量子算法：QFT与Shor算法

Quantum Fourier Transformation and Shor's Algorithm

魏文杰

癸卯年十月廿一2023年12月3日13:19:02

In[1]:=
```
<< Wolfram`QuantumFramework`
```

目录

量子算法预备

量子算法的应用清单[1]：应用领域[2]

- 凝聚态、核物理、高能物理、量子化学、组合(又称，离散)优化、连续优化、密码学与密码分析、微分方程、金融、经典数据的机器学习

重点介绍的两种量子算法

- 基于量子傅里叶变换QFT的算法
  - 相位估计 phase estimation
  - 1994 Shor算法，多项式时间的因数分解算法。因数分解 ∈ BQP
  - 离散对数问题 discrete log
  - 隐藏子群问题 hidden subgroup
  - QFT
    - quantum field theory
    - quantum fluctuation theorem
    - quantum Fourier transformation
- 量子搜索算法
  - 1996 Grover 算法对非结构化搜索问题进行了量子二次加速
  - 推广：振幅放大 amplitude amplification
    - 量子计算机要想对精确组合优化产生影响，必须做到以下两点之一：1）量子硬件的预期底层时钟速度和容错量子计算的开销取得巨大进步；或2）量子算法的开发大大超过Grover算法提供的四倍速度[1]。

其他量子算法

- 参见：量子算法动物园 以及综述 [1]
- 连续优化问题的相关算法：纳什均衡与线性规划问题
- 变分量子求解器 Variational Quantum Eigensolver
- 量子随机游走
- 1985 黑箱(oracle)算法：Deutsch 算法
  - 更多黑箱算法：Deutsch-Jozsa、Bernstein-Vazirani和Simon算法
- 量子通信协议：1984 Charles Bennett 和 Gilles Brassard 将量子理论应用于密码协议，并证明量子密钥分发(QKD)可以增强信息安全性
  - 2023新纪录：千公里以上的QKD
- 量子模拟算法
  - 1980s Richard Feynman：在量子计算机上模拟量子系统
  - 1996 Seth Lloyd：以小时间步长演化可以有效模拟任何包含少粒子相互作用的多体量子哈密顿量； 模拟所需的总时间呈多项式增长。另一个链接
- 新世代的攻防战：后(抗)量子加密 post-quantum cryptography

量子计算的威力：
BQP是否严格大于BPP？但愿存在某些问题是量子计算机的概率算法能高效解决，但是概率图灵机不能高效解决的
大规模的容错量子计算能否实现?

Deutsch算法

量子并行：Deutsch 算法

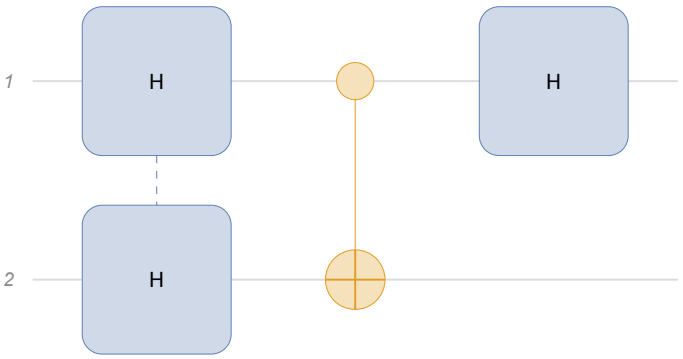
黑箱问题：给定黑箱函数 $f:\{0,1\}\to\{0,1\}$，判断 f 是平衡型还是常数型

量子并行：量子计算机能同时（在同一个幺正变换中）计算函数$f(x)$在许多x处的值

平衡型函数 f (x) = x 或者 f (x) = x ⊕ 1

In[2]:=
```
qc1 = QuantumCircuitOperator[{"H" → {1, 2}, "CNOT", "H"}];(*设f(x) = x*)
qc1["Diagram"]
QuantumCircuitMultiwayGraph[qc1];
```

