Dynamic Maximal Independent Set

Morteza Monemizadeh*

Abstract

Given a stream S of insertions and deletions of edges of an underlying graph G (with fixed vertex set V where n = |V| is the number of vertices of G), we propose a dynamic algorithm that maintains a maximal independent set (MIS) of G (at any time t of the stream S) with amortized update time $O(\log^3 n)$.

1 Introduction

Very recently at STOC 2018, Assadi, Onak, Schieber, and Solomon [1] proposed a deterministic dynamic algorithm for maintaining a maximal independent set (MIS) with amortized update time $O(\min(\Delta, m^{3/4}))$, where Δ is a fixed bound on the maximum degree in the graph and m is the (dynamically changing) number of edges. Later, Gupta and Khan [5] and independently, Du and Zhang [4] presented deterministic algorithms for dynamic MIS with update times of $O(m^{2/3})$ and $O(m^{2/3} \cdot \sqrt{\log m})$, respectively. Du and Zhang also gave a randomized algorithm with update time $\tilde{O}(\sqrt{m})$. Later at SODA 2019, Assadi, Onak, Schieber, and Solomon [2] developed the first fully dynamic (randomized) algorithm for maintaining a MIS with $\min(\tilde{O}(\sqrt{n}), \tilde{O}(m^{1/3}))$ expected amortized update time.

Here we develop the first randomized dynamic algorithm for MIS with amortized update time $O(\log^3 n)$. Our main result is stated in the following theorem.

Theorem 1 Let S be a stream of insertions and deletions of edges of an underlying unweighted graph G with a fixed vertex set V of size n = |V|. Then, there exists a randomized dynamic algorithm that maintains a maximal independent set of G using amortized $O(\log^3 n)$ update time.

Overview of Algorithm. To prove this theorem we first devise an offline MIS algorithm in Section 2 and then in Section 3 we show how to implement steps of this offline algorithm in a streaming fashion while maintaining a maximal independent set using fast update time. In Section 4 we give our dynamic algorithm that handles insertions and deletions. First we explain the offline algorithm.

Let G(V, E) be an undirected unweighted graph with n = |V| vertices and m = |E| edges. We consider k epochs during which we build levels L_1, \dots, L_k of independent vertices for $k = O(\log n)$. At the beginning of epoch i we assume we have a graph $G_i(V_i, E_i)$. For the first epoch, we let $G_1(V_1, E_1) = G(V, E)$.

At epoch i, we repeat the following sampling process for $s = O(\frac{n_i^2}{m_i})$ times: Repeat sampling (with replacement) a vertex $w \in V_i$ uniformly at random as long as $I_i \cup \{w\}$ is not an independent set in G_i . Once we sample a vertex w for which $I_i \cup \{w\}$ is an independent set, we then let $I_i = I_i \cup \{w\}$ and $N(I_i) = N(I_i) \cup N_{G_i}(w)$. If the graph G_i is sparse (i.e., $|E_I| \leq |V_i|$), we sample all vertices and the ordered set S_i will be a random shuffle of vertices of V_i .

^{*}Work was done while the author was at Amazon AI, Palo Alto, CA, USA. Email: m.monemizadeh@gmail.com.

We let $N(I_i)$ be the set of neighbors of I_i in the graph $G_i(V_i, E_i)$ and we remove I_i and $N(I_i)$ from $G_i(V_i, E_i)$. The level L_i consists of the vertex set V_i , the multiplicative inverse or reciprocal for the average degree of G_i which is $\frac{|V_i|}{|E_i|}$, and the independent set I_i and its neighbor set $N(I_i)$. We remove I_i and $N(I_i)$ from the graph G_i and recursively start the next epoch E_{i+1} .

Next we explain the idea behind our edge insertion and deletion subroutines. Suppose we have a level set $\mathcal{L} = \bigcup_{i=1}^k L_i$ of k levels of an underlying graph G(V,E) where each level L_i is a quadruple $L_i = (V_i, \frac{|V_i|}{|E_i|}, I_i, N(I_i))$ and $\bigcup_{i=1}^k I_i$ is a MIS of G.

Suppose we want to add an arbitrary edge e = (u, v) to G. Let $G' = G(V, E \cup \{e\})$. The amount of recomputation that the insertion of an edge e = (u, v) imposes while reconstructing a MIS of G' given the current MIS of G depends on where this edge is being inserted. We consider two types of insertions, *heavy* insertions and *light* insertions. Roughly speaking, an insertion is a heavy insertion if it changes the current maximal independent set; otherwise it is a light insertion. We show that heavy insertions are rare and the majority of insertions are in fact light insertions for which we do not need to do significant (re)-computation. So, we can use the budget that light insertions provides to us for heavy insertions.

Intuitively, we have the following observation. At an epoch i, the independent set I_i has $\Theta(\frac{n_i^2}{m_i})$ vertices, the cut $(I_i, N(I_i))$ consists of $\Theta(n_i)$ edges and G_i contains m_i edges. So, for any change in the independent set I_i , the adversary needs to update $\Theta(\frac{m_i^2}{n_i^2})$ edges of the graph $G_i(V_i, E_i)$. As an example, if G_i has n_i vertices and $n_i \sqrt{n_i}$ edges, then $|I_i|$ has $\Theta(\sqrt{n_i})$ vertices, the adversary needs to update $\Theta(n_i)$ edges in order to change I_i . The same happens for edge deletions.

1.1 Preliminaries

Let G(V, E) be an undirected unweighted graph with n = |V| vertices and m = |E| edges. We assume that there is a unique numbering for the vertices in V so that we can treat $v \in V$ as a unique number v for $1 \le v \le n = |V|$. We denote an edge in E with two endpoints $u, v \in V$ by (u, v). The graph G can have at most $\binom{n}{2} = n(n-1)/2$ edges. Thus, each edge can also be thought of as referring to a unique number between 1 and $\binom{n}{2}$. Here $[x] = \{1, 2, 3, \cdots, x\}$ when $x \in \mathbb{N}$.

Given a vertex $v \in V$ we let $N_G(v) = \{u \in V : (u,v) \in E\}$ be the neighborhood of v. We let $d_G(v) = |N_G(v)|$ be the degree of the vertex v. When it is clear from the context we often drop G from $d_G(v)$ and $N_G(v)$ and simply write them as d(v) and N(v). The average degree of the graph G is $d(G) = \frac{1}{n} \cdot \sum_{v \in V} d_G(v)$.

Next we define a maximal independent set.

Definition 2 (Maximal Independent Set (MIS)) Given an undirected Graph G(V, E), an independent set is a subset of nodes $U \subseteq V$, such that no two nodes in U are adjacent. An independent set is maximal if no node can be added without violating independence.

