Improved Quantum Query Upper Bounds Based on Classical Decision Trees

arXiv:2203.02968

Arjan Cornelissen[1], Nikhil S. Mande[2], Subhasree Patro[3]

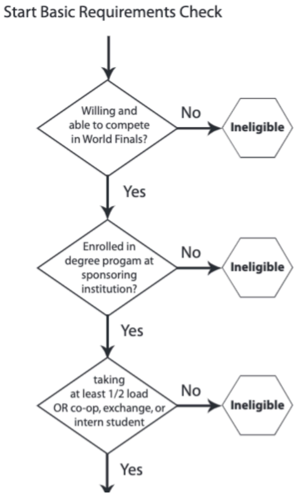
[1]QuSoft, University of Amsterdam
[2]QuSoft, CWI Amsterdam
[3]QuSoft, CWI Amsterdam

July 11th, 2022

Research Center for Quantum Software

Decision trees

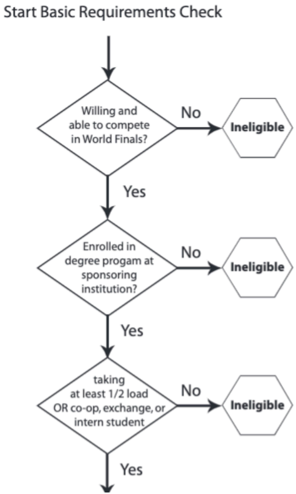
Decision trees



Decision trees

In general:

1. Rooted tree.
2. Every node has a decision rule.
3. Leafs are labeled by outputs.



Decision trees

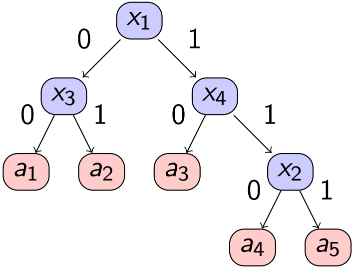
In general:

1. Rooted tree.
2. Every node has a decision rule.
3. Leafs are labeled by outputs.

For the purposes of this talk:

1. Input is a bit string $x \in \{0,1\}^n$,
2. Nodes are single bit queries.
3. Decision tree defines $f : \{0,1\}^n \to A$.

$x \in \{0,1\}^n$



Decision trees

In general:

1. Rooted tree.
2. Every node has a decision rule.
3. Leafs are labeled by outputs.

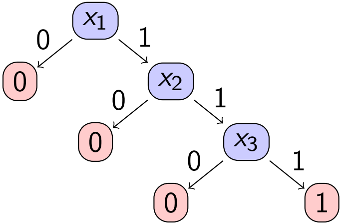
For the purposes of this talk:

1. Input is a bit string $x \in \{0,1\}^n$,
2. Nodes are single bit queries.
3. Decision tree defines $f : \{0,1\}^n \to A$.

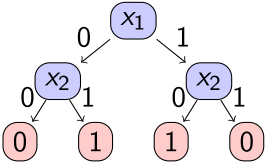
Examples:

1. *AND*-decision tree.
2. *PARITY*-decision tree.

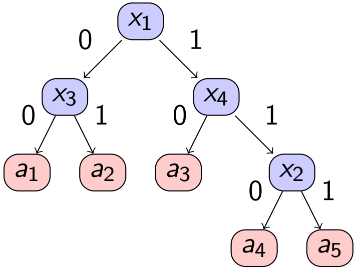
AND-tree



PARITY-tree



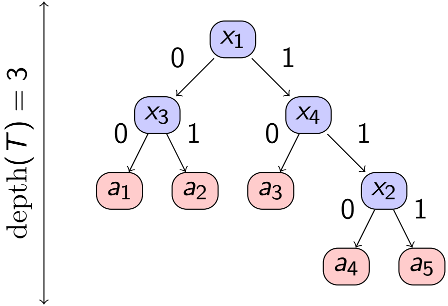
Decision tree measures



Decision tree measures

Measures:

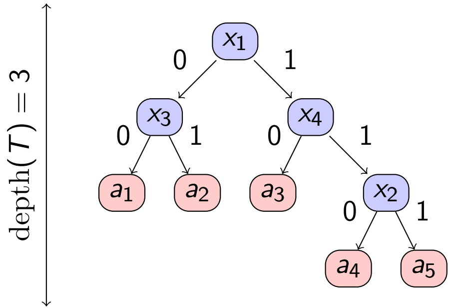
1. $\mathrm{depth}(T)$ – Depth
   Number of layers of decision nodes.



