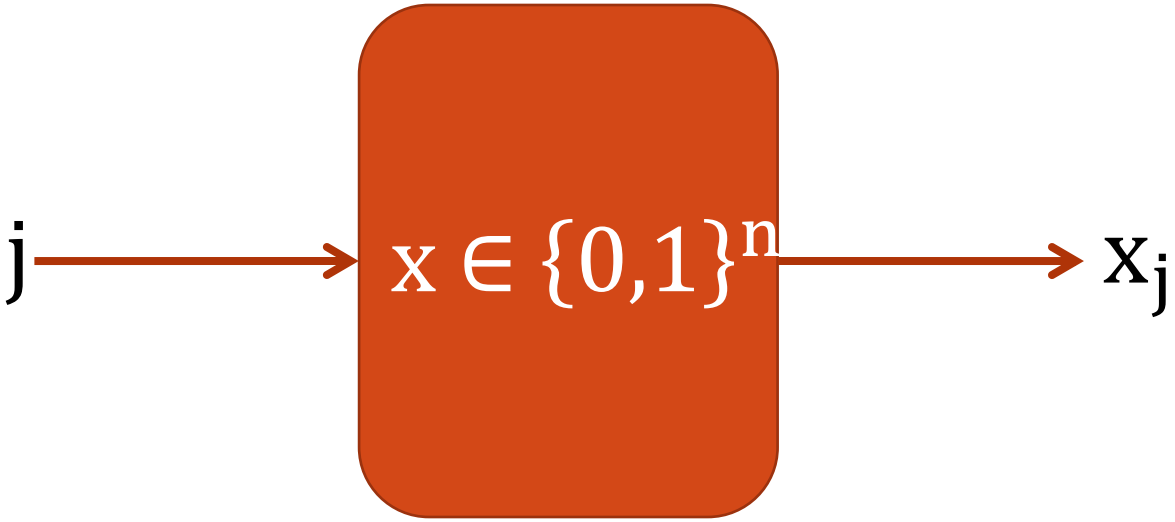
How to make quantum query algorithms

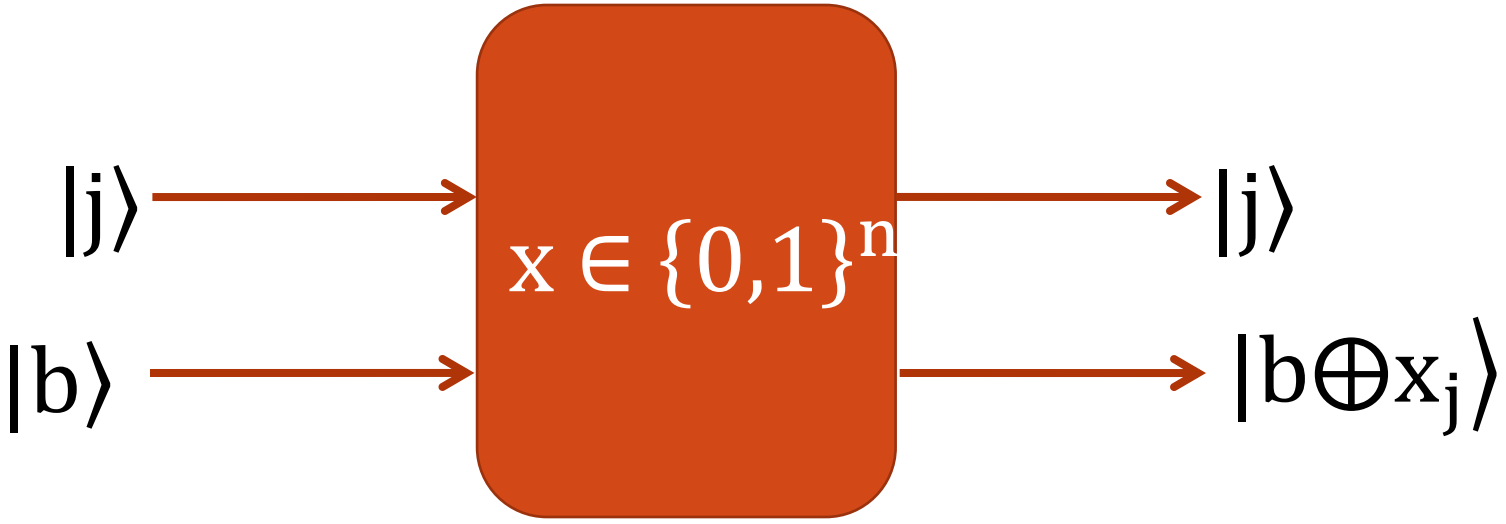
by
Rajat Mittal
IQC, University of Waterloo.

