Semi-streaming Model: Graph Matchings

Team 7
Sushma CS20BTECH11051
Monika CS20BTECH11026
Namita CS20BTECH11034
Akshay Santoshi CS21BTECH11012

December 20, 2024

Table of Contents

1. Semi-Streaming Model
2. Unweighted Bipartite Matching
3. Weighted Matching
4. Lower Bounds: Verifying a Maximum Matching
5. References

Motivation

Definition (Streaming Model)

Streaming is a model of computation on massive data sets that arrive sequentially in an arbitrary order. We have only poly-log(m) amount of storage space, m being the number of elements in the stream.

Most (massive) graph problems hadn't been explored. Reasons:

- very tight space constraint
- data access in sequential (potentially adversarial) order

The paper by Feigenbaum et al. [1] first explores the semi-streaming model with a more lenient space constraint and gives algorithms for a few graph problems under this model.

Definition (Semi-Streaming Model)

For an input graph $G(V, E)$, we have n poly-log(n) space, with $n = |V|$. Hence, we can store $G(V)$ but not $G(E)$.

Semi-Streaming Model

Definition (Graph Stream)

A graph stream $\sigma(G)$ is a sequence of edges $e_{i_1}, e_{i_2}, \ldots, e_{i_m}$ that appear one at a time as input to an algorithm, where $e_{i_j} \in G(E)$, $m = |E|$ and $i_1, i_2, \ldots, i_m$ is an arbitrary permutation of $[m]$.

Definition (Semi-Streaming Graph Algorithm)

It is an algorithm that computes a specific property of a graph G given $\sigma(G)$. In doing so, it uses $S(n, m)$ space, $T(n.m)$ time to process each edge of the stream and $P(n, m)$ sequential passes of the streams. It is required that $S(n, m)$ be $O(n \text{ poly-log}(n))$ and $P(n, m)$ be $O(\text{poly-log}(n))$.

We will cover a $\left(\frac{2}{3} - \epsilon\right)$-algorithm for unweighted bipartite matching and a $\left(\frac{1}{6}\right)$-algorithm for weighted matching under this model from the paper by Feigenbaum et al.

Graph Bipartiteness

Algorithm 1

This algorithm verifies graph bipartiteness during edge streaming. As edges stream in, we use a disjoint set data structure to maintain connected components of the graph so far. We also associate a sign with each vertex such that no edge connects 2 vertices of the same sign. If this condition ever fails even on flipping the sign of a vertex and the vertices in its connected component, the graph is non-bipartite.

Definition (length-3 augmenting path)

Given a matching M in a bipartite graph $G = (L \cup R, E)$, a length-3 augmenting path for an edge $(u, v) \in M$, where $u \in L$ and $v \in R$, is a quadruple (w_l, u, v, w_r) such that $(w_l, u), (v, w_r) \in E$ and w_l and w_r are free vertices. We call (u, w_l) the **left wing**, (v, w_r) the **right wing**, and w_l and w_r the **wing-tips**.

Simultaneously augmentable length-3 augmenting paths

Definition (Simultaneously augmentable length-3 augmenting paths)

A set of simultaneously augmentable length-3 augmenting paths is a set of length-3 augmenting paths that are vertex disjoint.

Algorithm 2

Input:

- Bipartite graph $G = (L \cup R, E)$
- Matching M for G
- Parameter $0 < \delta < 1$

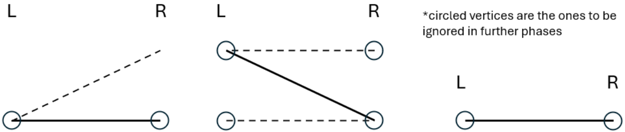
Output:

- simultaneously augmentable length-3 augmenting paths for G under M

Procedure

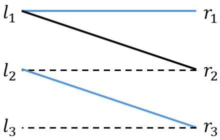
1. In one pass, find a maximal set of disjoint left wings LW. Terminate if the number of left wings $\leq \delta|M|$.
2. In 2nd pass, for edges in M with left wings, find a maximal set of disjoint right wings RW.
3. In 3rd pass, identify vertices that are:
   1. Endpoints of matched edges with a left wing in LW.
   2. Wing-tips of matched edges with both wings (1 each in LW and RW).
   3. Endpoints of matched edges that are no longer 3-augmentable.

