Incremental PageRank on Dynamic Graph

DMQA Open Seminar

2022.05.08

발표자 소개

❖ 강성현 (Seonghyeon Kang)

- 고려대학교 산업경영공학과
- Data Mining and Quality Analytics Lab (김성범 교수)
- 박사과정 (2021.3 ~)
- dream2globe@korea.ac.kr,
  shyeon.kang@gmail.com

❖ 관심 연구분야

- 머신러닝을 활용한 무선 통신시험 공정 설계
- Anomaly detection, Explainable AI

목차

개요: Graph 정의

❖ **정점(vertex, node)과 그 정점을 연결하는 간선(edge)을 하나로 모아 놓은 자료구조**

- 연결되어 있는 객체 간의 관계 표현이 필요한 다양한 도메인에서 활용되고 있음

Image credit: Medium

Social Networks

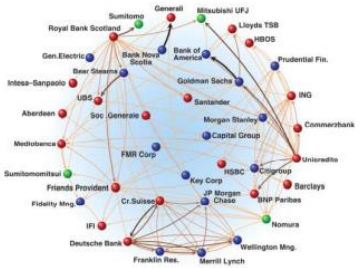
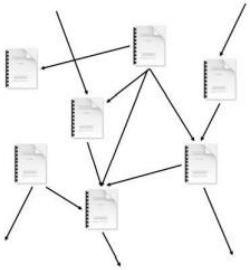


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks



Citation Networks

Image credit: Missoula Current News

Internet

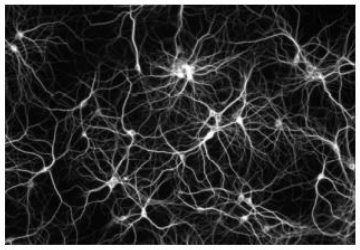


Image credit: The Conversation

Networks of Neurons

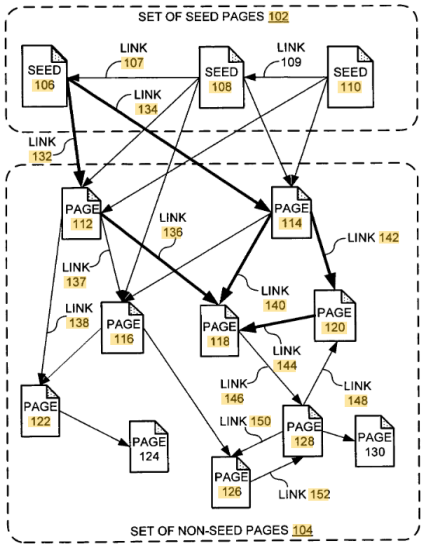
* CS224W: Machine Learning with Graphs 인용

개요: PageRank History

❖ PageRank는 website 중요도를 연결 관계에 기반하여 측정한 지표

- World Wide Web의 문서 간 관계를 표현하기 위해 **web page를 정점**으로 **hyperlink를 간선**으로 정의
- PageRank는 "Web Page"라는 용어와 Google 공동 설립자 "Larry Page"의 이름을 따서 명명되었으며, 오늘날 검색 엔진의 유일한 알고리즘은 아니나 여전히 활용되고 있음

Google 특허의 PageRank 설명



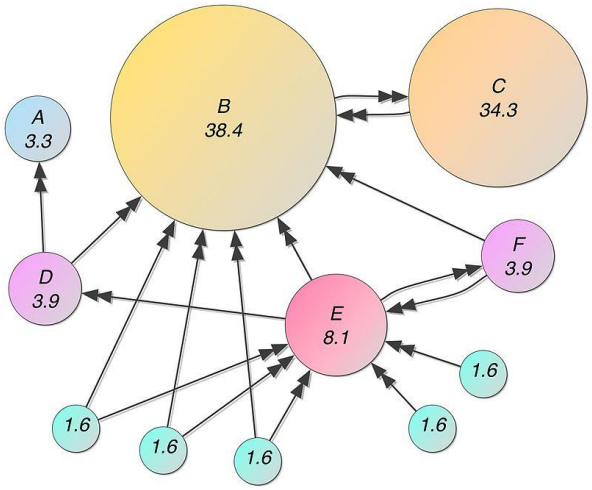
Twitter Mentions

PageRank: Traditional Method

❖ 모든 정점으로부터 도착 확률을 재귀적으로 계산함

- 연결된 페이지가 수와, 연결된 페이지의 중요도가 클 수록 신뢰할 수 있는 페이지로 간주함
- 정점들의 집합을 Γ, 정점 K∈Γ의 t번째 시도에서 중요도를 PR(K; t)라고 했을 때,
  PR(K; t+1) = G×PR(K; t)인 수렴 상태의 PR 값을 중요도로 간주함

