Title: What does the effective resistance of electrical circuits have to do with quantum algorithms?

Date: Dec 09, 2016 11:00 AM

URL: http://pirsa.org/16120025

Abstract: I will answer the question in the title. I will also describe a new quantum algorithm for Boolean formula evaluation and an improved analysis of an existing quantum algorithm for st-connectivity. Joint work with Stacey Jeffery.

What does the effective resistance of electrical circuits have to do with quantum algorithms?

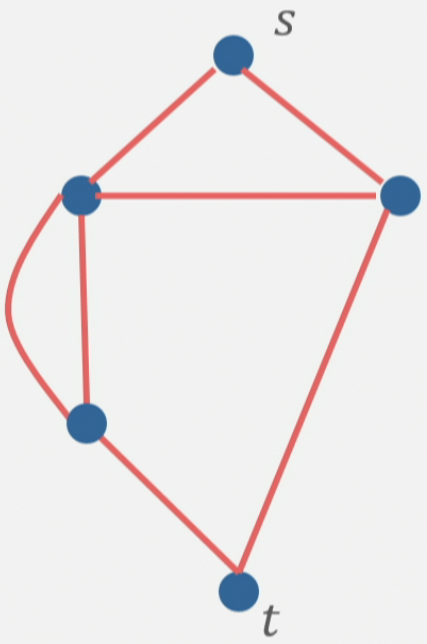
Shelby Kimmel
Stacey Jeffery (Caltech)

Joint Center for Quantum Information and Computer Science

Perimeter
12/9/2016

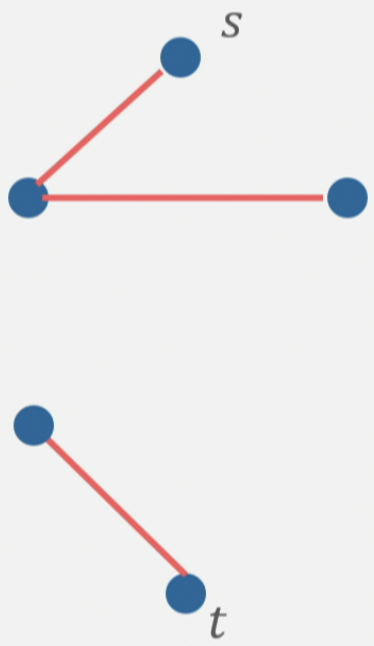
(Simplest) Answer

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



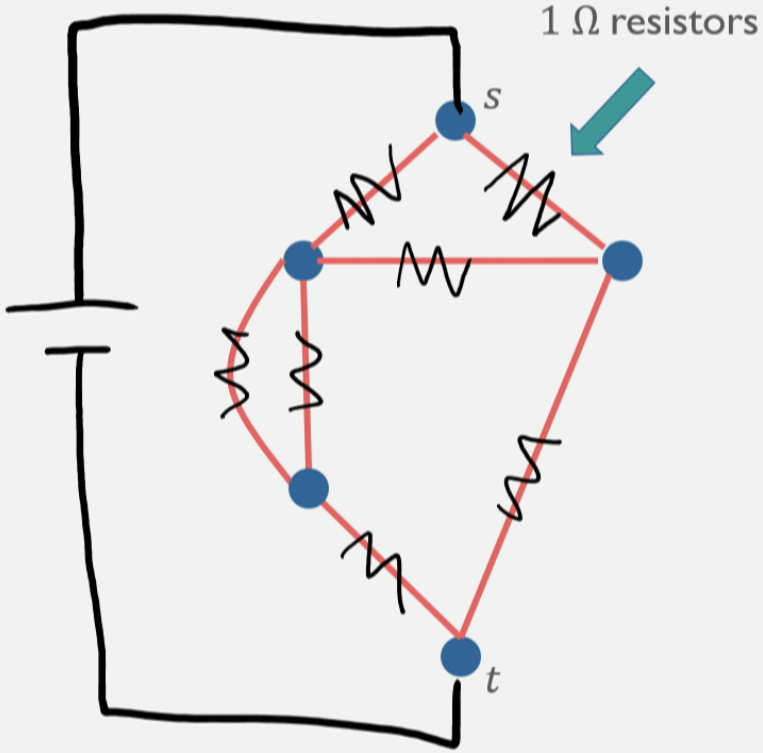
(Simplest) Answer

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



(Simplest) Answer

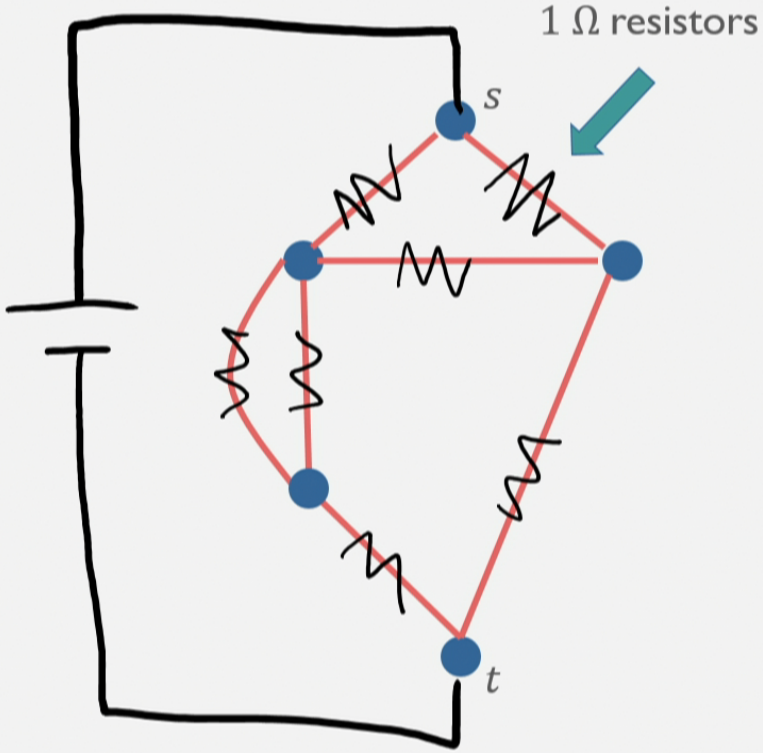
We can turn this into a circuit by attaching leads to s and t, and putting $1\ \Omega$ resistors wherever edges exist.



(Simplest) Answer

Speed of quantum algorithm for st-connectivity depends on effective resistance of this circuit! (Lower effective resistance -> quicker detection of path)

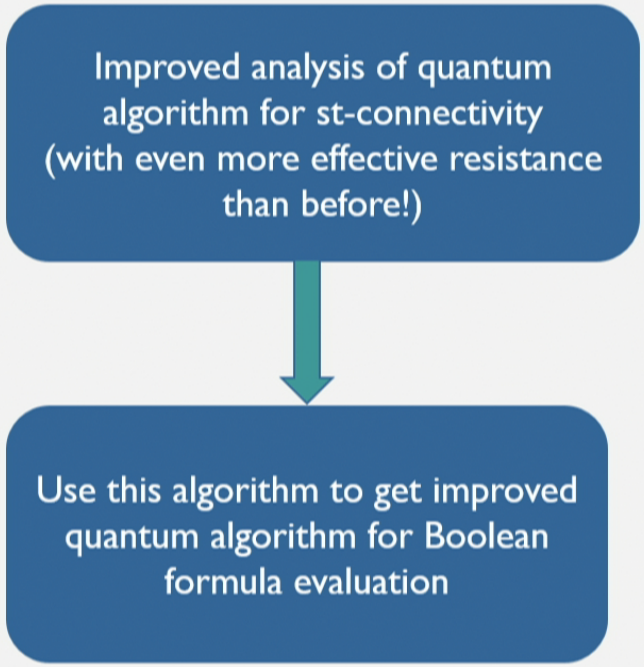
[Belovs, Reichardt '12]



Applications of st-Connectivity

- Important (social) network problem
- Problem is a useful subroutine for many problems
  - Is there a length-k path? [Belovs, Reichardt '12]
  - Is a graph a forest? [Cade, Montanaro, Belovs '16]
  - Is a graph bipartite? [Cade, Montanaro, Belovs '16]

Our results:


Improved analysis of quantum algorithm for st-connectivity (with even more effective resistance than before!)
Use this algorithm to get improved quantum algorithm for Boolean formula evaluation

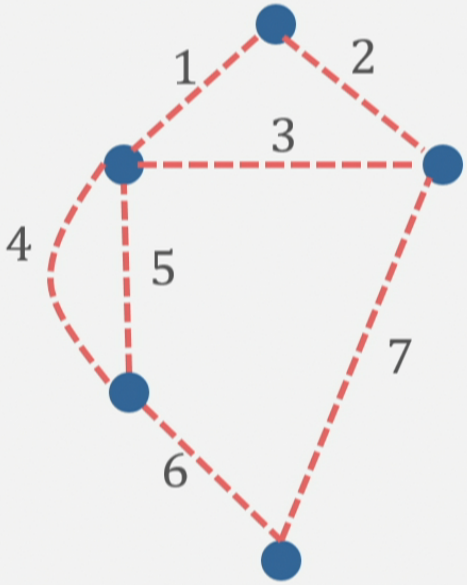
Outline

- Previous algorithm for st-connectivity
- Improved analysis for planar graphs
- Application to Boolean formulas

Black Box Algorithm

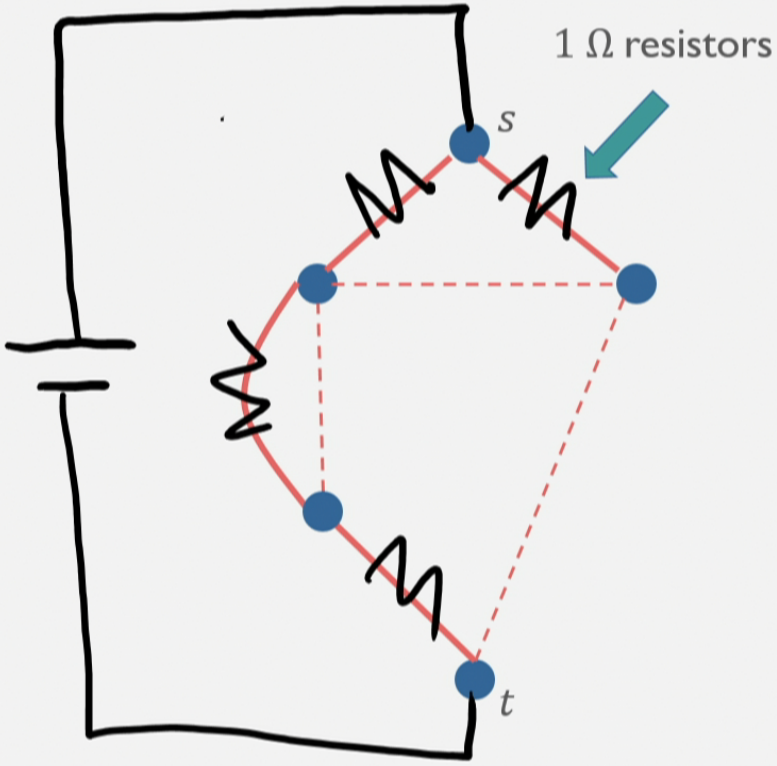
Edge label

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



Previous Quantum Algorithm

$R(G)$ is the effective resistance of the circuit created by attaching a voltage between s and t, and $1\ \Omega$ resistors at all edges.



Previous Quantum Algorithm

st-connectivity algorithm time/queries ~

$$\sqrt{\max_{G:connected} R(G)} \sqrt{\max_{G:not\ connected} |G|}$$

# of edges in graph G

[Belovs, Reichardt '12]

Previous Quantum Algorithm

st-connectivity algorithm time/queries ~

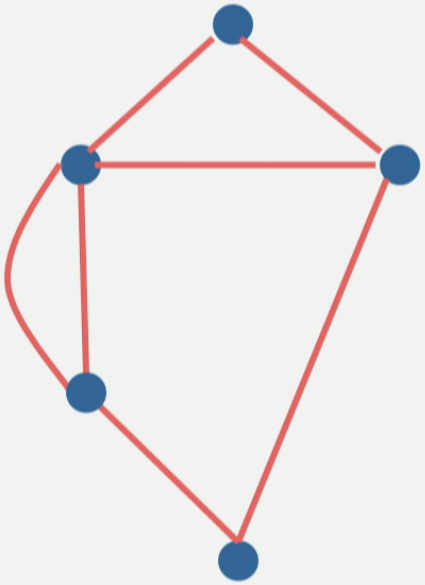
$$\sqrt{\max_{G:connected} R(G)} \sqrt{\max_{G:not\ connected} |G|}$$

# of edges in graph G

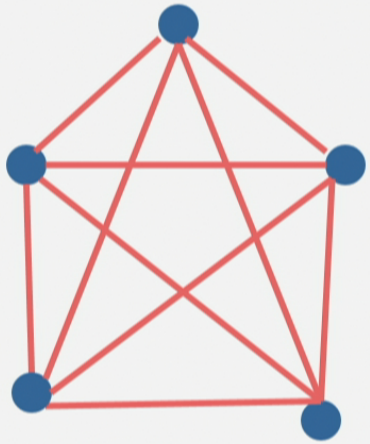
[Belovs, Reichardt '12]

Planar Graph

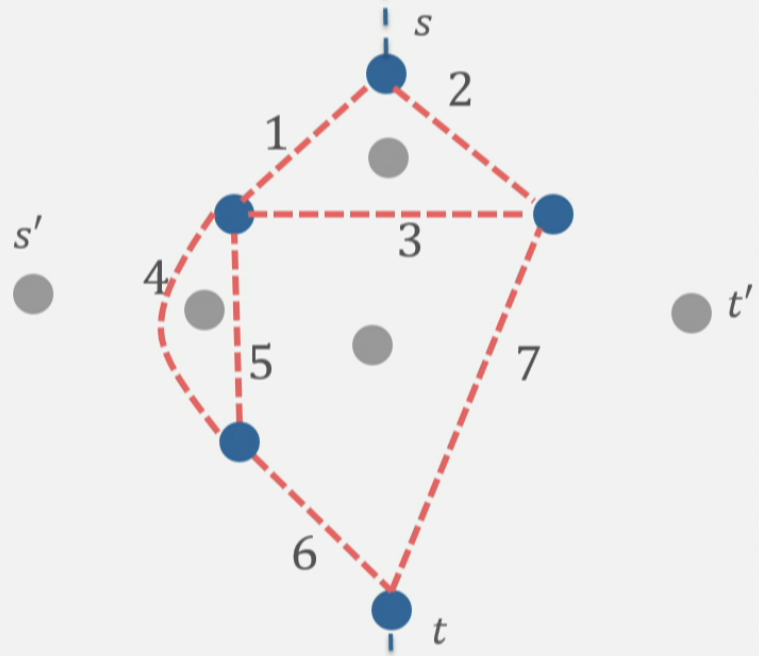
Planar



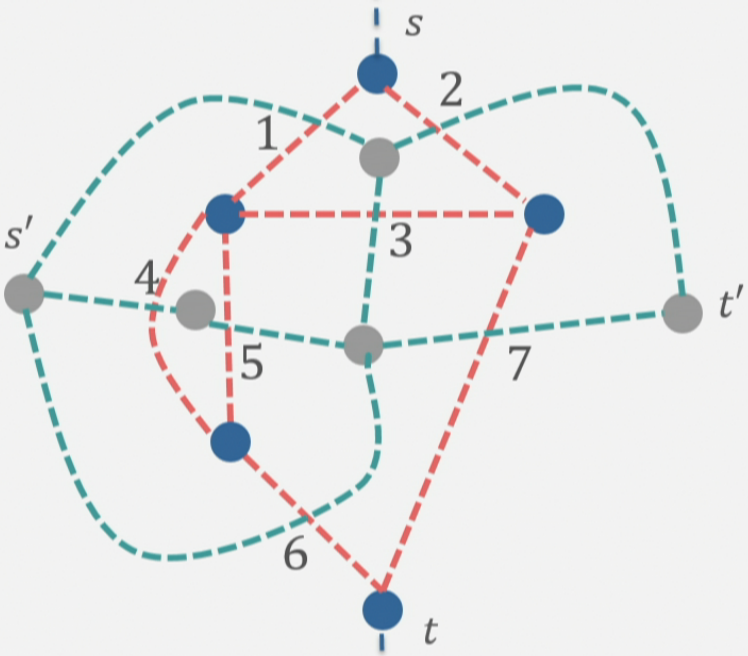
Not Planar



Planar Graph Dual(ish)

