Improved learning graph based quantum algorithms for TRIANGLE and ASSOCIATIVITY

Miklos Santha

CNRS, Université Paris Diderot
and
Centre for Quantum Technologies, NUS, Singapore

Joint work with

T. Lee CQT, Singapore
and
F. Magniez CNRS, Paris

Query complexity

Let $f : \mathcal{D} \to E$ with $\mathcal{D} \subseteq [d]^n$. Often $d = 2$ and $E = \{0, 1\}$.

Query complexity: Number of input queries needed to evaluate f.

Computational models: Deterministic, randomized, quantum

The gap between the deterministic and quantum complexities

- can be exponential [Simon'97, Shor'97]: Period finding, [EHK'99]: Hidden Subgroup Problem
- [BBCMW'01]: is at most polynomial for total functions

Quantum query complexity

Theorem [HLS'07, R'11, LMRSSz'11]: Let $f : \mathcal{D} \to \{0,1\}$ with $\mathcal{D} \subseteq [d]^n$. Then

$$
\begin{aligned}
Q(f) = \underset{u_{x,i}}{\text{minimize}} \quad & \max_{x \in \mathcal{D}} \sum_{i \in [n]} \|u_{x,i}\|^2 \\
\text{subject to} \quad & \sum_{\substack{i \in [n] \\ x_i \neq y_i}} \langle u_{x,i} | u_{y,i} \rangle = 1 \text{ for all } f(x) \neq f(y) \ ,
\end{aligned}
$$

where $u_{x,i} \in \mathbb{R}^m$, for $x \in \mathcal{D}$ and $i \in [n]$.

LEARNING GRAPHS [Belovs'11]

A learning graph $\mathcal{G}$ for $f : \mathcal{D} \to \{0,1\}$ with $\mathcal{D} \subseteq [d]^n$ is

- rooted, weighted and directed acyclic graph
- vertices labeled by $S \subseteq [n]$, the root is labeled by $\emptyset$
- An edge is $e = (S, S \cup \{i\})$ for $S \subseteq [n]$ and $i \notin S$

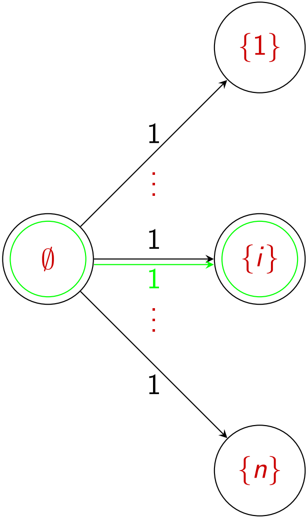
We must specify

- For every edge e its weight $w(e) \in \mathbb{R}^+$
- For every input $y \in f^{-1}(1)$ a unit flow from the root $\emptyset$, where all sinks are labeled by sets S containing a 1-certificate for y. The flow through edge e on y is denoted $p_y(e)$

We can also authorize edges $e = (S, S \cup S')$ for $S \cap S' = \emptyset$
Then by definition the length $\ell(e)$ of the edge e is $|S'|$

Learning graph for the OR function

Unit flow for the positive input $x = 0\ldots010\ldots0$, where $x_i = 1$.



Learning graphs

Learning graph complexity $\mathcal{LG}(f)$ of f

- Negative complexity of $\mathcal{G}$:

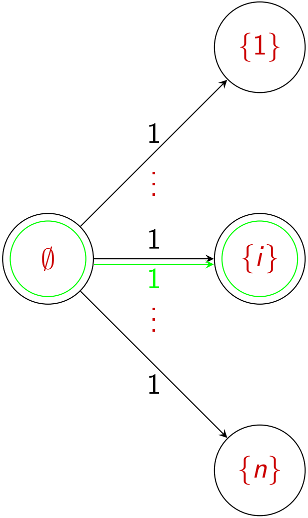
$$C_0(\mathcal{G}) = \sum_{e \in \mathcal{G}} \ell(e) w(e)$$

- Positive complexity of $\mathcal{G}$:

$$C_1(\mathcal{G}) = \max_{y \in f^{-1}(1)} \left(\sum_{e \in \mathcal{G}} \ell(e) \frac{p_y(e)^2}{w(e)} \right).$$

- Complexity of $\mathcal{G}$: $C(\mathcal{G}) = \sqrt{C_0(\mathcal{G}) C_1(\mathcal{G})}$.
- $\mathcal{LG}(f) = \min C(\mathcal{G})$ where $\mathcal{G}$ is a learning graph for f

Learning graph for the OR function



$C_0(\mathcal{G}) = n$ $\quad C_1(\mathcal{G}) = 1$ $\quad C(\mathcal{G}) = \sqrt{n}$ $\quad \mathcal{LG}(\mathrm{OR}) = O(\sqrt{n})$

Learning graph is a relaxation

Theorem[Belovs'11]: $Q(f) \leq \mathcal{LG}(f)$.

