Survey on Matching in the Graph-Stream Model

CS328 Project

Mrinal Anand (17110087) — Kishen Gowda (17110074) — Vraj Patel (17110174)

Advisor: Anirban Dasgupta

July 22, 2020

Indian Institute of Technology Gandhinagar

Introduction

Rise of Big Data

- In the last few decades, the world has witnessed exponential growth in the number and size of real world data.
- Analysing these massive data using classical algorithms is a challenging task.
- Two approaches to analyze such massive data:
    - Data Streaming
    - Distributed Processing

Graph Streaming

- Streaming algorithms sequentially scans the input data.
- Input data streams can be in random order or in adversarial order.
- Specifically, in graph streaming the input stream is of vertices or edges.
- Streaming algorithm uses $\mathcal{O}(\mathit{poly} \log n)$ (polylogarithmic) memory.

Semi-Streaming Model

- Polylogarithmic memory is insufficient for many graph problems [3].
- Even the basic bipartiteness or connectivity problems in graphs requires $\Omega(n)$ space.
- Semi-Streaming Model: Allow $\mathcal{O}(n \textit{poly} \log n)$ (or $\tilde{\mathcal{O}}(n)$) space [2, 8].

Maximum Matching

Maximum Matching Problem

Maximum Cardinality Matching

Given a graph $G = (V, E)$, find a subset $M \subseteq E$ of maximum size such that no two adjacent edges are selected.

- Simply called the (Unweighted) Maximum Matching Problem.
- **Maximum Bipartite Matching:** Maximum Matching problem on Bipartite Graphs

Known Results and Bounds

- Fastest Algorithm: $\mathcal{O}(m\sqrt{n})$ [7]
- Semi-Streaming Model:
  - Greedy Algorithm: $1/2$ approximation
  - Hardness: $1 - 1/e$ approximation [4]
  - Better than $1/2$? Open Problem

Algorithms with Multiple Passes or Random Stream Orders

Existing Semi-Streaming Algorithms

- Multiple passes of stream
- Random Stream Orders

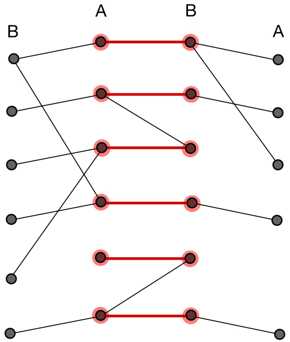
Existing Semi-Streaming Algorithms

- Multiple passes of stream
- Random Stream Orders
- Standard Approach: Finding Augmenting Paths
- McGregor[6]: $1/(1+\epsilon)$ approximation with constant number of passes (strongly dependent on ϵ)
- Feigenbaum et. al. [2]: $2/3-\epsilon$ approximation with $\mathcal{O}\left(\mathit{log}\frac{1}{\epsilon}/\epsilon\right)$ passes.

Three Pass Algorithm on Arbitrary Stream Orders

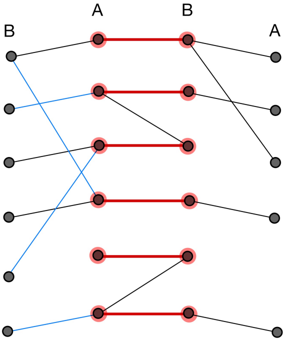
- **Idea:**
  - Compute a maximal matching M_G in one pass.
  - Utilize the second and third passes to find 3-augmenting paths.
  - Existence of 3-augmenting paths?
- **Lemma:** When Greedy is close to $1/2$ approximation, there exists many 3-augmenting paths.[5]

Three Pass Algorithm on Arbitrary Stream Orders



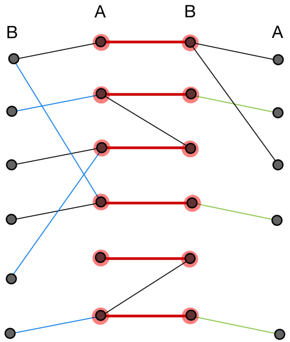
- **First Pass:** Compute a maximal matching M_G

