CSCI 1951-W Sublinear Algorithms for Big Data	Fall 2020
Lecture 6: More Bounded Degree Graph and Dense Graph Intro	
Lecturer: Jasper Lee	*Scribe: Champ Chairattana-apirom*

1 Overview

In the previous lecture, we introduced algorithms on bounded degree graphs. These graphs mainly depends on two techniques:

(1) Random sampling on the vertices or a vertex's neighbors

(2) Breadth First Search (BFS)

In today's lecture, we are going to talk about more algorithms on bounded degree graphs and an introduction to dense graphs and its model.

2 Cycle-freeness testing in Bounded Degree Graph

Cycle-freeness of a graph is equivalent to it being a forest of trees. The following is a well-known fact about forests.

Fact 6.1 *For a graph G with n vertices and k connected components, G is a cycle-free graph if and only if G has $m = n - k$ edges.*

From Fact 6.1, it implies that "If G is not cycle-free, then G has $m > n-k$ edges." Using this statement, we come up with an ϵ-far cycle-freeness testing algorithm, which estimates # of edges (m) and # of connected components (k).

Algorithm 6.2: ϵ-far from cycle-free graph testing via estimation of m and k

Input : Bounded degree graph G with n vertices and degree upper-bound of d
Output: Determining whether G is cycle-free

1. $\tilde{m} \leftarrow$ Estimate m to within additive error of $\frac{\epsilon dn}{100}$ using $O\left(\frac{1}{\epsilon^2}\right)$ queries.
2. $\tilde{k} \leftarrow$ Estimate k to within additive error of $\frac{\epsilon dn}{100}$ using $O\left(\frac{1}{d^2\epsilon^3}\right)$ queries.
3. Accept(Determine that G is cycle-free) if and only if $\tilde{m} + \tilde{k} \leq n + \frac{\epsilon dn}{50}$

Note: To estimate m, you estimate the number of non-empty cells in the $n \times d$ matrix (representation of the graph). The query complexity $O(\frac{1}{\epsilon^2})$ can be derived using Hoeffding's Inequality with additive error ϵdn.

Note: Algorithm 6.2 gives $O\left(\frac{1}{d^2\epsilon^3}\right)$ time and query complexity.

To analyze Algorithm 6.2, We then consider the definition of ϵ-farness from cycle-freeness.

Proposition 6.3 *If a graph G with n vertices has k connected components and is ϵ-far from being cycle-free, then $m \geq n - k + \frac{\epsilon dn}{2}$.*

Proof. Notion of ϵ-far from cycle-free meaning it needs to change at least $\frac{\epsilon dn}{2}$ edges to reach cycle-freeness. From Fact 6.1, it follows that $m \geq n - k + \frac{\epsilon dn}{2}$. □

The structural property(Completeness and Soundness) of this algorithm then follows from Fact 6.1 and Proposition 6.3.

Note: This algorithm is a 2-sided error algorithm. For 1-sided error, there is a lower bound of $\Omega(\sqrt{n})$

3 Subgraph-freeness testing in Bounded Degree Graph

Definition 6.4 (H-freeness) Let H be a fixed graph. A graph G is H-free if no subgraph of G is isomorphic to H.

Definition 6.5 (Radius and Center of H) Let the radius of a graph H (denoted by $rd(H)$) be the minimum r such that there exists some $v \in H$ such that for any $u \in H, d(u,v) \leq r$(d denoting the distance from u to v). Here, we call such v as a center of H. (Note: v might not be unique.)

Then, with the notion of center and radius, we construct the following algorithm.

Algorithm 6.6: ϵ-far from H-free testing

Input : Bounded degree graph G with n vertices and degree upper-bound of d, and a fixed graph H

Output: Determining whether G is H-free

Repeat for $O\left(\frac{1}{\epsilon}\right)$ times:

1. Pick a random vertex u. (Pretending that u is the center)
2. Run BFS for radius $rd(H)$ from u.
   (**Note:** The size of the subgraph from this BFS is bounded by $d^{O(rd(H))}$)
3. Check if the subgraph from the BFS is H-free

It follows that Algorithm 6.6 has query and time complexity of $O\left(\frac{1}{\epsilon} d^{O(rd(H))}\right)$. Next, we show the correctness of the algorithm.

The completeness of the algorithm is trivial.

