Quantum Fourier transform beyond Shor’s algorithm

András Gilyén

Alfréd Rényi Institute of Mathematics
Budapest, Hungary

Quantum Computing Graduate Summer School, Park City Mathematics Institute, 2023 July 19

Day 3 – Bernstein-Vazirani algorithm & Quantum Gradient Computation

The Bernstein-Vazirani algorithm (1992)

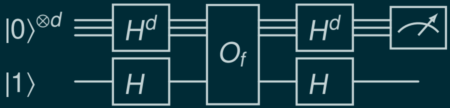
Problem

- Given a Boolean function $f\colon \{0,1\}^d \to \{0,1\}$ so that $f(x) = s \cdot x \pmod 2$; find s.
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$.
- Can be converted to phase oracle $\widetilde{O}_f\colon |x\rangle \mapsto (-1)^{f(x)}|x\rangle$.

The Bernstein-Vazirani algorithm (1992)

Problem

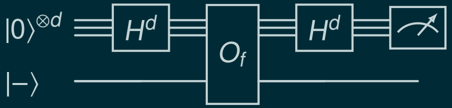
- Given a Boolean function $f\colon \{0,1\}^d \to \{0,1\}$ so that $f(x) = s \cdot x \pmod 2$; find s.
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$.
- Can be converted to phase oracle $\widetilde{O}_f\colon |x\rangle \mapsto (-1)^{f(x)}|x\rangle$.



The Bernstein-Vazirani algorithm (1992)

Problem

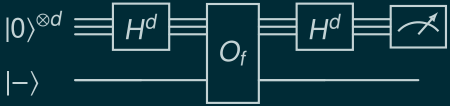
- Given a Boolean function $f\colon \{0,1\}^d \to \{0,1\}$ so that $f(x) = s \cdot x \pmod 2$; find s.
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$.
- Can be converted to phase oracle $\widetilde{O}_f\colon |x\rangle \mapsto (-1)^{f(x)}|x\rangle$.



The Bernstein-Vazirani algorithm (1992)

Problem

- Given a Boolean function $f\colon \{0,1\}^d \to \{0,1\}$ so that $f(x) = s \cdot x \pmod 2$; find s.
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$.
- Can be converted to phase oracle $\widetilde{O}_f\colon |x\rangle \mapsto (-1)^{f(x)}|x\rangle$.



Take away message

- Shows the power of Fourier transform (over the group $\mathbb{Z}_2^d$)
- (+1 Phase kickback is a surprising and useful quantum effect)

Jordan's quantum algorithm for gradients (2004)

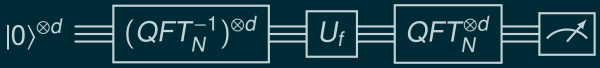
A generalization of the Bernstein-Vazirani algorithm

- Given a function $f\colon \mathbb{Z}_N^d \to \mathbb{Z}_N$ so that $f(x) = s \cdot x \pmod N$; find $s \in \mathbb{Z}_N^d$.
- The function is given as a phase oracle $U_f : |x\rangle \mapsto e^{\frac{2\pi i}{N} f(x)}|x\rangle = e^{2\pi i \frac{sx}{N}}|x\rangle$.

Jordan's quantum algorithm for gradients (2004)

A generalization of the Bernstein-Vazirani algorithm

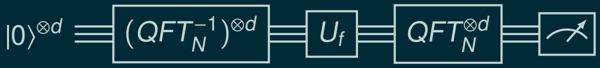
- Given a function $f : \mathbb{Z}_N^d \to \mathbb{Z}_N$ so that $f(x) = s \cdot x \pmod{N}$; find $s \in \mathbb{Z}_N^d$.
- The function is given as a phase oracle $U_f : |x\rangle \mapsto e^{\frac{2\pi i}{N} f(x)} |x\rangle = e^{2\pi i \frac{sx}{N}} |x\rangle$.



