On the Communication Complexity of Approximate Pattern Matching

Tomasz Kociumaka[1] **Jakob Nogler**[2] Philip Wellnitz[3]

[1]Max Planck Institute for Informatics, SIC ($\rightarrow$ INSAIT)

[2]ETH Zurich (work mostly carried out during summer internship at MPI)

[3]National Institute of Informatics, SOKENDAI

Pattern Matching (PM)

Text T, $|T| = n$ a b a a b a b a b a b b b a b b a a a b b b a b b a a

Pattern P, $|P| = m$ a b a b a b b b a b b a a

Pattern Matching (PM)

Text T, $\lvert T\rvert = n$	`a b a a b a b a b a b b b a b b a a a b b b a b b a a`
Pattern P, $\lvert P\rvert = m$	`a b a b a b b b a b b a a`

- **Exact PM:** Compute $\mathrm{Occ}(P, T) := \{x \mid T[x \mathinner{.\,.} x+m) = P\}$.

Pattern Matching (PM)

Text T, $|T| = n$ `a b a a b a b a b a b b b a b b a a a b b b a b b a a`

Pattern P, $|P| = m$ `a b a b a b b b a b b a a`

$k = 2$

- **Exact PM:** Compute $\mathrm{Occ}(P, T) := \{x \mid T[x \mathinner{.\,.} x+m) = P\}$.
- **PM with mismatches:** Compute $\mathrm{Occ}^H_k(P, T) := \{x \mid \mathrm{HD}(T[x \mathinner{.\,.} x+m), P) \leq k\}$.

Pattern Matching (PM)

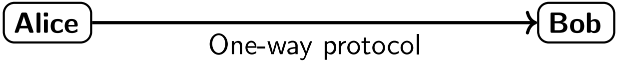
Text T, $|T| = n$ `a b a` ~~a~~ `b a b a b a b b b a b b a a a b b b a b b a a`

Pattern P, $|P| = m$ `a b a b a b b b a b b a a`

$k = 3$

- **Exact PM:** Compute $\mathrm{Occ}(P, T) := \{x \mid T[x \mathinner{.\,.} x + m) = P\}$.
- **PM with mismatches:** Compute $\mathrm{Occ}^H_k(P, T) := \{x \mid \mathrm{HD}(T[x \mathinner{.\,.} x + m), P) \leq k\}$.
- **PM with edits:** Compute $\mathrm{Occ}^E_k(P, T) := \{x \mid \exists y \ \mathrm{ED}(T[x \mathinner{.\,.} y), P) \leq k\}$.

Communication Complexity

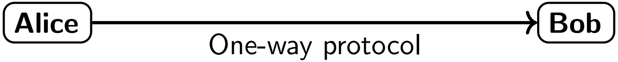


Communication Complexity

$$\textbf{Alice} \xrightarrow[\text{One-way protocol}]{} \textbf{Bob}$$

1 Alice receives a PM instance.
$Text\ T$, Pattern P, Threshold k

Communication Complexity



1. Alice receives a PM instance.
   Text T, Pattern P, Threshold k
2. Alice compresses the input.

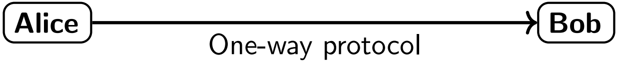
Communication Complexity

Alice $\xrightarrow{\hspace{6em}}$ **Bob**

One-way protocol

1. Alice receives a PM instance. *T*ext T, Pattern P, Threshold k
2. Alice compresses the input.
3. Alice sends compressed data to Bob.

Communication Complexity



1. Alice receives a PM instance. *Text* T, Pattern P, Threshold k
2. Alice compresses the input.
3. Alice sends compressed data to Bob.
4. Bob needs to reconstructs the output of the instance. *Set* $Occ_k^E(P, T)$

Communication Complexity

$$\textbf{Alice} \xrightarrow[\text{One-way protocol}]{} \textbf{Bob}$$

1. Alice receives a PM instance. *Text T, Pattern P, Threshold k*
2. Alice compresses the input.
3. Alice sends compressed data to Bob.
4. Bob needs to reconstructs the output of the instance. *Set $Occ_k^E(P, T)$*

Communication Complexity = "minimum # of machine words to send to Bob"

Example: Exact PM

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Text T	a	b	b	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	b
Pattern P	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b													

Example: Exact PM

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Text T	a	b	b	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	b
Pattern P	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b													

Bob needs to reconstruct $\mathrm{Occ}(P, T) = \{3, 6, 9, 12\}$

Example: Exact PM

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Text T	a	b	b	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b	b
Pattern P	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b													

