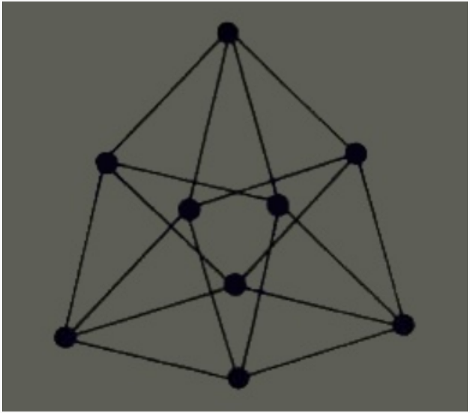
FAST AVERAGE CASE CONNECTIVITY ALGORITHMS ON RANDOM GRAPHS

Saurav Shekhar
Supervisor: Prof. Surender Baswana

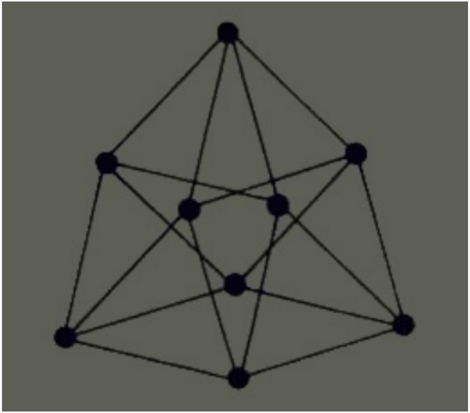
November 13, 2015

IIT Kanpur

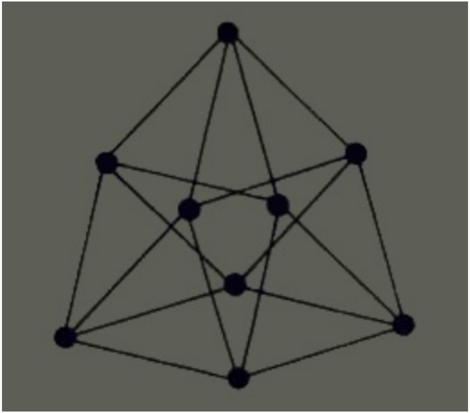
PRELIMINARIES



- Probability distribution over Graphs



- Probability distribution over Graphs
- Two major models, $G(n, m)$ and $G(n, p)$



- Probability distribution over Graphs
- Two major models, $G(n, m)$ and $G(n, p)$
- Entities(vertices and edges) independent of each other

Derived by Erdös and Rényi [2]. As edge probability p increases

- If $p < \frac{1}{n}$, all connected components of the graph will be of size $O(\log n)$ **whp**

Derived by Erdös and Rényi [2]. As edge probability p increases

- If $p < \frac{1}{n}$, all connected components of the graph will be of size $O(\log n)$ **whp**
- If $p > \frac{1}{n}$, the graph will have a giant component of size $O(n)$ **whp**

Derived by Erdös and Rényi [2]. As edge probability p increases

- If $p < \frac{1}{n}$, all connected components of the graph will be of size $O(\log n)$ **whp**
- If $p > \frac{1}{n}$, the graph will have a giant component of size $O(n)$ **whp**
- If $p < \frac{(1-\epsilon)\ln n}{n}$, the graph will be disconnected **whp**

Derived by Erdös and Rényi [2]. As edge probability p increases

- If $p < \frac{1}{n}$, all connected components of the graph will be of size $O(\log n)$ **whp**
- If $p > \frac{1}{n}$, the graph will have a giant component of size $O(n)$ **whp**
- If $p < \frac{(1-\epsilon)\ln n}{n}$, the graph will be disconnected **whp**
- If $p > \frac{(1+\epsilon)\ln n}{n}$, the graph will be connected **whp**

- Proofs using counting / Branching models

- Proofs using counting / Branching models
- Krivelevich and Sudakov [3] gave a very simple proof of first two results using DFS.

- Proofs using counting / Branching models
- Krivelevich and Sudakov [3] gave a very simple proof of first two results using DFS.

DFS: 3 sets of vertices, set S (visited vertices), T (unvisited vertices) $U = V \setminus (S \cup T)$ has 'visiting' vertices. Vertices in U form a stack.

- Proofs using counting / Branching models
- Krivelevich and Sudakov [3] gave a very simple proof of first two results using DFS.

DFS: 3 sets of vertices, set S (visited vertices), T (unvisited vertices) $U = V \setminus (S \cup T)$ has 'visiting' vertices. Vertices in U form a stack. In each round

- If U is empty, choose the first vertex from T and push it into U

- Proofs using counting / Branching models
- Krivelevich and Sudakov [3] gave a very simple proof of first two results using DFS.