PageRank score 예시



PageRank 계산 방법

damping factor

$$PR(K;t+1) = \frac{1-d}{N} + d\sum_{P\in\Gamma_K}\frac{PR(P;t)}{|\Gamma_P|}$$

전체 정점의 수 / 연결 정점의 수

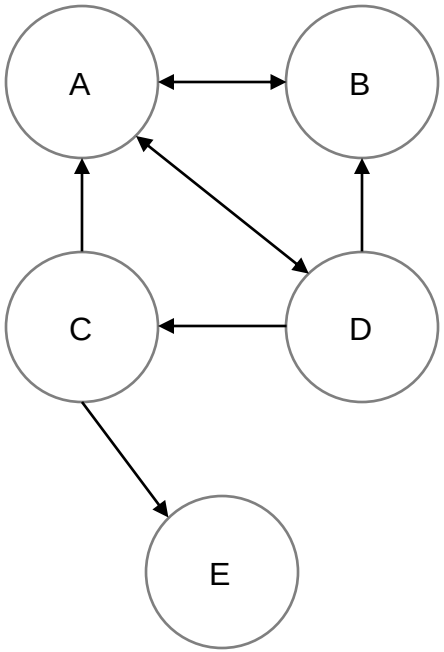
- ✓ $0 \le d \le 1$: 제동계수
- ✓ $\Gamma_K \in \Gamma$: K에서 인용한 정점들의 집합
- ✓ $|\Gamma_p|$: p를 인용한 정점의 개수

* 위키디피아

PageRank: Traditional Method

❖ 산출 방법

- 인접행렬(A) → 확률행렬(H) → 구글행렬(G) 순으로 계산



$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix}$$

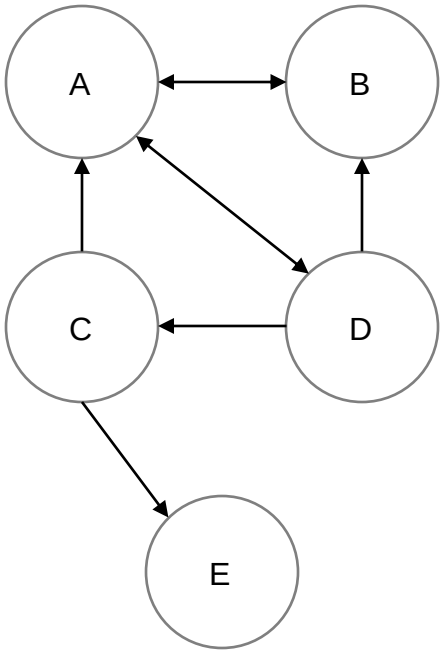
$$\begin{matrix} A & B & C & D & E \end{matrix}$$

* 산업수학혁신센터, https://icim.nims.re.kr/post/easyMath/837

PageRank: Traditional Method

❖ 산출 방법

- 인접행렬(A) → 확률행렬(H) → 구글행렬(G) 순으로 계산



$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 0 & 1 & \frac{1}{2} & \frac{1}{3} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

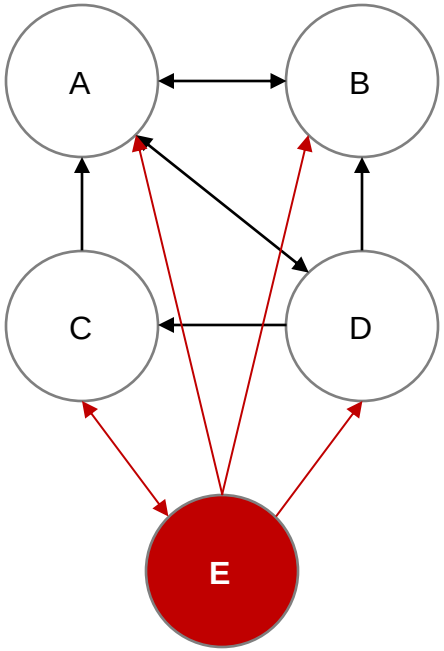
열 A_i를 A_i 성분의 합으로 나눔

* 산업수학혁신센터, https://icim.nims.re.kr/post/easyMath/837

PageRank: Traditional Method

❖ Perron-Frobenius 정리의 irreducible 조건 만족을 위한 장치 ①

- dangling node는 1/N(정점 수) 가중치로 모든 정점과 연결함



$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$S = \begin{bmatrix} 0 & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{5} \\ 1 & 0 & 0 & \frac{1}{3} & \frac{1}{5} \\ 0 & 0 & 0 & \frac{1}{3} & \frac{1}{5} \\ 1 & 0 & 0 & 0 & \frac{1}{5} \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{5} \end{bmatrix}$$