Three Pass Algorithm on Arbitrary Stream Orders



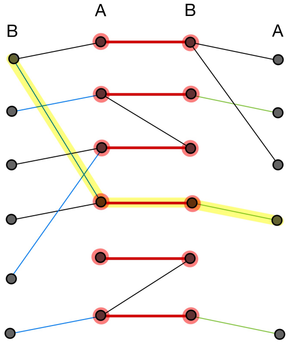
- **Second Pass:** Compute a maximal matching M_L between $\overline{A(M_G)}$ and $B(M_G)$

Three Pass Algorithm on Arbitrary Stream Orders



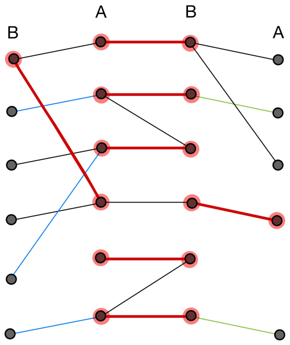
- **Third Pass:** Compute a maximal matching M_R between $\overline{B(M_G)}$ and $\{a \in A(M_G) : M_G(a) \in B(M_L)\}$

Three Pass Algorithm on Arbitrary Stream Orders



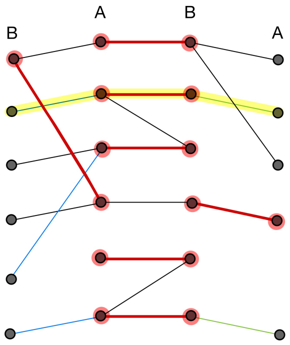
- **Third Pass:** Compute a maximal matching M_R between $\overline{B(M_G)}$ and $\{a \in A(M_G) : M_G(a) \in B(M_L)\}$

Three Pass Algorithm on Arbitrary Stream Orders



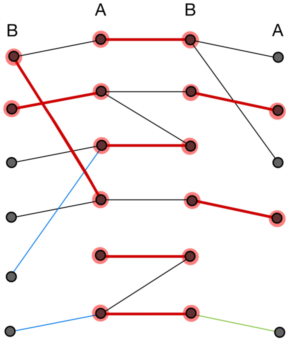
- **Third Pass:** Compute a maximal matching M_R between $\overline{B(M_G)}$ and $\{a \in A(M_G) : M_G(a) \in B(M_L)\}$

Three Pass Algorithm on Arbitrary Stream Orders



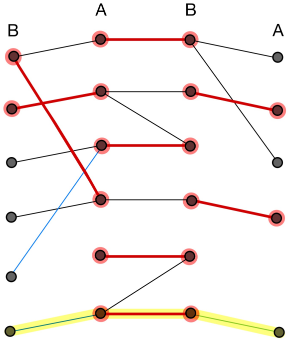
- **Third Pass:** Compute a maximal matching M_R between $\overline{B(M_G)}$ and $\{a \in A(M_G) : M_G(a) \in B(M_L)\}$

Three Pass Algorithm on Arbitrary Stream Orders



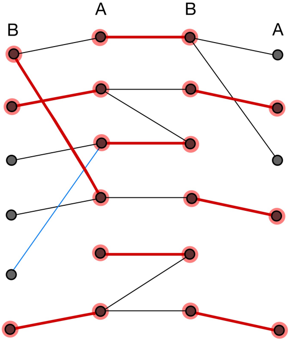
- **Third Pass:** Compute a maximal matching M_R between $\overline{B(M_G)}$ and $\{a \in A(M_G) : M_G(a) \in B(M_L)\}$

Three Pass Algorithm on Arbitrary Stream Orders



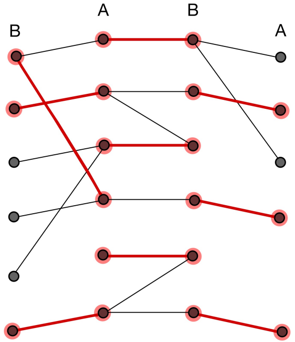
- **Third Pass:** Compute a maximal matching M_R between $\overline{B(M_G)}$ and $\{a \in A(M_G) : M_G(a) \in B(M_L)\}$

Three Pass Algorithm on Arbitrary Stream Orders



- **Third Pass:** Compute a maximal matching M_R between $\overline{B(M_G)}$ and $\{a \in A(M_G) : M_G(a) \in B(M_L)\}$

Three Pass Algorithm on Arbitrary Stream Orders



- **Third Pass:** Compute a maximal matching M_R between $\overline{B(M_G)}$ and $\{a \in A(M_G) : M_G(a) \in B(M_L)\}$

One-Pass Algorithm on Random Stream Orders

- **Lemma:** In expectation over all input edge sequences, if matching computed by Greedy algorithm is close to a $1/2$ approximation, then Greedy builds this matching early on, in other words, converges quickly [5].