Proof Let $E_i = \{e = (S, S \cup \{i\}) : i \notin S\}$;

$$u_{x,i} = \sum_{e \in E_i} \sqrt{w(e)}|S\rangle|x_S\rangle \quad \text{for} \quad f(x) = 0$$

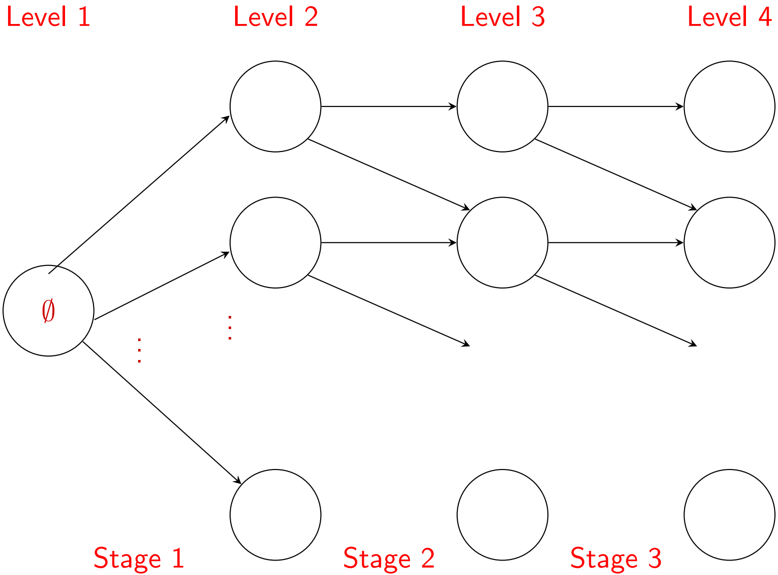
$$u_{y,i} = \sum_{e \in E_i} \frac{p_y(e)}{\sqrt{w(e)}}|S\rangle|y_S\rangle \quad \text{for} \quad f(y) = 1$$

Then $\sum_{i:x_i \neq y_i} \langle u_{x,i}|u_{y,i}\rangle$ is the flow through the cut

$$(\{S : S \subseteq I\}\,,\ \{S : I \subsetneq S\})$$

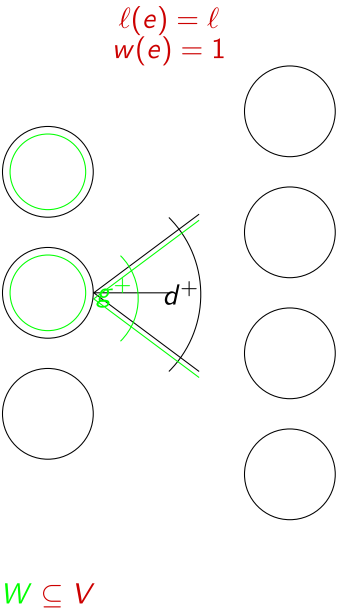
where $\quad I = \{i : x_i = y_i\}$

Learning graph in stages



Fact: Complexity of constant stages = sum of the complexities

Complexity of a stage



Complexity $= \ell \sqrt{\frac{|V|}{|W|}} \sqrt{\frac{d^+}{g^+}}$

$\frac{|V|}{|W|}$ = vertex ratio

$\frac{d+}{g+}$ = out-degree ratio

Several stages

$$C_i = \ell_i \sqrt{\frac{|V_i|}{|W_i|}} \sqrt{\frac{d_i^+}{g_i^+}}$$

The out-degre ratio is local to the stage, but the vertex ratio depends on the past

Evolution of the vertex ratio with constant in-degrees d^- and g^-:

$$|V_i| = |V_{i-1}| \frac{d_{i-1}^+}{d_i^-}; \qquad |W_i| = |W_{i-1}| \frac{g_{i-1}^+}{g_i^-}$$

$$\frac{|V_i|}{|W_i|} = \left(\frac{|V_{i-1}|}{|W_{i-1}|} \times \frac{d_{i-1}^+}{g_{i-1}^+} \right) : \frac{d_i^-}{g_i^-}$$