Given: Function $f: x \mapsto f(x)$



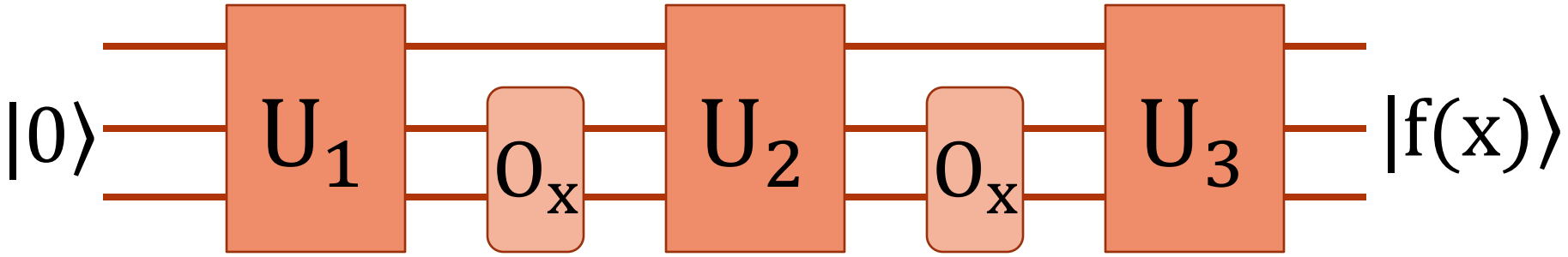
Least queries needed to evaluate $f(x)$

Given: Function $f: x \mapsto f(x)$



$Q(f) =$ Least quantum queries needed to evaluate $f(x)$

Query Algorithm



Unitary operators are free

Cost of algorithm = number of queries

Query algorithms

- Grover search

  unordered search in $O(\sqrt{n})$ queries
- AND/OR trees
- Element distinctness
- Graph collision
- Triangle finding

Many algorithms were given using quantum walks.

Lower bounds on quantum query complexity

Polynomial bound [BBCMW '01]:

- Separation [A '03]
- Element Distinctness [AS '04]
- Direct products [KŠdW '07]

Adversary bound [A '02]:

- SDP [BSS '03]
- Equivalence [ŠS '06]
- Multiplicative [Š '08]
- $\mathrm{Adv}^{\pm}(f)$ [HLŠ '07]

Lower bounds on quantum query complexity

Polynomial bound [BBCMW '01]:

- Separation [A '03]
- Element Distinctness [AS '04]
- Direct products [KŠdW '07]

Adversary bound [A '02]:

- SDP [BSS '03]
- Equivalence [ŠS '06]
- Multiplicative [Š '08]
- $\mathrm{Adv}^{\pm}(f)$ [HLŠ '07]

For any boolean function f:

$$Q(f) = \Theta(\mathrm{Adv}^{\pm}(f))$$

[R '09]

Vector set

Vector set : construction

Query algorithm for f : $D^n \longrightarrow \{0,1\}$

Construct vectors for

every element of D^n and $[n]$

	x	y	. . .	Z
1	$u_{x,1}$	$u_{y,1}$	.	$u_{z,1}$
.				
.	$u_{x,i}$	.	$u_{w,i}$	.
.				
n	$u_{x,n}$	$u_{y,n}$	.	$u_{z,n}$

Vector set : Example

Query algorithm for f : $\{0,1\}^n \longrightarrow \{0,1\}$

	000	001	. . .	111
1	$u_{000,1}$	$u_{001,1}$	.	$u_{111,1}$
.				
.	$u_{000,i}$	.	$u_{101,i}$	.
.				
n	$u_{000,n}$	$u_{001,n}$	.	$u_{111,n}$

Query algorithm for f : $D^n \longrightarrow \{0,1\}$

Follow constraint for $\forall\, x \in f^{-1}(0), y \in f^{-1}(1)$.

	x	y
1	$u_{x,1}$	$u_{y,1}$
.		
.	$u_{x,i}$	$u_{y,i}$
n	$u_{x,n}$	$u_{y,n}$

Query algorithm for f : $D^n \longrightarrow \{0,1\}$

Follow constraint for $\forall\, x \in f^{-1}(0), y \in f^{-1}(1)$.