Bob needs to reconstruct $\mathrm{Occ}(P, T) = \{3, 6, 9, 12\}$

Previous Results

Interesting case: $n \leq 3/2 \cdot m$

Previous Results

Interesting case: $n \leq 3/2 \cdot m$

	UB	**LB**	**Refererence**
Exact PM	$\mathcal{O}(1)$	$\Omega(1)$	Periodicity Lemma, **[FW65]**

Previous Results

Interesting case: $n \leq 3/2 \cdot m$

	UB	**LB**	**Refererence**
Exact PM	$\mathcal{O}(1)$	$\Omega(1)$	Periodicity Lemma, **[FW65]**
PM with mismatches	$\mathcal{O}(k)$	$\Omega(k)$	**[CKP19]**

Previous Results

Interesting case: $n \leq 3/2 \cdot m$

	UB	**LB**	**Refererence**
Exact PM	$\mathcal{O}(1)$	$\Omega(1)$	Periodicity Lemma, **[FW65]**
PM with mismatches	$\mathcal{O}(k)$	$\Omega(k)$	**[CKP19]**
PM with edits	$\mathcal{O}(k^3)$		**[CKW20]**
PM with edits	$\mathcal{O}(k \log m)$	$\Omega(k)$	**This Work**

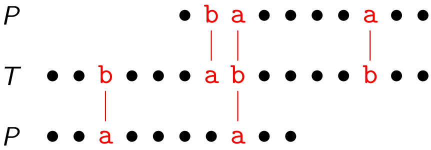
Finding a Period Structure

Suppose Alice takes a set S of alignments,

```
P           a b a b b a a a b b
            | | | | | | | | | |
T a b b b a a a b b b a a b b b
  | | |\ \ \ \ \| |
P a b a b b a a a b b
```

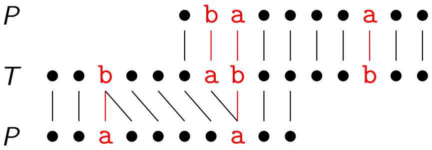
Finding a Period Structure

Suppose Alice takes a set S of alignments, and sends to Bob only the information about edits.



Finding a Period Structure

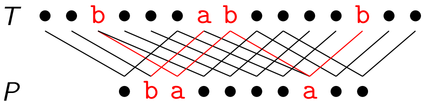
Suppose Alice takes a set S of alignments, and sends to Bob only the information about edits.



- Bob reconstructs the alignments in S.

Finding a Period Structure

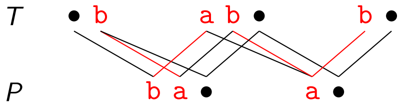
Suppose Alice takes a set S of alignments, and sends to Bob only the information about edits.



- Bob reconstructs the alignments in S.
- Bob makes a graph out of it.

Finding a Period Structure

Suppose Alice takes a set S of alignments, and sends to Bob only the information about edits.

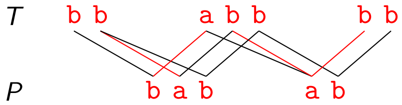


- Bob reconstructs the alignments in S.
- Bob makes a graph out of it.
- Bob selects red connected components,

.

Finding a Period Structure

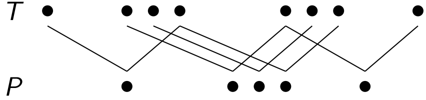
Suppose Alice takes a set S of alignments, and sends to Bob only the information about edits.



- Bob reconstructs the alignments in S.
- Bob makes a graph out of it.
- Bob selects red connected components, and propagates characters in them.

Finding a Period Structure

Suppose Alice takes a set S of alignments, and sends to Bob only the information about edits.

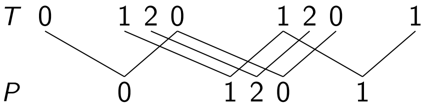


- Bob reconstructs the alignments in S.
- Bob makes a graph out of it.
- Bob selects red connected components, and propagates characters in them.
- Bob selects black connected components,

.

Finding a Period Structure

Suppose Alice takes a set S of alignments, and sends to Bob only the information about edits.



- Bob reconstructs the alignments in S.
- Bob makes a graph out of it.
- Bob selects red connected components, and propagates characters in them.
- Bob selects black connected components, and numbers them.

Finding a Period Structure

Suppose Alice takes a set S of alignments, and sends to Bob only the information about edits.

T	0			1	2	0			1	2	0		1
P				0			1	2	0		1		

- Bob reconstructs the alignments in S.
- Bob makes a graph out of it.
- Bob selects red connected components, and propagates characters in them.
- Bob selects black connected components, and numbers them.

