k-m lower bound.

Date

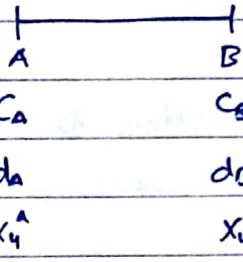
16TH JAN (MEETING 1)

Thm1. If a node has k-value < 0, this doesnot impact the answer set.

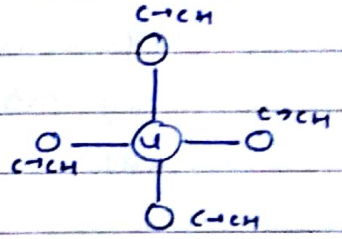
Thm2. Max. k-core value of node u after addition of m edges is ($k_{start\ time} + m$)

(WRONG)

My task: find lowerbound.



$C_B \leq d_B \leq d_A + MC$

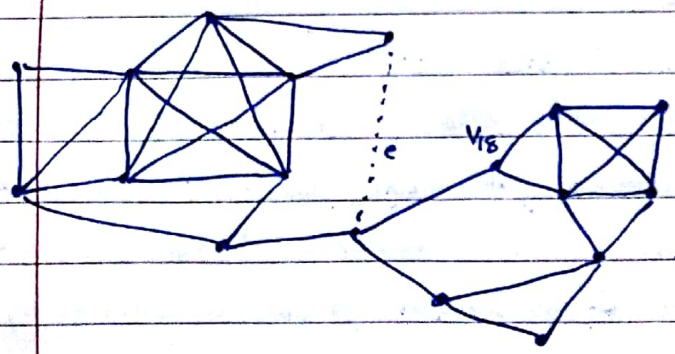


WRONG → $C_B \leq X_u^B \leq \text{Upperlimit}(X_u^B) = X_u^A + MC$

↓

$Y_u^A - MinC \leq \text{Lowerlimit}(Y_u^B) \leq Y_u^B \leq C_B$

WRONG Thm2. Proof of by example:



According to Thm2, Max. k-core of V_{18} after 0 edge insertion to V_{18} is 2+0 = 2.

But after adding edge e, k-core of V_{18} = 3.

Date

23RD JAN (MEETING 2)

- $G_1 + G_2$ $(K + k_2 + 1)$ → it may not hold.
- Upper bound
- Lower bound
- Proof of my algorithm.
- Check monotonicity of X_4, Y_4 in case of insertion & deletion.
  - ↳ Can increase/decrease.

Date

30TH JAN (MEETING 3)

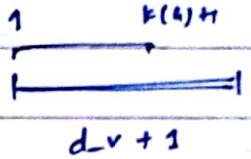
- Formulation of random ordering for degeneracy calculation.
  Shouldn't we have to ensure that there will atleast one right ordering such that $f(G) = k(G)$.
  $f(G)$ = max. back edges from a node in the ordering.
  $k(G)$ = max. k-core value of graph. = degeneracy

- Asia for break due to minors.

- Use of degeneracy of graph in our setting.

- Random Ordering with constraint such that there should be $\leq k(G)$ of neighbours whose index is less than the node.

$$P_0 = \frac{k(G)+1}{d_v+1}$$



$$d_{v_{max}} = c \cdot k(G)$$

$$P_0 \geq \frac{k(G)+1}{c\,k(G)+1} \geq \frac{1}{c}$$ (independence) → This assumption is wrong. $[\because P_1 \not\leq (\frac{1}{c})^n]$

$$P \geq \left(\frac{1}{c}\right)^n + \left(1-\left(\frac{1}{c}\right)^n\right)\left(\frac{1}{c}\right)^n + \cdots + \left(1-\left(\frac{1}{c}\right)^n\right)^{m-1}\left(\frac{1}{c}\right)^n$$

$$P \geq \left(\frac{1}{c}\right)^n \left[\frac{1-\left(1-\frac{1}{c^n}\right)^m}{1-\left(1-\frac{1}{c^n}\right)}\right]$$

$$P \geq 1-\left(1-\left(\frac{1}{c^n}\right)\right)^m$$

$$\bar{P} \leq \left(1-\frac{1}{c^n}\right)^m \leq e^{-\frac{m}{c^n}}$$

- Read Szekeres & Wilf Paper
- Implement my algorithm.
- Read Streaming algorithm paper.