DFS: 3 sets of vertices, set S (visited vertices), T (unvisited vertices) $U = V \setminus (S \cup T)$ has 'visiting' vertices. Vertices in U form a stack. In each round

- If U is empty, choose the first vertex from T and push it into U
- Else, the algorithm picks the top vertex u from the stack and scans T. On finding the first neighbor v of u in T, algorithm removes u from T and pushes it on top of U

- Proofs using counting / Branching models
- Krivelevich and Sudakov [3] gave a very simple proof of first two results using DFS.

DFS: 3 sets of vertices, set S (visited vertices), T (unvisited vertices) $U = V \setminus (S \cup T)$ has 'visiting' vertices. Vertices in U form a stack. In each round

- If U is empty, choose the first vertex from T and push it into U
- Else, the algorithm picks the top vertex u from the stack and scans T. On finding the first neighbor v of u in T, algorithm removes u from T and pushes it on top of U
- If u does not have a neighbor in T, pop u from U and insert it into S
- Query all remaining edges after $U \cup T = \phi$

- As long as U does not go empty, all vertices pushed into U belong to same connected component

- As long as U does not go empty, all vertices pushed into U belong to same connected component
- At each round either a vertex moves either from T to U, or from U to S

- As long as U does not go empty, all vertices pushed into U belong to same connected component
- At each round either a vertex moves either from T to U, or from U to S
- At any stage of the algorithm, there are no edges between current S and current T

- As long as U does not go empty, all vertices pushed into U belong to same connected component
- At each round either a vertex moves either from T to U, or from U to S
- At any stage of the algorithm, there are no edges between current S and current T

We can map each query to a Bernoulli random variable with parameter p.

- As long as U does not go empty, all vertices pushed into U belong to same connected component
- At each round either a vertex moves either from T to U, or from U to S
- At any stage of the algorithm, there are no edges between current S and current T

We can map each query to a Bernoulli random variable with parameter p. Then the graph obtained is in $G(n, p)$.

PHASE TRANSITIONS

- As long as U does not go empty, all vertices pushed into U belong to same connected component
- At each round either a vertex moves either from T to U, or from U to S
- At any stage of the algorithm, there are no edges between current S and current T

We can map each query to a Bernoulli random variable with parameter p. Then the graph obtained is in $G(n, p)$.
We can study the component structure using the properties of $\bar{X} = (X_i)_{i=1}^{N}$

Important Insights

- For small p, we will not get enough positive query results ($> k$ where $k = O(\log n)$) in a small interval kn so all components will be small in size ($O(\log n)$)

Important Insights

- For small p, we will not get enough positive query results ($> k$ where $k = O(\log n)$) in a small interval kn so all components will be small in size ($O(\log n)$)
- For p above threshold, after some N_0 time there will exist an interval of atleast a given length such that set U will not become empty during that interval

Important Insights

- For small p, we will not get enough positive query results ($> k$ where $k = O(\log n)$) in a small interval kn so all components will be small in size ($O(\log n)$)
- For p above threshold, after some N_0 time there will exist an interval of atleast a given length such that set U will not become empty during that interval
  Why not empty ?

Important Insights

- For small p, we will not get enough positive query results ($> k$ where $k = O(\log n)$) in a small interval kn so all components will be small in size ($O(\log n)$)
- For p above threshold, after some N_0 time there will exist an interval of atleast a given length such that set U will not become empty during that interval
  Why not empty ?
- $|S||T|$ will become much higher than expected no. of non-edges

Important Insights

- For small p, we will not get enough positive query results ($> k$ where $k = O(\log n)$) in a small interval kn so all components will be small in size ($O(\log n)$)
- For p above threshold, after some N_0 time there will exist an interval of atleast a given length such that set U will not become empty during that interval
  Why not empty ?
- $|S||T|$ will become much higher than expected no. of non-edges
  Giant component is formed in this continuous interval.

ALGORITHM OUTLINE

ALGORITHM OUTLINE

- Input graph with n vertices and m edges, in $G(n, m)$. Randomly permuted adjacency lists

- Input graph with n vertices and m edges, in $G(n, m)$. Randomly permuted adjacency lists
- For finding the connected component structure, it suffices to exhaust all edges in all except one component and explore enough edges in the remaining component so that we know it is connected

- Input graph with n vertices and m edges, in $G(n, m)$. Randomly permuted adjacency lists
- For finding the connected component structure, it suffices to exhaust all edges in all except one component and explore enough edges in the remaining component so that we know it is connected
- Remaining component large enough, will have to look at fewer edges

ALGORITHM OUTLINE

- Input graph with n vertices and m edges, in $G(n, m)$. Randomly permuted adjacency lists
- For finding the connected component structure, it suffices to exhaust all edges in all except one component and explore enough edges in the remaining component so that we know it is connected
- Remaining component large enough, will have to look at fewer edges
  Giant Component ?