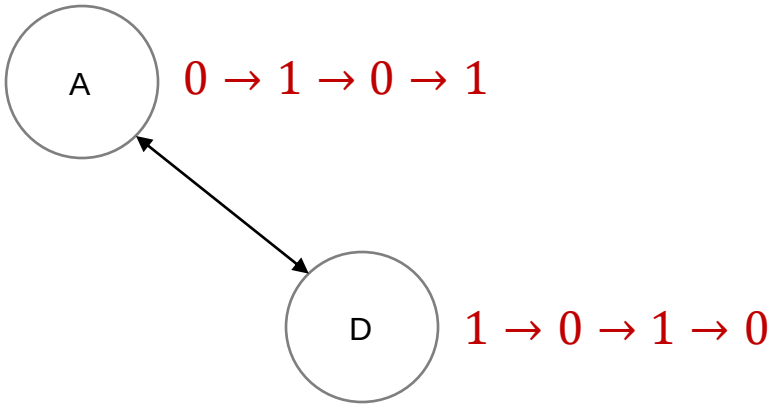
유일한 수렴값(stationary distribution)을 갖기 위한 irreducible 조건 만족 필요

* 산업수학혁신센터, https://icim.nims.re.kr/post/easyMath/837

PageRank: Traditional Method

❖ Perron-Frobenius 정리의 irreducible 조건 만족을 위한 장치 ②

- 값이 수렴하지 않고 진동하는 극단적인 사례

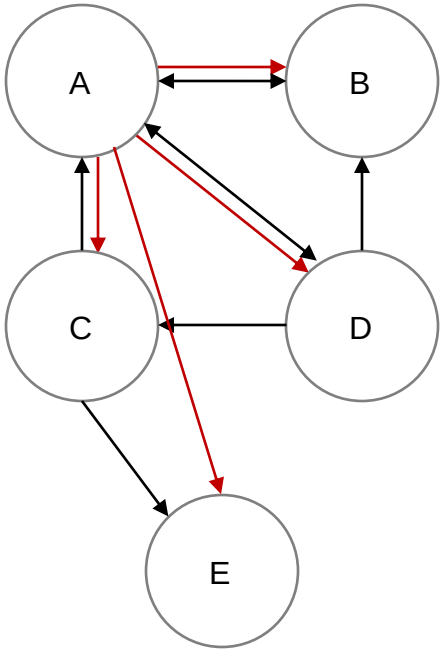


* 산업수학혁신센터, https://icim.nims.re.kr/post/easyMath/837

PageRank: Traditional Method

❖ Random web surfer의 개념을 도입

- 구글행렬(G): 제동 계수(damping factor)를 활용하여 낮은 가중치로 모든 정점과 연결함
- 일반적으로 d 값을 0.85로 설정하여 사용



$$G = dS + (1-d)E$$

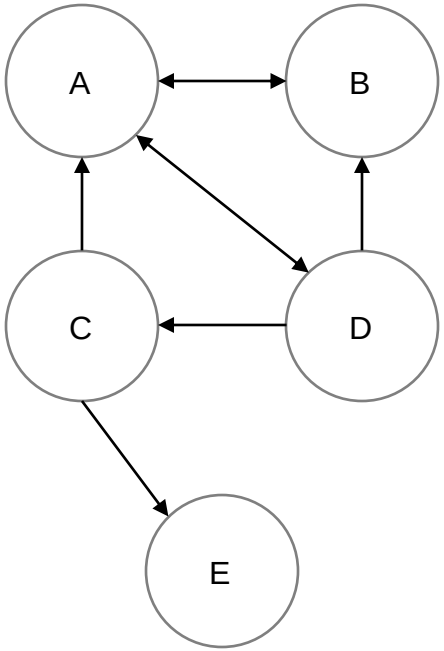
$$G = 0.85\begin{bmatrix} 0 & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{5} \\ 1 & 0 & 0 & \frac{1}{3} & \frac{1}{5} \\ 0 & 0 & 0 & \frac{1}{3} & \frac{1}{5} \\ 1 & 0 & 0 & 0 & \frac{1}{5} \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{5} \end{bmatrix} + 0.15\begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$$

