Path Detection: A Quantum Computing Primitive

Shelby Kimmel

Middlebury College

Based on work with
Stacey Jeffery: arXiv: 1704.00765 (Quantum vol 1 p 26)
Michael Jarret, Stacey Jeffery, Alvaro Piedrafita, in progress
How to make quantum algorithms accessible?
How to make quantum algorithms accessible?

• Need quantum algorithmic primitives
How to make quantum algorithms accessible?

• Need quantum algorithmic primitives
  1. Apply to a wide range of problems
  2. Easy to understand and analyze (without knowing quantum mechanics)
How to make quantum algorithms accessible?

• Need quantum algorithmic primitives
  1. Apply to a wide range of problems
  2. Easy to understand and analyze (without knowing quantum mechanics)
     – Ex: Searching unordered list of $n$ items
       – Classically, takes $\Omega(n)$ time
       – Quantumly, takes $O(\sqrt{n})$ time
How to make quantum algorithms accessible?

• Need quantum algorithmic primitives
  1. Apply to a wide range of problems
  2. Easy to understand and analyze (without knowing quantum mechanics)
    - Ex: Searching unordered list of $n$ items
      - Classically, takes $\Omega(n)$ time
      - Quantumly, takes $O(\sqrt{n})$ time

• New primitive: $st$-connectivity
Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
   1. Applies to a wide range of problems
   2. Easy to understand (without knowing quantum mechanics)
C. Applications and performance of algorithm
$st$-connectivity

$st$ – connectivity: is there a path from $s$ to $t$?
$st$-connectivity

$st$ – connectivity: is there a path from $s$ to $t$?
Black Box Model

Edge label

- $e_i = 1$ if $i^{th}$ edge is there
- $e_i = 0$ if edge is not there

Let $\mathcal{H}$ be the set of graphs $G$ that the black box might contain.
Figure of Merit

- Query Complexity
  - Number of uses (queries) of the black box
  - All other operations are free

- Under mild assumption, for our algorithm,
  quantum query complexity $\cong$ quantum time complexity
Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
   1. Applies to a wide range of problems
   2. Easy to understand (without knowing quantum mechanics)
C. Applications and performance of algorithm
Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
   1. Applies to a wide range of problems
      • Evaluating Boolean formulas reduces to st-connectivity
   2. Easy to understand (without knowing quantum mechanics)
C. Applications and performance of algorithm
Boolean Formulas

\[ \bigwedge \quad \text{AND: outputs 1 if all inputs are 1} \]

\[ \bigvee \quad \text{OR: outputs 1 if any input is 1} \]

\[ x_i \quad \text{Value 0 or 1} \]

\[ f(x) \]

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \]

\[ x_5 \quad x_6 \quad x_7 \quad x_8 \quad x_9 \quad x_{10} \]
Boolean Formulas

Read-once: $x_i$’s not fan out
Boolean Formulas

**Read-once**: $x_i$’s not fan out

**Read-many**: $x_i$ have fan out
Boolean Formula Applications

- Logic
- Designing electrical circuits
- Game theory (deciding who will win a game)
- Combinatorics and graph problems
- Linear programming
- Testing potential solution to an NP-complete problem
Application to Boolean Formulas

\( \wedge \): outputs 1 if all input subformulas have value 1

\( s \) and \( t \) are connected if all subgraphs are connected.
Application to Boolean Formulas

AND: outputs 1 if all input subformulas have value 1

\( s \) and \( t \) are connected if all subgraphs are connected

OR: outputs 1 if any input subformulas have value 1

\( s \) and \( t \) are connected if any subgraph is connected
Application to Boolean Formulas

\[ f(x) = \bigwedge_{i=1}^{10} \bigvee x_i \]

Diagram showing the Boolean formula tree and its corresponding graph with nodes \( x_1, x_2, \ldots, x_{10} \) and connections.
Application to Boolean Formulas

- If we put edges where \( x_i = 1 \), \( s \) and \( t \) are connected iff \( f(x) = 1 \).
Application to Boolean Formulas

\[ f(x) = \bigwedge \bigvee x_1 x_2 x_3 x_4 \bigvee x_5 x_6 \bigvee x_7 x_8 x_9 \]

\[ s \xrightarrow{x_1} x_3 \xrightarrow{x_5} x_7 \xrightarrow{x_8} x_9 \xrightarrow{x_4} x_6 \xrightarrow{x_5} x_7 \xrightarrow{x_8} x_9 \xrightarrow{x_4} x_6 \]
Application to Boolean Formulas

\[ f(x) = x_1 \land (x_2 \lor (x_3 \land x_4)) \land x_{10} \]

Diagram showing a tree structure with variables and logical operators.
Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
   1. Applies to a wide range of problems
      • Evaluating Boolean formulas reduces to st-connectivity
   2. Easy to understand (without knowing quantum mechanics)
C. Applications and performance of algorithm
Effective Resistance

