An Asymptotically Optimal Algorithm for Maximum Matching in Dynamic Streams

Vihan Shah

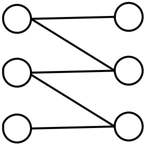
Department of Computer Science
Rutgers University

January 21, 2022

Joint work with Sepehr Assadi

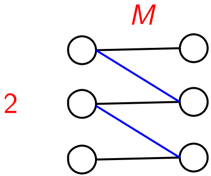
Matching Problem

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



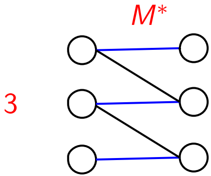
Matching Problem

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

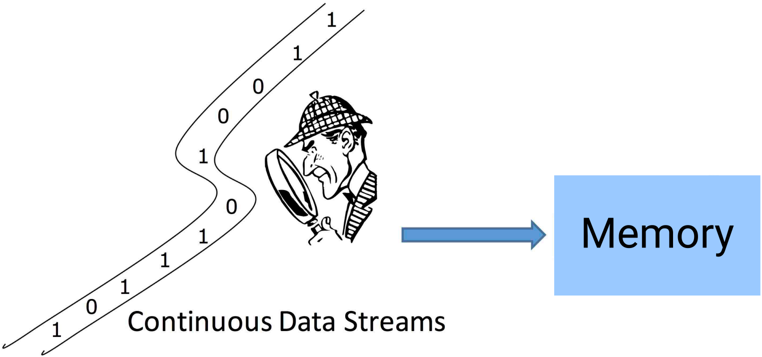


Matching Problem

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



Streaming Setting

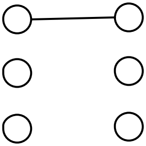


Streaming Setting

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

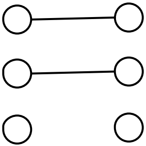
Streaming Setting

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



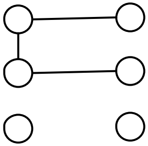
Streaming Setting

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



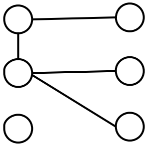
Streaming Setting

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



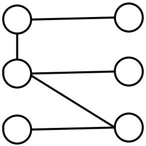
Streaming Setting

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



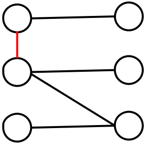
Streaming Setting

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



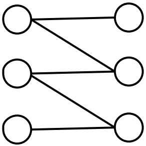
Streaming Setting

- $G = (V, E)$
- 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



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

Lower Bound

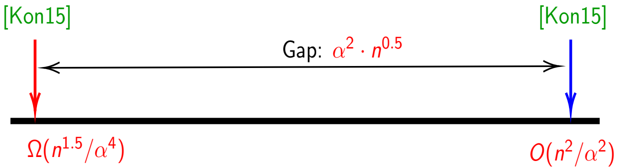
- Maximum Matching Lower bound: $\Omega(n^2)$ bits [FKM+05]
- Store the input: $O(n^2)$ bits
- No non-trivial solution

Approximation

- Question: What about an α 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 α and the space?

Previous Work

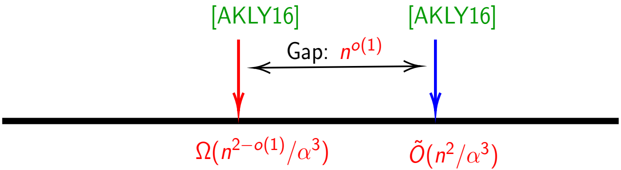
Result	Upper Bound	Lower Bound
[Kon15]	$O(n^2/\alpha^2)$	$\Omega(n^{1.5}/\alpha^4)$



Space-Approximation Tradeoff

Previous Work

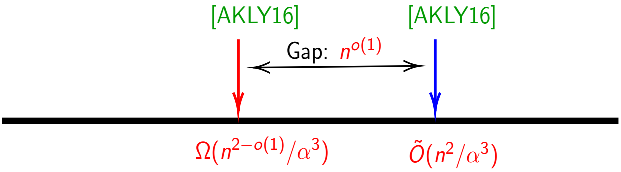
Result	Upper Bound	Lower Bound
[Kon15]	$O(n^2/\alpha^2)$	$\Omega(n^{1.5}/\alpha^4)$
[AKLY16]	$\tilde{O}(n^2/\alpha^3)$	$\Omega(n^{2-o(1)}/\alpha^3)$