   Ignore edges incident on any 1 of these vertices in further passes.
4. Store length-3 augmenting paths found. Repeat steps 1-4.



Algorithm 2: Execution

- In step 3 of the algorithm, we're basically identifying the vertices that can no longer become a part of length-3 augmenting paths in further passes.
- In order to correctly identify vertices that fall into the 3rd category, we associate an indicator variable with each edge of the matching which indicates if a left wing was detected for that edge in the 1st pass. If the variable stores *false*, it belongs to the 3rd category.



σ: $(l_1,r_1), (l_1,r_2), (l_2,r_2), (l_2,r_3), (l_3,r_3)$

$LW = \{(l_1,r_2)\}$

— matching
— LW
- - - unstored edges

After 1st phase, (l_1,r_1) and (l_1,r_2) will be ignored. We don't want (l_2,r_3) to be ignored.

Algorithm 3

Input:

- Bipartite graph $G = (L \cup R, E)$
- Parameter $0 < \epsilon < \frac{1}{3}$

Output:

- a $\left(\frac{2}{3} - \epsilon\right)$-factor matching on G

Procedure:

- Find a maximal matching M and bipartition of G in a single pass.
- For every $k = 1, 2, \ldots, \left\lceil \frac{\log(6\epsilon)}{\log\left(\frac{8}{9}\right)} \right\rceil$:
  - Execute Algorithm 2 with G, M, and $\delta = \frac{\epsilon}{2-3\epsilon}$.
  - For each edge $e = (u, v) \in M$ where an augmenting path (w_l, u, v, w_r) is found:
    - Remove e from M.
    - Append (u, w_l) and (w_r, v) to M.

Lemma 1

Lemma 1

The size of a maximal set of simultaneously augmentable length-3 augmenting paths is at least 1/3 of the size of a maximum set of simultaneously augmentable length-3 augmenting paths.

Proof.

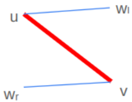
- $AP_{max} \triangleq$ A maximum set of simultaneously augmentable length-3 augmenting paths.
- $AP \triangleq$ A maximal set of simultaneously augmentable length-3 augmenting paths.
- $M \triangleq$ Matching considered

Each path in the maximal set destroys at most 3 paths that AP_{max} might have used. $\square$

Proof of Lemma 1

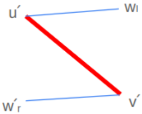
Proof.

Consider $(w_l, u, v, w_r) \notin AP_{max}$ but $\in AP$



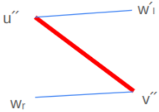
$(u,v) \in M$
$w_l \in R$ (left wing-tip)
$w_r \in L$ (right wing-tip)

1. $(w_l, u', v', w'_r) \in AP_{max}$ but $\notin AP$



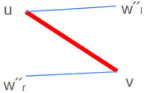
$(u',v') \in M$
$w_l \in R$ (left wing-tip)
$w'_r \in L$ (right wing-tip)

2. $(w'_l, u'', v'', w_r) \in AP_{max}$ but $\notin AP$



$(u'',v'') \in M$
$w'_l \in R$ (left wing-tip)
$w_r \in L$ (right wing-tip)

3. $(w''_l, u, v, w''_r) \in AP_{max}$ but $\notin AP$



$w''_l \in R$ (left wing-tip)
$w''_r \in L$ (right wing-tip)

Proof of Lemma 1 contd.

Proof.

From the above figure, we can see that a path in the maximal set destroyed 3 paths that AP_{max} has.

