Complexity of Searching for 2 by 2 Submatrices in Boolean Matrices

Daniel Průša\textsuperscript{1} and Michael Wehar\textsuperscript{2}

\textsuperscript{1}Czech Technical University, Prague, Czech Republic
\textsuperscript{2}Swarthmore College, Swarthmore, PA, USA

25th International Conference on Developments in Language Theory

August 16-20, 2021, Porto, Portugal
1. **Introduction**
   - Definitions
   - Motivation

2. **Results**
   - Algorithms
   - Hardness

3. **Conclusion**
   - Summary
   - Future Work
Given a Boolean matrix $M$

- **ANY $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$**: search $M$ for a submatrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$,
- **MIN $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$**: search $M$ for a submatrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ that encloses the minimum area of $M$,
- **MAX $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$**: search $M$ for a submatrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ that encloses the maximum area of $M$

for $a, b, c, d \in \{0, 1\}$. 
Examples

• ANY $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$:

\[
M = \begin{pmatrix}
1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0
\end{pmatrix}
\]

• MIN $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, MAX $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$:

\[
M = \begin{pmatrix}
1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0
\end{pmatrix}
\]
Motivation (Example)

Given a 2D table of employees and their skills,

- find two employees who know the same two programming languages.

|       | Java | C++ | Ruby | Pearl |
|-------|------|-----|------|-------|
| Miller| ✗    | ✗   | ✗    | ✗     |
| Smith |      | ✗   | ✗    |       |
| Taylor| ✗    |     | ✗    | ✗     |
Local picture languages (LOC)

\[ \Sigma \ldots \text{a finite alphabet} \]
\[ \# \notin \Sigma \ldots \text{the background symbol} \]
\[ \Theta \ldots \text{a set of } 2 \times 2 \text{ pictures (tiles) over } \Sigma \cup \{\#\} \]
\[ L(\Theta) \ldots \text{local picture language determined by } \Theta \]
Motivation (2D pattern matching)

2D pattern matching

- input is a picture (matrix) over a finite alphabet $\Sigma$,
- patterns to find are specified by a picture language over $\Sigma$,
- applicable to domains like spreadsheets, timetables, board games, puzzles, etc.

**Theorem (Mráz, Průša, Wehar 2019)**

Let $L$ be a local picture language. There is an algorithm searching for any/minimum/maximum matching subpicture from $L$ in time $O(mn \min\{m, n\})$ for pictures of size $m \times n$.

Is there a faster algorithm, working in time $O(mn)$?
Motivation (2D pattern matching)

Theorem (Mráz, Průša, Wehar 2019)

Searching for any matching 2D subpicture is Triangle Finding-hard for local picture languages over ternary alphabets.

- The proof is based on searching for submatrix \((\begin{array}{cc} 1 & 2 \\ 1 & 1 \end{array})\).
- What about binary alphabets?

⇒ It is worth it to study Four corners problems.
Triangle finding problem

Given a graph $G = (V, E)$, do there exist vertices $v_1, v_2, v_3 \in V$ such that $\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\} \in E$?

- Solvable in time $O(n^\omega)$ where $n = |V|$ and $\omega < 2.373$ denotes the matrix multiplication exponent.
- Unknown if there is an algorithm solving it in time $O(n^2)$. 
Conditional lower bounds based on Triangle Finding are known for several problems from the theory of formal languages:

- context-free grammar parsing,
- regular languages intersection emptiness,
- deciding if an NFA accepts a word of a given length.
1 Introduction
   • Definitions
   • Motivation

2 Results
   • Algorithms
   • Hardness

3 Conclusion
   • Summary
   • Future Work
Algorithms

Theorem

For any \( a, b, c, d \in \{0, 1\} \), there is an algorithm solving \( \text{ANY} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \) in time \( \tilde{O}(mn) \) for \( m \) by \( n \) Boolean matrices.