Decision tree measures

Measures:

1. $\mathrm{depth}(T)$ – Depth
   Number of layers of decision nodes.
2. $\mathrm{size}(T)$ – Size
   Number of decision nodes.

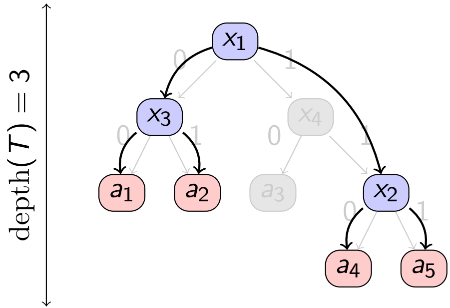


$\mathrm{size}(T) = 4$

Decision tree measures

Measures:

1. $\mathrm{depth}(T)$ – Depth
   Number of layers of decision nodes.
2. $\mathrm{size}(T)$ – Size
   Number of decision nodes.
3. $\mathrm{rank}(T)$ – Rank
   Depth of largest full binary subtree.

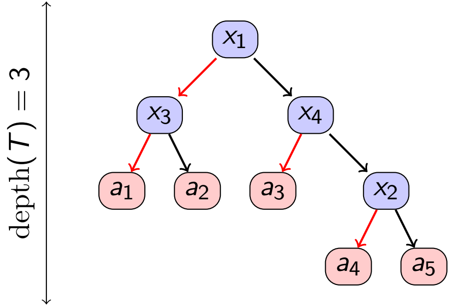


$$\begin{aligned}\mathrm{size}(T) &= 4\\ \mathrm{rank}(T) &= 2\end{aligned}$$

Decision tree measures

Measures:

1. $\mathrm{depth}(T)$ – Depth
   Number of layers of decision nodes.
2. $\mathrm{size}(T)$ – Size
   Number of decision nodes.
3. $\mathrm{rank}(T)$ – Rank
   Depth of largest full binary subtree.
4. $G(T)$ – Guessing complexity
   Most number of red edges in path.



$$\begin{aligned} \mathrm{size}(T) &= 4 \\ \mathrm{rank}(T) &= 2 \\ G(T) &= 2 \end{aligned}$$

Function measures

We can lift decision tree measures to function measures.

Function measures

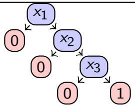
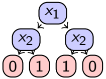
We can lift decision tree measures to function measures.

1. Let $f : \{0,1\}^n \to A$.
2. Let $\mathrm{m} \in \{\mathrm{depth}, \mathrm{size}, \mathrm{rank}, G\}$.
3. $\mathrm{m}(f) = \min_{T : T \text{ computes } f} \mathrm{m}(T)$.

Function measures

We can lift decision tree measures to function measures.

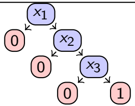
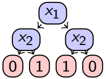
1. Let $f : \{0,1\}^n \to A$.
2. Let $\mathrm{m} \in \{\mathrm{depth}, \mathrm{size}, \mathrm{rank}, G\}$.
3. $\mathrm{m}(f) = \min_{T : T \text{ computes } f} \mathrm{m}(T)$.

f	AND	$PARITY$
T		
$\mathrm{depth}(f)$	n	n
$\mathrm{size}(f)$	n	$2^n - 1$
$\mathrm{rank}(f)$	1	n
$G(f)$	1	n

Function measures

We can lift decision tree measures to function measures.