Finding a Period Structure

Suppose Alice takes a set S of alignments, and sends to Bob only the information about edits.

$T_{|S}$ 0 1 2 0 1 2 0 1

$P_{|S}$ 0 1 2 0 1

- Bob reconstructs the alignments in S.
- Bob makes a graph out of it.
- Bob selects red connected components, and propagates characters in them.
- Bob selects black connected components, and numbers them.

Mapping Back the Periodic Structure to the Original Strings

$T_{|S}$ **0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5**

$P_{|S}$ **0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5**

Mapping Back the Periodic Structure to the Original Strings

$T_{|S}$ **0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5**

T b **0 1 2 3 4 5 6 7 8** a **0 1 2 3** b **4 5 6 7 8 0 1 2 3** a a **4 5 6 7 8** a **0 1 2 3 4 5 6 7 8 0 1 2** a **3 4 5**

$P_{|S}$ **0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5**

P a a **0 1 2 3 4 5 6 7 8 0 1 2 3** b b **4 5 6 7 8 0 1 2 3** a **4 5**

Mapping Back the Periodic Structure to the Original Strings

$T_{|S}$ 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5

T b 0 1 2 3 4 5 6 7 8 a 0 1 2 3 b 4 5 6 7 8 0 1 2 3 a a 4 5 6 7 8 a 0 1 2 3 4 5 6 7 8 0 1 2 a 3 4 5

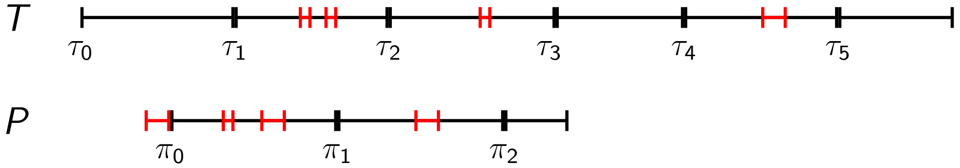
τ_0 τ_1 τ_2 τ_3 τ_4

$P_{|S}$ 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5

P a a 0 1 2 3 4 5 6 7 8 0 1 2 3 b b 4 5 6 7 8 0 1 2 3 a 4 5

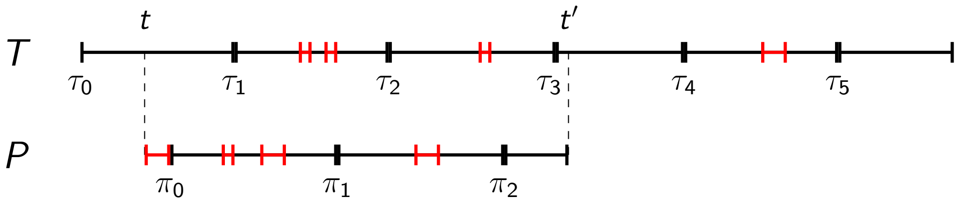
π_0 π_1 π_2

Alignments Covered by S



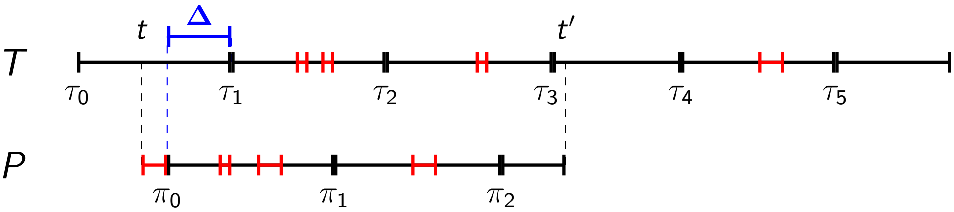
Alignments Covered by S

- Consider an arbitrary alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$.



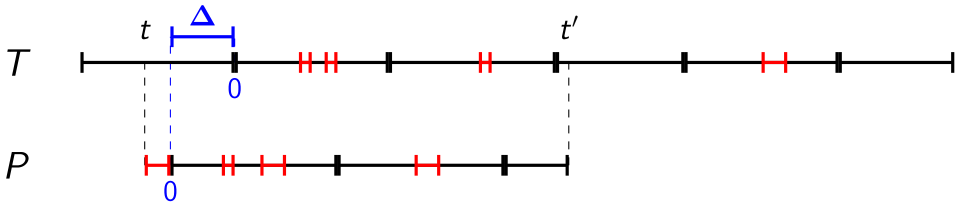
Alignments Covered by S

- Consider an arbitrary alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$.
- Compute $\boldsymbol{\Delta = \min_i |\tau_i - t - \pi_0|}$.