These are:

- Path involving left wing-tip
- Path involving right wing-tip
- Path involving matched-edge used

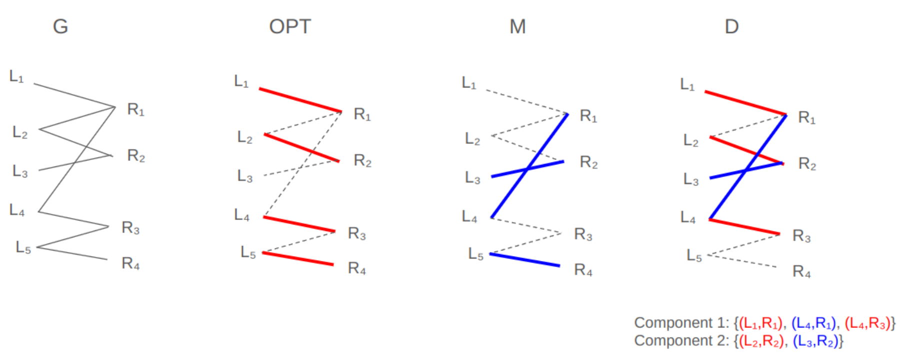
Therefore, a maximal set has a size of at least $1/3$ of the maximum set. $\square$

Lemma 2

Lemma 2

Let X be the maximum-sized set of simultaneously augmentable length-3 augmenting paths for a maximal matching M. Let $\alpha = \frac{|X|}{|M|}$ and OPT a maximum matching. Then $|M|(1+\alpha) \geq 2/3|OPT|$

Figure



Proof of Lemma 2

Proof.

- $D \triangleq$ symmetric difference $OPT \triangledown M$
- $y \triangleq$ no. of edges which common to both M and OPT
- $c_1 \triangleq$ no. of components in D with one edge from M and two edges from OPT
- $c_2 \triangleq$ (total no. of components in D) - c_1
- $|e_{M_i}| \triangleq$ no. of edges that come from M in the i^{th} component of D
- $|e_{OPT_i}| \triangleq$ no. of edges that come from OPT in the i^{th} component of D

In each connected component, the following hold true:

$\rightarrow$ $|e_{OPT_i}| \geq |e_{M_i}| \; \forall \; i$
(if not, defn. of OPT being maximum would not hold true, since we could replace edges in e_{OPT_i} with those in e_{M_i})

$\square$

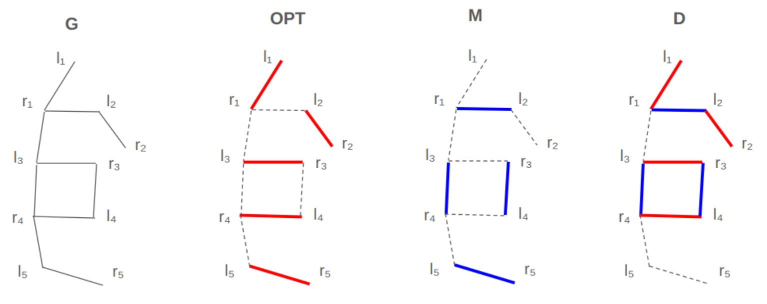
Proof of Lemma 2 contd.

Proof.

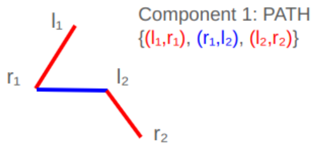
→ There is at most one more edge from OPT than there is from M.

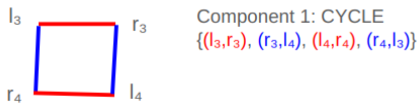
$$|e_{OPT_i}| \leq |e_{M_i}| + 1 \forall i \tag{1}$$

(Since OPT is a matching, the edges would be disjoint and hence for the component to be connected, we need edges from matching M to connect them.)



Proof.





(Therefore each vertex in the component will have degree either 2 or 1. So, the component is either a path or a cycle.)

→ No connected component consists of only a single edge that came from *OPT* (because *M* is maximal)

→ (Using defn. of *X*)

$$c_1 \leq |X| \tag{2}$$

□

Proof of Lemma 2 contd.

Proof.

