Quantum algorithms through composition of graphs

arXiv:2504.02115 and arXiv:2510.04973

Arjan Cornelissen[1]

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

October 28th, 2025

Quantum algorithmic frameworks

Quantum algorithmic frameworks

Setting:

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

Quantum algorithmic frameworks

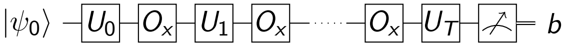
Setting:

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

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

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

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



Quantum algorithmic frameworks

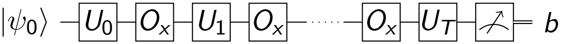
Setting:

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

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:

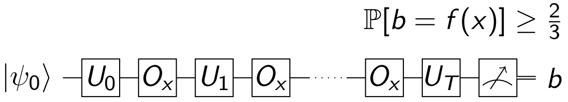
1. $\mathcal{D} \subseteq \{0,1\}^n$.
2. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.
3. $f : \mathcal{D} \to \{0,1\}$.

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.



Quantum algorithmic frameworks

Setting:

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

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.

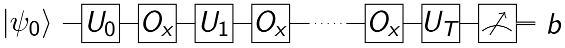
Framework:‘

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

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:‘

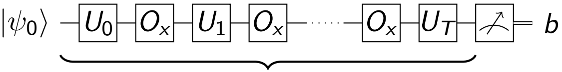
1. Define a mathematical object.
2. Convert object into quantum algorithm.

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]

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



$$T = O(\sqrt{\ell})$$

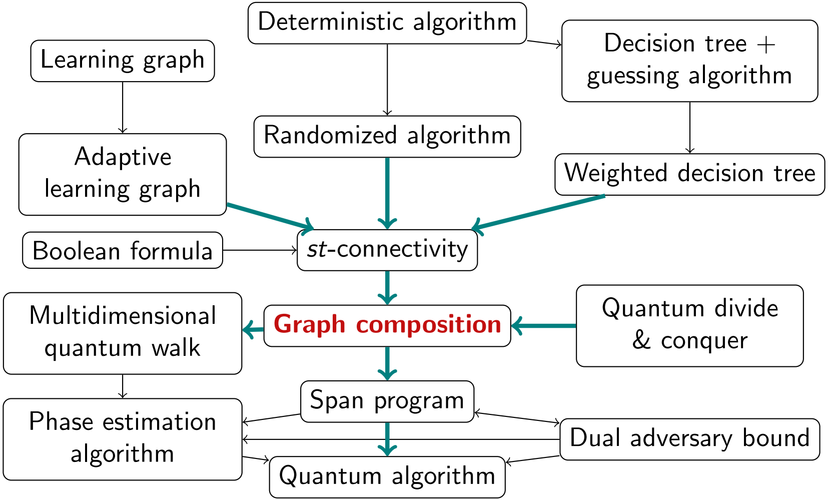
Quantum algorithmic framework landscape

Quantum algorithmic framework landscape

Overview:

→ Conversion between framework objects

→ New relations [This work]



Quantum algorithmic framework landscape

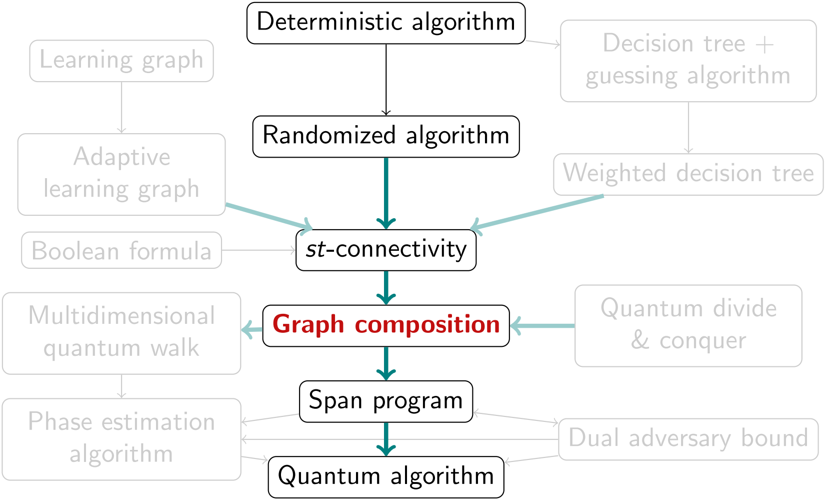
Overview:

→ 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]

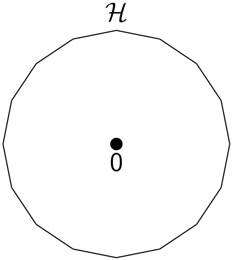
Span programs [RŠ12,Rei09,Rei11]

Four ingredients:

Span programs [RŠ12,Rei09,Rei11]

Four ingredients:

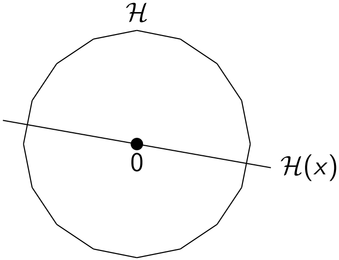
1. *Hilbert space:* $\mathcal{H}$.



Span programs [RŠ12,Rei09,Rei11]

Four ingredients:

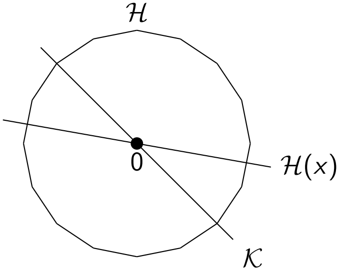
1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.



