- Let B_n denote the set of all n -qubit states.
- A **computational problem** $A = \{A_n\}$ is a collection of maps $A_n: P_n \subseteq B_n \rightarrow \{0, 1\}^n$.
- A **quantum algorithm** is a family of quantum gates $\{U_{2^n}\}$.
- A quantum algorithm **solves** A if for all $p \in P_n$, with probability at least $2/3$, $A_n(p)$ is equal to $U_{2^n}(p)$ followed by quantum measurement.
- A quantum algorithm is **efficient** if each U_{2^n} factors as $\text{poly}(n)$ elementary gates. Moreover, we need a technical uniformity condition requiring that the elementary gates are themselves classically computable in polynomial time.

Definition: Fourier transform

Let G be a finite group. Let \widehat{G} be a maximal collection of finite-dimensional inequivalent irreducible complex representations of G . Let d_ρ be the dimension of $\rho \in \widehat{G}$.

Definition: Fourier transform

A **Fourier transform** for G is an isomorphism

$$F_G: \mathbb{C}[G] \rightarrow \bigoplus_{\rho \in \widehat{G}} \text{End}(V)$$

$$g \mapsto \bigoplus_{\rho} \sqrt{\frac{d_\rho}{|G|}} \rho(g).$$

The Fourier transform is unitary

Because $\sum_{\rho} d_{\rho}^2 = |G|$, $F_G(g)$ may be identified with a vector of length $|G|$, with the notation:

$$F_G(g) = \sum_{\rho} \sqrt{\frac{d_{\rho}}{|G|}} \sum_{i,j=0}^{d_{\rho}-1} |\rho(g), i, j\rangle.$$

To be a candidate for a quantum gate, F_G must be unitary.

Proposition: F_G is unitary

If all the $\rho(g)$ are unitary then F_G is unitary.

This proposition is a corollary of Schur's lemma.

The Fourier transform is unitary

Because $\sum_{\rho} d_{\rho}^2 = |G|$, $F_G(g)$ may be identified with a vector of length $|G|$, with the notation:

$$F_G(g) = \sum_{\rho} \sqrt{\frac{d_{\rho}}{|G|}} \sum_{i,j=0}^{d_{\rho}-1} |\rho(g), i, j\rangle.$$

To be a candidate for a quantum gate, F_G must be unitary.

Proposition: F_G is unitary

If all the $\rho(g)$ are unitary then F_G is unitary.

This proposition is a corollary of Schur's lemma.

Example: cyclic group

Let $G = \mathbb{Z}/(n) = \langle g \rangle$. There are n one-dimensional irreps ρ_j of G , where $\rho_j(g) = \omega^j$ with $\omega = e^{2\pi i/n}$.

Thus, $F_{\mathbb{Z}/(n)}$ is

$$\frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)^2} \end{pmatrix}.$$

This is the well-known discrete Fourier transform.

Example: dihedral group

Let $G = D_{2n} = \langle x, y \mid y^2 = x^{2n} = 1, yxy = x^{-1} \rangle$. There are 4 one-dimensional irreps of D_{2n} which map the generators to ± 1 , and $2^{n-1} - 1$ two-dimensional irreps of the form

$$x \mapsto \begin{pmatrix} \omega^k & 0 \\ 0 & \omega^{-k} \end{pmatrix}, \quad y \mapsto \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

For example, when $n = 2$, F_{D_4} is

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & \omega & \omega^2 & \omega^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \omega^3 & \omega^2 & \omega \\ 0 & 0 & 0 & 0 & 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^3 & \omega^2 & \omega & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \end{pmatrix}.$$

Example: dihedral group

Let $G = D_{2n} = \langle x, y \mid y^2 = x^{2^n} = 1, yxy = x^{-1} \rangle$. There are 4 one-dimensional irreps of D_{2n} which map the generators to ± 1 , and $2^{n-1} - 1$ two-dimensional irreps of the form

$$x \mapsto \begin{pmatrix} \omega^k & 0 \\ 0 & \omega^{-k} \end{pmatrix}, \quad y \mapsto \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

For example, when $n = 2$, F_{D_4} is

$$\begin{pmatrix} \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} \\ \frac{1}{2} & \frac{1}{2}\omega & \frac{1}{2}\omega^2 & \frac{1}{2}\omega^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2}\omega^3 & \frac{1}{2}\omega^2 & \frac{1}{2}\omega \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2}\omega & \frac{1}{2}\omega^2 & \frac{1}{2}\omega^3 \\ \frac{1}{2} & \frac{1}{2}\omega^3 & \frac{1}{2}\omega^2 & \frac{1}{2}\omega & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \end{pmatrix}$$