KARP'S CONNECTED COMPONENTS ALGORITHM

- Two Stages, first sampling (find giant component)

- Two Stages, first sampling (find giant component)
- Second, elements not in the giant component grouped into components

Algorithm 1 Connected Components

```
procedure STAGE I(G)
    Let E_0 = ϕ
    while E_0 ≠ E do
        Let D = ϕ
        if |E − E_0| < n then
            D = E − E_0
        else
            D ← n distinct randomly selected edges from E − E_0
        end if
        Use DFS to find connected components of G_0 = (V, E_0)
        if There exists a connected component in G_0 of size > θn then
            STAGE II
            return
        end if
    end while
    return
end procedure
```

Algorithm 2 Connected Components

```
procedure STAGE II(G)
    Mark vertices of 'Giant component' as 'giant' and remaining as unmarked
    for v ∈ σ do
        if v is unmarked then
            Start DFS from vertex v
            if An edge leading to a giant vertex is found then
                Abort dfs and mark all vertices explored as 'giant'
            else
                Mark all vertices visited as a new component
            end if
        end if
    end for
end procedure
```

AVERAGE CASE RUNTIME ANALYSIS

Theorem 9b in [2]

Let $\varrho_{n,N}$ denote the size of the greatest component of $\Gamma_{n,N}$. If $N(n) \sim cn$ where $c > \frac{1}{2}$ we have for any $\eta > 0$

$$\lim_{n \to +\infty} \mathcal{P}\left(\left| \frac{\varrho_{n,N(n)}}{n} - G(c) \right| < \eta \right) = 1$$

where $G(c) = 1 - \frac{x(c)}{2c}$ and $x(c) = \sum_{k=1}^{\infty} \frac{k^{k-1}}{k!} (2ce^{-2c})^k$ is the solution satisfying satisfying $0 < x(c) < 1$ of the equation $x(c)e^{-x(c)} = 2ce^{-2c}$.

Theorem 9b in [2]

Let $\varrho_{n,N}$ denote the size of the greatest component of $\Gamma_{n,N}$. If $N(n) \sim cn$ where $c > \frac{1}{2}$ we have for any $\eta > 0$

$$\lim_{n \to +\infty} \mathcal{P}\left(\left| \frac{\varrho_{n,N(n)}}{n} - G(c) \right| < \eta \right) = 1$$

where $G(c) = 1 - \frac{x(c)}{2c}$ and $x(c) = \sum_{k=1}^{\infty} \frac{k^{k-1}}{k!}(2ce^{-2c})^k$ is the solution satisfying satisfying $0 < x(c) < 1$ of the equation $x(c)e^{-x(c)} = 2ce^{-2c}$.

For a random graph G with n vertices and n edges, there is a constant $\theta > 1/2$ such that **whp** there is a component in G with atleast θn vertices

- p_{in} be the probability that giant component exists in a random graph with n vertices and in edges. $p_{in} > p_{(i-1)n}$

- p_{in} be the probability that giant component exists in a random graph with n vertices and in edges. $p_{in} > p_{(i-1)n}$
- Probability that STAGE I takes i rounds is

- p_{in} be the probability that giant component exists in a random graph with n vertices and in edges. $p_{in} > p_{(i-1)n}$
- Probability that STAGE I takes i rounds is

$$
\begin{aligned}
\mathcal{P}(i) &= (1-p_n)(1-p_{2n})(1-p_{3n})\dots(1-p_{(i-1)n})p_{in} \\
&< (1-p_n)(1-p_n)(1-p_n)\dots(1-p_n)1 \\
&= (1-p_n)^{i-1}
\end{aligned}
$$

- p_{in} be the probability that giant component exists in a random graph with n vertices and in edges. $p_{in} > p_{(i-1)n}$
- Probability that STAGE I takes i rounds is

$$\begin{aligned}\mathcal{P}(i) &= (1 - p_n)(1 - p_{2n})(1 - p_{3n}) \ldots (1 - p_{(i-1)n})p_{in} \\ &< (1 - p_n)(1 - p_n)(1 - p_n) \ldots (1 - p_n)1 \\ &= (1 - p_n)^{i-1}\end{aligned}$$

