

# 基于 TDD 的量子模型检测中的可达性分析

## 硕士中期报告

高丁超  
导师：应圣钢

中国科学院软件研究所

2023 年 12 月 26 日



# 目录

- 1 毕业要求
- 2 背景介绍
- 3 解决方案

# 目录

## 1 毕业要求

## 2 背景介绍

### 3 解决方案



# 研究要求

## 发表论文要求：

至少作为排名前 3 的作者向 CCF-A/B 类高水平国际刊物/会议投稿了长文且评价意见不是很差。

**已投稿：** Xing Hong, Dingchao Gao 等. Image Computation for Quantum Transition Systems.

*ICCAD 2023 (CCF-B):*

投稿状态：拒稿

审稿人意见（5 分制）：2, 4, 4

*DAC 2024 (CCF-A):*

投稿状态：在审

预计反馈日期：2024 年 2 月 26 日

# 目录

- 1 毕业要求
- 2 背景介绍
- 3 解决方案

**标题:** 基于 TDD 的量子模型检测中的可达性分析

**总结:**

问题: 如何在量子系统中验证命题。

解决方案: 采用量子模型检测。

挑战: 原有的方法随着量子比特数量的增加, 资源需求指数级增长。

方法: 引入新的数据结构 TDD 对量子算法进行表示, 同时实现了优化算法进一步减少了时间消耗。

# 量子计算的关键概念

**量子比特 (Qubits):** the quantum version of the classic binary

**叠加态 (Superposition):**  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$

**纠缠 (Entanglement):**  $|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

**量子门 (Quantum Gates)**



# 量子门操作例子

**单量子门例子：**

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

**多量子门例子：**

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

**测量，比如 Z 基测量：** 状态为  $|0\rangle$  输出 1，状态为  $|1\rangle$  输出 -1。

# 量子计算例子

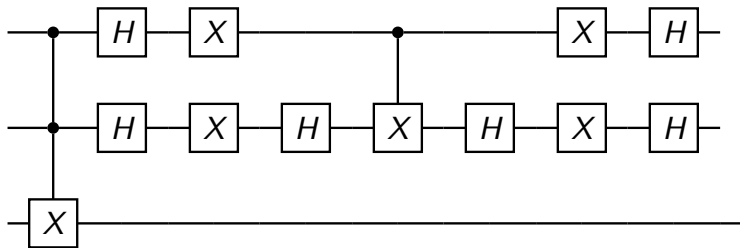


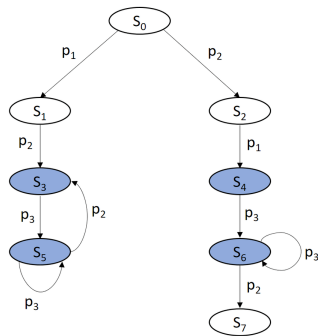
图: Grover\_3 算法的电路。

# 量子迁移系统

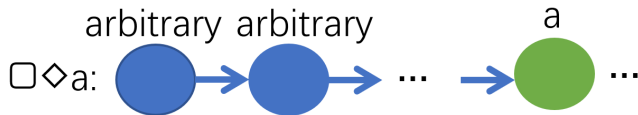
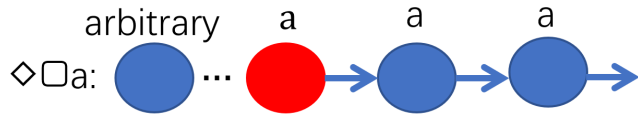
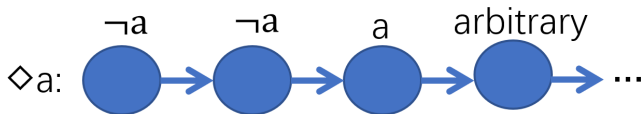
迁移系统 (transition system):  $(S, I, \Sigma, T)$

$$\text{where } \begin{cases} x = x_1, \dots, x_n \\ y = y_1, \dots, y_n \\ \sigma = \sigma_1, \dots, \sigma_m \end{cases}$$

