The PageRank citation ranking: Bringing order to the web.

Stanford InfoLab Technical Report (1999)

L Page, S Brin, R Motwani, T Winograd

CAU
Junseo, Yu

DMAIS Lab Seminar
07.17.2025

Content

Why Do We Rank the Web?

- Web pages proliferate without quality control or publishing costs
  - Large Volume
  - Heterogeneity

Why Do We Rank the Web? (cont.)

❑ Need Of Ranking

- The Web Is Full Of Countless Pages, Many Of Which Are Irrelevant Or Unhelpful
- Users Seek **Relevant** and **Important** Ones
  - *CAU Course Registration Page* vs *Personal Diary About Course Registration*

→ **Need For Ranking To Bring Order To The Chaos**

What Makes a Page Important?

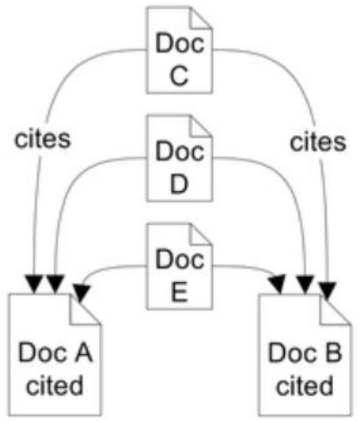
Existing Methods

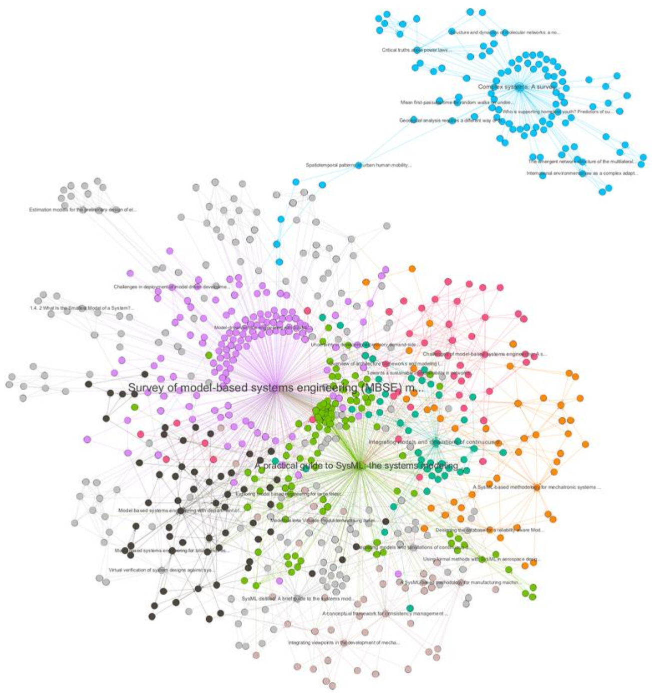
- Citation Analysis in Academia
- Web Hyperlink Structure
- Hubs and Authorities
- Backlink Count in Early Search Engines

What Makes a Page Important? (cont.)

Existing Methods

- **Citation Analysis in Academia**
  - Citation as Importance Proxy
    - Analogy: Hyperlinks ↔ Citations
  - Epidemic Model of Information Flow
    - Intuition: immaterial things can spread via network
  - However, Unlike academic papers,
    web content is produced without rigorous verification
- Web Hyperlink Structure
- Hubs and Authorities
- Backlink Count in Early Search Engines

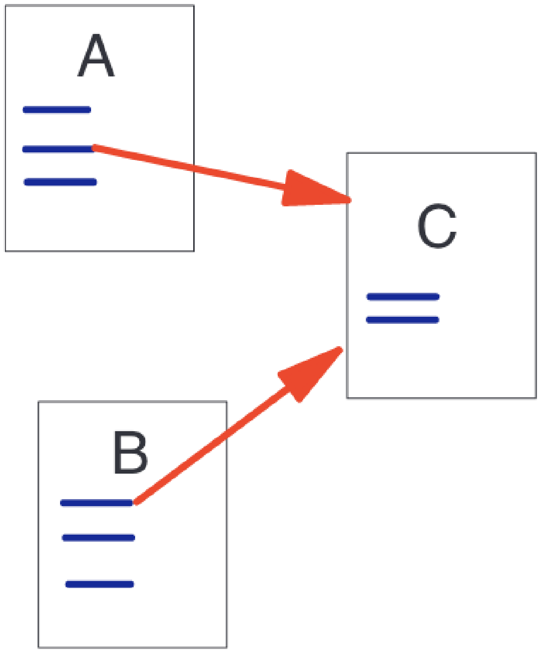




What Makes a Page Important? (cont.)

❑ Existing Methods

- Citation Analysis in Academia
- **Web Hyperlink Structure**
  - Many researchers attempted to analyze the web using its link structure
  - But these efforts did not lead to a quantitative measure of importance.
- Hubs and Authorities
- Backlink Count in Early Search Engines



What Makes a Page Important? (cont.)

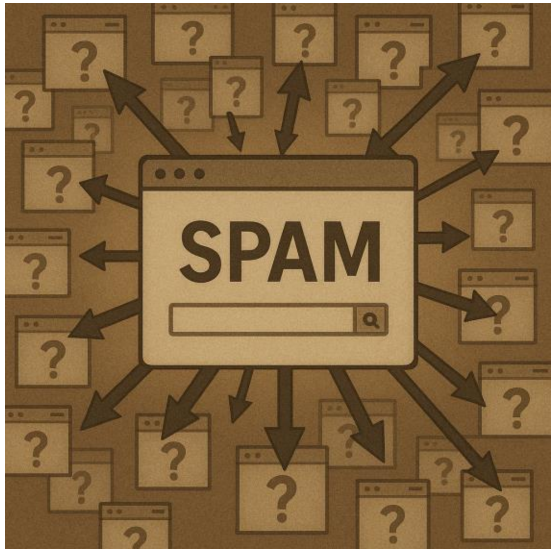
Existing Methods

- Citation Analysis in Academia
- Web Hyperlink Structure
- **Hubs and Authorities**
  - Kleinberg (1998) tried to divides the web into hubs and authorities via the HITS algorithm
  - Good **hubs** link to many authoritative pages, and **authoritative** pages are linked by many good hubs
  - Introduce eigenvector-based importance scoring
  - However, it was **query-dependent** and vulnerable to **spam**
- Backlink Count in Early Search Engines

What Makes a Page Important? (cont.)

❑ Existing Methods

- Citation Analysis in Academia
- Web Hyperlink Structure
- Hubs and Authorities
- Backlink Count in Early Search Engines
  - Many search engines have measured page quality based on **the number of backlinks**
  - Easily manipulated (e.g., spam, self-links)
  - No notion of link source credibility