* 산업수학혁신센터, https://icim.nims.re.kr/post/easyMath/837

PageRank: Traditional Method

❖ Irreducible 조건을 만족하는 구글행렬

- $X_{k+1} = GX_k$ 인 G행렬를 산출하기 위해 반복적으로 연산함
- 초기값 X_0은 보통 모든 행렬의 원소를 $\frac{1}{N}$ (N은 정점 수)로 설정



$$X_1 = GX_0 = \begin{bmatrix} \frac{3}{100} & \frac{22}{25} & \frac{91}{200} & \frac{47}{150} & \frac{1}{5} \\ \frac{91}{200} & \frac{3}{100} & \frac{3}{100} & \frac{47}{150} & \frac{1}{5} \\ \frac{3}{100} & \frac{3}{100} & \frac{3}{100} & \frac{47}{150} & \frac{1}{5} \\ \frac{91}{200} & \frac{3}{100} & \frac{3}{100} & \frac{3}{100} & \frac{1}{5} \\ \frac{3}{100} & \frac{3}{100} & \frac{91}{200} & \frac{3}{100} & \frac{1}{5} \end{bmatrix} \begin{bmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{bmatrix}$$

$$X_2 = GX_1$$

$$\vdots$$

$$X_{k+1} = GX_k$$

$$X \approx GX$$

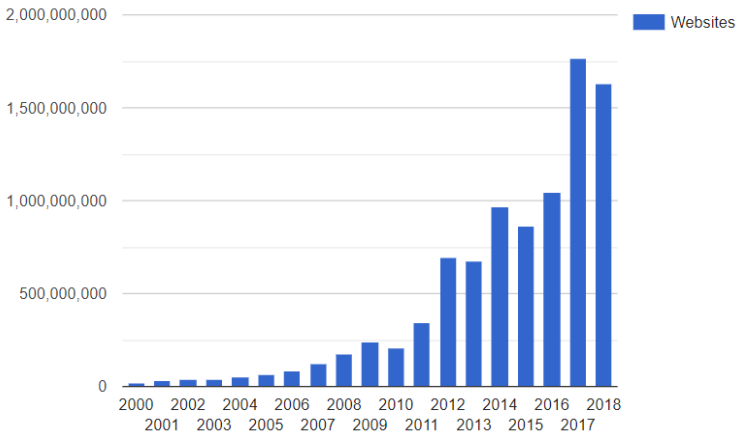
* 산업수학혁신센터, https://icim.nims.re.kr/post/easyMath/837

Incremental PageRank: 현 시점에서의 한계 상황

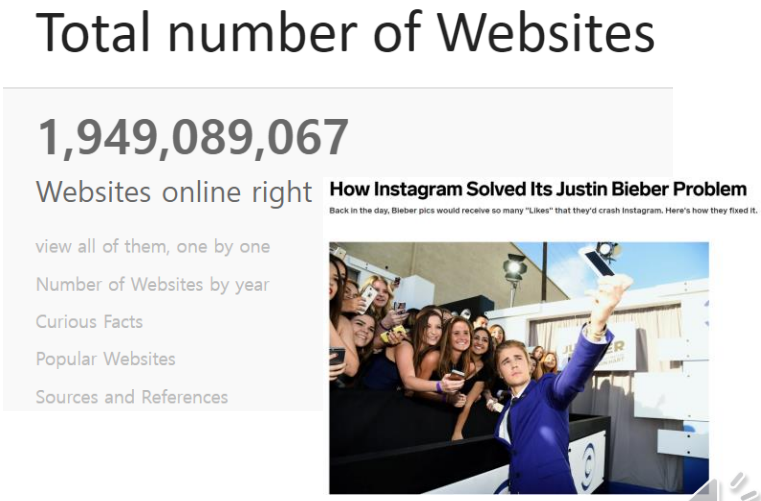
❖ Web의 매우 큰 규모와 동적 변화에 대응하기 어려움

- 현재 웹사이트는 약 19억 개이며, 사이트 간 관계 또한 끊임없이 변함
- PageRank score를 갱신하기 위해 처음부터 계산하는 것은 상당한 컴퓨팅 리소스를 소비하며, 응답 지연이 짧은 응용 프로그램일수록 사용이 어려움