There is a simple greedy algorithm that reports a MIS of G. In particular, we scan the nodes of G in arbitrary order. If a node u does not violate independence, we add u to the MIS. If u violates independence, we discard u.

Dynamic Model. Let S be a stream S of insertions and deletions of edges. We define time t to be the t-th operation (i.e., insertion or deletion) of stream S. Let I_t be a maximal independent set of an underlying graph $G_t(V, E_t)$ whose edge set E_t is the set of edges that are inserted up to time t but not deleted. The

update time of a dynamic algorithm \mathcal{A} is the time that \mathcal{A} needs to compute a MIS I_t of graph $G_t(V, E_t)$ given a MIS I_{t-1} of graph $G_{t-1}(V, E_{t-1})$. The update time can be worst-case or amortized.

Query Model. We assume the input graph G(V, E) is represented as an adjacency list. We could also assume that G is represented as an adjacency matrix, but adjacency matrices are often suitable for dense graphs where $|E| = \Theta(|V|^2)$. In dynamic scenarios we may end up with many edge deletions so that the graph become very sparse for which the adjacency matrix representation may not be appropriate. The complexity of dynamic algorithms for graph problems is often measured based on number of neighbor queries where for every vertex $v \in V$, we query its i-th neighbor. We assume that a neighbor querie takes constant time. Therefore, querying the full neighborhood of a vertex $v \in V$ takes $O(d_G(V))$ time. We let $\mathcal{Q}(\mathcal{A}, G)$ be the number of neighbor queries that an algorithm \mathcal{A} makes to compute a function.

In this paper, we use the following concentration bound.

Lemma 3 (Additive Chernoff Bound) [3] Let Y_1, \dots, Y_m denote m identically distributed and independent random variables such that $\mathbf{E}[Y_i] = p$ for $1 \le i \le n$ for a fixed $0 \le p \le 1$. Let $0 < t < 1, t \ge p$. For $Y = \sum_{i=1}^m Y_i$ it holds that

$$\mathbf{Pr}[Y \geq t \cdot m] \leq \left\lceil \left(\frac{p}{t}\right)^t \cdot \left(\frac{1-p}{1-t}\right)^{(1-t)} \right\rceil^m.$$

2 Maximal Independent Set (MIS)

The pseudocode of our offline MIS algorithm is given in Algorithm (1) Maximal-Independent-Set.

Algorithm 1 Maximal-Independent-Set

```
Input: Unweighted undirected graph G(V, E) with n = |V| vertices and m = |E| edges.
 1: Let i = 0 and G_i(V_i, E_i) = G(V, E). Let c = 34.
 2: while V_i \neq \emptyset do
        Let j=0, I_i=N(I_i)=\emptyset, n_i=|V_i|, m_i=|E_i|, and t=\frac{n_i^2}{c\cdot m_i}.
        while j \le max(t, 1) do
 4:
            while TRUE do
 5:
               Sample a vertex v \in V_r uniformly at random.
 6:
               if I_r \cup \{v\} is an independent set in the induced graph of V_r then
 7:
 8:
                  Break the true while loop.
            Let I_i = I_i \cup \{v\}, N(I_r) = N(I_r) \cup N_{G_R}(v) and j = j + 1.
 9:
        Let level L_i be the quadruple (V_i, \frac{n_i}{m_i}, I_i, N(I_i)).
 Let G_{i+1}(V_{i+1}, E_{i+1}) be the indued subgraph on V_{i+1} = V_i \setminus (I_i \cup N(I_i)).
10:
        Let i = i + 1.
Output: Return the level set \mathcal{L} = \bigcup_{i=1}^k L_i where k = |\mathcal{L}| = O(\log n) is the number of levels.
```

The MIS algorithm Algorithm (1) Maximal-Independent-Set is the same as the following MIS algorithm. Let G(V, E) be an undirected unweighted graph with n = |V| vertices and m = |E| edges. We consider k epochs during which we build levels L_1, \dots, L_k of independent vertices for $k = O(\log n)$. At the beginning of epoch i we assume we have a graph $G_i(V_i, E_i)$. For the first epoch, we let $G_1(V_1, E_1) = G(V, E)$.

At epoch i, we sample an ordered set $S_i \subseteq V_i$ of vertices with probability $\frac{|V_i|}{|E_i|}$ and we let I_i be a MIS that we find greedily for the induced sub-graph $H(S_i, E[S_i])$. If the graph G_i is sparse (i.e., $|E_I| \leq |V_i|$), we sample all vertices and the ordered set S_i will be a random shuffle of vertices of V_i .

We let $N(I_i)$ be the set of neighbors of I_i in the graph $G_i(V_i, E_i)$ and we remove I_i and $N(I_i)$ from $G_i(V_i, E_i)$. The level L_i consists of the vertex set V_i , the multiplicative inverse or reciprocal for the average degree of G_i which is $\frac{|V_i|}{|E_i|}$, and the independent set I_i and its neighbor set $N(I_i)$. We remove I_i and $N(I_i)$ from the graph G_i and recursively start the next epoch E_{i+1} .

The pseudocode of this algorithm is given Algorithm (2) Maximal-Independent-Set (Subset-Sampling).

Algorithm 2 Maximal-Independent-Set (Subset-Sampling)

Input: Unweighted undirected graph G(V, E) with n = |V| vertices and m = |E| edges.

- 1: Let i = 1 and $G_i(V_i, E_i) = G(V, E)$.
- 2: while $V_i \neq \emptyset$ do
- 3: Let S_i be a sample set where each vertex $v \in V_i$ is sampled with probability $\Pr[v] = \min(\frac{|V_i|}{|E_i|}, 1)$.
- 4: Let $H(S_i, E[S_i])$ be the induced subgraph of S_i , where $E[S_i] = \{(u, v) \in E_i : u \in S_i \text{ and } v \in S_i\}$.
- 5: Let I_i be the output MIS of the greedy MIS for the graph $H(S_i, E[S_i])$.
- 6: Let $N(I_i) = \{ v \in V_i \setminus I_i : \exists u \in I_i \text{ and } (u, v) \in H(S_i, E[S_i]) \}$ be the neighbor set of I_i .
- 7: Let level L_i be the quadruple $(V_i, \frac{|V_i|}{|E_i|}, I_i, N(I_i))$.
- 8: Let $G_{i+1}(V_{i+1}, E_{i+1})$ be the indued subgraph on $V_{i+1} = V_i \setminus (I_i \cup N(I_i))$.
- 9: Let i = i + 1.

Output: Return the levels $\mathcal{L} = \cup_{i=1}^k L_i$ where k is the number of levels.