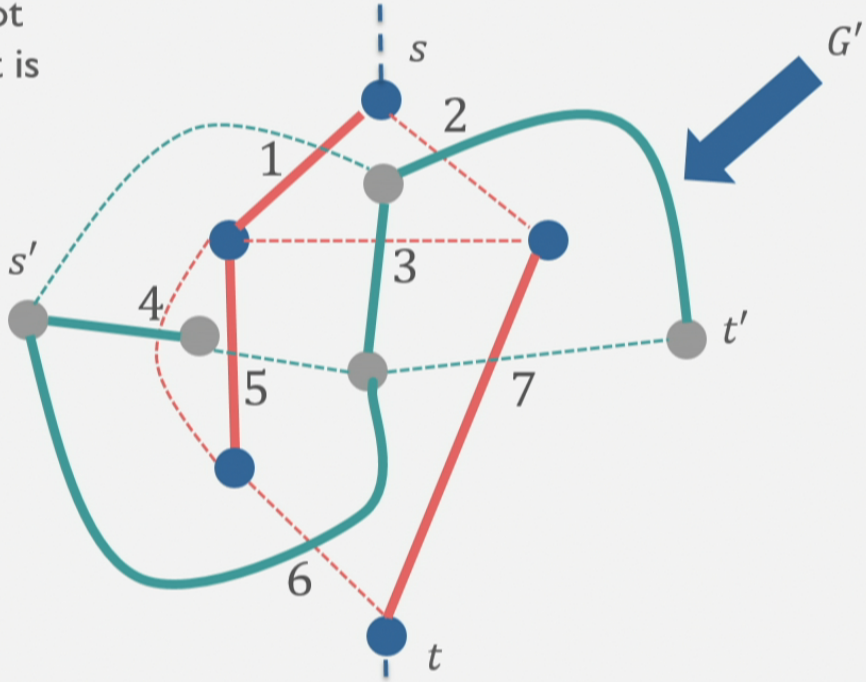


Planar Graph Dual(ish)

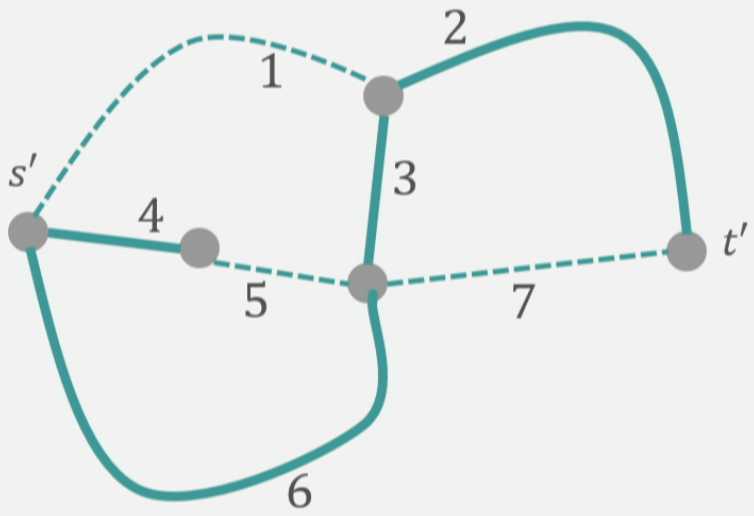


Planar Graph Dual(ish)

- If an edge is not present in G, it is present in G'

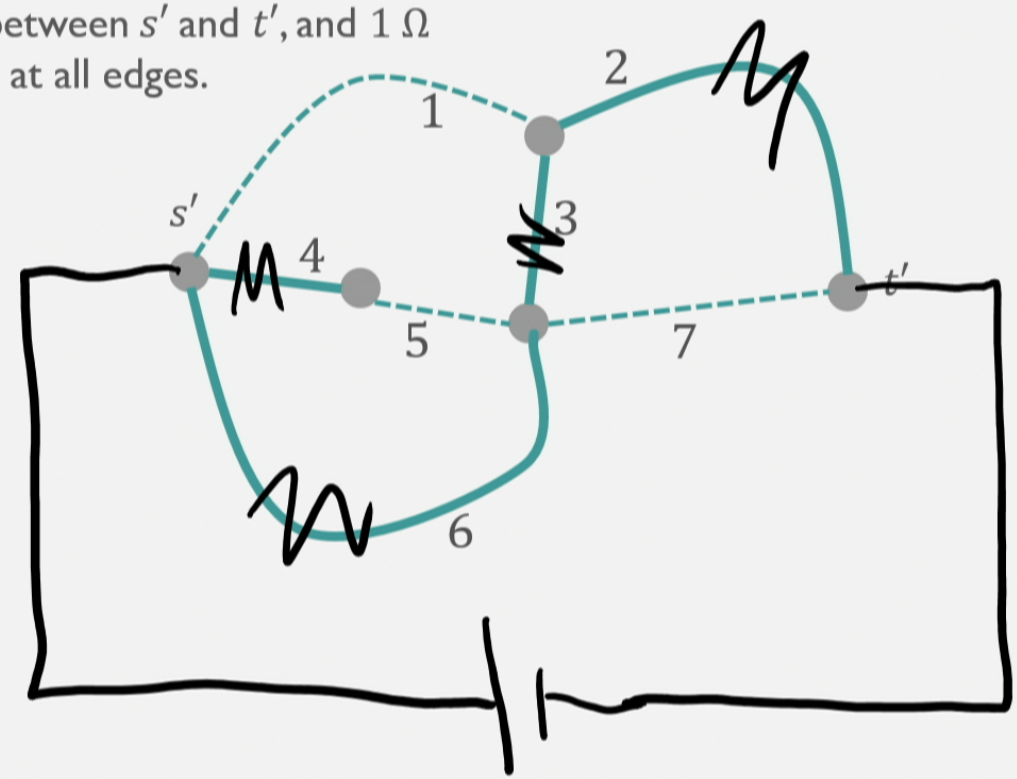


Planar Graph Dual(ish)



Planar Graph Dual(ish)

$R(G')$ is the effective resistance of the circuit created by attaching a voltage between s' and t', and 1 Ω resistors at all edges.



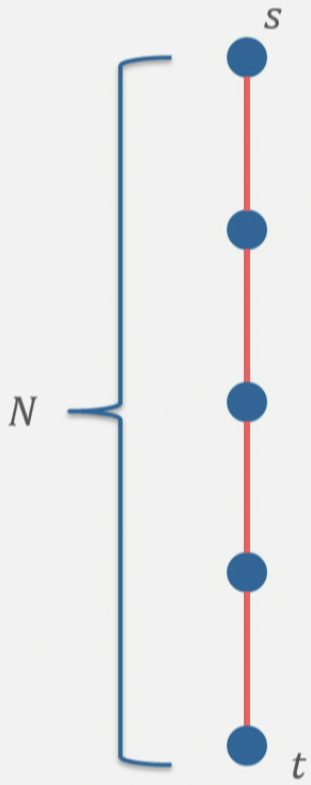
Improved Quantum Algorithm for st-connectivity