Claim

In all components other than those in c_1, the ratio of e_{M_i} to e_{OPT_i} is at least 2 : 3

Proof of Claim

Case I: Components with $|e_{M_i}| = 1$ and $|e_{OPT_i}| = 1$. Ratio is 1 : 1 in this case and hence claim holds true.
Case II: $|e_{M_i}| \geq 1$
Using equation (1), we get

$$\frac{|e_{OPT_i}|}{|e_{M_i}|} \leq 1 + \frac{1}{|e_{M_i}|} \leq \frac{3}{2} \tag{3}$$

$$\implies \frac{|e_{M_i}|}{|e_{OPT_i}|} \geq \frac{2}{3} \tag{4}$$

Proof of Lemma 2 contd.

Proof.

We can write the following equations

$$|M| = c_1 + y + \sum_{i=1}^{c_2} |e_{M_i}| \tag{5}$$

$$|OPT| = 2c_1 + y + \sum_{i=1}^{c_2} |e_{OPT_i}| \tag{6}$$

Using equation (2) and equation (4),

$$|M| + |X| \geq 2c_1 + y + \sum_{i=1}^{c_2} |e_{M_i}| \tag{7}$$

$$\geq 2c_1 + y + \sum_{i=1}^{c_2} \frac{2}{3} |e_{OPT_i}| \tag{8}$$

Proof of Lemma 2 contd.

Proof.

$$|M| + |X| \geq \frac{4}{3}c_1 + \frac{2}{3}y + \sum_{i=1}^{c_2} \frac{2}{3}|e_{OPT_i}| \tag{9}$$

$$= \frac{2}{3}(2c_1 + y + \sum_{i=1}^{c_2} |e_{OPT_i}|) \tag{10}$$

$$= \frac{2}{3}|OPT| \tag{11}$$

Substituting $|X| = \alpha|M|$, we get

$$|M| + \alpha|M| \geq \frac{2}{3}|OPT| \tag{12}$$

$$|M|(1+\alpha) \geq \frac{2}{3}|OPT| \tag{13}$$

Lemma 3

Lemma 3

Algorithm 2 finds $\frac{\alpha|M|-2\delta|M|}{3}$ simultaneously augmentable length-3 augmenting paths in $3/\delta$ passes.

Proof.

- $L(M) \triangleq$ set of the end vertices in M that are in L
- $V_L(M) \triangleq \{v \in R | v$ is free w.r.t. M and $\exists u \in L(M) s.t. (u, v) \in E\ \}$
- *phase* $\triangleq$ one repetition of step $1-4$ in Algorithm 2

Algo 2 terminates only when no. of left wings found in the 1^{st} pass of a phase is $\leq \delta M$ as defined in Algo 2.
The no. of phases is at most $1/\delta$ because at least $\delta|M|$ edges in M are removed at each phase.
Since each phase has 3 passes, we can say that no. of passes is at most $3/\delta$ □

Proof of Lemma 3 contd.

Proof.

- $G' \triangleq$ graph restricted to the remaining vertices in $L(M)$ and $V_L(M)$ when the Algo. 2 terminates.
- $G'' \triangleq G \backslash G'$
- $X \triangleq$ A maximum sized set of simultaneously augmentable length-3 augmenting paths for the given matching M in G

Claim

A maximum set of simultaneously augmentable length-3 augmenting paths in G'' would have a size at least $\alpha|M| - 2\delta|M|$.

Proof of claim

$\rightarrow$ When Algo. 2 terminates, the no. of left wings found is at least $\delta|M|$.

$\rightarrow$ This set of left-wings form a maximal matching in G'.

Proof of Lemma 3 contd.

Proof.

Proof of claim contd.

- → Hence there are fewer than $2\delta|M|$ disjoint left wings that could have been found at this phase.
- → Consequently, there are fewer than $2\delta|M|$ simultaneously augmentable length-3 augmenting paths in G'.
- → A maximum set of simultaneously augmentable length-3 augmenting paths in G'' would have a size at least $\alpha|M| - 2\delta|M|$.