2.1 Analysis

First we prove that the induced subgraph $H(S_i, E[S_i])$ is sparse, that is, $|E[S_i]| \le c \cdot |S_i|$ for a constant $c \ge 1$.

Lemma 4 Let $G_i(V_i, E_i)$ be an undirected unweighted graph at the beginning of epoch i of Algorithm 2. Assume that $|E_i| > |V_i|$. With probability at least 1/2, the number of vertices in S_i is $|S_i| \ge \frac{n_i^2}{4m_i} \ge \frac{|E[S_i]|}{16}$.

Proof : Let $n_i = |V_i|$ be the number of vertices in V_i and $m_i = |E_i|$ be the number of edges in E_i . Suppose the vertices in V_i are v_1, \cdots, v_{n_i} . Corresponding to the vertex v_j we define an indicator random variable X_j for the event that v_j is sampled. We define a random variable $X = \sum_{j \in [n_i]} X_j$. Since $\mathbf{E}[X_j] = \mathbf{Pr}[X_j] = \frac{|V_i|}{|E_i|} = \frac{n_i}{m_i}$, we have $\mathbf{E}[X] = \frac{n_i^2}{m_i}$. Using Markov Inequality, $\mathbf{Pr}[X \ge \frac{1}{4} \cdot \frac{n_i^2}{m_i}] \le 1/4$.

Next suppose the edges in E_i are e_1, \dots, v_{m_i} . Corresponding to the edge $e_j = (u_j, v_j)$ we define an indicator random variable Y_j for the event that E_j is in $E[S_i]$. We define a random variable $Y = \sum_{j \in [m_i]} Y_j$. Since $\mathbf{E}[Y_j] = \mathbf{Pr}[Y_j] = \mathbf{Pr}[u_j, v_j \in S_i] = \mathbf{Pr}[u_j \in S_i] \cdot \mathbf{Pr}[v_j \in S_i] = (\frac{|V_i|}{|E_i|})^2 = (\frac{n_i}{m_i})^2$. We then have $\mathbf{E}[Y] = \frac{n_i^2}{m_i}$. Using Markov Inequality, $\mathbf{Pr}[X \ge 4 \cdot \frac{n_i^2}{m_i}] \le 1/4$.

Thus, using the union bound, with probability at least 1/2, $|S_i| \ge \frac{n_i^2}{4m_i} \ge \frac{|E[S_i]|}{16}$.

Now we prove that the independent set I_i reported at Epoch i of Algorithm 2 is relatively big with respect to the sampled set size S_i and also a constant fraction of vertices in V_i are neighbors of I_i that can be removed once we recurse the sampling process for the graph G_{i+1} .

Lemma 5 Let $H(S_i, E[S_i])$ be the induced subgraph reported at Epoch i of Algorithm 2. Then, the random greedy algorithm for the maximal independent set problem returns an independent set I_i of size $|I_i| \ge \frac{|S_i|}{3d}$.

Proof : First we find the lower bound on the size of the independent set I_i . Using Lemma 4 we have $|S_i| \geq \frac{n_i^2}{4m_i} \geq \frac{|E[S_i]|}{16}$. Therefore, the average degree of $H(S_i, E[S_i])$ is upper bounded by $d(G_i) = \frac{|E[S_i]|}{|S_i|} \leq 16$ which means that the independent set I_i is of size $|I_i| \geq \frac{|S_i|}{2(d(G_i)+1)} = \frac{|S_i|}{34}$.

Lemma 6 Let I_i be the independent set in the graph $G_i(V_i, H_i)$ that is reported by Algorithm 2. Let $N(I_i) = \{v \in V_i | \exists u \in I_i : (u, v) \in E_i\}$ be the set of vertices in G_i that are neighbors of I_i . Then, we have $\mathbf{Pr}[|N(I_i)| \geq \frac{n_i}{900}] \geq 2/3$.

Proof : Let us consider the independent set $I_i = \{u_1, \cdots, u_t\}$ in the graph $G_i(V_i, E_i)$ where $t = \frac{|V_i|^2}{34 \cdot |E_i|}$. Suppose when we sample the vertex u_j , the set $N(I_i^{j-1})$ is the set of vertices of G_i that are neighbor to one of the vertices u_1, \cdots, u_{j-1} . That is, $N(I_i^{j-1}) = \{v \in V_i | \exists 1 \leq \ell \leq j-1 : (v, u_\ell) \in E_i\}$. Assume that $|N(I_i^{j-1})| < |V_i|/2$; otherwise, removing the pair set $(I_i, N(I_i))$ from the graph G_i drops the number of vertices by half and we can recurse with the induced subgraph of the remaining vertex set.

Now suppose we sample the vertex u_j . We define a random variable X_j for the number of vertices in $V_i \setminus N(I_i^{j-1})$ that are neighbors of u_j . In expectation we have $\mathbf{E}[X_j] = d_{G_i}(u_j) \cdot \frac{|V_i \setminus N(I_i^{j-1})|}{|V_i|}$ where $d_{G_i}(u_j)$ is the degree of u_j in G_i . Let us define a random variable $X = \sum_{i=1}^t X_j$. We then have

$$\mathbf{E}[X] = \sum_{j=1}^t \mathbf{E}[X_j] = \sum_{j=1}^t d_{G_i}(u_j) \cdot \frac{|V_i \backslash N(I_i^{j-1})|}{|V_i|} \geq \frac{1}{2} \cdot \sum_{j=1}^t d_{G_i}(u_j) \ .$$

Now corresponding to the vertex u_j we define a random variable Y_j for the degree of u_j . We also define a random variable $Y = \sum_{j \in [t]} Y_j$. Observe that $\mathbf{E}[Y_j] = \frac{m_i}{n_i}$. Therefore, we have

$$\mathbf{E}[Y] = t \cdot \frac{m_i}{n_i} \ge \frac{n_i^2}{34 \cdot m_i} \cdot \frac{m_i}{n_i} = \frac{n_i}{34} .$$

We then apply Markov Inequality to obtain

$$\text{Pr}[Y \leq \frac{n_i}{136}] = \text{Pr}[\sum_{u_i \in I_i} d_{G_i}(u_j) \leq \frac{n_i}{136}] \leq 1/4 \ .$$

Therefore, with probability at least 3/4, $\sum_{u_i \in I_i} d_{G_i}(u_j) \ge \frac{n_i}{136}$.

This essentially yields $\mathbf{E}[X] \geq \frac{1}{2} \cdot \sum_{j=1}^t d_{G_i}(u_j) \geq \frac{n_i}{272}$ and we apply the Markov inequality to prove that $\mathbf{Pr}[|N(I_i)| \geq \frac{n_i}{900}] \geq 2/3$.