Space-Approximation Tradeoff

Previous Work

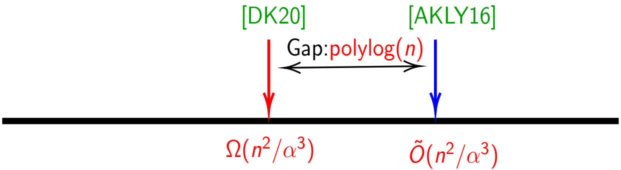
Result	Upper Bound	Lower Bound
[Kon15]	$O(n^2/\alpha^2)$	$\Omega(n^{1.5}/\alpha^4)$
[AKLY16]	$\tilde{O}(n^2/\alpha^3)$	$\Omega(n^{2-o(1)}/\alpha^3)$
[CCE+16]	$\tilde{O}(n^2/\alpha^3)$	



Space-Approximation Tradeoff

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]		$\Omega(n^2/\alpha^3)$



Space-Approximation Tradeoff

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 $\mathrm{polylog}(n)$ bits
- These types of $\mathrm{polylog}(n)$ 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 [JST11]
- $\Omega(\log^3 n)$ bits are also necessary [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 $\Omega(n \log^3 n)$ ([NY19])

Our Result

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

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 α-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 α-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 α-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 polylog(n) overhead

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

$$\frac{n}{\alpha} > \frac{n^2}{\alpha^3}$$

Algorithm

We will now show how to prove this!

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

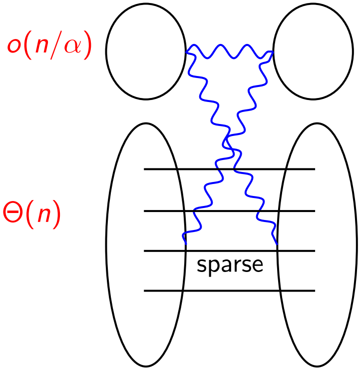
Simplifying Assumptions for this talk:

- 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!

Hard Instances

A hard instance from previous work [Kon15, AKLY16, DK20]:



Lower Bound [DK20]: $\Omega(n^2/\alpha^3)$ bits

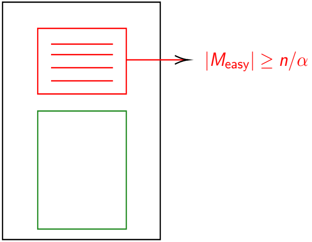
Approach

1. Match or Sparsify:
   - Either find a large matching
   - Or identify hard instances similar to hard instances of previous work
2. Solve the hard instances

Note: We run these algorithms in parallel

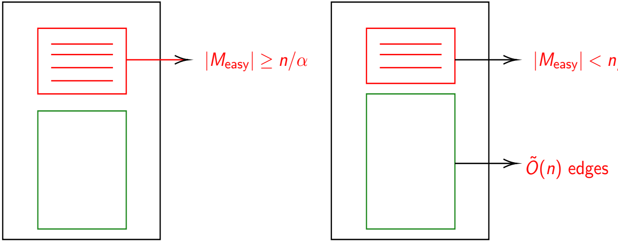
Match Or Sparsify

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



Match Or Sparsify

1. Find a matching M_{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 $\Omega(n)$



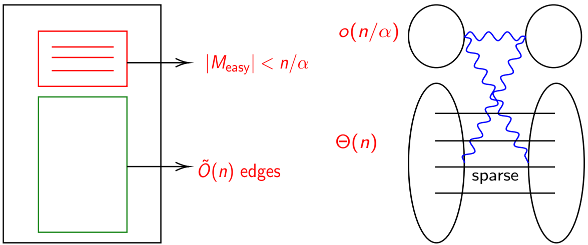
Match Or Sparsify

Idea:

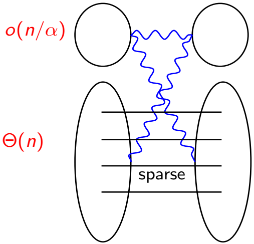
- Sample $O(n^2/\alpha^3 \text{polylog}(n))$ random edges
- L_0-samplers take space $\text{polylog}(n)$
- M_{easy} is a greedy matching over the sampled edges
- Similar to residual greedy property of matching (used in [Ahn+18, Kon18])
- Different proof but along the same lines