Proposition 6.7 (Completeness of Algorithm 6.6) *Algorithm 6.6 accepts all H-free graphs.*

To prove soundness, we use the following definition of the notion of a detecting vertex.

Definition 6.8 (Detecting vertex) Vertex v is *detecting* if it is a center of a copy of H in G. Meaning, if we pick *detecting* vertex v, then we will find always H as a subgraph of the BFS subgraph.

Proposition 6.9 *If G is ϵ-far from H-free, then there are at least $\frac{\epsilon n}{2}$ detecting vertices.*

Proof. Suppose we have fewer than $\frac{\epsilon n}{2}$ detecting vertices, we can remove all edges incident to those vertices with fewer than ϵdn changes in the graph representation, which contradicts the notion of ϵ-farness □

Proposition 6.10 (Soundness of Algorithm 6.6) *Algorithm 6.6 rejects an ϵ-far from H-free with constant probability $\frac{2}{3}$.*

Proof. It follows from Proposition 6.9 that the probability of sampling a vertex to be a detecting vertex is at least $\frac{\epsilon}{2} = O(\epsilon)$. Then, sampling $O\left(\frac{1}{\epsilon}\right)$ random vertices,

$$\mathbb{P}[\text{Algorithm 6.6 failing}] \leq (1 - O(\epsilon))^{O\left(\frac{1}{\epsilon}\right)} \leq e^{-O\left(\frac{1}{\epsilon}\right)O(\epsilon)} = e^{-O(1)} \leq \frac{1}{3}$$

□

4 Bipartiteness testing in Bounded Degree Graph

Definition (Bipartite) A graph G is bipartite if vertices G can be partitioned into two disjoint sets U, V such that $U \cup V = G$ and E_G(the set of edges of G) is a subset of $U \times V$

Fact 6.11 *A graph G is bipartite if and only if G has no odd-length cycles.*

In this case, if we construct an algorithm using BFS it might not work since graphs can have very long cycles where BFS would have to explore $O(d^{\text{cycle length}})$ vertices which grows exponentially.

Idea Instead of exploring radius r neighborhood, take random walks of length $O(\text{polylog}(n))$.

Definition 6.12 (Random Walk) A random walk of length l starting at u is a path $(u, v_1, v_2, ..., v_l)$ such that $v_{i+1} \leftarrow \text{Unif}(\mathcal{N}(v_i))$.

Algorithm 6.13: ϵ-far from bipartite testing

Input : Bounded degree graph G with n vertices and degree upper-bound of d
Output: Determining whether G is bipartite
Repeat for $O\left(\frac{1}{\epsilon}\right)$ times:

1. Uniformly pick a random vertex u.
2. Try to find odd length cycle through u.
   (i) Perform $m = \sqrt{n}\,\text{poly}\left(\frac{\log n}{\epsilon}\right)$ random walks of length $l = \text{poly}\left(\frac{\log n}{\epsilon}\right)$.
   (ii) Record the explored vertices into two sets R_0, R_1 being the set of vertices reachable from u in even, odd number of steps respectively.
   (iii) Reject if $R_0 \cap R_1 \neq \emptyset$

Algorithm 6.13 has query and time complexity of $O\left(\sqrt{n}\,\text{poly}\left(\frac{\log n}{\epsilon}\right)\log d\right)$.

Remark $\log d$ term is from binary-searching to find the degree of a vertex v.
Why?: Because we don't know the degree but only knows that it is bounded by d. Also, we need the degree to draw the neighbors uniformly.

Proposition 6.14 (Completeness of Algorithm 6.13) *Algorithm 6.13 accepts all bipartite graphs.*

The completeness proposition follows from the definition of bipartite graph.

Theorem 6.15 (Soundness of Algorithm 6.13 (Hard)) *Algorithm 6.13 rejects a graph that is ϵ-far from bipartiteness with probability $\geq \frac{1}{3}$.*

Note: The proof of Theorem 6.15 can be found in this at `http://www.eng.tau.ac.il/~danar/Public-pdf/bip.pdf`).

