An Asymptotically Optimal Algorithm for Maximum Matching in Dynamic Streams

Vihan Shah

Department of Computer Science
Rutgers University

February 1, 2022

Joint work with Sepehr Assadi
Matching Problem

- Graph $G = (V, E)$
- Matching: $M \subseteq E$, $(V, M)$ has max degree 1
- Maximum matching: Matching $M^*$ of the largest size
Streaming Setting

Continuous Data Streams → Memory
Streaming Setting

- $G = (V, E)$
- Edges of $G$ appear in a stream
- Dynamic Stream: Insertions or Deletions
Streaming Setting

- $G = (V, E)$
- Edges of $G$ appear in a stream
- Dynamic Stream: Insertions or Deletions

![Graph Diagram]
Streaming Setting

- \( G = (V, E) \)
- Edges of \( G \) appear in a stream
- Dynamic Stream: Insertions or Deletions

\[
\begin{align*}
\text{\circle{}} - \text{\circle{}} \\
\text{\circle{}} - \text{\circle{}} \\
\text{\circle{}} - \text{\circle{}}
\end{align*}
\]
Streaming Setting

- \( G = (V, E) \)
- Edges of \( G \) appear in a stream
- Dynamic Stream: Insertions or Deletions

Diagram:

```
  O---O
 |
 O---O
 |
 O   O
```
Streaming Setting

- $G = (V, E)$
- Edges of $G$ appear in a stream
- Dynamic Stream: Insertions or Deletions
Streaming Setting

- \( G = (V, E) \)
- Edges of \( G \) appear in a stream
- Dynamic Stream: Insertions or Deletions
Streaming Setting

- $G = (V, E)$
- Edges of $G$ appear in a stream
- Dynamic Stream: Insertions or Deletions
Streaming Setting

- \( G = (V, E) \)
- Edges of \( G \) appear in a stream
- Dynamic Stream: Insertions or Deletions

![Diagram of a graph with edges]

Vihan Shah
Dynamic Streaming Matching
February 1, 2022
Streaming Setting

- $G = (V, E)$
- Edges of $G$ appear in a stream
- Dynamic Stream: Insertions or Deletions
- Output a solution at the end of the stream
- Goal: Minimize Memory
Introduction

Lower Bound

- Maximum Matching Lower bound: $\Omega(n^2)$ bits [FKM+05]
- Store the input: $O(n^2)$ bits
- No non-trivial solution
Approximation

- Question: What about an $\alpha$ approximation?
- Return a matching $M$ of size at least $\frac{|M^*|}{\alpha}$
- Can we get $o(n^2)$ space?
- What is the trade off between $\alpha$ and the space?
## Previous Work

| Result  | Upper Bound      | Lower Bound      |
|---------|------------------|------------------|
| [Kon15] | $O(n^2/\alpha^2)$ | $\Omega(n^{1.5}/\alpha^4)$ |

**Space-Approximation Tradeoff**

Gap: $\alpha^2 \cdot n^{0.5}$
## Previous Work

| Result      | Upper Bound     | Lower Bound                  |
|-------------|-----------------|-----------------------------|
| [Kon15]     | $O(n^2/\alpha^2)$ | $\Omega(n^{1.5}/\alpha^4)$ |
| [AKLY16]    | $\tilde{O}(n^2/\alpha^3)$ | $\Omega(n^{2-o(1)}/\alpha^3)$ |

**Space-Approximation Tradeoff**

[AKLY16] $\tilde{O}(n^2/\alpha^3)$

Gap: $n^{o(1)}$
## Previous Work

| Result   | Upper Bound   | Lower Bound        |
|----------|---------------|--------------------|
| [Kon15]  | $O(n^2/\alpha^2)$ | $\Omega(n^{1.5}/\alpha^4)$ |
| [AKLY16] | $\tilde{O}(n^2/\alpha^3)$ | $\Omega(n^{2-o(1)}/\alpha^3)$ |
| [CCE+16] | $\tilde{O}(n^2/\alpha^3)$ |                   |

**Gap:** $n^{o(1)}$  

**Space-Approximation Tradeoff**

1. ![Diagram showing the tradeoff between space and approximation error](diagram.png)
## Previous Work

| Result        | Upper Bound       | Lower Bound            |
|---------------|-------------------|------------------------|
| [Kon15]       | $O(n^2/\alpha^2)$ | $\Omega(n^{1.5}/\alpha^4)$ |
| [AKLY16]      | $\tilde{O}(n^2/\alpha^3)$ | $\Omega(n^{2-o(1)}/\alpha^3)$ |
| [CCE+16]      | $\tilde{O}(n^2/\alpha^3)$ |                         |
| [DK20]        | $\Omega(n^2/\alpha^3)$ | $\tilde{O}(n^2/\alpha^3)$ |