Similarity to Hard Instances

The instances we focus on are qualitatively same as the hard instances



Solving Hard Instances

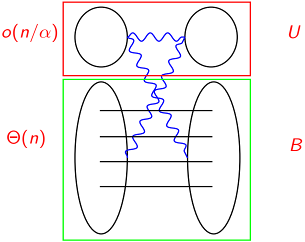


Analysis of [DK20]:

- We need n^2/α^3 edges
- Space: $O((n^2/\alpha^3) \cdot \log(n))$ bits
- L_0-samplers: $O((n^2/\alpha^3) \cdot \text{polylog}(n))$ bits

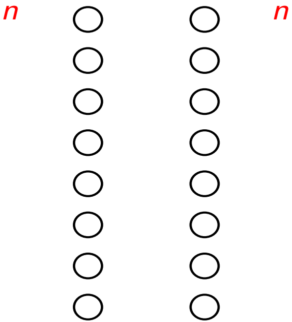
Solving Hard Instances

We know the partition U, B at the end of the stream from Match Or Sparsify step



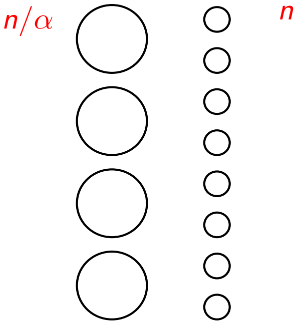
Grouping

Consider the bipartite graph



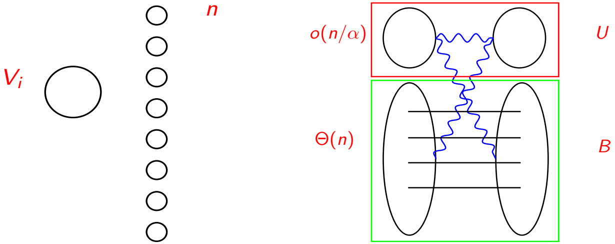
Grouping

Partition left randomly into groups of size α



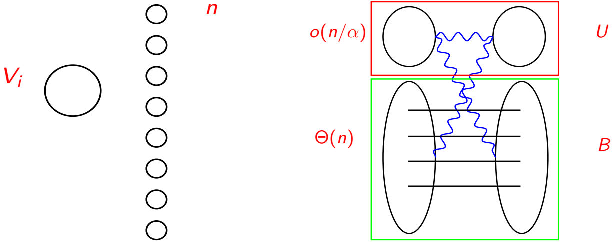
Grouping

V_i lies within B with probability $1 - o(1)$



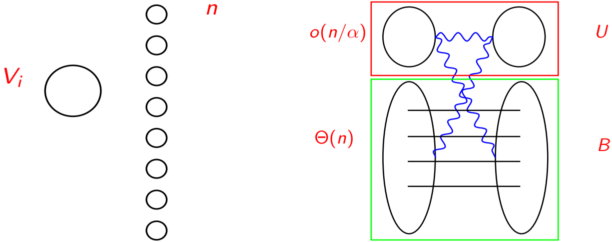
Grouping

Focus on group V_i that lies within B



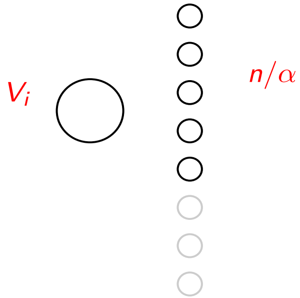
Grouping

V_i has α edges to B; We just need one edge;



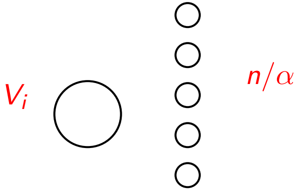
Grouping

$1/\alpha$ fraction of vertices on right are in the neighborhood of V_i



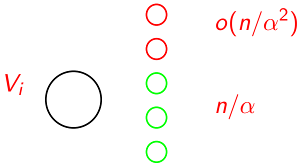
Grouping

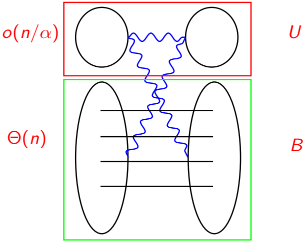
V_i has n/α vertices in its neighborhood



