Tight Bounds for Linear Sketches of Approximate Matchings

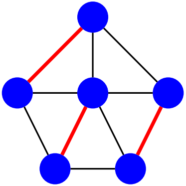
Sepehr Assadi

University of Pennsylvania

Joint work with Sanjeev Khanna (Penn),Yang Li (Penn), and Grigory Yaroslavtsev (Penn)

Matchings in Graphs

- Matching: A collection of vertex-disjoint edges.



- Perfect Matching: Every vertex is in the matching.

Maximum Matching problem: Find a matching with a largest number of edges.

Matchings in Graphs

Maximum matching is a fundamental problem with many applications.

- Many celebrated algorithms for matchings: Ford-Fulkerson, Edmond's, Hopcroft-Karp, Mucha-Sankowski, Madry's, ...
- Studied in various computational models: distributed, dynamic, online, streaming, ...

This talk: sublinear space algorithms for computing approximate matchings in dynamic graph streams.

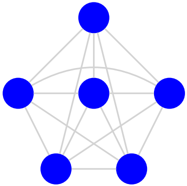
Dynamic Graph Streams

- The input graph is presented as a sequence of edge insertions and deletions.

Stream:

Edge-frequency vector:

$\vec{f} = \left[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\right]$



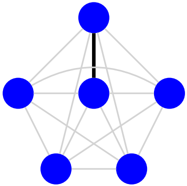
Dynamic Graph Streams

- The input graph is presented as a sequence of edge insertions and deletions.

Stream: $+e_1$

Edge-frequency vector:

$$\vec{f} = \left[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\right]$$



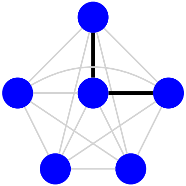
Dynamic Graph Streams

- The input graph is presented as a sequence of edge insertions and deletions.

Stream: $+e_1, +e_7$

Edge-frequency vector:

$$\vec{f} = \left[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0\right]$$



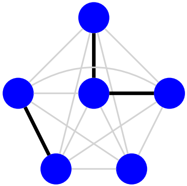
Dynamic Graph Streams

- The input graph is presented as a sequence of edge insertions and deletions.

Stream: $+e_1, +e_7, +e_{11}$

Edge-frequency vector:

$$\vec{f} = \left[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0\right]$$



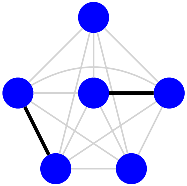
Dynamic Graph Streams

- The input graph is presented as a sequence of edge insertions and deletions.

Stream: $+e_1, +e_7, +e_{11}, -e_1$

Edge-frequency vector:

$\vec{f} = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0]$



Dynamic Graph Streams

- The input graph is presented as a sequence of edge insertions and deletions.
- Algorithm makes a single pass over the entire input but only has a small space to store information about the input as it passes by.
- At the end of the sequence, the algorithm outputs a solution using the stored information.

Linear Sketches

For a graph G with n vertices:

- Let $\vec{f}$ be the edge-frequency vector representing G.
- Let M be an $s \times n^2$ dimensional matrix (possibly randomly chosen) for some parameter s.
- The s-dimensional vector $M \cdot \vec{f}$ is a linear sketch of G:

$$\Big[\quad M \quad \Big]_{s \times n^2} \cdot \Big[\vec{f} \Big]_{n^2} = \Big[\; \Big]_{s}$$

- Requires $O(s)$ for storage $\implies$ $O(s)$ size for storing the graph instead of $O(n^2)$.

Linear Sketches and Dynamic Graph Streams

Linear sketches are main technique for computing in dynamic graph streams:

- Maintain a linear sketch of the input graph during the stream.
  - When an edge e_i is updated: $M \cdot (\overrightarrow{f} \pm \overrightarrow{e_i}) = M \cdot \overrightarrow{f} \pm M \cdot \overrightarrow{e_i}$
- At the end of the stream, apply an arbitrary function to $M \cdot \overrightarrow{f}$ to compute the answer.
- Space requirement of the algorithm:
  $O(s)$ for the linear sketch + random bits needed for storing M implicitly.

Dynamic graph stream algorithms and linear sketches are (essentially) equivalent [AHLW16, LNW14].

Results in Dynamic Graph Streams

Linear sketches proved to be useful for various graph problems:

- Connectivity, edge connectivity, minimum spanning tree, spectral sparsification, triangle counting, densest subgraph, . . .
- Most of them have essentially the same space requirement as the best streaming algorithm for insertion-only streams.

An important missing problem is the maximum matching problem.

Matching in Graph Streams

Insertion-only streams:

- Exact computation requires $\Omega(n^2)$ space [FKM+05].
- 2-approximation in $O(n)$ space is trivial but no better than 2-approximation in $o(n^2)$ space is known.
- Beating $\frac{e}{e-1}$-approximation requires $n^{1+\Omega(1/\log\log n)}$ space [Kap13, GKK12].
- Lots and lots of other results: [McG05] [FKM+05] [EKS09] [ELMS11] [GKK12] [KMM12] [Zel12] [AGM12] [AG13b] [Kap13] [GO13] [KKS14] [CS14] [EHL+15] [AG13a] ...