Example: symmetric group

Let $G = S_3$. There are three irreps of G : trivial, sign, and standard. When written in Young's orthonormal form, the actions of the generators are

$$(12) \mapsto \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad (23) \mapsto \frac{1}{2} \begin{pmatrix} -1 & \sqrt{3} \\ \sqrt{3} & 1 \end{pmatrix}.$$

Therefore, F_{S_3} is

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 \\ 0 & 0 & \sqrt{3} & \sqrt{3} & -\sqrt{3} & -\sqrt{3} \\ 0 & 0 & \sqrt{3} & -\sqrt{3} & \sqrt{3} & -\sqrt{3} \\ 1 & -1 & 1 & -1 & -1 & 1 \end{pmatrix}.$$

Example: symmetric group

Let $G = S_3$. There are three irreps of G : trivial, sign, and standard. When written in Young's orthonormal form, the actions of the generators are

$$(12) \mapsto \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad (23) \mapsto \frac{1}{2} \begin{pmatrix} -1 & \sqrt{3} \\ \sqrt{3} & 1 \end{pmatrix}.$$

Therefore, F_{S_3} is

$$\begin{pmatrix} \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & -\frac{1}{2\sqrt{3}} & -\frac{1}{2\sqrt{3}} & -\frac{1}{2\sqrt{3}} & -\frac{1}{2\sqrt{3}} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & \frac{1}{2\sqrt{3}} & -\frac{1}{2\sqrt{3}} & -\frac{1}{2\sqrt{3}} & \frac{1}{2\sqrt{3}} \end{pmatrix}.$$

General technique for efficiently computing F_G

Inductively, let H be a subgroup of G of index n .

- Choose a basis so that $\rho|_H(h)$ is block diagonal for all h .
- Choose a set of coset representatives g_1, \dots, g_n for which multiplication by $\rho(g_k)$ is fast in our chosen basis.
- Rewrite each element $g = g_k h$.

- Compute F_H to get $\sum_{\rho' \in \widehat{H}} \sum_{i,j=0}^{d_{\rho'}-1} |\rho'(h), i, j\rangle$.

- Match up $\rho'(h)$ in blocks to obtain $\sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle$.

- Multiply by $\rho(g_k)$ to get what we want:

$$\sum_{\rho} \rho(g_k) \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle = \sum_{\rho} \sum_{i,j=0}^{d_\rho-1} |\rho(g), i, j\rangle.$$

General technique for efficiently computing F_G

Inductively, let H be a subgroup of G of index n .

- Choose a basis so that $\rho|_H(h)$ is block diagonal for all h .
- Choose a set of coset representatives g_1, \dots, g_n for which multiplication by $\rho(g_k)$ is fast in our chosen basis.
- Rewrite each element $g = g_k h$.

- Compute F_H to get $\sum_{\rho' \in \widehat{H}} \sum_{i,j=0}^{d_{\rho'}-1} |\rho'(h), i, j\rangle$.

- Match up $\rho'(h)$ in blocks to obtain $\sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle$.

- Multiply by $\rho(g_k)$ to get what we want:

$$\sum_{\rho} \rho(g_k) \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle = \sum_{\rho} \sum_{i,j=0}^{d_\rho-1} |\rho(g), i, j\rangle.$$

General technique for efficiently computing F_G

Inductively, let H be a subgroup of G of index n .

- Choose a basis so that $\rho|_H(h)$ is block diagonal for all h .
- Choose a set of coset representatives g_1, \dots, g_n for which multiplication by $\rho(g_k)$ is fast in our chosen basis.
- Rewrite each element $g = g_k h$.

- Compute F_H to get $\sum_{\rho' \in \widehat{H}} \sum_{i,j=0}^{d_{\rho'}-1} |\rho'(h), i, j\rangle$.

- Match up $\rho'(h)$ in blocks to obtain $\sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle$.

- Multiply by $\rho(g_k)$ to get what we want:

$$\sum_{\rho} \rho(g_k) \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle = \sum_{\rho} \sum_{i,j=0}^{d_\rho-1} |\rho(g), i, j\rangle.$$

General technique for efficiently computing F_G

Inductively, let H be a subgroup of G of index n .

- Choose a basis so that $\rho|_H(h)$ is block diagonal for all h .
- Choose a set of coset representatives g_1, \dots, g_n for which multiplication by $\rho(g_k)$ is fast in our chosen basis.
- Rewrite each element $g = g_k h$.

- Compute F_H to get $\sum_{\rho' \in \widehat{H}} \sum_{i,j=0}^{d_{\rho'}-1} |\rho'(h), i, j\rangle$.