Span programs [RŠ12,Rei09,Rei11]

Four ingredients:

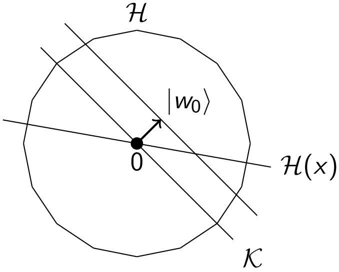
1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.



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:* $|w_0\rangle \in \mathcal{K}^\perp$.



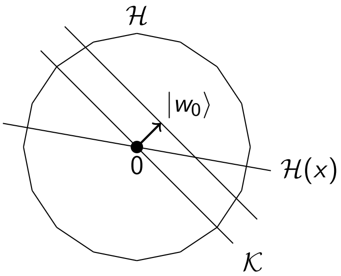
Span programs [RŠ12,Rei09,Rei11]

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



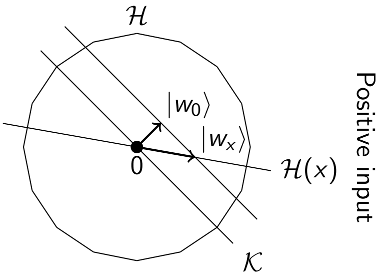
Span programs [RŠ12,Rei09,Rei11]

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:* $w_+(x,\mathcal{P}) = \||w_x\rangle\|^2$.



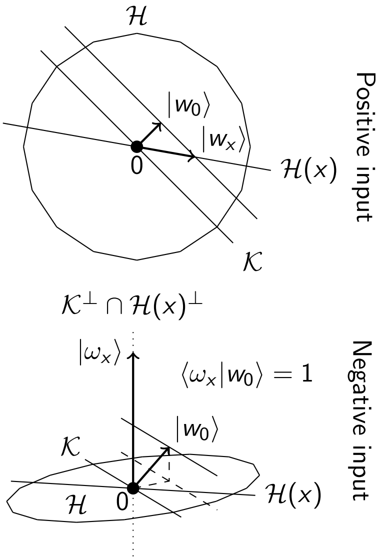
Span programs [RŠ12,Rei09,Rei11]

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:* $w_-(x, \mathcal{P}) = \||\omega_x\rangle\|^2$.



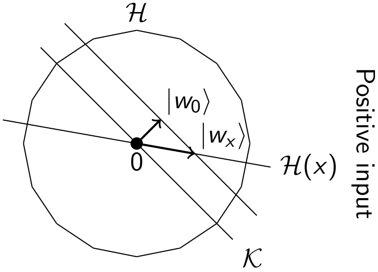
Span programs [RŠ12,Rei09,Rei11]

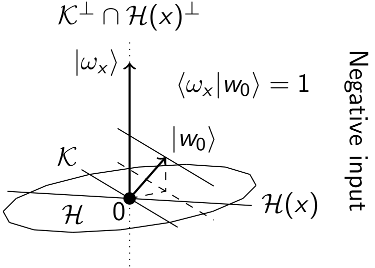
Four ingredients:

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:* $w_-(x,\mathcal{P}) = \||\omega_x\rangle\|^2$.
4. $C(\mathcal{P}) = \sqrt{\max_{x \in f^{-1}(0)} w_-(x,\mathcal{P}) \cdot \max_{x \in f^{-1}(1)} w_+(x,\mathcal{P})}$.





Span programs [RŠ12,Rei09,Rei11]

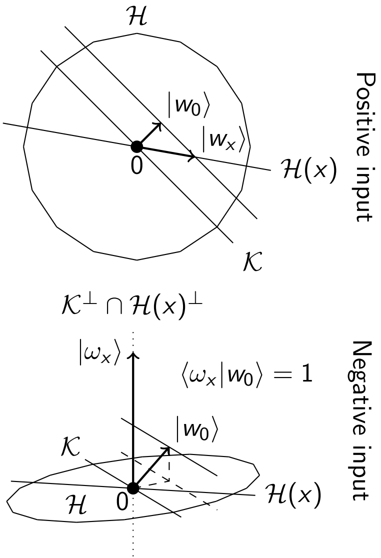
Four ingredients:

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:* $w_-(x,\mathcal{P}) = \||\omega_x\rangle\|^2$.
4. $C(\mathcal{P}) = \sqrt{\max_{x \in f^{-1}(0)} w_-(x,\mathcal{P}) \cdot \max_{x \in f^{-1}(1)} w_+(x,\mathcal{P})}$.

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



Span program manipulations

1. *Scalar multiplication* ($\alpha > 0$):

$$\begin{array}{c} \mathcal{P} \\ \shortparallel \\ \left\{ \begin{array}{c} \mathcal{H} \\ \mathcal{H}(x) \\ \mathcal{K} \\ |w_0\rangle \end{array} \right\} \end{array} \mapsto \begin{array}{c} \alpha\mathcal{P} \\ \shortparallel \\ \left\{ \begin{array}{c} \mathcal{H} \\ \mathcal{H}(x) \\ \mathcal{K} \\ \sqrt{\alpha}\,|w_0\rangle \end{array} \right\} \end{array}.$$

Span program manipulations

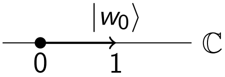
1. *Scalar multiplication* ($\alpha > 0$):
   1. $w_+(x, \alpha\mathcal{P}) = \alpha w_+(x, \mathcal{P})$
   2. $w_-(x, \alpha\mathcal{P}) = \frac{w_-(x,\mathcal{P})}{\alpha}$.