Dynamic graph streams:

- Prior to our work, no non-trivial results were known for single-pass algorithms.

Our Results

We provide a complete resolution of matchings in dynamic graph streams:

Theorem (Upper bound)

For any $0 \leq \epsilon \leq 1/2$, space of $\tilde{O}(n^{2-3\epsilon})$ is sufficient for computing an n^{ϵ}-approximate maximum matching.

Our Results

We provide a complete resolution of matchings in dynamic graph streams:

Theorem (Upper bound)

For any $0 \leq \epsilon \leq 1/2$, space of $\tilde{O}(n^{2-3\epsilon})$ is sufficient for computing an n^{ϵ}-approximate maximum matching.

Theorem (Lower bound)

For any $\epsilon \geq 0$, space of $\tilde{\Omega}(n^{2-3\epsilon})$ is necessary for computing an n^{ϵ}-approximate maximum matching.

Our Results

We provide a complete resolution of matchings in dynamic graph streams:

Theorem (Upper bound)

For any $0 \leq \epsilon \leq 1/2$, space of $\tilde{O}(n^{2-3\epsilon})$ is sufficient for computing an n^{ϵ}-approximate maximum matching.

Theorem (Lower bound)

For any $\epsilon \geq 0$, space of $\tilde{\Omega}(n^{2-3\epsilon})$ is necessary for computing an n^{ϵ}-approximate maximum matching.

For $\epsilon > 1/2$, $\tilde{O}(n^{1-\epsilon})$ space is necessary and sufficient for an n^{ϵ}-approximation.

Recent Related Work

Two recent results obtained independently and concurrently:

	Upper bound	Lower bound
[Kon15]	$\tilde{O}(n^{2-2\epsilon})$	$\tilde{\Omega}(n^{3/2-4\epsilon})$
[CCE+16]	$\tilde{O}(n^{2-3\epsilon})$ $(\epsilon \leq 1/2)$	-
This work	$\tilde{O}(n^{2-3\epsilon})$ $(\epsilon \leq 1/2)$ $\tilde{O}(n^{1-\epsilon})$ $(\epsilon > 1/2)$	$\tilde{\Omega}(n^{2-3\epsilon})$

Upper Bound

n^ϵ-Approximation for Matchings

Theorem (Upper bound)

For any $0 < \epsilon \leq 1/2$, space of $\tilde{O}(n^{2-3\epsilon})$ is sufficient for computing an n^ϵ-approximate maximum matching in dynamic graph streams.

- The algorithm needs only to store a linear sketch.
- W.l.o.g. we can restrict our attention to bipartite graphs.
- For simplicity, assume there is a perfect matching $M^\star$ in the input graph $G(L, R, E)$.

ℓ_0-sampler

Problem: How can we recover one edge from the edge-frequency vector $\vec{f}$ of G defined by a dynamic graph stream in a small space?

ℓ_0-sampler

Problem: How can we recover one edge from the edge-frequency vector $\overrightarrow{f}$ of G defined by a dynamic graph stream in a small space?

Toy problem 1: What if we are promised that at the end of the stream, there is exactly one edge in G, i.e., $\|\overrightarrow{f}\|_0 = 1$?

ℓ_0-sampler

Problem: How can we recover one edge from the edge-frequency vector $\overrightarrow{f}$ of G defined by a dynamic graph stream in a small space?

Toy problem 1: What if we are promised that at the end of the stream, there is exactly one edge in G, i.e., $\|\overrightarrow{f}\|_0 = 1$?

Solution:

1. Let $M = \left[1, 2, \ldots, n^2\right]$.
2. Return $M \cdot \overrightarrow{f}$.

ℓ_0-sampler

Problem: How can we recover one edge from the edge-frequency vector $\overrightarrow{f}$ of G defined by a dynamic graph stream in a small space?

Toy problem 2: If there is exactly one edge in G return it, otherwise output FAIL.

ℓ_0-sampler

Problem: How can we recover one edge from the edge-frequency vector $\overrightarrow{f}$ of G defined by a dynamic graph stream in a small space?

Toy problem 2: If there is exactly one edge in G return it, otherwise output FAIL.

Solution:

1. Let $M = \begin{bmatrix} 1, 2, \ldots, n^2 \\ 1, 1, \ldots, 1 \end{bmatrix}$.
2. Let $\begin{bmatrix} x \\ y \end{bmatrix} = M \cdot \overrightarrow{f}$; if $y = 1$ return x, otherwise output FAIL.

ℓ_0-sampler

Problem: How can we recover one edge from the edge-frequency vector $\overrightarrow{f}$ of G defined by a dynamic graph stream in a small space?