Grouping

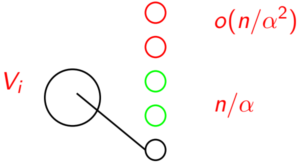
$o(n/\alpha^2)$ from U; n/α from B;

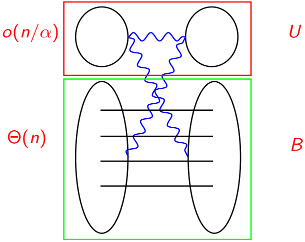




Grouping

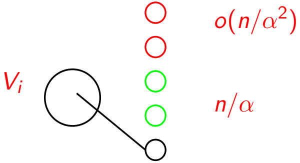
V_i has just 1 edge in B

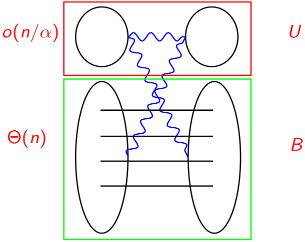




Grouping

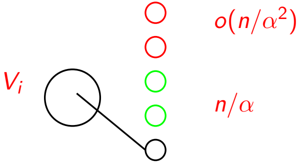
Can we find this one neighbor efficiently?

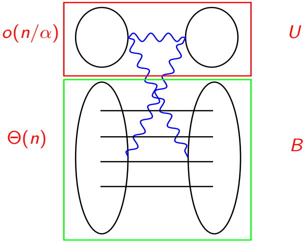




Grouping

- This is like the set disjointness problem from communication complexity
- Need to find a vertex that has an edge from V_i and is from B

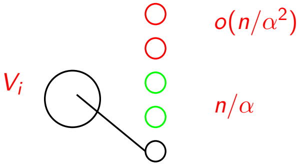


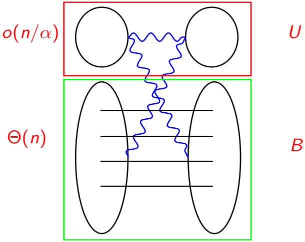


Recovery

Need to find a vertex that is from B and also has an edge from V_i

- Trivial solution: $O(n \log n/\alpha^2)$ bits
- Goal: $O(n/\alpha^2 + \log n)$ bits
- So n/α groups will imply space of $O(n^2/\alpha^3)$ bits





Recovery

Idea:

- Represent the neighborhood of V_i as a binary vector
- Compute inner products with random vectors

Recovery

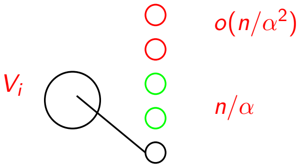
Idea:

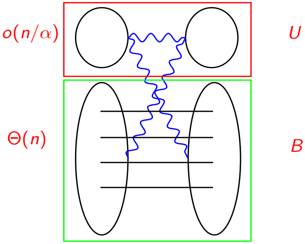
- Represent the neighborhood of V_i as a binary vector
- Compute inner products with random vectors
- Recovery: Go over all possible neighbor vectors and check if the inner products match

Recovery

Idea:

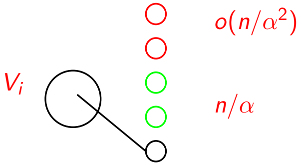
- Number of possible neighbor vectors: $2^{o(n/\alpha^2)} \cdot n$
- Space: $O(n/\alpha^2 + \log n)$ bits





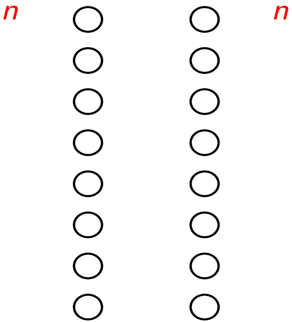
Issues

- We can find the neighbor of V_i
- But we do not know the name of the endpoint in V_i
- Cannot recover an edge
- We need grouping on the right too