Note that the set of length-3 augmenting paths found by Algo. 2 form a maximal set w.r.t G''. (Since Algo 2. uses disjoint wing sets)
Using the claim proved above and Lemma 1, we get that the size of a maximal set is at least $\frac{\alpha|M|-2\delta|M|}{3}$. □

Bipartite Matching: Main Theorem

Theorem 1

For any $0 < \epsilon < \frac{1}{3}$ and a bipartite graph, Algorithm 3 finds a $(\frac{2}{3} - \epsilon)$-approximation maximum matching in $O((\log 1/\epsilon)/\epsilon)$ passes and $O(n \log n)$ space. Each edge is processed in $O(1)$ time except the first pass, where we spend $O(n)$ time per edge in find and union operations of sets.

Proof.

- OPT $\triangleq$ size of maximum matching
- $M_i \triangleq$ matching M of the algorithm after ith iteration
- $X_i \triangleq$ maximum-sized set of simultaneously augmentable length-3 augmenting paths for M_i
- $Y_i \triangleq$ set of simultaneously augmentable length-3 augmenting paths found by *Algorithm 2* for M_i

□

Proof of Theorem 1

Proof.

- $\alpha_i \triangleq |X_i|/|M_i|$
- $s_i \triangleq |M_i|/\text{OPT}$

Case I: $\exists i : \alpha_i \leq \frac{3\epsilon}{2-3\epsilon}$. Using *Lemma 2*,

$$|M_i| \geq \frac{2}{3(1+\alpha_i)}\text{OPT} \geq \frac{2}{3\frac{2}{2-3\epsilon}}\text{OPT} \geq \left(\frac{2}{3} - \epsilon\right)\text{OPT} \tag{14}$$

$$\text{Since } \forall j \ |M_{j+1}| \geq |M_j|, \implies |M_k| \geq |M_i| \geq \left(\frac{2}{3} - \epsilon\right)\text{OPT} \tag{15}$$

Case II: $\alpha_i > \frac{3\epsilon}{2-3\epsilon} \ \forall i \in [k]$. Using *Lemma 2*,

$$\alpha_i|M_i| \geq \frac{2}{3}\text{OPT} - |M_i| \implies \alpha_i s_i \geq \frac{2}{3} - s_i \tag{16}$$

□

Proof of Theorem 1 contd.

Proof.

Also, $\delta = \frac{\epsilon}{2-3\epsilon} \leq \frac{\alpha_i}{3}\ \forall i$. By *Lemma 3*,

$$Y_i \geq \frac{\alpha_i - 2\delta}{3}|M_i| \geq \frac{\alpha_i}{9}|M_i| \tag{17}$$

$$|M_{i+1}| = |M_i| + |Y_i| \geq \left(1 + \frac{\alpha_i}{9}\right)|M_i| \tag{18}$$

$$\implies s_{i+1} \geq s_i + \frac{\alpha_i s_i}{9} \geq s_i + \frac{1}{9}\left(\frac{2}{3} - s_i\right) = \frac{8}{9}s_i + \frac{2}{7} \tag{19}$$

As M_0 is a maximal matching, $s_0 \geq 1/2$. Solving the above recurrence gives $s_k \geq \frac{2}{3} - \frac{1}{6}(\frac{8}{9})^k \geq \frac{2}{3} - \epsilon$

$$\text{Number of passes } \leq k \cdot \frac{3}{\delta} = \left\lceil \log_{\frac{9}{8}} \frac{1}{6\epsilon} \right\rceil \frac{6-9\epsilon}{\epsilon} = O\left(\frac{\log(1/\epsilon)}{\epsilon}\right) \tag{20}$$

Weighted Matching Algorithm

Algorithm 4

We maintain a matching M at all times. When we see a new edge e, we compare $w(e)$ with $w(C)$, the sum of the weights of the edges of $C = \{e' \in M \mid e' \text{ and } e \text{ share an end point}\}$.

- If $w(e) > 2w(C)$, we update $M \leftarrow M \cup \{e\} \backslash C$.
- If $w(e) \leq 2w(C)$, we ignore e.

Theorem 2

Algorithm 4 gives $\frac{1}{6}$-factor weighted matching in 1 pass, $\mathcal{O}(n \log n)$ space.