We can increase the success probability of Algorithm (1) Maximal-Independent-Set to $1 - \delta/n^3$ by creating $x = 3\log(n/\delta)$ runs R_1, \cdots, R_x of this algorithm in parallel and report the MIS of the run R_i whose neighborhood size is at least $\frac{n_i}{900}$.

Next we prove that at the end of a level L_i , for each vertex $v \in V_i$, either v is deleted from the remaining graph G_{i+1} or the degree of v in G_{i+1} is upper-bounded by $O(\log n \cdot \frac{m_i}{n_i})$.

Lemma 7 Let L_i be a level in the level set \mathcal{L} . With probability at least $1-1/n^2$, each vertex $v \in V_i$ is either added to $I_i \cup N(I_i)$ and will not appear in G_{i+1} or the degree of v in the subgraph $G_{i+1}(V_{i+1}, E_{i+1})$ is $d_{G_{i+1}}(v) \leq \frac{3c \log n \cdot m_i}{n_i}$.

Proof : Let us consider a graph $G_i(V_i, E_i)$ at a level L_i where $n_i = |V_i|$ and $m_i = |E_i|$. In the beginning of the random sampling process at level L_i , both I_i and $N(I_i)$ are empty sets. Our sampling subroutine repeats the following process for $s = \frac{n_i^2}{cm_i}$ times: Repeat sampling (with replacement) a vertex $w \in V_i$ uniformly at random as long as $I_i \cup \{w\}$ is not an independent set in G_i . Once we sample a vertex w for which $I_i \cup \{w\}$ is an independent set, we then let $I_i = I_i \cup \{w\}$ and $N(I_i) = N(I_i) \cup N_{G_i}(w)$.

Let us consider the process of building the independent set I_i incrementally. That is, at each step $t \in [s]$, let $I_i^t = \{w_1, \cdots, w_t\}$ be an independent set that we found for G_i . Let $N(I_i^t)$ be the set of neighbors of I_i^t till step t. Let $v \in V_i$ be a vertex with the neighbor set $N_{G_i}(v)$ and degree $d_{G_i}(w)$. Suppose $d_{G_i}(v) \geq \frac{3c \log n \cdot m_i}{n_i}$ as otherwise nothing left to prove.

Let $X_{\nu}^t = N_{G_i}(\nu) \cup \{\nu\} \setminus (I_i^t \cup N(I_i^t))$ be the set of neighbors of ν (including ν) that are not in the independent set I_i^t or adjacent to a vertex in I_i^t . Observe that if at step t we sample a vertex $w \in X_{\nu}^t$, then $I_i^{t-1} \cup \{w\}$ will be an independent set and we can let $\nu_t = w$. If that happens, $\nu \in I_i \cup N(I_i)$ and the vertex ν is eliminated from G_{i+1} . So, suppose this does not happen. We then define a random event \mathcal{BAD}_{ν}^t for $|X_{\nu}^t| \geq \frac{3c \log n \cdot m_i}{n_i}$ but $\nu_t \notin X_{\nu}^t$ at step t.

Observe that
$$\mathbf{Pr}[\mathcal{BAD}_{v}^{t}] \leq 1 - \frac{\frac{3c \log n \cdot m_{i}}{n_{i}}}{n_{i}} = 1 - \frac{3 \log n \cdot m_{i}}{n_{i}^{2}}$$
. Then,

$$\mathbf{Pr}[\mathcal{BAD}_{\nu}^{1} \wedge \cdots \wedge \mathcal{BAD}_{\nu}^{s}]$$

$$\begin{split} &= \text{Pr}[\mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{1}] \cdot \text{Pr}[\mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{2}|\mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{1}] \cdot \text{Pr}[\mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{3}|\mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{1} \wedge \mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{2}] \cdots \text{Pr}[\mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{s}|\mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{1} \wedge \mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{2} \wedge \cdots \wedge \mathcal{B}\mathcal{A}\mathcal{D}_{\nu}^{s-1}] \\ &\leq (1 - \frac{3\log n \cdot m_{i}}{n_{i}^{2}})^{s} = (1 - \frac{3c\log n \cdot m_{i}}{n_{i}^{2}})^{\frac{n_{i}^{2}}{cm_{i}}} \leq e^{-3\log n} \leq 1/n^{3} \ . \end{split}$$

Using the union bound argument with probability at least $1-1/n^2$, each vertex $v \in V_i$ is either added to $I_i \cup N(I_i)$ and will not appear in G_{i+1} or the degree of v in the subgraph $G_{i+1}(V_{i+1}, E_{i+1})$ is $d_{G_{i+1}}(v) \leq \frac{3c \log n \cdot m_i}{n_i}$.

3 Edge Insertion and Deletion

Here in this section we describe our edge insertion and deletion subroutines.

3.1 Insertion and Deletion Subroutines

Let us first consider the insertion of an edge e = (u, v). The insertion of e can trigger one of the following cases:

Let $\mathcal{L}=\cup_{i=1}^k L_i$ be a level set of a graph G(V,E). Let $1\leq i\leq j\leq k$ be two level indices. An edge insertion $e=(u,\nu)$ triggers

- $(i \leftrightarrow j)$ -Light Insertion if $u \in N(I_i)$ and either $v \in N(I_i)$ or $v \in V_i$.
- $(i \leftarrow j)$ -Light Promotion if $u \in I_i$ and $v \in N(I_i)$.
- $(i \leftarrow j)$ -Heavy Promotion if $u \in I_i$ and $v \in I_j$.