Out[3]=



In[5]:=
```
steps = ComposeList[qc1["Operators"], QuantumState[{0, 1, 0, 0}]];
```

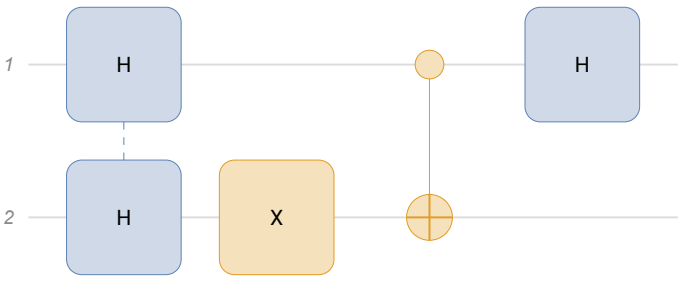
In[6]:=
```
Grid[Transpose[{Style[#, Bold] & /@ {"初态", "经过前两个H门", "经过Uf", "经过最后的H门"},
    #["Formula"] & /@ steps[[ ;; ]]}], Frame → All, Alignment → Left]
```

Out[6]=

初态	$\|01\rangle$
经过前两个H门	$\frac{1}{2}\|00\rangle-\frac{1}{2}\|01\rangle+\frac{1}{2}\|10\rangle-\frac{1}{2}\|11\rangle$
经过U_f	$\frac{1}{2}\|00\rangle-\frac{1}{2}\|01\rangle-\frac{1}{2}\|10\rangle+\frac{1}{2}\|11\rangle$
经过最后的H门	$\frac{1}{\sqrt{2}}\|10\rangle-\frac{1}{\sqrt{2}}\|11\rangle$

In[7]:=
```
qc2 = QuantumCircuitOperator[{"H" → {1, 2}, "X" → 2, "CNOT", "H"}];(*f(x)=x⊕1*)
qc2["Diagram"]
qc2[QuantumState[{0, 1, 0, 0}]]["Formula"]
```

Out[8]=

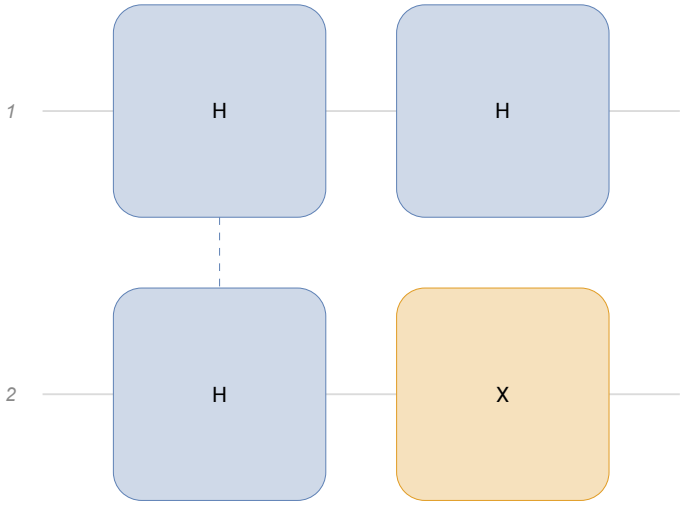


Out[9]=
$$-\frac{1}{\sqrt{2}}\left|10\right\rangle+\frac{1}{\sqrt{2}}\left|11\right\rangle$$

常数型函数 f(x)=1或 f(x)=0

In[10]:=
```
qc3 = QuantumCircuitOperator[{"H" → {1, 2}, "X" → 2, "H"}];(*f(x)=1*)
qc3["Diagram"]
qc3[QuantumState[{0, 1, 0, 0}]]["Formula"]
```