Toy problem 3: Suppose there are exactly D edges in G, i.e., $\|\overrightarrow{f}\|_0 = D$; return one edge in G w.p. $2/3$, otherwise output FAIL.

ℓ_0-sampler

Problem: How can we recover one edge from the edge-frequency vector $\overrightarrow{f}$ of G defined by a dynamic graph stream in a small space?

Toy problem 3: Suppose there are exactly D edges in G, i.e., $\|\overrightarrow{f}\|_0 = D$; return one edge in G w.p. $2/3$, otherwise output FAIL.

Solution:

1. Randomly sample $\frac{D}{n^2}$ edge slots from G, i.e., $\frac{1}{D}$ fraction of rows in $\overrightarrow{f}$.
2. Run the algorithm from the previous part over the sub-sampled graph.

ℓ_0-sampler

Problem: How can we recover one edge from the edge-frequency vector $\overrightarrow{f}$ of G defined by a dynamic graph stream in a small space?

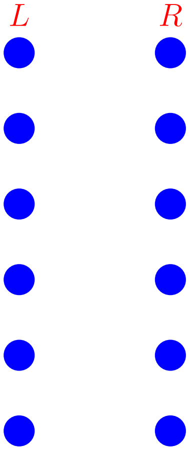
Theorem ([JST11])

There exists a distribution of over $\text{polylog}(N) \times N$ dimensional matrices M, such that for any $x \in \mathbb{R}^N$, one random non-zero element of x can be reconstructed from $M \cdot x$ w.h.p.

ℓ_0-samplers allow us to recover an edge between any two groups of pre-specified vertices in dynamic graph streams.

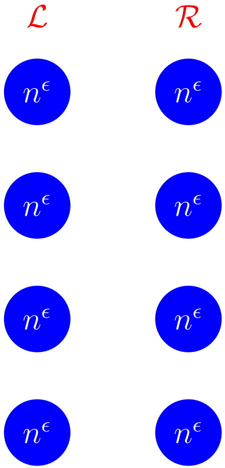
Warm-up: An $\tilde{O}(n^{2-2\epsilon})$ Space Algorithm

1. Group vertices in L and R into $n^{1-\epsilon}$ groups.



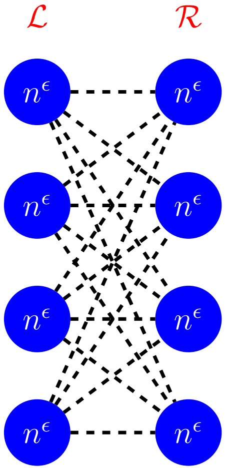
Warm-up: An $\tilde{O}(n^{2-2\epsilon})$ Space Algorithm

1. Group vertices in L and R into $n^{1-\epsilon}$ groups.



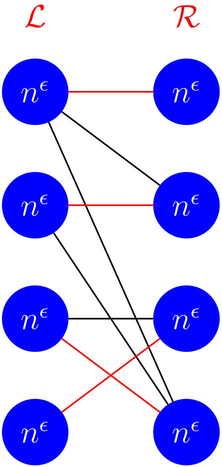
Warm-up: An $\tilde{O}(n^{2-2\epsilon})$ Space Algorithm

1. Group vertices in L and R into $n^{1-\epsilon}$ groups.
2. Maintain one ℓ_0-sampler between any group in $\mathcal{L}$ and any group in $\mathcal{R}$.



Warm-up: An $\tilde{O}(n^{2-2\epsilon})$ Space Algorithm

1. Group vertices in L and R into $n^{1-\epsilon}$ groups.
2. Maintain one ℓ_0-sampler between any group in $\mathcal{L}$ and any group in $\mathcal{R}$.
3. At the end of the stream, sample one edge from each ℓ_0-sampler and compute a maximum matching on sampled edges.



Analysis of the $\tilde{O}(n^{2-2\epsilon})$ Space Algorithm

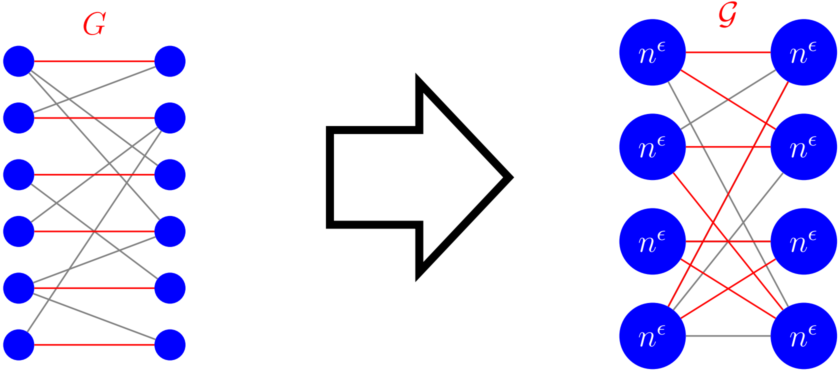
Space requirement:

- We picked $n^{2-2\epsilon}$ ℓ_0-samplers: one per each pair in $(\mathcal{L}, \mathcal{R})$.
- Each ℓ_0-sampler requires $\mathrm{polylog}(n)$ space.
- Total of $\tilde{O}(n^{2-2\epsilon})$ space.