1. Let $f : \{0,1\}^n \to A$.
2. Let $\mathrm{m} \in \{\mathrm{depth}, \mathrm{size}, \mathrm{rank}, G\}$.
3. $\mathrm{m}(f) = \min\limits_{T : T \text{ computes } f} \mathrm{m}(T)$.

f	AND	$PARITY$
T		
$\mathrm{depth}(f)$	n	n
$\mathrm{size}(f)$	n	$2^n - 1$
$\mathrm{rank}(f)$	1	n
$G(f)$	1	n

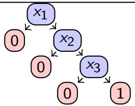
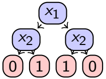
Randomized measures:

1. Let $\mathcal{T}$ be a family of decision trees. It approximately computes f, if $\forall x, \mathbb{P}_{T \in_R \mathcal{T}}[T(x) = f(x)] \geq \frac{2}{3}$.

Function measures

We can lift decision tree measures to function measures.

1. Let $f : \{0,1\}^n \to A$.
2. Let $\mathrm{m} \in \{\mathrm{depth}, \mathrm{size}, \mathrm{rank}, G\}$.
3. $\mathrm{m}(f) = \min_{T : T \text{ computes } f} \mathrm{m}(T)$.

f	AND	$PARITY$
T		
$\mathrm{depth}(f)$	n	n
$\mathrm{size}(f)$	n	$2^n - 1$
$\mathrm{rank}(f)$	1	n
$G(f)$	1	n

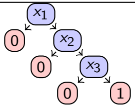
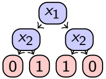
Randomized measures:

1. Let $\mathcal{T}$ be a family of decision trees. It approximately computes f, if $\forall x, \mathbb{P}_{T \in_R \mathcal{T}}[T(x) = f(x)] \geq \frac{2}{3}$.
2. Let $\mathrm{m} \in \{\mathrm{depth}, \mathrm{size}, \mathrm{rank}, G\}$, $\mathrm{rm}(f) = \min_{\mathcal{T} \text{ approx. computes } f} \max_{T \in \mathcal{T}} \mathrm{m}(f)$.

Function measures

We can lift decision tree measures to function measures.

1. Let $f : \{0,1\}^n \to A$.
2. Let $\mathrm{m} \in \{\mathrm{depth}, \mathrm{size}, \mathrm{rank}, G\}$.
3. $\mathrm{m}(f) = \min\limits_{T : T \text{ computes } f} \mathrm{m}(T)$.

f	*AND*	*PARITY*
T		
$\mathrm{depth}(f)$	n	n
$\mathrm{size}(f)$	n	$2^n - 1$
$\mathrm{rank}(f)$	1	n
$G(f)$	1	n

Randomized measures:

1. Let $\mathcal{T}$ be a family of decision trees. It approximately computes f, if $\forall x, \underset{T \in_R \mathcal{T}}{\mathbb{P}}[T(x) = f(x)] \geq \frac{2}{3}$.
2. Let $\mathrm{m} \in \{\mathrm{depth}, \mathrm{size}, \mathrm{rank}, G\}$, $\mathrm{rm}(f) = \min\limits_{\mathcal{T} \text{ approx. computes } f} \max\limits_{T \in \mathcal{T}} \mathrm{m}(f)$.
3. Can make a big difference!
   - $\exists f : \mathrm{rdepth}(f) \ll \mathrm{depth}(f)$ [SW86;ABB+17;MRS18]

Results

Our results:

Results

Our results:

1. Guessing complexity equals rank.
   - Answers open question from [LL16].

Results

Our results:

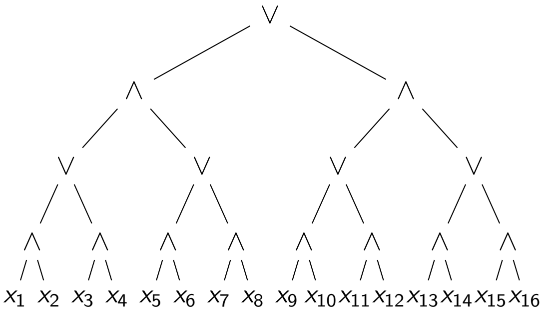
1. Guessing complexity equals rank.
   - Answers open question from [LL16].
2. Separation between rank and randomized rank.
   - $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.