The in-degree ratio $\frac{d_i^-}{g_i^-}$ decreases the complexity

It depends on some well chosen database

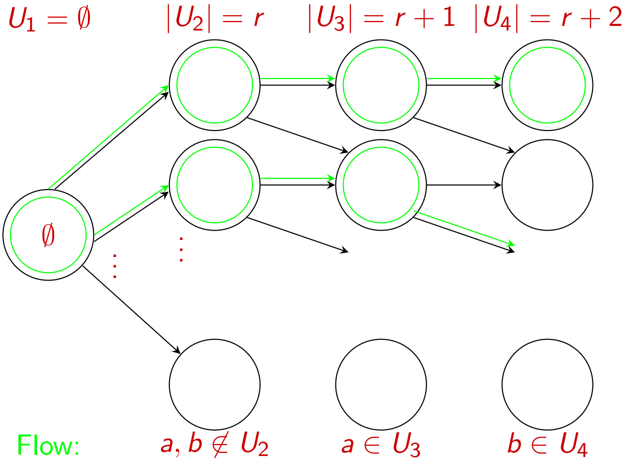
Example: Element distinctness

Element Distinctness
Oracle Input: A function $f : [n] \to [n]$.
Question: Is there a pair of distinct elements $i, j \in [n]$ such that $f(i) = f(j)$?

For every positive instance f we fix $a \neq b$ such that $f(a) = f(b)$.

Complexity of ED

$V_i = \{U_i \subseteq [n] : ...\}$



Vertex ratio at stage 3: $\frac{|V_2|}{|W_2|} = 1;\quad \frac{|V_3|}{|W_3|} = \frac{d_2^+}{g_2^+} : \frac{d_3^-}{g_3^-} = \frac{n}{1} : \frac{r}{1} = \frac{n}{r}$

Complexity: $C(\mathcal{G}) = C_1 + C_3 = r + n/\sqrt{r} = n^{2/3}$

Triangle and Subgraph$_H$

Triangle
Oracle Input: The adjacency matrix $A : \binom{n}{2} \to \{0, 1\}$ of a graph G on vertex set $[n]$.
Question: Is there a triangle in G?

Let $H = ([k], E(H))$ be some fixed k-vertex graph.

Subgraph$_H$
Oracle Input: The adjacency matrix $A : \binom{n}{2} \to \{0, 1\}$ of a graph G on vertex set $[n]$.
Question: Is there a copy of H in G?

Learning graph based algorithms

[Magniez-Santha-Szegedy'03]: $Q(\textsc{Triangle}) = O(n^{1.3})$
Database is the complete graph

[Belovs'11]: $Q(\textsc{Triangle}) = O(n^{35/27}) = O(n^{1.296})$
Sparsification: maintain just a random database where edge slots are chosen with probability $0 \leq s \leq 1$.

[Zhu'11, Lee-Magniez-Santha'11]: $Q(\textsc{Subgraph}_H) = O(n^{2-2/k-t})$, where $t = t(k, m, d) > 0$. Random database is the union of regular bipartite graphs reflecting the structure of the subgraph

[Belovs'12]: $Q(k\text{-}\textsc{Distinctness}) = O\left(n^{1-\frac{2^{k-2}}{2^k-1}}\right)$
More general learning graph: It depends also on the value of the queried variables

Our algorithms

- $Q(\textsc{Triangle}) = O(n^{9/7}) = O(n^{1.285})$
- Generalized algorithm for $\textsc{Subgraph}_H$
- $Q(\textsc{Associativity}) = O(n^{10/7}) = O(n^{1.428})$

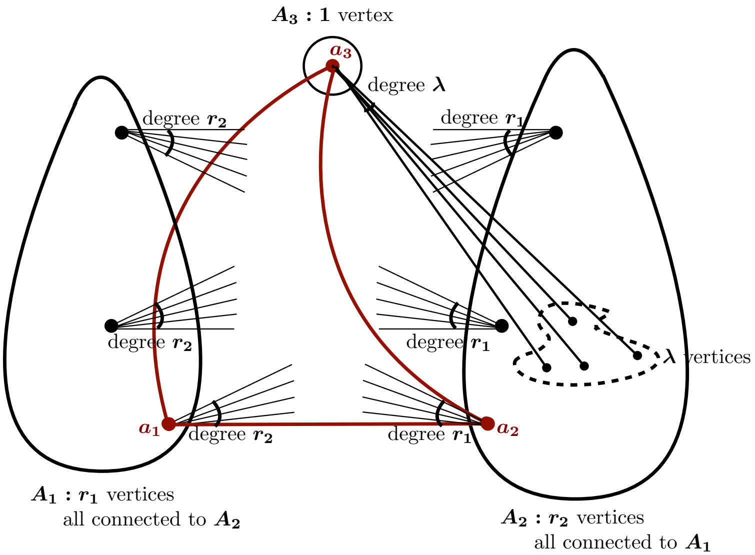
Triangle: The algorithm

For every positive instance A we fix three vertices a_1, a_2, a_3 such that they form a triangle