First suppose the insertion of an edge e = (u, v) triggers a light insertion. That is, there exists $1 \le i \le j \le k$ for which $u \in N(I_i)$ and either $v \in N(I_i)$ or $v \in V_i$. We then only need to add e to the neighborhood

```
Algorithm 3 Edge Insertion and Deletion Subroutines
Edge-Insertion (The level set \mathcal{L} = \bigcup_{i=1}^k L_i and an edge e = (u, v))
  1: if u \in I_i and v \in N(I_i) for i, j \in [k] and i < j then
         Invoke \mathcal{L} = Light-Promotion (\mathcal{L}, v, i)
  3: if u \in I_i and v \in I_j for i, j \in [k] and i \le j then
         Invoke \mathcal{L} = Light-Promotion (\mathcal{L}, v, i)
         Invoke \mathcal{L} = Heavy-Promotion (\mathcal{L}, v)
Edge-Deletion (The level set \mathcal{L} = \bigcup_{i=1}^k L_i and an edge e = (u, v))
  1: if u \in I_i and \nu \in N(I_i) \backslash N(I_i \backslash \{u\}) then
         Invoke \mathcal{L} = Demotion(\mathcal{L}, v, i)
Light-Promotion (The level set \mathcal{L} = \bigcup_{i=1}^k L_i, a vertex \nu and a level r < L(\nu))
  1: Let j = L(v) be the level of the vertex v.
  2: for level r < \ell \le j do
        Let V_{\ell} = V_{\ell} \setminus \{v\} where V_{\ell} is the vertex set in the level L_{\ell} \in \mathcal{L}.
  3: N(I_r) = N(I_r) \cup \{v\} and N(I_i) = N(I_i) \setminus \{v\}.
Heavy-Promotion (The level set \mathcal{L} = \cup_{i=1}^k L_i and a vertex \nu with level j = L(\nu))
  1: Let F = N(I_i) \setminus N(I_i \setminus \{\nu\}) be the neighbors of N(I_i) that become free if we remove \nu from I_i.
  2: Let I_i = I_i \setminus \{v\} be the independent set I_i after removal of v.
  3: for each vertex w \in F do
         Invoke \mathcal{L} = Demotion(\mathcal{L}, w, j)
Demotion (The level set \mathcal{L} = \bigcup_{i=1}^k L_i and a vertex w that is in N(I_j) for j \in [k])
  1: Let P(w) = N_G(w) \cap \mathcal{I} be the set of neighbors of w that are in MIS \mathcal{I} = \bigcup_{i=1}^k I_i.
  2: if P(w) is not empty then
        Let z \in P(w) be a vertex with the lowest level L(z) \le \min_{x \in P(w)} L(x).
         N(I_j) = N(I_j) \setminus \{w\} and N(I_{L(z)}) = N(I_{L(z)} \cup \{w\}.
  4:
  5: else
  6:
         for r in range (j, k) do
            Sample w with probability \mathbf{Pr}[w] = \frac{n_r}{cm_r}.
  7:
            if w is sampled and I_r = I_r \cup \{w\} is an independent set in G_r then
  8:
               Let I_r = I_r \cup \{w\}.
  9:
               for each vertex z \in N_{G_n}(w) do
10:
                   Invoke \mathcal{L} = \text{Light-Promotion}(\mathcal{L}, z, r)
11:
               Break the loop for r.
12:
Output: Return the level set \mathcal{L} = \bigcup_{i=1}^k L_i.
```

of $\mathfrak u$ and $\mathfrak v$, i.e., $N_G(\mathfrak u)$ and $N_G(\mathfrak v)$, and add $\mathfrak e$ to the neighbor set $N(I_i)$. We also need to update the density $\frac{n_r}{m_r}$ for $1 \le r \le j$. The density update of each level is done automatically and we move it to the pseudocode of Algorithm (4) Dynamic-MIS for the sake of simplicity of insertion and deletion subroutines.

Second suppose the insertion of an edge $e=(\mathfrak{u},\nu)$ triggers a light promotion. That is, there exists $1\leq i\leq j\leq k$ for which $\mathfrak{u}\in I_i$ and $\mathfrak{v}\in N(I_j)$. We promote \mathfrak{v} from the neighbor set $N(I_i)$ up to the neighbor set $N(I_i)$. We then eliminate \mathfrak{v} from each vertex set V_ℓ for $i<\ell\leq j$. Since $k=O(\log n)$, the light promotion subroutine takes $O(\log n)$ time.

Finally, we consider the case when the insertion of an edge e = (u, v) triggers a heavy promotion. That is, there exists $1 \le i \le j \le k$ for which $u \in I_i$ and $v \in I_j$. The vertex v is moved from I_j to $N(I_i)$. By this operation, all neighbors of v in G_j that are not incident to any other vertex in $I_j \setminus \{v\}$ (that is, $w \in F = N(I_j) \setminus N(I_j \setminus \{v\})$) become free. For every such a vertex w we demote w. That is, if there exists a vertex in one of independent sets I_r for $r \ge j$ we demote w to the level L_r and add it to $N(I_r)$. Otherwise, we check to see if we can add w to an independent set I_r for $r \ge j$. In particular, for each level L_r for $j \le r \le k$ with probability $\frac{n_r}{cm_r}$ and only if $I_r \cup \{w\}$ is an independent set in G_r we add w to I_r and promote vertices in $N_{G_r}(w)$ to the level L_r and add them to $N(I_r)$. Since w is not adjacent to any vertex in an independent set I_r , the promotion of vertices $N_{G_r}(w)$ takes at most $d_{G_r}(w) \le d_{G_i}(w)$ time.

As for the deletion of an arbitrary edge $e=(\mathfrak{u},\nu)$, if $\mathfrak{u}\in I_i$ and $\nu\in N(I_i)\setminus N(I_i\setminus\{\mathfrak{u}\})$, we demote the vertex ν . That is, if ν is adjacent to any independent set $I_{r\geq i}$, we demote ν to $N(I_r)$, otherwise we downsample ν with probability $\frac{n_r}{cm_r}$ for $r\geq j$ and check if we can add it to I_r the same as edge insertion.

Finally at any time t if there exists a level L_r whose density $\frac{m_r}{n_r}$ is increased or decreased by a factor of at least two, we recompute the maximal independent sets of all levels $L_{\ell \geq r}$. The density update of each level is done automatically and is moved to the pseudocode of Algorithm (4) Dynamic-MIS.

3.2 Analysis

Let $\mathcal{L} = \bigcup_{i=1}^k L_i$ be a level set of an underlying graph G(V, E). Recall that given \mathcal{L} , the reported maximal independent set is $\mathcal{I} = \bigcup_{i=1}^k I_i$. Let $e = (u, v) \in E$ be an arbitrary edge added to the graph G.

We first find an upper-bound for the probability that adding an arbitrary edge triggers a heavy promotion.

Lemma 8 Let $1 \le i \le j \le k$ be two level indices. Let c = 200. The probability that adding an arbitrary edge $e = (\mathfrak{u}, \mathfrak{v})$ triggers an $(i \leftarrow j)$ -heavy promotion is at most $\frac{2}{c^2} \cdot \frac{\mathfrak{n}_i}{\mathfrak{m}_i} \cdot \frac{\mathfrak{n}_j}{\mathfrak{m}_j}$. That is,

$$\text{Pr}[u \in I_i \text{ and } v \in I_j] \leq \frac{2}{c^2} \cdot \frac{n_i}{m_i} \cdot \frac{n_j}{m_i} = \frac{2}{c^2} \cdot d^{-1}(G_i) \cdot d^{-1}(G_j) \ ,$$

where $d^{-1}(G_i)$ and $d^{-1}(G_j)$ are the multiplicative inverses or reciprocals for the average degree of G_i and G_j , respectively.

