A Multi-tool for your Quantum Algorithmic Toolbox

Shelby Kimmel
Middlebury College

Based on work with
• Stacey Jeffery: arXiv: 1704.00765 (Quantum vol 1 p 26)
• Bohua Zhan, Avinatan Hassidim, arXiv:1101.0796 (ITCS 2012)
Quantum Tools for Classical Problems

- Quantum Fourier Transform
- Grover Search
- Quantum Approximate Optimization Algorithm (QAOA)
Quantum Tools for Classical Problems

- Quantum Fourier Transform (good if highly structured)
- Grover Search (good if unstructured)
- Quantum Approximate Optimization Algorithm (QAOA) (hard to analyze)
Multi-tool

✓ Structured
✓ Unstructured
✓ Easy creation
✓ Easy, provable, non-quantum analysis

❖ Query model
Multi-tool

✓ Structured
✓ Unstructured
✓ Easy creation
✓ Easy, provable, non-quantum analysis

Classical Problem → st-connectivity problem → quantum span-program algorithm

Reduce Analyze Apply
Reduction

Problem: Bit string $x = x_1x_2x_3$ contains a 1?
Reduction

Problem: Bit string $x = x_1x_2x_3$ contains a 1?
Reduction (Decision Tree Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Turn classical algorithm into decision tree:

**Algorithm:**
- For each bit:
  - If 1: Output 1, end
- Output 0
Reduction (Decision Tree Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Turn classical algorithm into decision tree:

**Algorithm:**
- For each bit:
  - If 1: Output 1, end
- Output 0
Reduction (Decision Tree Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Turn classical algorithm into decision tree:

Algorithm:
- For each bit:
  - If 1: Output 1, end
- Output 0
Reduction (Decision Tree Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Turn classical algorithm into decision tree:

**Algorithm:**
- For each bit:
  - If 1: Output 1, end
- Output 0
Reduction (Decision Tree Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Turn classical algorithm into decision tree:

Algorithm:
- For each bit:
  - If 1: Output 1, end
- Output 0
Reduction (Decision Tree Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Turn classical algorithm into decision tree:

**Algorithm:**
- For each bit:
  - If 1: Output 1, end
- Output 0
Reduction (Decision Tree Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Turn classical algorithm into decision tree:

$$x = 011$$
Reduction (Decision Tree Approach)

Problem: Bit string $x = x_1 x_2 x_3$ contains a 1?

Idea: Turn classical algorithm into decision tree:

$x = 000$

![Decision Tree Diagram]
Reduction (Parallel Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Take advantage of parallel paths

$\begin{align*}
  x_1 &= 1 \\
  x_2 &= 1 \\
  x_3 &= 1 \\
\end{align*}$
Reduction (Parallel Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Take advantage of parallel paths

$x = 011$

$x_1 = 1$

$x_2 = 1$

$x_3 = 1$
Reduction (Parallel Approach)

Problem: Bit string $x = x_1x_2x_3$ contains a 1?

Idea: Take advantage of parallel paths

$x = 000$

$x_1 = 1$

$x_2 = 1$

$x_3 = 1$

[Beigi, Taghavi `12], [JJKP, `18]
Multi-tool

✓ Structured
✓ Unstructured
✓ Easy creation
✓ Easy, provable, non-quantum analysis

Classical Problem → st-connectivity problem → quantum span-program algorithm

Reduce  Analyze  Apply
Analysis

Assign a weight to each edge:
Analysis

For each input, can calculate
• effective resistance (if path)
• effective capacitance (if cut)
Analysis

Then Quantum Query Complexity is:

\[ O(\sqrt{(\text{max } C)(\text{max } R)}) \]

max \( R \): max effective resistance over connected instances
max \( C \): max effective capacitance over not connected instances

[Belovs, Reichardt, `12], [JJKP, `18]
Multi-tool

- Structured
- Unstructured
- Easy creation
- Easy, provable, non-quantum analysis
Algorithm

Apply phase estimation to a unitary that is a product of two unitaries,
• One depends on input $x$
• One depends on underlying graph

Space complexity scales $\log(\text{no. edges, vertices})$ in graph.

[Belovs, Reichardt `12]
Performance

• Read-once Boolean formulas (query optimal) [JK]
• Total connectivity (query optimal) [JJKP]
• Cycle detection (query optimal) [DKW]
• Even length cycle detection [DKW]
• Bipartiteness (query optimal) [DKW]
• Directed st-connectivity (query optimal) (Beigi, Taghavi `19)
• Directed smallest cycle (query optimal) (Beigi, Taghavi `19)
• Topological sort (Beigi, Taghavi `19)
• Connected components (Beigi, Taghavi `19)
• Strongly connected components (Beigi, Taghavi `19)
• k-cycle at vertex v (Beigi, Taghavi `19)
• st-connectivity (query optimal) (Reichardt, Belovs `12)
• Maximum bipartite matching (Lin, Lin `16; Beigi and Taghavi `19)
• Maximum matching (K, Witter, in preparation)
• Super-polynomial speed-up for game evaluation [ZHK]

*There are alternative optimal algorithms for some problems
Summary
• A multi-tool for algorithm design that is accessible

Open Problems
• How to set weights?
• When is this approach optimal?

Funding:
- Middlebury
- ARO

People:
- Stacey Jeffery
- Michael Jarret
- Alvaro Piedrafita
- Teal Witter
- Kai De Lorenzo