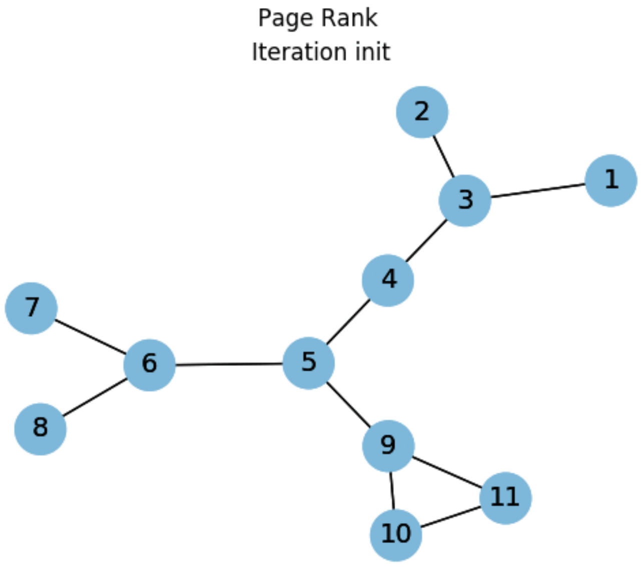


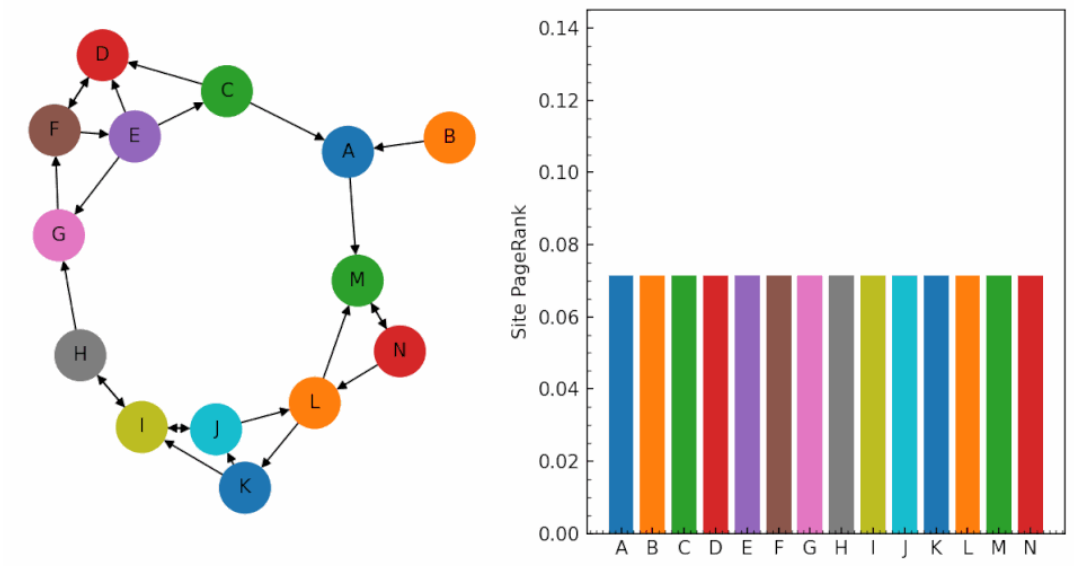
What Makes a Page Important? (cont.)

- **Requirements & Solutions**
  - **Query Independent Importance**
    - Assign the absolute importance to the web pages
    - Measure the importance of web pages **objectively**, even **without** any human **intention**
  - **Robust Against Manipulation**
    - Need to reduce the impact of easily manipulated and meaningless backlinks
    - The influence of A backlink should depend on its quality
- The Link Structure of the web
  - There are objective information; Link structure
    - Of course, other types of information could be considered, but we will focus on the link structure
  - However, simply reflecting the link structure could not capture the importance effectively

Intuition of PageRank: Recursive Importance

- **A Page Is Important, If It Is Connected With Important Pages By Backlinks**
  - Not all imponrtance of backlinks are equal — links from high-ranked pages matter more
  - **Importance** flows recursively through the web graph
  - As this process repeats, importance **circulates** through the graph
  - Over time, this flow converges to a stable and consistent distribution

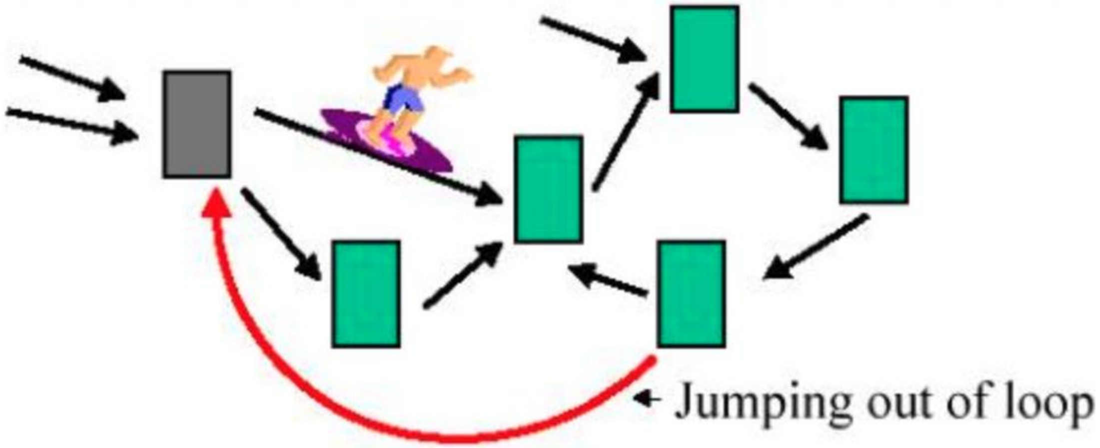


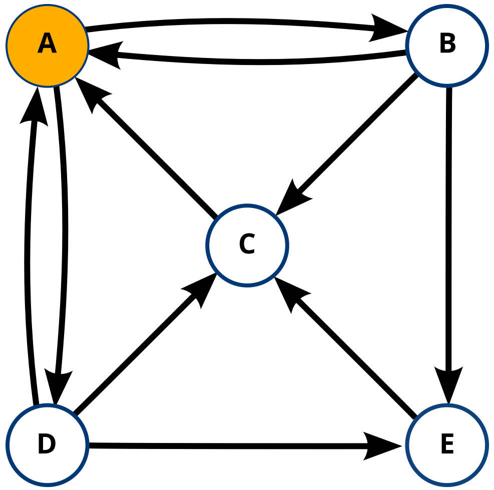


Intuition of PageRank:

- Intuitive and helpful metaphor of **PageRank**
- The random surfer moves along the structure of the web pages
- The final PageRank value represents the probability

  that the surfer ends up at a specific node after infinitely many steps
- This simulates how users navigate by following links



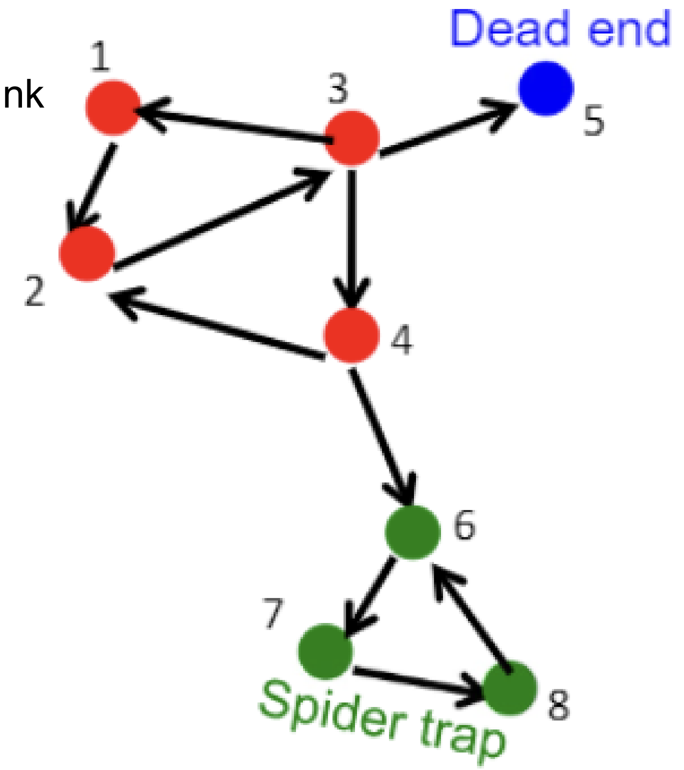


Problems: Rank Sink

❑ Problems

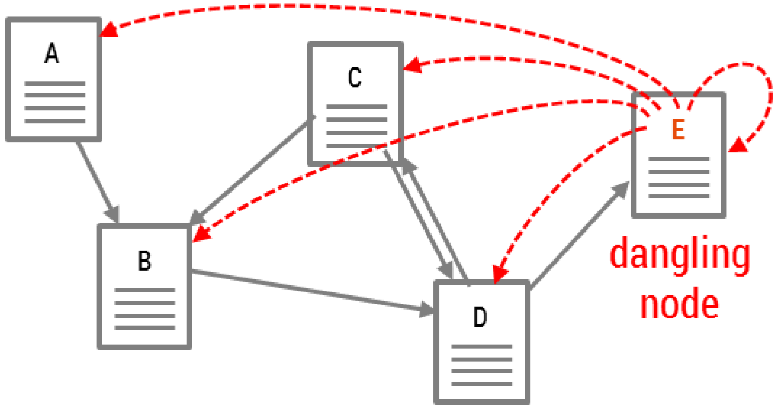
- Consider the situation
  - **Spider Trap:** Page cycles pointed by some other pages but no other outgoing link
  - **Dangling Node (Dead end):** A page pointed by some other pages but no other outgoing link
- This loop will accumulate rank but never distribute any rank outside

  → **Rank Sink**
  - It is called a sink because the rank would sink into those points.
  - Once entered, the poor random surfer can not escape this sink



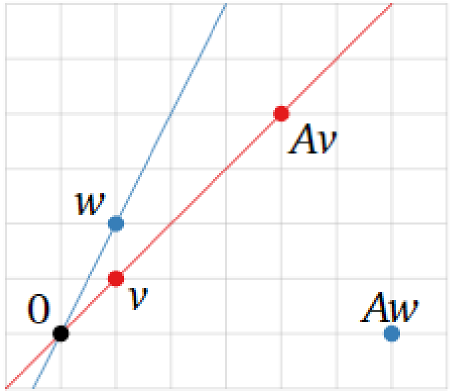
❑ Solutions

- Make the rank would be distributed
- Make the random suffer can exit
- To keep the door slightly open