Alignments Covered by S

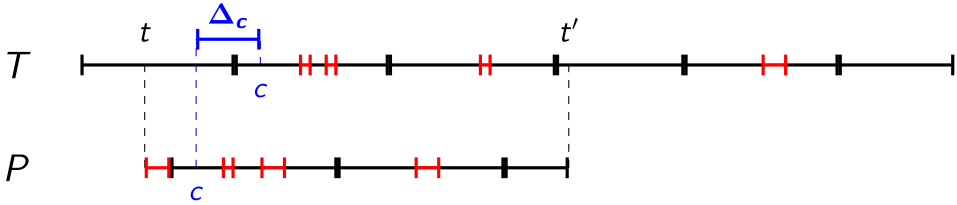
- Consider an arbitrary alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$.
- Compute $\boldsymbol{\Delta = \min_i |\tau_i - t - \pi_0|}$.



Case $\boldsymbol{\Delta} = \widetilde{\Omega}(k)$ (large)

Alignments Covered by S

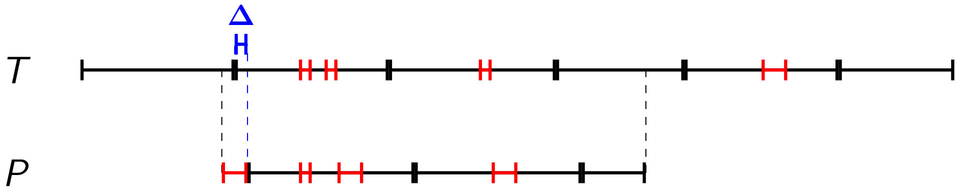
- Consider an arbitrary alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$.
- Compute $\boldsymbol{\Delta = \min_i |\tau_i - t - \pi_0|}$.



Case $\Delta = \widetilde{\Omega}(k)$ (large) $\implies$ if $\mathcal{X}$ is added to S, the # of black components at least halves.

Alignments Covered by S

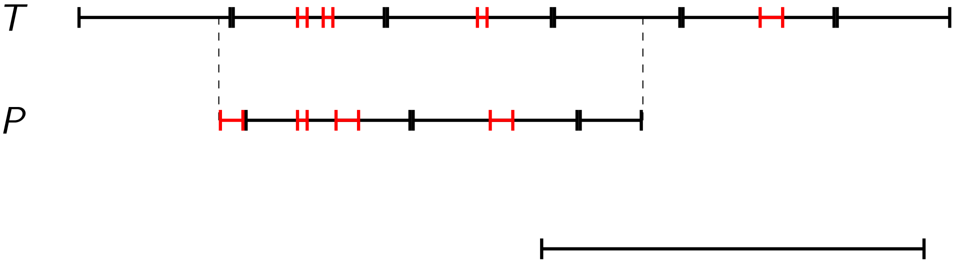
- Consider an arbitrary alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$.
- Compute $\boldsymbol{\Delta = \min_i |\tau_i - t - \pi_0|}$.



Case $\boldsymbol{\Delta} = \widetilde{\mathcal{O}}(k)$ (small)

Alignments Covered by S

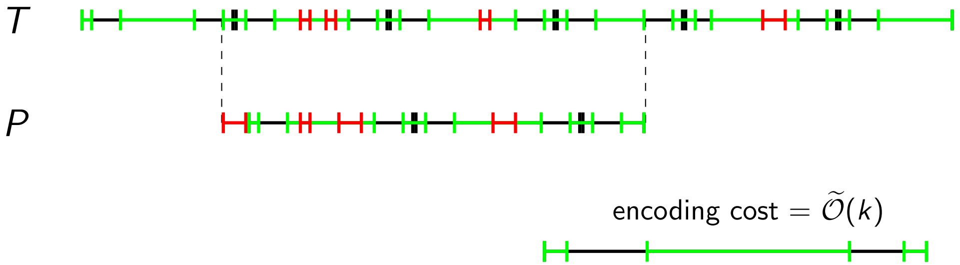
- Consider an arbitrary alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$.
- Compute $\boldsymbol{\Delta = \min_i |\tau_i - t - \pi_0|}$.



