Quantum algorithms through graph composition

arXiv:2504.02115

Arjan Cornelissen[1]

[1]Simons Institute, University of California, Berkeley, California

April 20th, 2026

SIMONS INSTITUTE for the Theory of Computing

Quantum algorithmic frameworks

Quantum algorithmic frameworks

Setting:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

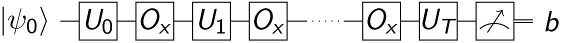
Quantum algorithmic frameworks

Setting:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Goal: Design algorithm $\mathcal{A}$:

1. Circuit: $U_T O_x U_{T-1} O_x \cdots O_x U_0$.



Quantum algorithmic frameworks

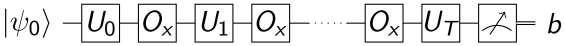
Setting:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Goal: Design algorithm $\mathcal{A}$:

1. Circuit: $U_T O_x U_{T-1} O_x \cdots O_x U_0$.
2. $\forall x \in \mathcal{D}, \mathbb{P}[\mathcal{A}(O_x) = f(x)] \geq \frac{2}{3}$.
3. With minimal no. queries T.

$\mathbb{P}[b = f(x)] \geq \frac{2}{3}$



Quantum algorithmic frameworks

Setting:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

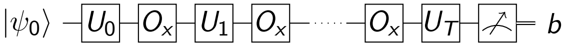
Goal: Design algorithm $\mathcal{A}$:

1. Circuit: $U_T O_x U_{T-1} O_x \cdots O_x U_0$.
2. $\forall x \in \mathcal{D}, \mathbb{P}[\mathcal{A}(O_x) = f(x)] \geq \frac{2}{3}$.
3. With minimal no. queries T.

Framework:

1. Define a mathematical object.
2. Convert object into quantum algorithm.

$\mathbb{P}[b = f(x)] \geq \frac{2}{3}$



Quantum algorithmic frameworks

Setting:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Goal: Design algorithm $\mathcal{A}$:

1. Circuit: $U_T O_x U_{T-1} O_x \cdots O_x U_0$.
2. $\forall x \in \mathcal{D}, \mathbb{P}[\mathcal{A}(O_x) = f(x)] \geq \frac{2}{3}$.
3. With minimal no. queries T.

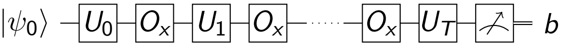
Framework:

1. Define a mathematical object.
2. Convert object into quantum algorithm.

Example: Boolean formula evaluation

$$f(x) = \underbrace{x_1 \wedge (x_2 \vee x_3) \vee (x_2 \wedge x_5) \vee x_3}_{\text{length } \ell}$$

$$\mathbb{P}[b = f(x)] \geq \frac{2}{3}$$



Quantum algorithmic frameworks

Setting:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Goal: Design algorithm $\mathcal{A}$:

1. Circuit: $U_T O_x U_{T-1} O_x \cdots O_x U_0$.
2. $\forall x \in \mathcal{D}, \mathbb{P}[\mathcal{A}(O_x) = f(x)] \geq \frac{2}{3}$.
3. With minimal no. queries T.

Framework:

1. Define a mathematical object.
2. Convert object into quantum algorithm.

Example: Boolean formula evaluation

$$f(x) = \underbrace{x_1 \wedge (x_2 \vee x_3) \vee (x_2 \wedge x_5) \vee x_3}_{\text{length } \ell}$$

[Rei11]

$$\mathbb{P}[b = f(x)] \geq \frac{2}{3}$$

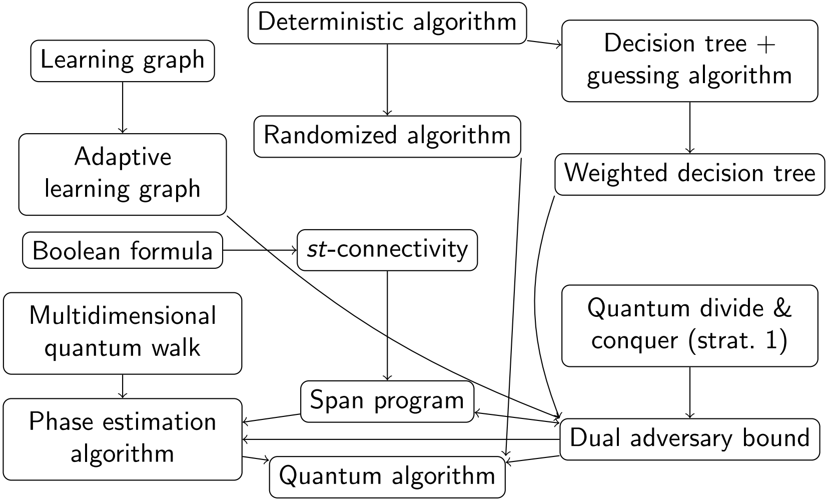
$$|\psi_0\rangle \; \underbrace{U_0 \; O_x \; U_1 \; O_x \cdots O_x \; U_T \; \text{measure}}_{T = O(\sqrt{\ell})} = b$$

Quantum algorithmic framework landscape

Quantum algorithmic framework landscape

Overview:

→ Conversion between framework objects

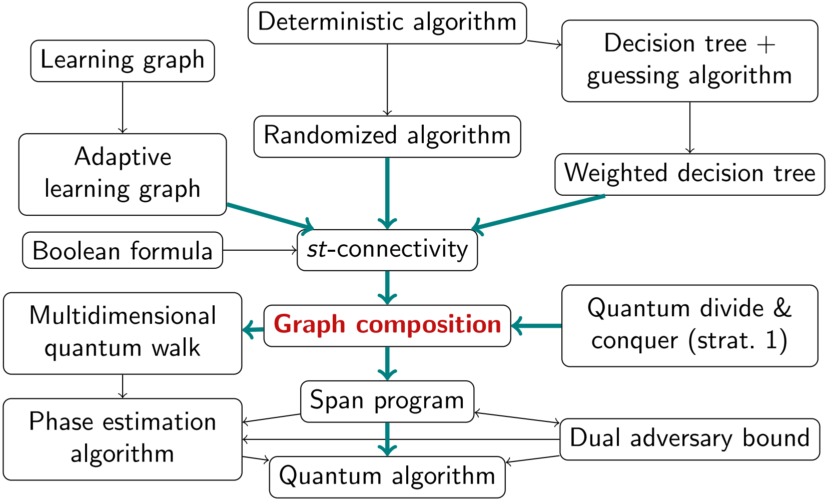


Quantum algorithmic framework landscape

Overview:

→ Conversion between framework objects

→ New relations [This work]