- Match up $\rho'(h)$ in blocks to obtain $\sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle$.

- Multiply by $\rho(g_k)$ to get what we want:

$$\sum_{\rho} \rho(g_k) \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle = \sum_{\rho} \sum_{i,j=0}^{d_\rho-1} |\rho(g), i, j\rangle.$$

General technique for efficiently computing F_G

Inductively, let H be a subgroup of G of index n .

- Choose a basis so that $\rho|_H(h)$ is block diagonal for all h .
- Choose a set of coset representatives g_1, \dots, g_n for which multiplication by $\rho(g_k)$ is fast in our chosen basis.
- Rewrite each element $g = g_k h$.

- Compute F_H to get $\sum_{\rho' \in \widehat{H}} \sum_{i,j=0}^{d_{\rho'}-1} |\rho'(h), i, j\rangle$.

- Match up $\rho'(h)$ in blocks to obtain $\sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle$.

- Multiply by $\rho(g_k)$ to get what we want:

$$\sum_{\rho} \rho(g_k) \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle = \sum_{\rho} \sum_{i,j=0}^{d_\rho-1} |\rho(g), i, j\rangle.$$

General technique for efficiently computing F_G

Inductively, let H be a subgroup of G of index n .

- Choose a basis so that $\rho|_H(h)$ is block diagonal for all h .
- Choose a set of coset representatives g_1, \dots, g_n for which multiplication by $\rho(g_k)$ is fast in our chosen basis.
- Rewrite each element $g = g_k h$.

- Compute F_H to get $\sum_{\rho' \in \widehat{H}} \sum_{i,j=0}^{d_{\rho'}-1} |\rho'(h), i, j\rangle$.

- Match up $\rho'(h)$ in blocks to obtain $\sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle$.

- Multiply by $\rho(g_k)$ to get what we want:

$$\sum_{\rho} \rho(g_k) \sum_{i,j=0}^{d_\rho-1} |\rho(h), i, j\rangle = \sum_{\rho} \sum_{i,j=0}^{d_\rho-1} |\rho(g), i, j\rangle.$$

Algorithm: cyclic group

Theorem: efficient algorithm for $\mathbb{Z}/(2^n)$

There is an efficient quantum algorithm for computing the F_G when $G = \mathbb{Z}/(2^n)$.

- Inductively, use the subgroup $H = \mathbb{Z}/(2^{n-1})$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Letting $G = \langle g \rangle$, write $|g^{2j+b}\rangle = |j\rangle |b\rangle$.
- We want the map $|j\rangle |b\rangle \mapsto \sum_i \omega^{(2j+b)i} |i\rangle$.
- Computing F_H we get $\sum_i \omega^{2ij} |i\rangle |b\rangle$.
- When $b = 1$ we need to multiply $|i\rangle |b\rangle$ by ω^i . This can be done recursively using $O(n)$ controlled single qubit gates.
- Permute qubits as necessary.

Algorithm: cyclic group

Theorem: efficient algorithm for $\mathbb{Z}/(2^n)$

There is an efficient quantum algorithm for computing the F_G when $G = \mathbb{Z}/(2^n)$.

- Inductively, use the subgroup $H = \mathbb{Z}/(2^{n-1})$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Letting $G = \langle g \rangle$, write $|g^{2j+b}\rangle = |j\rangle |b\rangle$.
- We want the map $|j\rangle |b\rangle \mapsto \sum_i \omega^{(2j+b)i} |i\rangle$.
- Computing F_H we get $\sum_i \omega^{2ij} |i\rangle |b\rangle$.
- When $b = 1$ we need to multiply $|i\rangle |b\rangle$ by ω^i . This can be done recursively using $O(n)$ controlled single qubit gates.
- Permute qubits as necessary.

Algorithm: cyclic group

Theorem: efficient algorithm for $\mathbb{Z}/(2^n)$

There is an efficient quantum algorithm for computing the F_G when $G = \mathbb{Z}/(2^n)$.

- Inductively, use the subgroup $H = \mathbb{Z}/(2^{n-1})$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Letting $G = \langle g \rangle$, write $|g^{2j+b}\rangle = |j\rangle |b\rangle$.
- We want the map $|j\rangle |b\rangle \mapsto \sum_i \omega^{(2j+b)i} |i\rangle$.
- Computing F_H we get $\sum_i \omega^{2ij} |i\rangle |b\rangle$.
- When $b = 1$ we need to multiply $|i\rangle |b\rangle$ by ω^i . This can be done recursively using $O(n)$ controlled single qubit gates.
- Permute qubits as necessary.

Algorithm: cyclic group

Theorem: efficient algorithm for $\mathbb{Z}/(2^n)$

There is an efficient quantum algorithm for computing the F_G when $G = \mathbb{Z}/(2^n)$.