Results

Our results:

1. Guessing complexity equals rank.
   - Answers open question from [LL16].
2. Separation between rank and randomized rank.
   - $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.

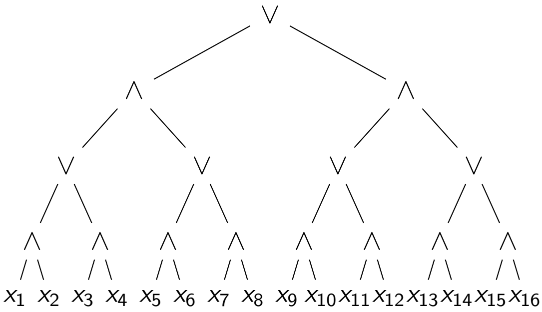
AND-OR tree



Our results:

1. Guessing complexity equals rank.
   - Answers open question from [LL16].
2. Separation between rank and randomized rank.
   - $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.

AND-OR tree



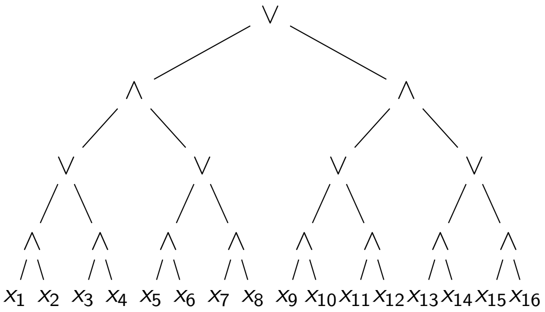
$$\begin{aligned} \mathrm{rrank}(f) &= \mathcal{O}(n^{0.753\ldots}) \text{ [SW86]}, \\ \mathrm{rank}(f) &= \Theta(n). \end{aligned}$$

Results

Our results:

1. Guessing complexity equals rank.
   - Answers open question from [LL16].
2. Separation between rank and randomized rank.
   - $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.
   - Proof via Prover-Delayer games. [PI00]

AND-OR tree

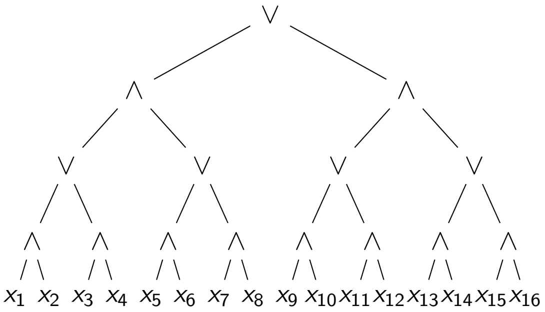


$$\begin{aligned} \mathrm{rrank}(f) &= \mathcal{O}(n^{0.753\ldots}) \text{ [SW86]}, \\ \mathrm{rank}(f) &= \Theta(n). \end{aligned}$$

Our results:

1. Guessing complexity equals rank.
    - Answers open question from [LL16].
2. Separation between rank and randomized rank.
    - $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.
    - Proof via Prover-Delayer games. [PI00]
3. Improve best-known construction for quantum query algorithms from decision trees.

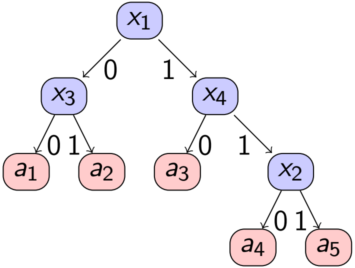
AND-OR tree



$$\mathrm{rrank}(f) = \mathcal{O}(n^{0.753\ldots}) \text{ [SW86]},$$
$$\mathrm{rank}(f) = \Theta(n).$$

Decision tree to quantum algorithm – overview

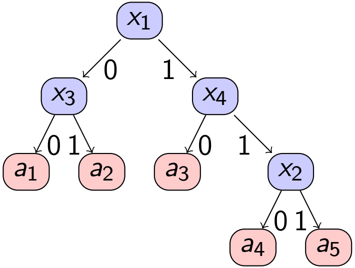
Goal: Take a decision tree T and construct a quantum query algorithm from it.