Quantum algorithmic framework landscape

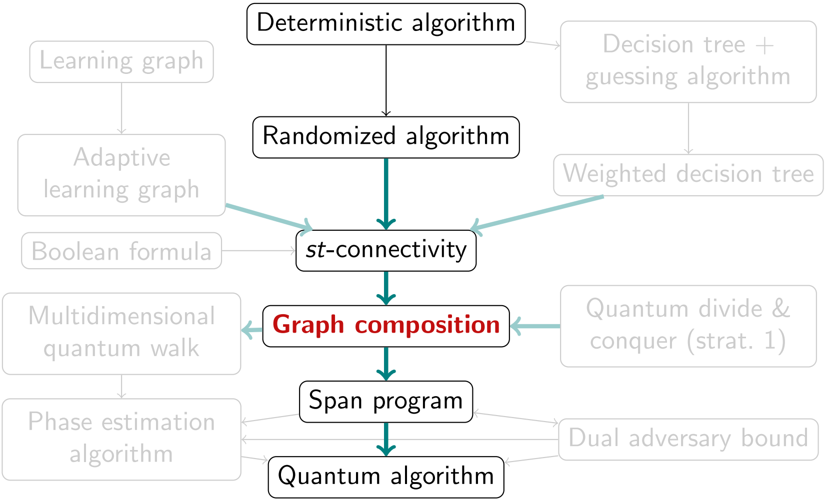
Overview:

→ Conversion between framework objects

→ New relations [This work]

This talk:

1. Span programs
2. Graph composition
3. *st*-connectivity examples
4. Randomized → *st*-connectivity

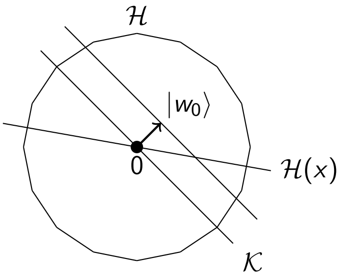


Span programs [RŠ12,Rei09,Rei11]

Span programs [RŠ12,Rei09,Rei11]

Four ingredients:

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

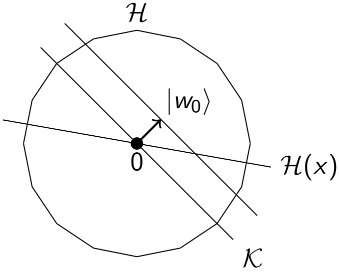


Four ingredients:

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^{\perp}$.

Computes function: $f : \mathcal{D} \to \{0, 1\}$

1. $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.



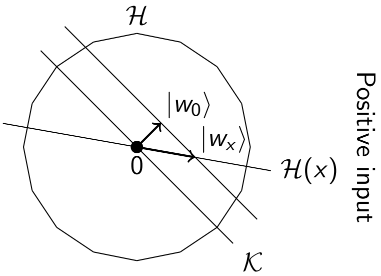
Span programs [RŠ12,Rei09,Rei11]

Four ingredients:

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

Computes function: $f : \mathcal{D} \to \{0, 1\}$

1. $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.
2. *Positive inputs:* $w_+(x, \mathcal{P}) = \||w_x\rangle\|^2$.



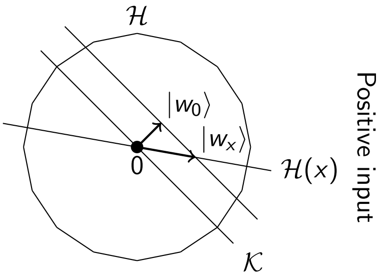
Span programs [RŠ12,Rei09,Rei11]

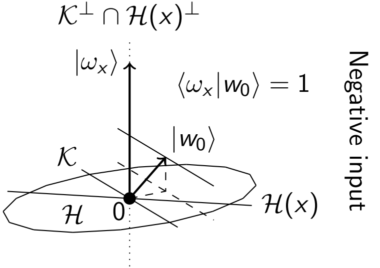
Four ingredients:

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

Computes function: $f : \mathcal{D} \to \{0,1\}$

1. $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.
2. *Positive inputs:* $w_+(x,\mathcal{P}) = \||w_x\rangle\|^2$.
3. *Negative inputs:* $w_-(x,\mathcal{P}) = \||\omega_x\rangle\|^2$.





Span programs [RŠ12,Rei09,Rei11]

Four ingredients:

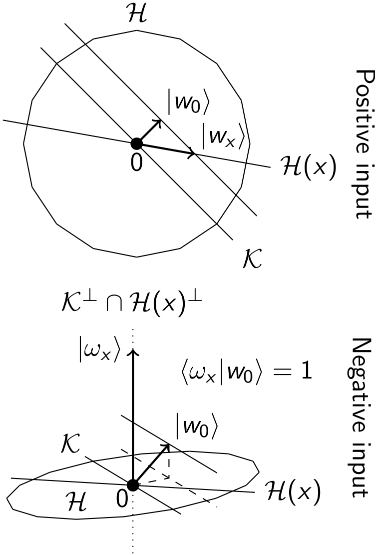
1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

Computes function: $f : \mathcal{D} \to \{0,1\}$

1. $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.
2. *Positive inputs:* $w_+(x,\mathcal{P}) = \||w_x\rangle\|^2$.
3. *Negative inputs:* $w_-(x,\mathcal{P}) = \||\omega_x\rangle\|^2$.

Thm: Algorithm with $O(C(\mathcal{P}))$ queries to $2\Pi_{\mathcal{H}(x)} - I$.

$$C(\mathcal{P}) = \sqrt{\max_{x \in f^{-1}(0)} w_-(x,\mathcal{P}) \cdot \max_{x \in f^{-1}(1)} w_+(x,\mathcal{P})}.$$



Four ingredients:

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

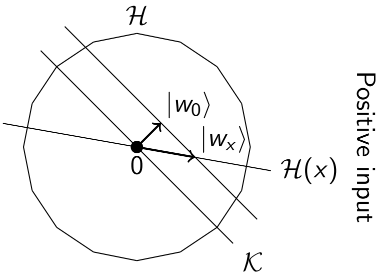
Computes function: $f : \mathcal{D} \to \{0,1\}$

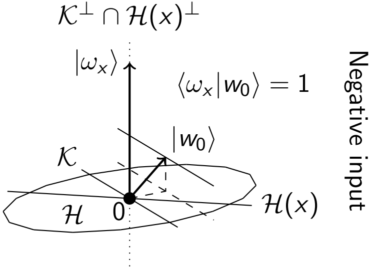
1. $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.
2. *Positive inputs:* $w_+(x,\mathcal{P}) = \||w_x\rangle\|^2$.
3. *Negative inputs:* $w_-(x,\mathcal{P}) = \||\omega_x\rangle\|^2$.

