

Last week

- Definition of tree-decomposition
- Algorithm for Independent Set on graph with small treewidth,
NP-Hard $\rightarrow O(2^{\text{treewidth}} \cdot |V|)$
- Fixed-parameter Tractability (FPT)

Assume that input has some quantity of
tree-width
degree (maximum #links associating to 1 node)

We want to find an algorithm with time complexity

$f(d)$ [polynomial of input size]

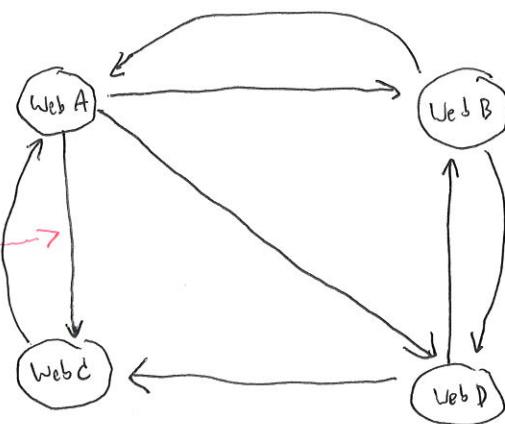
$\hookrightarrow f$ can be arbitrarily large function such as 2^d or d^d .

Because d is small, $f(d)$ will not be that large.

Web graph

We can access
web c by
clicking a link
on web A

[Leskovec et al. 2014, Chapter 5]



Set of nodes = set of webs with a searched key words

Set of links = $\{(u, v) : v \text{ can be accessed by}$

\nearrow clicking a link at $u\}$
not set but
tuple

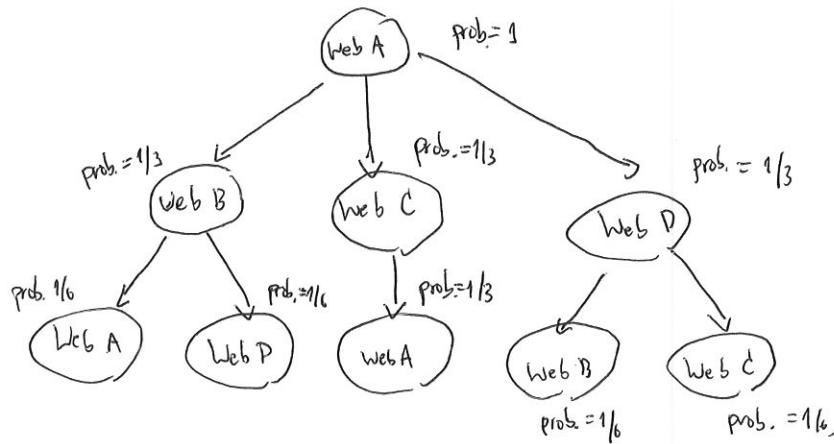
Page Rank : A way to calculate importance of nodes webs

- A drunk man is surfing the internet.
- He will click a link on pages randomly with uniform probability.
- We will calculate the probability of being in each page after infinite steps.

Step 1

Step 2

Step 3



$$\text{Prob} [\text{Web A}] = 1/6 + 1/3 = 1/2$$

$$\text{Pr} [\text{Web B}] = \text{Pr} [\text{Web C}] = \text{Pr} [\text{Web D}] = 1/6$$

Closed form

$$\underbrace{P_{t+1}(A)}_{\text{prob. for web A at time } t+1} = P_t(A)/2 + P_t(C)$$

$$P_{t+1}(B) = P_t(A)/3 + P_t(D)/2$$

$$P_{t+1}(C) = P_t(A)/3 + P_t(D)/2$$

$$P_{t+1}(D) = P_t(A)/3 + P_t(B)/2$$

Assume that, at infinite steps, $P_{t+1}(A) = P_t(A)$, $P_{t+1}(B) = P_t(B)$, ...

$$P(A) = P(B)/2 + P(C)$$

$$P(B) = P(A)/3 + P(D)/2$$

$$P(C) = P(A)/3 + P(D)/2$$

$$P(D) = P(A)/3 + P(B)/2$$

$$1. \quad \begin{bmatrix} p(A) \\ p(B) \\ p(C) \\ p(D) \end{bmatrix} = \begin{bmatrix} 0 & 1/2 & 1/3 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p(A) \\ p(B) \\ p(C) \\ p(D) \end{bmatrix}$$

M

. ✓ . ✓

∴ $\begin{bmatrix} p(A) \\ p(B) \\ p(C) \\ p(D) \end{bmatrix}$ is an eigenvector of M corresponding to eigenvalue $\lambda = 1$

$$\therefore \begin{bmatrix} p(A) \\ p(B) \\ p(C) \\ p(D) \end{bmatrix} = \begin{bmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{bmatrix}$$

- Web A has more importance than Webs B, C, D
- Webs B, C, D have equal importances.

Properties of PageRank

- You will have more scores when you have links from a page with large score.

$$P(\text{your web}) = \frac{1}{\# \text{linked}}_{\text{from } A} p(A) + \frac{1}{\# \text{linked}}_{\text{from } B} p(B) + \dots + \frac{1}{\# \text{linked}}_{\text{from } P} p(P)$$

when $p(A)$ is large
 $P(\text{your web})$ tends to
be large.

$\{A, \dots, P\}$: Set of webs with links
to your web

- You will have a larger score if a distinguished person endorses you. than when 1,000 unknown persons endorse you.

- PageRank will prevent the case that someone creates 1,000 webs that have links to their own pages.

Personalized Page Rank

- Google will provide different results for different users

Conventional Page Rank : $p = M_p$

↓
 a vector that contains probabilities of each web
 ↗
 Matrix that include the fact that the link is clicked randomly.