Out[11]=

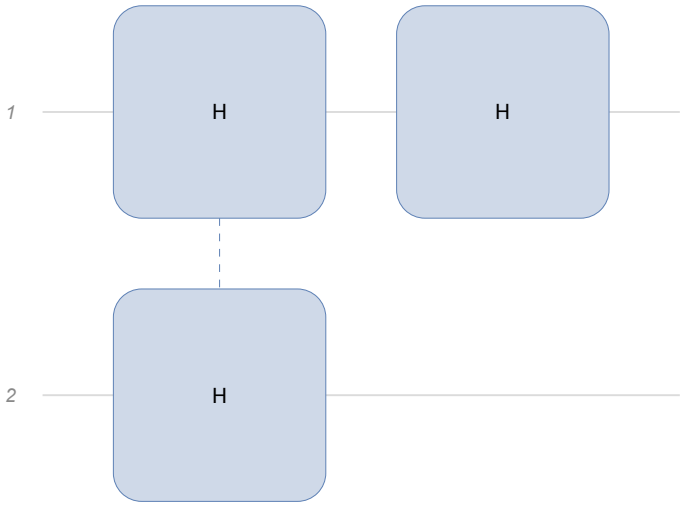


Out[12]=

$$-\frac{1}{\sqrt{2}}\left|00\right\rangle+\frac{1}{\sqrt{2}}\left|01\right\rangle$$

In[13]:=
```
qc4 = QuantumCircuitOperator[{"H" → {1, 2}, "H"}];(*f(x)=1*)
qc4["Diagram"]
qc4[QuantumState[{0, 1, 0, 0}]]["Formula"]
```

Out[14]=



Out[15]=

$$\frac{1}{\sqrt{2}}\left|00\right\rangle-\frac{1}{\sqrt{2}}\left|01\right\rangle$$

- 根据第一个qubit的测量结果，我们就能知道函数是平衡型还是常数型，也即计算了f(0)⊕f(1)的值，对于常数型f(0)⊕f(1)=0，对于平衡型f(0)⊕f(1)=1

- 每个量子线路都只使用了一次U_f，
  即只需要一次计算就完成了经典计算需要两次计算才能完成的任务
- 启发：对于那些经典计算获得的信息量超过问题的答案能够提供的信息量的问题（尝试计算信息量），也即经典计算存在计算冗余的问题，量子计算具有潜在的加速可能
- 判断平衡型和常数型的量子算法推广到 f 具有 n 个slot的情况，称为Deutsch-Jozsa算法，参见 Sec.1.4.4

