Streaming Algorithms

Meng-Tsung Tsai

05/04/2018

Programming Assignment #1

Finding the articulation points in a given n-node m-edge graph G, using a single pass and O(n) space.

Recall that $B_1 \cup B_2$ is a 2-VC sparse certificate of G where B_1 is any spanning forest of G and B_2 is any spanning forest of $G \backslash B_1$.

It implies that a node v is an articulation point in $B_1 \cup B_2$ if and only if v is an articulation point in G.

Programming Assignment #1

Finding the articulation points in a given n-node m-edge graph G, using a single pass and O(n) space.

Recall that $B_1 \cup B_2$ is a 2-VC sparse certificate of G where B_1 is any spanning forest of G and B_2 is any spanning forest of $G \backslash B_1$.

Unfortunately, any p-pass algorithm that computes a BFS tree of an n-node graph requires $\Omega(n^2/p)$ space.

Programming Assignment #1

Finding the articulation points in a given n-node m-edge graph G, using a single pass and O(n) space.

Recall that $B_1 \cup B_2$ is a 2-VC sparse certificate of G where B_1 is any spanning forest of G and B_2 is any spanning forest of $G \backslash B_1$.

Indeed, two BFS spanning forests form a 2-VC strong certificate. That is, for any two graphs G and H. $B_1(H) \cup B_2(H) \cup G$ is a 2-VC sparse certificate of $H \cup G$.

It yields a single-pass, O(m+n)-time, O(n)-space streaming algorithm to compute articulation points.

References

- "Sparsification - A Technique for Speeding Up Dynamic Graph Algorithms," Eppstein et al. (1997)
- "Finding Graph Matchings in Data Streams," McGregor (2005)
- "Additive Combinatorics," Vu and Tao (2010)
- "Streaming Algorithms for Independent Sets," Halldorsson et al. (2010)

Matching for General Graphs

Problem Definition

Input: a sequence of edges $e_1, e_2, \ldots, e_m$ of an n-node m-edge undirected graph G.

Output: a matching M so that $|M| \geq OPT/(1+\varepsilon)$ for any given constant $\varepsilon > 0$.

Goal: using O(n polylog n) space and O(1) passes.

Idea

Step 1. Grab a maximal matching M. // using O(1) space, 1 pass

Step 2.

Case 1: $|M| \ll |M_{opt}|$.

There are many augmenting paths of short length, and augment a portion of them.

Case 2: $|M| \sim |M_{opt}|$

|M| is already a good approximation.

Threshold

Let $\alpha_i|M|$ denote the number of connected components in $M \oplus M_{opt}$ in which there are i edges from M and i+1 edges from M_{opt}.

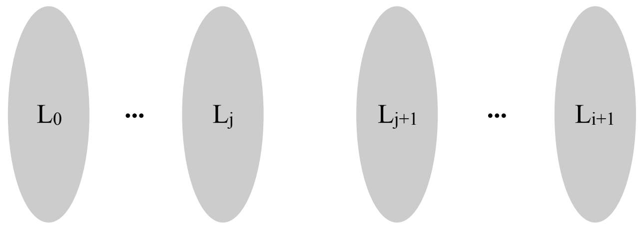
Claim. If there exists an integer $k \geq 1$ so that

$$\max_{1 \leq i \leq k} \alpha_i \leq 1/(2k^2(k+1)),$$

then $|M| \geq |M_{opt}|/(1+1/k)$.

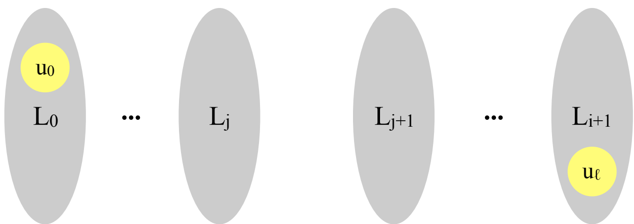
Thus, given an $\varepsilon > 0$, pick a sufficiently large k so that $1/k < \varepsilon$, and reduce the number of length-ℓ augmenting paths to within $|M|/(2k^2(k+1))$ for every $\ell \leq 2k+1$.

Random Projection



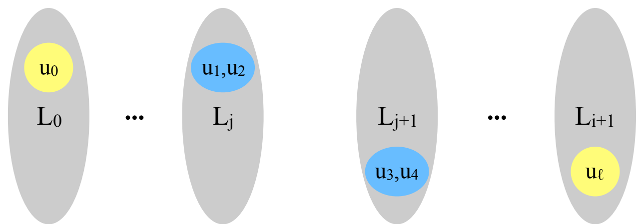
Randomly projecting matched edges and unmatched nodes onto a layered graph.