Personalized PageRank : $P = (1-\beta)M_p + \beta q$ ↗ personalized vector

- Users sometimes go to an address box and press URLs of the web they want to visit.
- The probability distribution of webs accessed by the address box is q .

Ex $q = [0.1 \ 0.3 \ 0.5 \ 0.2]^T$

↑
Users may like Web C but not Web A

- The probability that users go to the address box is β .

Ex $p = 0.5$

$$\begin{bmatrix} p(A) \\ p(B) \\ p(C) \\ p(D) \end{bmatrix} = 0.5 \begin{bmatrix} 0 & 1/2 & 1/4 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix} \begin{bmatrix} p(A) \\ p(B) \\ p(C) \\ p(D) \end{bmatrix} + 0.5 \begin{bmatrix} 0.1 \\ 0.3 \\ 0.5 \\ 0.2 \end{bmatrix}$$

$$I \cdot p = (1-\beta) \cdot M \cdot p + \beta \cdot q$$

$$I \cdot p = (M - \beta M) \cdot p + \beta q$$

$$I \cdot p - (M - \beta M) \cdot p = \beta q$$

$$(I - M + \beta M) \cdot p = \beta q$$

$$p = \underbrace{(I - M + \beta M)^{-1}}_{\text{can be solved by inverting this matrix}} \cdot (\beta q)$$

(can be solved by inverting this matrix)

→ LU decompositon

$$I - M + \beta M = L U$$

$$L U p = \beta q$$

$$p = U^{-1} L^{-1} (\beta q)$$

LU decomposition

$$(I - M + \beta \cdot M) = \begin{bmatrix} \text{some numbers} & \text{all zeros} \\ \text{all zeros} & \text{some numbers} \end{bmatrix}$$

Ex

we want to make these numbers (

$$\begin{bmatrix} 1 & -1/4 & -1/2 & 0 \\ -1/6 & 1 & 0 & -1/4 \\ -1/6 & 0 & 1 & -1/4 \\ -1/6 & -1/4 & 0 & 1 \end{bmatrix} + 1/6 \cdot \begin{bmatrix} 1 & -1/4 & -1/2 & 0 \\ -1/6 & 1 & 0 & -1/4 \\ -1/6 & 0 & 1 & -1/4 \\ -1/6 & -1/4 & 0 & 1 \end{bmatrix}$$

very close to Laplacian matrix

=
we will do this operation from all nonzero entries in the lower half of the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1/6 & 1 & 0 & 0 \\ -1/6 & 0 & 1 & 0 \\ -1/6 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -1/4 & -1/2 & 0 \\ 23/24 & -1/12 & -1/4 & 0 \\ -1/24 & 11/12 & -1/4 & 0 \\ -1/24 & -1/12 & 1 & 0 \end{bmatrix}$$

we eliminate all 0's in this column

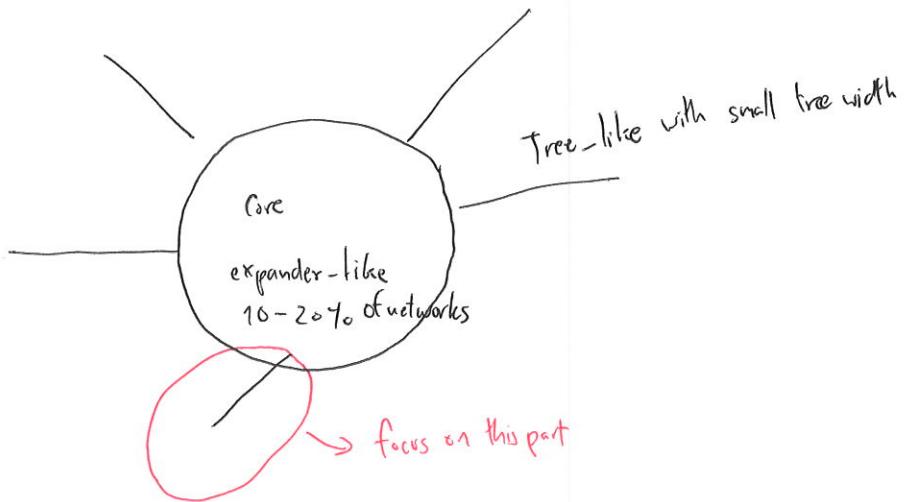
but we have more non-zero's here.

non-zeros at

- 1) diagonal line
- 2) row i column j when j has a link to i

Question: Which row order would increase the minimum number of non-zeros?

Social Network



(Core-Tree) decomposition [Machava et al. KDD 2014]

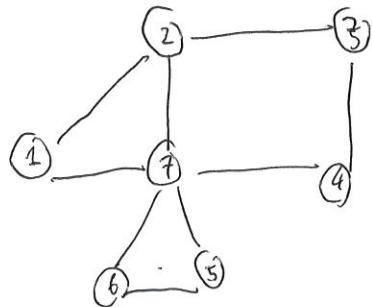
A tree decomposition such that

"for all node i

- o We have the order of nodes $1, \dots, n$ such that

(neighbors of i) $\cap \{i+1, \dots, n\}$ is contained in some bag

Ex



$$\boxed{5, 6, 7} - \boxed{1, 2, 7} - \boxed{2, 7, 3} - \boxed{3, 4, 7}$$

We begin the LU decomposition with order $1, \dots, 7$.

The last node to be eliminated is at the center of the network, with edges to a lot of nodes.
(These nodes will lead to a lot of non-zero entries)

As a result, we will have smaller numbers of added non-zeros in the matrix.

↳ faster LU decomposition.