$$\begin{array}{ccc} \mathcal{P} & & \alpha\mathcal{P} \\ \shortparallel & & \shortparallel \\ \left\{\begin{array}{c} \mathcal{H} \\ \mathcal{H}(x) \\ \mathcal{K} \\ |w_0\rangle \end{array}\right\} & \mapsto & \left\{\begin{array}{c} \mathcal{H} \\ \mathcal{H}(x) \\ \mathcal{K} \\ \sqrt{\alpha}\,|w_0\rangle \end{array}\right\}. \end{array}$$

Span program manipulations

1. *Scalar multiplication* $(\alpha > 0)$:
   1. $w_+(x, \alpha\mathcal{P}) = \alpha w_+(x, \mathcal{P})$
   2. $w_-(x, \alpha\mathcal{P}) = \frac{w_-(x,\mathcal{P})}{\alpha}$.
2. *Trivial span program* (query x_j):
   1. $\mathcal{H} = \mathbb{C}$
   2. $\mathcal{H}(x) = \begin{cases} \mathbb{C}, & \text{if } x_j = 1, \\ \{0\}, & \text{otherwise.} \end{cases}$
   3. $\mathcal{K} = \{0\}$.
   4. $|w_0\rangle = 1$.

$$\underset{\left\{\begin{matrix} \mathcal{H} \\ \mathcal{H}(x) \\ \mathcal{K} \\ |w_0\rangle \end{matrix}\right\}}{\overset{\mathcal{P}}{\shortparallel}} \mapsto \underset{\left\{\begin{matrix} \mathcal{H} \\ \mathcal{H}(x) \\ \mathcal{K} \\ \sqrt{\alpha}\,|w_0\rangle \end{matrix}\right\}}{\overset{\alpha\mathcal{P}}{\shortparallel}}.$$



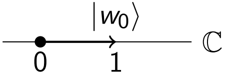
Span program manipulations

1. *Scalar multiplication* ($\alpha > 0$):
   1. $w_+(x, \alpha\mathcal{P}) = \alpha w_+(x, \mathcal{P})$
   2. $w_-(x, \alpha\mathcal{P}) = \frac{w_-(x,\mathcal{P})}{\alpha}$.
2. *Trivial span program* (query x_j):
   1. $\mathcal{H} = \mathbb{C}$
   2. $\mathcal{H}(x) = \begin{cases} \mathbb{C}, & \text{if } x_j = 1, \\ \{0\}, & \text{otherwise.} \end{cases}$
   3. $\mathcal{K} = \{0\}$.
   4. $|w_0\rangle = 1$.

   Then,
   1. For $x_j = 1$: $w_+(x, \mathcal{P}) = 1$.
   2. For $x_j = 0$: $w_-(x, \mathcal{P}) = 1$.

   $\Rightarrow C(x_j) = 1$.

$$\begin{array}{ccc} \mathcal{P} & & \alpha\mathcal{P} \\ \shortparallel & & \shortparallel \\ \left\{\begin{array}{c} \mathcal{H} \\ \mathcal{H}(x) \\ \mathcal{K} \\ |w_0\rangle \end{array}\right\} & \mapsto & \left\{\begin{array}{c} \mathcal{H} \\ \mathcal{H}(x) \\ \mathcal{K} \\ \sqrt{\alpha}\,|w_0\rangle \end{array}\right\}. \end{array}$$

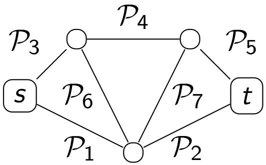


Graph composition [This work]

Graph composition [This work]

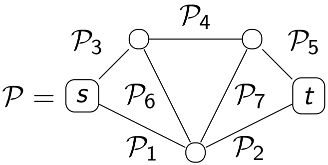
Ingredients:

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 [This work]

Graph composition:



Ingredients:

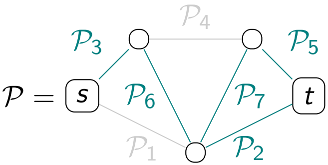
1. Undirected graph $G = (V, E)$.
2. Source and target vertices $s, t \in V$.
3. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph composition: Span program $\mathcal{P}$, such that

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 [This work]

Positive input:



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: Span program $\mathcal{P}$, such that

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 [This work]

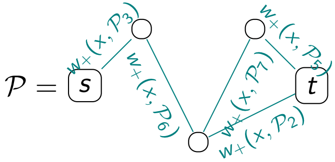
Ingredients:

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: Span program $\mathcal{P}$, such that

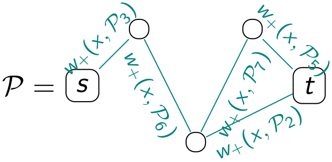
1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.

Positive input:



Graph composition [This work]

Positive input:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

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: Span program $\mathcal{P}$, such that

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. $w_+(x, \mathcal{P}) = R^{\mathit{eff}}_{s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.

Graph composition [This work]

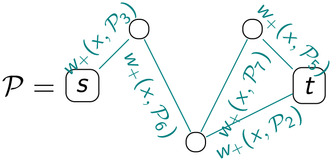
Ingredients:

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: Span program $\mathcal{P}$, such that

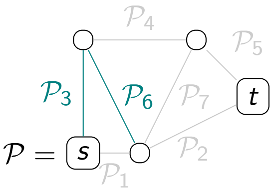
1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.
2. $w_+(x, \mathcal{P}) = R^{\mathrm{eff}}_{s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.

Positive input:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

Negative input:



Graph composition [This work]

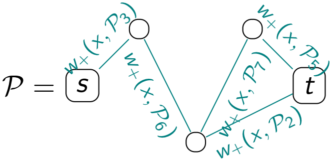
Ingredients:

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: Span program $\mathcal{P}$, such that

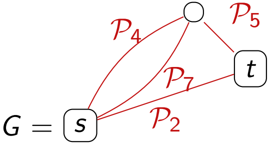
1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.
2. $w_+(x, \mathcal{P}) = R^{\mathit{eff}}_{s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.