Thm: Algorithm with $O(C(\mathcal{P}))$ queries to $2\Pi_{\mathcal{H}(x)} - I$.

$$C(\mathcal{P}) = \sqrt{\max_{x \in f^{-1}(0)} w_-(x,\mathcal{P}) \cdot \max_{x \in f^{-1}(1)} w_+(x,\mathcal{P})}.$$

Thm: "Complete" for computing boolean functions.



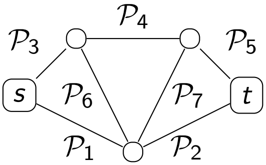


Graph composition [This work]

Ingredients:

1. Undirected graph $G = (V, E)$.
2. Source and target vertices $s, t \in V$.
3. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph composition:



Graph composition [This work]

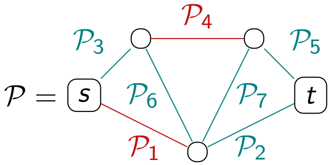
Ingredients:

1. Undirected graph $G = (V, E)$.
2. Source and target vertices $s, t \in V$.
3. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph composition: Compiles a span program $\mathcal{P}$:

1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.

Graph composition:



Graph composition [This work]

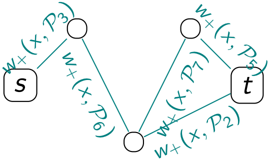
Ingredients:

1. Undirected graph $G = (V, E)$.
2. Source and target vertices $s, t \in V$.
3. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph composition: Compiles a span program $\mathcal{P}$:

1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.
2. $w_+(x, \mathcal{P}) = R^{eff}_{s,t,r_+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.

Positive input:

$\mathcal{P} =$ 

$$w_+(x, \mathcal{P}) = R^{eff}_{s,t,r_+}$$

Graph composition [This work]

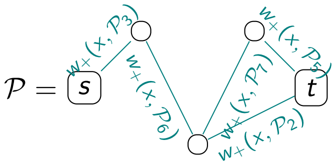
Ingredients:

1. Undirected graph $G = (V, E)$.
2. Source and target vertices $s, t \in V$.
3. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph composition: Compiles a span program $\mathcal{P}$:

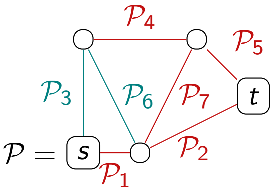
1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.
2. $w_+(x, \mathcal{P}) = R^{\mathrm{eff}}_{s,t,r_+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.

Positive input:



$$w_+(x, \mathcal{P}) = R^{\mathrm{eff}}_{s,t,r_+}$$

Negative input:



Graph composition [This work]

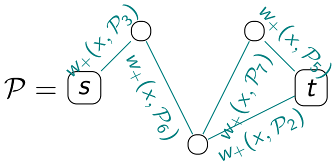
Ingredients:

1. Undirected graph $G = (V, E)$.
2. Source and target vertices $s, t \in V$.
3. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph composition: Compiles a span program $\mathcal{P}$:

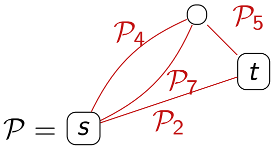
1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.
2. $w_+(x, \mathcal{P}) = R^{\mathrm{eff}}_{s,t,r_+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.

Positive input:



$$w_+(x, \mathcal{P}) = R^{\mathrm{eff}}_{s,t,r_+}$$

Negative input:



Graph composition [This work]

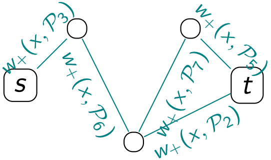
Ingredients:

1. Undirected graph $G = (V, E)$.
2. Source and target vertices $s, t \in V$.
3. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph composition: Compiles a span program $\mathcal{P}$:

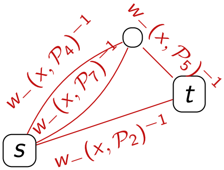
1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.
2. $w_+(x, \mathcal{P}) = R^{eff}_{s,t,r_+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
3. $w_-(x, \mathcal{P}) = (R^{eff}_{s,t,r_-})^{-1}$ with $r_-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive input:

$\mathcal{P} =$ 

$$w_+(x, \mathcal{P}) = R^{eff}_{s,t,r_+}$$

Negative input:

$\mathcal{P} =$ 

$$w_-(x, \mathcal{P}) = (R^{eff}_{s,t,r_-})^{-1}$$

Graph composition [This work]

Ingredients:

1. Undirected graph $G = (V, E)$.
2. Source and target vertices $s, t \in V$.
3. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

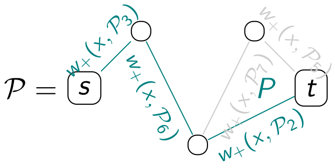
Graph composition: Compiles a span program $\mathcal{P}$:

1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.
2. $w_+(x, \mathcal{P}) = R^{eff}_{s,t,r_+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
3. $w_-(x, \mathcal{P}) = (R^{eff}_{s,t,r_-})^{-1}$ with $r_-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Path-cut theorem:

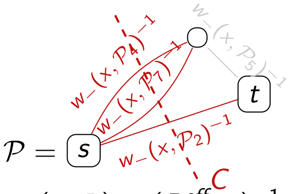
1. For P an st-path: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. For C an st-cut: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



$$w_+(x, \mathcal{P}) = R^{eff}_{s,t,r_+}$$

Negative input:



$$w_-(x, \mathcal{P}) = (R^{eff}_{s,t,r_-})^{-1}$$

st-connectivity [BR12,JK17,JJKP18]

st-connectivity [BR12,JK17,JJKP18]

Special case of graph composition:

1. *Trivial span programs:* single bit queries.

st-connectivity [BR12,JK17,JJKP18]

Special case of graph composition:

1. *Trivial span programs:* single bit queries.

For $\alpha > 0$, $j \in [n]$:

αx_j

$w_+(\cdots 1 \cdots) = \alpha$

$w_-(\cdots 0 \cdots) = \frac{1}{\alpha}$

$\alpha(\neg x_j)$

$w_+(\cdots 0 \cdots) = \alpha$

$w_-(\cdots 1 \cdots) = \frac{1}{\alpha}$

st-connectivity [BR12,JK17,JJKP18]

Special case of graph composition:

1. *Trivial span programs:* single bit queries.

Example: The OR-function

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$

$$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$

For $\alpha > 0$, $j \in [n]$:

αx_j

$w_+(\cdots 1 \cdots) = \alpha$

$w_-(\cdots 0 \cdots) = \frac{1}{\alpha}$

$\alpha(\neg x_j)$

$w_+(\cdots 0 \cdots) = \alpha$

$w_-(\cdots 1 \cdots) = \frac{1}{\alpha}$

st-connectivity [BR12,JK17,JJKP18]

Special case of graph composition:

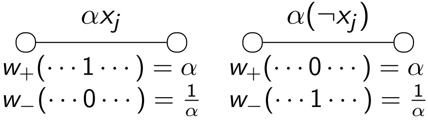
1. *Trivial span programs:* single bit queries.