Analysis of the $\tilde{O}(n^{2-2\epsilon})$ Space Algorithm

Approximation factor:

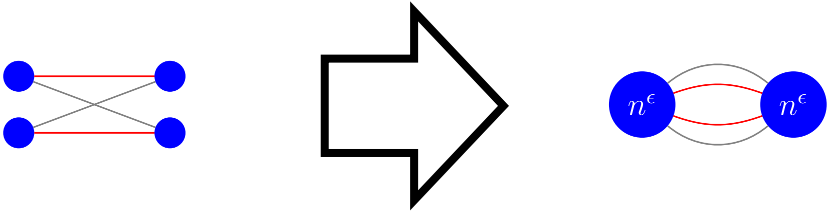
- The perfect matching $M^\star$ in G induces an n^ϵ-regular (multi-)graph in the grouped graph $\mathcal{G}$.



Analysis of the $\tilde{O}(n^{2-2\epsilon})$ Space Algorithm

Approximation factor:

- The perfect matching $M^\star$ in G induces an n^ϵ-regular (multi-)graph in the grouped graph $\mathcal{G}$.
- The ℓ_0-sampler between each matchable pair (connected by an edge in $M^\star$) in $\mathcal{G}$ returns an edge (not necessarily from $M^\star$).



Analysis of the $\tilde{O}(n^{2-2\epsilon})$ Space Algorithm

Approximation factor:

- The perfect matching $M^\star$ in G induces an n^ϵ-regular (multi-)graph in the grouped graph $\mathcal{G}$.
- The ℓ_0-sampler between each matchable pair (connected by an edge in $M^\star$) in $\mathcal{G}$ returns an edge (not necessarily from $M^\star$).
- The sampled edges have a matching of size $n^{1-\epsilon}$, i.e., an n^ϵ-approximate maximum matching.

Improving to an $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

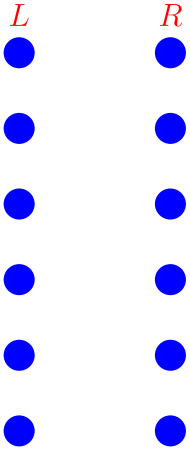
Insight:

- The graph $\mathcal{G}$ in the previous algorithm is an n^{ϵ}-regular (multi-)graph.
- Any n^{ϵ}-regular graph has n^{ϵ} edge-disjoint perfect matching.
- The previous algorithm focused only on a single perfect matching in $\mathcal{G}$.

Can we sub-sample edges of $\mathcal{G}$ while still maintaining a large matching in the sampled graph?

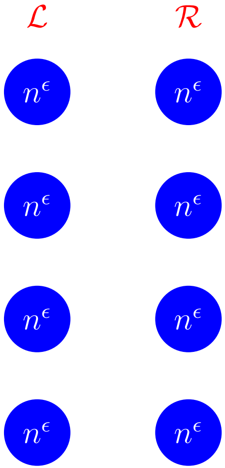
An $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

1. Randomly group vertices in L and R into $n^{1-\epsilon}$ groups.



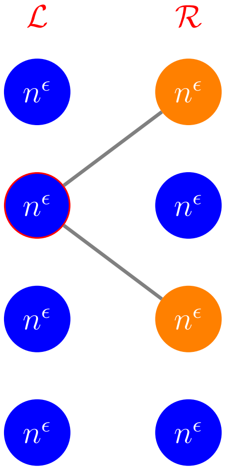
An $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

1. Randomly group vertices in L and R into $n^{1-\epsilon}$ groups.



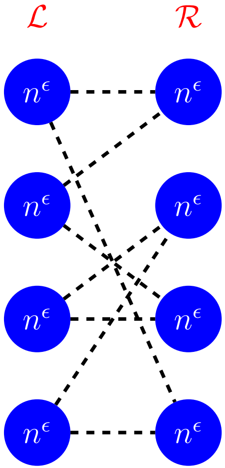
An $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

1. Randomly group vertices in L and R into $n^{1-\epsilon}$ groups.
2. For each group $L_i \in \mathcal{L}$, pick $n^{1-2\epsilon}$ partner group in $\mathcal{R}$ uniformly at random.



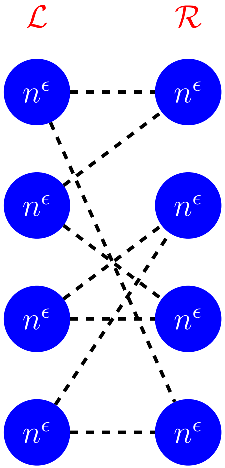
An $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

1. Randomly group vertices in L and R into $n^{1-\epsilon}$ groups.
2. For each group $L_i \in \mathcal{L}$, pick $n^{1-2\epsilon}$ partner group in $\mathcal{R}$ uniformly at random.
3. Maintain one ℓ_0-sampler between any two partner group in $\mathcal{L}$ and $\mathcal{R}$.