- It is sufficient to look at only four cases: \( \text{ANY} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \), \( \text{ANY} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \), \( \text{ANY} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \), and \( \text{ANY} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \).
- We apply a different strategy to solve each case.
Algorithm for ANY $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

Input: $M \in \{0, 1\}^{m,n}$ (assume $m \geq n$)

1: $S := \emptyset$
2: for each row $r$ of $M$ do
3:     $R := \emptyset$
4:     for each $i$ s.t. $M(r, i) = 1$ do
5:         add $i$ to $R$
6:     end for
7:     for each $i, j \in R$ s.t. $i < j$ do
8:         if $S$ contains $(i, j)$ then
9:             return matching submatrix
10:        else
11:            add $(i, j)$ to $S$
12:        end if
13:     end for
14: end for

$O(n^2 + mn) = O(mn)$
Hardness

**Theorem**

\[
\text{MAX } \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ is Triangle Finding-hard for any } a, b, c, d \in \{0, 1\}.
\]

Although MAX \( \begin{bmatrix} a & b \\ c & d \end{bmatrix} \) is hard, we do introduce a new approach.

**Theorem**

For any \( a, b, c, d \in \{0, 1\} \), there is an algorithm solving MAX \( \begin{bmatrix} a & b \\ c & d \end{bmatrix} \) in time \( O\left(mn \cdot \left(\min\{m, n\}\right)^{0.5302}\right) \) for \( m \) by \( n \) Boolean matrices.
**Theorem**

\[ \text{MAX}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ cannot be solved in time } O(mn) \text{ unless Triangle Finding can be solved in time } O(n^2). \]

\[
G = (V, E) \text{ contains a triangle iff } M \text{ contains submatrix } \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ that encloses area of at least } 3|V|^2 \text{ entries.}
\]
Algorithm for $\text{MAX} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

Given a Boolean matrix $\mathbf{M}$.
For each two distinct rows $i$ and $j$, we need to know
- minimum $c_1$ such that $\mathbf{M}(i, c_1) = \mathbf{M}(j, c_1) = 1$,
- maximum $c_2$ such that $\mathbf{M}(i, c_2) = \mathbf{M}(j, c_2) = 1$.

If $c_1$, $c_2$ exist and $c_1 \neq c_2$, then the maximum area enclosed by a submatrix $(\begin{array}{c} 1 \\ 1 \end{array})$ with entries in rows $i$ and $j$ is $(|i - j| + 1) \cdot (c_2 - c_1 + 1)$.

The indices $c_1$, $c_2$ can be obtained for all pairs $i$, $j$ from the minimum and maximum witness matrix for the product $\mathbf{MM}^T$. 
Minimum/maximum witness matrix

Let \( A \) and \( B \) be two Boolean matrices of dimensions \( m \times n \) and \( n \times m \), respectively.

A minimum/maximum witness matrix for the Boolean product \( C = AB \) is the \( m \times m \) matrix \( W \) fulfilling:

- If \( C(i,j) = 0 \), then \( W(i,j) = 0 \).
- Otherwise, \( W(i,j) \) is equal to the minimum/maximum index \( k \) such that \( A(i,k) = B(k,j) = 1 \).

Theorem (Cohen, Yuster 2014)

There is an algorithm computing the minimum/maximum witness matrices in time \( O(mn \cdot (\min\{m, n\})^{0.5302}) \).
Problem Complexities

- any/min/max subpicture matching for local picture languages is in time $O(mn \min\{m, n\})$
- $\text{ANY} \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ over Boolean matrices is in time $\tilde{O}(mn)$
- $\text{MAX} \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ over Boolean matrices is Triangle Finding-hard, but still is in time $O\left(mn \cdot (\min\{m, n\})^{0.5302}\right)$

Also, $\text{MIN} \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ over Boolean matrices is in time $\tilde{O}(mn)$. 
Questions

- We introduce four different algorithms for solving the \( \text{ANY } \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \), \( \text{ANY } \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \), \( \text{ANY } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \), and \( \text{ANY } \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \) problems. Is there a unified strategy for solving all \( \text{ANY } \begin{bmatrix} a & b \\ c & d \end{bmatrix} \) cases over Boolean matrices in time \( \tilde{O}(mn) \)?

- \( \text{ANY } \begin{bmatrix} a & b \\ c & d \end{bmatrix} \) over Boolean matrices can be viewed as a subpicture matching problem for a specific subclass of local picture languages. What are the complexities of the subpicture matching problems for other local picture languages?
Thank you!