Planar graph† st-connectivity algorithm time/queries =

$$\sqrt{\max_{G:connected} R(G)}\ \sqrt{\max_{G:not\ connected} R(G')}$$

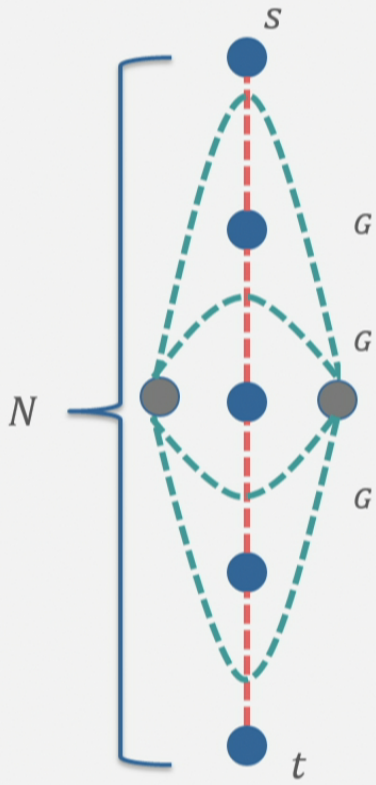
† with s, t on same face

Example 1



$$\max_{G:connected} R(G) = N$$

Example 1

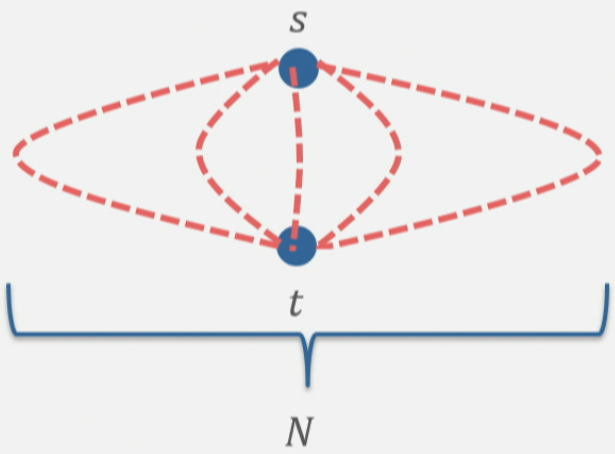


$$\max_{G:connected} R(G) = N$$

$$\max_{G:not\ connected} |G| = N - 1$$

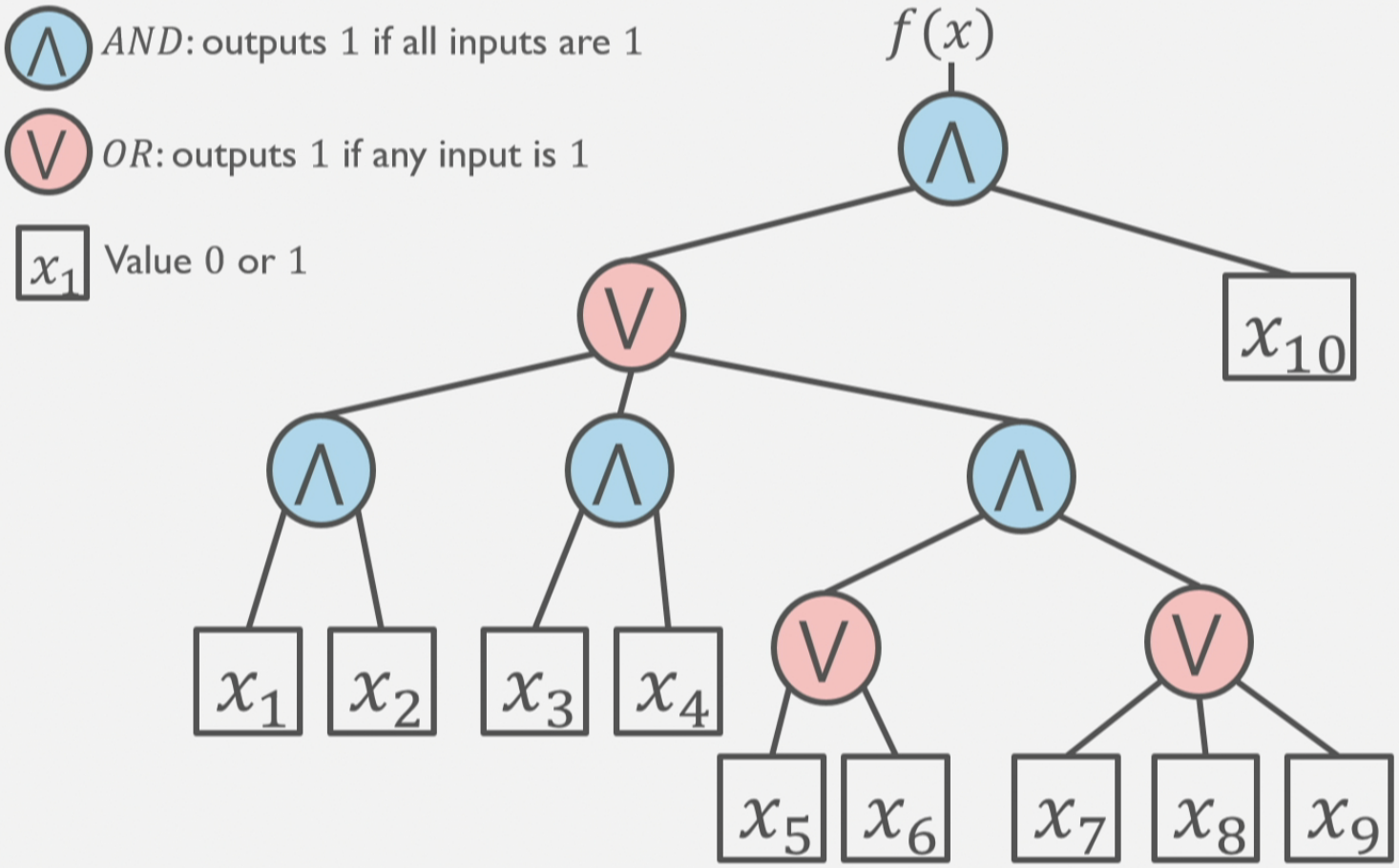
$$\max_{G:not\ connected} R(G') = 1$$

Example 2



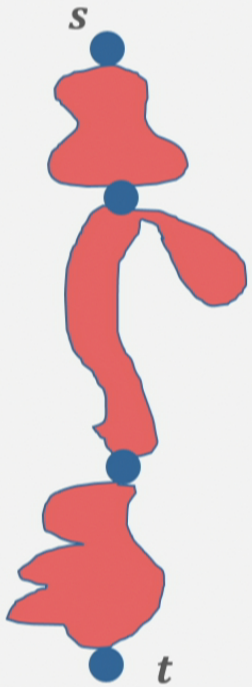
- Old analysis: $O(\sqrt{N})$ complexity
- New analysis: $O(\sqrt{N})$ complexity