PageRank as Graph Mining Tools

- Let A be a square matrix with the rows and column corresponding to web pages
  - $A_{u,v} = \begin{cases} \frac{1}{N_u} & \text{if there is an edge from } u \text{ to } v \\ 0 & \text{otherwise} \end{cases}$ $\quad A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
- If we treat R (the relative importance of each web page) as a vector over the web graph
  - $R = cAR \Rightarrow AR = \frac{1}{c}R$
  - ***R*** is an eigenvector of the matrix ***A***, and the constant ***c*** is the corresponding eigenvalue
  - Multiplying by A does not change the link structure of the web pages
  - The inherent **dominant eigenvector of A** can be computed by repeatedly applying ***A*** to any initial vector
  - Therefore, **PageRank** can be interpreted as a method for mining structural features of the graph



v is an eigenvector

w is not an eigenvector

PageRank Formulation: Naïve version

- Importance flows recursively through the web graph
- The importance of the node is the sum of the normalized importance of its backlinks
- The rank of a page is divided among its forward links evenly to contribute to the ranks of the pages they point to

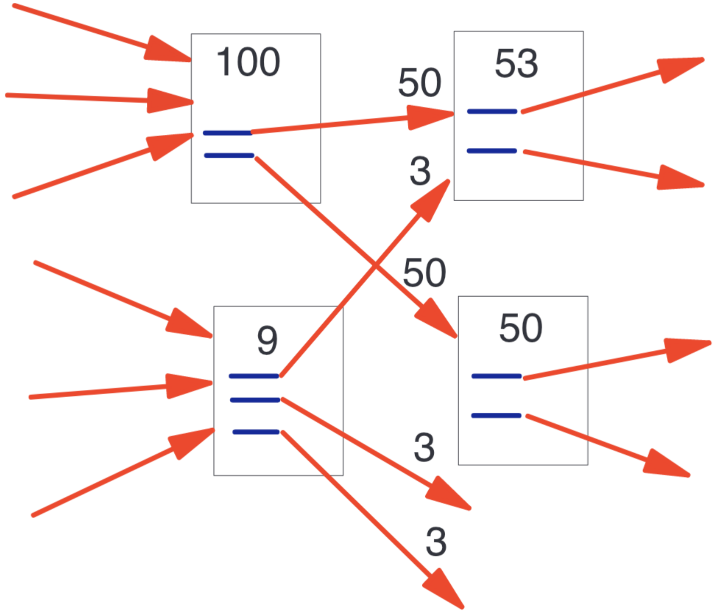
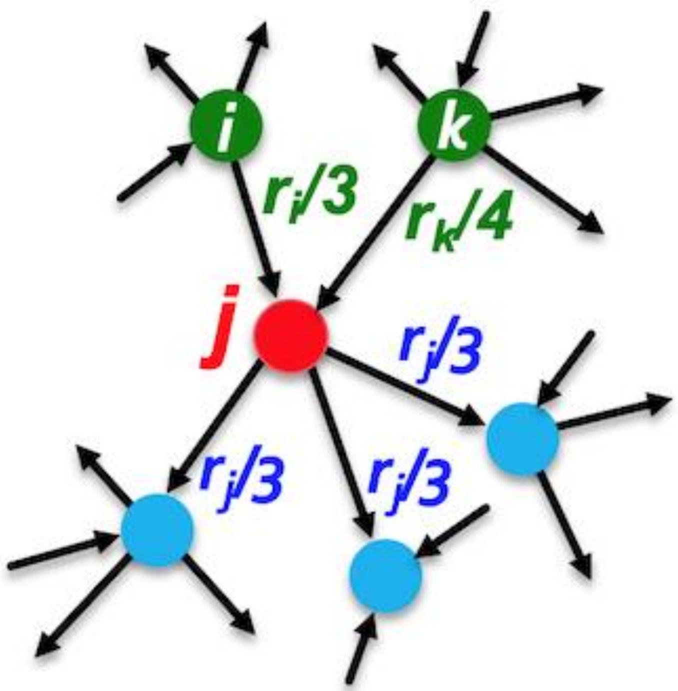


Figure 2: Simplified PageRank Calculation



$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

F_u: the set of forward pages of u (outward direction of u)

B_u: the set of backward pages of u (inward direction of u)

$N_u = |F_u|$

C: A factor used for normalization

PageRank Formulation: Navie version (cont.)

- ❑ Importance flows recursively through the web graph
- ❑ The importance of the node is the sum of the normalized importance of its backlinks
- ❑ The rank of a page is divided among its forward links evenly to contribute to the ranks of the pages they point to

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

F_u: the set of forward pages of u (outward direction of u)

B_u: the set of backward pages of u (inward direction of u)

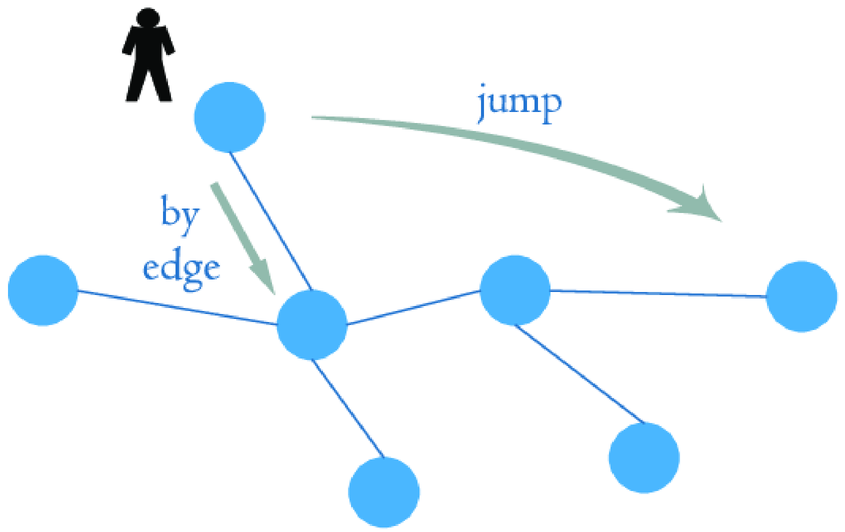
$N_u = |F_u|$

C: A factor used for normalization

Why do we need the constant c?

- ❑ If c > 1,
  the recursive amplification effect would become stronger,
  and the system may diverge
- ❑ Even when c = 1,
  the rank values can converge in some cases
- ❑ Introducing a damping factor c < 1,
  helps values gradually converges

PageRank Formulation: Modified version



❑ Solutions of Rank Sink

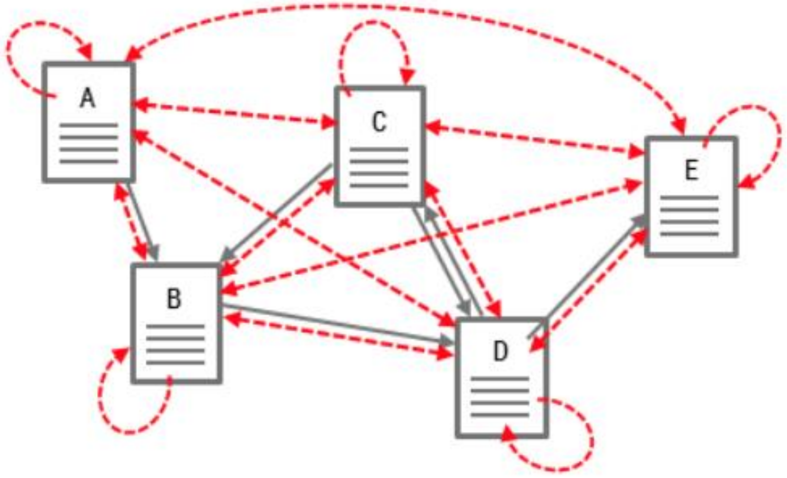
- Make the rank would be distributed
- Make the random suffer can exit
- To keep the door slightly open

❑ Formulation

- E(u): A source of rank
  - This allows the random surfer to escape by jumping to a random page

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u)$$

such that c is maximized and $||R'||_1 = 1$ ($||R'||_1$ denotes the L_1 norm of R').