- Inductively, use the subgroup $H = \mathbb{Z}/(2^{n-1})$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Letting $G = \langle g \rangle$, write $|g^{2j+b}\rangle = |j\rangle |b\rangle$.
- We want the map $|j\rangle |b\rangle \mapsto \sum_i \omega^{(2j+b)i} |i\rangle$.
- Computing F_H we get $\sum_i \omega^{2ij} |i\rangle |b\rangle$.
- When $b = 1$ we need to multiply $|i\rangle |b\rangle$ by ω^i . This can be done recursively using $O(n)$ controlled single qubit gates.
- Permute qubits as necessary.

Algorithm: cyclic group

Theorem: efficient algorithm for $\mathbb{Z}/(2^n)$

There is an efficient quantum algorithm for computing the F_G when $G = \mathbb{Z}/(2^n)$.

- Inductively, use the subgroup $H = \mathbb{Z}/(2^{n-1})$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Letting $G = \langle g \rangle$, write $|g^{2j+b}\rangle = |j\rangle |b\rangle$.
- We want the map $|j\rangle |b\rangle \mapsto \sum_i \omega^{(2j+b)i} |i\rangle$.
- Computing F_H we get $\sum_i \omega^{2ij} |i\rangle |b\rangle$.
- When $b = 1$ we need to multiply $|i\rangle |b\rangle$ by ω^i . This can be done recursively using $O(n)$ controlled single qubit gates.
- Permute qubits as necessary.

Algorithm: dihedral group

Theorem [Hoyer '98]: efficient algorithm for D_{2^n}

There is an efficient quantum algorithm for computing F_G when $G = D_{2^n}$.

- Use the subgroup $H = \mathbb{Z}/(2^n)$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Write any element in D_{2^n} as $|j\rangle |b\rangle$, where $j \in \mathbb{Z}/(2^n)$.
- Computing F_H , we get $\sum_{i=0}^{2^n-1} |\omega^{ij}, 0, b\rangle$.
- We want to rearrange this to get roughly

$$\rho(b) \sum_{i=0}^{2^n-1} \left(|\omega^{ij}, 0, 0\rangle + |0, 0, 1\rangle + |0, 1, 0\rangle + |\omega^{-ij}, 1, 1\rangle \right).$$

- Rearrange qubits to get ω^{ij} and ω^{-ij} together. Multiplication of $\rho(b)$ can be done by conditionally replacing ω with ω^{-1} and then flipping the last qubit.

Algorithm: dihedral group

Theorem [Hoyer '98]: efficient algorithm for D_{2^n}

There is an efficient quantum algorithm for computing F_G when $G = D_{2^n}$.

- Use the subgroup $H = \mathbb{Z}/(2^n)$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Write any element in D_{2^n} as $|j\rangle |b\rangle$, where $j \in \mathbb{Z}/(2^n)$.
- Computing F_H , we get $\sum_{i=0}^{2^n-1} |\omega^{ij}, 0, b\rangle$.
- We want to rearrange this to get roughly

$$\rho(b) \sum_{i=0}^{2^n-1} \left(|\omega^{ij}, 0, 0\rangle + |0, 0, 1\rangle + |0, 1, 0\rangle + |\omega^{-ij}, 1, 1\rangle \right).$$

- Rearrange qubits to get ω^{ij} and ω^{-ij} together. Multiplication of $\rho(b)$ can be done by conditionally replacing ω with ω^{-1} and then flipping the last qubit.

Algorithm: dihedral group

Theorem [Hoyer '98]: efficient algorithm for D_{2^n}

There is an efficient quantum algorithm for computing F_G when $G = D_{2^n}$.

- Use the subgroup $H = \mathbb{Z}/(2^n)$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Write any element in D_{2^n} as $|j\rangle |b\rangle$, where $j \in \mathbb{Z}/(2^n)$.
- Computing F_H , we get $\sum_{i=0}^{2^n-1} |\omega^{ij}, 0, b\rangle$.
- We want to rearrange this to get roughly

$$\rho(b) \sum_{i=0}^{2^n-1} \left(|\omega^{ij}, 0, 0\rangle + |0, 0, 1\rangle + |0, 1, 0\rangle + |\omega^{-ij}, 1, 1\rangle \right).$$

- Rearrange qubits to get ω^{ij} and ω^{-ij} together. Multiplication of $\rho(b)$ can be done by conditionally replacing ω with ω^{-1} and then flipping the last qubit.

Algorithm: dihedral group

Theorem [Hoyer '98]: efficient algorithm for D_{2^n}

There is an efficient quantum algorithm for computing F_G when $G = D_{2^n}$.