1. Setup: Load a complete bipartite graph between A_1 and A_2 of respective cardinality $r_1 = n^{4/7}$ and $r_2 = n^{5/7}$
2. Load a_1: Add a_1 to A_1 and connect it to all A_2
3. Load a_2: Add a_2 to A_2 and connect it to all A_1
4. Load a_3: Pick a_3 and connect it with $\lambda = n^{3/7}$ edges to A_2
5. Load $\{a_2, a_3\}$
6. Load $\{a_1, a_3\}$

Vertex sets in the bipartite graphs database can be unbalanced

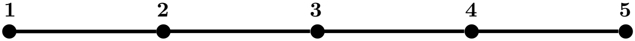
Triangle: The algorithm



Abstract language for detecting subgraphs

Let $H = ([k], E(H))$ be some fixed k-vertex graph

Example: 4-Path



Loading schedule: sequence $S = s_1 s_2 \ldots s_{k+m}$ which enumerates all vertices and edges of H.
Example: $S = [1, 2, 4, 3, (2, 1), (2, 3), (3, 4), 5, (5, 4)]$.

L-graph vertices: regular k-partite graphs with classes $A_1, \ldots, A_k$, and bipartite graphs E_{ij} between A_i and A_j for $\{i, j\} \in E(H)$.

Parameters: Set sizes $\{r_i\}$ and vertex degrees $\{d_{ij}\}$

Example: $r_1 = n, r_2 = n^{4/7}, r_3 = n^{6/7}, r_4 = n^{5/7}, r_5 = 1$;
$d_{21} = n^{6/7}, d_{23} = n^{6/7}, d_{34} = n^{5/7}, d_{54} = 1$.

Abstract language for detecting subgraphs

Theorem: There is an explicit function ϕ such that

$$\mathcal{LG}(\textsc{Subgraph}_H) \le \phi(S, \{r_i\}, \{d_{ij}\}).$$

Example: $\mathcal{LG}(4\text{-}\textsc{Path}) = O(n^{10/7})$

Best parameters can be found by linear programming:
`https://github.com/troyjlee/learning_graph_lp`

Theorem is extendable to:

- H is directed with possible self-loops
- Constant number of 1-certificates instead of just one
- Functions on labeled graphs:

  Let $f : [n]^{n\times n} \to \{0, 1\}$ be such that all minimal 1-certificate graphs are isomorphic to a fixed graph H. Then

  $$\mathcal{LG}(f) \le \mathcal{LG}(\textsc{Subgraph}_H)$$

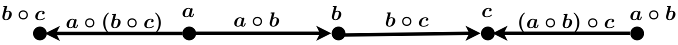
ASSOCIATIVITY

Oracle Input: Operation $\circ : [n] \times [n] \to [n]$
Question: $\exists$ a triple (a, b, c) such that $(a \circ b) \circ c \neq a \circ (b \circ c)$?

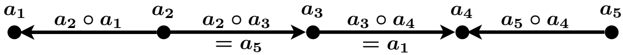
Grover search: $Q(\text{ASSOCIATIVITY}) = O(n^{3/2})$

Theorem: $Q(\text{ASSOC}) = O(n^{10/7}) = O(n^{1.428})$

Proof

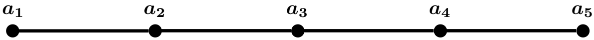


Certificate $\iff$ $a \circ (b \circ c) \neq (a \circ b) \circ c$



Certificate $\iff$ $(a_2 \circ a_3 = a_5,\ a_3 \circ a_4 = a_1$ and $a_2 \circ a_1 \neq a_5 \circ a_4)$

Certificate graph:



$Q(\text{ASSOC}) \leq \mathcal{LG}(\text{ASSOC}) \leq \mathcal{LG}(\text{4-PATH}) = O(n^{10/7}))$

Conclusion

Recent results:

- [Jeffery, Kothari, Magniez'12]: Can simulate our algorithms by quantum walks
- [Belovs, Rosmanis'12]: Our triangle algorithm is the best non-adaptive learning graph algorithm

Open problems: Complexity of

- Triangle
- Graph Collision
- k-Distinctness
- Associativity
- Matrix Product Verification