Application to Boolean Formulas



Application to Boolean Formulas

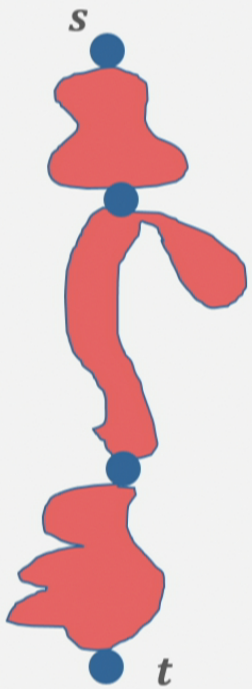
AND: outputs 1 if all inputs are 1



s and t are connected if all subgraphs are connected

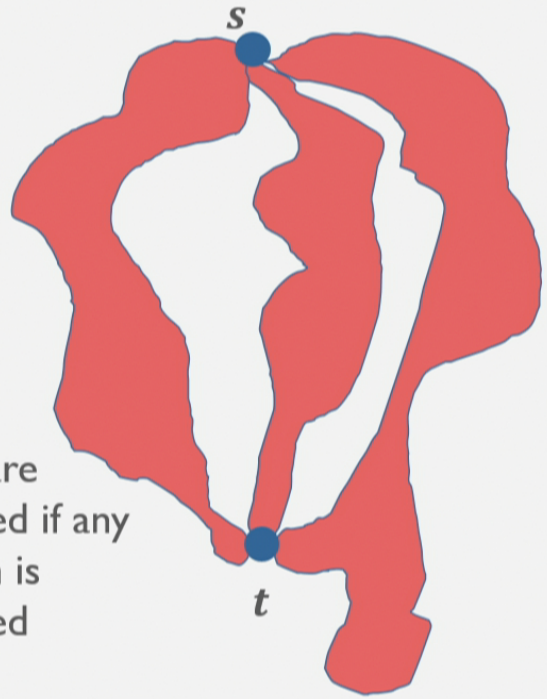
Application to Boolean Formulas

$\wedge$ AND: outputs 1 if all inputs are 1



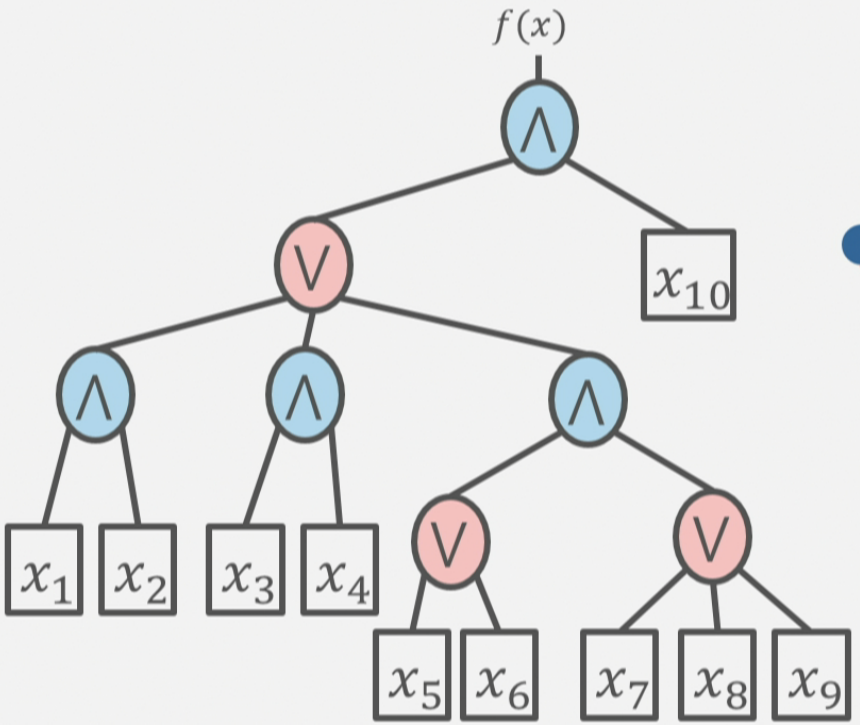
s and t are connected if all subgraphs are connected

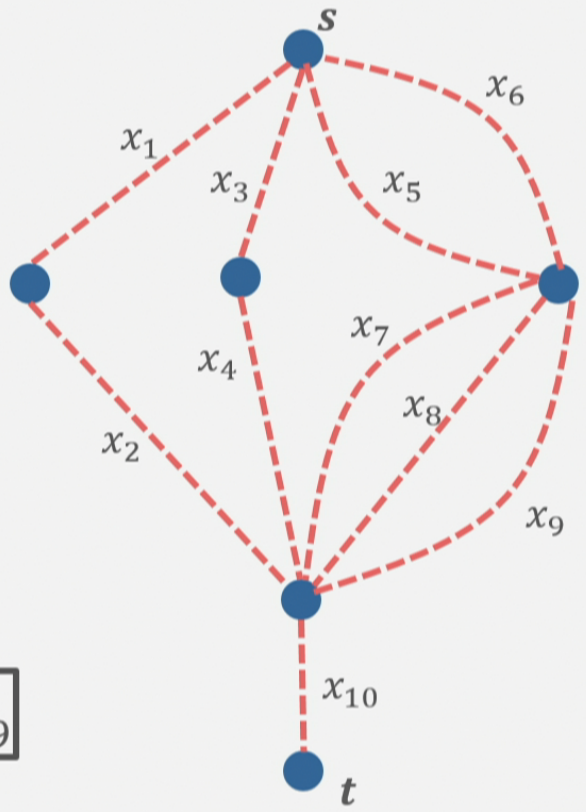
$\vee$ OR: outputs 1 if any input is 1



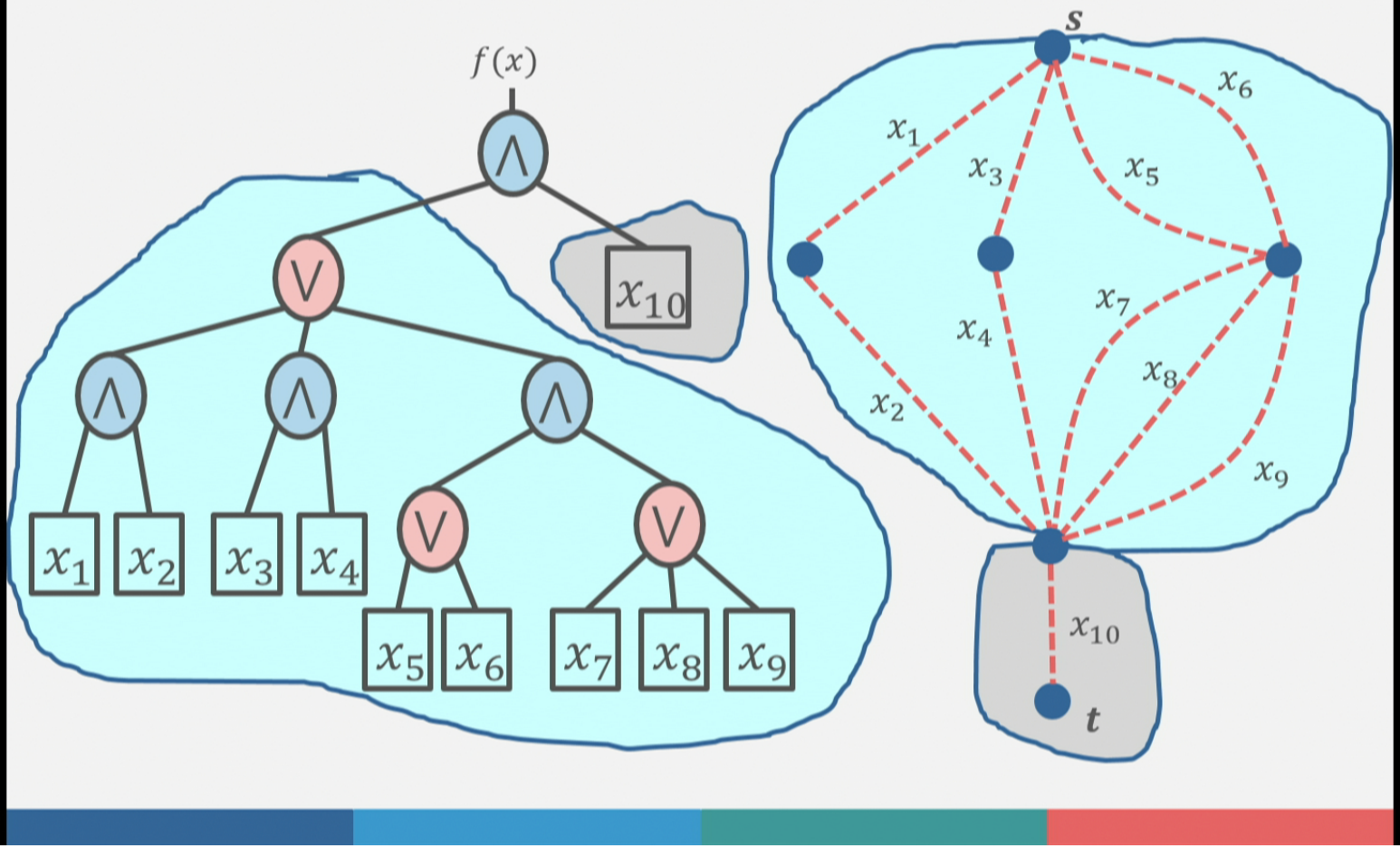
s and t are connected if any subgraph is connected