An $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

1. Randomly group vertices in L and R into $n^{1-\epsilon}$ groups.
2. For each group $L_i \in \mathcal{L}$, pick $n^{1-2\epsilon}$ partner group in $\mathcal{R}$ uniformly at random.
3. Maintain one ℓ_0-sampler between any two partner group in $\mathcal{L}$ and $\mathcal{R}$.
4. At the end of the stream, sample one edge from each ℓ_0-sampler and compute a maximum matching on sampled edges.



Analysis of the $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

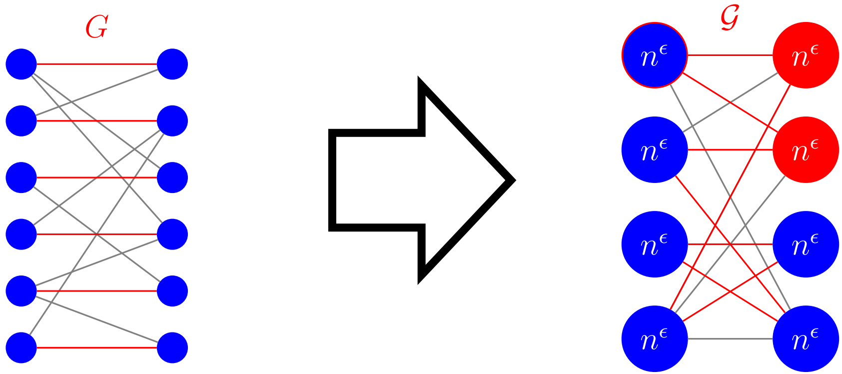
Space requirement:

- We picked $n^{1-\epsilon} \cdot n^{1-2\epsilon} = n^{2-3\epsilon}$ ℓ_0-samplers: one per each partner pair in $(\mathcal{L}, \mathcal{R})$.
- Each ℓ_0-sampler requires $\text{polylog}(n)$ space.
- Total of $\tilde{O}(n^{2-3\epsilon})$ space.

Analysis of the $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

Approximation factor:

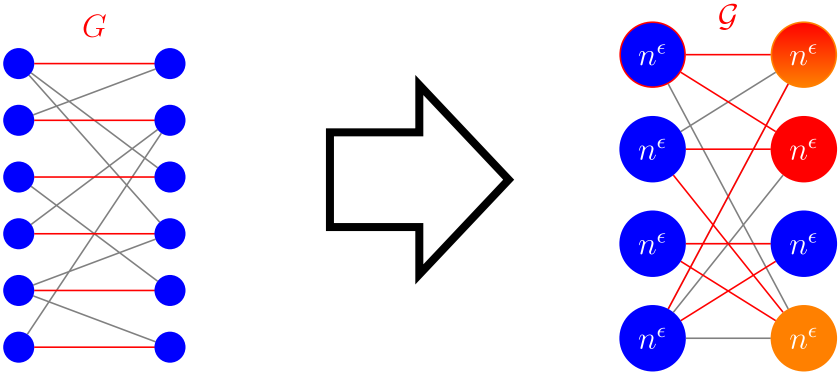
- For each $L_i \in \mathcal{L}$, $\Omega(n^{\epsilon})$ groups in $\mathcal{R}$ are matchable (connected by an edge in $M^{\star}$).



Analysis of the $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

Approximation factor:

- For each $L_i \in \mathcal{L}$, $\Omega(n^\epsilon)$ groups in $\mathcal{R}$ are matchable (connected by an edge in $M^\star$).
- For each $L_i \in \mathcal{L}$, one matchable group $R_j \in \mathcal{R}$ is a partner.



Analysis of the $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

Approximation factor:

- For each $L_i \in \mathcal{L}$, $\Omega(n^{\epsilon})$ groups in $\mathcal{R}$ are matchable (connected by an edge in $M^{\star}$).
- For each $L_i \in \mathcal{L}$, one matchable group $R_j \in \mathcal{R}$ is a partner.
- For each $L_i \in \mathcal{L}$, the matchable partner is chosen uniformly at random from all matchable groups.
- $\dots$
- The sampled edges have a matching of size $\Omega(n^{1-\epsilon})$, i.e., an $O(n^{\epsilon})$-approximate maximum matching.

Analysis of the $\tilde{O}(n^{2-3\epsilon})$ Space Algorithm

Approximation factor:

- For each $L_i \in \mathcal{L}$, $\Omega(n^{\epsilon})$ groups in $\mathcal{R}$ are matchable (connected by an edge in $M^{\star}$).
- For each $L_i \in \mathcal{L}$, one matchable group $R_j \in \mathcal{R}$ is a partner.
- For each $L_i \in \mathcal{L}$, the matchable partner is chosen uniformly at random from all matchable groups.
- $\cdots$
- The sampled edges have a matching of size $\Omega(n^{1-\epsilon})$, i.e., an $O(n^{\epsilon})$-approximate maximum matching.