**Space-Approximation Tradeoff**

[DK20]: $\Omega(n^2/\alpha^3) \rightarrow \tilde{O}(n^2/\alpha^3)$

[AKLY16]: $\tilde{O}(n^2/\alpha^3) \rightarrow \Omega(n^{2-o(1)}/\alpha^3)$

Gap: polylog($n$)
Previous work

- Best known upper bound: $\tilde{O}(n^2/\alpha^3)$ bits ([AKLY16])
- Best known lower bound: $\Omega(n^2/\alpha^3)$ bits ([DK20])
- Gap of $\text{polylog}(n)$ bits
- These types of $\text{polylog}(n)$ gaps appear frequently in dynamic streams
- One key reason is a main technique for finding edges in a dynamic streams
Previous work

$L_0$-Samplers:

- It is **non-trivial** to find even one edge in a dynamic stream

- $L_0$-Samplers are a **key tool** to solve this problem

- They can sample an edge uniformly at random from a set of pairs of vertices undergoing edge insertions and deletions
Previous work

- $L_0$-Samplers can be implemented in $O(\log^3 n)$ bits of space (\cite{JST11})
- $\Omega(\log^3 n)$ bits are also necessary (\cite{Kap+17})
- Many problems in streaming have the polylog($n$) overhead because of the use of $L_0$-samplers
- Connectivity has a lower bound of $\Omega(n \log^3 n)$ (\cite{NY19})
Our Result

We prove asymptotically optimal bounds on the space-approximation tradeoff:

There is a dynamic streaming algorithm that with high probability outputs an \( \alpha \)-approximation to maximum matching using \( O\left(\frac{n^2}{\alpha^3}\right) \) bits of space for any \( \alpha \ll n^{1/2} \).

This closes the gap up to constant factors.

Some problems do not need the \( \text{polylog}(n) \) overhead:

If \( \alpha > n^{1/2} \) then there is not enough space to output the answer:

\[ n^\alpha > n^{2/3} \]
Our Result

We prove asymptotically optimal bounds on the space-approximation tradeoff:

**Result**

*There is a dynamic streaming algorithm that with high probability outputs an $\alpha$-approximation to maximum matching using $O(n^2/\alpha^3)$ bits of space for any $\alpha \ll n^{1/2}$.*
Our Result

We prove asymptotically optimal bounds on the space-approximation tradeoff:

**Result**

There is a dynamic streaming algorithm that with high probability outputs an $\alpha$-approximation to maximum matching using $O(n^2/\alpha^3)$ bits of space for any $\alpha \ll n^{1/2}$

This closes the gap up to constant factors

Some problems do not need the $\text{polylog}(n)$ overhead
Our Result

We prove asymptotically optimal bounds on the space-approximation tradeoff:

**Result**

There is a dynamic streaming algorithm that with high probability outputs an $\alpha$-approximation to maximum matching using $O(\frac{n^2}{\alpha^3})$ bits of space for any $\alpha \ll n^{1/2}$.

This closes the gap up to constant factors.

Some problems do not need the polylog($n$) overhead.

If $\alpha > n^{1/2}$ then there is not enough space to output the answer:

$$\frac{n}{\alpha} > \frac{n^2}{\alpha^3}$$
We will now give a proof sketch
Assumptions

Simplifying Assumptions for this talk:

- The input graph is bipartite
- The maximum matching has size $\Omega(n)$
- Getting an $\Theta(\alpha)$ approximation is enough
Assumptions

Simplifying Assumptions for this talk:

- The input graph is bipartite
- The maximum matching has size $\Omega(n)$
- Getting an $\Theta(\alpha)$ approximation is enough

All these assumptions can be lifted!
Approach

1. Match or Sparsify:
   - Either find a large matching
   - Or identify hard instances

2. Solve the hard instances

Note: We run these algorithms in parallel
Find a matching $M_{\text{easy}}$ in space $O(n^2/\alpha^3)$ bits such that:

- Either $|M_{\text{easy}}| = \Omega(n/\alpha)$
Find a matching $M_{\text{easy}}$ in space $O(n^2/\alpha^3)$ bits such that:

- Either $|M_{\text{easy}}| = \Omega(n/\alpha)$
- Or Subgraph induced on unmatched vertices has $\tilde{O}(n)$ edges and a matching of size $\Omega(n)$
Match Or Sparsify