Proof: We define a random event $\mathcal{HP}_{i\leftarrow j}$ for $u\in I_i$ and $v\in I_j$. We sample vertices in the graphs G_i and G_j with probabilities $\frac{n_i}{cm_i}$ and $\frac{n_j}{cm_j}$, respectively. Therefore, $\mathbf{E}[|I_i|] = \frac{n_i^2}{cm_i}$ and $\mathbf{E}[|I_j|] = \frac{n_j^2}{cm_j}$. Since $n_j \leq n_i$, we then have

$$\textbf{Pr}[\mathcal{H}_{i \leftarrow j}] = \frac{\frac{n_i^2}{cm_i} \cdot \frac{n_j^2}{cm_j}}{\binom{n_i}{2}} = \frac{2 \cdot \frac{n_i^2}{cm_i} \cdot \frac{n_j^2}{cm_j}}{n_i(n_i-1)} \leq \frac{2n_in_j}{c^2m_im_j} = \frac{2}{c^2} \cdot \frac{n_i}{m_i} \cdot \frac{n_j}{m_j} = \frac{2}{c^2} \cdot d^{-1}(G_i) \cdot d^{-1}(G_j) \ .$$

We can also define a random event \mathcal{HP} if there exist two indices $1 \le i \le j \le k$ for which we have

 $u \in I_i$ and $v \in I_i$.

$$\begin{split} \textbf{Pr}[\mathcal{HP}] &= \frac{\binom{|\mathcal{I}|}{1}}{\binom{n}{2}} = \frac{|\mathcal{I}| \cdot (|\mathcal{I}|-1)}{n(n-1)} \leq \sum_{i \in [k]} \sum_{j \in [k]} \textbf{Pr}[\mathcal{H}_{i \leftarrow j}] \leq \sum_{i \in [k]} \sum_{j \in [k]} 2 \frac{n_i}{cm_i} \cdot \frac{n_j}{cm_j} \\ &= \sum_{i \in [k]} \sum_{j \in [k]} \frac{2}{c^2} \cdot d^{-1}(G_i) \cdot d^{-1}(G_j) \enspace . \end{split}$$

Next we bound the expected number of queries that our insertion and deletion subroutines need to recompute a maximal independent set after a light insertion, a light promotion or a heavy promotion happen.

Lemma 9 Let $e = (u, v) \in E$ be an arbitrary edge. Let c_{LI} be a large enough constant. Suppose that e triggers a light insertion which happens if there exists $1 \le i \le j \le k$ for which $u \in N(I_i)$ and either $v \in N(I_i)$ or $v \in V_j$. Then, $Q(Light-Insertion(v), G) = c_{LI} \log n$.

Proof : If e triggers a light insertion, we need to add e to the neighborhood of u and v, i.e., $N_G(u)$ and $N_G(v)$, and add e to the neighbor set $N(I_i)$. We also need to update the density $\frac{n_r}{m_r}$ for $1 \le r \le j$. This can be done using three query and update operations plus $O(\log n)$ density updates. Thus, $\mathcal{Q}(Light\text{-Insertion}(v), G) = c_{LI} \cdot \log n$ for large enough constant c_{LI} .

Lemma 10 Let $e = (u, v) \in E$ be an arbitrary edge. Let c_{LP} be a large enough constant. Suppose that e triggers a light promotion that occurs when there exists $1 \le i \le j \le k$ for which $u \in I_i$ and $v \in N(I_j)$. Then, $\mathcal{Q}(\textit{Light-Promotion}(v), G) = c_{LP} \cdot \log n$.

Proof : We promote ν from $N(I_j)$ up to $N(I_i)$. We then eliminate ν from each vertex set V_ℓ for $i < \ell \le j$. Since $k = O(\log n)$, the light promotion subroutine takes $\mathcal{Q}(Light\text{-Promotion}(\nu)), G) = c_{LP} \cdot \log n$ time for large enough constant c_{LI} .

Lemma 11 Let $e = (u, v) \in E$ be an arbitrary edge. Let c_{HP} be a large enough constant. Suppose that e triggers a heavy promotion that occurs when there exists $1 \le i \le j \le k$ for which $u \in I_i$ and $v \in I_j$. Then,

$$\mathbf{E}[\mathcal{Q}(\textit{Heavy-Promotion}(\nu),G)] \leq c_{\mathsf{HP}}\log n \cdot \mathbf{E}[\sum_{w \in N_{G_j}(\nu)} d_{G_j}(w)] \leq c_{\mathsf{HP}}\log n \cdot \frac{3c\log n \cdot m_i}{n_i} \cdot \frac{m_j}{n_j} \ .$$

Proof : The vertex ν is moved from I_j to $N(I_i)$. By this operation, all neighbors of ν in G_j that are not incident to any other vertex in $I_j \setminus \{\nu\}$ (that is, vertices in $F = N(I_j) \setminus N(I_j \setminus \{\nu\})$) become free. For each vertex $w \in F$ one of the following cases can happen.

Case 1: If there exists a vertex in one of independent sets I_r for $r \geq j$, we then demote w to the level L_r and add it to $N(I_r)$. To this end, we need to query the neighborhood of v in G_j which takes $O(d_{G_j}(v))$. Observe that $\mathbf{E}[d_{G_j}(v)] = \frac{m_i}{n_i}$. We also need to update the vertex sets V_ℓ and update the density $\frac{m_\ell}{n_\ell}$ for $j < \ell \leq r$ what needs $O(\log n)$ query updates.

Case 2: Otherwise, for each level L_r for $j \leq r \leq k$ with probability $\frac{n_r}{cm_r}$ and only if $I_r \cup \{w\}$ is an independent set in G_r we add w to I_r and promote vertices in $N_{G_r}(w)$ to the level L_r and add them to $N(I_r)$. Since w is not adjacent to any vertex in an independent set I_r , the promotion of vertices $N_{G_r}(w)$ takes at most $d_{G_r}(w) \leq d_{G_j}(w) \cdot O(\log n)$ time where we need to update the sets $V_{\ell > r}$ and the density $\frac{m_\ell}{n_\ell}$ as the vertices in $N_{G_r}(w)$ are promoted to the level r.

Let us study the expected value of the random variable $X = \sum_{w \in N_{G_j}(v)} d_{G_j}(w)$. We consider two cases either i < j or i = j.