Conclusion: There exists an n^{ϵ}-approximation algorithm for matchings in $\tilde{O}(n^{2-3\epsilon})$ space.

Lower Bound

Lower Bound for n^ϵ-Approximation

Theorem (Lower bound)

For any $\epsilon \geq 0$, space of $\tilde{\Omega}(n^{2-3\epsilon})$ is necessary for computing an n^ϵ-approximate maximum matching.

- We prove the lower bound for linear sketches.
- Combined with the work of [AHLW16], this provides a tight lower bound for all dynamic graph stream algorithms.

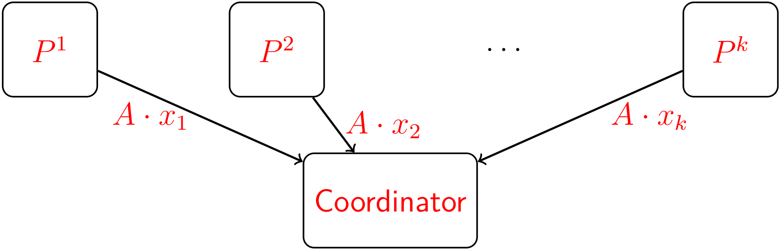
Simultaneous Communication Model

We prove the lower bound using simultaneous communication complexity:

- The input graph is edge partitioned between k players $P^1, \ldots, P^k$.
- There exists another party called the coordinator, with no input.
- Players simultaneously send a message to the coordinator and the coordinator outputs the final matching.
- Communication measure: maximum # of bits send by any player.
- Players have access to public randomness.

Connection to Linear Sketches

If there exists a randomized linear sketch A of size s for a problem P, then the randomized simultaneous communication complexity of P is at most $O(s)$.



$$A \cdot x = A \cdot (x_1 + \ldots + x_k)$$

Hence, a communication lower bound in this model implies an identical space lower bound for linear sketching algorithms.

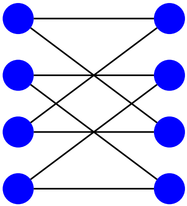
Ruzsa-Szemerédi Graphs

We prove our lower bound using a construction based on Ruzsa-Szemerédi graphs.

Definition ((r, t)-RS graphs)

A graph $G(V, E)$ whose edges can be partitioned into t induced matchings of size r each.

1. Example. A $(2, 4)$-RS graph on 8 vertices:



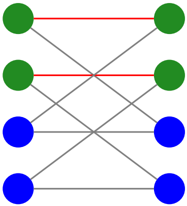
Ruzsa-Szemerédi Graphs

We prove our lower bound using a construction based on Ruzsa-Szemerédi graphs.

Definition ((r,t)-RS graphs)

A graph $G(V,E)$ whose edges can be partitioned into t induced matchings of size r each.

1. Example. A $(2,4)$-RS graph on 8 vertices:



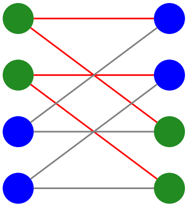
Ruzsa-Szemerédi Graphs

We prove our lower bound using a construction based on Ruzsa-Szemerédi graphs.

Definition ((r,t)-RS graphs)

A graph $G(V,E)$ whose edges can be partitioned into t induced matchings of size r each.

1. Example. A $(2,4)$-RS graph on 8 vertices:



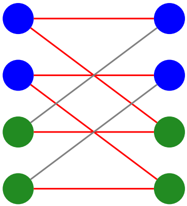
Ruzsa-Szemerédi Graphs

We prove our lower bound using a construction based on Ruzsa-Szemerédi graphs.

Definition ((r, t)-RS graphs)

A graph $G(V, E)$ whose edges can be partitioned into t induced matchings of size r each.

1. Example. A $(2, 4)$-RS graph on 8 vertices:



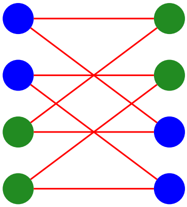
Ruzsa-Szemerédi Graphs

We prove our lower bound using a construction based on Ruzsa-Szemerédi graphs.

Definition ((r, t)-RS graphs)

A graph $G(V, E)$ whose edges can be partitioned into t induced matchings of size r each.

1. Example. A $(2, 4)$-RS graph on 8 vertices:



Ruzsa-Szemerédi Graphs

How dense a graph with many large induced matching can be?

Ruzsa-Szemerédi Graphs

How dense a graph with many large induced matching can be?

Theorem ([AMS12])

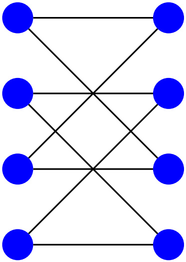
There exists an (r, t)-RS graph on N vertices and $\Omega(N^2)$ edges with $t = N^{1+o(1)}$ induced matchings of size $r = N^{1-o(1)}$.

