Separations in Query Complexity
Based on Pointer Functions

Alexander Belov
CWI

Joint work with: Andris Ambainis, Kaspars Balodis, Troy Lee, Miklos Santha, and Juris Smotrovs
Introduction
**Computation Models**

- **Introduction**
- **Computation Models**
- **Separtions**
- **A Previous Result**
- **Our Main Results**
- ** Göös-Pitassi-Watson**
- **Our Modifications**
- **$R_1$ versus $R_0$**
- **$R_0$ versus $D$**
- **Conclusion**

### $D$: Deterministic (Decision Tree)

![Decision Tree Diagram]

- **Complexity**
  - **on input:** Number of queries (length of the path) 2 or 3
  - **in total:** Worst input (depth of the tree) 3
\( \mathcal{D} \): Deterministic (Decision Tree)

\( \mathcal{R} \): Randomized (Probability distribution on decision trees)

**Complexity**

- **on input:** Expected number of queries 2 or \( \frac{8}{3} \)
- **in total:** Worst input \( \frac{8}{3} \)
$D$: Deterministic (Decision Tree)

$R$: Randomized (Probability distribution on decision trees)

$R_0$: Zero-error (Las Vegas)
- always outputs a correct output

$R_1$: One-sided error
- always rejects a negative input
- accepts a positive input with probability $\geq \frac{1}{2}$
  (or vice versa)

$R_2$: Bounded-error (Monte Carlo)
- rejects a negative input with probability $\geq \frac{2}{3}$
- accepts a positive input with probability $\geq \frac{2}{3}$
Computation Models

Introduction
Computation Models
Separations
A Previous Result
Our Main Results
Göös-Pitassi-Watson
Our Modifications
R₁ versus R₀
R₀ versus D
Conclusion

\[D:\text{ Deterministic (Decision Tree)}\]
\[R:\text{ Randomized (Probability distribution on decision trees)}\]
\[R₀:\text{ Zero-error (Las Vegas)}\]
\[R₁:\text{ One-sided error}\]
\[R₂:\text{ Bounded-error (Monte Carlo)}\]
\[Q:\text{ Quantum bounded-error}\]
\[Q_E:\text{ Quantum exact}\]
Easy for **partial** functions
Easy for **partial** functions

**Example:** Deutsch-Jozsa problem (almost)

- **Reject** iff all input variables are zeroes
  
  \[
  \begin{array}{cccccccc}
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  \end{array}
  \]

- **Accept** iff exactly half of the variables are ones
  
  \[
  \begin{array}{cccccccc}
  0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
  \end{array}
  \]
Easy for **partial** functions

**Example:** Deutsch-Jozsa problem (almost)

- **Reject** iff all input variables are zeroes
  
  $\begin{array}{cccccccccc}
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  \end{array}$

- **Accept** iff exactly half of the variables are ones

  $\begin{array}{cccccccccccc}
  0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
  \end{array}$

  $R_1 = 1$
Easy for **partial** functions

**Example:** Deutsch-Jozsa problem (almost)

- **Reject** iff all input variables are zeroes

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

- **Accept** iff exactly half of the variables are ones

\[
\begin{array}{cccccccccccc}
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0
\end{array}
\]

\[R_1 = 1, \quad Q_E = 1,\]
Easy for **partial** functions

**Example:** Deutsch-Jozsa problem (almost)

- **Reject** iff all input variables are zeroes
  
  \[
  \begin{array}{cccccccc}
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  \end{array}
  \]

- **Accept** iff exactly half of the variables are ones
  
  \[
  \begin{array}{cccccccc}
  0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
  \end{array}
  \]

\[ R_1 = 1, \quad Q_E = 1, \quad R_0 = n/2 + 1 \]

\[
\begin{array}{ccc}
0 & 0 & 00
\end{array}
\]
Easy for \textit{partial} functions

\textbf{Example:} Deutsch-Jozsa problem (almost)

- \textbf{Reject} iff all input variables are zeroes

\begin{center}
\begin{tabular}{ccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{tabular}
\end{center}

- \textbf{Accept} Total Functions — ???

\begin{center}
\begin{tabular}{cc}
0 & 0 \\
\hline
0 & 0
\end{tabular}
\end{center}

\begin{align*}
R_1 &= 1, \\
Q_E &= 1, \\
R_0 &= n/2 + 1
\end{align*}
Iterated NAND: record-holder for $R_0, R_1, R_2$ versus $D$
Iterated NAND: record-holder for $R_0$, $R_1$, $R_2$ versus $D$

