Learning-graph-based quantum algorithm for k-distinctness

Alexander Belov
University of Latvia

January 22, 2013
QIP 2013, Beijing, China

IEGULDĪJUMS TAVĀ NĀKOTNĒ

This work has been supported by the European Social Fund within the project "Support for Doctoral Studies at University of Latvia"

Introduction

Bounded 1-certificate complexity

We saw (the first talk) that for a function without any structure, e.g.,

k-sum problem:

Given $x_1, \ldots, x_n \in [q]$, detect whether there exist pairwise distinct $a_1, \ldots, a_k$ such that $x_{a_1} + x_{a_2} + \cdots + x_{a_k}$ is divisible by q.

quantum walk on the Johnson graph gives $O(n^{k/(k+1)})$ queries, and this is optimal.

Structure

In Miklos Santha's talk, we saw that if there is additional structure (not all certificate positions are allowed), we can do better, e.g.:

Triangle problem:

Given $x_{i,j} \in \{0,1\}$, with $1 \leq i < j \leq n$, detect whether there exist $1 \leq a < b < c \leq n$ such that $x_{a,b} = x_{a,c} = x_{b,c} = 1$.

Can be done with learning graphs in $O(n^{9/7})$ quantum queries.
Better than $O(n^{3/2})$ that would be possible without the structure.

Main Question

Simplification II: Only consider the *positions* of certificates inside the input string.

Main Question

~~**Simplification II:** Only consider the *positions* of certificates inside the input string.~~

What if we consider the values of the variables as well?

Plus: We can pursue consistent certificates, and drop inconsistent ones, thus, reducing the complexity.

Minus: Greater diversity makes the algorithm harder to analyze.

Values

Considering values, we certainly can do better:

k-threshold problem:

Given $x_1, \ldots, x_n \in \{0,1\}$, detect whether $\sum_{i=1}^{n} x_i \geq k$.

- Can be easily solved in $O(\sqrt{n})$ queries using Grover search.

Values

Considering values, we certainly can do better:

k-threshold problem:

Given $x_1, \ldots, x_n \in \{0, 1\}$, detect whether $\sum_{i=1}^{n} x_i \geq k$.

- Can be easily solved in $O(\sqrt{n})$ queries using Grover search.
- Well... it's too simple.

k-distinctness problem

We arrive at our main problem:

k-distinctness problem:

Given $x_1, \ldots, x_n \in [q]$, detect whether there exist $a_1, \ldots, a_k$, all distinct, such that $x_{a_1} = x_{a_2} = \cdots = x_{a_k}$.

- Quantum walk algorithm solving the problem in $O(n^{k/(k+1)})$ queries.
- Best known lower bound is $\Omega(n^{2/3})$.

k-distinctness problem

We arrive at our main problem:

k-distinctness problem:

Given $x_1, \ldots, x_n \in [q]$, detect whether there exist $a_1, \ldots, a_k$, all distinct, such that $x_{a_1} = x_{a_2} = \cdots = x_{a_k}$.

- Quantum walk algorithm solving the problem in $O(n^{k/(k+1)})$ queries.
- Best known lower bound is $\Omega(n^{2/3})$.

- We developed a quantum algorithm with query complexity

$$O\left(n^{1-2^{k-2}/(2^k-1)}\right) = o(n^{3/4}).$$

Pursuing consistent certificates

How does it look like?

Similarly as in Miklos Santha's talk for Element Distinctness.

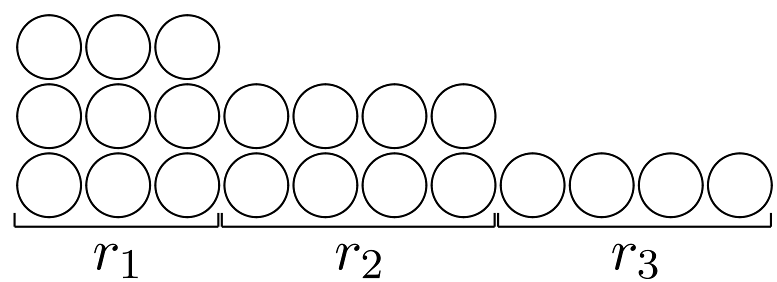
Let $a_1, \ldots, a_k$ be a 1-certificate in the input.