Positive input:



$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$.

Negative input:



Graph composition [This work]

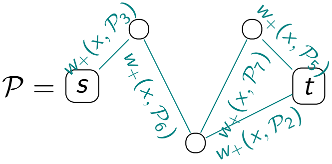
Ingredients:

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: Span program $\mathcal{P}$, such that

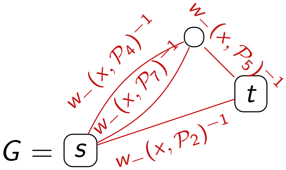
1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.
2. $w_+(x, \mathcal{P}) = R^{\mathit{eff}}_{s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.

Positive input:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

Negative input:



Graph composition [This work]

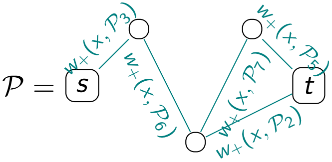
Ingredients:

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: Span program $\mathcal{P}$, such that

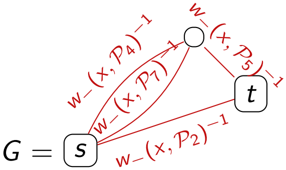
1. $\mathcal{P}$ computes whether s and t are connected by a path $e_1, \dots e_k$, such that x is positive for all $\mathcal{P}_{e_j}$.
2. $w_+(x, \mathcal{P}) = R^{\mathrm{eff}}_{s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
3. $w_-(x, \mathcal{P}) = (R^{\mathrm{eff}}_{s,t,r^-})^{-1}$ with $r^-(e)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

Positive input:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

Negative input:



$$w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}.$$

Graph composition [This work]

Ingredients:

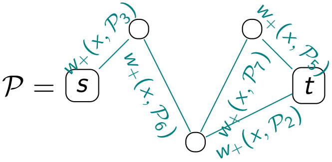
1. Undirected graph $G = (V, E)$.
2. Source and target vertices $s, t \in V$.
3. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph composition: Span program $\mathcal{P}$, such that

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 $r_+(e) = w_+(x, \mathcal{P}_e)$.
3. $w_-(x, \mathcal{P}) = (R^{\mathrm{eff}}_{s,t,r^-})^{-1}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

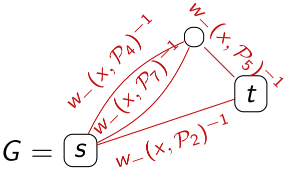
Remark: Recovers st-connectivity with just trivial span programs. [BR12,JK17,JJKP18]

Positive input:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

Negative input:



$$w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}.$$

Path-cut theorem

Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

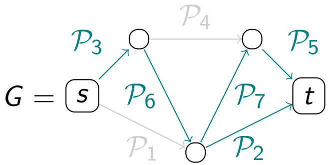
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:

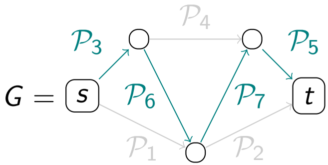


Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

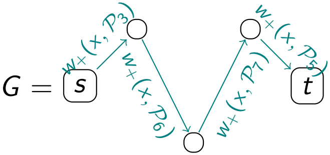
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



Path-cut theorem

Positive input:



Theorem: For all $x \in \mathcal{D}$,

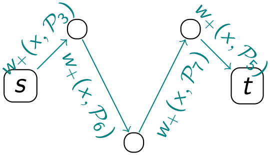
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Path-cut theorem

Positive input:

Theorem: For all $x \in \mathcal{D}$,

1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

$G =$ 

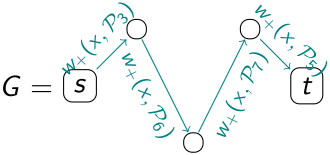
$$w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e).$$

Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

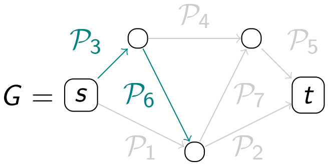
1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



$w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.

Negative input:

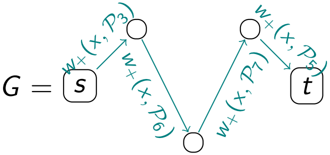


Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

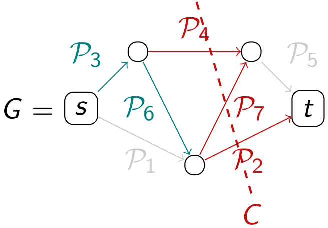
1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



$w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.

Negative input:

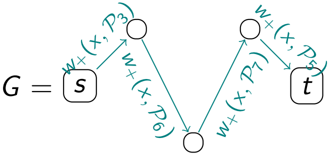


Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

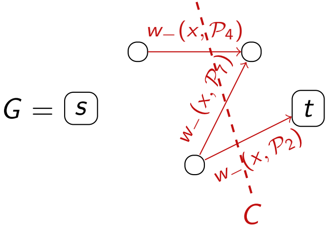
1. Let P be a path from s to t: $w_+(x,\mathcal{P}) \leq \sum_{e \in P} w_+(x,\mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x,\mathcal{P}) \leq \sum_{e \in C} w_-(x,\mathcal{P}_e)$.

Positive input:



$$w_+(x,\mathcal{P}) \leq \sum_{e \in P} w_+(x,\mathcal{P}_e).$$

Negative input:

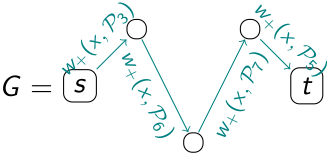


Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

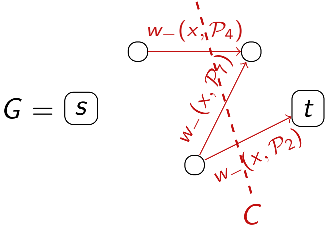
1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



$w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.