量子迁移系统:  $(\mathcal{H}, \mathcal{H}_0, Act, \{U_\alpha, \alpha \in Act\})$



# 可达性问题



# 量子模型检测例子

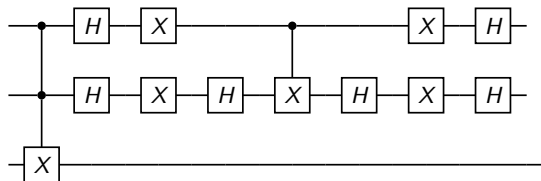


图: Grover\_3 算法的电路。

oracle 为  $ccx$ , 即  $O|x\rangle|y\rangle = |x\rangle|f(x) \oplus y\rangle, f(x) = x_1 \wedge x_2$ 。

**model:**  $(\mathcal{H}_8, S = span\{|++-\rangle, |11-\rangle\}, \{1\}, \mathcal{T}_1), \mathcal{T}_1 = (2|\Psi\rangle\langle\Psi| - I)O$

**property:**  $\mathcal{T}_1(S) = S$

# 张量决策图 (TDD)

**TDD 定义：**由节点集  $V$ 、边集  $E$ 、索引函数  $index$ 、值函数  $value$ 、低高边映射  $low/high$  和权重  $w$  组成。

- 节点集  $V$  分为非终端节点  $V_N$  和终端节点  $V_T$ ，且有唯一根节点  $r_{\mathcal{F}}$ 。
- 边集  $E$  包含所有低边  $(v, low(v))$  和高边  $(v, high(v))$ 。
- 索引函数  $index$  分配索引，值函数  $value$  赋予终端节点复数值， $w$  为边赋权重，特别是根边权重  $w_{\mathcal{F}}$ 。

# TDD 例子

$$P = \frac{1}{6} \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 \end{bmatrix}$$

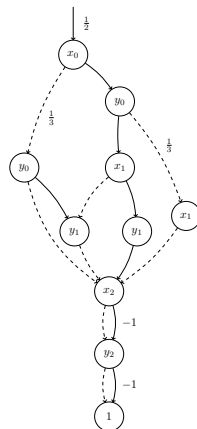
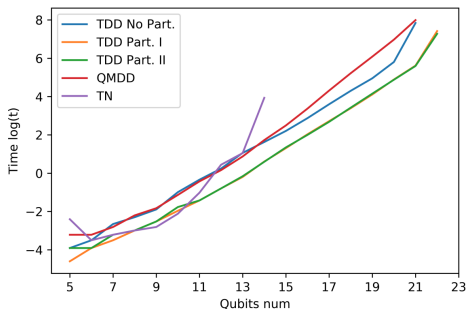


图: 可以用 10 个 TDD 节点表示一个  $8 \times 8$  的矩阵。其中 TDD 虚线表示低点，实线表示高边。

## 相关工作效率比较



图：应用不同技术对 QFT 算法进行模拟的时间对比

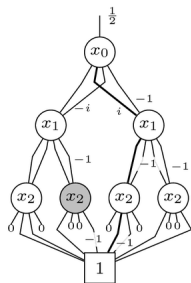
TDD No part, TDD part I, TDD part II 为不同的 TDD 收缩算法。

QMDD 为量子多值决策图，是一种常用的模型检测方法。

TN 为 Google 的 tensor network，是一种常用的张量方法。



# 相关工作-QMDD



TDD 只有高边和低边，表示更简洁。

图: 一个 QMDD 的示例。

# 目录

- ## 1 毕业要求

- ## 2 背景介绍

- ### 3 解决方案

## 解决方案简介：

**基本方法：**将转移关系和初态转化为 TDD 表示，然后计算系统的状态转移。

**改进算法：**

- **addition partition:** 寻找依赖最多的索引项，从而分割线路。
- **contraction partition:** 通过预设的参数进行线路分割。