The last k steps in the learning graph are as on the right:

$\vdots$
Load a_1
Load a_2
$\vdots$
Load a_k

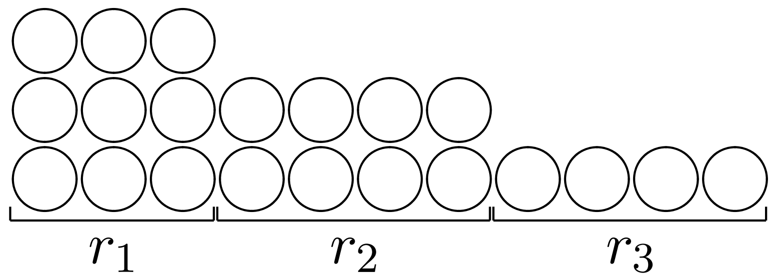
Assume before that the vertices of the learning graphs ($\subseteq [n]$) contain

r_1 unique elements, r_2 pairs of equal elements, …, r_{k-1} $(k-1)$-tuples of equal elements.



How does it look like?

Assume before that the vertices of the learning graphs ($\subseteq [n]$) contain



The complexity of loading $a_1, \ldots, a_k$ is $O(n/\sqrt{\min\{r_1, \ldots, r_{k-1}\}})$.

Proof. As for element distinctness: When a_i is loaded, $(i-1)$-tuple of equal elements $\{a_1, \ldots, a_{i-1}\}$ is hidden among $r_{i-1}+1$ such tuples in a vertex of the learning graph. □

Preparation of the state

In the quantum walk on the Johnson graph algorithm, $S \subseteq [n]$ is chosen uniformly at random from subsets of size r.
Thus, r_{k-1} is very small: $O(n \cdot r^{k-1}/n^{k-1})$.

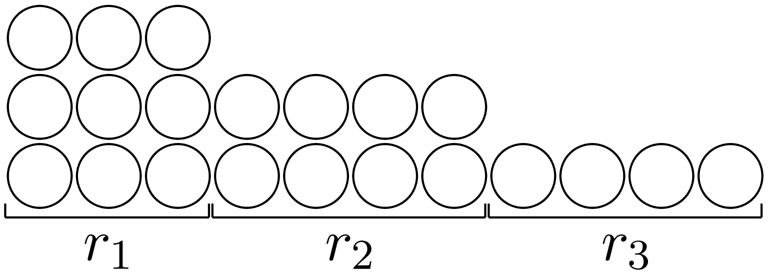
Using the values, we can "distill" subsets containing large number of large tuples of equal elements.

Preparation of the state

In the quantum walk on the Johnson graph algorithm, $S \subseteq [n]$ is chosen uniformly at random from subsets of size r.
Thus, r_{k-1} is very small: $O(n \cdot r^{k-1}/n^{k-1})$.

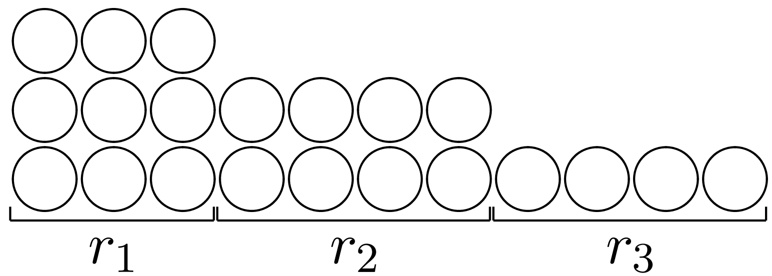
Using the values, we can "distill" subsets containing large number of large tuples of equal elements.

Related Question
What is the complexity of preparing the uniform superposition over all $S \subseteq [n]$ of the form



Preparation of the state

Preparation of uniform superposition over all $S \subseteq [n]$ that contain



Tentative Plan

1.	Start with the uniform superposition of $(r_1 + \cdots + r_{k-1})$-subsets.
2.	Find $r_2 + \cdots + r_{k-1}$ elements equal to elements in the current set.
3.	Find $r_3 + \cdots + r_{k-1}$ elements equal to two elements in the current set.
	$\vdots$
$k-1$.	Find r_{k-1} elements equal to $k-2$ elements in the current set.

Preparation of the state

1.	Start with the uniform superposition of $(r_1 + \cdots + r_{k-1})$-subsets.
2.	Find $r_2 + \cdots + r_{k-1}$ elements equal to elements in the current set.
3.	Find $r_3 + \cdots + r_{k-1}$ elements equal to two elements in the current set.
	$\vdots$
$k-1$.	Find r_{k-1} elements equal to $k-2$ elements in the current set.