Negative input:



$w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

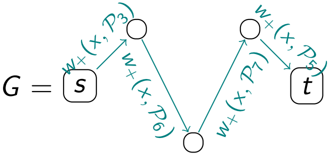
Theorem: For all $x \in \mathcal{D}$,

1. Let P be a path from s to t:
   $w_+(x,\mathcal{P}) \leq \sum_{e \in P} w_+(x,\mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x,\mathcal{P}) \leq \sum_{e \in C} w_-(x,\mathcal{P}_e)$.

Properties:

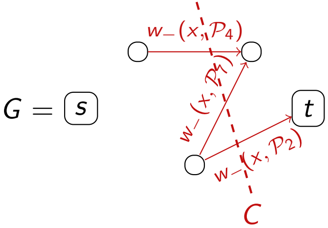
1. Simpler (less-powerful) version.
2. Still powerful enough for many applications.

Positive input:



$w_+(x,\mathcal{P}) \leq \sum_{e \in P} w_+(x,\mathcal{P}_e)$.

Negative input:



$w_-(x,\mathcal{P}) \leq \sum_{e \in C} w_-(x,\mathcal{P}_e)$.

Example: OR-function

Example: OR-function

The OR-function:

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

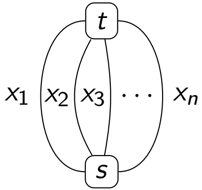
   $$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$

Example: OR-function

The OR-function:

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

Graph composition for OR_n:

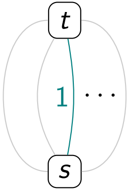


Example: OR-function

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. $w_+(x) \leq 1$.

Graph composition for OR_n:



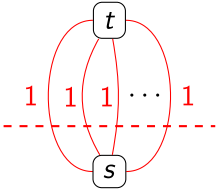
$x = 0010\cdots0 \Rightarrow w_+(x) \leq 1$

Example: OR-function

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. $w_-(x) \leq n$.

Graph composition for OR_n:



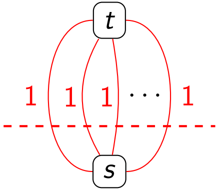
$x = 0000 \cdots 0 \Rightarrow w_-(x) = n$

Example: OR-function

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. $C(\mathcal{P}) \leq \sqrt{n}$.

Graph composition for OR_n:



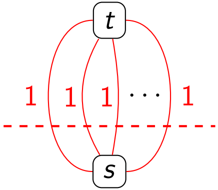
$x = 0000\cdots 0 \Rightarrow w_-(x) = n$

Example: OR-function

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. $C(\mathcal{P}) \leq \sqrt{n}$.
5. $\Rightarrow Q(\mathrm{OR}_n) \in O(\sqrt{n})$.

Graph composition for OR_n:



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

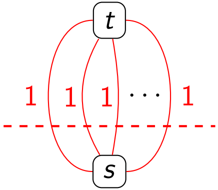
Example: OR-function

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. $C(\mathcal{P}) \leq \sqrt{n}$.
5. $\Rightarrow Q(\mathrm{OR}_n) \in O(\sqrt{n})$.

Quadratic speed-up for search.

Graph composition for OR_n:



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

Example: Threshold function

Example: Threshold function

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

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

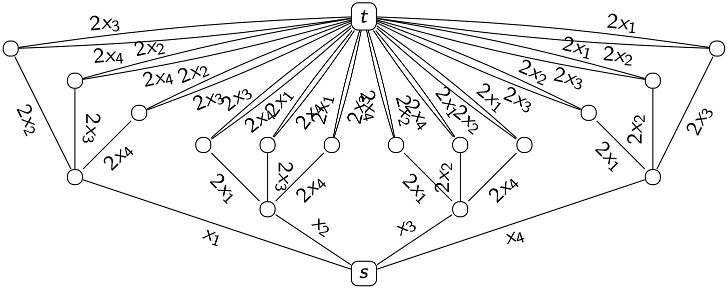
Example: Threshold function

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

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

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

Graph composition for Th_4^3*:*

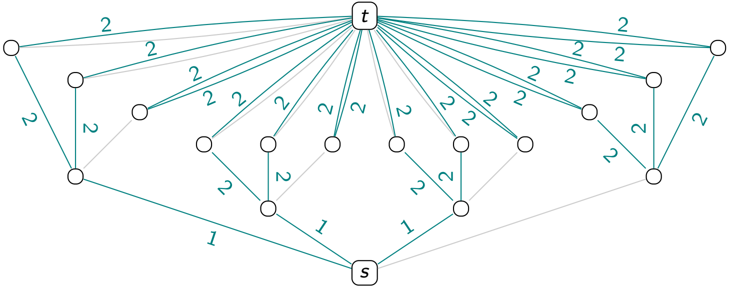


Example: Threshold function

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. $w_+(x) = \frac{1}{|x|-k+1}$

Graph composition for Th_4^3*:*



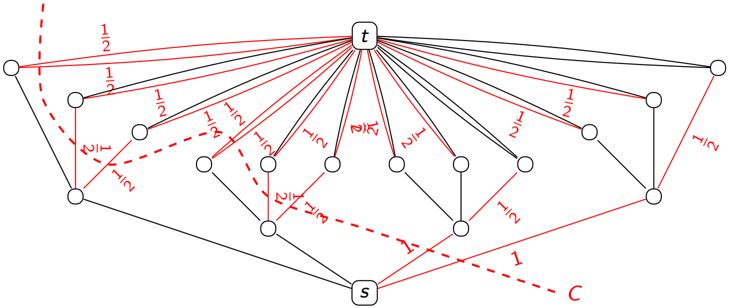
$x = 1110 \Rightarrow w_+(x) = 1$

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. $w_-(x) = \frac{k(n-k+1)}{k-|x|}$