Website 증가 추이(2000 ~ 2018)



초 단위로 증가하는 Website 수



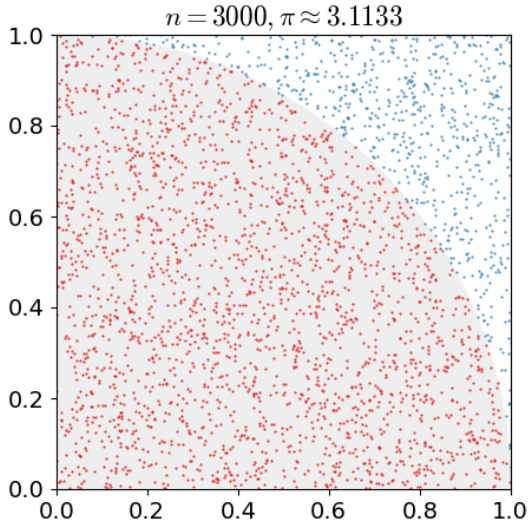
* https://www.internetlivestats.com/total-number-of-websites/

Incremental PageRank: 아이디어

❖ Monte Carlo 방법을 활용한 근사 PageRank을 구함

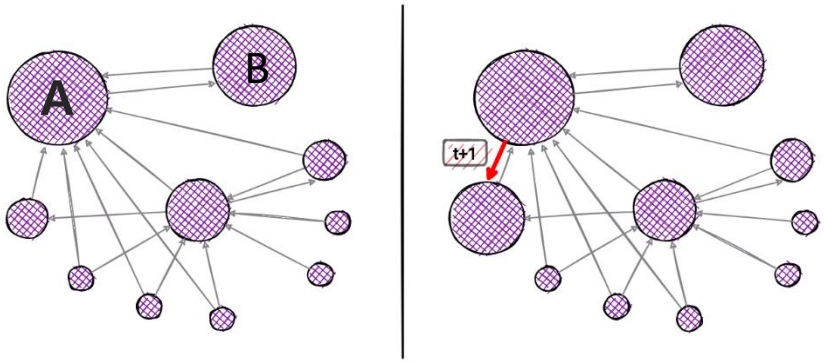
- Monte Carlo 방법: 반복된 무작위 추출을 이용하여 함수 값을 근사하는 알고리즘을 지칭
- 새로운 연결이 전체 PageRank 순서에 큰 영향을 주지는 않을 것이라는 가정
- **도착 확률 → 지나갈 확률**로 관점을 전환한 것이 핵심 아이디어임

Monte Carlo 방법을 통한 원주율 구하기



* 원에 속한 점 개수를 전체 점 개수로 나눈 비율은 π/4를 근사

그래프 전체 score 영향은 미미할 것임



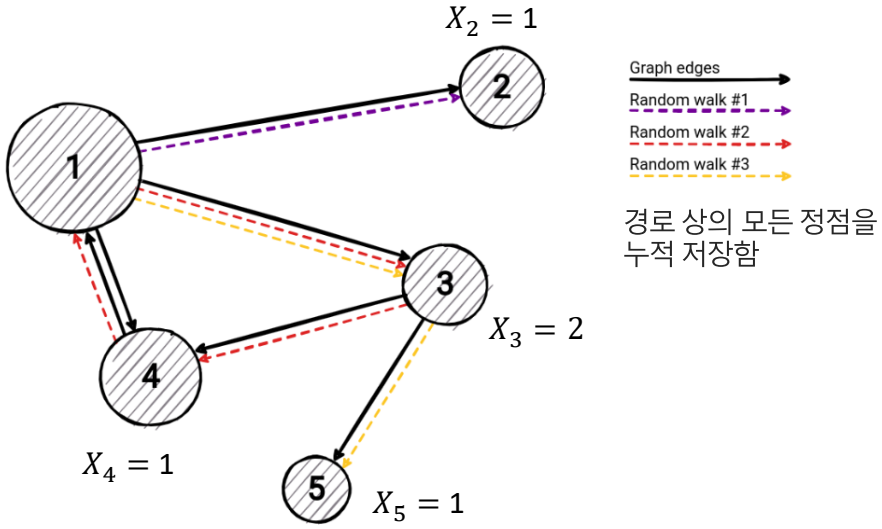
Incremental PageRank: 접근 방법

❖ 산출 방법

- 각 정점에서 R번의 random walk을 시작, 모든 경로를 기록함
- 이 때 PageRank score는 아래 근사 공식과 같이 근사할 수 있음