- Use the subgroup $H = \mathbb{Z}/(2^n)$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Write any element in D_{2^n} as $|j\rangle |b\rangle$, where $j \in \mathbb{Z}/(2^n)$.
- Computing F_H , we get $\sum_{i=0}^{2^n-1} |\omega^{ij}, 0, b\rangle$.
- We want to rearrange this to get roughly

$$\rho(b) \sum_{i=0}^{2^n-1} \left(|\omega^{ij}, 0, 0\rangle + |0, 0, 1\rangle + |0, 1, 0\rangle + |\omega^{-ij}, 1, 1\rangle \right).$$

- Rearrange qubits to get ω^{ij} and ω^{-ij} together. Multiplication of $\rho(b)$ can be done by conditionally replacing ω with ω^{-1} and then flipping the last qubit.

Algorithm: dihedral group

Theorem [Hoyer '98]: efficient algorithm for D_{2^n}

There is an efficient quantum algorithm for computing F_G when $G = D_{2^n}$.

- Use the subgroup $H = \mathbb{Z}/(2^n)$ of index 2. The coset reps are $b = 0$ and $b = 1$.
- Write any element in D_{2^n} as $|j\rangle |b\rangle$, where $j \in \mathbb{Z}/(2^n)$.
- Computing F_H , we get $\sum_{i=0}^{2^n-1} |\omega^{ij}, 0, b\rangle$.
- We want to rearrange this to get roughly

$$\rho(b) \sum_{i=0}^{2^n-1} \left(|\omega^{ij}, 0, 0\rangle + |0, 0, 1\rangle + |0, 1, 0\rangle + |\omega^{-ij}, 1, 1\rangle \right).$$

- Rearrange qubits to get ω^{ij} and ω^{-ij} together. Multiplication of $\rho(b)$ can be done by conditionally replacing ω with ω^{-1} and then flipping the last qubit.

Algorithm: symmetric group

Theorem [Beals '97]: efficient algorithm for S_n

There is an efficient quantum algorithm for computing F_G when $G = S_n$.

- Inductively, use the subgroup $H = S_{n-1}$. The coset representatives are $\tau_k = (k, n)$. Write each element in S_n as $|h, k\rangle$.
- Choose Young's orthonormal basis. In this basis, the irreps of G restricted to H are block diagonal.
- Computing F_H , we get $\sum_{\mu} \sum_{i,j=0}^{d_{\mu}-1} |\rho_{\mu}(h), i, j\rangle$.
- We want to get $\sum_{\lambda} \rho(\tau_k) \sum_{i,j=0}^{d_{\lambda}-1} |\rho_{\lambda}(h), i, j\rangle$.
- The branching algorithm tells us how to rearrange the ρ_{μ} to obtain ρ_{λ} , and multiplication by $\rho(\tau_k)$ is easy in this basis.

Algorithm: symmetric group

Theorem [Beals '97]: efficient algorithm for S_n

There is an efficient quantum algorithm for computing F_G when $G = S_n$.

- Inductively, use the subgroup $H = S_{n-1}$. The coset representatives are $\tau_k = (k, n)$. Write each element in S_n as $|h, k\rangle$.
- Choose Young's orthonormal basis. In this basis, the irreps of G restricted to H are block diagonal.
- Computing F_H , we get $\sum_{\mu} \sum_{i,j=0}^{d_{\mu}-1} |\rho_{\mu}(h), i, j\rangle$.
- We want to get $\sum_{\lambda} \rho(\tau_k) \sum_{i,j=0}^{d_{\lambda}-1} |\rho_{\lambda}(h), i, j\rangle$.
- The branching algorithm tells us how to rearrange the ρ_{μ} to obtain ρ_{λ} , and multiplication by $\rho(\tau_k)$ is easy in this basis.

Algorithm: symmetric group

Theorem [Beals '97]: efficient algorithm for S_n

There is an efficient quantum algorithm for computing F_G when $G = S_n$.

- Inductively, use the subgroup $H = S_{n-1}$. The coset representatives are $\tau_k = (k, n)$. Write each element in S_n as $|h, k\rangle$.
- Choose Young's orthonormal basis. In this basis, the irreps of G restricted to H are block diagonal.
- Computing F_H , we get $\sum_{\mu} \sum_{i,j=0}^{d_{\mu}-1} |\rho_{\mu}(h), i, j\rangle$.
- We want to get $\sum_{\lambda} \rho(\tau_k) \sum_{i,j=0}^{d_{\lambda}-1} |\rho_{\lambda}(h), i, j\rangle$.
- The branching algorithm tells us how to rearrange the ρ_{μ} to obtain ρ_{λ} , and multiplication by $\rho(\tau_k)$ is easy in this basis.