	x	y	product
1	$u_{x,1}$	$u_{y,1}$	$< u_{x,1}, u_{y,1} > .(x_1 \neq y_1)$
.			+
.	$u_{x,i}$	$u_{y,i}$	$< u_{x,i}, u_{y,i} > .(x_i \neq y_i)$
.			+
n	$u_{x,n}$	$u_{y,n}$	$< u_{x,n}, u_{y,n} > .(x_n \neq y_n)$

$$\sum_{(x_i \neq y_i)} < u_{x,i}, u_{y,i} > = 1 \qquad \forall\, x, y.$$

The dual of adversary bound

- Vector set : solution of the dual of adversary bound

- The value of the solution :

$$\max_{z \in D^n} (\text{length of } u_z)$$

$$\max_{z \in D^n} \left(\sum_i \left\| u_{z,i} \right\|^2 \right)$$

- The value of best construction : Q(f)

Algorithmic applications

Q. Can we develop algorithms using solution of dual?

A. Not easy, because of the great number of constraints in SDP.

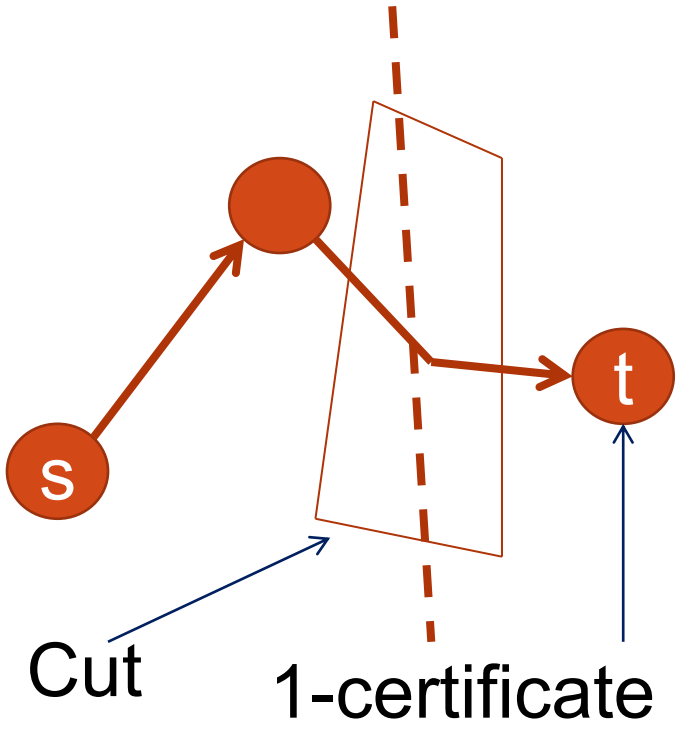
1. Formula evaluation [FGG07,RŠ08]
   (optimal formula evaluation algorithms
   for any read-once formula)