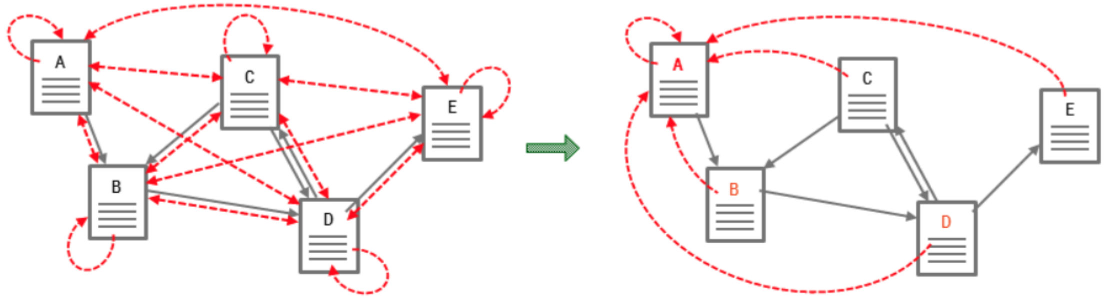
PageRank Formulation: Personalized PageRank

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u)$$

- How to modify **E(u)** is up to the designer
  - In the default setting, **E(u)** is uniform over all nodes (Left side of the figure)
  - Can modify E(u) to consist entirely of a single node A (Right side of the figure)

    → **In the Personalized PageRank (PPR),**

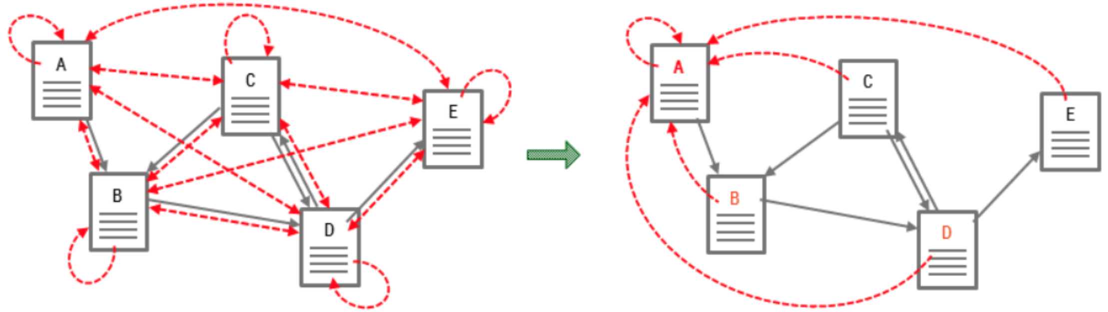
- **Interpretation**
  - The random surfer always restarts at a single node
  - This shifts the ranking from a global perspective to a local one, anchored in that node
  - The importance of other nodes is determined by their proximity and connectivity to that node
  - The final value of PPR becomes the relevance score with respect to that node



PageRank Formulation: PPR (cont.)

Applications

- ❑ Local Graph Clustering
  - Starting from a single node, the PPR vector naturally reveals its community
  - This happens because the random walk remains localized due to the fixed restart point
  - In a social network, starting from a person, PPR highlights their friend group
  - In a knowledge graph, starting from an entity, PPR uncovers semantically related concepts
- ❑ Personalized Search and Recommendation
  - PPR ranks nodes based on their proximity to a user's context



PageRank Formulation: Modern Form

- The Original Formulation of PageRank

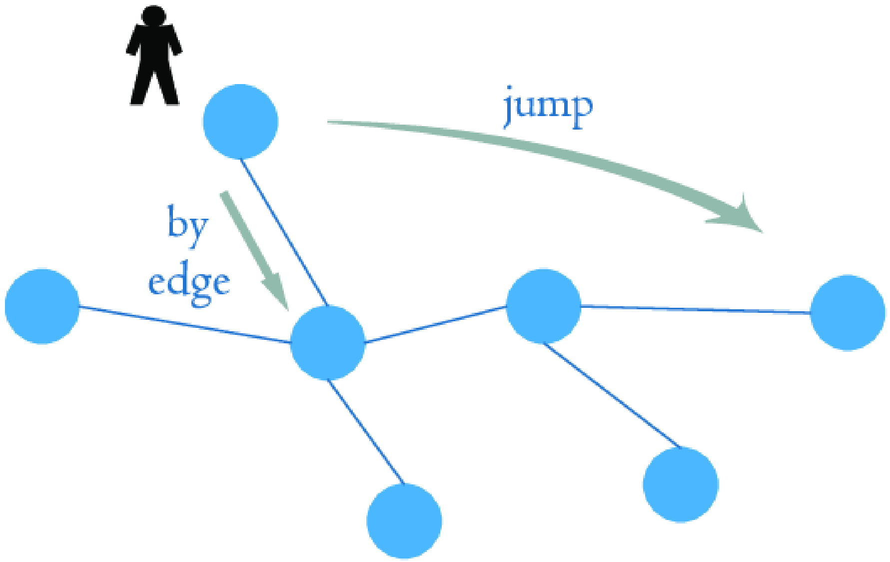
$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u)$$

  - Required manual normalization after iteration to maintain $||R||_1 = 1$
  - Update the importance value individually (Lacks global structural insight)

- The Modern Formulation of PageRank

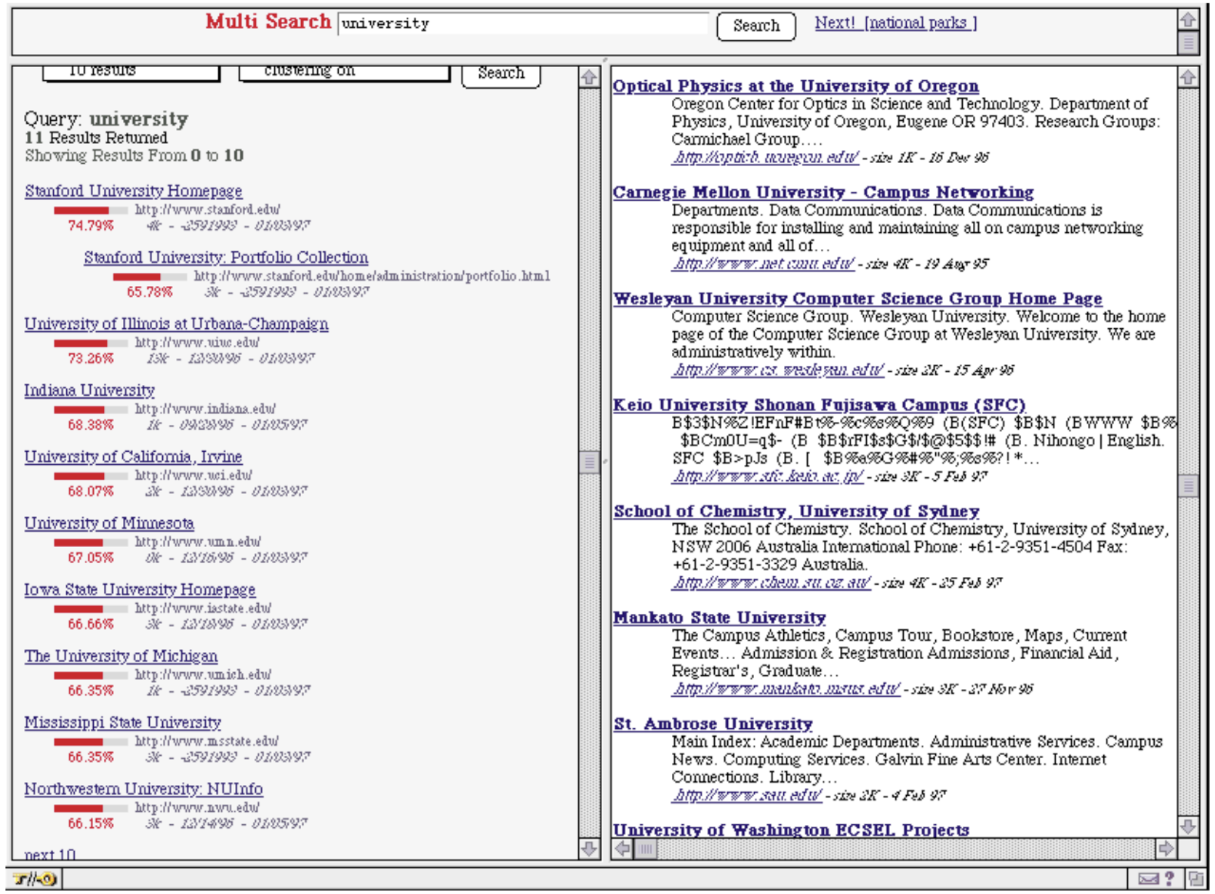
$$\boldsymbol{R} = (1 - c)\boldsymbol{E} + c\boldsymbol{A}\boldsymbol{R}$$