Algorithm: symmetric group

Theorem [Beals '97]: efficient algorithm for S_n

There is an efficient quantum algorithm for computing F_G when $G = S_n$.

- Inductively, use the subgroup $H = S_{n-1}$. The coset representatives are $\tau_k = (k, n)$. Write each element in S_n as $|h, k\rangle$.
- Choose Young's orthonormal basis. In this basis, the irreps of G restricted to H are block diagonal.
- Computing F_H , we get $\sum_{\mu} \sum_{i,j=0}^{d_{\mu}-1} |\rho_{\mu}(h), i, j\rangle$.
- We want to get $\sum_{\lambda} \rho(\tau_k) \sum_{i,j=0}^{d_{\lambda}-1} |\rho_{\lambda}(h), i, j\rangle$.
- The branching algorithm tells us how to rearrange the ρ_{μ} to obtain ρ_{λ} , and multiplication by $\rho(\tau_k)$ is easy in this basis.

Algorithm: symmetric group

Theorem [Beals '97]: efficient algorithm for S_n

There is an efficient quantum algorithm for computing F_G when $G = S_n$.

- Inductively, use the subgroup $H = S_{n-1}$. The coset representatives are $\tau_k = (k, n)$. Write each element in S_n as $|h, k\rangle$.
- Choose Young's orthonormal basis. In this basis, the irreps of G restricted to H are block diagonal.
- Computing F_H , we get $\sum_{\mu} \sum_{i,j=0}^{d_{\mu}-1} |\rho_{\mu}(h), i, j\rangle$.
- We want to get $\sum_{\lambda} \rho(\tau_k) \sum_{i,j=0}^{d_{\lambda}-1} |\rho_{\lambda}(h), i, j\rangle$.
- The branching algorithm tells us how to rearrange the ρ_{μ} to obtain ρ_{λ} , and multiplication by $\rho(\tau_k)$ is easy in this basis.

Algorithm: symmetric group

Theorem [Beals '97]: efficient algorithm for S_n

There is an efficient quantum algorithm for computing F_G when $G = S_n$.

- Inductively, use the subgroup $H = S_{n-1}$. The coset representatives are $\tau_k = (k, n)$. Write each element in S_n as $|h, k\rangle$.
- Choose Young's orthonormal basis. In this basis, the irreps of G restricted to H are block diagonal.
- Computing F_H , we get $\sum_{\mu} \sum_{i,j=0}^{d_{\mu}-1} |\rho_{\mu}(h), i, j\rangle$.
- We want to get $\sum_{\lambda} \rho(\tau_k) \sum_{i,j=0}^{d_{\lambda}-1} |\rho_{\lambda}(h), i, j\rangle$.
- The branching algorithm tells us how to rearrange the ρ_{μ} to obtain ρ_{λ} , and multiplication by $\rho(\tau_k)$ is easy in this basis.

Oracles

Let $\mathcal{F} = \{f_n: \{0, 1\}^n \rightarrow \{0, 1\}^m\}$ be a family of functions.

Unitary form of f_n

The **unitary form** of f_n , $V_{f_n} \in \mathcal{U}(2^{n+m}, \mathbb{C})$, is defined on the basis by

$$V_{f_n} |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle.$$

Recall that a quantum algorithm is a family of unitary matrices $\{U_{2^n}\}$.

Oracular algorithm

An **oracular algorithm** is one in which U_{2^n} factors as elementary gates and copies of V_{f_n} . An oracular algorithm is efficient if the number of elementary gates and V_{f_n} 's in its factorization is $\text{poly}(n)$.

Oracles

Let $\mathcal{F} = \{f_n: \{0, 1\}^n \rightarrow \{0, 1\}^m\}$ be a family of functions.

Unitary form of f_n

The **unitary form** of f_n , $V_{f_n} \in \mathcal{U}(2^{n+m}, \mathbb{C})$, is defined on the basis by

$$V_{f_n} |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle.$$

Recall that a quantum algorithm is a family of unitary matrices $\{U_{2^n}\}$.

Oracular algorithm

An **oracular algorithm** is one in which U_{2^n} factors as elementary gates and copies of V_{f_n} . An oracular algorithm is efficient if the number of elementary gates and V_{f_n} 's in its factorization is $\text{poly}(n)$.

Hidden subgroup problem

Let G be a finite group.

Definition: coset-separating function

A function $f: G \rightarrow \mathbb{C}$ is **coset-separating** if there exists a subgroup H of G so that $f(g) = f(g')$ if and only if $gH = g'H$.