Graph composition for Th_4^3*:*



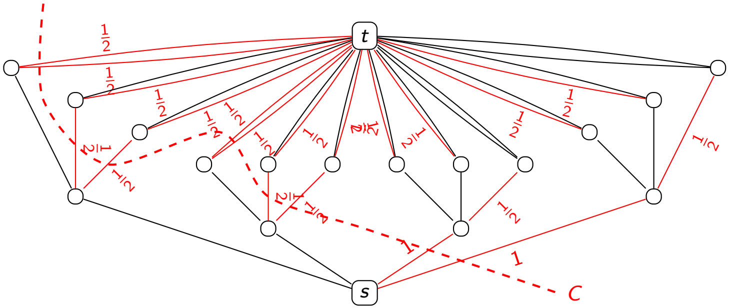
$x = 1100 \Rightarrow w_-(x) = 6$

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. $C(\mathcal{P}) = \sqrt{k(n-k+1)}$.

Graph composition for Th_4^3*:*



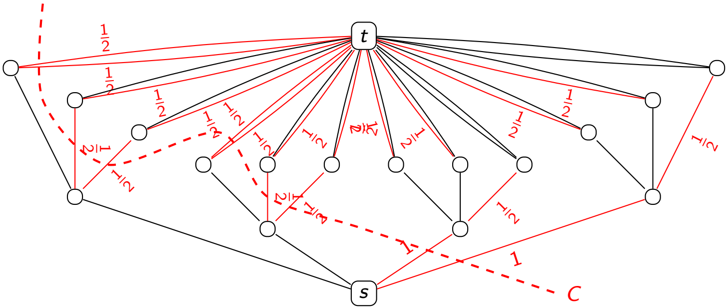
$x = 1100 \Rightarrow w_-(x) = 6$

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. $C(\mathcal{P}) = \sqrt{k(n-k+1)}$.
5. $\Rightarrow Q(\mathrm{Th}_n^k) \in O(\sqrt{k(n-k+1)})$.

Graph composition for Th_4^3*:*



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

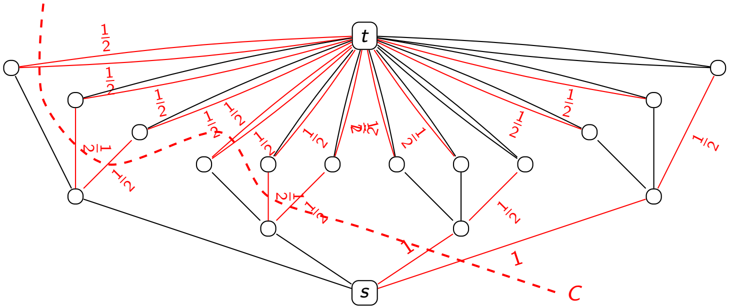
Example: Threshold function

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. $C(\mathcal{P}) = \sqrt{k(n-k+1)}$.
5. $\Rightarrow Q(\mathrm{Th}_n^k) \in O(\sqrt{k(n-k+1)})$.

Known to be optimal!

Graph composition for Th_4^3*:*



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

Example: Threshold function

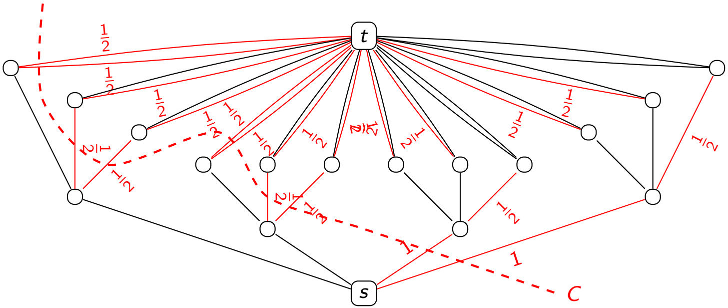
The threshold function: $(k \in [n])$

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. $C(\mathcal{P}) = \sqrt{k(n-k+1)}$.
5. $\Rightarrow Q(\mathrm{Th}_n^k) \in O(\sqrt{k(n-k+1)})$.

Known to be optimal!

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

Graph composition for Th_4^3*:*



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

Classical algorithms → *st*-connectivity

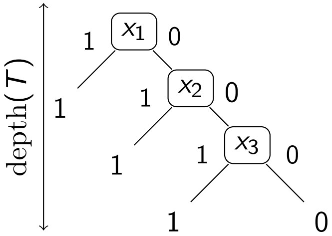
Classical algorithms $\to$ *st*-connectivity

Deterministic $\to$ st-connectivity:

Classical algorithms → *st*-connectivity

Deterministic → st-connectivity:

1. Decision tree T.

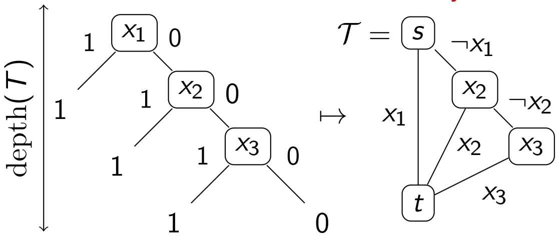


Classical algorithms $\to$ *st*-connectivity

Deterministic $\to$ st-connectivity:

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

Decision tree $\to$ st-connectivity:

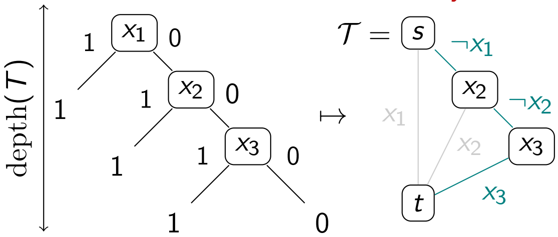


Classical algorithms → *st*-connectivity

Deterministic → st-connectivity:

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

Decision tree → st-connectivity:

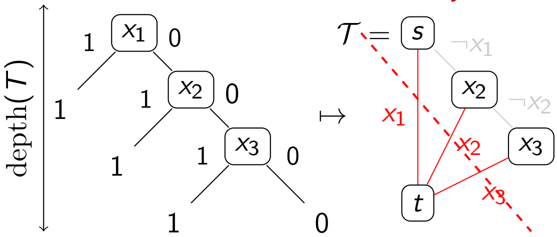


Classical algorithms $\to$ *st*-connectivity

Deterministic $\to$ st-connectivity:

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

Decision tree $\to$ st-connectivity:

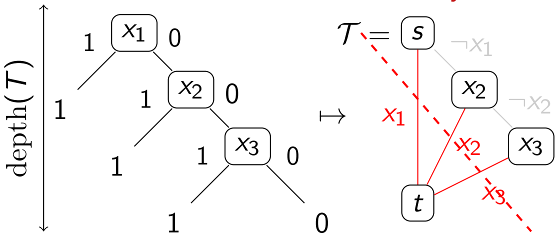


Classical algorithms $\to$ *st*-connectivity

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:

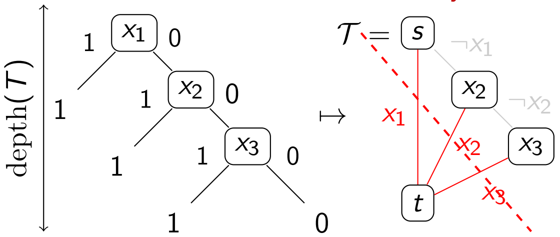


Classical algorithms $\to$ *st*-connectivity

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. $C(\mathcal{T}) \leq \mathrm{depth}(T)$.

Decision tree $\to$ st-connectivity:



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

Classical algorithms $\to$ *st*-connectivity

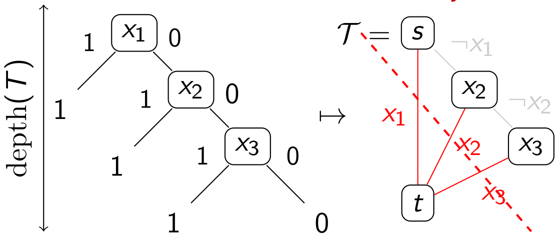
Deterministic $\to$ st-connectivity:

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)

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

Decision tree $\to$ st-connectivity:



Classical algorithms $\to$ *st*-connectivity

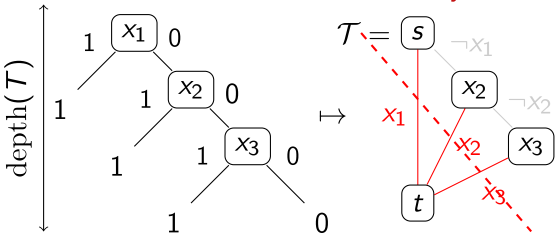
Deterministic $\to$ st-connectivity:

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)

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.

Decision tree $\to$ st-connectivity:



Classical algorithms → *st*-connectivity

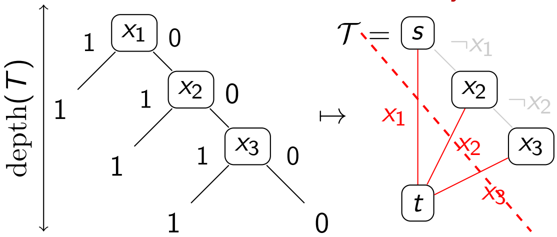
Deterministic → st-connectivity:

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. $C(\mathcal{T}) \leq \mathrm{depth}(T)$.

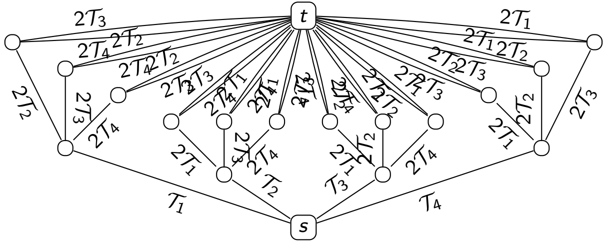
Randomized → st-connectivity: (sketch)

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.

Decision tree → st-connectivity:



Gapped majority:



Classical algorithms → *st*-connectivity

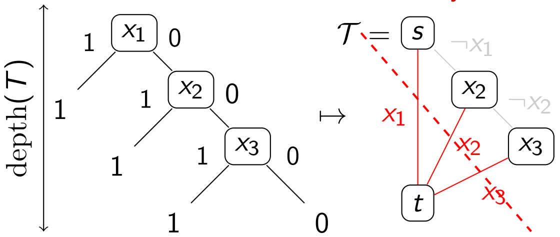
Deterministic → st-connectivity:

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. $C(\mathcal{T}) \leq \mathrm{depth}(T)$.