Idea:

- Sample $O(n^2/\alpha^3 \text{polylog}(n))$ random edges

- $L_0$-samplers take space $\text{polylog}(n)$

- $M_{\text{easy}}$ is a greedy matching over the sampled edges

- Similar to residual greedy property of matching (used in [Ahn+18, Kon18])

- Different proof but along the same lines
We know the partition at the end of the stream from Match Or Sparsify step

\[ |M_{\text{easy}}| < \frac{n}{\alpha} \]

\[ \tilde{O}(n) \text{ edges} \]
Consider the bipartite graph

\[ n \quad \bigcirc \quad \bigcirc \quad n \]

\[ \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \]

\[ \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \]

\[ |M_{\text{easy}}| < \frac{n}{\alpha} \]

\[ O(n) \text{ edges} \]
Grouping

Random grouping on both sides

\[ \frac{n}{\alpha} \quad \text{or} \quad \frac{n}{\alpha} \]

\[ \begin{array}{c}
\text{\(\tilde{O}(n)\) edges} \\
|M_{\text{easy}}| < \frac{n}{\alpha}
\end{array} \]
Grouping

$\frac{1}{\alpha}$ fraction of groups on right are in the neighborhood of $V_i$

Done to reduce the neighbors of $V_i$
Recovery

- There are $\Omega(n/\alpha)$ pairs of groups with exactly one edge between them.

- $V_i, V_j$ do not contain any vertices of $M_{\text{easy}}$.
Recovery

Want to recover the edge between $V_i$ and $V_j$

$|M_{easy}| < n/\alpha$

$\tilde{O}(n)$ edges
Recovery

- $V_i$ does not contain any vertices of $M_{\text{easy}}$
- Neighbors of $V_i$: $O(n/\alpha^2)$
- Trivial solution: $O((n/\alpha^2) \cdot \log n)$ bits
Recovery

- Goal: $O(n/\alpha^2)$ bits
- So $n/\alpha$ groups will imply space of $O(n^2/\alpha^3)$ bits
- $V_j$ does not contain any vertices of $M_{\text{easy}}$
- Recover $N(V_i) - M_{\text{easy}}$

$V_i \overset{\text{edges}}{\longrightarrow} V_j$

$|M_{\text{easy}}| < n/\alpha$

$\tilde{O}(n)$ edges
Sparse neighborhood recovery sketch

- Given $V_i$ at the beginning
- Given $M_{\text{easy}}$ at the end
- Output: $N(V_i) - M_{\text{easy}}$
- Space: $O(n/\alpha^2)$ bits
Grouping

$V_j$ lies completely within $N(V_i) - M_{easy}$
Recovery

- We know \( u \) is a neighbor of \( V_i \) (from Neighborhood sketch of \( V_i \))
- We know \( v \) is a neighbor of \( V_j \) (from Neighborhood sketch of \( V_j \))
- Thus, \((u, v)\) must be an edge
Summary

Concluding Remarks
Summary

- There is a dynamic streaming algorithm that whp outputs an $\alpha$-approximation to maximum matching using $O(n^2/\alpha^3)$ bits of space.

[DK20] refers to a specific paper, and [NY19] is another paper. The overhead of $L_0$-samplers is not always necessary, unlike [NY19].
Summary

- There is a dynamic streaming algorithm that whp outputs an $\alpha$-approximation to maximum matching using $O(n^2/\alpha^3)$ bits of space.

- The lower bound of [DK20] is $\Omega(n^2/\alpha^3)$ bits making our algorithm optimal.
Summary

- There is a dynamic streaming algorithm that w.h.p outputs an $\alpha$-approximation to maximum matching using $O(n^2/\alpha^3)$ bits of space.

- The lower bound of [DK20] is $\Omega(n^2/\alpha^3)$ bits making our algorithm optimal.

- $\text{polylog}(n)$ overhead of $L_0$-samplers is not always necessary (Unlike [NY19]).
Open Problems

- These $\text{polylog}(n)$ overheads due to use of $L_0$-samplers are prevalent in dynamic stream literature.

- Can our techniques be used to bypass $\text{polylog}(n)$ overheads for other problems:
  - E.g. Vertex Cover, Dominating Set, Vertex Connectivity
Open Problems

- These $\text{polylog}(n)$ overheads due to use of $L_0$-samplers are prevalent in dynamic stream literature.

- Can our techniques be used to bypass $\text{polylog}(n)$ overheads for other problems:
  - E.g. Vertex Cover, Dominating Set, Vertex Connectivity

Thank you!