Date

31ST JAN (MEETING 4)

- Big node. - for lower bound. - ravelling - unravelling ✓
- Work out all the proofs.
- Include V_4 in algorithm & VPruning.
- Mix Color & RecolorInsert
- Don't consider nodes of not_reqd set in algo.
- Upper and lower bound from algo itself.

* Big Node Idea for lower/Upper bound: A —— t —— B $G = (V, E)$

let at any point of time t, we have 2 sets of nodes.

Set L = all the nodes whose core value is greater than 0 at any time from A to t.

Set S = V - L

let B = set of edges that we are going to update between A to B.

let not_reqd = set of vertices whose K value is always less than 0 b/w A to B.

let P = { u | ((u,v) ∈ V & u ∈ S & v ∈ L & u ∉ not_reqd)
& ((u,v) ∈ B & u ∈ S & v ∈ L & u ∉ not_reqd) }

INSERTION / DELETION : (u,v),

(i) u ∈ S, v ∈ S: If $V_c \cap P$ not empty then Insert (u,v) else ignore.

(ii) ELSE Insert (u,v).

- Efficient V_c Calculation: Maintain V_c for all nodes in P.

Now, if (u,v) st. u ∉ V_c & v ∉ V_c then ignore.

if (u,v) st. u ∉ V_c & v ∈ V_c then if K(u) < K(v) then ignore.

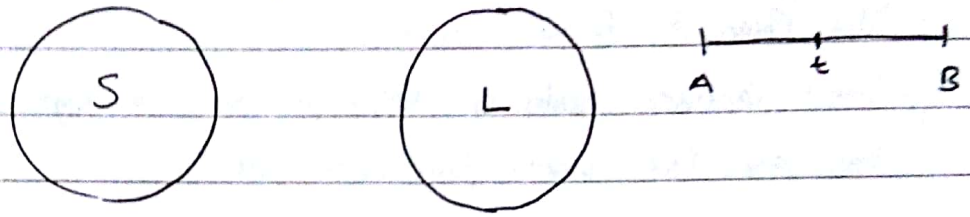
Else Insert(u,v).

Date

6TH FEB (MEETING 5)

- Dataset issue. - Data generation model. ✓

- Sayan Sir's Version of Big Node Idea:



L = Set of all the nodes whose core value is greater than equal to Θ. till time t.

S = Set of all the nodes that are not in L.

d_L = degree of node in G_L.

Case 1: Edge (u,v) s.t. $u \in L$, $v \in L$.

(i) Insertion: $d_L(u) = d_L(u) + 1$; $d_L(v) = d_L(v) + 1$

(ii) Deletion: If $d_L(u) - 1 \geq \Theta$ and $d_L(v) - 1 \geq \Theta$ then $d_L(u)$--; $d_L(v)$--;
WLOG If $d_L(u) < \Theta$ then (If $MCD(u) \geq \Theta$ then $d_L(u)$--; }
else $d_L(u)$--; start BFS my algo from u with calculating the nodes in the BFS. } Note that we have to calculate MCD of nodes in L manually.

$MCD(u) = d_L(u)$ + No. of neighbors in S s.t. core value is Θ.

Same for v.

Case 2: Edge (u,v) s.t. $u \in S$, $v \in S$

(i) Insertion: Use insertion algorithm. Note: Core value of all the nodes in L is assumed to be exactly Θ. Stop when we discover any node from L. during finding of V-c.

(ii) Deletion: Use deletion algorithm. We need to propagate V-c into L also.

Case 3: Edge (u,v) s.t. $u \in S$, $v \in L$

(i) Insertion: Use insertion algorithm. but only on u side.

(ii) Deletion: Use deletion algorithm.

* Note: In set S, either a node has core value Θ or greater than Θ. It doesn't matter. ⇒ we can ignore all edges s.t. $u \in S$, $v \in S$ & K-core of both of them is $\geq \Theta$. Just update the mcd.

Date

20TH FEB (MEETING 6)

- Read Streaming Paper
- Implement BigNode
- Implement V_c less deletion.
- Use YPruning.
- Make 2014 Paper code work.
- Proof of correctness.