Example: The OR-function

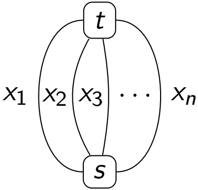
1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$

$$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$

For $\alpha > 0$, $j \in [n]$:



st-graph for OR_n*:*



Special case of graph composition:

1. *Trivial span programs:* single bit queries.

Example: The OR-function

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$
2. $w_+(x) \leq 1$.

For $\alpha > 0$, $j \in [n]$:

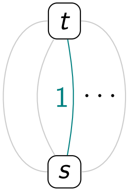
αx_j

$w_+(\cdots 1 \cdots) = \alpha$
$w_-(\cdots 0 \cdots) = \frac{1}{\alpha}$

$\alpha(\neg x_j)$

$w_+(\cdots 0 \cdots) = \alpha$
$w_-(\cdots 1 \cdots) = \frac{1}{\alpha}$

st-graph for OR_n*:*



$x = 0010\cdots 0 \Rightarrow w_+(x) \leq 1$

st-connectivity [BR12,JK17,JJKP18]

Special case of graph composition:

1. *Trivial span programs:* single bit queries.

Example: The OR-function

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$
$$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$
2. $w_+(x) \leq 1$.
3. $w_-(x) \leq n$.

For $\alpha > 0$, $j \in [n]$:

αx_j

$w_+(\cdots 1 \cdots) = \alpha$

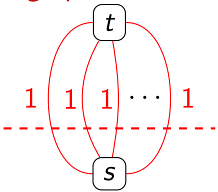
$w_-(\cdots 0 \cdots) = \frac{1}{\alpha}$

$\alpha(\neg x_j)$

$w_+(\cdots 0 \cdots) = \alpha$

$w_-(\cdots 1 \cdots) = \frac{1}{\alpha}$

st-graph for OR_n*:*



$x = 0000 \cdots 0 \Rightarrow w_-(x) = n$

st-connectivity [BR12,JK17,JJKP18]

Special case of graph composition:

1. *Trivial span programs:* single bit queries.

Example: The OR-function

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$
2. $w_+(x) \leq 1$.
3. $w_-(x) \leq n$.
4. $\Rightarrow C(\mathcal{P}) \leq \sqrt{n}$.
5. $\Rightarrow Q(\mathrm{OR}_n) \in O(\sqrt{n})$.

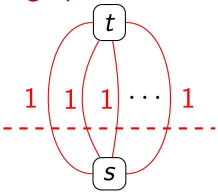
For $\alpha > 0$, $j \in [n]$:

αx_j | $\alpha(\neg x_j)$

$w_+(\cdots 1 \cdots) = \alpha$ | $w_+(\cdots 0 \cdots) = \alpha$

$w_-(\cdots 0 \cdots) = \frac{1}{\alpha}$ | $w_-(\cdots 1 \cdots) = \frac{1}{\alpha}$

st-graph for OR_n*:*



$x = 0000\cdots 0 \Rightarrow w_-(x) = n$

st-connectivity [BR12,JK17,JJKP18]

Special case of graph composition:

1. *Trivial span programs:* single bit queries.

Example: The OR-function

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$

   $$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$
2. $w_+(x) \leq 1$.
3. $w_-(x) \leq n$.
4. $\Rightarrow C(\mathcal{P}) \leq \sqrt{n}$.
5. $\Rightarrow \mathrm{Q}(\mathrm{OR}_n) \in O(\sqrt{n})$.

Quadratic speed-up for search.

For $\alpha > 0$, $j \in [n]$:

αx_j

$w_+(\cdots 1 \cdots) = \alpha$

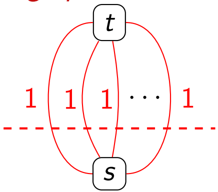
$w_-(\cdots 0 \cdots) = \frac{1}{\alpha}$

$\alpha(\neg x_j)$

$w_+(\cdots 0 \cdots) = \alpha$

$w_-(\cdots 1 \cdots) = \frac{1}{\alpha}$

st-graph for OR_n*:*



$x = 0000 \cdots 0 \Rightarrow w_-(x) = n$

Example: Threshold function

Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
$$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$

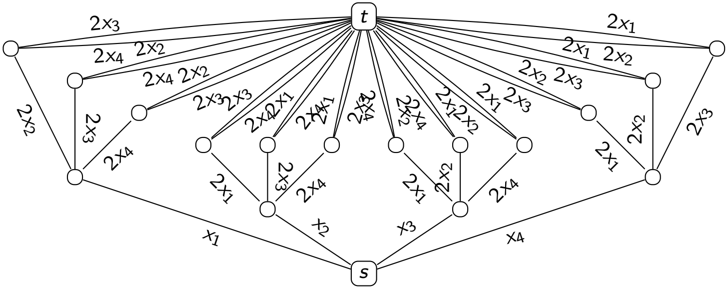
Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$

$$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$

st-graph for Th_4^3:

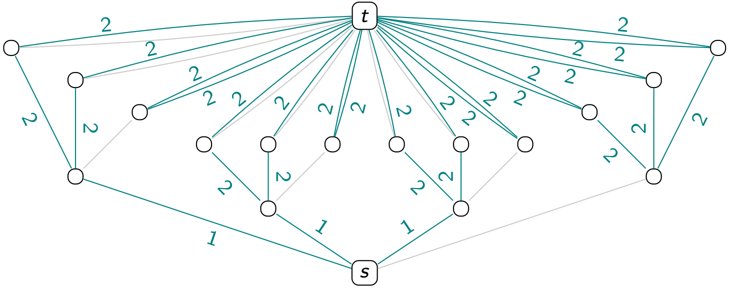


Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$

st-graph for Th_4^3*:*



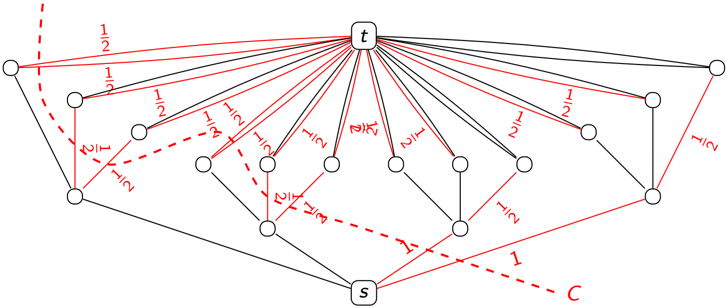
$x = 1110 \Rightarrow w_+(x) = 1$

Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
$$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$
3. $w_-(x) = \frac{k(n-k+1)}{k-|x|}$