Example: coset-separating function

The function $f: \mathbb{Z}/(2n) \rightarrow \{0, 1\}$ which sends evens to 0 and odds to 1 is coset-separating with respect to the subgroup $H = \mathbb{Z}/(n)$.

Definition: hidden subgroup problem (HSP)

Given a coset-separating function f on G , modeled as an oracle, determine a generating set for the subgroup H .

Hidden subgroup problem

Let G be a finite group.

Definition: coset-separating function

A function $f: G \rightarrow \mathbb{C}$ is **coset-separating** if there exists a subgroup H of G so that $f(g) = f(g')$ if and only if $gH = g'H$.

Example: coset-separating function

The function $f: \mathbb{Z}/(2n) \rightarrow \{0, 1\}$ which sends evens to 0 and odds to 1 is coset-separating with respect to the subgroup $H = \mathbb{Z}/(n)$.

Definition: hidden subgroup problem (HSP)

Given a coset-separating function f on G , modeled as an oracle, determine a generating set for the subgroup H .

Fourier sampling

Fourier sampling algorithm

- 1 Prepare the state $\sum_g |g\rangle |0\rangle$.
- 2 Query the oracle, yielding $\sum_g |g\rangle |f(g)\rangle$.
- 3 Measure the second register, observing a value $f(g)$ uniformly at random, and leaving the coset $\sum_{h \in H} |gh\rangle$.
- 4 Apply F_G , to get

$$\sum_{h \in H} \sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(gh), i, j\rangle.$$

At this point, we can measure our qubits. Or, we can follow with another quantum algorithm.

Fourier sampling

Fourier sampling algorithm

- 1 Prepare the state $\sum_g |g\rangle |0\rangle$.
- 2 Query the oracle, yielding $\sum_g |g\rangle |f(g)\rangle$.
- 3 Measure the second register, observing a value $f(g)$ uniformly at random, and leaving the coset $\sum_{h \in H} |gh\rangle$.
- 4 Apply F_G , to get

$$\sum_{h \in H} \sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(gh), i, j\rangle.$$

At this point, we can measure our qubits. Or, we can follow with another quantum algorithm.

Fourier sampling

Fourier sampling algorithm

- 1 Prepare the state $\sum_g |g\rangle |0\rangle$.
- 2 Query the oracle, yielding $\sum_g |g\rangle |f(g)\rangle$.
- 3 Measure the second register, observing a value $f(g)$ uniformly at random, and leaving the coset $\sum_{h \in H} |gh\rangle$.
- 4 Apply F_G , to get

$$\sum_{h \in H} \sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(gh), i, j\rangle.$$

At this point, we can measure our qubits. Or, we can follow with another quantum algorithm.

Fourier sampling

Fourier sampling algorithm

- 1 Prepare the state $\sum_g |g\rangle |0\rangle$.
- 2 Query the oracle, yielding $\sum_g |g\rangle |f(g)\rangle$.
- 3 Measure the second register, observing a value $f(g)$ uniformly at random, and leaving the coset $\sum_{h \in H} |gh\rangle$.
- 4 Apply F_G , to get

$$\sum_{h \in H} \sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(gh), i, j\rangle.$$

At this point, we can measure our qubits. Or, we can follow with another quantum algorithm.

Fourier sampling

Fourier sampling algorithm

- 1 Prepare the state $\sum_g |g\rangle |0\rangle$.
- 2 Query the oracle, yielding $\sum_g |g\rangle |f(g)\rangle$.
- 3 Measure the second register, observing a value $f(g)$ uniformly at random, and leaving the coset $\sum_{h \in H} |gh\rangle$.
- 4 Apply F_G , to get

$$\sum_{h \in H} \sum_{\rho \in \widehat{G}} \sum_{i,j=0}^{d_\rho-1} |\rho(gh), i, j\rangle.$$

At this point, we can measure our qubits. Or, we can follow with another quantum algorithm.

Cyclic HSP

Theorem

Fourier sampling efficiently solves the hidden subgroup problem for $\mathbb{Z}/(2^n)$.

The decision-problem version of the hidden subgroup problem for a finite group is whether the hidden subgroup is non-trivial.

Corollary: oracle separation

Using a coset-separating family of functions \mathcal{F} as an oracle gives an oracle separation of BQP from BPP. In other words,

$$\text{BPP}^{\mathcal{F}} \neq \text{BQP}^{\mathcal{F}}.$$

Cyclic HSP

Theorem

Fourier sampling efficiently solves the hidden subgroup problem for $\mathbb{Z}/(2^n)$.

The decision-problem version of the hidden subgroup problem for a finite group is whether the hidden subgroup is non-trivial.