For the case when i < j for every $w \in N_{G_j}(v)$ using Lemma 7 with probability at least $1 - 1/n^2$ we have $d_{G_j}(w) < \frac{3c \log n \cdot m_i}{n_i}$. Since the edge e = (u, v) is chosen arbitrary, we have $\mathbf{E}[d_{G_j}(v)] = \frac{m_j}{n_j}$. Therefore,

$$\mathbf{E}[X] = \mathbf{E}[\sum_{w \in N_{G_j}(v)} d_{G_j}(w)] \le \frac{3c \log n \cdot m_i}{n_i} \cdot \mathbf{E}[d_{G_j}(v)] = \frac{3c \log n \cdot m_i}{n_i} \cdot \frac{m_j}{n_j} ,$$

what yields

$$\mathcal{Q}(\text{Heavy-Promotion}(\nu)), G) \leq c_{\text{HP}} \log n \cdot \mathbf{E}[\sum_{w \in N_{G_i}(\nu)} d_{G_j}(w)] = c_{\text{HP}} \log n \cdot \frac{3c \log n \cdot m_i}{n_i} \cdot \frac{m_j}{n_j} \ .$$

The harder case is when i=j, especially when i=j=1 where we need to upper-bound the term $\mathbf{E}[\sum_{w\in N_{G_j}(\nu)}d_{G_j}(w)]$. Observe that we can choose either \mathfrak{u} or ν . So, the question boils down to study the expected sum of degrees of neighbors of a random vertex in G_i for which we use Claim 12 to show that $\mathbf{E}[\sum_{w\in N_{G_i}(\nu)}d_{G_j}(w)]=\frac{\mathfrak{m}_i}{\mathfrak{n}_i}\cdot\frac{\mathfrak{m}_i}{\mathfrak{n}_i}$ what proves this lemma.

Claim 12 Let ν be a vertex that we sample uniformly at random from an independent set I_i of a level L_i for $i \in [k]$. Then, $\mathbf{E}[\sum_{w \in N_{G_i}(\nu)} d_{G_j}(w)] = \frac{m_i}{n_i} \cdot \frac{m_i}{n_i}$.

Proof : Let us define a random variable X corresponding to the value $\sum_{w \in N_{G_j}(v)} d_{G_j}(w)$ of a random vertex $v \in I_i$. Then, we have

$$\begin{split} \mathbf{E}[X] &= \mathbf{E}[\sum_{w \in N_{G_j}(\nu)} d_{G_j}(w)] = \sum_{\nu \in G_i} \mathbf{Pr}[\nu] d_{G_i}(\nu) \cdot \sum_{w \in G_i} \mathbf{Pr}[\nu] \text{ incident to } w \text{ in } G_i] \\ &= \sum_{\nu \in G_i} \mathbf{Pr}[\nu] d_{G_i}(\nu) \cdot \sum_{w \in G_i} \frac{d_{G_i}(w)}{n_i} = \sum_{\nu \in G_i} \frac{d_{G_i}(\nu)}{n_i} \cdot \sum_{w \in G_i} \frac{d_{G_i}(w)}{n_i} = \frac{m_i}{n_i} \cdot \frac{m_i}{n_i} \end{split}$$

Lemma 13 Let e = (u, v) be an arbitrary edge that is added to a graph G(V, E) whose level set is $\mathcal{L} = \bigcup_{i=1}^k L_i$. The expected number of queries that Algorithm Edge-Insertion makes to update the level set \mathcal{L} is $\mathbf{E}[\mathcal{Q}(Edge\text{-Insertion}(e = (u, v))), G)] = c \cdot \log^2 n$ where c is a large enough constant.

Proof: Suppose that $u \in L_i$ and $v \in L_j$ for $i \le j$. We define a random variable X for the number of queries that the insertion of an arbitrary edge e = (u, v) causes. In the following we study the expectation of X.

We define a random events $\mathcal{LI}_{i\leftarrow j}(e)$, $\mathcal{LP}_{i\leftarrow j}(e)$, and $\mathcal{HP}_{i\leftarrow j}(e)$ when the insertion of an arbitrary edge $e=(\mathfrak{u},\nu)$ triggers a light insertion, a light promotion and a heavy promotion, respectively. Observe that $\mathbf{Pr}[\mathcal{LI}_{i\leftarrow j}(e)] + \mathbf{Pr}[\mathcal{LP}_{i\leftarrow j}(e)] + \mathbf{Pr}[\mathcal{HP}_{i\leftarrow j}(e)] = 1$. Recall that using Lemmas 9, 10, 11, $\mathcal{Q}(\text{Light-Insertion}(\nu), G) = c_{LI} \cdot \log n$, $\mathcal{Q}(\text{Light-Promotion}(\nu), G) = c_{LP} \cdot \log n$, and

$$\mathbf{E}[\mathcal{Q}(\text{Heavy-Promotion}(\nu),G)] \leq c_{\text{HP}}\log n \cdot \mathbf{E}[\sum_{w \in N_{G_i}(\nu)} d_{G_j}(w)] = c_{\text{HP}}\log n \cdot \frac{3c\log n \cdot m_i}{n_i} \cdot \frac{m_j}{n_j} \enspace .$$

$$\begin{split} \mathbf{E}[X] &= \textbf{Pr}[\mathcal{LI}_{i \leftarrow j}(e)] \cdot \mathcal{Q}(\text{Light-Insertion}(\nu)), G) + \textbf{Pr}[\mathcal{LP}_{i \leftarrow j}(e)] \cdot \mathcal{Q}(\text{Light-Promotion}(\nu), G) \\ &+ \textbf{Pr}[\mathcal{HP}_{i \leftarrow j}(e)] \cdot \mathbf{E}[\mathcal{Q}(\text{Heavy-Promotion}(\nu), G)] \\ &\leq (c_{LI} + c_{LP}) \cdot \log n + \textbf{Pr}[\mathcal{HP}_{i \leftarrow j}(e)] \cdot c_{HP} \log n \cdot \frac{3c \log n \cdot m_i}{n_i} \cdot \frac{m_j}{n_j} \\ &\leq (c_{LI} + c_{LP} + c_{HP}) \log n \cdot (1 + \frac{6 \log n}{c}) \leq \frac{7(c_{LI} + c_{LP} + c_{HP})}{c} \cdot \log^2 n \enspace , \end{split}$$

since $\mathbf{Pr}[\mathcal{LI}_{i\leftarrow j}(e)] + \mathbf{Pr}[\mathcal{LP}_{i\leftarrow j}(e)] \leq 1$ and according to Lemma 8 the event $\mathcal{HP}_{i\leftarrow j}$ occurs for an arbitrary edge e = (u, v) with probability $\mathbf{Pr}[\mathcal{HP}_{i\leftarrow j}] \leq \frac{2}{c^2} \cdot \frac{n_i}{m_i} \cdot \frac{n_j}{m_j}$.

Corollary 14 Let e = (u, v) be an arbitrary edge that is deleted from a graph G(V, E) whose level set is $\mathcal{L} = \cup_{i=1}^k L_i$. The expected number of queries that Algorithm Edge-Deletion makes to update the level set \mathcal{L} is $\mathbf{E}[\mathcal{Q}(Edge-Deletion(e = (u, v))), G)] = c \cdot \log^2 n$ where c is a large enough constant.