Jordan's quantum algorithm for gradients (2004)

A generalization of the Bernstein-Vazirani algorithm

- Given a function $f\colon \mathbb{Z}_N^d \to \mathbb{Z}_N$ so that $f(x) = s \cdot x \pmod{N}$; find $s \in \mathbb{Z}_N^d$.
- The function is given as a phase oracle $U_f\colon |x\rangle \mapsto e^{\frac{2\pi i}{N} f(x)}|x\rangle = e^{2\pi i \frac{sx}{N}}|x\rangle$.



$$\text{Recall: } QFT_N\colon |j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{\ell=0}^{N-1} e^{-2\pi i \frac{j\ell}{N}} |\ell\rangle$$

Quantum Gradient Computation Algorithm

Quantum Gradient Computation Algorithm

Quantum Fourier Transform - extracting linear phase factors

Let $\varepsilon = \frac{1}{N}$ be the precision we want to achieve, and set

$$G = \left\{ \frac{0}{N}, \frac{1}{N}, \ldots, \frac{N-1}{N} \right\}.$$

Quantum Gradient Computation Algorithm

Quantum Fourier Transform - extracting linear phase factors

Let $\varepsilon = \frac{1}{N}$ be the precision we want to achieve, and set

$$G = \left\{ \frac{0}{N}, \frac{1}{N}, \ldots, \frac{N-1}{N} \right\}.$$

Suppose $x, k \in G$ are quantum (basis) states, then

$$\sum_{x \in G} |x\rangle \frac{e^{2\pi i (Nxk)}}{\sqrt{N}} \xrightarrow{QFT_N} |k\rangle.$$

Quantum Gradient Computation Algorithm

Gradient computation - S. Jordan's algorithm (2004)

Input: phase oracle $O_f : \left|\vec{x}\right\rangle \rightarrow \left|\vec{x}\right\rangle e^{2\pi i f(\vec{x})}$, where $\vec{x} \in G^d$.
Output: gradient with (hopefully) $\varepsilon = 1/N$ coordinate-wise precision

Quantum Gradient Computation Algorithm

Gradient computation - S. Jordan's algorithm (2004)

Input: phase oracle $O_f : \left|\vec{x}\right\rangle \to \left|\vec{x}\right\rangle e^{2\pi i f(\vec{x})}$, where $\vec{x} \in G^d$.
Output: gradient with (hopefully) $\varepsilon = 1/N$ coordinate-wise precision

Assumption: $f(\vec{x}) \approx f(\vec{0}) + \vec{x}\nabla f(\vec{0})$, then

Quantum Gradient Computation Algorithm

Gradient computation - S. Jordan's algorithm (2004)

Input: phase oracle $O_f : |\vec{x}\rangle \to |\vec{x}\rangle e^{2\pi i f(\vec{x})}$, where $\vec{x} \in G^d$.
Output: gradient with (hopefully) $\varepsilon = 1/N$ coordinate-wise precision

Assumption: $f(\vec{x}) \approx f(\vec{0}) + \vec{x}\nabla f(\vec{0})$, then

$$\sum_{\vec{x}} \frac{|\vec{x}\rangle}{N^{\frac{d}{2}}} \xrightarrow[N\times]{O_f} \sum_{\vec{x}} |\vec{x}\rangle \frac{e^{2\pi i N f(\vec{x})}}{N^{\frac{d}{2}}} \approx e^{2\pi i N f(\vec{0})} \sum_{\vec{x}} |\vec{x}\rangle \frac{e^{2\pi i \left(N\vec{x}\nabla f(\vec{0})\right)}}{N^{\frac{d}{2}}} \xrightarrow[\otimes d]{QFT_N} \approx \left|\nabla f(\vec{0})\right\rangle.$$

Quantum Gradient Computation Algorithm

Gradient computation - S. Jordan's algorithm (2004)

Input: phase oracle $O_f : |\vec{x}\rangle \to |\vec{x}\rangle e^{2\pi i f(\vec{x})}$, where $\vec{x} \in G^d$.
Output: gradient with (hopefully) $\varepsilon = 1/N$ coordinate-wise precision

Assumption: $f(\vec{x}) \approx f(\vec{0}) + \vec{x}\nabla f(\vec{0})$, then

$$\sum_{\vec{x}} \frac{|\vec{x}\rangle}{N^{\frac{d}{2}}} \xrightarrow[N\times]{O_f} \sum_{\vec{x}} |\vec{x}\rangle \frac{e^{2\pi i N f(\vec{x})}}{N^{\frac{d}{2}}} \approx e^{2\pi i N f(\vec{0})} \sum_{\vec{x}} |\vec{x}\rangle \frac{e^{2\pi i \left(N\vec{x}\nabla f(\vec{0})\right)}}{N^{\frac{d}{2}}} \xrightarrow[\otimes d]{QFT_N} \approx \left|\nabla f(\vec{0})\right\rangle.$$

Exponential speed-up?

- If we have a circuit computing f it introduces small overheads.
- "Cheap gradient principle": $\leq 4\times$ overhead for gradient computation