Algorithmic applications

Q. Can we develop algorithms using filtered factorization norm?

A. Not easy, because of the great number of constraints in SDP.

2. Learning graphs [Bel11]
(Element Distinctness and $n^{1.296}$ algorithm for Triangle Finding)



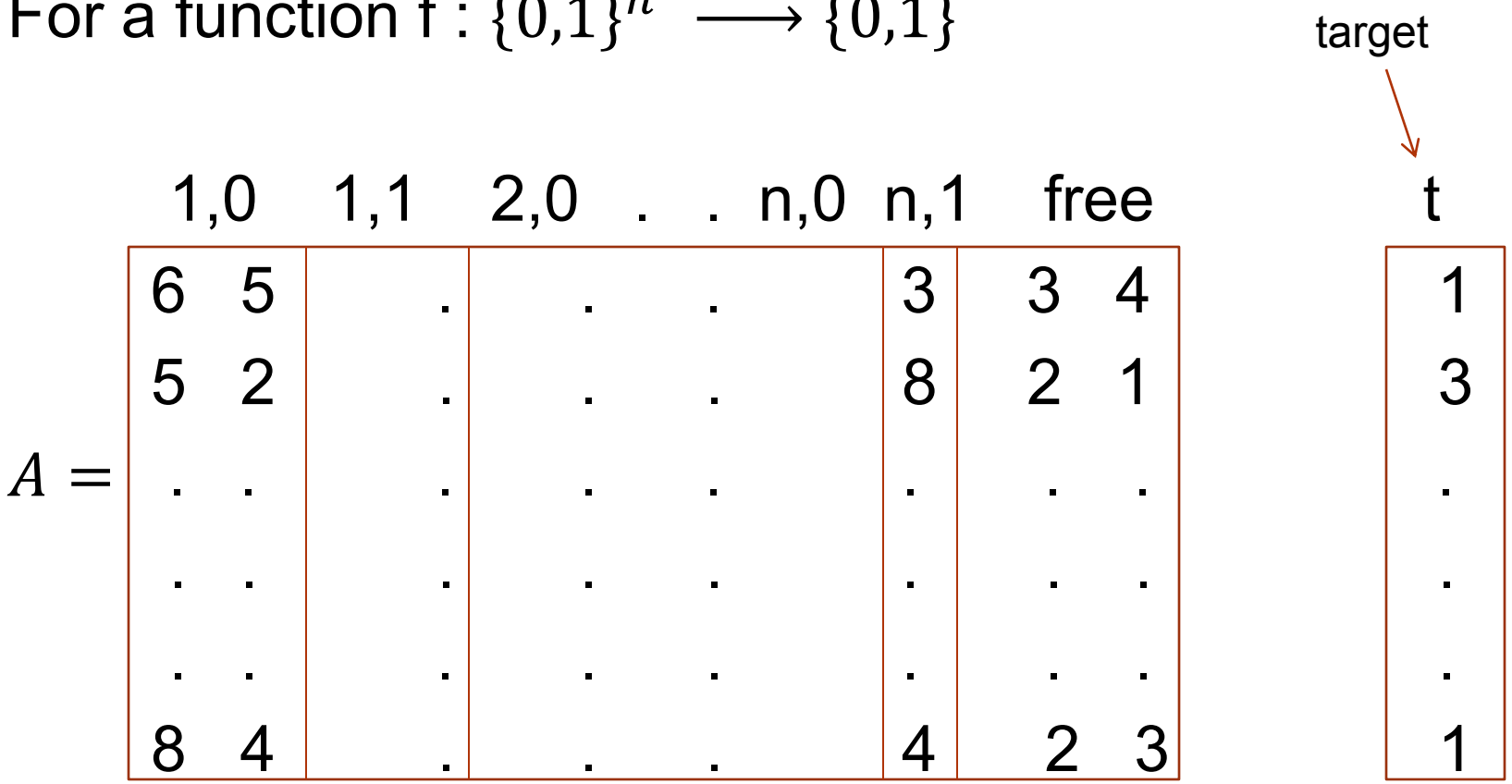
Outline

- Span programs
- Learning graphs
- Comparison
- Open problems

Span Programs

Span program

- For a function f : $\{0,1\}^n \longrightarrow \{0,1\}$



Span program

Useful vectors for $z \in \{0,1\}^n$:
free vectors + vectors in column j, z_j

target

$A =$

1,0	1,1	2,0	. .	n,0	n,1	free	t
6 5	.	.	.		3	3 4	1
5 2	.	.	.		8	2 1	3
. .	.	.	.		.	. .	.
. .	.	.	.		.	. .	.
. .	.	.	.		.	. .	.
8 4	.	.	.		4	2 3	1

Useful vectors for $z \in \{0,1\}^n$:

free vectors + vectors in column j, z_j

$f(y) = 1 \Rightarrow$

t : as linear combination of useful vectors for y

Witness: The coefficients of linear combination

Witness size : The length of witness vector

Useful vectors for $z \in \{0,1\}^n$:

free vectors + vectors in column j, z_j

$f(x) = 0 \Rightarrow$

t : NOT a linear combination of useful vectors for x

Witness: w: (<w,t> =1) & (<w,v> = 0, if v useful).

Witness size: The length of Aw.

Equivalence to dual solution

- Every span program can be converted to a canonical span program

  Fixed vector space for columns

  No free vectors

- Canonical span program is equivalent to a solution of dual adversary.

- Complexity of best span program is the query complexity of f.