  - In the j^{th} step, number of edges are nj. If we do a naive brute force each time, the total time taken is $O(\sum_{j=1}^{i} nj) = O(ni^2)$

STAGE I ANALYSIS

- p_{in} be the probability that giant component exists in a random graph with n vertices and in edges. $p_{in} > p_{(i-1)n}$
- Probability that STAGE I takes i rounds is

$$
\begin{aligned}
\mathcal{P}(i) &= (1 - p_n)(1 - p_{2n})(1 - p_{3n}) \dots (1 - p_{(i-1)n})p_{in} \\
&< (1 - p_n)(1 - p_n)(1 - p_n) \dots (1 - p_n)1 \\
&= (1 - p_n)^{i-1}
\end{aligned}
$$

  - In the j^{th} step, number of edges are nj. If we do a naive brute force each time, the total time taken is $O(\sum_{j=1}^{i} nj) = O(ni^2)$
  - Improve it by ‘remembering‘ the connected component data found in previous round

Anyway, expected runtime is

Anyway, expected runtime is

$$O\Big(\sum_{i=1}^{\infty} ni^2(1-p_n)^{i-1}\Big) = O\Big(\frac{n}{p_n^3}\Big) = O(n) \text{ as } n \to \infty$$

Stage II

- Typical step consists of examining some entry w in $\mathsf{adj}(v)$

Stage II

- Typical step consists of examining some entry w in $\mathsf{adj}(v)$
- Edges not incident on giant component being explored one by one

Stage II

- Typical step consists of examining some entry w in $\mathrm{adj}(v)$
- Edges not incident on giant component being explored one by one
- Probability of finding a giant component vertex increases

Stage II

- Typical step consists of examining some entry w in $\mathsf{adj}(v)$
- Edges not incident on giant component being explored one by one
- Probability of finding a giant component vertex increases
- If all vertices equally likely, p(encountering giant comp vertex) = $\theta n/(n-1) \geq \theta$

Stage II

- Typical step consists of examining some entry w in $\mathsf{adj}(v)$
- Edges not incident on giant component being explored one by one
- Probability of finding a giant component vertex increases
- If all vertices equally likely, p(encountering giant comp vertex) = $\theta n/(n-1) \geq \theta$
- All vertices not equally likely

Lemma

Let C_0 be the giant component vertex set in G_0 found during Stage I of the algorithm. Whenever an entry $w \in adj(v)$ is examined, the probability that $w \in C_0$, given that $\{v, w\} \notin E_0$ is $\geq \theta$.

- For any vertex v, the number enteries $w \in \mathbf{adj}(v)$ examined such that $w \in C_0$ is ≤ 1

Lemma

Let C_0 be the giant component vertex set in G_0 found during Stage I of the algorithm. Whenever an entry $w \in adj(v)$ is examined, the probability that $w \in C_0$, given that $\{v, w\} \notin E_0$ is $\geq \theta$.

- For any vertex v, the number enteries $w \in \mathrm{adj}(v)$ examined such that $w \in C_0$ is ≤ 1
- For $\{v, w\} \notin E_0$, probability that w is in C_0 is $\geq \theta$

Lemma

Let C_0 be the giant component vertex set in G_0 found during Stage I of the algorithm. Whenever an entry $w \in adj(v)$ is examined, the probability that $w \in C_0$, given that $\{v, w\} \notin E_0$ is $\geq \theta$.

- For any vertex v, the number enteries $w \in \mathbf{adj}(v)$ examined such that $w \in C_0$ is ≤ 1
- For $\{v, w\} \notin E_0$, probability that w is in C_0 is $\geq \theta$
- Expected number of w examined where $\{v, w\} \notin E_0$ is $\leq 1/\theta$

Lemma

Let C_0 be the giant component vertex set in G_0 found during Stage I of the algorithm. Whenever an entry $w \in adj(v)$ is examined, the probability that $w \in C_0$, given that $\{v, w\} \notin E_0$ is $\geq \theta$.

