Complexity of Dense Linear Operators

Vladimir V. Podolskii[1]

joint work with Alexander Kulikov, Ivan Mikhailin, Andrey Mokhov

[1] Steklov Mathematical Institute, Moscow
Higher School of Economics, Moscow

HSE Open Lectures

Setting

Consider a Boolean matrix $A \in \{0,1\}^{n\times n}$
Consider variables $x = (x_1, \ldots, x_n)$ over $\{0,1\}$

We want to compute a Boolean linear operator Ax

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad Ax = \begin{pmatrix} x_1 \vee x_3 \\ x_1 \vee x_2 \\ x_2 \vee x_3 \vee x_4 \\ x_1 \vee x_2 \vee x_3 \vee x_4 \end{pmatrix}$$

Model

- The computation is a Boolean circuit consisting of OR gates
- We start with variables $x_1, \ldots, x_n$
- In one step we can compute OR of two previously computed expressions
- Want to compute all the outputs and minimize the number of steps

Example

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$x_1 \vee x_3$
$x_1 \vee x_2$
$x_2 \vee x_3 \vee x_4$
$x_1 \vee x_2 \vee x_3 \vee x_4$

Computation:
$x_1 \vee x_3$ — output
$x_1 \vee x_2$ — output
$x_3 \vee x_4$
$x_2 \vee (x_3 \vee x_4)$ — output
$x_1 \vee (x_2 \vee x_3 \vee x_4)$ — output

Basic facts

- One of the simplest Boolean circuit complexity models, studied since 50's
- Trivial upper bound: $O(n^2)$
- Counting lower bound: $\Omega(n^2/\log n)$
- Non-trivial upper bound: $O(n^2/\log n)$ (Lupanov '56)
- The best explicit lower bound: $\Omega(n^{2-o(1)})$ (Nechiporuk '70)

General setting

Consider a Boolean matrix $A \in \{0,1\}^{n \times n}$
Consider variables $x = (x_1, \ldots, x_n)$ over some semigroup $(S, \circ)$.

We want to compute a linear operator Ax.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad Ax = \begin{pmatrix} x_1 \circ x_3 \\ x_1 \circ x_2 \\ x_2 \circ x_3 \circ x_4 \\ x_1 \circ x_2 \circ x_3 \circ x_4 \end{pmatrix}$$

Some semigroups

- Boolean semigroup: $(\{0,1\}, \vee)$
- Integers with addition: $(\mathbb{Z}, +)$
- $\{0,1\}$ with addition: $(\{0,1\}, \oplus)$
- Tropical semigroup: $(\mathbb{Z}, \min)$

The Problem

Problem: Suppose $A \in \{0,1\}^{n \times n}$ is very dense, that is A has $O(n)$ zeros. How hard is it to compute Ax?

The Problem

Problem: Suppose $A \in \{0,1\}^{n \times n}$ is very dense, that is A has $O(n)$ zeros. How hard is it to compute Ax?

Simple observations:

- If instead we consider A containing $O(n)$ ones, the complexity is trivially $O(n)$

The Problem

Problem: Suppose $A \in \{0,1\}^{n \times n}$ is very dense, that is A has $O(n)$ zeros. How hard is it to compute Ax?

Simple observations:

- If instead we consider A containing $O(n)$ ones, the complexity is trivially $O(n)$
- If $(S, \circ)$ has an inverse operation (is a group), the complexity is trivially $O(n)$

Motivation

- Want to understand the structure of semigroups that are not groups

Motivation

- Want to understand the structure of semigroups that are not groups
- Important examples: Boolean semigroup and tropical semigroup

Motivation

- Want to understand the structure of semigroups that are not groups
- Important examples: Boolean semigroup and tropical semigroup
- Famous problem of similar flavor: matrix multiplication
  Given two matrices A, B, how many operations are needed to compute $A \cdot B$?