※ Fast Incremental and Personalized PageRank (2010)

Random walk segments(n=5, R=3 예시)



경로 상의 모든 정점을 누적 저장함

PageRank 근사 공식

$$\pi_v = \frac{X_v}{nR/\epsilon}$$

정점별 방문 횟수

Random Walk 길이

- ✓ π_v: PageRank value for v node
- ✓ X_v: Total number of times that any of the stored walk segments visit v
- ✓ n: Total number of network nodes
- ✓ ϵ: Transmission probability (논문에서는 0.2)

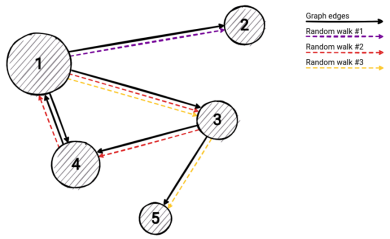
Incremental PageRank: 접근 방법

❖ 기록된 path를 재활용하여 PageRank score를 갱신

Algorithm 1 Personalized PageRank Walk Using Walk Segments

Input: Source node w, required length L of the walk
Output: A personalized PageRank walk P_w for source node w of length at least L

Start the walk at w: $P_w \leftarrow [w]$ 지나간 정점을 저장하는 공간
while length$(P_w) < L$ **do**
  $u \leftarrow$ last node in P_w
  Generate a uniformly random number $\beta \in [0, 1]$
  if $\beta < \epsilon$ **then** 초기 정점으로 돌아갈 확률
    Reset the walk to w: $P_w \leftarrow P_w \cdot \text{append}(w)$
  else
    if u has an unused walk segment Q remaining in memory **then** 기존 지나가지 않았던 정점을 우선하여 이동
      Add Q to the end of P_w: $P_w \leftarrow P_w \cdot \text{append}(Q)$
      Then, reset the walk to w: $P_w \leftarrow P_w \cdot \text{append}(w)$
    else
      if u was previously fetched **then**
        Take a random edge (u, v) out of u
        Add v to the end of P_w: $P_w \leftarrow P_w \cdot \text{append}(v)$
      else
        Do a fetch at u 이웃 정점의 정보와 random walk 정보를 가져옴
      end if
    end if
  end if
end while

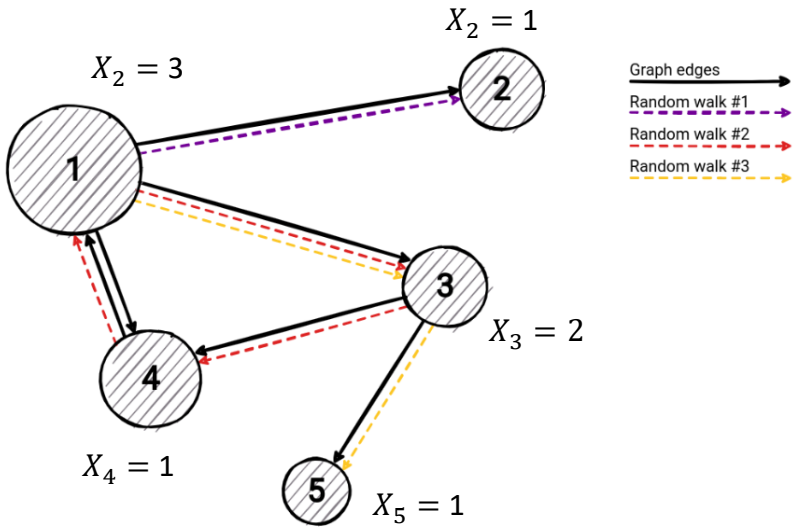


Incremental PageRank: 접근 방법

❖ 기록된 path를 재활용하여 PageRank score를 갱신

- 시작점이 중요하지는 않음, 예시에서는 node 1를 사용함

Random walk segments



PageRank process

✓ Random walk #1 : [1, 2, reset]

✓ Random walk #2 : [1, 2, 1, 3, 4, reset]

✓ Random walk #3 : [1, 2, 1, 3, 4, 1, 3, 5]