Application to Boolean Formulas

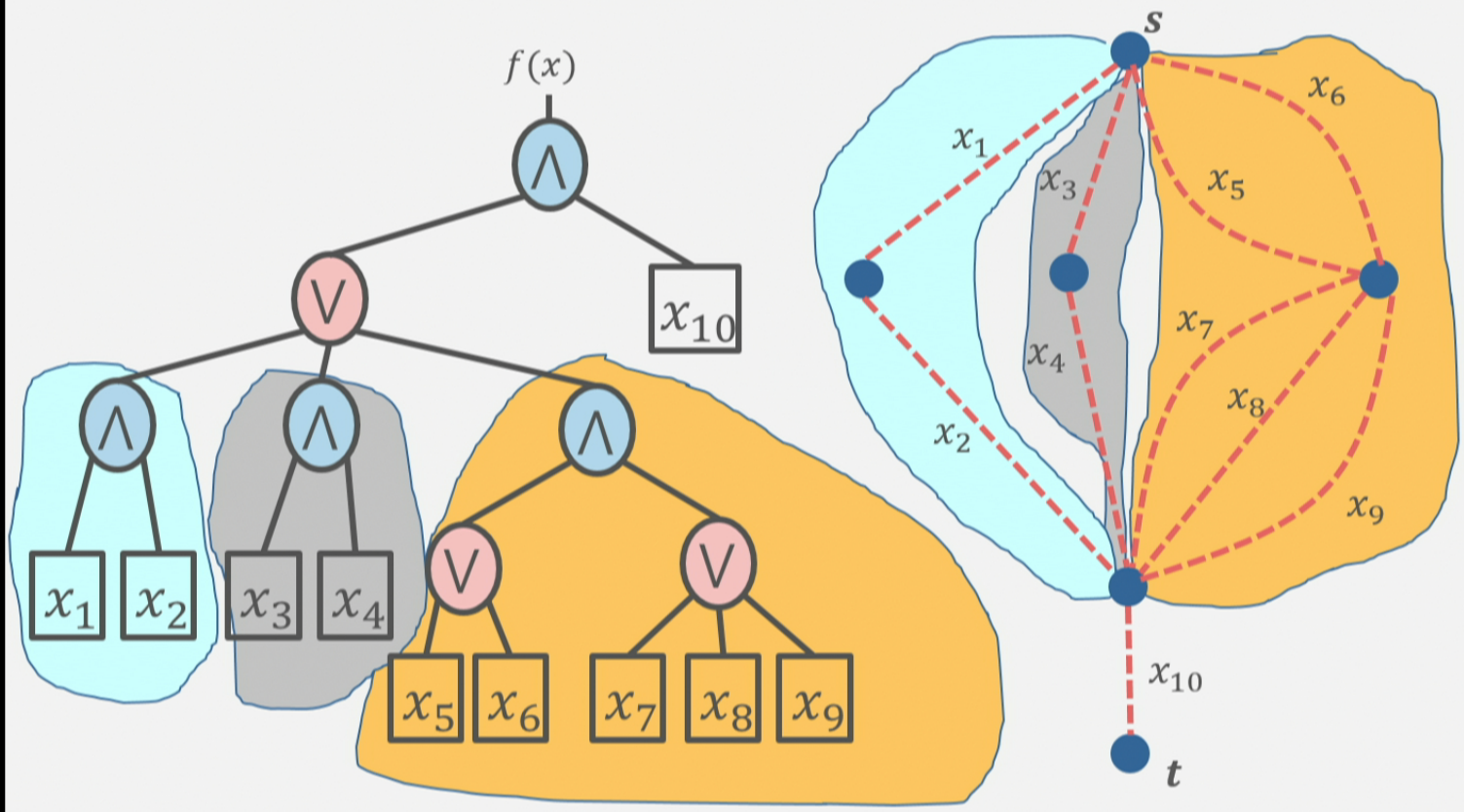




Application to Boolean Formulas



Application to Boolean Formulas



Application to Boolean Formulas

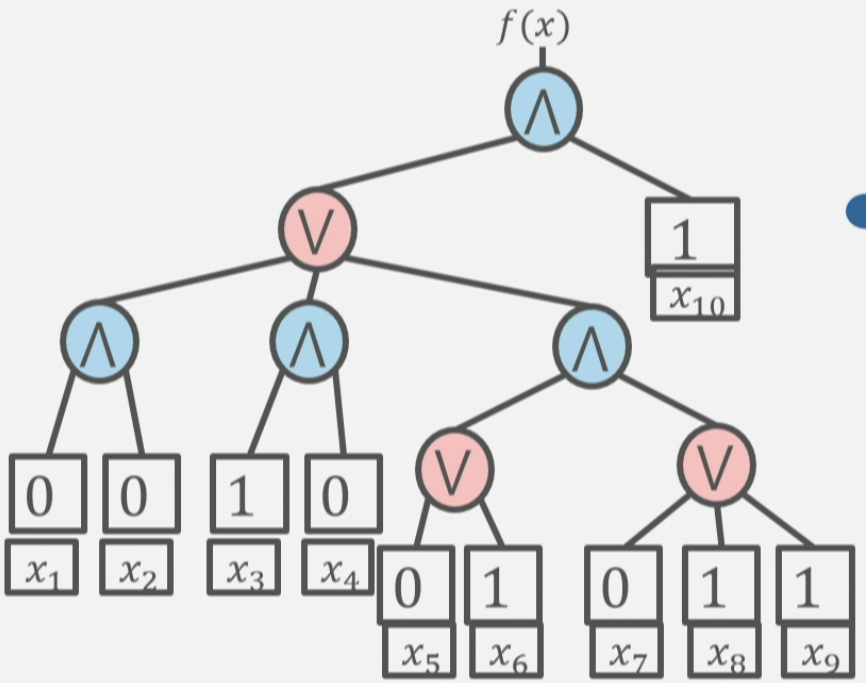
- The graph associated with a formula will always be planar, with s, t on external face.
- Can use our st-connectivity algorithm! Time required depends on the effective resistance of circuit corresponding to corresponding graph.

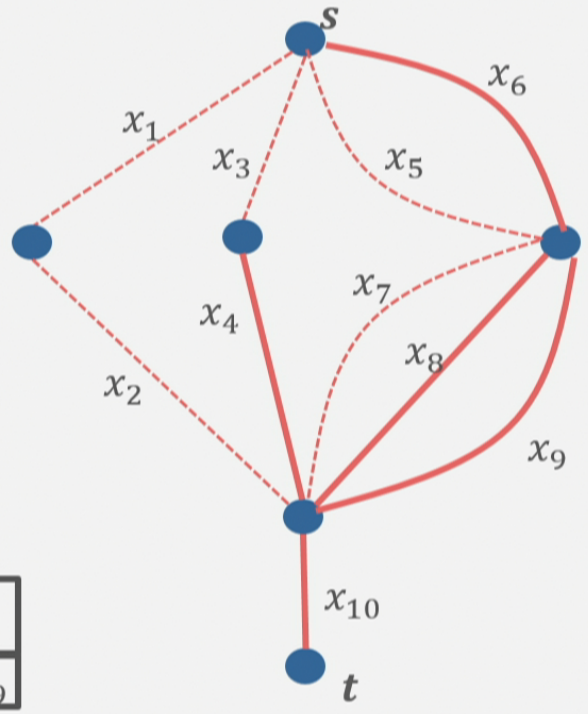
Application to Boolean Formulas

- The graph associated with a formula will always be planar, with s, t on external face.
- Can use our st-connectivity algorithm! Time required depends on the effective resistance of circuit corresponding to corresponding graph.

- This algorithm is pretty good! (for read-once formulas)
  - Gives a simple proof of $O(\sqrt{N})$ bound on N input formulas
  - Improves scope of superpolynomial quantum-classical separation of [Zhan et al '14]

Application to Boolean Formulas





- If we put edges where $x_i = 1$, s and t are connected iff $f(x) = 1$!

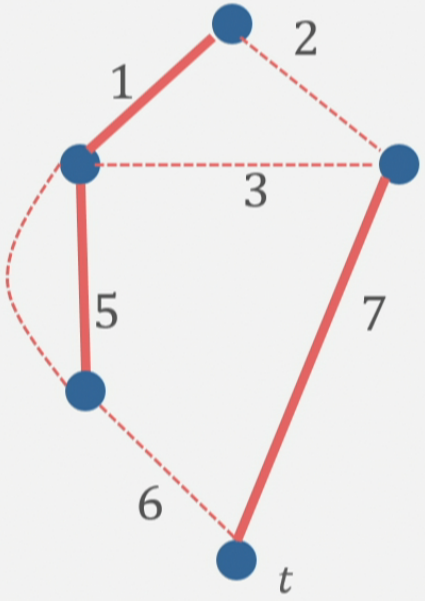
Application to Boolean Formulas

- The graph associated with a formula will always be planar, with s, t on external face.
- Can use our st-connectivity algorithm! Time required depends on the effective resistance of circuit corresponding to corresponding graph.