$\tilde{\Omega}(n^{2-3\epsilon})$ Lower Bound: Distribution

- Parameters:

$$k \approx n^{\epsilon}, r = n^{1-\epsilon-o(1)}, t = n^{1-\epsilon}$$

- Each of the k players is given an (r, t)-RS graph on $n^{1-\epsilon}$ vertices.



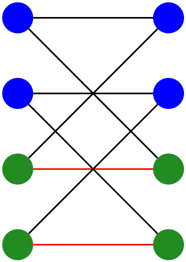
Local view of P^i

$\tilde{\Omega}(n^{2-3\epsilon})$ Lower Bound: Distribution

- Parameters:

$$k \approx n^{\epsilon}, r = n^{1-\epsilon-o(1)}, t = n^{1-\epsilon}$$

- Each of the k players is given an (r, t)-RS graph on $n^{1-\epsilon}$ vertices.
- One induced matching (red edges) of each player's graph is special.



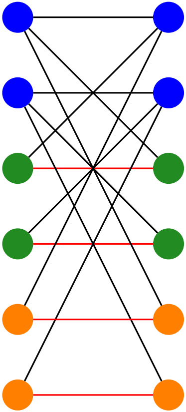
Special matching of P^i

$\tilde{\Omega}(n^{2-3\epsilon})$ Lower Bound: Distribution

- Parameters:

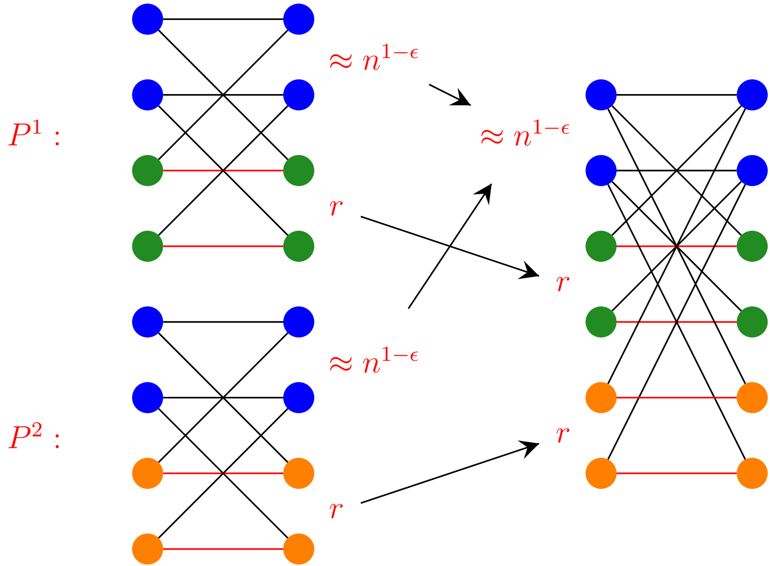
$$k \approx n^{\epsilon}, r = n^{1-\epsilon-o(1)}, t = n^{1-\epsilon}$$

- Each of the k players is given an (r,t)-RS graph on $n^{1-\epsilon}$ vertices.
- One induced matching (red edges) of each player's graph is special.
- Across the players, vertices in the special matchings are unique, while other vertices are shared.



Global view

$\tilde{\Omega}(n^{2-3\epsilon})$ Lower Bound: Distribution

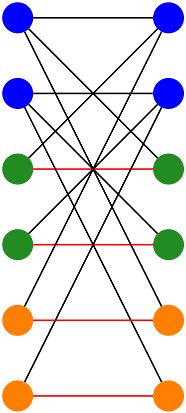


$\tilde{\Omega}(n^{2-3\epsilon})$ Lower Bound: Distribution

Parameters:

$$k \approx n^{\epsilon}, r = n^{1-\epsilon-o(1)}, t = n^{1-\epsilon}$$

- Special matchings are necessary for any large matching.



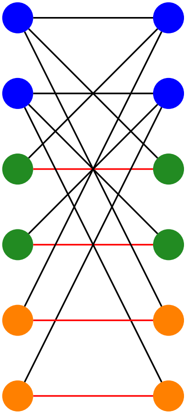
Global view

$\tilde{\Omega}(n^{2-3\epsilon})$ Lower Bound: Distribution

Parameters:

$$k \approx n^{\epsilon}, r = n^{1-\epsilon-o(1)}, t = n^{1-\epsilon}$$

- Special matchings are necessary for any large matching.
- To obtain $o(n^{2-3\epsilon})$ communication, the players have to compress their graph by an $\Omega(n^{\epsilon})$ factor.