Motivation

- Want to understand the structure of semigroups that are not groups
- Important examples: Boolean semigroup and tropical semigroup
- Famous problem of similar flavor: matrix multiplication
  Given two matrices A, B, how many operations are needed to compute $A \cdot B$?
- Non-trivial upper bound over integers: $O(n^{2.373})$ (V. Williams '12)

Motivation

- Want to understand the structure of semigroups that are not groups
- Important examples: Boolean semigroup and tropical semigroup
- Famous problem of similar flavor: matrix multiplication
  Given two matrices A, B, how many operations are needed to compute $A \cdot B$?
- Non-trivial upper bound over integers: $O(n^{2.373})$ (V. Williams '12)
- Best known upper bound over tropical semiring is $O(n^3/e^{\sqrt{\log n}})$ (R. Williams '14)

Motivation

- Want to understand the structure of semigroups that are not groups
- Important examples: Boolean semigroup and tropical semigroup
- Famous problem of similar flavor: matrix multiplication
  Given two matrices A, B, how many operations are needed to compute $A \cdot B$?
- Non-trivial upper bound over integers: $O(n^{2.373})$ (V. Williams '12)
- Best known upper bound over tropical semiring is $O(n^3/e^{\sqrt{\log n}})$ (R. Williams '14)
- Other motivation: connection to range minimum query problem (will see later)

Main results

Theorem

If $(S, \circ)$ *is a commutative semigroup, then* Ax *can be computed in* $O(n)$ *operations for dense* A

Main results

Theorem
If $(S, \circ)$ is a commutative semigroup, then Ax can be computed in $O(n)$ operations for dense A

Theorem
If $(S, \circ)$ is strongly non-commutative semigroup, then the maximal complexity of Ax is $\Theta(n\alpha(n))$ operations for dense A

Here $\alpha(n)$ is the inverse Ackermann function

Main results

Theorem

If $(S, \circ)$ is a commutative semigroup, then Ax can be computed in $O(n)$ operations for dense A

Theorem

If $(S, \circ)$ is strongly non-commutative semigroup, then the maximal complexity of Ax is $\Theta(n\alpha(n))$ operations for dense A

Here $\alpha(n)$ is the inverse Ackermann function

$\alpha(n)$ growth is extremely slow

For all practical needs we can assume $\alpha(n) \leqslant 4$

Plan

- Upper bound + connection to RMQ
- A bit on lower bounds (+ connection to RMQ)

Upper bound

Theorem (restated)

If $(S, \circ)$ is a commutative semiring, then Ax can be computed is $\Theta(n)$ for dense A

- Let's concentrate on Boolean case: $(\{0,1\}, \vee)$
- The general case is the same
- So, $A \in \{0,1\}^{n\times n}$ has $O(n)$ zeros, we want to compute Ax using $O(n)$ operations

Warm Up

Consider

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

How would we compute Ax?

Warm Up

Consider

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

How would we compute Ax?

Compute:

all prefixes

x_1

$x_1 \vee x_2$

$x_1 \vee x_2 \vee x_3$

$x_1 \vee x_2 \vee x_3 \vee x_4$

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$

Warm Up

Consider

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

How would we compute Ax?

Compute:

all prefixes	$x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6$
x_1	$x_3 \vee x_4 \vee x_5 \vee x_6$
$x_1 \vee x_2$	$x_4 \vee x_5 \vee x_6$
$x_1 \vee x_2 \vee x_3$	$x_5 \vee x_6$
$x_1 \vee x_2 \vee x_3 \vee x_4$	x_6
$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$	all suffixes

Warm Up

Consider

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

How would we compute Ax?

Compute:

$$\begin{array}{lcr} & \text{match} & \\ \text{all prefixes} & & x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6 \\ x_1 & \leftrightarrow & x_3 \vee x_4 \vee x_5 \vee x_6 \\ x_1 \vee x_2 & \leftrightarrow & x_4 \vee x_5 \vee x_6 \\ x_1 \vee x_2 \vee x_3 & \leftrightarrow & x_5 \vee x_6 \\ x_1 \vee x_2 \vee x_3 \vee x_4 & \leftrightarrow & x_6 \\ x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 & & \text{all suffixes} \end{array}$$

Warm Up-2

Suppose $A \in \{0, 1\}^{n \times n}$ has 2 zeros in each row. How would we compute Ax?