  - Clean probabilistic interpretation
    - random walk with prob ***c*** + teleport with prob ***1-c***
  - Ensures at $||R||_1 = 1$ every iteration
  - Enables efficient computation via power iteration and better aligns



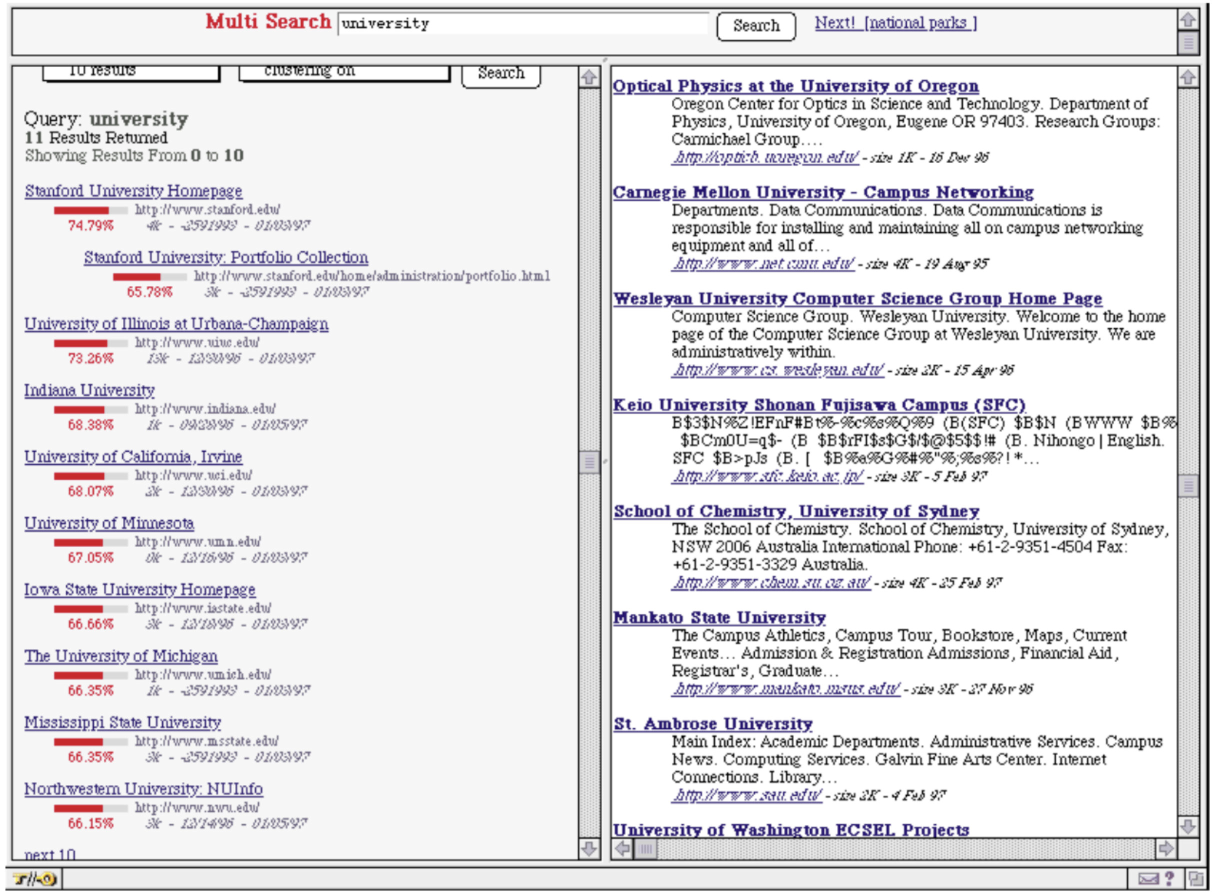
1. Topic-sensitive pagerank. TH Haveliwala. WWW 02
2. Deeper inside pagerank. AN Langville, CD Meyer. Internet Mathematics '04

Applications: Search Engine



- ❑ Traditional Search Engine such as Altavista returns random looking web pages that match the query
  - Altavista seems using URL length as a quality heuristic

Applications: Search Engine (cont.)



- However, by leveraging the PageRank
  - Universities we consider important are actually ranked at the top

Applications: Search Engine (cont.)

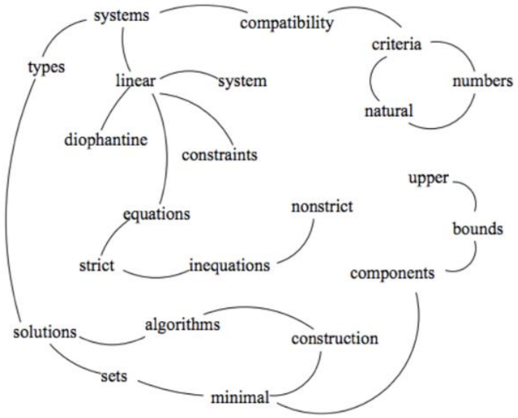
- PageRank have strong advantage in Common Case
  - Identifies globally important pages
    - Provided a global importance signal, resistant to keyword spamming
    - Prioritizes pages that are frequently cited or linked to by others
    - Reflects user preferences as a natural signal of importance
  - In the other hands, this might have disadvantage of precision perspective
    - PageRank provides query-independent global importance in a way
    - Google supplements this aspect by combining keyword search with “keyword”
    - Google utilizes a number of factors to rank search results
      (Including standard IR measures, proximity, anchor text, and PageRank)

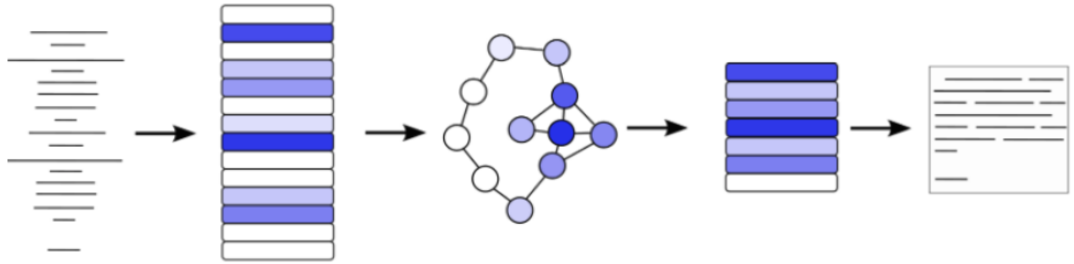
Applications: Other Examples

- **PageRank** captures **the structural importance** of nodes by analyzing the overall link structure of the **graph**
- Therefore this method could be applied…
  - TextRank
  - Biological Network Analysis
  - Influence Detection in Citation Graphs
  - Product/Content Recommendation

Applications: Other Examples

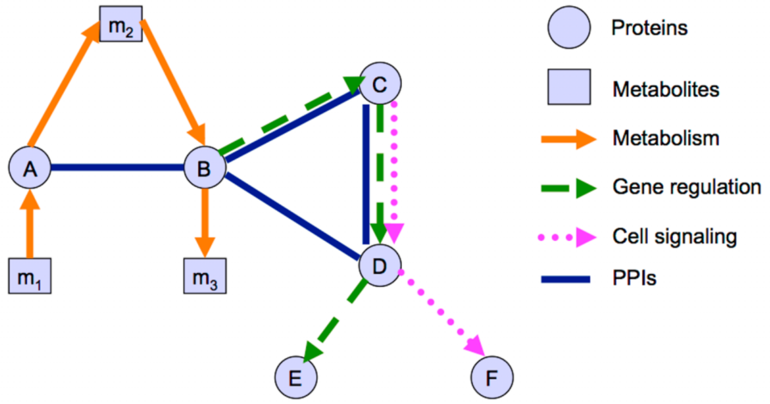
- **PageRank** captures **the structural importance** of nodes by analyzing the overall link structure of the **graph**
- Therefore this method could be applied…
  - **TextRank**
    - Used in keyword or sentence extraction and summarization by modeling sentences or words as nodes
  - Biological Network Analysis
  - Influence Detection in Citation Graphs
  - Product/Content Recommendation