Features

- Easy to manipulate span programs

  Complementation

  Composition

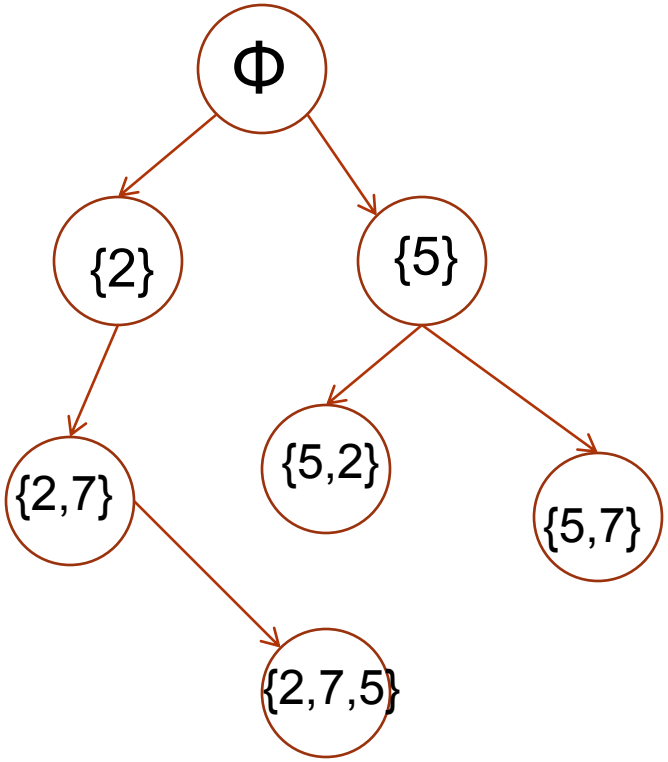
- Optimal formula evaluation algorithms

- These were used to show query algorithms using adversary bound.

Learning Graph

Learning graph

- For a function f : $D^n \longrightarrow \{0,1\}$
- Need to construct a graph

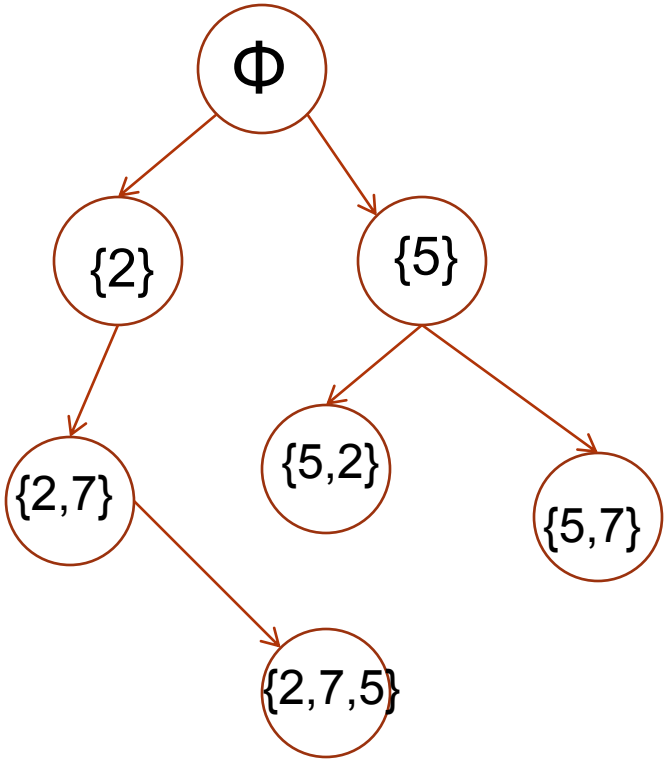


Vertex: $S \subseteq [n]$

Edge: $j \in [n]$

Learning graph

- For a function f : $D^n \longrightarrow \{0,1\}$
- Need to construct a graph



Edges:

$S \xrightarrow{j} S \cup \{j\}$ $\quad S \subseteq [n], j \in [n]$

Every edge has weight w_e

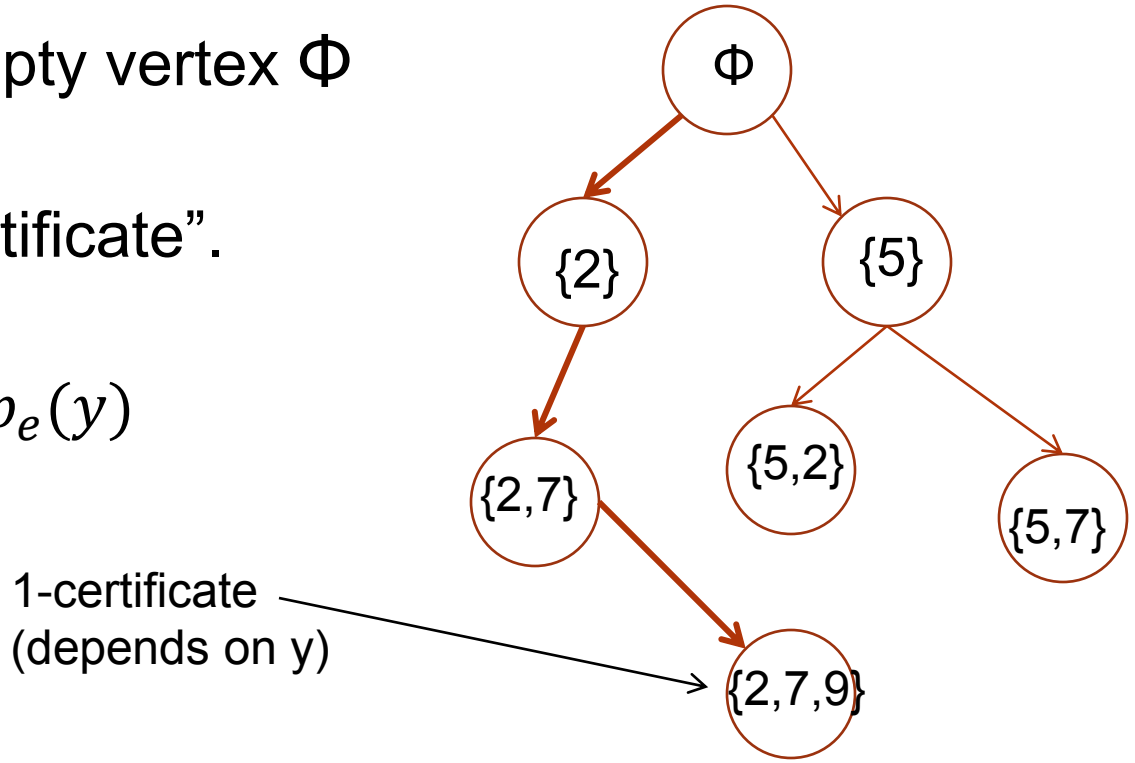
Flow for 1-input

- For every $y \in f^{-1}(1)$, there is a flow of value 1

Source: The empty vertex Φ

Sink: Any "1-certificate".

Flow in edge e: $p_e(y)$

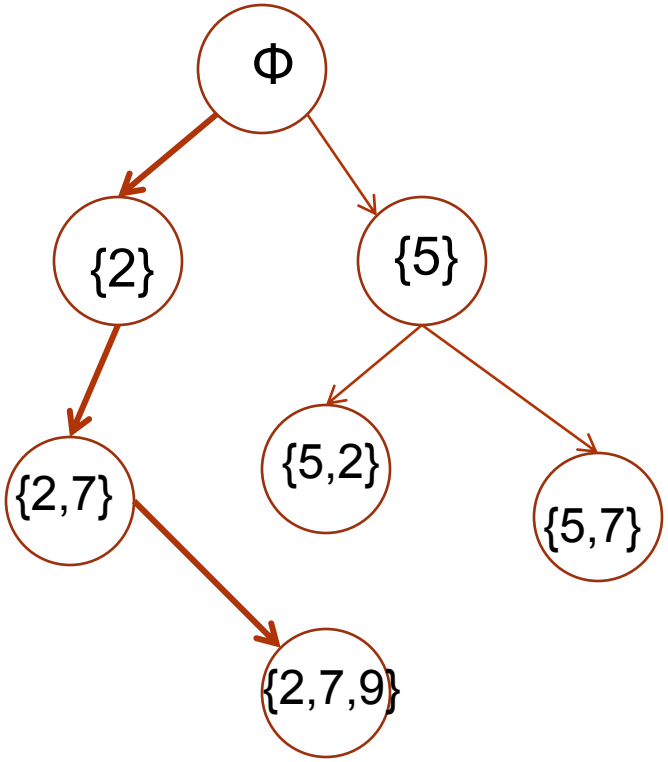


Complexity of learning graph

$$C^0 = \max_x \sum_e w_e$$

$$C^1 = \max_y \sum_e \frac{p_e(y)^2}{w_e}$$

$$C = \sqrt{C^0 C^1}$$



Reduction

- Convert learning graph into vector construction

For every edge $S \rightarrow S \cup \{j\}$:

need $2^{|S|}$ coordinates in $u_{z,j}$

	j, e, α (S) ← Assignment on S
u_x	$\sqrt{w_e}$
u_y	$\frac{p_e(y)}{\sqrt{w_e}}$

Constraint for dual adversary

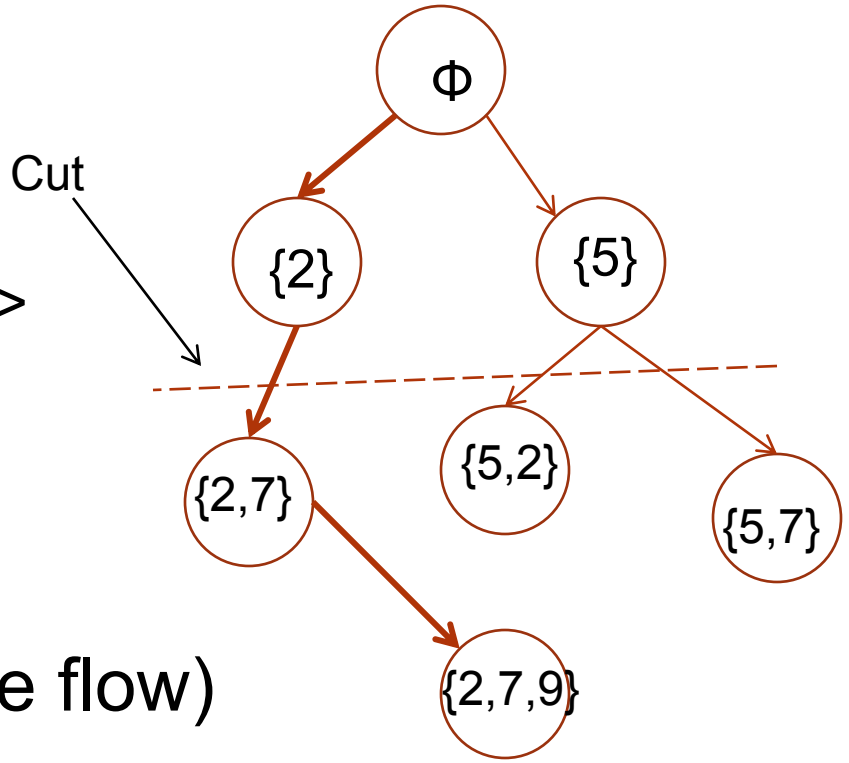
Constraint:

$$\sum_{(x_i \neq y_i)} < u_{x,i}, u_{y,i} >$$

$$= \sum_{e:cut} \frac{p_e(y)}{\sqrt{w_e}} \sqrt{w_e}$$

$$= \sum_{e:cut} p_e(y)$$

$$= 1 \quad \text{(value of the flow)}$$



Important results

- Element distinctness:

  Showed the previous bound of $O(n^{\frac{2}{3}})$

- Triangle finding:

  Improved the upper bound to $O(n^{1.296})$

- k-element distinctness:

  Improved under certain conditions

Limitations and improvement

- Certificate complexity barrier

  Learning graph complexity $\geq$ 1-certificate complexity

- Alphabet size barrier

  Only depends on certificate structure

- Improved learning graph : overcomes both barriers

  Gives better complexity for “OR of AND”

Comparison

Span Program

- Easy to manipulate
- Equivalent to dual
- ?????

Learning graph

- Can’t compose it well
- Weaker than dual
- Easy to construct

Solution of dual adversary

- $\forall\, x, y$: $\langle u_x | v_y \rangle =$ ...
- Symmetric
- Tight

Learning graph / Span programs

- $\forall\, x$: ...
- $\forall\, y$: ...
- Intuitive

Separation of constraints really help
Need more combinatorial constructions to make query algorithms

Open problems

Constructing dual solution

- Need other techniques to construct dual solution.

  Easy to manipulate

  Easy to construct

  Tight

- Dual of “positive” adversary

  $\sum_{(x_i \neq y_i)} < u_{x,i}, u_{y,i} > \geq 1$

  When is positive adversary tight ?

Other query algorithms

- Graph collision

  Important in triangle finding.

- Triangle finding

- k-element distinctness

Thank you