st-graph for Th_4^3*:*



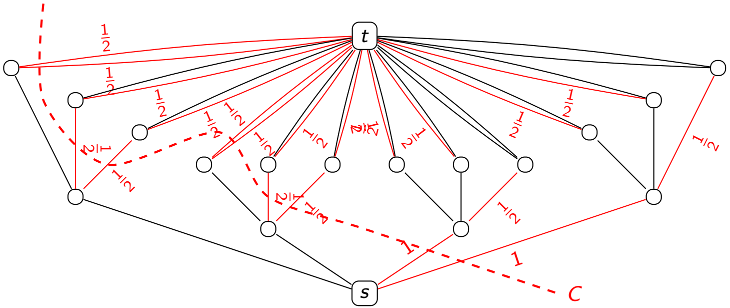
$x = 1100 \Rightarrow w_-(x) = 6$

Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$
3. $w_-(x) = \frac{k(n-k+1)}{k-|x|}$
4. $\Rightarrow C(\mathcal{P}) = \sqrt{k(n-k+1)}$.
5. $\Rightarrow Q(\mathrm{Th}_n^k) \in O(\sqrt{k(n-k+1)})$.

st-graph for Th_4^3*:*



$x = 1100 \Rightarrow w_-(x) = 6$

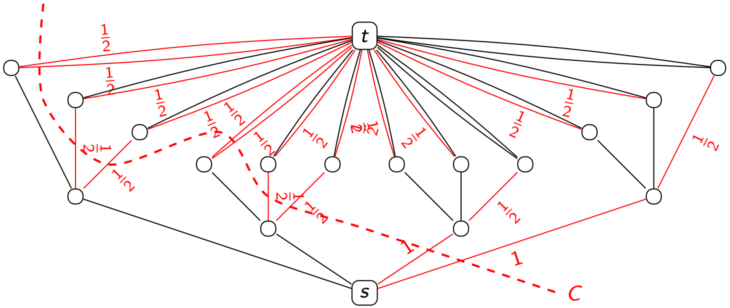
Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$
3. $w_-(x) = \frac{k(n-k+1)}{k-|x|}$
4. $\Rightarrow C(\mathcal{P}) = \sqrt{k(n-k+1)}.$
5. $\Rightarrow \mathrm{Q}(\mathrm{Th}_n^k) \in O(\sqrt{k(n-k+1)}).$

Known to be optimal!

st-graph for Th_4^3*:*



$x = 1100 \Rightarrow w_-(x) = 6$

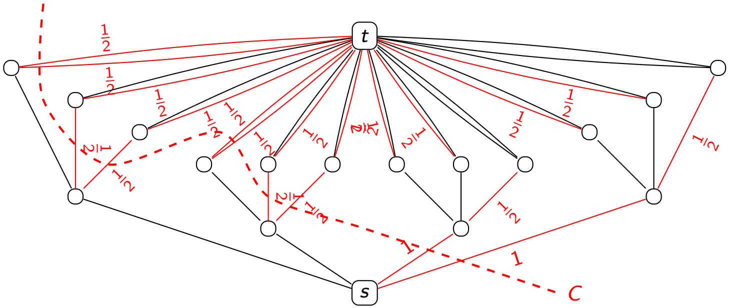
The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
$$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$
3. $w_-(x) = \frac{k(n-k+1)}{k-|x|}$
4. $\Rightarrow C(\mathcal{P}) = \sqrt{k(n-k+1)}$.
5. $\Rightarrow \mathrm{Q}(\mathrm{Th}_n^k) \in O(\sqrt{k(n-k+1)})$.

Known to be optimal!

Remark: With $k = n/2$, it also solves gapped majority in $O(1)$ queries.

st-graph for Th_4^3*:*



$x = 1100 \Rightarrow w_-(x) = 6$

Classical algorithms → st-connectivity

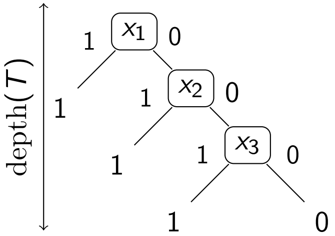
Classical algorithms → *st*-connectivity

Deterministic → st-connectivity:

Classical algorithms $\to$ *st*-connectivity

Deterministic $\to$ st-connectivity:

1. Decision tree T.

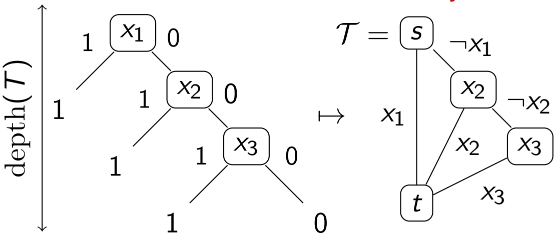


Classical algorithms $\rightarrow$ *st*-connectivity

Deterministic $\rightarrow$ st-connectivity:

1. Decision tree T.
2. Conversion into *st*-connectivity.

Decision tree $\rightarrow$ st-connectivity:

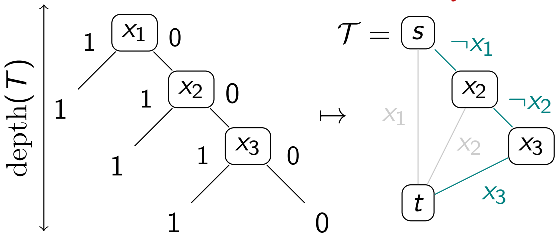


Classical algorithms $\to$ *st*-connectivity

Deterministic $\to$ st-connectivity:

1. Decision tree T.
2. Conversion into *st*-connectivity.
3. $w_+(x, \mathcal{T}) \leq \mathrm{depth}(T)$.

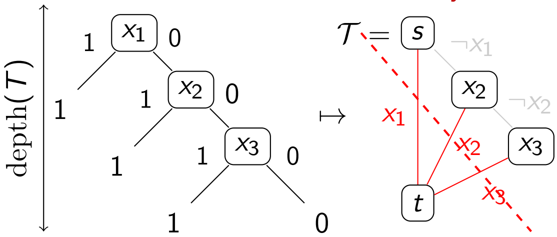
Decision tree $\to$ st-connectivity:



Deterministic $\to$ st-connectivity:

1. Decision tree T.
2. Conversion into *st*-connectivity.
3. $w_+(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
4. $w_-(x, \mathcal{T}) \leq \mathrm{depth}(T)$.

Decision tree $\to$ st-connectivity:

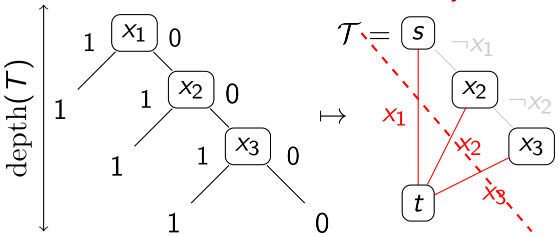


Classical algorithms $\to$ *st*-connectivity

Deterministic $\to$ st-connectivity:

1. Decision tree T.
2. Conversion into *st*-connectivity.
3. $w_+(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
4. $w_-(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
5. $C(\mathcal{T}) \leq \mathrm{depth}(T)$.

