Lecture Note

November 24, 2021

We consider the maximum weight matching in the streaming setting. In streaming, edges in $E$ are given to us one by one. But we do not have enough space to store all the edges (otherwise, we can just apply the offline algorithm to solve it exactly). From time to time, we are forced to throw away edges. The entire challenge of the streaming consists thus in making good decisions on when and which to throw away elements.

Assuming that all weights are integers, we show an algorithm that uses $O(n \log W \epsilon)$ space, where $W = \max_{e \in E} w(e)$ and it guarantees that we output a matching that is a $1/2 - \epsilon$ approximation.

To make things simple, let us forget about the space issue for the moment. Let us associate with every vertex $v \in V$ with a variable $\phi_v$. Initially, $\phi_v = 0$. We also prepare a global stack $S$ to store the edges. Initially $S = \emptyset$. For every new arriving edge $e$, we will compute a gain $g(e)$ if we decide to store it.

\begin{algorithm}
\textbf{while} a new edge $e = (u,v)$ arrives \\
\hspace{1cm} If $w(e) > \phi_u + \phi_v$ then \\
\hspace{2cm} $g(e) = w(e) - \phi_u - \phi_v$ \\
\hspace{2cm} $\phi_u = \phi_u + g(e)$ \\
\hspace{2cm} $\phi_v = \phi_v + g(e)$ \\
\hspace{2cm} push $e$ onto stack $S$
\end{algorithm}

When we have seen all edges, we do a “reverse” greedy: pop edges from $S$ one by one and add them into the matching $M$, if possible.

This finishes the description of the algorithm. Below let us use the notation $u \succ e$ to denote the set of edges incident on $u$ and arrive before $e$ in the streaming. Let us write down the primal (matching) and dual (cover) programs.

\begin{align*}
\max & \sum_{e \in E} w_e x_e \\
\sum_{e \in E(v)} x_e & \leq 1 \quad \forall v \in V \\
x_e & \geq 0 \quad \forall e \in E
\end{align*}

\begin{align*}
\min & \sum_{v \in V} y_v \\
y_u + y_v & \geq w_e \quad \forall e = (u,v) \in E \\
y_v & \geq 0 \quad \forall v \in V
\end{align*}
Lemma 1 The set of variables \( \{ \phi_u \}_{u \in V} \) is a feasible dual solution.

Proof: We prove by arguing that all edges \( e = (u, v) \) is properly covered. Consider the moment that \( e \) arrives. If \( e \) is thrown away, then \( \phi_u + \phi_v \geq w(e) \). If \( e \) is stored in \( S \), \( \phi_u + \phi_v = g(e) + w(e) \geq w(e) \). As \( \phi_u \) and \( \phi_v \) are monotonically increasing, we are sure that in the end \( \phi_u + \phi_v \geq w(e) \).

Corollary 2 \( w(M_{opt}) \leq \sum_{u \in V} \phi_u \).

Proof: This follows from the previous lemma and the LP weak duality.

Lemma 3 \( \sum_{e \in S} g(e) = 2 \sum_{u \in V} \phi_u \).

We next argue that the reverse greedy captures all the gains.

Lemma 4 The matching \( M \) output by the algorithm has \( w(M) \geq \sum_{e \in S} g(e) \).

Proof: Observe that if an edge \( e = (u, v) \in S \), is stored,

\[
    w(e) = g(e) + \sum_{e' \in u \succ e} g(e') + \sum_{e' \in v \succ e} g(e').
\]

We will prove the lemma by “charging” the gain of every edge \( e \in S \) to some edge \( e \in M \). If \( e \in S \cap M \), then \( e \) is charged to itself. If \( e = (u, v) \in S \setminus M \), we use the property of greedy: when \( e \) is popped, the fact that it is not part of \( M \) means that \( u \) or \( v \) is already matched. Say it is \( u \) and it is matched by \( e' \). In this case, we charge \( e' \) to \( e \). Now by Equation (1), our charging scheme proves the lemma.

By Corollary 2 and Lemmas 3 and 4, we prove the theorem.

Theorem 5 \( M \) is a 0.5 approximation.

Saving Space

We now explain how to save space. We just did a little modification to the algorithm:

While a new edge \( e = (u, v) \) arrives

If \( w(e) > (1 + \epsilon)\phi_u + \phi_v \) then

\[ g(e) = w(e) - \phi_u - \phi_v, \]

\[ \phi_u = \phi_u + g(e), \]

\[ \phi_v = \phi_v + g(e), \]

push \( e \) onto stack \( S \)
Lemma 1 can be rephrased as: \((1 + \epsilon)\{\phi_u\}_{u \in V}\) is a feasible dual. Corollary 2 can be rephrased as \(w(M_{\text{opt}}) \leq (1 + \epsilon) \sum_{u \in V} \phi_u\). Lemmas 3 and 4 remained unchanged. Theorem 5 can be rephrased as \(M\) being a \(1/2 - \epsilon\) approximation. We just need to argue the space usage.

**Theorem 6** Let \(W = \max_{e \in E} w(e)\). Then the total space required is \(O\left(\frac{\log W}{\epsilon} n\right)\).

**Proof:** For a vertex \(v\), let \(W' = \max_{e \in \delta(v)} w(e)\). We claim that the number of edges in \(\delta(v)\) stored in \(S\) is at most \(O(\log_{1+\epsilon} W')\). If the claim holds, then as \(W' \leq W\) and there are only \(n\) vertices, the theorem follows.

The claim follows by the following two observations: (1) as (by assumption) all weights are integers, the first time that \(\phi_u\) grows beyond 0, \(\phi_u \geq 1\). (2) Each time a new edge in \(\delta(v)\) is stored, \(\phi_v\) grows by at least the factor of \(1 + \epsilon\).

\(\Box\)