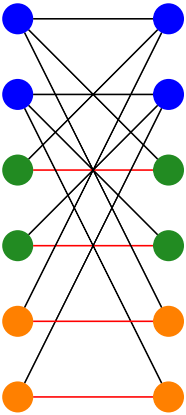
Global view

$\tilde{\Omega}(n^{2-3\epsilon})$ Lower Bound: Distribution

Parameters:

$$k \approx n^{\epsilon}, r = n^{1-\epsilon-o(1)}, t = n^{1-\epsilon}$$

- Special matchings are necessary for any large matching.
- To obtain $o(n^{2-3\epsilon})$ communication, the players have to compress their graph by an $\Omega(n^{\epsilon})$ factor.
- Players are oblivious to the identity of their special matching.



Global view

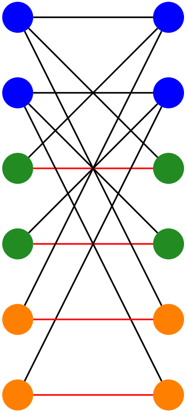
$\tilde{\Omega}(n^{2-3\epsilon})$ Lower Bound: Distribution

Parameters:

$$k \approx n^{\epsilon}, r = n^{1-\epsilon-o(1)}, t = n^{1-\epsilon}$$

- Special matchings are necessary for any large matching.
- To obtain $o(n^{2-3\epsilon})$ communication, the players have to compress their graph by an $\Omega(n^{\epsilon})$ factor.
- Players are oblivious to the identity of their special matching.

Conclusion: Assuming each player sends only $o(n^{2-3\epsilon})$ bits, the coordinator cannot output an n^{ϵ}-approximate maximum matching.



Global view

Conclusion and Open Problems

Space of $\tilde{O}(n^{2-3\epsilon})$ is both sufficient and necessary for computing an n^{ϵ}-approximate maximum matching in dynamic graph streams.

Open question: Can we improve the trivial 2-approximation algorithm for matchings in insertion-only streams?

Questions?

Kook Jin Ahn and Sudipto Guha.
Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints.
CoRR, abs/1307.4359, 2013.

Kook Jin Ahn and Sudipto Guha.
Linear programming in the semi-streaming model with application to the maximum matching problem.
Inf. Comput., 222:59–79, 2013.

Kook Jin Ahn, Sudipto Guha, and Andrew McGregor.
Graph sketches: sparsification, spanners, and subgraphs.
In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 5–14, 2012.

Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff.
Additive error norm approximation and new characterizations in turnstile streams.

Manuscript, 2016.

Noga Alon, Ankur Moitra, and Benny Sudakov.
Nearly complete graphs decomposable into large induced matchings and their applications.
In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC*, pages 1079–1090, 2012.

Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova.
Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams.
In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1326–1344, 2016.

Michael Crouch and Daniel S. Stubbs.

Improved streaming algorithms for weighted matching, via unweighted matching.
In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 96–104, 2014.

Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak.
Streaming algorithms for estimating the matching size in planar graphs and beyond.
In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1217–1233, 2015.

Sebastian Eggert, Lasse Kliemann, and Anand Srivastav.
Bipartite graph matchings in the semi-streaming model.
In *Algorithms - ESA 2009, 17th Annual European Symposium*, pages 492–503, 2009.

Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev.
Improved approximation guarantees for weighted matching in the semi-streaming model.
SIAM J. Discrete Math., 25(3):1251–1265, 2011.

Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang.
On graph problems in a semi-streaming model.
Theor. Comput. Sci., 348(2-3):207–216, 2005.

Ashish Goel, Michael Kapralov, and Sanjeev Khanna.
On the communication and streaming complexity of maximum bipartite matching.
In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 468–485. SIAM, 2012.

Venkatesan Guruswami and Krzysztof Onak.
Superlinear lower bounds for multipass graph processing.

In *Proceedings of the 28th Conference on Computational Complexity, CCC*, pages 287–298, 2013.

Hossein Jowhari, Mert Sağlam, and Gábor Tardos.
Tight bounds for lp samplers, finding duplicates in streams, and related problems.
In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS*, pages 49–58, 2011.

Michael Kapralov.
Better bounds for matchings in the streaming model.
In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1679–1697, 2013.

Michael Kapralov, Sanjeev Khanna, and Madhu Sudan.
Approximating matching size from random streams.
In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 734–751, 2014.

Christian Konrad, Frédéric Magniez, and Claire Mathieu.
Maximum matching in semi-streaming with few passes.
In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX, and 16th International Workshop, RANDOM*, pages 231–242, 2012.

Christian Konrad.
Maximum matching in turnstile streams.
In *Algorithms - ESA 2015 - 23rd Annual European Symposium*, pages 840–852, 2015.

Yi Li, Huy L. Nguyen, and David P. Woodruff.
Turnstile streaming algorithms might as well be linear sketches.
In *Symposium on Theory of Computing, STOC*, pages 174–183, 2014.

Andrew McGregor.
Finding graph matchings in data streams.

In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX and 9th InternationalWorkshop on Randomization and Computation, RANDOM*, pages 170–181, 2005.

Mariano Zelke.
Weighted matching in the semi-streaming model.
Algorithmica, 62(1-2):1–20, 2012.