Decision tree $\to$ st-connectivity:



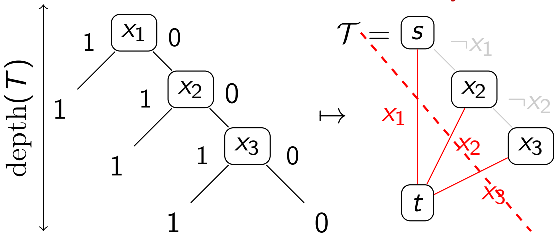
Classical algorithms $\to$ *st*-connectivity

Deterministic $\to$ st-connectivity:

1. Decision tree T.
2. Conversion into *st*-connectivity.
3. $w_+(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
4. $w_-(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
5. $C(\mathcal{T}) \leq \mathrm{depth}(T)$.

Randomized $\to$ st-connectivity: (sketch)

Decision tree $\to$ st-connectivity:



Classical algorithms $\rightarrow$ *st*-connectivity

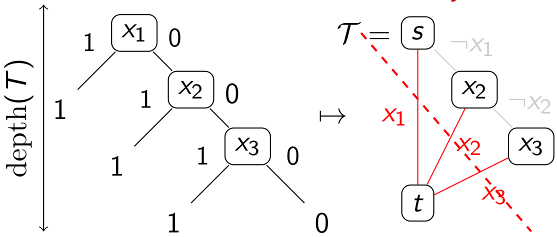
Deterministic $\rightarrow$ st-connectivity:

1. Decision tree T.
2. Conversion into *st*-connectivity.
3. $w_+(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
4. $w_-(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
5. $C(\mathcal{T}) \leq \mathrm{depth}(T)$.

Randomized $\rightarrow$ st-connectivity: (sketch)

1. Decision trees $\{T_j\}_{j=1}^N$
   Probability distribution $\{p_j\}_{j=1}^N$.

Decision tree $\rightarrow$ st-connectivity:

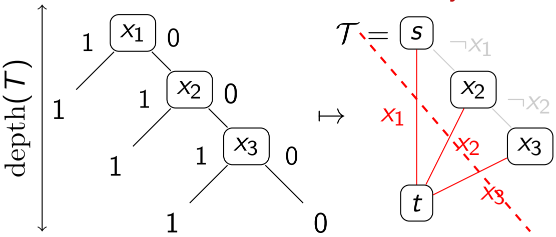


Classical algorithms → *st*-connectivity

Deterministic → st-connectivity:

1. Decision tree T.
2. Conversion into *st*-connectivity.
3. $w_+(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
4. $w_-(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
5. $C(\mathcal{T}) \leq \mathrm{depth}(T)$.

Decision tree → st-connectivity:



Randomized → st-connectivity: (sketch)

1. Decision trees $\{T_j\}_{j=1}^N$
   Probability distribution $\{p_j\}_{j=1}^N$.
2. *Wlog:* uniform distribution.
   $\Rightarrow$ gapped majority on N bits.

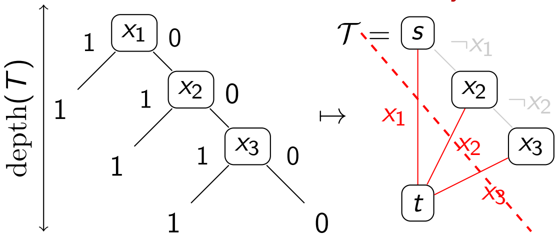
Deterministic → st-connectivity:

1. Decision tree T.
2. Conversion into *st*-connectivity.
3. $w_+(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
4. $w_-(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
5. $C(\mathcal{T}) \leq \mathrm{depth}(T)$.

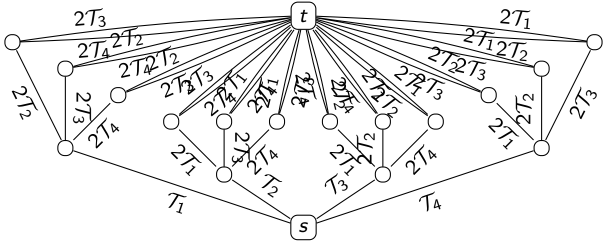
Randomized → st-connectivity: (sketch)

1. Decision trees $\{T_j\}_{j=1}^N$
   Probability distribution $\{p_j\}_{j=1}^N$.
2. *Wlog:* uniform distribution.
   ⇒ gapped majority on N bits.

Decision tree → st-connectivity:



Gapped majority:



Classical algorithms → *st*-connectivity

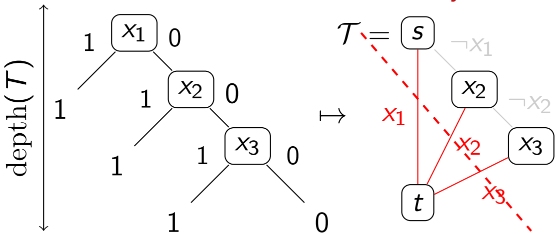
Deterministic → st-connectivity:

1. Decision tree T.
2. Conversion into st-connectivity.
3. $w_+(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
4. $w_-(x, \mathcal{T}) \leq \mathrm{depth}(T)$.
5. $C(\mathcal{T}) \leq \mathrm{depth}(T)$.

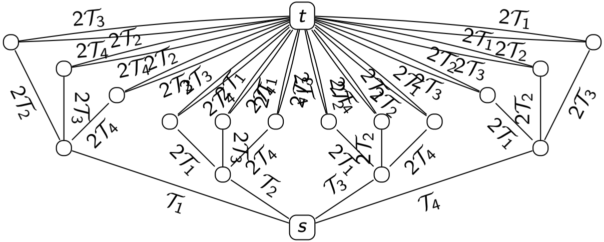
Randomized → st-connectivity: (sketch)

1. Decision trees $\{T_j\}_{j=1}^N$
   Probability distribution $\{p_j\}_{j=1}^N$.
2. *Wlog:* uniform distribution.
   $\Rightarrow$ gapped majority on N bits.
3. $C(\mathcal{P}) \in O(1 \cdot \max_j \mathrm{depth}(T_j))$.