- This algorithm is pretty good! (for read-once formulas)
  - Gives a simple proof of $O(\sqrt{N})$ bound on N input formulas
  - Improves scope of superpolynomial quantum-classical separation of [Zhan et al '14]

Graph Notation

- $E(G)$: edges of graph G.
- $V(G)$: vertices of graph G.
- For an undirected edge e, we denote by $\vec{e}$ and $\overleftarrow{e}$ the edge with an direction



Quantum Query Algorithm

- Let $f: D \to \{0,1\}$, for $D \subseteq \{0,1\}^N$ be a known function.
- Suppose you are given access to a quantum oracle for the input:

- Want to evaluate $f(x_1, x_2, \ldots, x_N)$ using a quantum algorithm while querying box as few time as possible.

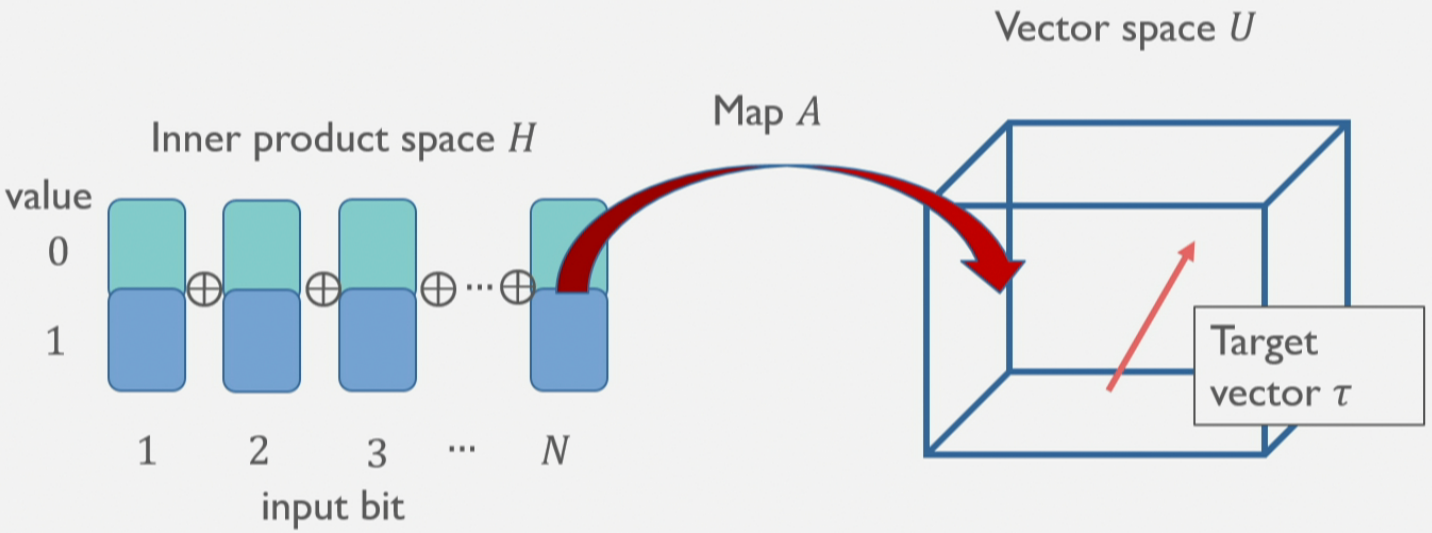
Span Program Quantum Algorithm

- A span program is a linear algebraic way of representing a (Boolean) function.
- There are an infinite number of span programs that represent a single function.

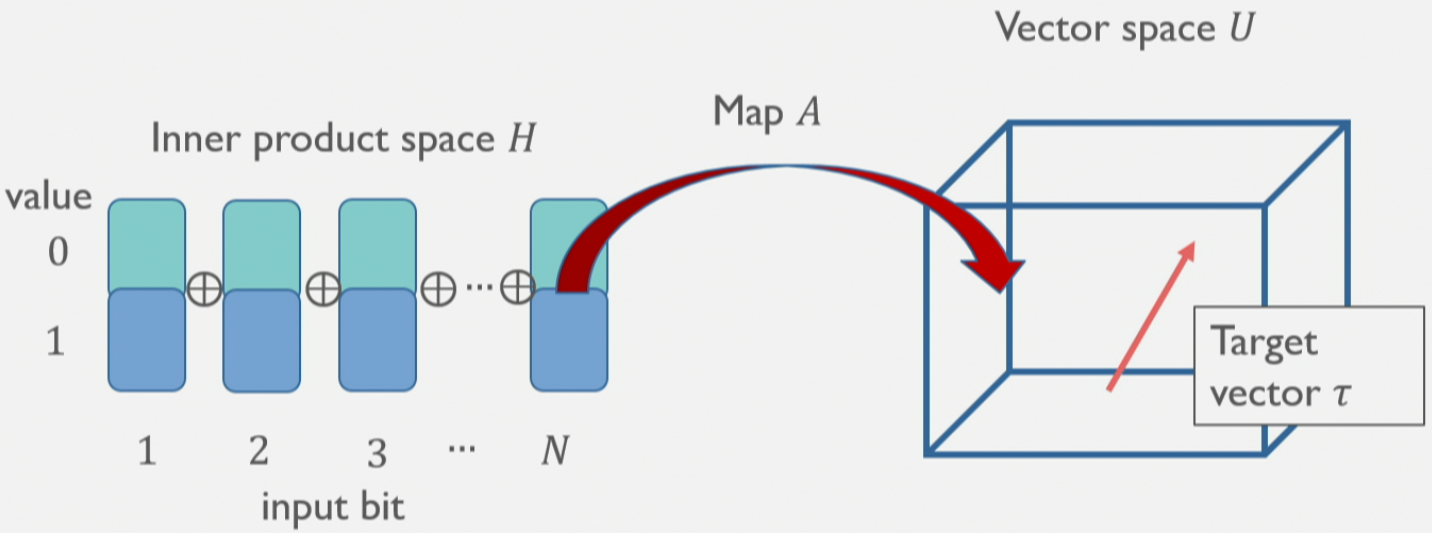
Span Program Quantum Algorithm

- A span program is a linear algebraic way of representing a (Boolean) function.
- There are an infinite number of span programs that represent a single function.
- Given a span program for a function, can create a quantum query algorithm to evaluate the function. The complexity of the algorithm depends on properties of the span program.

Span Programs

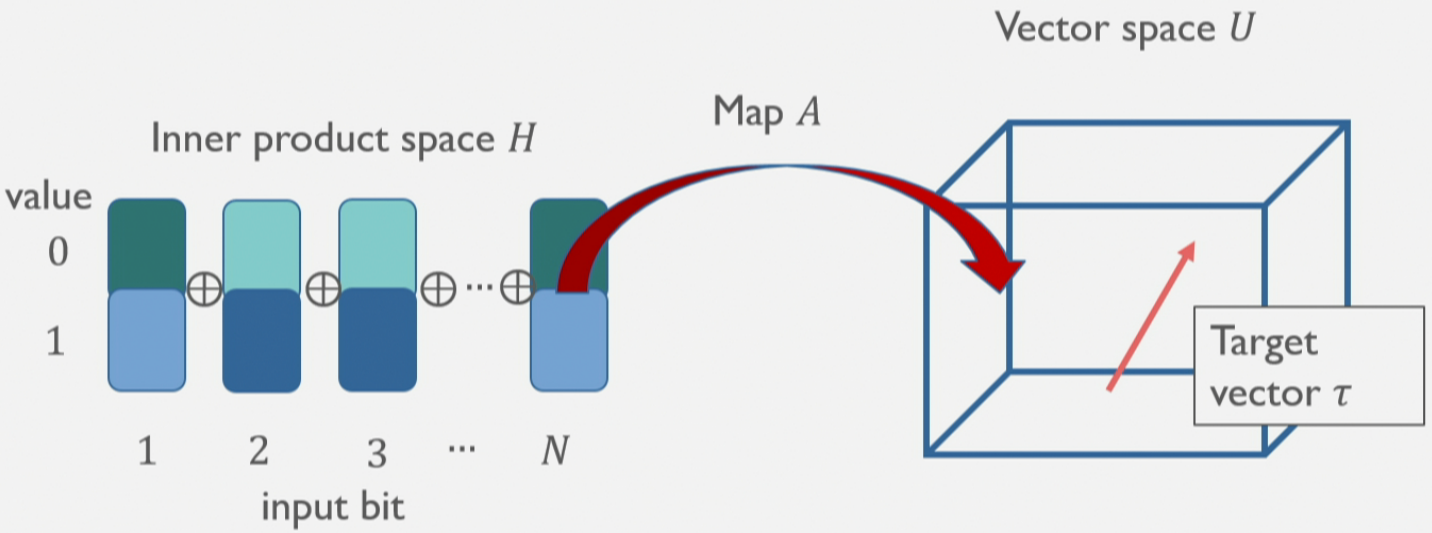


Span Programs



$x = 011 \ldots 0$

Span Programs



$x = 011 \ldots 0$

If there is a witness in the inner product space corresponding to x that gets mapped to τ, we say x is a 1-valued input.