Warm Up-2

Suppose $A \in \{0,1\}^{n\times n}$ has 2 zeros in each row. How would we compute Ax?

Permute columns of A randomly and cut in two halves.

$$\left[\begin{array}{ccc|ccc}
 & 0 & & & 0 & \\
0 & & & & & 0 \\
 & & 0 & 0 & & \\
 & & & 0 & 0 & \\
 & & & & 0 & 0 \\
0 & 0 & & & &
\end{array}\right]$$

Warm Up-2

Suppose $A \in \{0,1\}^{n\times n}$ has 2 zeros in each row. How would we compute Ax?

Permute columns of A randomly and cut in two halves.

$$\left[\begin{array}{ccc|ccc}
 & \color{red}{0} & & & \color{red}{0} & \\
\color{red}{0} & & & & & \color{red}{0} \\
 & & \color{red}{0} & \color{red}{0} & & \\
\hline
 & & & 0 & 0 & \\
 & & & & 0 & 0 \\
0 & 0 & & & &
\end{array}\right]$$

With positive probability in at least half of the rows the zeros will be splitted.
Compute them by the previous algorithm, compute the rest recursively.

Warm Up-2

Suppose $A \in \{0,1\}^{n \times n}$ has 2 zeros in each row. How would we compute Ax?

Permute columns of A randomly and cut in two halves.

$$
\left[\begin{array}{ccc|ccc}
 & 0 & & & 0 & \\
0 & & & & & 0 \\
 & & 0 & 0 & & \\
\hline
 & & & 0 & 0 & \\
 & & & & 0 & 0 \\
0 & 0 & & & &
\end{array}\right]
$$

We get the recurrence

$$T(n) = Cn + T(n/2),$$

where $T(n)$ is the complexity for matrices with n rows

Towards the General Case

How would we solve the general case?

First idea: Connection to Range Minimum Query problem (RMQ)

This is the standard setting in theory of algorithms

We are given an array of numbers $x_1, \ldots, x_n$. We want a data structure to answer queries of the form

$$\min\{x_i \mid l \leqslant i \leqslant r\} = ?$$

for integer l and r.

Reduction to RMQ

Consider

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Split each row into intervals

$$\begin{array}{lll} x_1 & x_3 \vee x_4 & x_6 \\ x_1 \vee x_2 & x_4 \vee x_5 \vee x_6 & \\ x_1 & x_3 \vee x_4 \vee x_5 & \\ x_1 \vee x_2 \vee x_3 & x_5 \vee x_6 & \\ x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6 & & \\ x_1 & x_4 \vee x_5 \vee x_6 & \end{array}$$

There are $O(n)$ intervals in total, so we reduced our problem to the offline version of RMQ (intervals are given in advance)

Complexity of RMQ

Unfortunately, best constructions for RMQ give only $O(n\alpha(n))$ complexity in our model, where $\alpha(n)$ is an inverse Ackermann function

Complexity of RMQ

Unfortunately, best constructions for RMQ give only $O(n\alpha(n))$ complexity in our model, where $\alpha(n)$ is an inverse Ackermann function

Moreover, the following is true

Theorem (Chazelle, Rozenberg '91)

There are range matrices $A \in \{0,1\}^{n\times n}$ with the complexity $\Omega(n\alpha(n))$

So, the reduction to RMQ is not enough

Idea

$$A = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{bmatrix}$$

There is a simple construction with complexity $O(n \log n)$

Idea

$$A = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{bmatrix}$$

There is a simple construction with complexity $O(n \log n)$

- Compute all prefixes and suffixes

Idea

$$A = \left[\begin{array}{ccc|ccc} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array}\right]$$

There is a simple construction with complexity $O(n \log n)$

- Compute all prefixes and suffixes
- Split the matrix in two halves vertically

Idea

$$A = \left[\begin{array}{ccc|ccc} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array}\right]$$