Goal: Take a decision tree T and construct a quantum query algorithm from it.

Prior work:

1. $\mathcal{O}(\sqrt{G(T)\mathrm{depth}(T)})$-query algorithm.

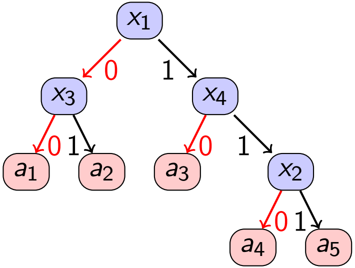


Decision tree to quantum algorithm – overview

Goal: Take a decision tree T and construct a quantum query algorithm from it.

Prior work:

1. $\mathcal{O}(\sqrt{G(T)\mathrm{depth}(T)})$-query algorithm.
   - Iteratively use minimum finding to find first red edge [LL16].

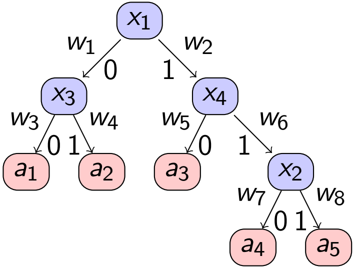


Decision tree to quantum algorithm – overview

Goal: Take a decision tree T and construct a quantum query algorithm from it.

Prior work:

1. $\mathcal{O}(\sqrt{G(T)\mathrm{depth}(T)})$-query algorithm.
   - Iteratively use minimum finding to find first red edge [LL16].
   - Direct span program construction [BT20].
     - Requires weight assignment to the edges.

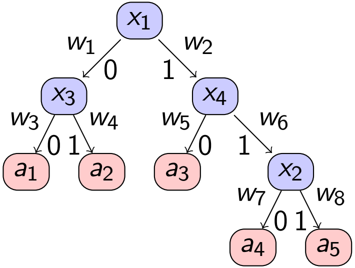


Decision tree to quantum algorithm – overview

Goal: Take a decision tree T and construct a quantum query algorithm from it.

Prior work:

1. $\mathcal{O}(\sqrt{G(T)\mathrm{depth}(T)})$-query algorithm.
   - Iteratively use minimum finding to find first red edge [LL16].
   - Direct span program construction [BT20].
     - Requires weight assignment to the edges.
     - *Open question:* better weight assigments?

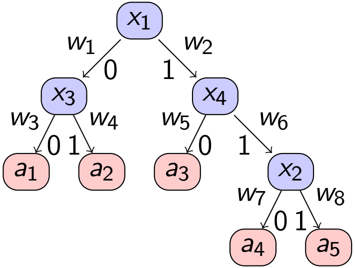


Decision tree to quantum algorithm – overview

Goal: Take a decision tree T and construct a quantum query algorithm from it.

Prior work:

1. $\mathcal{O}(\sqrt{G(T)\mathrm{depth}(T)})$-query algorithm.
   - Iteratively use minimum finding to find first red edge [LL16].
   - Direct span program construction [BT20].
     - Requires weight assignment to the edges.
     - *Open question:* better weight assigments?
   - Time-efficient implementation [BTT21].

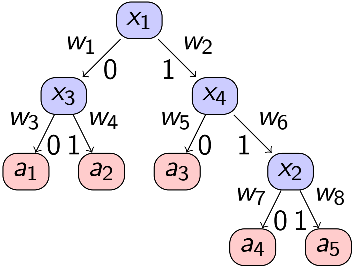


Decision tree to quantum algorithm – overview

Goal: Take a decision tree T and construct a quantum query algorithm from it.

Prior work:

1. $\mathcal{O}(\sqrt{G(T)\mathrm{depth}(T)})$-query algorithm.
   - Iteratively use minimum finding to find first red edge [LL16].
   - Direct span program construction [BT20].
     - Requires weight assignment to the edges.
     - *Open question:* better weight assigments?
   - Time-efficient implementation [BTT21].
2. Improved weights for the oracle identification problem [Tag21].