Decision tree → st-connectivity:



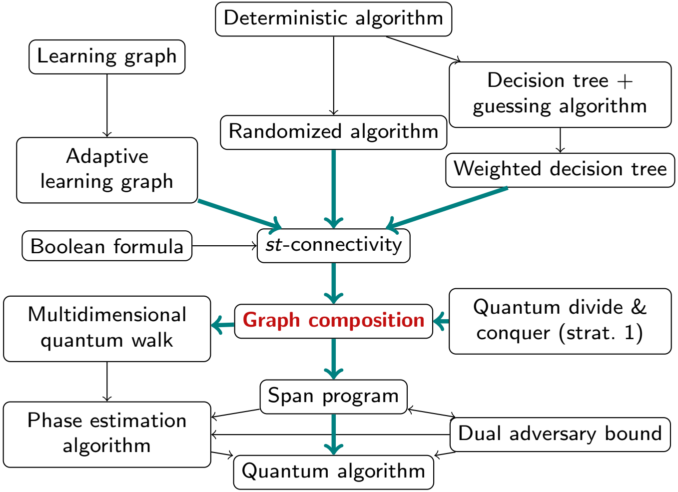
Gapped majority:



Summary

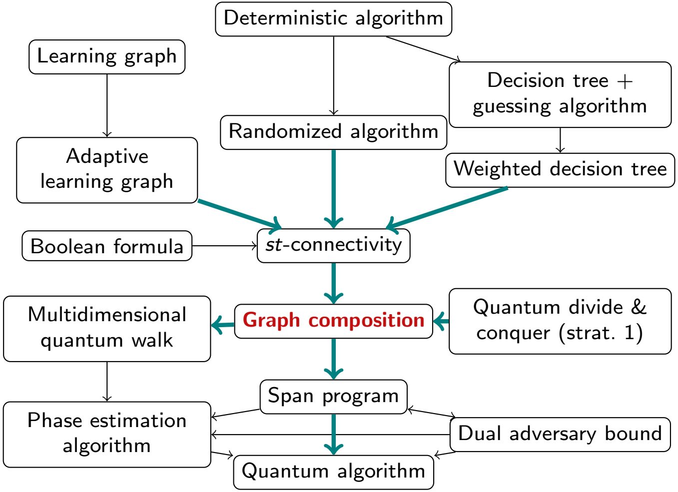
Summary

Relations between algorithmic frameworks



Summary

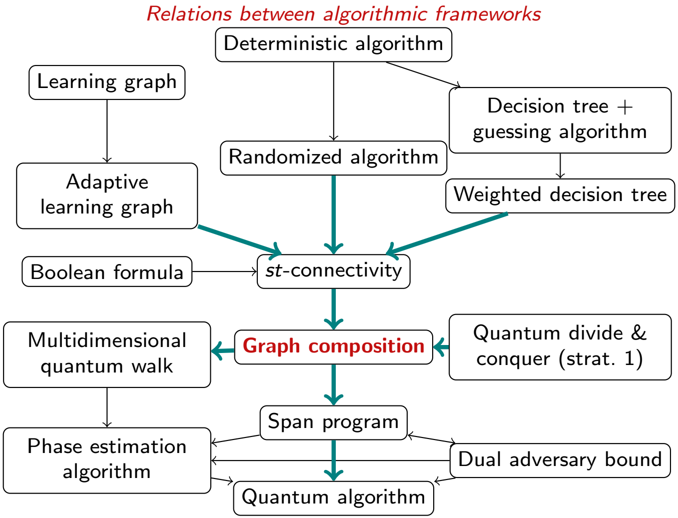
Relations between algorithmic frameworks



In this talk:

1. Span programs
2. Graph composition
3. Examples
4. Randomized $\rightarrow$ *st*-connectivity.

Summary



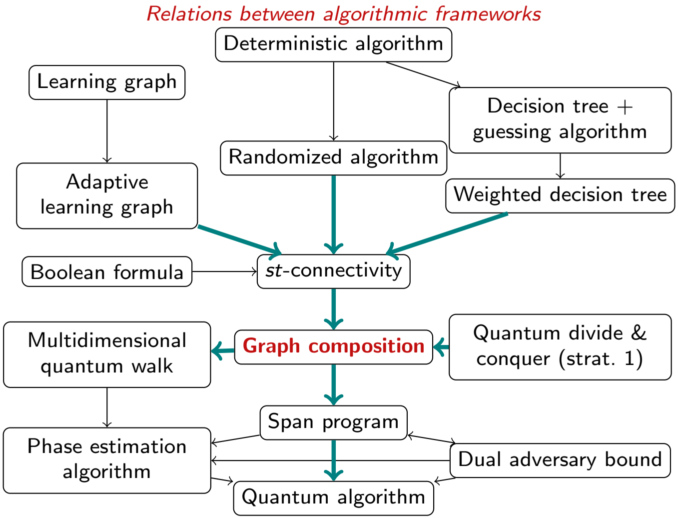
In this talk:

1. Span programs
2. Graph composition
3. Examples
4. Randomized $\to$ *st*-connectivity.

In the papers:

1. Time-efficient implementation of graph composition
2. Generalization to switches
3. Quantum walks frameworks
4. More examples

Summary



In this talk:

1. Span programs
2. Graph composition
3. Examples
4. Randomized $\rightarrow$ *st*-connectivity.

In the papers:

1. Time-efficient implementation of graph composition
2. Generalization to switches
3. Quantum walks frameworks
4. More examples

Open question:

Limitations of *st*-connectivity?

Thanks!

Thanks for your attention!

ajcornelissen@outlook.com

Example: the $\Sigma^*20^*2\Sigma^*$-problem

Example: the $\Sigma^*20^*2\Sigma^*$-problem

$\Sigma = \{0, 1, 2\}$, $f : \Sigma^n \to \{0, 1\}$.

Example: the $\Sigma^*20^*2\Sigma^*$-problem

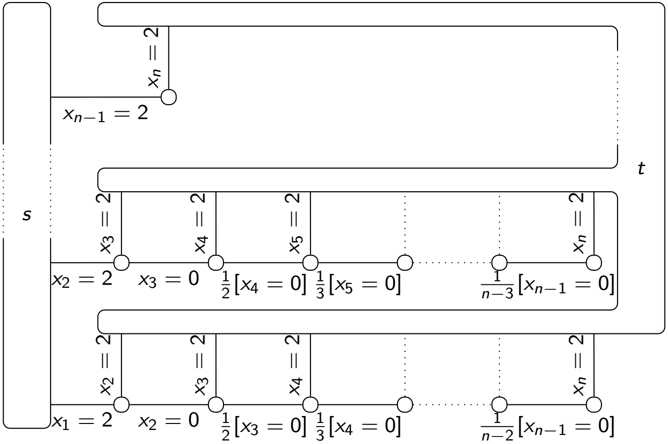
$\Sigma = \{0, 1, 2\}$, $f : \Sigma^n \to \{0, 1\}$.

1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.

Example: the $\Sigma^*20^*2\Sigma^*$-problem

$\Sigma = \{0, 1, 2\}$, $f : \Sigma^n \to \{0, 1\}$.