5 Dense Graph

5.1 Model

Last lecture, we mentioned about models/representation of the graphs and we said that with dense graphs it is better to use adjacency matrix. So, we define G as an $n \times n$ matrix with entry $(u, v) = 1$ if and only if (u, v) is an edge in G.(**Note:** Here we don't have weights on edges.) Thus, we define the query

$$\text{query}(u, v) = G_{u,v}$$

Then, the input size of the graph is n^2. Accordingly, we define the distance notion by

$$d(G, G') = \frac{1}{n^2}(\# \text{ of distinct bits in } G \text{ and } G')$$

Thus, ϵ-farness means $O(\epsilon n^2)$ changes on edges.
Note: This large gap of $O(\epsilon n^2)$ makes the model not always applicable on some problems.

5.2 Hamiltonian Cycles

Hamiltonian cycle detection problem is one of the examples that this model cannot give a good algorithm or bounds on.

Definition 6.16 (Hamiltonian Cycle) A hamiltonian cycle on graph G is a cycle that contains all the vertices in G exactly once.

In general case(without notion of ϵ-farness), detecting if a graph has any hamiltonian cycle is an NP-Hard problem. However, for ϵ-far property testing, the algorithm just accepts anything. This works because of the following proposition.

Proposition 6.17 *No graph is ϵ-far from having a Hamiltonian cycle for any $\epsilon = \omega\left(\frac{1}{n}\right)$.*

Proof. Suppose there is such a graph that is ϵ-far from Hamiltonian cycle graph with $\epsilon = \omega\left(\frac{1}{n}\right)$ However, you can create a Hamiltonian cycle with at most $2n$ changes, resulting in contradiction. □

There are also other example problems that are trivialized by this model:

(a) Connectedness testing

(b) Test whether G contains some sparse subgraph H.

(c) Testing other sparse properties.

6 Biclique testing in Dense Graph

Definition 6.18 (Biclique/Complete Bipartite Graph) A graph $G = (V, E)$ is a biclique if there exists a bipartition (V_1, V_2) of V such that $E \cong V_1 \times V_2$.

Proposition 6.19 *Suppose a graph G is ϵ-far from being a biclique. Then, for any bipartition (V_1, V_2) of V, there exists at least $\epsilon n^2/2$ pairs of vertices (u, v) that violates (V_1, V_2) i.e.*

1. *If u, v are in the same set, then (u, v) is an edge in G.*
2. *If u, v are not in the same set, then (u, v) is not an edge in G.*

Proof. Suppose for contradiction that there exists some bipartition (V_1, V_2) with less than $\epsilon n^2/2$ violating pairs. Therefore, we can flip the bits of these pairs which will take less than ϵn^2(edges are bidirectional) to change G to a biclique. Thus, G is less than ϵ-far from biclique, contradiction. □

Using this proposition, we construct an algorithm.

Algorithm 6.20: ϵ-far from biclique testing

Input : Dense graph G as adjacency matrix
Output: Determining whether G is biclique
Repeat for $O\left(\frac{1}{\epsilon}\right)$ times:

1. Pick a random vertex u.
2. Pick a random pair v, w.
3. Check if $\{u, v, w\}$ is a biclique. Reject if not.

Note: $\{u, v, w\}$ is a biclique means that there are two possible cases for them:

1. They form a line graph of length 2. (At least one of them not in the same partition as u)
2. They form a graph without edges. (All are in the same partition as u)

Proposition 6.21 (Completeness of Algorithm 6.20) *Algorithm 6.13 accepts all bicliques.*

Proposition 6.22 (Soundness of Algorithm 6.20) *A single iteration of Algorithm 6.20 rejects a graph G which is ϵ-far from biclique with probability $\geq O(\epsilon)$.*

Proof. After the algorithm fixes u, we can consider the bipartition

$$(\mathcal{N}(u), V_G \setminus \mathcal{N}(u) \cup \{u\})$$

From Proposition 6.19, sampling a violating pair have

$$\mathbb{P}[\text{Algorithm 6.20 rejects}] = \mathbb{P}[(v, w) \text{ is a violating pair}] \geq \frac{\epsilon n^2}{2} \cdot \frac{1}{n^2} = \frac{\epsilon}{2} = O(\epsilon)$$

□

From Proposition 6.22, with $O\left(\frac{1}{\epsilon}\right)$ iterations, the success probability will be a constant $\left(\frac{2}{3}\right)$.

Note: Here we use a proving technique that fixing/forcing the structure of the problem, and then, we check for the violation, which we will also see in the next lecture.