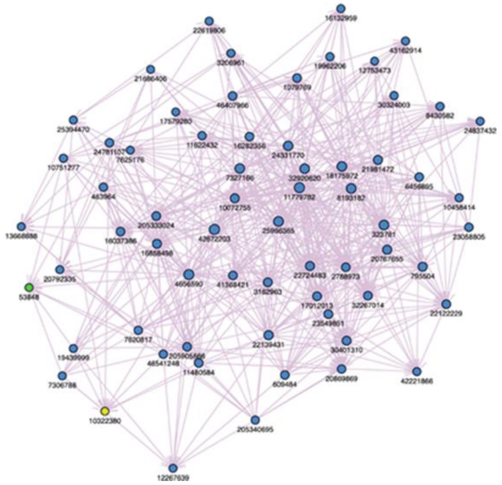
Applications: Other Examples

- **PageRank** captures **the structural importance** of nodes by analyzing the overall link structure of the **graph**
- Therefore this method could be applied…
  - TextRank
  - **Biological Network Analysis**
    - Identify essential proteins or genes by analyzing the structure of protein–protein interaction
  - Influence Detection in Citation Graphs
  - Product/Content Recommendation



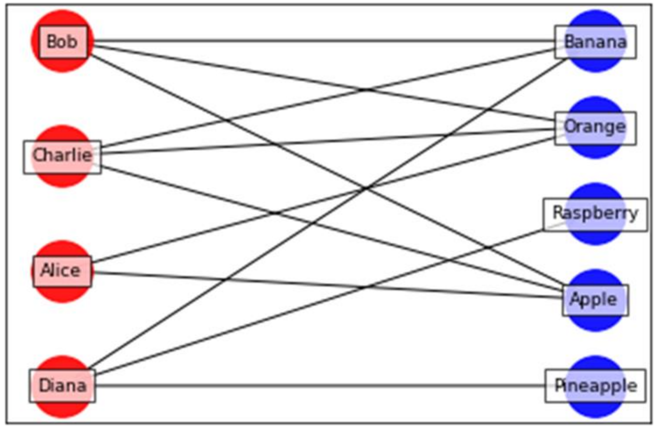
Applications: Other Examples

- **PageRank** captures **the structural importance** of nodes by analyzing the overall link structure of the **graph**
- Therefore this method could be applied…
  - TextRank
  - Biological Network Analysis
  - **Influence Detection in Citation Graphs**
    - Helps measure the impact of scientific papers or authors by modeling citations as directed edges
  - Product/Content Recommendation



Applications: Other Examples

- **PageRank** captures **the structural importance** of nodes by analyzing the overall link structure of the **graph**
- Therefore this method could be applied…
  - TextRank
  - Biological Network Analysis
  - Influence Detection in Citation Graphs
  - **Product/Content Recommendation**
    - Ranks items based on their connections in user–item interaction graphs



Discussion & Implication

❑ Structural Node Importance

- Provides a way to quantify **the structural importance** of nodes based on link topology
- Eigenvector-based **Centrality**
  - Acts as a powerful centrality measure that recursively reflects the influence of neighboring nodes

❑ Domain-Agnostic Extensibility

- Can be adapted to any domain where relationships can be represented as a graph, without relying on domain-specific content
- Successfully applied to citation networks, biological systems, NLP, and recommender systems

Fast random walk with restart and its applications

ICDM(2006)

H Tong, C Faloutsos, JY Pan

CAU
Junseo, Yu

DMAIS Lab Seminar
07.17.2025

Content

- **Introduction**
  - Background of Random Walk with Restart
  - Why RWR is Computationally Challenging
  - Two Key Properties of Real-World Graphs
- **Proposed Methods**
  - B_LIN
  - Time/Space Complexity of B_LIN
- **Experiments & Results**
- **Conclusion**

Background of Random Walk with Restart

- The Rising Importance of defining the relevance score between two nodes
  - One very successful technique is based on **random walk with restart (RWR)** → **Personalized PageRank**
    - $\vec{r}_i = c\tilde{\mathbf{W}}\vec{r}_i + (1-c)\vec{e}_i$
    - The PPR Value of **node j** (probability of the random suffer remaining node j) reflects its relevance to **node i**
- The RWR require tailored vector for specific queries (nodes)
  - The typical PageRank not personalized one only require one static vector invariance with specific queries
  - RWR need bunch of calculations
  - Therefore, the time/speed complexity become important problem

'Jet' 'Plane' 'Runway'

'Texture' 'Candy' 'Background'

Why RWR is Computationally Challenging

Need of Inversion Matrix

- $\mathbf{r}_i = c \cdot \tilde{W} \cdot \mathbf{r}_i + (1-c) \cdot \mathbf{e}_i$
- $\mathbf{r}_i - c \cdot \tilde{W} \cdot \mathbf{r}_i = (1-c) \cdot \mathbf{e}_i$
- $(I - c \cdot \tilde{W}) \cdot \mathbf{r}_i = (1-c) \cdot \mathbf{e}_i$
- $\mathbf{r}_i = (1-c) \cdot (I - c \cdot \tilde{W})^{-1} \cdot \mathbf{e}_i$
- $Q^{-1} = (I - c \cdot \tilde{W})^{-1}$
- Computation of **inversion Q** must be performed **for every query node *i***
  → query-dependent calculation

$\tilde{W}$: the normalized weighted matrix associated with W

$\mathbf{e}_i$: n × 1 starting vector, the i-th element 1 and 0 for other

Existing Method

- **OnTheFly:** Compute the rank vector **on the fly (instantly)** by power iteration
- **PreCompute: Pre-compute and store** the inversion of a matrix

Why RWR is Computationally Challenging

❑ OnTheFly

- Don't store or precompute Q^{-1}
- Solve the linear system iteratively on the fly (instantly) in query-time

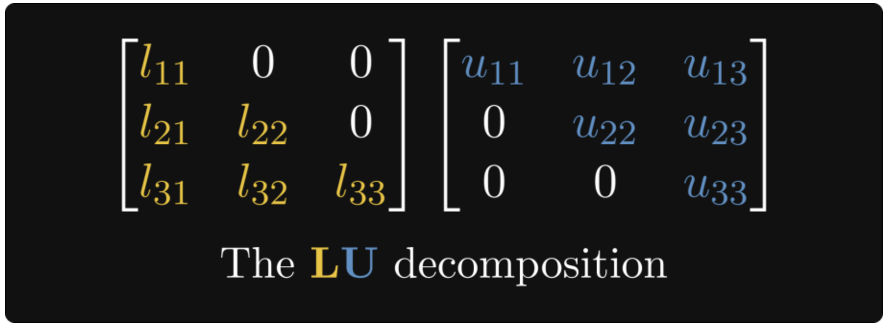
$$\vec{r}_i = c\widetilde{W}\vec{r}_i + (1-c)\vec{e}_i$$

- However the space complexity become **O(T • m)**
  - Why?
    - At query time, need m iterations: O(T)
    - Each iteration involves sparse matrix-vector multiplication: Cost per iteration = O(T)
      (T = number of non-zero elements in $\tilde{\mathbf{W}}$)

❑ PreCompute

Why RWR is Computationally Challenging

- ❑ **OnTheFly**
- ❑ **PreCompute**
  - Pre-compute and store the $Q^{-1} = (I - c\widetilde{W})^{-1}$
  - Can get result in real-time $\vec{r}_i = (1-c)Q^{-1}\vec{e}_i$
  - However the time/space compexitiy become O(n^3) /O(n^2)
    - Why **O(n^3)** in time complexity?
      - Inverting an n×n matrix (Q) requires **cubic** time using:
        - Gaussian Elimination
        - LU Decomposition
        - Matrix inversion algorithms
    - Why **O(n^2)** in space complexity?
      - Q^{-1} is a dense **n x n matrix**



Two Key Properties of Real-World Graphs

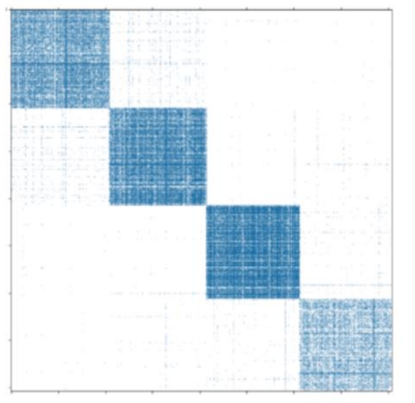
- Block-wise / Community Structure
- Linear Correlation (Low-Rank Approximation)