Deutsch-Jozsa算法

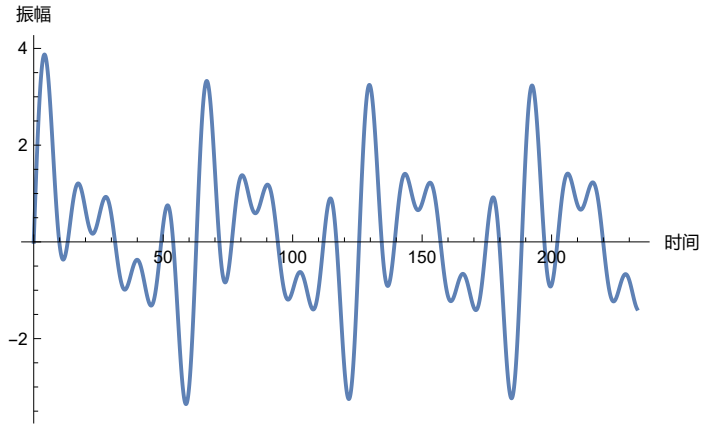
量子傅里叶变换

经典的离散傅里叶变换

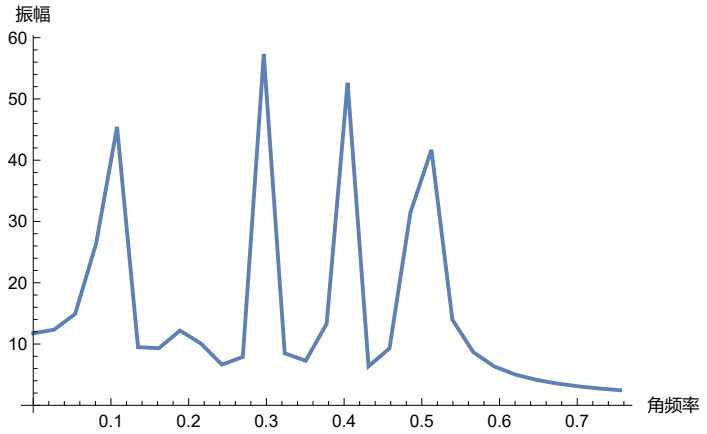
- $y_s = \frac{1}{\sqrt{N}} \sum_{r=1}^{N} u_r \, e^{2\pi i (r-1)(s-1)/N}$, s =1,..,N

```
Clear[f];
num = 13 400;(*总采样点数*)
timespan = 233;(*采样持续时间*)
bandwidth = num / (2 timespan);(*采样率、带宽(实际上是不同的概念)*)
f[t_] := (Sin[0.1 t] + E^(-0.03 t) Sin[0.2 t] + Sin[0.3 t] + Sin[0.4 t] + Sin[0.5 t]);
mat2 = Table[f[timespan * i / num], {i, 0, num}];
ListLinePlot[Table[{timespan * i / num, f[timespan * i / num]}, {i, 0, num}],
  AxesLabel → {"时间", "振幅"}]
four = Abs@Fourier@mat2;
da = Table[{k * 2 * Pi / timespan, four[[k + 1]]}, {k, 0, bandwidth}];
ListLinePlot[da, PlotRange → All, AxesLabel → {"角频率", "振幅"}]
```

Out[45]=



Out[48]=



量子Fourier变换

- $y_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i \, j \, k / N} u_k$, j =0, … , N-1
  - 注意Fourier变换是线性变换，因此才有显然的量子类比
- $\left| j \right\rangle \to \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i \, j \, k / 2^n} \left| k \right\rangle \; j = 0, \ldots, 2^n - 1$
  - 计算基的态 $| j \rangle = | j_1 \, j_2 \ldots j_N \rangle$ 取代了数列的项

- 约定：低位比特在线路图中靠上，在bracket中靠左 $|j_1\, j_2 \ldots\rangle$，即，越靠上或越靠左的比特的索引序号越小。(这种约定导致做加法时向低位进位，01+01=10)
- 序列的长度为 N，至少需要多少比特？$1+\log_2 N$

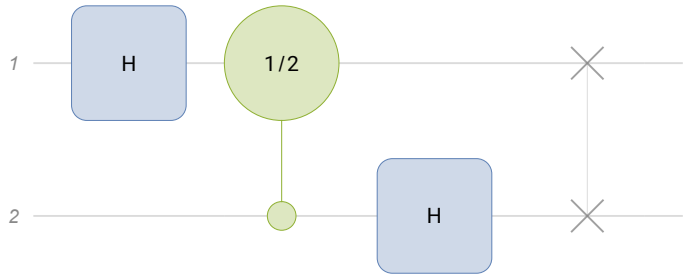
- 因式分解
  - $$|j\rangle \longrightarrow \mathrm{QFT}|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i jk/N} |k\rangle = \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left(|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle\right)$$
  - 上式右边的因式分解能帮助我们利用简单的门逻辑门设计 Fourier 变换线路。
- 写成大家喜闻乐见的bracket算符形式
  - $$\mathrm{QFT} := \frac{1}{\sqrt{N}} \sum_{j,k=0}^{N-1} \exp\left(2\pi i \frac{kj}{N}\right) |k\rangle\langle j|$$