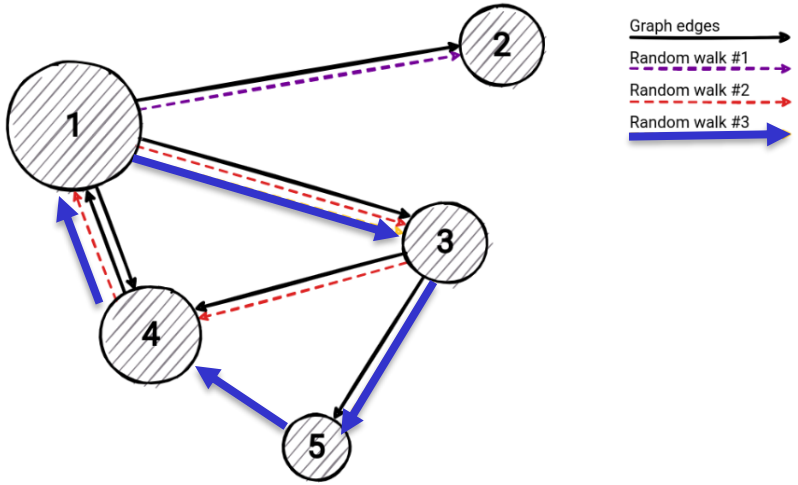
☞ PageRank Vector = [3/8, 1/8, 2/8, 1/8, 1/8]

Incremental PageRank: 접근 방법

❖ 기록된 path를 재활용하여 PageRank score를 갱신

- 갱신된 정점을 지나갔었던 random walk 기록만 제거 후
- 새로운 edge가 생겼다면 우선하여 path로 반영함

Random walk segments(R=1, initial node=1)



✓ Random walk #1 : [1, 2]

✓ Random walk #2 : [1, 2, 1, 3, 4]

~~✓ Random walk #3 : [1, 2, 1, 3, 4, 1, 3, 5]~~

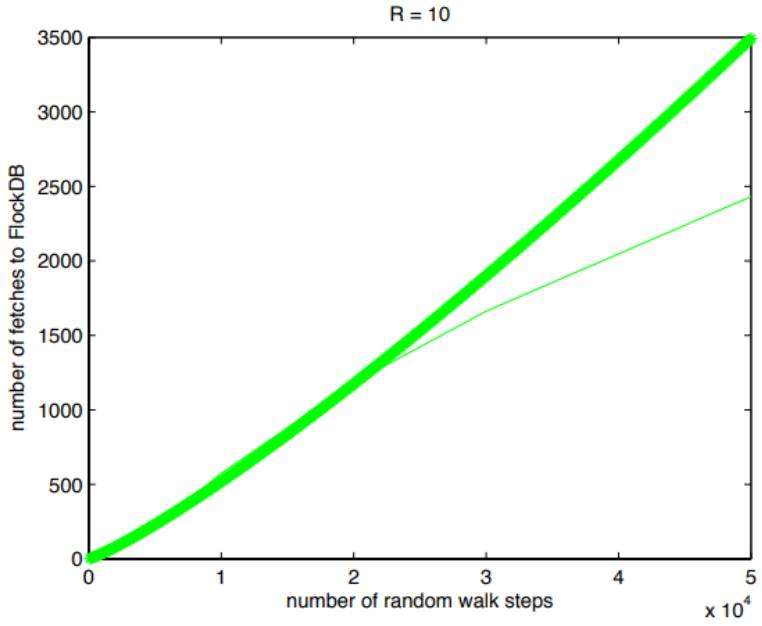
✓ Random walk #4 : [1, 2, 1, 3, 4, 1, 3, 5, 4, 1]