- For any vertex v, the number enteries $w \in \mathbf{adj}(v)$ examined such that $w \in C_0$ is ≤ 1
- For $\{v, w\} \notin E_0$, probability that w is in C_0 is $\geq \theta$
- Expected number of w examined where $\{v, w\} \notin E_0$ is $\leq 1/\theta$
- Summing over all $v \in V$, expected number of edges examined in Stage II is $O(n/\theta) = O(n)$

- Consider k^{th} examination in Stage II

PROOF OF LEMMA

- Consider k^{th} examination in Stage II
- Edges examined : E_0 and $\alpha = (v_1, w_1), \ldots (v_{k-1}, w_{k-1})$

PROOF OF LEMMA

- Consider k^{th} examination in Stage II
- Edges examined : E_0 and $\alpha = (v_1, w_1), \ldots (v_{k-1}, w_{k-1})$
- i^{th} edge examination selects w_i from $adj(v_i)$

- Consider k^{th} examination in Stage II
- Edges examined : E_0 and $\alpha = (v_1, w_1), \ldots (v_{k-1}, w_{k-1})$
- i^{th} edge examination selects w_i from $adj(v_i)$

Let $\hat{G} = (V, \hat{E})$ where $\hat{E} = E - E_0$ be the ‘residual graph‘ at end of Stage I

- Consider k^{th} examination in Stage II
- Edges examined : E_0 and $\alpha = (v_1, w_1), \dots (v_{k-1}, w_{k-1})$
- i^{th} edge examination selects w_i from $adj(v_i)$

Let $\hat{G} = (V, \hat{E})$ where $\hat{E} = E - E_0$ be the ‘residual graph‘ at end of Stage I

1. $|\hat{E}| = m - |E_0|$

- Consider k^{th} examination in Stage II
- Edges examined : E_0 and $\alpha = (v_1, w_1), \dots (v_{k-1}, w_{k-1})$
- i^{th} edge examination selects w_i from $adj(v_i)$

Let $\hat{G} = (V, \hat{E})$ where $\hat{E} = E - E_0$ be the ‘residual graph‘ at end of Stage I

1. $|\hat{E}| = m - |E_0|$
2. $\hat{E} \cap E_0 = \phi$

- Consider k^{th} examination in Stage II
- Edges examined : E_0 and $\alpha = (v_1, w_1), \ldots (v_{k-1}, w_{k-1})$
- i^{th} edge examination selects w_i from $adj(v_i)$

Let $\hat{G} = (V, \hat{E})$ where $\hat{E} = E - E_0$ be the ‘residual graph‘ at end of Stage I

1. $|\hat{E}| = m - |E_0|$
2. $\hat{E} \cap E_0 = \phi$
   Given $\hat{G}, E_0$ the probability that algorithm produces trace α is denoted by $p(\alpha|\hat{G}, E_0)$. α satisfies the following condition:

PROOF OF LEMMA

- Consider k^{th} examination in Stage II
- Edges examined : E_0 and $\alpha = (v_1, w_1), \ldots (v_{k-1}, w_{k-1})$
- i^{th} edge examination selects w_i from $adj(v_i)$

Let $\hat{G} = (V, \hat{E})$ where $\hat{E} = E - E_0$ be the 'residual graph' at end of Stage I

1. $|\hat{E}| = m - |E_0|$
2. $\hat{E} \cap E_0 = \phi$
   Given $\hat{G}, E_0$ the probability that algorithm produces trace α is denoted by $p(\alpha|\hat{G}, E_0)$. α satisfies the following condition:
3. If (v_i, w_i) occurs in α, then $\{v_i, w_i\} \in E_0 \cup \hat{E}$

PROOF OF LEMMA

- Consider k^{th} examination in Stage II
- Edges examined : E_0 and $\alpha = (v_1, w_1), \ldots (v_{k-1}, w_{k-1})$
- i^{th} edge examination selects w_i from $adj(v_i)$

Let $\hat{G} = (V, \hat{E})$ where $\hat{E} = E - E_0$ be the ‘residual graph‘ at end of Stage I

1. $|\hat{E}| = m - |E_0|$
2. $\hat{E} \cap E_0 = \phi$
   Given $\hat{G}, E_0$ the probability that algorithm produces trace α is denoted by $p(\alpha|\hat{G}, E_0)$. α satisfies the following condition:
3. If (v_i, w_i) occurs in α, then $\{v_i, w_i\} \in E_0 \cup \hat{E}$
   Let S denote set of all graphs which satisfy (1) - (3)

Consider the i^{th} examination.

Consider the i^{th} examination.

- v_i will be uniquely determined by E_0 and $(v_1, w_1), (v_2, w_2), \ldots (v_{i-1}, w_{i-1})$

PROOF OF LEMMA

Consider the i^{th} examination.

- v_i will be uniquely determined by E_0 and $(v_1, w_1), (v_2, w_2), \ldots (v_{i-1}, w_{i-1})$
- Let b_i edges incident on v_i be selected in E_0

Consider the i^{th} examination.