Random Projection



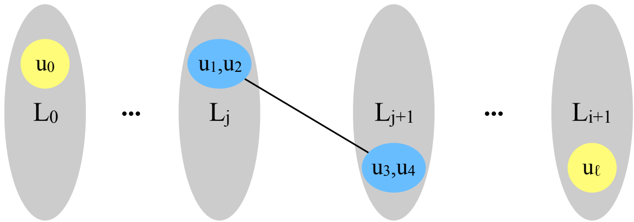
Place each unmatched node uniformly at random to L_0 or L_{i+1}.

Random Projection



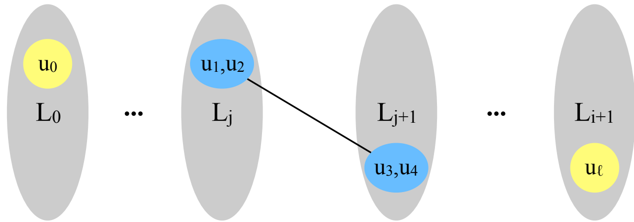
If $\{u, v\}$ is an edge in the matching, create a node s and map s uniformly at random into one of the following 2i choices:
a node s marked with $u \rightsquigarrow v$, placed in L_j for some j in [1, i] or
a node s marked with $v \rightsquigarrow u$, placed in L_j for some j in [1, i].

Random Projection



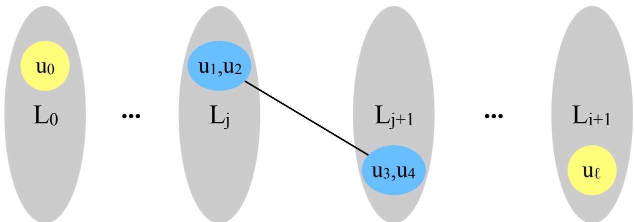
Connect each pair of nodes, say ($s_{u1,u2}$, $s_{u3,u4}$), in adjacent layers by an edge if there is an unmatched edge (u_2, u_3) in the input graph.

Random Projection



Each augmenting path in the input graph forms a path in the layered graph with probability $1/(2(2i)^i)$, a constant.

Random Projection

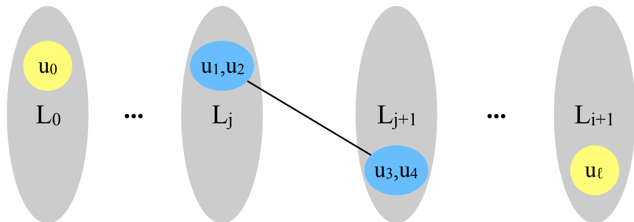


Each of the $\alpha_i|M|$ augmenting paths of length $2i+1$ is included in the layered graph independently with a constant probability.

$\Rightarrow$

w.h.p. a constant fraction of such paths can be found in the layered graph.

Random Projection



Because there are constant layers, BFS can be done in O(1) passes. By a backtracking BFS, a constant fraction of augmenting paths can be found.

Independent Sets

Problem Definition

Input: a sequence of edges $e_1, e_2, \ldots, e_m$ of an n-node m-edge undirected graph G.

Output: an independent set S so that $|S| \geq n^2/(n+2m)$.

Goal: using O(n polylog n) space and a single passes.

When $m = O(n)$, the output is a O(1)-apx of the maximum independent set.

Turan's Theorem

Any n-node m-edge graph has an independent set of size $n^2/(n+2m)$.

Here is a constructive proof.

Step 1. Assign a random ordering to the n nodes.

Step 2. Pick those nodes whose assigned number is the minimum among its neighbor nodes.

Clearly, the selected nodes form an independent set.

Turan's Theorem

Any n-node m-edge graph has an independent set of size $n^2/(n+2m)$.

Here is a constructive proof.

Step 1. Assign a random ordering to the n nodes.

Step 2. Pick those nodes whose assigned number is the minimum among its neighbor nodes.

The expected number of selected nodes is $\sum_{x \in V} 1/(1+\deg(x)) \leq \sum_{x \in V} 1/(1+2m/n)$.

Turan's Theorem

Any n-node m-edge graph has an independent set of size $n^2/(n+2m)$.

Here is a constructive proof.

Step 1. Assign a random ordering to the n nodes.

Step 2. Pick those nodes whose assigned number is the minimum among its neighbor nodes.

Thus, there **must** exist an independent set of size $n^2/(n+2m)$.

Exercise 1

Give a single pass semi-streaming algorithm to compute an independent set whose expected size is $n^2/(n+2m)$.

Exercise 2

Think about the relationship between independent sets and matchings.