量子线路

两比特 N = 4 ,n = 2

In[27]:=
```
qft2 = QuantumCircuitOperator["Fourier"];
% // TraditionalForm
```

Out[28]//TraditionalForm=



In[29]:=
```
qft2["Operators"][2]["Matrix"] // Normal // MatrixForm
```

Out[29]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}$$

In[30]:=
```
qft2steps = ComposeList[qft2["Operators"], QuantumState[{0, 1, 0, 0}]];
Grid[Transpose[{Style[#, Bold] & /@ {"初态", "经过H₁门", "经过CU", "经过H₂门", "经过SWAP"},
    #["Formula"] & /@ qft2steps[[ ;; ]]}], Frame → All, Alignment → Left]
```

Out[31]=

初态	$\lvert 01\rangle$
经过H_1门	$\frac{1}{\sqrt{2}} \lvert 01\rangle + \frac{1}{\sqrt{2}} \lvert 11\rangle$
经过CU	$\frac{1}{\sqrt{2}} \lvert 01\rangle + \frac{i}{\sqrt{2}} \lvert 11\rangle$
经过H_2门	$\frac{1}{2} \lvert 00\rangle - \frac{1}{2} \lvert 01\rangle + \frac{i}{2} \lvert 10\rangle - \frac{i}{2} \lvert 11\rangle$
经过SWAP	$\frac{1}{2} \lvert 00\rangle + \frac{i}{2} \lvert 01\rangle - \frac{1}{2} \lvert 10\rangle - \frac{i}{2} \lvert 11\rangle$

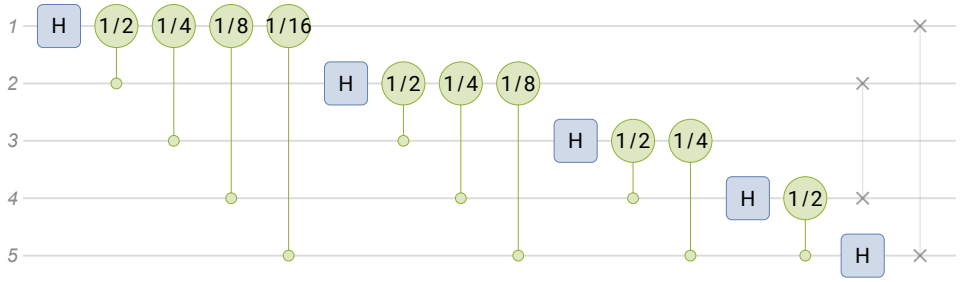
In[32]:=
```
Fourier[{0, 1, 0, 0}] // Rationalize(*初态等效的经典输入*)
```

Out[32]=

$$\left\{\frac{1}{2}, \frac{i}{2}, -\frac{1}{2}, -\frac{i}{2}\right\}$$

In[33]:=
```
qft5 = QuantumCircuitOperator[{"Fourier", 5}];
%["Diagram"](*最后交换高低位约定*)
```

Out[34]=



In[35]:=
```
(⌗["Matrix"] // Normal // MatrixForm) & /@ (qft5["Operators"][[2 ;; 5]])(*qft5的第二到第五个算符*)
```

Out[35]=

$$\left\{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&i\end{pmatrix}, \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{i\pi}{4}}\end{pmatrix}, \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{i\pi}{8}}\end{pmatrix}, \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{i\pi}{16}}\end{pmatrix}\right\}$$

In[36]:=
```
qft5[1/Sqrt[2] QuantumState["00001"] + 1/Sqrt[2] QuantumState["00011"]]["Matrix"] // Normal // Flatten // N;
SparseArray[{2 → 1/Sqrt[2], 4 → 1/Sqrt[2]}, 32] // Fourier;
% - %% // Chop
```

Out[38]=

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

相位估计

因式分解问题

量子傅里叶变换的一般应用

1 Quantum algorithms :
A survey of applications and end – to – end complexities

2 文中所说的“端到端”是指解决用户感兴趣的整个问题的成本，而不仅仅是运行作为完整解决方案子程序的特定量子电路的成本。