We have [Snir’85, Saks & Wigderson’86]:

$$R_0 = R_1 = R_2 = O(n^{0.7537...}), \quad D = n$$
It is known [Nisan’89]

\[ D = O(R_1^2) \]

We get functions with:

\[ D = \tilde{\Theta}(R_0^2) \]

\[ R_0 = \tilde{\Theta}(R_1^2) \]
It is known [Nisan’89]

\[ D = O(R_1^2) \]

We get functions with:

\[ D = \tilde{\Theta}(R_0^2) \]
\[ R_0 = \tilde{\Theta}(R_1^2) \]

The last one also saturates [Kulkarni & Tal’13, Midrijānis’05]

\[ R_0 = \tilde{O}(R_2^2) \]
Göös-Pitassi-Watson

Paper
Goal
$D$ versus 1-certificates
Pointers
Features
Our Modifications

$R_1$ versus $R_0$

$R_0$ versus $D$

Conclusion
Deterministic Communication vs. Partition Number

Mika Göös  
Toniann Pitassi  
Thomas Watson

Department of Computer Science, University of Toronto

April 1, 2015

Abstract

We show that deterministic communication complexity can be superlogarithmic in the partition number of the associated communication matrix. We also obtain near-optimal deterministic lower bounds for the Clique vs. Independent Set problem, which in particular yields new lower bounds for the log-rank conjecture. All these results follow from a simple adaptation of a communication-to-query simulation theorem of Raz and McKenzie (Combinatorica 1999) together with lower bounds for the analogous query complexity questions.
Clique vs. Independent Set in communication complexity

Reduce to a problem in query complexity: Find a function that
- has large deterministic complexity
- has small unambiguous 1-certificates

There exists a number of 1-certificates such that each positive input satisfies exactly one of them.
Function on $nm$ Boolean variables

- **Accept** iff there exists a unique all-1 column

- $D = nm$

- short 1-certificates $(n + m - 1)$, **BUT not** unambiguous.
Function on $nm$ Boolean variables

- **Accept** iff there exists a unique all-1 column

$$D = nm$$

- short 1-certificates $(n + m - 1)$, **BUT not** unambiguous. Should specify which zero to take in each column
Alphabet: $\{0, 1\} \times ([n] \times [m] \cup \{\perp\})$

Not Boolean, but we can encode using $O(\log(n + m))$ bits.

Accept iff

- There is a (unique) all-1 column $b$;
- in $b$, there is a unique element $r$ with non-zero pointer;
- following the pointers from $r$, we traverse through exactly one zero in each column but $b$. 
Still have $D = nm$

short unambiguous 1-certificates $(n + m - 1)$
Highly elusive (flexible)

Still traversable (if know where to start).
Our Modifications

Introduction

Göös-Pitassi-Watson

Our Modifications

Binary Tree
Definition (base)

$R_1$ versus $R_0$

$R_0$ versus $D$

Conclusion
Instead of a list

we use a balanced binary tree

■ More elusive
■ Random access
Accept iff

- There is a (unique) all-1 column $b$;
- in $b$, there is a unique element $r$ with non-zero pointers;
- for each $j \neq b$, following a path $T(j)$ from $r$ gives a zero in the $j$th column.
- Some additional information is contained in the leaves (to be defined).
$R_1$ versus $R_0$
• **NO** separation was known even between $R_2$ and $R_0$.

(Iterated functions are not of much help here.)
Recall the separation for a partial function

- **Reject** iff all input variables are zeroes

  ![0 0 0 0 0 0 0 0 0](image)

- **Accept** iff exactly half of the variables are ones

  ![0 1 0 1 1 0 0 1](image)
Add a back pointer to each variable.

Accept iff

- exactly $m/2$ of the leaves back point to the root $r$. 
A column is **good** if it contains a leaf back pointing to the root of a legitimate tree.

- A positive input contains exactly \( m/2 \) good columns.
- A negative input contains no good columns.

A total function looks like a partial function.
Deterministic subroutine

Given a column $c \in [m]$, accept iff it is good.

On each step we either

- eliminate a column: it is not the all-1 column; or
- eliminate an element in column $c$: it is not a leaf of the tree.
Deterministic subroutine

Given a column $c \in [m]$, accept iff it is good.

- **While** there is $\geq 2$ non-eliminated columns:
  - Let $a$ be a non-eliminated element in $c$. If none, **reject**.
  - Let $r$ be the back pointer of $a$, and $b$ be the column of $r$.
  - Let $j$ be a non-eliminated column $\neq b$.
  - If the path $T(j)$ from $r$ ends in a zero in column $j$, eliminate column $j$.
    Otherwise, eliminate element $a$.