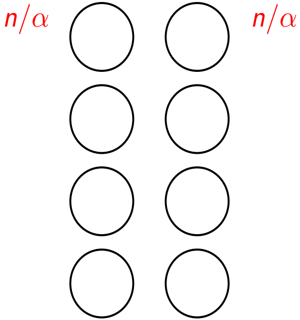
Grouping

Consider the bipartite graph



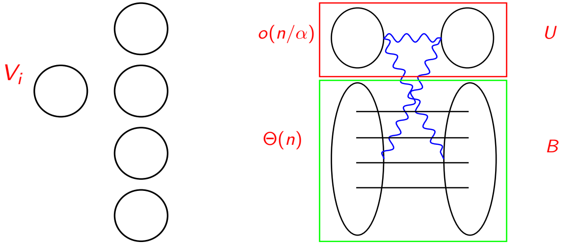
Grouping

Random grouping on both sides



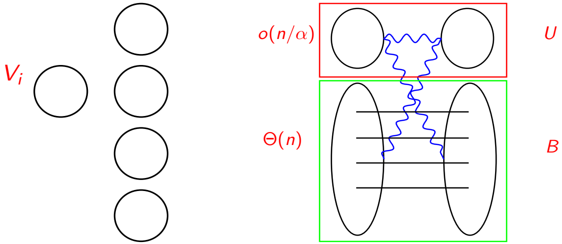
Grouping

V_i lies within B with probability $1 - o(1)$



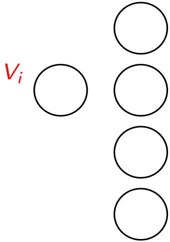
Grouping

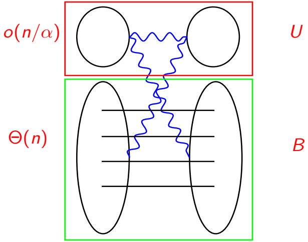
Focus on group V_i that lies within B



Grouping

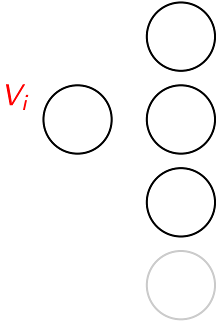
V_i has α edges to B; We just need one edge;





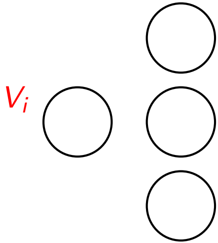
Grouping

$1/\alpha$ fraction of groups on right are in the neighborhood of V_i



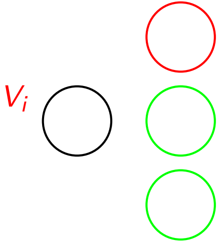
Grouping

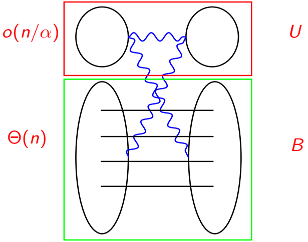
V_i has n/α^2 groups in its neighborhood



Grouping

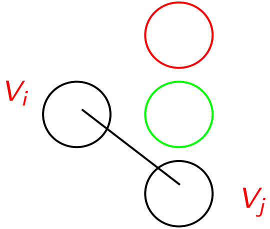
The green groups lie completely within B

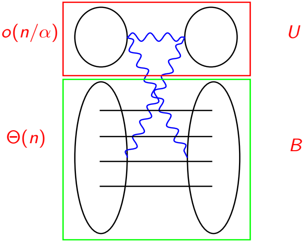




Grouping

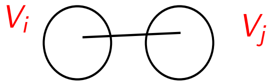
V_i has an edge to V_j





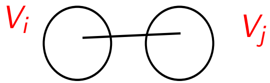
Recovery

- There may be multiple edges between V_i and V_j
- But there is just one edge between them with high constant probability



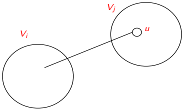
Grouping

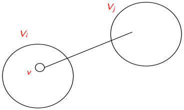
Want to recover the edge between V_i and V_j



Recovery

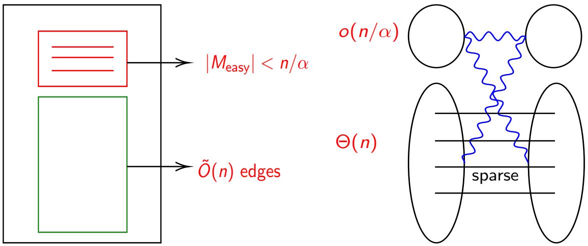
- We know u is a neighbor of V_i (from Neighborhood sketch of V_i)
- We know v is a neighbor of V_j (from Neighborhood sketch of V_j)
- Thus, (u, v) must be an edge





