COMS E6998-9: Algorithms for Massive Data (Spring'19) Mar 26, 2019

Lecture 17: Sublinear Time Algorithms

Instructor: *Alex Andoni* Scribes: *Mark Dijkstra*

1 Introduction

We begin this lecture by continuing the discussion of Minimum Spanning Trees (MST) and proving the validity of the algorithm introduced in the previous lecture. Afterwards we discussed a new problem of approximating the Vertex Cover problem in sublinear time by introducing Maximal Matchings and Maximal Independent Sets. We left off by discussing the creation of an oracle that will tell us whether a vertex, $v \in G'$, belongs to a specific instance of a Maximal Independent Set of the graph. The next lecture will discuss how one could design this local oracle.

2 MST

2.1 Problem Definition

We are given an MST in graph G, and degree d, $u_{i,j} \in \{1, 2, \ldots, M\}$

Theorem: We can estimate MST up to $1+\epsilon$ factor in $\left(\frac{dM}{\epsilon}\right)^{O(1)}$

1. $MST \geq n - 1$ (to connect all vertices, need n - 1 edges)
2. $MST = n - 1 + \sum_{i=1}^{n} CC_i - 1$
3. $CC_i = \#$ connected components (denoted CC) in graph G_i with edges such that $u_{edge} \leq i$, that is edges where cost are at most i

Problem: Computing number of connected components in G_i

$\alpha_v = \frac{1}{\text{size of CC of V}}$, note that $\alpha_v \in (0, 1]$

Fact: $\sum_v \alpha_v = \#CC$

Algorithm:

- sample $k = O(\frac{1}{\delta^2})$ vertices v
- explore the CC of v using (dfs or bfs), until see $> \frac{1}{\delta}$ vertices
  - Set $d_v = max\left\{\frac{1}{\text{size of CC of v}}, \delta\right\}$
- output $\hat{c} = \frac{n}{k} \sum_{i=1}^{k} \hat{\alpha}_{v_i}$

2.2 MST Estimator Claims

Claim 1. $|\mathbb{E}[\hat{C} - C]| \leq \delta n$

Proof.

$$\mathbb{E}[\hat{C}] = \frac{n}{k} \cdot \sum_{i=1}^{k} \mathbb{E}_{v_i} \hat{\alpha_{v_i}} \geq \frac{n}{k} \sum_{i=1}^{k} \mathbb{E}_{v_i}[\alpha_{v_i}] = \sum_{v} \alpha_v = C \leq \frac{n}{k} \sum i = 1^k \mathbb{E}_{v_i}[\delta_{v_i} + \delta] = C + \frac{n}{k} \sum_{i=1}^{k} \delta = C + \delta n$$

□

Claim 2. $var[\hat{C}] \leq \frac{n}{k} \cdot (c + \delta^2 n) = o\big(\delta^2 n\big)\big)$

Proof.

$$[Var[\hat{C}] = \frac{n^2}{k^2} \sum_{i=1}^{k} Var_{v_i}[\hat{\alpha_{v_i}}] \leq \frac{n^2}{k} \mathbb{E}[\hat{\alpha}_v^2] \leq \frac{n^2}{k} (\mathbb{E}_v[\alpha_v^2] + \delta^2) \leq \frac{n}{k} [n \cdot \mathbb{E}_v[\alpha_v] + \delta^2 n] = \frac{n}{k} [c + \delta^2 n]$$

□

Claim 3. *Number of queries is* $O\big(\frac{d}{\delta^3}\big)$

Proof. # queries refers to how many times we look at the graph; so
queries $= O(k) \cdot \frac{1}{\delta} \cdot d = O\Big(\frac{d}{\delta^3}\Big)$, note we multiply by d to check all of the neighbors of a vertex. □

Thus, we can conclude that the algorithm for MST in which we:
- Estimate $\hat{CC}_i$ for $d = \frac{\epsilon}{M}$
- $\hat{MST} = n - 1 + \sum_{i=1}^{M} (\hat{CC}_i - 1)$

Claim 4. $\hat{MST}$ *is a* $(1 + \epsilon)$*-factor approx to MST cost w/ 90% probability*

Proof.

$$|\mathbb{E}[\hat{MST} - MST]| \leq \sum_{i=1}^{M} |\hat{CC}_i - CC_i| \leq k \cdot \delta_n \leq \epsilon n \leq 2 \cdot \epsilon \cdot MST$$

$$Var[\hat{MST}] = \sum_{i=1}^{M} Var[\hat{CC}_i] \leq \sum_{i=1}^{M} O(\delta^2 n) \cdot (CC_i + \delta^2 n)$$

$$\leq O(\delta^2 n) \cdot (MST + \delta^2 Mn) = O(\delta^4 Mn + \delta^2 n MST)$$

by Chebyshev with probability $\geq 90\%$ we have that $\hat{MST} \in \mathbb{E}[\hat{MST}] \pm \sqrt{10 \cdot var}$. Furthermore we have that

$$\sqrt{var} \leq O\Big(\frac{\epsilon^4}{M^4} Mn^2 + \frac{\epsilon^2}{M^2} n \cdot MST\Big)^{\frac{1}{2}} \leq O(\epsilon^2 \cdot MST^2)^{\frac{1}{2}} = O(\epsilon \cdot MST)$$