Notations

- $w(S) \triangleq \sum_{e \in S} w(e)$, where $S \subseteq E$
- born edge $\triangleq$ an edge that is ever part of M

Proof of Theorem 2

Notations

- killed edge $\triangleq$ a born edge murdered (i.e., removed from M) by a newer, heavier edge.
- survivor edge $\triangleq$ a born edge that is never killed
- $S \triangleq$ set of survivor edges
- $T(e) \triangleq C_1 \cup C_2 \cup \ldots \ \forall e \in S$, where
  - $C_0 = \{e\}$
  - $C_1 = \{\text{the edges murdered by e}\}$
  - $C_i = \bigcup_{e' \in C_{i-1}} \{\text{the edges murdered by e'}\}$
- OPT $\triangleq$ a maximum weighted matching on G
- $M_{all} \triangleq \bigcup_{e \in S}(T(e) \cup e)$

Claim

$$w(T(e)) \leq w(e) \tag{21}$$

Proof of Theorem 2 contd.

Proof.

For each murdering edge e, $w(e)$ is at least twice the cost of murdered edges, and an edge has at most one murderer. Hence

$$w(C_i) \geq 2w(C_{i+1}) \; \forall i \tag{22}$$

$$\implies 2w(T(e)) = \sum_{i \geq 1} 2w(C_i) \leq \sum_{i \geq 0} w(C_i) = w(T(e)) + w(e) \tag{23}$$

□

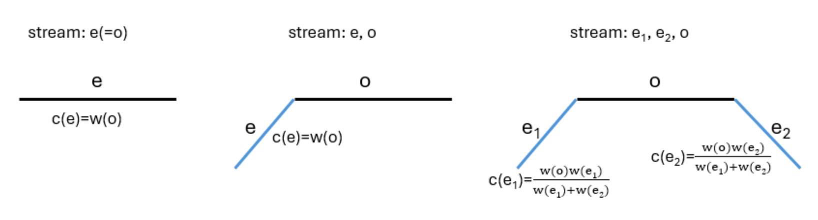
Proof of Theorem 2.

We charge costs of edges in OPT to edges in M_{all} . An edge $e \in M_{all}$ is **accountable** to $o \in$ OPT if $e = o$ or if o wasn't born because e was in M when o arrived.

Proof of Theorem 2 contd.

Proof.

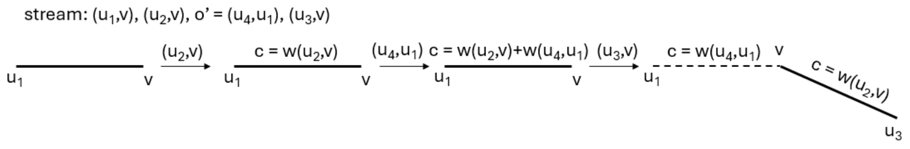
- Case I: If only one edge e is accountable to o (implying $w(o) \leq 2w(e)$), we assign charge $c_o(e) \triangleq w(o) \leq 2w(e)$.
- Case II: If two edges e_1 and e_2 are accountable to o (implying $w(o) \leq 2(w(e_1) + w(e_2))$), we assign charges $c_o(e_1) \triangleq \frac{w(o)w(e_1)}{w(e_1)+w(e_2)} \leq 2w(e_1)$ and $c_o(e_2) \triangleq \frac{w(o)w(e_2)}{w(e_1)+w(e_2)} \leq 2w(e_2)$



Proof of Theorem 2 contd.

Proof.

We redistribute charges as follows: for distinct u_1, u_2, u_3, if an edge $e = (u_1, v)$ is charged by $o = (u_2, v)$, and then killed by another edge $e' = (u_3, v)$, the charge is transferred from e to e'. Since $w(e') \geq 2w(e) \geq c_o(e)$, hence $c_o(e') = c_o(e) \leq 2w(e')$. This ensures that each killed edge is charged by at most one edge in OPT.



$$w(\mathrm{OPT}) \leq \sum_{e \in S} (2w(T(e)) + 4w(e)) \leq 6w(S) \text{ (Using claim)} \tag{24}$$

Lower Bound on $s - t$ Connectivity

Lemma 4

Testing for $s - t$ connectivity in a directed graph $G = (V, E)$ requires $\Omega(m)$ bits of space, where $|E| = m$.