Two Key Properties of Real-World Graphs

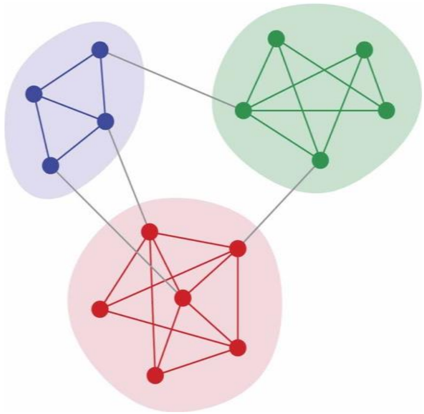
❑ Block-wise / Community Structure

- In many real-world graphs, nodes are not connected uniformly
- Nodes tend to form tightly connected groups, with sparse connections across groups ***(Homophily)***
- Can calculate block-by-block while still preserving most of the relevance information
- Means the local/fine resolution estimation (within each group)

$$\tilde{W} = \begin{bmatrix} \text{dense} & \text{sparse} \\ \text{sparse} & \text{dense} \end{bmatrix}$$



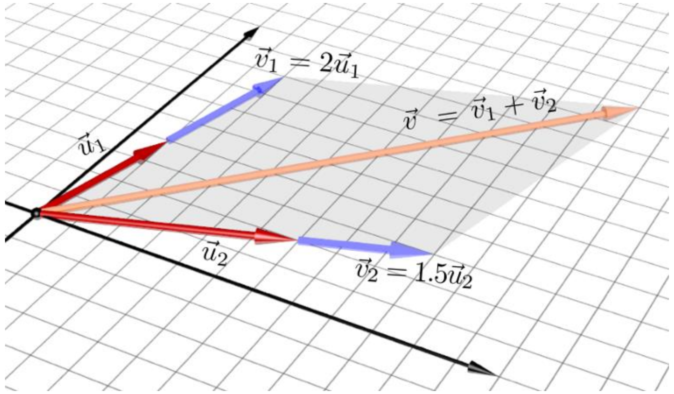
$$\tilde{\mathbf{W}}_1 = \begin{pmatrix} \tilde{\mathbf{W}}_{1,1} & 0 & \dots & 0 \\ 0 & \tilde{\mathbf{W}}_{1,2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \tilde{\mathbf{W}}_{1,k} \end{pmatrix}$$



❑ Linear Correlation (Low-Rank Approximation)

Two Key Properties of Real-World Graphs (cont.)

- ❑ **Block-wise / Community Structure**
- ❑ **Linear Correlation (Low-Rank Approximation)**
  - In real-world graphs,
    - Nodes with shared neighbors often exhibit similar neighborhood patterns
    - Example: friends in the same community or authors in the same research area
  - This creates linear correlation among nodes
    - Linear correlation: one vector is a linear combination of another

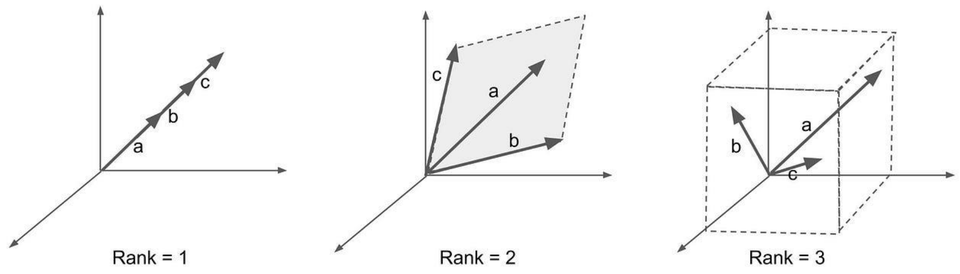


Linear combination

Two Key Properties of Real-World Graphs (cont.)

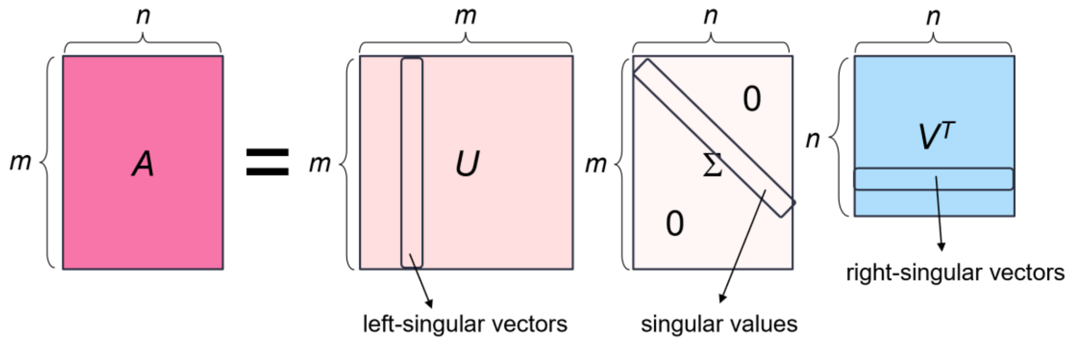
- **Block-wise / Community Structure**
- **Linear Correlation (Low-Rank Approximation)**
  - Linear Correlation leads the matrix to **low rank matrix**
    - **The rank of a matrix**: The number of linearly independent rows/columns
  - Low rank → redundant or correlated structure

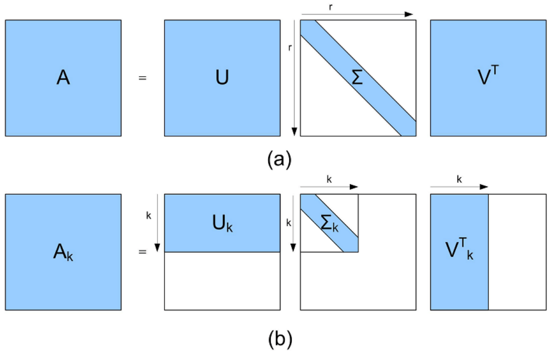
$$A = \begin{bmatrix} a & b & c \end{bmatrix}$$



Two Key Properties of Real-World Graphs (cont.)

- ❑ **Block-wise / Community Structure**
- ❑ **Linear Correlation (Low-Rank Approximation)**
  - Use Singular Value Decomposition (SVD):
    - $A = U\Sigma V^T$
  - By keeping only the **top r** ≪ n (The number of rank) singular values:
    - We get a compact, approximate version of the graph structure
    - Reduces computation and storage significantly
    - **Efficient approximation** of **Q^{-1}**





Proposed Method: B_LIN Overview

- **Core Idea**: **Decompose the graph and reuse the structure**
  - **Split** the graph into **intra-community** and **cross-community** links
  - **Precompute** inside partitions, and approximate across partitions
- **Pre-computation Stage (Off-line)**
  - **Partition** graph into k blocks (**intra-community**)
  - For each partition ***i***, **compute** and **store** inverse of matrix
  - Do row-rank **approximation** of **cross-community**
  - Calculate and Store $\boldsymbol{Q_1^{-1}}$ cross-community influence $\boldsymbol{\tilde{\Lambda}}$
- **Query Stage (On-line)**
  - For any query node ***i***, calculate the results with in only a few matrix-vector multiplications

Proposed Method: B_LIN Detail (cont.)

Pre-computation Stage (Off-line)

- Partition graph into k blocks
- Decompose matrix: $\tilde{\mathbf{W}}_1 + \tilde{\mathbf{W}}_2$
  - $\tilde{\mathbf{W}}_1$: Block-diagonal submatrix for local computations
  - $\tilde{\mathbf{W}}_2$: Off-diagonal correction matrix for cross-block influence
- For each partition i, compute and store inverse of matrix:
  - $Q_{1,i}^{-1} = (\mathbf{I} - c\tilde{\mathbf{W}}_{1,i})^{-1}$
- Calculate and Store $\mathbf{Q}_1^{-1}$ cross-community influence $\tilde{\mathbf{\Lambda}}$

$$\tilde{\mathbf{W}}_1 = \begin{pmatrix} \tilde{\mathbf{W}}_{1,1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{W}}_{1,2} & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots \\ \mathbf{0} & \dots & \mathbf{0} & \tilde{\mathbf{W}}_{1,k} \end{pmatrix}$$

$$\mathbf{Q}_1^{-1} = \begin{pmatrix} \mathbf{Q}_{1,1}^{-1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{1,2}^{-1} & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{Q}_{1,k}^{-1} \end{pmatrix}$$

(a) An initial graph (b) A partitioning result

Proposed Method: B_LIN Detail (cont.)