- v_i will be uniquely determined by E_0 and $(v_1, w_1), (v_2, w_2), \ldots (v_{i-1}, w_{i-1})$
- Let b_i edges incident on v_i be selected in E_0
- If a_i vertices have already been selected from $\mathsf{adj}(v_i)$ and $d(v_i, G)$ denotes the degree of v_i in graph G,

PROOF OF LEMMA

Consider the i^{th} examination.

- v_i will be uniquely determined by E_0 and $(v_1, w_1), (v_2, w_2), \dots (v_{i-1}, w_{i-1})$
- Let b_i edges incident on v_i be selected in E_0
- If a_i vertices have already been selected from $\mathbf{adj}(v_i)$ and $d(v_i, G)$ denotes the degree of v_i in graph G,

$$\begin{aligned}
\mathcal{P}(w_i | E_0, \dots (v_{i-1}, w_{i-1})) &= \frac{1}{d(v_i, G) - a_i} \\
\Rightarrow \mathcal{P}((v_i, w_i) | \hat{G}, E_0, \dots (v_{i-1}, w_{i-1})) &= \frac{1}{d(v_i, G) - a_i} = \frac{1}{d(v_i, \hat{G}) - a_i + b_i} \\
\Rightarrow p(\alpha | \hat{G}, E_0) &= \prod_{i=1}^{k-1} \frac{1}{d(v_i, \hat{G}) - a_i + b_i} \qquad (1)
\end{aligned}$$

Consider the i^{th} examination.

- v_i will be uniquely determined by E_0 and $(v_1, w_1), (v_2, w_2), \ldots (v_{i-1}, w_{i-1})$
- Let b_i edges incident on v_i be selected in E_0
- If a_i vertices have already been selected from $\mathbf{adj}(v_i)$ and $d(v_i, G)$ denotes the degree of v_i in graph G,

$$\mathcal{P}(w_i|E_0, \ldots (v_{i-1}, w_{i-1})) = \frac{1}{d(v_i, G) - a_i}$$

$$\Rightarrow \mathcal{P}((v_i, w_i)|\hat{G}, E_0, \ldots (v_{i-1}, w_{i-1})) = \frac{1}{d(v_i, G) - a_i} = \frac{1}{d(v_i, \hat{G}) - a_i + b_i}$$

$$\Rightarrow p(\alpha|\hat{G}, E_0) = \prod_{i=1}^{k-1} \frac{1}{d(v_i, \hat{G}) - a_i + b_i} \quad (1)$$

Note that the function is decreasing with $d(v_i, \hat{G})$. Define function $f_{u,\alpha}$ by

Note that the function is decreasing with $d(v_i, \hat{G})$. Define function $f_{u,\alpha}$ by

$$f_{u,\alpha}(d(u, \hat{G})) = \prod_{i, v_i = u} \frac{1}{d(v_i, \hat{G}) - a_i + b_i}$$

Note that the function is decreasing with $d(v_i, \hat{G})$. Define function $f_{u,\alpha}$ by

$$f_{u,\alpha}(d(u, \hat{G})) = \prod_{i, v_i = u} \frac{1}{d(v_i, \hat{G}) - a_i + b_i}$$

We can now write

$$p(\alpha|\hat{G}, E_0) = \prod_{u \in V - C_0} f_{u,\alpha}(d(u, \hat{G})) \tag{2}$$

We've
$p(\hat{G}|E_0)$ is constant ($= \frac{1}{\binom{N-|E_0|}{m-|E_0|}}$). Using Baye's theorem,

We've
$p(\hat{G}|E_0)$ is constant ($= \frac{1}{\binom{N-|E_0|}{m-|E_0|}}$). Using Baye's theorem,

$$p(\hat{G}|\alpha, E_0) = \frac{p(\alpha|\hat{G}, E_0)p(\hat{G}|E_0)}{p(\alpha|E_0)}$$
$$\Rightarrow p(\hat{G}|\alpha, E_0) = c \prod_{u \in V - C_0} f_{u,\alpha}(d(u, \hat{G})) \tag{3}$$

After α has occurred, we are choosing $w_k \in \mathsf{adj}(v_k)$

After α has occurred, we are choosing $w_k \in \textbf{adj}(v_k)$

- Call w_k eligible if $w_k \neq v_k$, $\textbf{adj}(w_k)$ is not exhausted and $\{v_k, w_k\} \notin E_0$

After α has occurred, we are choosing $w_k \in \mathbf{adj}(v_k)$

- Call w_k eligible if $w_k \neq v_k$, $\mathbf{adj}(w_k)$ is not exhausted and $\{v_k, w_k\} \notin E_0$
- Let $x \in C_0$, we'll show $\mathcal{P}(w_k = x | w_k \textit{ is eligible}) \geq 1/(n-1)$