Graph $G$: $S \rightarrow t$
Effective Resistance
Effective Resistance

$R_{s,t}(G)$ unit resistor

1 unit resistor
Effective Resistance

$R_{s,t}(G)$ unit resistor

1 unit resistors

Properties of $R_{s,t}(G)$
- Small if many short paths from $s$ to $t$
- Large if few long paths from $s$ to $t$
- Infinite if $s$ and $t$ not connected
Effective Resistance

\[ s \rightarrow t \]

1 unit of flow

\[ S \rightarrow T \]

1 unit of flow
Effective Resistance

Valid flow:
• 1 unit in at $s$
• 1 unit out at $t$
• At all other nodes, zero net flow

\[ f \text{ unit of flow} \]
\[ 1 - f \text{ unit of flow} \]
Effective Resistance

Flow energy:
\[ \sum_{edges} (\text{flow on edge})^2 \]
Effective Resistance

Flow energy: \[ \sum_{edges} (flow \text{ on edge})^2 \]

Effective Resistance: \( R_{s,t}(G) \)
Smallest energy of any valid flow from \( s \) to \( t \) on \( G \).
Effective Capacitance

Graph $G'$:
Effective Capacitance

1 unit capacitors

0 resistance wires (short circuit)
Effective Capacitance

\[ C_{s,t}(G') \] unit capacitor

Properties of \( C_{s,t}(G') \)
- Small if many small cuts
- Large if one large cuts
- Infinite if \( s \) and \( t \) connected

1 unit capacitors

0 resistance wires (short circuit)
Effective Capacitance

Valid potential energy:
- 1 at $s$
- 0 at $t$
- Potential energy difference is 0 across edge
Effective Capacitance

Valid potential energy:
- 1 at $s$
- 0 at $t$
- Potential energy difference is 0 across edge
Effective Capacitance

Cut energy:
$$\sum_{edges} \left(\text{Potential Energy Difference}\right)^2$$

Effective Capacitance: $C_{s,t}(G')$
Smallest cut energy of any valid potential energy between $s$ to $t$ on $G'$.
Algorithm Performance:

st-connectivity algorithm complexity =

$$O\left(\sqrt{\max_{G \in \mathcal{H}: \text{connected}} R_{s,t}(G)} \sqrt{\max_{G' \in \mathcal{H}: \text{not connected}} C_{s,t}(G')}\right)$$
Algorithm Performance:

st-connectivity algorithm complexity =

\[
O\left(\sqrt{\max_{G \in \mathcal{H}: \text{connected}} R_{s,t}(G)} \sqrt{\max_{G' \in \mathcal{H}: \text{not connected}} C_{s,t}(G')}\right)
\]

[Belovs, Reichard, '12]  
[JJKP, in progress]
Example

What is quantum complexity of deciding \( \text{AND}(x_1, x_2, \ldots, x_N) \), promised

- All \( x_i = 1 \), or
- At least \( \sqrt{N} \) input variables are 0.
Example

What is quantum complexity of deciding $\text{AND}(x_1, x_2, \ldots, x_N)$, promised
• All $x_i = 1$, or
• At least $\sqrt{N}$ input variables are 0.

What is quantum complexity of deciding if
• $s$ and $t$ are connected, or
• At least $\sqrt{N}$ edges are missing
What is quantum complexity of deciding if
- $s$ and $t$ are connected, or
- At least $\sqrt{N}$ edges are missing

\[
\sqrt{\max_{G \in \mathcal{H}: \text{connected}} R_{s,t}(G)} \quad \sqrt{\max_{G' \in \mathcal{H}: \text{not connected}} C_{s,t}(G')}
\]
Example

What is quantum complexity of deciding if
- $s$ and $t$ are connected, or
- At least $\sqrt{N}$ edges are missing

$$\sqrt{\max_{G \in \mathcal{H} : \text{connected}} R_{s,t}(G)} \sqrt{\max_{G' \in \mathcal{H} : \text{not connected}} C_{s,t}(G')}$$

$$\max_{G \in \mathcal{H} : \text{connected}} R_{s,t}(G) = N$$
What is quantum complexity of deciding if

- $s$ and $t$ are connected, or
- At least $\sqrt{N}$ edges are missing

\[
\sqrt{\max_{G \in \mathcal{H}: \text{connected}} R_{s,t}(G)} \quad \sqrt{\max_{G' \in \mathcal{H}: \text{not connected}} C_{s,t}(G')} \]