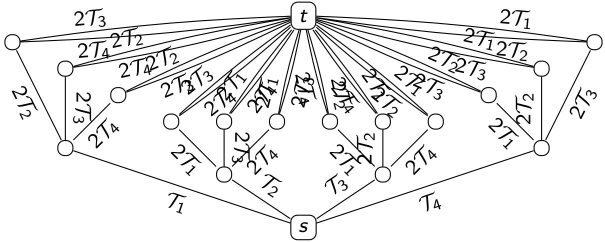
Randomized → st-connectivity: (sketch)

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:

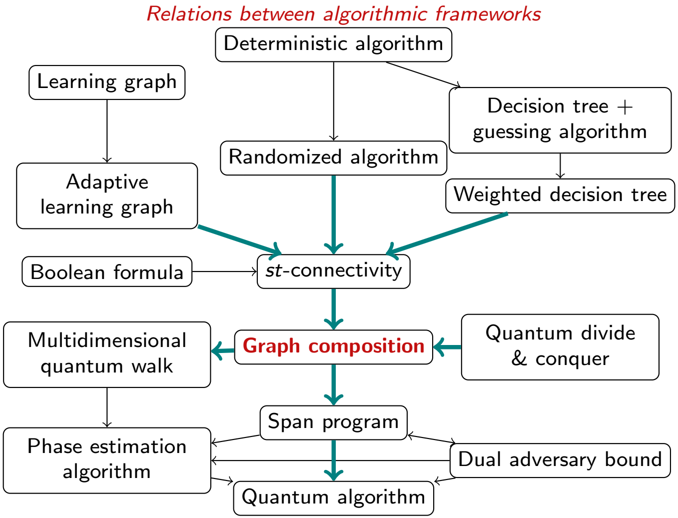


Gapped majority:



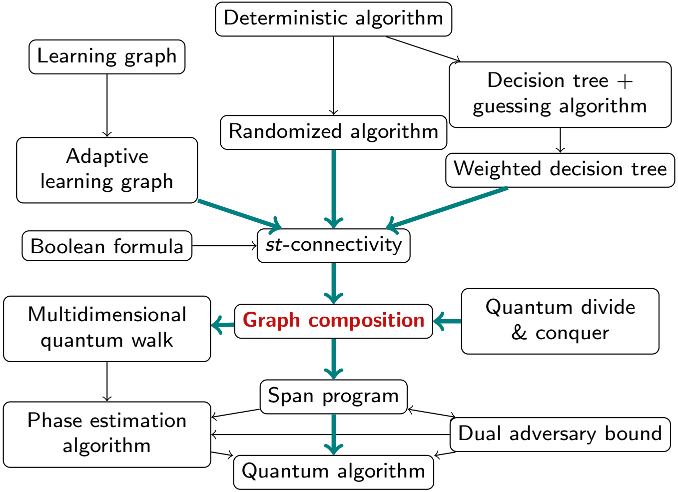
Summary

Summary


Relations between algorithmic frameworks
Deterministic algorithm
Learning graph
Decision tree + guessing algorithm
Randomized algorithm
Adaptive learning graph
Weighted decision tree
Boolean formula
st-connectivity
Multidimensional quantum walk
Graph composition
Quantum divide & conquer
Span program
Phase estimation algorithm
Dual adversary bound
Quantum algorithm

Summary

Relations between algorithmic frameworks

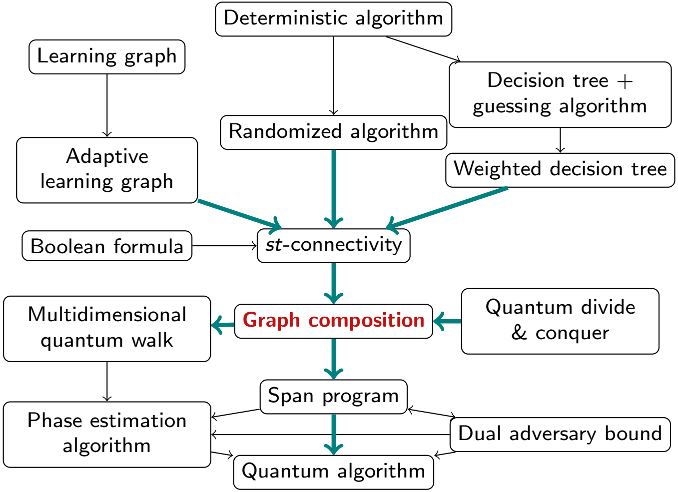


In this talk:

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

Summary

Relations between algorithmic frameworks



In this talk:

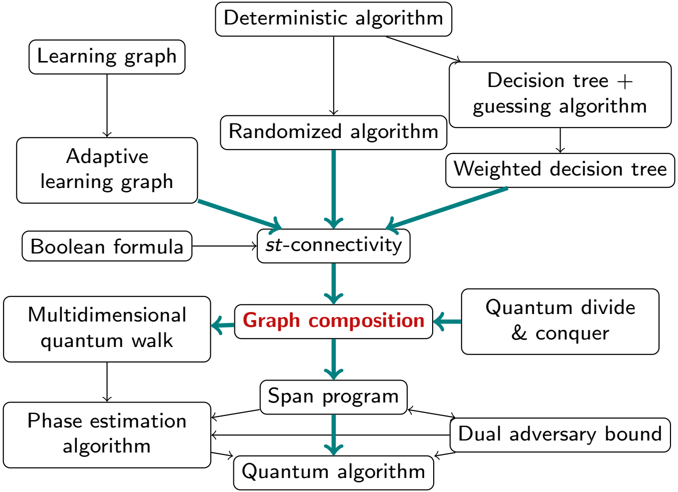
1. Span programs
2. Graph composition
3. Examples
4. Randomized $\rightarrow$ *st*-connectivity.

In the papers:

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

Summary

Relations between algorithmic frameworks



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

Open question:

Limitations of *st*-connectivity?

Thanks!

Thanks for your attention!

`ajcornelissen@outlook.com`