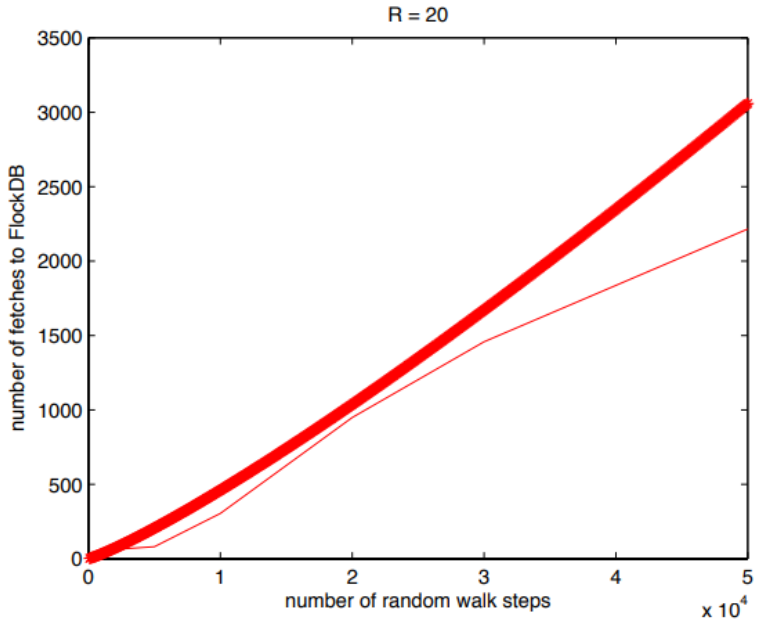
☞ PageRank Vector = [3/10, 1/10, 2/10, 2/10, 2/10]

실험결과

❖ Twitter Network를 활용하여 성능을 검증함

- 100명의 임의 사용자를 활용하였으며, 각 사용자의 friends는 20~30명 수준
- R의 변화에도 fetch 수에는 큰 변화가 없으며,
  10 만번의 random walk에도 fetch수는 2,500회로 매우 낮음



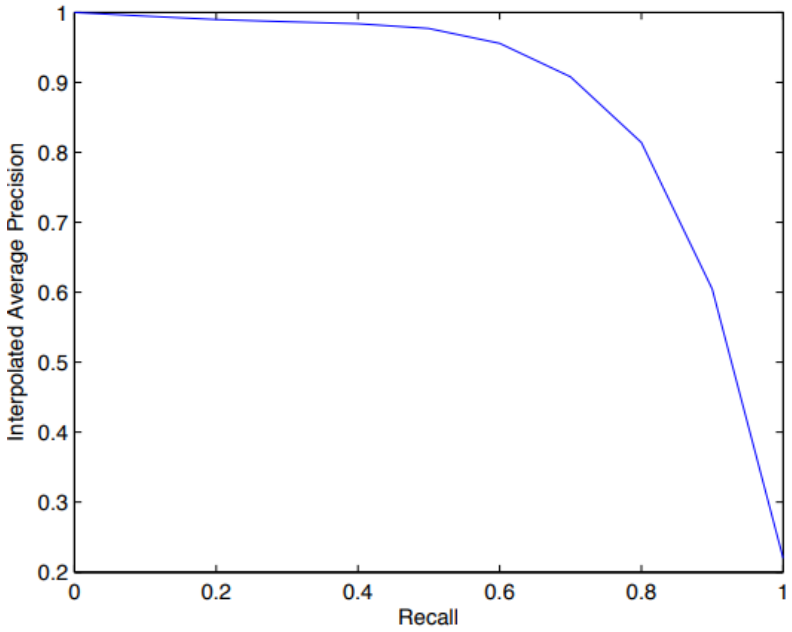


* 두꺼운 선은 이론적인 Max 추정 값, 얇은 선은 실험 결과

실험결과

❖ 적은 random walk 시도에서도 준수한 성능을 기록함

- 50,000 번의 임의보행 결과(상위 100개)를 "참"이라고 가정하고, 5,000 번의 임의보행 시 결과와 비교함
- 재현율 0.7에서 정밀도가 0.9라는 의미는 상위 77개 결과 중에서 상위 100개의 " 참" 결과 중 70개가 검색되었음을 의미



요약

❖ PageRank는 website의 참조 링크를 활용하여 중요도를 계산하는 알고리즘

- 웹사이트를 정점(node), 참조 링크를 간선(edge)으로 정의한 그래프를 활용
- 모든 정점에서 임의보행을 무한대에 가깝게 실행한 후 각 정점에 도착할 확률을 중요도로 간주함

❖ Incremental PageRank는 Web 규모에서도 성능을 보장함

- Monte Carlo 방법을 활용하여 산정을 간소화
- Random walk를 충분히 실행한 후 각 정점 별 방문 횟수를 중요도로 간주함
- RageRank는 웹의 규모 확장에 대응하기 오늘날까지도 연구가 확장 중이며,
  Monte Carlo 방법을 제안했던 Fast Incremental and Personalized PageRank (2010) 내용을 기반한 논문이 다수
  → Fast Incremental PageRank on Dynamic Networks (2019)
  → Towards PageRank Update in a Streaming Graph by Incremental Random Walk (2022)

감사합니다