Example
What is quantum complexity of deciding if
• $s$ and $t$ are connected, or
• At least $\sqrt{N}$ edges are missing

\[ \sqrt{ \max_{G \in \mathcal{H}: \text{connected}} R_{s,t}(G) } \quad \sqrt{ \max_{G' \in \mathcal{H}: \text{not connected}} C_{s,t}(G') } \]
What is quantum complexity of deciding if
• \( s \) and \( t \) are connected, or
• At least \( \sqrt{N} \) edges are missing

\[
\sqrt{\max_{G \in \mathcal{H}: \text{connected}} R_{s,t}(G)} - \sqrt{\max_{G' \in \mathcal{H}: \text{not connected}} C_{s,t}(G')}
\]
What is quantum complexity of deciding if
- \( s \) and \( t \) are connected, or
- At least \( \sqrt{N} \) edges are missing

\[
\sqrt{\max_{G \in H: \text{connected}} R_{s,t}(G)} \cdot \sqrt{\max_{G' \in H: \text{not connected}} C_{s,t}(G')}
\]

\[
\max_{G' \in H: \text{not connected}} C_{s,t}(G') = \sqrt{N} \times \left(\frac{1}{\sqrt{N}}\right)^2 = \frac{1}{\sqrt{N}}
\]
What is quantum complexity of deciding if
• $s$ and $t$ are connected, or
• At least $\sqrt{N}$ edges are missing

$$\sqrt{\max_{G \in \mathcal{H}: \text{connected}} R_{s,t}(G)} \quad \sqrt{\max_{G \in \mathcal{H}: \text{not connected}} R_{s',t'}(G')}$$

Quantum complexity is $O(N^{1/4})$
What is quantum complexity of deciding if
• $s$ and $t$ are connected, or
• At least $\sqrt{N}$ edges are missing

Quantum complexity is $O\left(\frac{1}{\sqrt{N}}\right)$

Randomized classical complexity is $\Omega\left(N^{1/2}\right)$
New Example

Connectivity – is every vertex connected to every other vertex?
New Example

Connectivity – is every vertex connected to every other vertex?

Connectivity =
(st – conn) ∧ (su – conn) ∧ (uv – conn) ...

[Diagram of a connected graph with vertices labeled s, t, 1, 2, 3, 4, 5, 6, 7]
Connectivity – is every vertex connected to every other vertex?

Connectivity =
(st − conn) ∧ (su − conn) ∧ (uv − conn) …
New Example

Connectivity – is every vertex connected to every other vertex?

Results:
• Worst case: $O(n^{3/2})$ ($n = \#$ vertices)
• Promised
  • YES – diameter is $D$
  • NO – every connected component has at most $n^*$ vertices
  • $O(\sqrt{nn^*D})$
New Example

Connectivity – is every vertex connected to every other vertex?

Results:
- Worst case: $O(n^{3/2})$ ($n = \# \text{ vertices}$)
- Promised
  - YES – diameter is $D$
  - NO – every connected component has at most $K$ vertices
  - $O(\sqrt{nKD})$

(Diameter result previously discovered by Arins using slightly different approach)
The Algorithm

Span Program
- Span vectors
- Target vector

The input to the problem determines which subset of span vectors are allowed.
The Algorithm

Span Program
• Span vectors
• Target vector

The input to the problem determines which subset of span vectors are allowed.

If target vector is in span of the allowed span vectors, then function evaluates to 1 on that input. Otherwise, evaluates to 0.

Thus span program encodes a function.
The Algorithm

Span Program
- Span vectors
- Target vector

The input to the problem determines which subset of span vectors are allowed.

If target vector is in span of the allowed span vectors, then function evaluates to 1 on that input. Otherwise, evaluates to 0.

Thus span program encodes a function.

Infinite number of span programs can encode the same function.

Given a span program, can create a quantum algorithm to evaluate the corresponding function (create a quantum walk whose dispersion operators are based on the vectors)
The Algorithm

Span Program
- Span vectors
- Target vector

Given a span program, can create a quantum algorithm to evaluate the corresponding function (create a quantum walk whose dispersion operators are based on the vectors)
The Algorithm

Span Program
- Span vectors
- Target vector

Given a span program, can create a quantum algorithm to evaluate the corresponding function (create a quantum walk whose dispersion operators are based on the vectors)

The efficiency of the span program is a (relatively) simple function of the vectors.
The Algorithm

Span Program
- Span vectors
- Target vector

Given a span program, can create a quantum algorithm to evaluate the corresponding function (create a quantum walk whose dispersion operators are based on the vectors)

The efficiency of the span program is a (relatively) simple function of the vectors.

There is always a span program algorithm that is optimal (and many that are not optimal.)
Open Questions and Current Directions

• When is our algorithm optimal for Boolean formulas? (Especially partial/read-many formulas)
• Are there other problems that reduce to st-connectivity? (Perhaps all?)
• What is the classical time/query complexity of st-connectivity in the black box model? Under the promise of small capacitance/resistance?
• Does our reduction from formulas to connectivity give good classical algorithms too?
• How to choose weights?