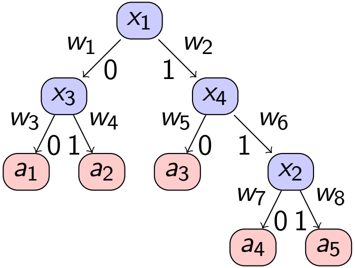


Decision tree to quantum algorithm – overview

Goal: Take a decision tree T and construct a quantum query algorithm from it.

Prior work:

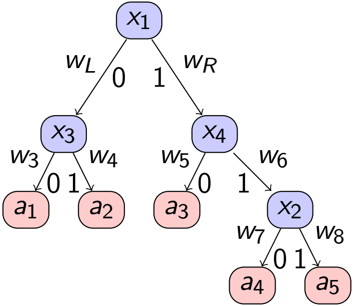
1. $\mathcal{O}(\sqrt{G(T)\mathrm{depth}(T)})$-query algorithm.
   - Iteratively use minimum finding to find first red edge [LL16].
   - Direct span program construction [BT20].
     - Requires weight assignment to the edges.
     - *Open question:* better weight assigments?
   - Time-efficient implementation [BTT21].
2. Improved weights for the oracle identification problem [Tag21].



Our contribution: we provide the optimal weight assignment.

Decision tree to quantum algorithm – optimal weight assignment

Goal: Take a decision tree and construct a quantum algorithm from it.

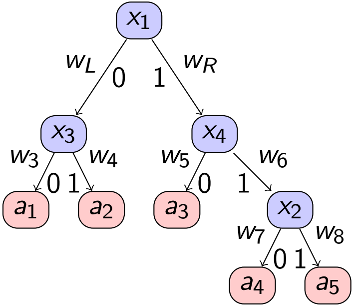


Decision tree to quantum algorithm – optimal weight assignment

Goal: Take a decision tree and construct a quantum algorithm from it.

1. Construction of [BT20]: Let
   - $W_+ = \max_P \{\sum_{e \in P} \frac{1}{w_e}\}$.
   - $W_- = \max_P \{\sum_{e \in \overline{P}} w_e\}$.
   - $C = \sqrt{W_+ W_-}$.

   $\Rightarrow \mathcal{O}(C)$-query algorithm.

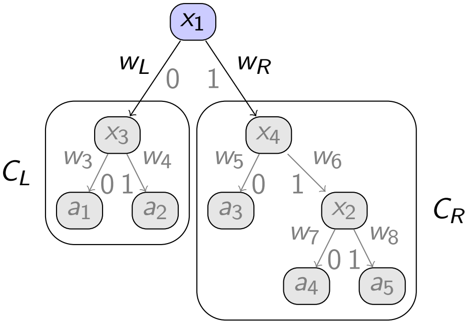


Decision tree to quantum algorithm – optimal weight assignment

Goal: Take a decision tree and construct a quantum algorithm from it.

1. Construction of [BT20]: Let
   - $W_+ = \max_P \{\sum_{e \in P} \frac{1}{w_e}\}$.
   - $W_- = \max_P \{\sum_{e \in \overline{P}} w_e\}$.
   - $C = \sqrt{W_+ W_-}$.

   $\Rightarrow \mathcal{O}(C)$-query algorithm.

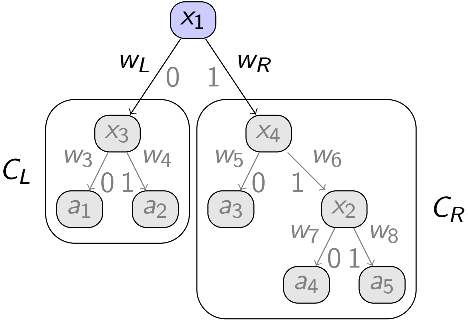


Decision tree to quantum algorithm – optimal weight assignment

Goal: Take a decision tree and construct a quantum algorithm from it.

1. Construction of [BT20]: Let
   - $W_+ = \max_P\{\sum_{e\in P} \frac{1}{w_e}\}$.
   - $W_- = \max_P\{\sum_{e\in \overline{P}} w_e\}$.
   - $C = \sqrt{W_+ W_-}$.