Corollary: oracle separation

Using a coset-separating family of functions \mathcal{F} as an oracle gives an oracle separation of BQP from BPP. In other words,

$$\text{BPP}^{\mathcal{F}} \neq \text{BQP}^{\mathcal{F}}.$$

Dihedral HSP

Theorem [Kuperberg '04]

Fourier sampling followed by a sieve algorithm yields a sub-exponential quantum algorithm solving the hidden subgroup problem for D_{2^n} .

- Finding a hidden subgroup of D_{2^n} reduces to finding those of the form $H = \langle yx^s \rangle$. Finding the parity of s is enough for recursion.
- Make a list of $O(2^{\sqrt{n}})$ states using the Fourier sampling algorithm. Each state is of the form $|\psi_k\rangle \propto |0\rangle + e^{2\pi iks/2^n} |1\rangle$.
- Use a sieve of depth $O(\sqrt{n})$ to create the state $|\psi_{2^{n-1}}\rangle \propto |0\rangle + (-1)^s |1\rangle$ with high probability.
- Measure the parity of s .

Dihedral HSP

Theorem [Kuperberg '04]

Fourier sampling followed by a sieve algorithm yields a sub-exponential quantum algorithm solving the hidden subgroup problem for D_{2^n} .

- Finding a hidden subgroup of D_{2^n} reduces to finding those of the form $H = \langle yx^s \rangle$. Finding the parity of s is enough for recursion.
- Make a list of $O(2^{\sqrt{n}})$ states using the Fourier sampling algorithm. Each state is of the form $|\psi_k\rangle \propto |0\rangle + e^{2\pi iks/2^n} |1\rangle$.
- Use a sieve of depth $O(\sqrt{n})$ to create the state $|\psi_{2^{n-1}}\rangle \propto |0\rangle + (-1)^s |1\rangle$ with high probability.
- Measure the parity of s .

Dihedral HSP

Theorem [Kuperberg '04]

Fourier sampling followed by a sieve algorithm yields a sub-exponential quantum algorithm solving the hidden subgroup problem for D_{2^n} .

- Finding a hidden subgroup of D_{2^n} reduces to finding those of the form $H = \langle yx^s \rangle$. Finding the parity of s is enough for recursion.
- Make a list of $O(2^{\sqrt{n}})$ states using the Fourier sampling algorithm. Each state is of the form $|\psi_k\rangle \propto |0\rangle + e^{2\pi iks/2^n} |1\rangle$.
- Use a sieve of depth $O(\sqrt{n})$ to create the state $|\psi_{2^{n-1}}\rangle \propto |0\rangle + (-1)^s |1\rangle$ with high probability.
- Measure the parity of s .

Dihedral HSP

Theorem [Kuperberg '04]

Fourier sampling followed by a sieve algorithm yields a sub-exponential quantum algorithm solving the hidden subgroup problem for D_{2^n} .

- Finding a hidden subgroup of D_{2^n} reduces to finding those of the form $H = \langle yx^s \rangle$. Finding the parity of s is enough for recursion.
- Make a list of $O(2^{\sqrt{n}})$ states using the Fourier sampling algorithm. Each state is of the form $|\psi_k\rangle \propto |0\rangle + e^{2\pi iks/2^n} |1\rangle$.
- Use a sieve of depth $O(\sqrt{n})$ to create the state $|\psi_{2^{n-1}}\rangle \propto |0\rangle + (-1)^s |1\rangle$ with high probability.
- Measure the parity of s .

Symmetric HSP

Theorem [Moore, Russell, Schulman '05]

Fourier sampling alone cannot be used to find an efficient quantum algorithm solving the hidden subgroup problem for S_n .

However, this does not preclude the possibility of algorithms (such as Kuperberg's sieve) which can follow Fourier sampling to solve the problem.

Theorem [Ettinger, Hoyer, Knill '04]

For any group G , a polynomial number of samples from the Fourier sampling algorithm is sufficient to determine generators for H information-theoretically.

Unfortunately, it is unknown how to determine the generators for H from this information in polynomial time for S_n .

Symmetric HSP

Theorem [Moore, Russell, Schulman '05]

Fourier sampling alone cannot be used to find an efficient quantum algorithm solving the hidden subgroup problem for S_n .

However, this does not preclude the possibility of algorithms (such as Kuperberg's sieve) which can follow Fourier sampling to solve the problem.

Theorem [Ettinger, Hoyer, Knill '04]

For any group G , a polynomial number of samples from the Fourier sampling algorithm is sufficient to determine generators for H information-theoretically.

Unfortunately, it is unknown how to determine the generators for H from this information in polynomial time for S_n .