One-Pass Algorithm on Random Stream Orders

- **Lemma:** In expectation over all input edge sequences, if matching computed by Greedy algorithm is close to a $1/2$ approximation, then Greedy builds this matching early on, in other words, converges quickly [5].
- **Idea:**
  - Split the input stream into 3 phases.
  - First Phase: Compute a greedy matching M_0.
  - Second and Third Phases: Find 3-augmenting paths.
  - Also, compute greedy matching M_G in parallel.
  - Maximum of M_G and augmented M_0 (with $M_1 \cup M_2$).

One-Pass Algorithm on Random Stream Orders

- **Lemma:** In expectation over all input edge sequences, if matching computed by Greedy algorithm is close to a $1/2$ approximation, then Greedy builds this matching early on, in other words, converges quickly [5].
- **Idea:**
  - Split the input stream into 3 phases.
  - First Phase: Compute a greedy matching M_0.
  - Second and Third Phases: Find 3-augmenting paths.
  - Also, compute greedy matching M_G in parallel.
  - Maximum of M_G and augmented M_0 (with $M_1 \cup M_2$).
- 0.505 approximation, 0.503 approximation (General Graphs)

Empirical Results

Table 1: Datasets Used (From SuiteSparse Matrix Collection)

Dataset	Nodes	Edges	MBM
IMDB	1,324,748	3,782,463	250,516
NotreDameActors	520,223	1,470,404	114,762

Empirical Results

Table 1: Results Obtained (Average over 10 shuffles)

Opt.	Gdy. & 1 P	2 P(R)	2 P(D)	3 P
250,516	215,997.6	227,317	238,831.2	240,097.6
114,762	97,293.2	100,744.5	106,448.6	108,300.5

What is covered in our Survey?

- Focused on Maximum Cardinality Matching problem.
- Discussed bounds and hardness briefly.
- Surveyed algorithms dealing with multiple passes on arbitrary stream orders.
- Also Surveyed algorithms dealing with random stream orders (single and multiple passes).
- Have described the techniques presented in [6, 5, 1] in detail

Future Work

- Survey techniques from results on more specific cases: Planar graphs, low-arboricity graphs, etc.
- Survey results on Weighted Matching.
- Parameterized Perspective analysis.
- Dynamic Graph streams.
- Results from Online Matching.
- Techniques from algorithms from other Graph problems in the streaming model (A general survey on Graph Streaming Algorithms.)

References i

Alireza Farhadi, MohammadTaghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi.
Approximate maximum matching in random streams.
In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, page 1773–1785, USA, 2020. Society for Industrial and Applied Mathematics.

References ii

- Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang.
  On graph problems in a semi-streaming model.
  Theoretical Computer Science, 348:207–216, 12 2005.
- Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang.
  Graph distances in the data-stream model.
  SIAM Journal on Computing, 38(5):1709–1727, 2009.

Mikhail Kapralov.
Better bounds for matchings in the streaming model.
In *SODA*, 2013.

Christian Konrad, Frédéric Magniez, and Claire Mathieu.
Maximum matching in semi-streaming with few passes.
In Anupam Gupta, Klaus Jansen, José Rolim, and Rocco Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*,

pages 231–242, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

Andrew Mcgregor.
Finding graph matchings in data streams.
In *APPROX-RANDOM*, 2005.

Silvio Micali and Vijay Vazirani.
An o(sqrt(v)e) algorithm for finding maximum matching in general graphs.
pages 17–27, 10 1980.

S. Muthukrishnan.
Data streams: algorithms and applications.
Foundations and Trends in Theoretical Computer Science, 1, 2003.