For a positive integer d

For a positive integer d

- let $G_d = \{\hat{G} \in S | d(v_k, \hat{G}) = d\}$

For a positive integer d

- let $G_d = \{\hat{G} \in S | d(v_k, \hat{G}) = d\}$
- Define $G_{d+} = G_d \cap \{\hat{G} = (V, \hat{E} | \{v_k, x\} \in \hat{E}\}$

For a positive integer d

- let $G_d = \{\hat{G} \in S | d(v_k, \hat{G}) = d\}$
- Define $G_{d+} = G_d \cap \{\hat{G} = (V, \hat{E} | \{v_k, x\} \in \hat{E}\}$
- And $G_{d-} = G_d \setminus G_{d+}$

For a positive integer d

- let $G_d = \{\hat{G} \in S | d(v_k, \hat{G}) = d\}$
- Define $G_{d+} = G_d \cap \{\hat{G} = (V, \hat{E} | \{v_k, x\} \in \hat{E}\}$
- And $G_{d-} = G_d \setminus G_{d+}$
- Define relation R on $G_{d+} \times G_{d-}$ by $(H_1, H_2) \in R$ iff $H_1 \in G_{d+}, H_2 \in G_{d-}$ and H_2 is obtained from H_1 by deleting edge $\{v_k, x\}$ and adding some other edge $\{v_k, x'\}$ incident to v_k such that x' is eligible.

- If $(H_1, H_2) \in R$ then in going from H_1 to H_2, we remove $\{v_k, x\}$ and add $\{v_k, x'\}$. If $x' \in C_0$ then $p(\hat{G}|\alpha, E_0)$ remains the same. If $x' \notin C_0$ then $d(x', \hat{G})$ increases, decreasing $p(\hat{G}|\alpha, E_0)$. This implies, $p(H_1|\alpha, E_0) \geq p(H_2|\alpha, E_0)$

PROOF OF LEMMA

- If $(H_1, H_2) \in R$ then in going from H_1 to H_2, we remove $\{v_k, x\}$ and add $\{v_k, x'\}$. If $x' \in C_0$ then $p(\hat{G}|\alpha, E_0)$ remains the same. If $x' \notin C_0$ then $d(x', \hat{G})$ increases, decreasing $p(\hat{G}|\alpha, E_0)$. This implies, $p(H_1|\alpha, E_0) \geq p(H_2|\alpha, E_0)$
- Fixing H_1, we have $\leq (n-1-d)$ options for x' (x' must be eligible). Thus for fixed H_1 we have the atmost $(n-1-d)H_2$.

- If $(H_1, H_2) \in R$ then in going from H_1 to H_2, we remove $\{v_k, x\}$ and add $\{v_k, x'\}$. If $x' \in C_0$ then $p(\hat{G}|\alpha, E_0)$ remains the same. If $x' \notin C_0$ then $d(x', \hat{G})$ increases, decreasing $p(\hat{G}|\alpha, E_0)$. This implies, $p(H_1|\alpha, E_0) \geq p(H_2|\alpha, E_0)$
- Fixing H_1, we have $\leq (n-1-d)$ options for x' (x' must be eligible). Thus for fixed H_1 we have the atmost $(n-1-d)H_2$.
- For fixed H_2, we can replace d of them with x. Thus, there are d graphs H_1 with $(H_1, H_2) \in R$

- If $(H_1, H_2) \in R$ then in going from H_1 to H_2, we remove $\{v_k, x\}$ and add $\{v_k, x'\}$. If $x' \in C_0$ then $p(\hat{G}|\alpha, E_0)$ remains the same. If $x' \notin C_0$ then $d(x', \hat{G})$ increases, decreasing $p(\hat{G}|\alpha, E_0)$. This implies, $p(H_1|\alpha, E_0) \geq p(H_2|\alpha, E_0)$
- Fixing H_1, we have $\leq (n-1-d)$ options for x' (x' must be eligible). Thus for fixed H_1 we have the atmost $(n-1-d)H_2$.
- For fixed H_2, we can replace d of them with x. Thus, there are d graphs H_1 with $(H_1, H_2) \in R$

Sum over $p(H_1|\alpha, E_0) \geq p(H_2|\alpha, E_0)$ over all pairs $(H_1, H_2) \in R$