Challenges

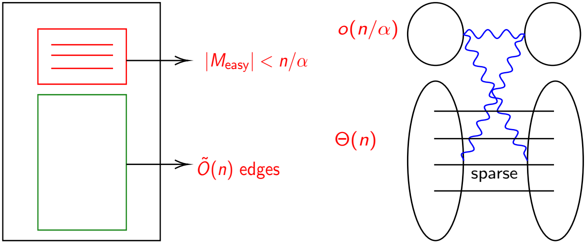
We need to solve a more general problem



Challenges

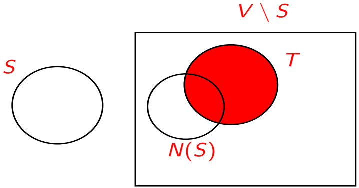
Challenges:

- $\tilde{O}(n)$ edges
- Cannot bound the degree of vertices with a constant



Sparse Neighborhood Recovery

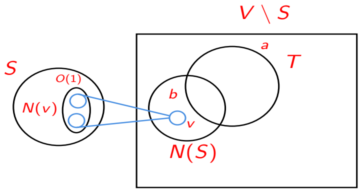
- $G = (V, E)$ specified in a dynamic stream
- $S \subseteq V$ known before the stream
- $T \subseteq V$ revealed after the stream
- Goal: Return $N(S) - T$



Sparse Neighborhood Recovery

Promises:

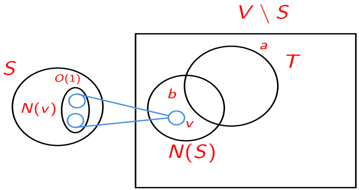
1. $|T| \leq a$;
2. $|N(S) - T| \leq b$;
3. for every vertex $v \in N(S) - T$, $|S \cap N(v)| = O(1)$



Sparse Neighborhood Recovery

Space:

1. Trivial solution: $O(a \log n + b \log n)$ bits
2. Goal: $O(a + b \log n)$ bits



Sparse Neighborhood Recovery

Solution:

1. We can solve the problem using previous ideas of inner products
2. Problems:
   - Exponential time for recovery
   - Random bits needed is much more than space budget

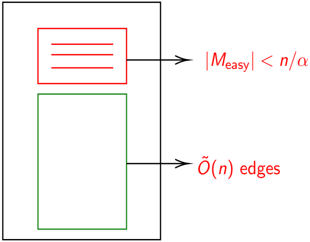
Sparse Neighborhood Recovery

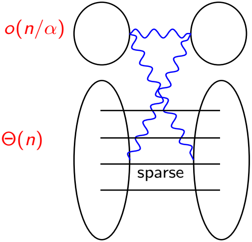
Solution:

1. We can solve the problem using previous ideas of inner products
2. Problems:
   - Exponential time for recovery
   - Random bits needed is much more than space budget
3. Solution using ideas from sparse recovery (complicated)
4. Space bound: $O(a + b \log n)$ bits
5. This bound is information-theoretically optimal

Solving General Hard Instance

- Using sparse neighborhood recovery sketch we can solve the general hard instance
- Space: $O(n^2/\alpha^3)$ bits





Summary

Concluding Remarks

Summary

1. Match or Sparsify: In $O(n^2/\alpha^3)$ bits of space
   - We either get a large matching
   - Or get a hard instance that is sparse and contains a large matching
2. Our sparse recovery sketches can be used to solve these hard instances in $O(n^2/\alpha^3)$ bits
3. We run both algorithms in parallel and get the final algorithm

Summary

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

Summary

- There is a dynamic streaming algorithm that whp outputs an α-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 whp outputs an α-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
- $\mathrm{polylog}(n)$ overhead of L_0-samplers is not always necessary (Unlike [NY19])

Open Problems

- These polylog(n) overheads due to use of L_0-samplers are prevalent in dynamic stream literature
- Can our techniques be used to bypass polylog(n) overheads for other problems:
  - E.g. Vertex Cover, Dominating Set, Vertex Connectivity

Open Problems

- These polylog(n) overheads due to use of L_0-samplers are prevalent in dynamic stream literature
- Can our techniques be used to bypass polylog(n) overheads for other problems:
  - E.g. Vertex Cover, Dominating Set, Vertex Connectivity

Thank you!