Case $\Delta = \widetilde{\mathcal{O}}(k)$ (small)

Alignments Covered by S

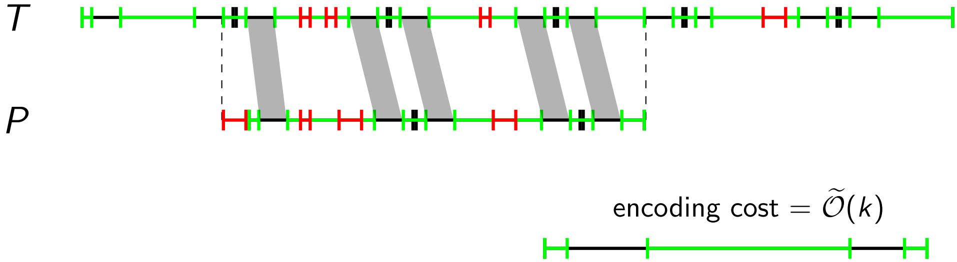
- Consider an arbitrary alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$.
- Compute $\boldsymbol{\Delta = \min_i |\tau_i - t - \pi_0|}$.



Case $\boldsymbol{\Delta} = \widetilde{\mathcal{O}}(k)$ (small)

Alignments Covered by S

- Consider an arbitrary alignment $\mathcal{X} : P \rightsquigarrow T[\,t \,..\, t'\,)$.
- Compute $\boldsymbol{\Delta = \min_i |\tau_i - t - \pi_0|}$.



Case $\boldsymbol{\Delta} = \widetilde{\mathcal{O}}(k)$ (small) $\implies$ there exists alignment with the same cost of $\mathcal{X}$ that matches characters in the same uncovered black components.

Pattern Matching in the Quantum Setting

We apply our communication complexity results to the quantum setting:

- Input string S given as oracle where queries can be made in **superposition**

Pattern Matching in the Quantum Setting

We apply our communication complexity results to the quantum setting:

- Input string S given as oracle where queries can be made in **superposition**
- **Query complexity** $Q(n)$: counts number of queries to oracle

Pattern Matching in the Quantum Setting

We apply our communication complexity results to the quantum setting:

- Input string S given as oracle where queries can be made in **superposition**
- **Query complexity** $Q(n)$: counts number of queries to oracle
- **Time complexity** $T(n)$: also counts the number of elementary gates

Pattern Matching in the Quantum Setting

We apply our communication complexity results to the quantum setting:

- Input string S given as oracle where queries can be made in **superposition**
- **Query complexity** $Q(n)$: counts number of queries to oracle
- **Time complexity** $T(n)$: also counts the number of elementary gates

Theorem [KN**W'24]**

There is a quantum algorithm that, given a PM with edits instance:

- computes $\mathrm{Occ}_k^E(P, T)$ using $\widehat{\mathcal{O}}(n/m \cdot \sqrt{km})$ queries and $\widehat{\mathcal{O}}(n/m \cdot (\sqrt{k}m + k^{3.5}))$ time;
- decides whether $\mathrm{Occ}_k^E(P, T) \neq \emptyset$ using $\widehat{\mathcal{O}}(\sqrt{n/m} \cdot \sqrt{km})$ queries and $\widehat{\mathcal{O}}(\sqrt{n/m} \cdot (\sqrt{k}m + k^{3.5}))$ time.

Pattern Matching in the Quantum Setting

We apply our communication complexity results to the quantum setting:

- Input string S given as oracle where queries can be made in **superposition**
- **Query complexity** $Q(n)$: counts number of queries to oracle
- **Time complexity** $T(n)$: also counts the number of elementary gates

Theorem [KNW'24]

There is a quantum algorithm that, given a PM with edits instance:

- computes $\mathrm{Occ}_k^E(P, T)$ using $\widehat{\mathcal{O}}(n/m \cdot \sqrt{km})$ queries and $\widehat{\mathcal{O}}(n/m \cdot (\sqrt{k}m + k^{3.5}))$ time;
- decides whether $\mathrm{Occ}_k^E(P, T) \neq \emptyset$ using $\widehat{\mathcal{O}}(\sqrt{n/m} \cdot \sqrt{km})$ queries and $\widehat{\mathcal{O}}(\sqrt{n/m} \cdot (\sqrt{k}m + k^{3.5}))$ time.

➤ The number of queries is optimal for $k = o(m)$ (up to a subpolynomial factor).

Main ingredients

- Structural insights for approximate pattern matching **[CKW20]**.

Main ingredients

- Structural insights for approximate pattern matching **[CKW20]**.
- Quantum algorithm for computing bounded edit distance **[GJKT24]**.

Main ingredients

- Structural insights for approximate pattern matching **[CKW20]**.
- Quantum algorithm for computing bounded edit distance **[GJKT24]**.
- Quantum *Gap Edit Distance* algorithm (adapted from **[GKKS22]**).

Main ingredients

- Structural insights for approximate pattern matching **[CKW20]**.
- Quantum algorithm for computing bounded edit distance **[GJKT24]**.
- Quantum *Gap Edit Distance* algorithm (adapted from **[GKKS22]**).
- Our communication complexity results.

Thanks!