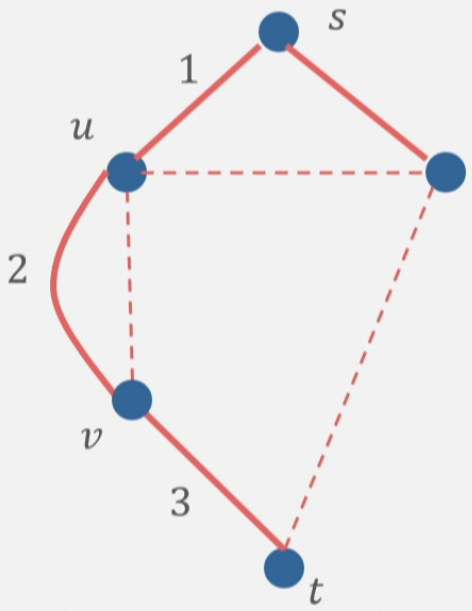
Span Programs

Span program for function on domain $D \subseteq \{0,1\}^N$	s-t connectivity example
Finite dimensional inner product space $H = H_1 \oplus H_2 \oplus \cdots \oplus H_N$ where $H_j = H_{j,0} + H_{j,1}$	
For $x \in \{0,1\}^N$, $H(x) = H_{1,x_1} \oplus H_{2,x_2} \oplus \cdots \oplus H_{N,x_N}$	
Vector space U	
Target vector $\tau \in U$	
Linear operator $A: H \to U$	
Then $x \in \{0,1\}^N$ is a "1-input" if $\exists \lvert w\rangle \in H(x): A\lvert w\rangle = \tau$. "0-input" otherwise	

Span Programs

Span program for function on domain $D \subseteq \{0,1\}^N$	s-t connectivity example
Finite dimensional inner product space $H = H_1 \oplus H_2 \oplus \cdots \oplus H_N$ where $H_j = H_{j,0} + H_{j,1}$	For edge $e \in E(G)$, $H_{e,0} = \{0\}, H_{e,1} = \text{span}\{\|\vec{e}\rangle, \|\overleftarrow{e}\rangle\}$
For $x \in \{0,1\}^N$, $H(x) = H_{1,x_1} \oplus H_{2,x_2} \oplus \cdots \oplus H_{N,x_N}$	$H(x) = \text{span}\{\|\vec{e}\rangle, \|\overleftarrow{e}\rangle : e \in E(G)\}$
Vector space U	$U = \text{span}\{\|u\rangle : u \in V(G)\}$
Target vector $\tau \in U$	$\tau = \|s\rangle - \|t\rangle$
Linear operator $A: H \to U$	$A = \sum_{e \in G : u,v \text{ are ordered endpoints of } \vec{e}} (\|u\rangle - \|v\rangle)\langle\vec{e}\|$
Then $x \in \{0,1\}^N$ is a "1-input" if $\exists \|w\rangle \in H(x): A\|w\rangle = \tau$. "0-input" otherwise	In this case, choose $\|w\rangle$ to be any directed path, or linear combination of directed paths.

Example



1. $|w\rangle = |1\rangle + |2\rangle + |3\rangle$
2. $A|w\rangle = (|s\rangle - |u\rangle) + (|u\rangle - |v\rangle) + (|v\rangle - |t\rangle)$
   $= |s\rangle - |t\rangle$

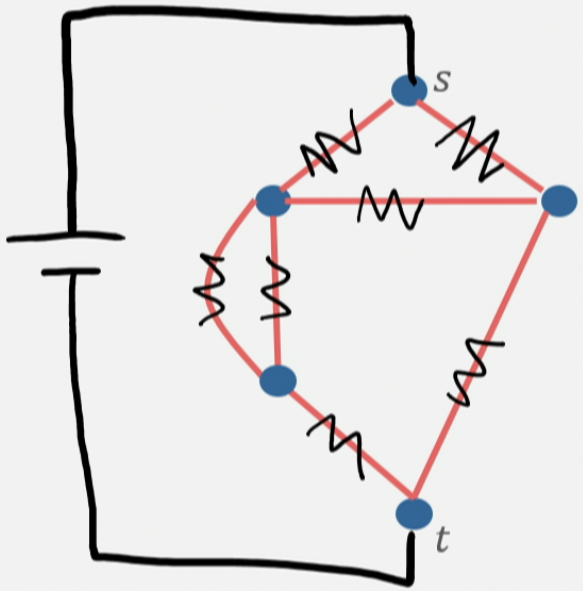
Query Complexity

$$Q(P_f) = \sqrt{\max_{x \in f^{-1}(1)} w_+(x,P)} \sqrt{\max_{x \in f^{-1}(0)} w_-(x,P)}$$

- $w_+(x,P) = \min\{\||w\rangle\|^2 : |w\rangle \in H(x), A|w\rangle = \tau\}$
- $w_-(x,P) = \min\{\|\omega A\|^2 : \omega \text{ a linear operator } \omega: U \to \mathbb{R}, \omega A \Pi_{H(x)} = 0, \omega\tau = 1\}$

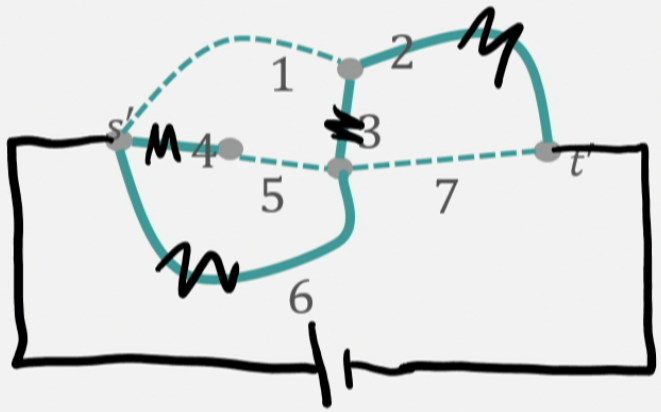
Query Complexity

- $w_+(x, P) = \min\{\||w\rangle\|^2 : |w\rangle \in H(x), A|w\rangle = \tau\}$
- Valid witnesses are linear combinations of st-paths.
- Minimizing witness weights st-paths with exactly the same weighting as current flowing through a circuit.



Query Complexity

- $w_{-}(x, P) = \min\{\|\omega A\|^2 : \omega \text{ a linear operator } \omega: U \to \mathbb{R}, \omega A \Pi_{H(x)} = 0, \omega\tau = 1\}$
- Valid witnesses are linear combinations of st-cuts, with ω taking value 1 above cut and 0 below cut.
- st-cut in G corresponds to st-path in G', so witness is a linear combination of st-paths in G'.
- Minimizing witness weights these paths with exactly the same weighting as current flowing through a circuit.



Query Complexity

$$Q(P_f) = \sqrt{\max_{x \in f^{-1}(1)} w_+(x, P)} \sqrt{\max_{x \in f^{-1}(0)} w_-(x, P)}$$

$$Q(P_f) = \sqrt{\max_{G:connected} R(G)} \sqrt{\max_{G:not\ connected} R(G')}$$

Open Questions

- When is our algorithm optimal for Boolean formulas?
- Can we extend these ideas to non-planar graphs?
- Are there other problems that reduce to st-connectivity?
- What is the classical time/query complexity of st-connectivity? Can we relate it to effective resistance?

Partial Results:
arXiv:1511.02235