There is a simple construction with complexity $O(n \log n)$

- Compute all prefixes and suffixes
- Split the matrix in two halves vertically
- Repeat in both halves recursively

Idea

$$A = \left[\begin{array}{ccc|ccc} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array}\right]$$

There is a simple construction with complexity $O(n \log n)$

- Compute all prefixes and suffixes
- Split the matrix in two halves vertically
- Repeat in both halves recursively
- Now we can compute any range in just one operation, just pick the first vertical line that crosses it

Next Idea

Lemma

Suppose $A \in \{0,1\}^{n \times n}$ has $O(n)$ zeros and each range is of length at least $\log n$. Then we can compute Ax in $O(n)$ operations

- Split into blocks of size $\log n$
- Each range intersects block boundary!
- Compute prefixes and suffixes in each block in $O(n)$ operations
- Apply the previous idea on top of blocks
- Now for each range we can compute its 'complete' part in one operation and add 'incomplete' prefix and suffix in two more operations

Yet Another Next Idea

Lemma

Suppose $A \in \{0,1\}^{n\times n}$ has $O(n)$ zeros and at most $\log n$ zeros in each row. Then we can compute it in $O(n)$ operations

- Permute blocks randomly
- With high probability most zeros will be far from each other
- Thus most ranges are long
- Compute them by the previous idea
- Compute all short ranges by brute force (there are few of them)

Final Idea

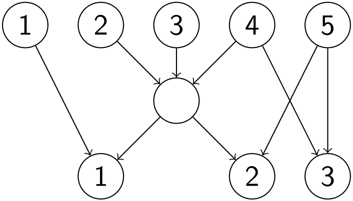
Theorem

Suppose $A \in \{0,1\}^{n\times n}$ *has complexity* s*. Then* A^T *also has complexity* s

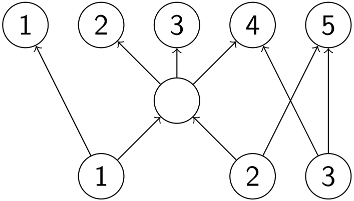
Final Idea

Theorem

Suppose $A \in \{0,1\}^{n \times n}$ has complexity s. Then A^T also has complexity s



$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



$$A^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

So, What Do We Have?

- Can compute Ax in $O(n \log n)$ operations
- If each row has at most $\log n$ zeros, can compute Ax in $O(n)$ opetations
- Ax and $A^T y$ have the same complexity

Completing the Proof

$$A = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{bmatrix}$$

Suppose we have A with $O(n)$ zeros

Completing the Proof

$$A = \left[\begin{array}{cccccc} * & * & * & * & * & * \\ * & * & * & * & * & * \\ \hline * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array}\right]$$

Suppose we have A with $O(n)$ zeros

- Split rows in two parts: with more than $\log n$ zeros and with at most $\log n$ zeros

Completing the Proof

$$A = \left[\begin{array}{cccccc} * & * & * & * & * & * \\ * & * & * & * & * & * \\ \hline * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array}\right]$$

Suppose we have A with $O(n)$ zeros

- Split rows in two parts: with more than $\log n$ zeros and with at most $\log n$ zeros
- We know how to compute the second part

Completing the Proof

$$A = \left[\begin{array}{cccccc} * & * & * & * & * & * \\ * & * & * & * & * & * \\ \hline * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array}\right]$$

Suppose we have A with $O(n)$ zeros

- Split rows in two parts: with more than $\log n$ zeros and with at most $\log n$ zeros
- We know how to compute the second part
- The first part has at most $n/\log n$ rows

Completing the Proof

$$A = \left[\begin{array}{cccccc} * & * & * & * & * & * \\ * & * & * & * & * & * \\ \hline * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array}\right]$$

Suppose we have A with $O(n)$ zeros

- Split rows in two parts: with more than $\log n$ zeros and with at most $\log n$ zeros
- We know how to compute the second part
- The first part has at most $n/\log n$ rows
- Transpose the first part