Thus we have that $\hat{MST} \in MST \pm O(\epsilon \cdot MST) = MST(1 \pm O(\epsilon))$.
This gives us time: $M \cdot O(d/\delta^3) = O(M \cdot d \cdot \frac{M^3}{\epsilon^3}) = O(\frac{dM^4}{\epsilon^3})$ □

Lower bound: linear dependence on M is necessary

3 Sublinear Time Estimator for Vertex Cover

3.1 Problem Formulation

Note that this will be a local algorithm

Definition: Vertex cover of a graph is a set of vertices, V, such that each edge of the graph is incident to at least one vertex of V.
Goal: We want to estimate a minimum vertex cover

3.2 Background

Theorem 5 (Gavril-Yannakakis)**.** *Can compute factor 2-approximation in poly-time. Output V s.t.*

1. *V is a vertex cover (VC)*
2. *Minimum Vertex Cover, $MVC \leq |V| \leq 2 \cdot MVC$*

Proof. Let M = maximal matching in graph G
A *matching* is a set of edges of a graph such that no edges share a common vertex; a *maximal matching* is a *matching* such that you cannot add any edges to the set. Note that a *maximal matching* is not necessarily *maximum* in so far that, the number of edges in a maximal matching is strictly $\leq$ number of edges in a maximum matching. Furthermore, there can be many maximal matchings for a given graph.

1. We will denote a maximal matching as M
2. V = number of vertices in M
3. $|V| = 2 \cdot |M|$

1) For $\forall$ edges, one endpoint is in V otherwise M is not maximal $\implies$ V is a vertex cover
2) for each edge $\in$ M at least one endpoint is in the MVX $\implies |M| \leq |MVC| \implies |V| \leq 2|MVC|$

□

Theorem: Can compute some $\hat{VC}$ s.t. $MVC - \epsilon n \leq \hat{VC} \leq 2 \cdot MVC + \epsilon n$ (2, ϵn)-approximation in time $O(d \cdot 2^d)$, where d = max degree (can be replaced with $d = \frac{m}{n}$) [Nguyen-Orak '08]

3.2.1 Idea:

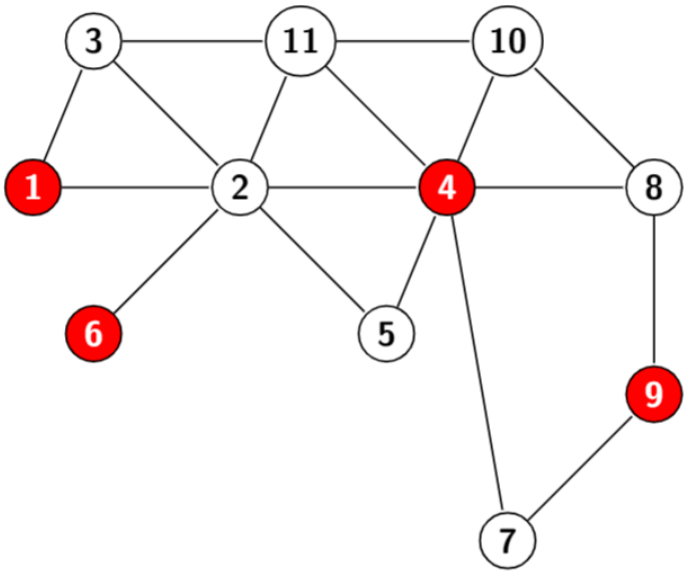
Estimate maximal matching size in G by sampling

This is logically equivalent to finding a maximal independent set in G', where G' = graph vertices = edges in G. So, an edge $(e, e') \to e \,\&\, e'$ share a vertex in G.

3.2.2 Algorithm for MIS

Here we have an algorithm to find a single Maximal Independent Set. It is very contingent on the order in which vertices of a graph are visited, and it further exemplifies that there are multiple Maximal Independent Sets for any given graph (i.e. many MIS and not unique).

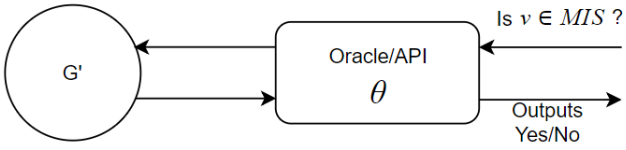
- Set $I = \emptyset$
- Consider $v \in V$ of some G in order:
  add v to $I \iff \forall u \in N(v)$, s.t. $u \notin I$



Above we see the algorithm ran on the graph to produce a MIS of size 4, with $I = \{1, 4, 6, 9\}$

4 Local Oracle

We will now begin discussion for the local oracle for MIS and will continue in the subsequent lecture.



A logical question entails: which MIS are we concerned with? The oracle works with respect to a single MIS. Furthermore, it is noteworthy that $\forall v \in G$ that $\exists$ a MIS s.t. $v \in MIS$ of G.

Guarantee on Oracle: $\exists$ a set I which is

1. MIS
2. $I = I(G'$, randomness of O). Not dependent on questions investigated, so that Yes/No answers are consistent with I, namely, whether our vertex belongs to our I.

Next lecture we design the local oracle.