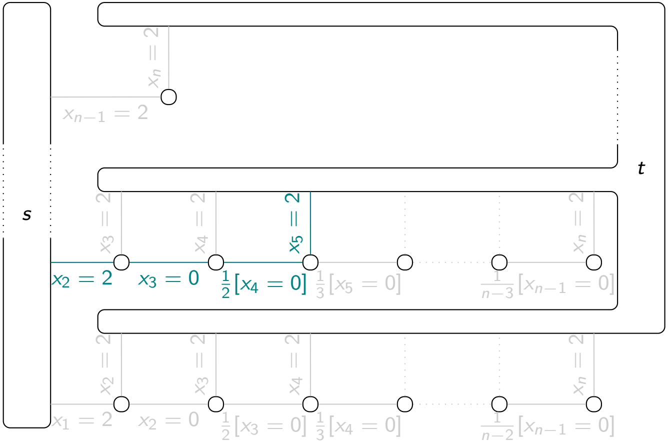
1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.



Example: the $\Sigma^*20^*2\Sigma^*$-problem

$\Sigma = \{0,1,2\}$, $f : \Sigma^n \to \{0,1\}$.

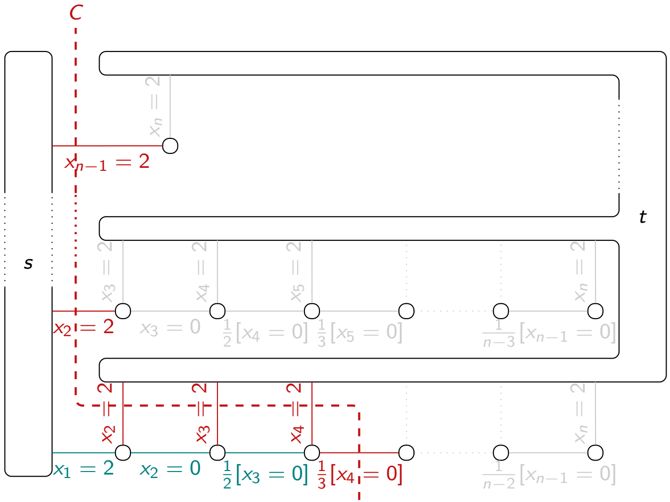
1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.
2. Let x be a positive instance.
   $x = \cdots 0102\underbrace{000000}_{\text{length } \ell}2100\cdots$
   $\Rightarrow w_+(x, \mathcal{P}) \leq 1 + \sum_{j=1}^{\ell} \frac{1}{j} + 1 \in O(\log(n))$.



Example: the $\Sigma^*20^*2\Sigma^*$-problem

$\Sigma = \{0,1,2\}$, $f : \Sigma^n \to \{0,1\}$.

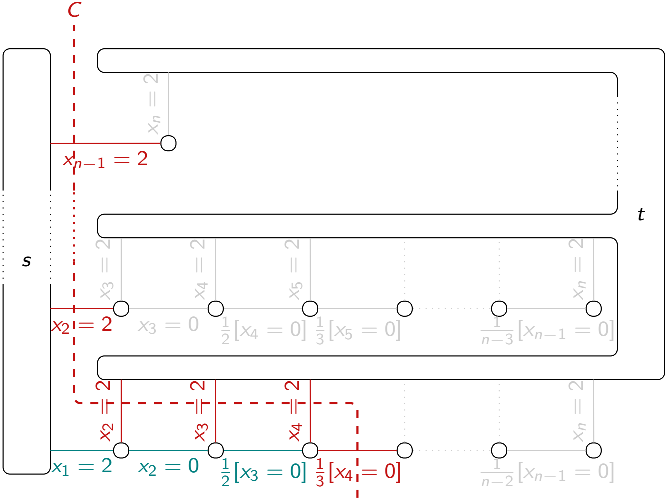
1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.
2. Let x be a positive instance.
   $x = \cdots 0102\underbrace{000000}_{\text{length } \ell}2100\cdots$
   $\Rightarrow w_+(x,\mathcal{P}) \leq$
   $1 + \sum_{j=1}^{\ell} \frac{1}{j} + 1 \in O(\log(n))$.
3. Let x be a negative instance.
   $x = 2\underbrace{001}_{\ell_1=3}102\underbrace{0001}_{\ell_2=4}002\underbrace{001}_{\ell_3=3}\cdots$
   $\Rightarrow w_-(x,\mathcal{P}) \leq$
   $n + \sum_{j=1}^{k} 2\ell_j \in O(n)$.



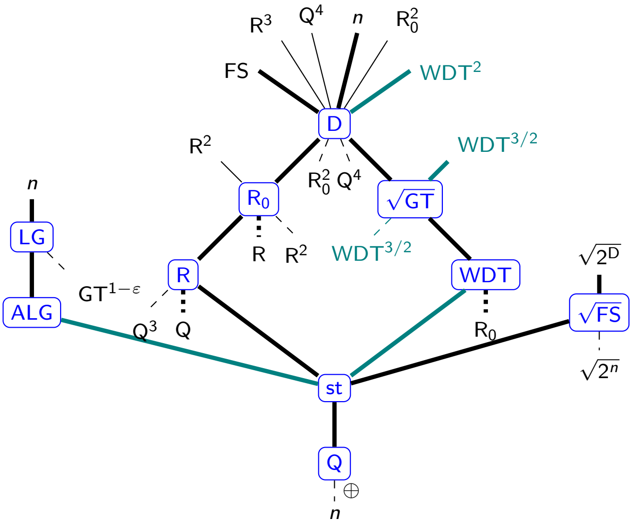
Example: the $\Sigma^*20^*2\Sigma^*$-problem

$\Sigma = \{0,1,2\}$, $f : \Sigma^n \to \{0,1\}$.

1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.
2. Let x be a positive instance.
   $x = \cdots 0102\underbrace{000000}_{\text{length } \ell}2100\cdots$
   $\Rightarrow w_+(x,\mathcal{P}) \leq 1 + \sum_{j=1}^{\ell} \frac{1}{j} + 1 \in O(\log(n))$.
3. Let x be a negative instance.
   $x = 2\underbrace{001}_{\ell_1=3}102\underbrace{0001}_{\ell_2=4}002\underbrace{001}_{\ell_3=3}\cdots$
   $\Rightarrow w_-(x,\mathcal{P}) \leq n + \sum_{j=1}^{k} 2\ell_j \in O(n)$.
4. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.



Complexity measure relations for total boolean functions



Legend:

1. Solid: relation.
2. Dashed: separation.
3. **Fat:** for partial functions.
4. Dotted: unbounded separation.
5. Green: new in this work.

Open questions:

1. Unbounded separation:
   1. Between Q and st?
   2. Between st and R?