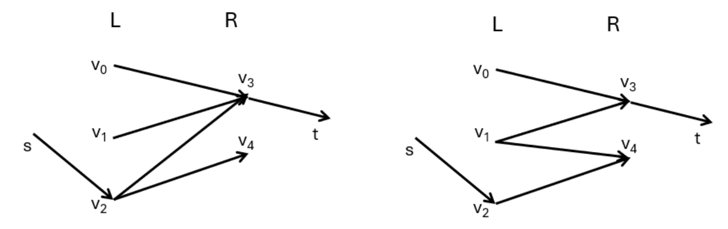
Proof.

- $\mathcal{G} \triangleq$ set of all directed bipartite graphs
- $\mathcal{F} \triangleq \{G = (L \cup R \cup \{s, t\}, E) \in \mathcal{G} : (u, v) \in E \implies (u \in L, v \in R) \vee ((u, v) = (s, l)) \vee ((u, v) = (r, t))\}$, where l and r are fixed vertices in L and R respectively.

Suppose the stream provides all $L - R$ edges first, then (s, l) and (r, t). At the point when all edges from L to R have appeared, any correct algorithm will have different memory configuration for each graph in $\mathcal{F}$ since there are continuations that could lead to different answers for any two graphs in $\mathcal{F}$. □

Proof of Lemma 4

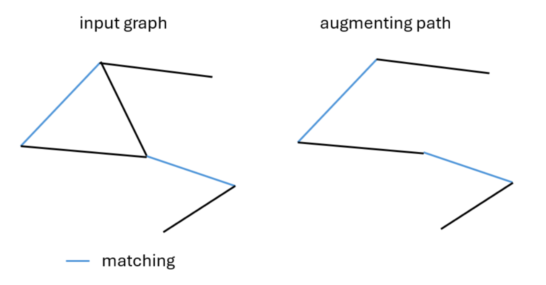
Proof.



To have a unique memory configuration for each graph, we need $\Omega(\log_2 |\mathcal{F}|)$ bits. Even for the restricted case when $|L| = |R| = (n/2) - 1$, the number of such graphs is $((2^{(n/2)-1})^{(n/2)-1} = 2^{\Omega(n^2)}$, which gives $\Omega(\log_2 |\mathcal{F}|)$ to be $\Omega(n^2) = \Omega(m)$. □

Verifying Maximum Matching: Main Theorem

Definition (Augmenting Path)

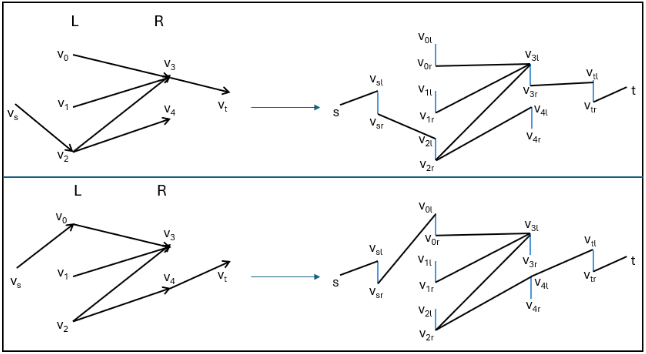


Theorem 3

Given a bipartite graph $G = (L \cup R, E)$, determining existence of an augmenting path from $s \in R$ to $t \in L$ requires $\Omega(m)$ storage.

Proof.

Given $G = (L \cup R \cup \{v_s, v_t\}) \in \mathcal{F}$, we construct an undirected bipartite graph G' with nodes s, t such that $\exists$ an augmenting path from s to t in G' iff $\exists$ a directed path from v_s to v_t in G.



□

References

J. Feigenbaum, S. Kannan, A. McGregor, S. Suri and J. Zhang, "On graph problems in a semi-streaming model," *Theoretical Computer Science, Volume 348, Issues 2–3, Pages 207-216*, 8 December 2005.

Thank You