   $\Rightarrow \mathcal{O}(C)$-query algorithm.



$$w_R = \frac{1}{w_L} = \frac{C_L - C_R + \sqrt{(C_L - C_R)^2 + 4}}{2}.$$

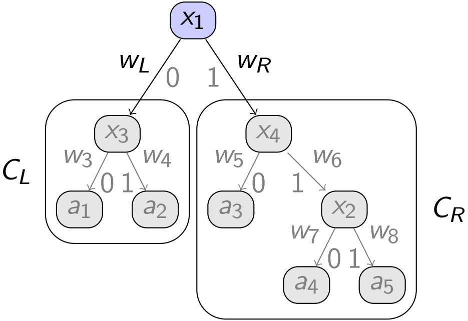
$$\Rightarrow C = \frac{C_L + C_R + \sqrt{(C_L - C_R)^2 + 4}}{2}.$$

Decision tree to quantum algorithm – optimal weight assignment

Goal: Take a decision tree and construct a quantum algorithm from it.

1. Construction of [BT20]: Let
   - $W_+ = \max_P\{\sum_{e\in P} \frac{1}{w_e}\}$.
   - $W_- = \max_P\{\sum_{e\in \overline{P}} w_e\}$.
   - $C = \sqrt{W_+ W_-}$.

   $\Rightarrow \mathcal{O}(C)$-query algorithm.
2. Optimal & constructive assignment.



$$w_R = \frac{1}{w_L} = \frac{C_L - C_R + \sqrt{(C_L - C_R)^2 + 4}}{2}.$$

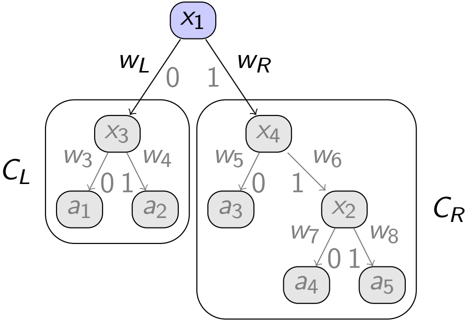
$$\Rightarrow C = \frac{C_L + C_R + \sqrt{(C_L - C_R)^2 + 4}}{2}.$$

Decision tree to quantum algorithm – optimal weight assignment

Goal: Take a decision tree and construct a quantum algorithm from it.

1. Construction of [BT20]: Let
   - $W_+ = \max_P \{\sum_{e \in P} \frac{1}{w_e}\}$.
   - $W_- = \max_P \{\sum_{e \in \overline{P}} w_e\}$.
   - $C = \sqrt{W_+ W_-}$.

   $\Rightarrow \mathcal{O}(C)$-query algorithm.
2. Optimal & constructive assignment.
3. $\Rightarrow \mathcal{O}(\sqrt{\mathrm{size}(T)})$-query algorithm.



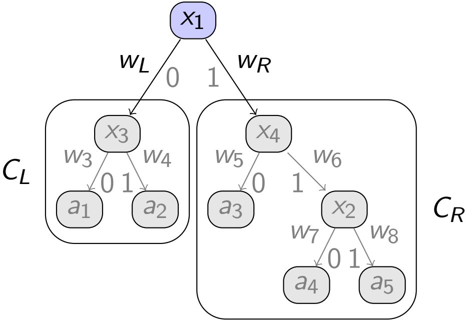
$$w_R = \frac{1}{w_L} = \frac{C_L - C_R + \sqrt{(C_L - C_R)^2 + 4}}{2}.$$
$$\Rightarrow C = \frac{C_L + C_R + \sqrt{(C_L - C_R)^2 + 4}}{2}.$$

Decision tree to quantum algorithm – optimal weight assignment

Goal: Take a decision tree and construct a quantum algorithm from it.

1. Construction of [BT20]: Let
   - $W_+ = \max_P \{\sum_{e \in P} \frac{1}{w_e}\}$.
   - $W_- = \max_P \{\sum_{e \in \overline{P}} w_e\}$.
   - $C = \sqrt{W_+ W_-}$.