Sum over $p(H_1|\alpha, E_0) \geq p(H_2|\alpha, E_0)$ over all pairs $(H_1, H_2) \in R$

$$(n-1-d) \sum_{H_1 \in G_{d+}} p(H_1|\alpha, E_0) \geq d \sum_{H_2 \in G_{d-}} p(H_2|\alpha, E_0)$$

$$\Rightarrow \sum_{H \in G_d} p(H|\alpha, E_0) \geq \frac{d}{n-1} \sum_{\hat{G} \in G_d} p(\hat{G}|\alpha, E_0)$$

Sum over $p(H_1|\alpha, E_0) \geq p(H_2|\alpha, E_0)$ over all pairs $(H_1, H_2) \in R$

$$(n-1-d)\sum_{H_1 \in G_{d+}} p(H_1|\alpha, E_0) \geq d \sum_{H_2 \in G_{d-}} p(H_2|\alpha, E_0)$$

$$\Rightarrow \sum_{H \in G_d} p(H|\alpha, E_0) \geq \frac{d}{n-1} \sum_{\hat{G} \in G_d} p(\hat{G}|\alpha, E_0)$$

Thus, conditional probability that $\hat{G}$ contains $\{v_k, x\}$ is $\geq d/(n-1)$.

Sum over $p(H_1|\alpha, E_0) \geq p(H_2|\alpha, E_0)$ over all pairs $(H_1, H_2) \in R$

$$(n-1-d)\sum_{H_1 \in G_{d+}} p(H_1|\alpha, E_0) \geq d \sum_{H_2 \in G_{d-}} p(H_2|\alpha, E_0)$$

$$\Rightarrow \sum_{H \in G_d} p(H|\alpha, E_0) \geq \frac{d}{n-1} \sum_{\hat{G} \in G_d} p(\hat{G}|\alpha, E_0)$$

Thus, conditional probability that $\hat{G}$ contains $\{v_k, x\}$ is $\geq d/(n-1)$.
The probability that $w_k = x$ is $\geq (\frac{1}{d})\frac{d}{n-1} = \frac{1}{n-1}$

Sum over $p(H_1|\alpha, E_0) \geq p(H_2|\alpha, E_0)$ over all pairs $(H_1, H_2) \in R$

$$(n-1-d)\sum_{H_1 \in G_{d+}} p(H_1|\alpha, E_0) \geq d \sum_{H_2 \in G_{d-}} p(H_2|\alpha, E_0)$$

$$\Rightarrow \sum_{H \in G_d} p(H|\alpha, E_0) \geq \frac{d}{n-1} \sum_{\hat{G} \in G_d} p(\hat{G}|\alpha, E_0)$$

Thus, conditional probability that $\hat{G}$ contains $\{v_k, x\}$ is $\geq d/(n-1)$.
The probability that $w_k = x$ is $\geq (\frac{1}{d})\frac{d}{n-1} = \frac{1}{n-1}$
Summing over all x in C_0 gives $\mathcal{P}(w_k \in C_0) \geq \theta n/(n-1) \geq \theta$

Thus the algorithm finds all connected components of a graph with n nodes in average case $O(n)$ time.
Future work

Thus the algorithm finds all connected components of a graph with n nodes in average case $O(n)$ time.

Future work

We could use these techniques to analyse more algorithms.

Thus the algorithm finds all connected components of a graph with n nodes in average case $O(n)$ time.

Future work

We could use these techniques to analyse more algorithms.

Recently Baswana et al [1] published two algorithms for maintaining the DFS tree in an incremental graph, one with complexity $O(n^{3/2}\sqrt{m})$ and other with complexity $O(n^2)$.

Thus the algorithm finds all connected components of a graph with n nodes in average case $O(n)$ time.

Future work

We could use these techniques to analyse more algorithms.

Recently Baswana et al [1] published two algorithms for maintaining the DFS tree in an incremental graph, one with complexity $O(n^{3/2}\sqrt{m})$ and other with complexity $O(n^2)$.

It has been shown that on random graphs the first algorithm takes time closer to $O(n^2\sqrt{\log n})$. We believe that further work is possible on this and it can be reduced to $O(n^2)$.

ACKNOWLEDGEMENT

I would like to thank Prof. Surender Baswana for his motivation and guidance.

BIBLIOGRAPHY

Surender Baswana and Shahbaz Khan.
Incremental algorithm for maintaining dfs tree for undirected graphs.
In *Automata, Languages, and Programming*, pages 138–149. Springer Berlin Heidelberg, 2014.

Paul Erdos and Alfréd Rényi.
On the evolution of random graphs.
Bull. Inst. Internat. Statist, 38(4):343–347, 1961.

Benny Sudakov Michael Krivelevich.
The phase transition in random graphs - a simple proof.

Thank You !