**创新点：**

- 通过 TDD，可以自动化的验证更大规模的量子算法。
- 通过 C++ 重构，实现了比过去 TDD 更好的运行效率。

# addition partition

- 将量子电路转换为索引依赖图  $G$ 。
- 通过图  $G$  的连通度选择索引进行电路分割。

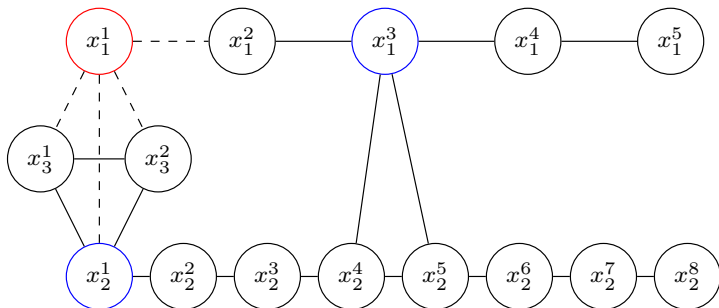


图: Grover\_3 电路的索引依赖图。对索引项  $x_3^1, x_3^2$  进行线路分割, 效果更好。

## Contraction partition

- 确定预设参数  $k_1$  和  $k_2$ 。
- 分割电路，每部分包括最多  $k_1$  个量子比特，连接最多  $k_2$  个多比特门。

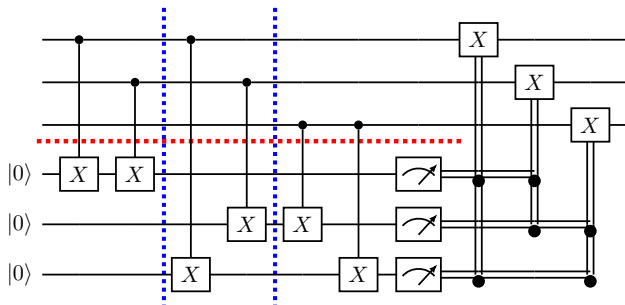


图: 对 bit flip 电路进行划分, 其中  $k_1=3, k_2=2$ 。

# 工作成果

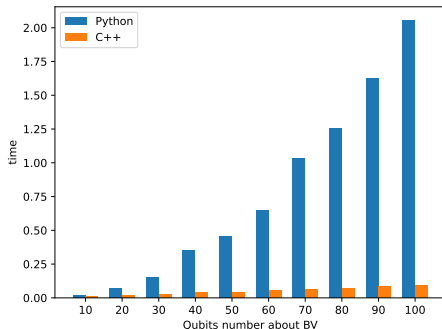
benchmark	basic	addition	contraction
Grover 20	~5 分	~4 分	~4 秒
Quantum Fourier Transform 20	~20 分	~11 分	<1 秒
Quantum Random walk 20	~6 分	~4 分	~15 秒
Bernstein-Vazirani 100	~7 秒	~7 秒	~0.4 秒
GHZ 500	~3 秒	~1.5 秒	~1.7 秒

表: 对不同量子算法计算一步迁移的时间消耗

- 对于有特殊结构的算法, 如 GHZ 算法, addition partiton 有更好的执行效率。
- 对于一般的电路, contraction partition 的执行效率更好。

## 工作成果

- 通过 C++ 重构 TDD，改进了内存管理，从而加快了计算效率。



图：用 python 和 c++ 不同版本的 TDD 运行 Bernstein-Vazirani 算法的时间效率比较

# 未来计划

- 应用等价性，可以化简数据结构。
- 结合优化算法与 C++ 的优势，进一步提高执行效率。

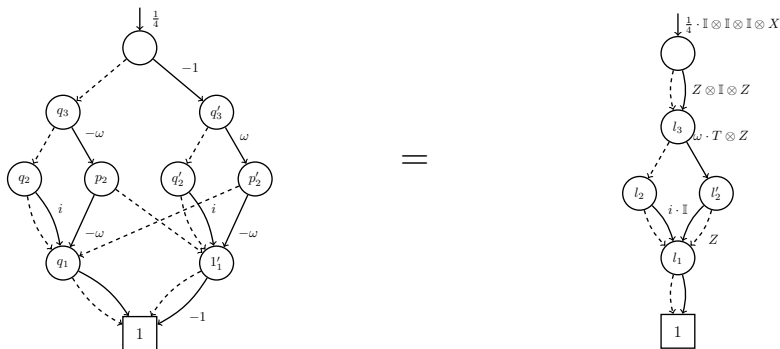


图: Local Invertible Map-DD



# 谢谢