We may assume there is unique k-tuple of equal elements in any positive input.
We may assume there are $\Omega(n)$ $(k-1)$-tuples of equal elements.
Assume also $r_1 > r_2 > \cdots > r_{k-1}$.

Preparation of the state

1.	Start with the uniform superposition of $(r_1 + \cdots + r_{k-1})$-subsets.
2.	Find $r_2 + \cdots + r_{k-1}$ elements equal to elements in the current set.
3.	Find $r_3 + \cdots + r_{k-1}$ elements equal to two elements in the current set.
	$\vdots$
$k-1$.	Find r_{k-1} elements equal to $k-2$ elements in the current set.

We may assume there is unique k-tuple of equal elements in any positive input.
We may assume there are $\Omega(n)$ $(k-1)$-tuples of equal elements.
Assume also $r_1 > r_2 > \cdots > r_{k-1}$.
Then, complexity of preparing the state is:

$$r_1 + r_2\sqrt{\frac{n}{r_1}} + r_3\sqrt{\frac{n}{r_2}} + \cdots + r_{k-1}\sqrt{\frac{n}{r_{k-2}}}.$$

Total complexity

Assume $r_1 > r_2 > \cdots > r_{k-1}$.
Complexity of preparing the uniform superposition is:

$$r_1 + r_2\sqrt{\frac{n}{r_1}} + r_3\sqrt{\frac{n}{r_2}} + \cdots + r_{k-1}\sqrt{\frac{n}{r_{k-2}}}.$$

Complexity of the final stage

$$n/\sqrt{r_{k-1}}.$$

Total complexity is optimized to

$$O\left(n^{1-2^{k-2}/(2^k-1)}\right) = o(n^{3/4}).$$

Diversity

Preparation of the state

1.	Start with the uniform superposition of $(r_1 + \cdots + r_{k-1})$-subsets.
2.	Find $r_2 + \cdots + r_{k-1}$ elements equal to elements in the current set.
3.	Find $r_3 + \cdots + r_{k-1}$ elements equal to two elements in the current set.
	$\vdots$
$k-1$.	Find r_{k-1} elements equal to $k-2$ elements in the current set.

Preparation of the state

1.	Start with the uniform superposition of $(r_1 + \cdots + r_{k-1})$-subsets.
2.	Find $r_2 + \cdots + r_{k-1}$ elements equal to elements in the current set.
3.	Find $r_3 + \cdots + r_{k-1}$ elements equal to two elements in the current set.
	$\vdots$
$k-1$.	Find r_{k-1} elements equal to $k-2$ elements in the current set.

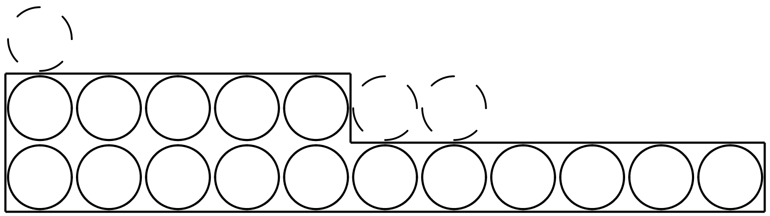
This algorithm does not generate the uniform superposition, nor a state close to it!

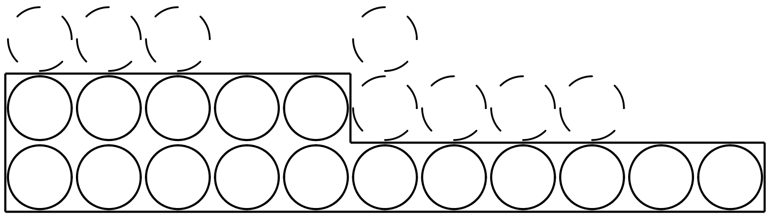
Preparation of the state

Assume both states have amplitudes α.

Perform Grover search for an element making a pair with an element in S.

Preparation of the state





Assume both states have amplitudes α.

Perform Grover search for an element making a pair with an element in S.

Assume the Grover search works perfectly for both subsets.
Then the amplitude is subdivided into:

$\alpha/\sqrt{2}$ $\qquad$ $\alpha/\sqrt{5}$

This accumulates with each step, and we get an exponential bias.

Summary

Summary

- We saw gains and losses of using values of the variables.
- These problems can be solved for k-distinctness, but I will not go into the detail.

Open Problem

- Obtain a similar framework for these types of problems, as it was done in the first presentation (learning graphs).
- Prove matching lower bound for k-distinctness.

Thank you!