Completing the Proof

$$A = \left[\begin{array}{cccccc} * & * & * & * & * & * \\ * & * & * & * & * & * \\ \hline * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array}\right]$$

Suppose we have A with $O(n)$ zeros

- Split rows in two parts: with more than $\log n$ zeros and with at most $\log n$ zeros
- We know how to compute the second part
- The first part has at most $n/\log n$ rows
- Transpose the first part
- The transposition is computable in $O(n)$ operations

Non-commutative Case

Recall, that we heavily used commutativity even for the case when A has two zeros in each row: permutation of columns

Non-commutative Case

Recall, that we heavily used commutativity even for the case when A has two zeros in each row: permutation of columns

We next show that this is unavoidable:

Theorem

If $(S, \circ)$ is strongly non-commutative semiring, then there is A with at most 2 zeros in each row such that the complexity of Ax is $\Omega(n\alpha(n))$

Examples

- Recall Boolean case. There are equivalences on variables:

$$x_1 \vee x_1 = x_1, \qquad x_1 \vee x_2 = x_2 \vee x_1$$

Examples

- Recall Boolean case. There are equivalences on variables:

$$x_1 \vee x_1 = x_1, \quad x_1 \vee x_2 = x_2 \vee x_1$$

- The first is idempotency, the second is commutativity

Examples

- Recall Boolean case. There are equivalences on variables:

$$x_1 \vee x_1 = x_1, \quad x_1 \vee x_2 = x_2 \vee x_1$$

- The first is idempotency, the second is commutativity
- The first example for us is free semigroup: no equivalences on variables

Examples

- Recall Boolean case. There are equivalences on variables:

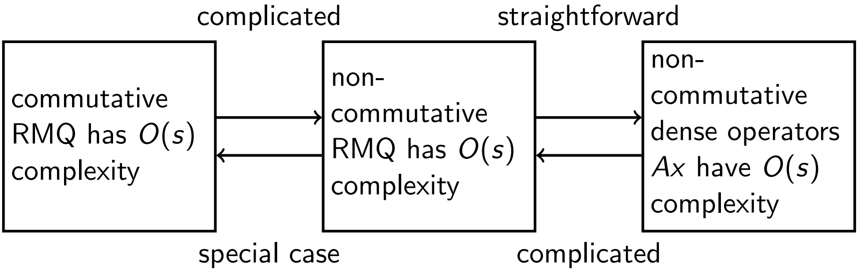
$$x_1 \vee x_1 = x_1, \qquad x_1 \vee x_2 = x_2 \vee x_1$$

- The first is idempotency, the second is commutativity
- The first example for us is free semigroup: no equivalences on variables
- The second example is free idempotent semigroup; the only equivalences are

$$x \circ x = x$$

Idempotent Case

We prove the following problem equivalences



Non-idempotent Case

- Suppose we have a small circuit
- Just factorize the whole computation by idempotency relations
- Now we have a small circuit for idempotent case
- This is a contradiction
- Basically, non-idempotent case is harder

Open problem

Problem (Jukna '19)

Consider $A \in \{0,1\}^{n\times n}$, denote by $\overline{A}$ the bit-wise negation of A. How large can

$$\frac{Complexity(\overline{A}x)}{Complexity(Ax)}$$

be (over Boolean semigroup)?

Open problem

Problem (Jukna '19)

Consider $A \in \{0,1\}^{n \times n}$, denote by $\overline{A}$ the bit-wise negation of A. How large can

$$\frac{\mathit{Complexity}(\overline{A}x)}{\mathit{Complexity}(Ax)}$$

be (over Boolean semigroup)?

Our result rules out the possibility to achieve gap with sparse matrix

Open problem

Problem (Jukna '19)

Consider $A \in \{0,1\}^{n \times n}$, denote by $\overline{A}$ the bit-wise negation of A. How large can

$$\frac{Complexity(\overline{A}x)}{Complexity(Ax)}$$

be (over Boolean semigroup)?

Our result rules out the possibility to achieve gap with sparse matrix

Thank you for attention!