   $\Rightarrow \mathcal{O}(C)$-query algorithm.
2. Optimal & constructive assignment.
3. $\Rightarrow \mathcal{O}(\sqrt{\mathrm{size}(T)})$-query algorithm.
4. $\exists T : \sqrt{\mathrm{size}(T)} \ll \sqrt{G(T)\mathrm{depth}(T)}$.



$$w_R = \frac{1}{w_L} = \frac{C_L - C_R + \sqrt{(C_L - C_R)^2 + 4}}{2}.$$

$$\Rightarrow C = \frac{C_L + C_R + \sqrt{(C_L - C_R)^2 + 4}}{2}.$$

Summary & open questions

Summary: Three results related to decision trees:

1. Guessing complexity equals rank – $G(T) = \mathrm{rank}(T)$.
2. Separation rank vs. randomized rank – $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.
3. Optimal weight assignment for span program construction.

Summary & open questions

Summary: Three results related to decision trees:

1. Guessing complexity equals rank – $G(T) = \mathrm{rank}(T)$.
2. Separation rank vs. randomized rank – $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.
3. Optimal weight assignment for span program construction.

Open questions:

1. Polynomial relation rank vs. randomized rank, $\mathrm{rank}(f) = \mathrm{poly}(\mathrm{rrank}(f))$?

Summary & open questions

Summary: Three results related to decision trees:

1. Guessing complexity equals rank – $G(T) = \mathrm{rank}(T)$.
2. Separation rank vs. randomized rank – $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.
3. Optimal weight assignment for span program construction.

Open questions:

1. Polynomial relation rank vs. randomized rank, $\mathrm{rank}(f) = \mathrm{poly}(\mathrm{rrank}(f))$?
2. Optimal weights for non-binary input case?

Summary & open questions

Summary: Three results related to decision trees:

1. Guessing complexity equals rank – $G(T) = \mathrm{rank}(T)$.
2. Separation rank vs. randomized rank – $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.
3. Optimal weight assignment for span program construction.

Open questions:

1. Polynomial relation rank vs. randomized rank, $\mathrm{rank}(f) = \mathrm{poly}(\mathrm{rrank}(f))$?
2. Optimal weights for non-binary input case?
3. Decision trees with more complicated queries, i.e. Hadamard queries?

Summary & open questions

Summary: Three results related to decision trees:

1. Guessing complexity equals rank – $G(T) = \mathrm{rank}(T)$.
2. Separation rank vs. randomized rank – $\exists f : \mathrm{rrank}(f) \ll \mathrm{rank}(f)$.
3. Optimal weight assignment for span program construction.

Open questions:

1. Polynomial relation rank vs. randomized rank, $\mathrm{rank}(f) = \mathrm{poly}(\mathrm{rrank}(f))$?
2. Optimal weights for non-binary input case?
3. Decision trees with more complicated queries, i.e. Hadamard queries?

Thanks for your attention!

`arjan@cwi.nl`

References

[ABB+17] Ambainis, Balodis, Belovs, Lee, Santha, Smotrovs. *Separations in query complexity based on pointer functions*, 2017.

[BT20] Beigi, Taghavi. *Quantum speed-up based on classical decision trees*, 2020.

[BTT21] Beigi, Taghavi, Tajdini. *Time and query optimal quantum algorithms based on decision trees*, 2021.

[LL16] Lin, Lin. *Upper bounds on quantum query complexity inspired by Elitzur-Vaidman bomb tester*, 2016.

[MRS18] Mukhopadhyay, Radhakrishnan, Sanyal. *Separation Between Deterministic and Randomized Query Complexity*, 2018.

[PI00] Pudlák, Impagliazzo. *A lower bound for DLL algorithms for k-sat*, 2000.

[SW86] Saks, Wigderson. *Probabilistic Boolean Decision Trees and the Complexity of Evaluating Game Trees*, 1986.

[Tag21] Taghavi. *Simplified quantum algorithm for the oracle identification problem*, 2021.