The proof of this corollary is the same as the proof of Lemma 13 and we omit it here.

4 Analysis for Stream of Insertions and Deletions

In this section we present our dynamic algorithm. The pseudocode of this algorithm is given in below. Lemma 15 proves that the amortized update time of this algorithm is $O(\log^3 n)$.

Algorithm 4 Dynamic-MIS

Input: A Sequence $S = \{ \text{Update}(e_1 = (u_1, v_1)), \dots, \text{Update}(e_z = (u_z, v_z)) \}$ of edge updates to a graph G where $\text{Update}(e_\ell)$ is either $\text{Insertion}(e_\ell)$ or $\text{Deletion}(e_\ell)$.

- 1: Let G(V, E) be undirected unweighted graph whose vertex set V of size n = |V| is fixed and edge set E that can be initialized to an empty set.
- 2: Initialize $y = 4 \log(n/\delta)$ runs R_1, \dots, R_n in parallel.
- 3: for R_r where $r \in [y]$ in parallel do
- 4: Invoke Algorithm (2) Maximal-Independent-Set(G(V, E)) whose output is a level set $\mathcal{L}^r = \bigcup_{i=1}^k L_i$.
- 5: **for** each update Update (e_{ℓ}) **do**
- 6: **if** Update(e_{ℓ}) is Insertion(e_{ℓ}) **then**
- 7: Invoke $\mathcal{L}^{r} = Edge-Insertion(e_{\ell})$.
- 8: **else**
- 9: Invoke $\mathcal{L}^{r} = Edge-Deletion(e_{\ell})$.
- 10: **if** there exists a level L_i whose density $\frac{m_i}{n_i}$ changes by a factor 2 **then**
- 11: Invoke Algorithm (2) Maximal-Independent-Set($G_i(V_i, E_i)$) that updates level set $\mathcal{L}^r = \bigcup_{i=1}^k L_i$.
- 12: **if** the number of queries $\mathcal{Q}(\mathcal{A},G)$ that the run R_r made up to now is greater than $3cz \cdot log^3 n$ **then**
- 13: Stop the run R_r .

Output: At any time $t \in [z]$, report the MIS maintained by a level set $\mathcal{L}^r = \bigcup_{i=1}^k L_i$ whose run R_r survives.

Theorem 15 Let G = (V, E) be an undirected unweighted graph whose level set is $\mathcal{L} = \bigcup_{i=1}^k L_i$. Let $\mathcal{S} = \{Update(e_1 = (u_1, v_1)), \cdots, Update(e_z = (u_z, v_z))\}$ be a sequence of edge updates to a graph G

where $Update(e_{\ell} = (u_{\ell}, v_{\ell}))$ is either $Insertion(e_{\ell} = (u_{\ell}, v_{\ell}))$ or $Deletion(e_{\ell} = (u_{\ell}, v_{\ell}))$. Let $0 < \delta < 1$ be a parameter. There is a randomized algorithm that with probability at least $1 - \delta/n^2$, applies this sequence of updates to G and updates the level set \mathcal{L} in time $O(z \cdot \log^3 n)$. That is, the amortized update time of this algorithm is $O(\log^3 n)$.

Proof : Let us define z random variables X_1, \dots, X_z corresponding to these edge updates where X_ℓ corresponds to the number of queries that the update of the edge $e_\ell = (\mathfrak{u}_\ell, \nu_\ell)$ needs to update the level set $\mathcal{L} = \cup_{i=1}^k \mathsf{L}_i$. From Lemma 13 and Corollary 14, we have

$$\begin{split} \mathbf{E}[X_\ell] &\leq \text{max}(\mathbf{E}[\mathcal{Q}(\text{Edge-Insertion}(e_\ell)), G)], \mathbf{E}[\mathcal{Q}(\text{Edge-Deletion}(e_\ell)), G)]) = c \cdot \log^2 n \ . \\ \text{Let } X &= \sum_{\ell=1}^z X_\ell. \text{ We then have } \mathbf{E}[X] = cz \cdot \log^2 n. \text{ Using Markov Inequality,} \\ \mathbf{Pr}[X &\geq 3cz \cdot \log^2 n] \leq 1/3 \ . \end{split}$$

Next we increase the probability of correctness to $1 - \delta/n^3$. For the sequence Update($e_1 = (u_1, v_1)$), \cdots , Update($e_z = (u_z, v_z)$) of edge updates, we run $y = 4 \log(n/\delta)$ instances of Algorithms Edge-Insertion and Edge-Deletion in parallel. Let R_1, \cdots, R_y be the set of these y runs. At any time $1 \le t \le z$, if we observe that the sum of the number of queries that a run R_r makes from the beginning of the sequence (i.e., time 1) up to t is greater than $3cz \cdot \log^2 n$, we stop the run R_r .

Let Y_r corresponds to the run R_r such that $Y_r=1$ if for the r-th run the sum of the number of queries that R_r makes from time 1 to t is is greater than $3cz \cdot \log^2 n$, and $Y_r=0$ otherwise. Therefore, $\mathbf{E}[Y_r]=p \le 1/3$. Let $\alpha=1/2$ and $Y=\sum_{r=1}^{y}Y_r$. Using additive Chernoff Bound 3 we then have,

$$\mathbf{Pr}[Y \geq y/2] \leq \left\lceil (\frac{p}{\alpha})^{\alpha} \cdot \left(\frac{1-p}{1-\alpha}\right)^{1-\alpha} \right\rceil^{y} \leq \left\lceil \sqrt{2/3} \cdot \sqrt{\frac{2/3}{1/2}} \right\rceil^{y} \leq (\sqrt{8/9})^{y} \leq \delta/n^{4} \ ,$$

for $y \ge 4\log_{\sqrt{9/8}}(n/\delta)$. Be the relation between logarithms we then have $y \ge 4\log(n/\delta)$.

We can assume that $z \le n(n+1)/2 = n^2/2 + n/2 \le n^2$. Since after every n^2 updates we re-run the MIS algorithm (i.e., Algorithm 2) from the beginning. Therefore, using a union bound, with probability at least $1 - \delta/n^2$ after every update there exists at least one run that survives.

References

- [1] Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear update time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 815–826, 2018.
- [2] Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear in n update time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2019, San Diego, California, USA, January 6-9, 2019, pages 1919–1936, 2019.
- [3] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23(4):493 507, 1952.
- [4] Yuhao Du and Hengjie Zhang. Improved algorithms for fully dynamic maximal independent set. *CoRR*, abs/1804.08908, 2018.
- [5] Manoj Gupta and Shahbaz Khan. Simple dynamic algorithms for maximal independent set and other problems. *CoRR*, abs/1804.01823, 2018.