- Verify the only non-eliminated column.
On each iteration of the loop, either an element or a column gets eliminated. At most $n + m$ iterations.

**Complexity:** $\tilde{O}(n + m)$.

Sticking into Deutsch-Jozsa, get $R_1$ and $Q_E$ upper bound of $\tilde{O}(n + m)$. 
(Negative) input with exactly one zero in each column.

- An $R_0$ algorithm can reject only if it has found $m/2$ zeroes.

Requires $\Omega(nm)$ queries.
Upper bound for $R_1$ and $Q_E$ is $\tilde{O}(n + m)$.

Lower bound for a $R_0$ algorithm is $\Omega(nm)$.

Taking $n = m$, we get a quadratic separation between $R_1$ and $R_0$, as well as between $Q_E$ and $R_0$.

**NB.** The previous separation was [Ambainis’12]:

$$Q_E = O(R_0^{0.8675...})$$
$R_0$ versus $D$
Back pointers are to columns.

Accept iff

□ ... □
□ all the leaves back point to the all-1 column $b$. 
Adversary Method.
Let \( n = 2m \).
If the \( k \)th element is queried in a column:
- If \( k \leq m \), return 1.
- Otherwise, return 0 with back pointer to column \( k - m \).

At the end, the column contains \( m \) 1 and \( m \) 0 with back pointers to all columns 1, 2, \ldots, \( m \).
- The algorithm does not know the value of the function until it has queried \( > m \) elements in each of \( m \) columns.

Lower bound: \( \Omega(m^2) \).
Each column contains a back pointer to the all-1 column.  
**BUT** which one is the right one—?

We try each back pointer by querying few elements in the column, and proceed to a one where no zeroes were found.

- Even if this is not the all-1 column, we can arrange that it contains fewer zeroes whp.
Algorithm

- Let $c$ be the first column, and $k \leftarrow n$.
- **While** $k > 1$,
  - Let $c \leftarrow \text{ProcessColumn}(c, k)$, and $k \leftarrow k/2$.

**ProcessColumn** (column $c$, integer $k$)

- Query all elements in column $c$.
- **If** there are no zeroes, verify column $c$.
- **If** there are more then $k$ zeroes, query all $nm$ variables, and output the value of the function.
- **For** each zero $a$:
  - Let $j$ be the back pointer of $a$.
  - Query $\tilde{O}(n/k)$ elements in column $j$. (Probability $< \frac{1}{(nm)^2}$ that no zero found if there are $> k/2$ of them).
  - **If** no zero was found, return $j$.
- Reject
Take $n = 2m$.

- Lower bound for a $D$ algorithm is $\Omega(m^2)$.
- Upper bound for a $R_0$ algorithm is $\tilde{O}(n + m)$.

We get a quadratic separation between $R_0$ and $D$. 
Take \( n = 2m \).

- Lower bound for a \( D \) algorithm is \( \Omega(m^2) \).
- Upper bound for a \( R_0 \) algorithm is \( \tilde{O}(n + m) \).

We get a quadratic separation between \( R_0 \) and \( D \).

- Also, upper bound for a \( Q \) algorithm is \( \tilde{O}(\sqrt{n + m}) \).

We get a quartic separation between \( Q \) and \( D \).

NB. Previous separation was quadratic: Grover’s search.
Conclusion
Results

\[
R_1 = \tilde{O}(R_0^{1/2}) \\
Q_E = \tilde{O}(R_0^{1/2}) \\
R_0 = \tilde{O}(D^{1/2}) \\
Q = \tilde{O}(D^{1/4}) \\
Q = \tilde{O}(R_0^{1/3}) \\
Q_E = \tilde{O}(R_0^{2/3}) \\
\deg = \tilde{O}(R_0^{1/4})
\]
Open Problems

We have resolved $R_2 \leftrightarrow R_0$ and $R_1 \leftrightarrow D$.

Can we resolve $R_2 \leftrightarrow D$ too?
Known: $R_2 = \Omega(D^{1/3})$ and $R_2 = \tilde{O}(D^{1/2})$.

- Can we overcome the “certificate complexity barrier”? Obtain a function with $R_2 = o(C)$?

- The same about $Q \leftrightarrow D$
  Known: $Q = \Omega(D^{1/6})$ and $Q = \tilde{O}(D^{1/4})$.

- and $Q_E \leftrightarrow D$?
  Known: $Q_E = \Omega(D^{1/3})$ and $Q_E = \tilde{O}(D^{1/2})$. 
Any questions?