- Calculate and Store $\mathbf{Q}_1^{-1}$ cross-community influence $\tilde{\boldsymbol{\Lambda}}$

- For Calculating $\mathbf{r}_i$

$$\mathbf{r}_i = c \cdot \tilde{W} \cdot \mathbf{r}_i + (1-c) \cdot \mathbf{e}_i$$

$$\mathbf{r}_i - c \cdot \tilde{W} \cdot \mathbf{r}_i = (1-c) \cdot \mathbf{e}_i$$

$$(I - c \cdot \tilde{W}) \cdot \mathbf{r}_i = (1-c) \cdot \mathbf{e}_i$$

$$\mathbf{r}_i = (1-c) \cdot (I - c \cdot \tilde{W})^{-1} \cdot \mathbf{e}_i$$

$$Q^{-1} = (I - c \cdot \tilde{W})^{-1}$$

- After Decompostion, We can write the **Q** as

$$Q = I - c\widetilde{W} = I - c(\widetilde{W}_1 + \widetilde{W}_2) = Q_1 - c\widetilde{W}_2.$$

$$\widetilde{W}_2 \approx U\,S\,V \quad \Longrightarrow \quad c\widetilde{W}_2 \approx U\,(cS)\,V.$$

$$Q = Q_1 - U\,(cS)\,V.$$

- The Woodbury matrix identity

$$(A + U\,C\,V)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + V\,A^{-1}U\right)^{-1}V\,A^{-1}$$

- Therefore, we can do inversion of **Q**

$$(Q_1 - UCV)^{-1} = Q_1^{-1} + Q_1^{-1}U\left(C^{-1} - V\,Q_1^{-1}U\right)^{-1}V\,Q_1^{-1}$$

$$C^{-1} = (cS)^{-1} = \tfrac{1}{c}\,S^{-1}.$$

$$\left(C^{-1} - V\,Q_1^{-1}U\right)^{-1} = c\left(S^{-1} - c\,V\,Q_1^{-1}U\right)^{-1} = c\widetilde{\Lambda},$$

- The final form of **Q** is

$$\begin{aligned} Q^{-1} &= (Q_1 - U(cS)V)^{-1} \\ &= Q_1^{-1} + Q_1^{-1}U\,(c\widetilde{\Lambda})\,V\,Q_1^{-1} \\ &= Q_1^{-1} + c\,Q_1^{-1}U\,\widetilde{\Lambda}\,V\,Q_1^{-1}. \end{aligned}$$

- In the query-time, we could leverage the pre-compute results

$$\vec{r_i} = (1-c)(\mathbf{Q}_1^{-1}\vec{e_i} + c\mathbf{Q}_1^{-1}\mathbf{U}\tilde{\boldsymbol{\Lambda}}\mathbf{V}\mathbf{Q}_1^{-1}\vec{e_i}).$$

Proposed Method: NB_LIN

- The number of partitions k determines the trade-off between local and global computation
- k = 1 → PreCompute
  - The entire graph is treated as a single partition
    - $\mathbf{\tilde{W}_1} = \mathbf{\tilde{W}}$ & $\mathbf{\tilde{W}_2} = \mathbf{0}$
- k = n → NB_LIN (Simplified B_LIN)
  - Each node is its own partition → no within-partition links
    - $\mathbf{\tilde{W}_1} = \mathbf{0}$ & $\mathbf{\tilde{W}_2} = \mathbf{\tilde{W}}$ & $\mathbf{Q_1} = \mathbf{I}$
  - Off-line (Preprocessing):
    - Compute low-rank approximation: $\mathbf{\tilde{W}} \approx \mathbf{U\ S\ V}$
    - Compute correction matrix: $\mathbf{\tilde{\Lambda}} = (\mathbf{S}^{-1} - c\mathbf{VU})^{-1}$
  - On-line (Query):
    - For any query node i, compute: $(1-c)(\vec{e_i} + c\mathbf{U}\tilde{\mathbf{\Lambda}}\mathbf{V}\vec{e_i}).$
  - Implications
    - NB_LIN trades local precision for global speed.
    - Great when speed and memory are the top priorities.

Time/Space Complexity of B_LIN

- **On-line computational cost**
  - only need a few matrix-vector multiplication operations

$$
\begin{aligned}
\vec{r}_0 &\leftarrow \mathbf{Q}_1^{-1}\vec{e}_i \\
\vec{r}_i &\leftarrow \mathbf{V}\vec{r}_0 \\
\vec{r}_i &\leftarrow \tilde{\mathbf{\Lambda}}\vec{r}_i \\
\vec{r}_i &\leftarrow \mathbf{U}\vec{r}_i \\
\vec{r}_i &\leftarrow \mathbf{Q}_1^{-1}\vec{r}_i \\
\vec{r}_i &\leftarrow (1-c)(\vec{r}_0 + c\vec{r}_i)
\end{aligned}
$$

$$\vec{r}_i = (1-c)(\mathbf{Q}_1^{-1}\vec{e}_i + c\mathbf{Q}_1^{-1}\mathbf{U}\tilde{\mathbf{\Lambda}}\mathbf{V}\mathbf{Q}_1^{-1}\vec{e}_i).$$

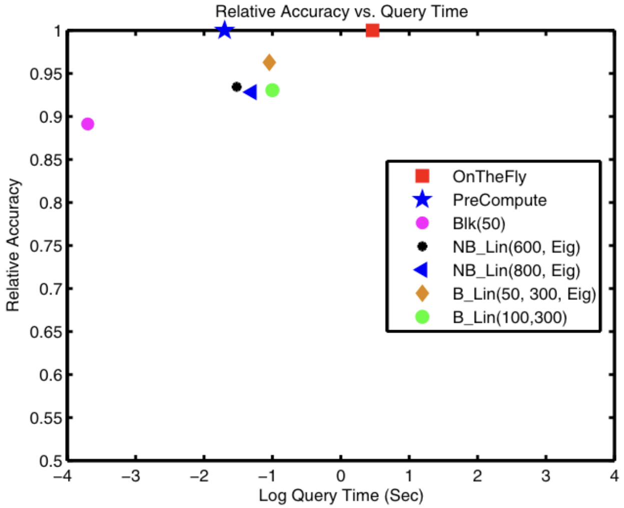
Time/Space Complexity of B_LIN (cont.)

- **Pre-computational cost**
  - Instead of computing the inverse of a full graph *n x n* matrix
  - B_LIN dramatically reduces computation
  - The main steps that require calculations
    - Inverse of each k small matrices
    - Low-rank Approximation
    - Inversion of $\tilde{\boldsymbol{\Lambda}}$
- **Pre-storage cost**
  - **B_LIN needs to store**
    - **k+1** small inverse matrices including $\tilde{\boldsymbol{\Lambda}}$
    - **One** low-rank matrix **U** of size n x t
    - **One** matrix **V** of size t x n
  - **Optimization**
    - **Sparsification**: Most values in the matrices are very close to zero
    - **Exploiting Symmetry:**
      If the graph is normalized with a symmetric Laplacian, only need to store **half** the values in each matrix

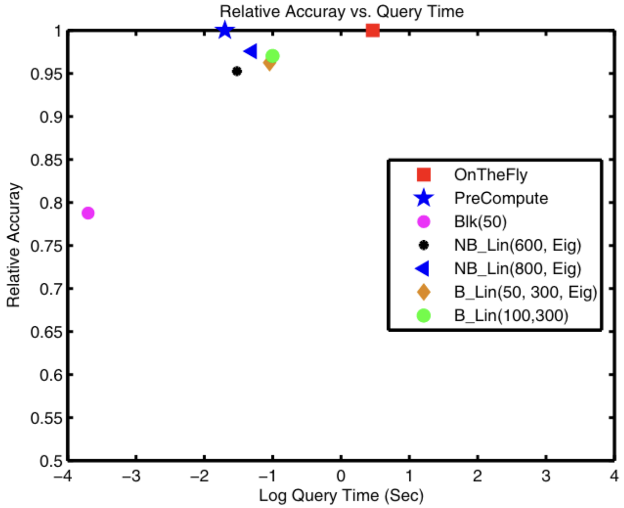
Experiments

- **Datasets Used:**
  - CoIR (5K images, 774K edges) – image retrieval (CBIR)
  - CoMMG (52K nodes) – cross-modal captioning (CMCD)
  - AP (315K nodes, 1.8M edges) – author-paper graph (Ceps, NF)
- **Applications Evaluated:**
  - CBIR = Content-Based Image Retrieval
  - CMCD = Cross-Modal Correlation Discovery
  - CePS = Center-Piece Subgraph Discovery
  - NF = Neighborhood Formulation
- **Evaluation Metrics**
  - Relative Accuracy (RelAcu)
  - Relative Score (RelScore)
  - Efficiency Metrics
    - Query Time (QT)
    - Pre-computational Time (PT)
    - Pre-storage Cost (PS)

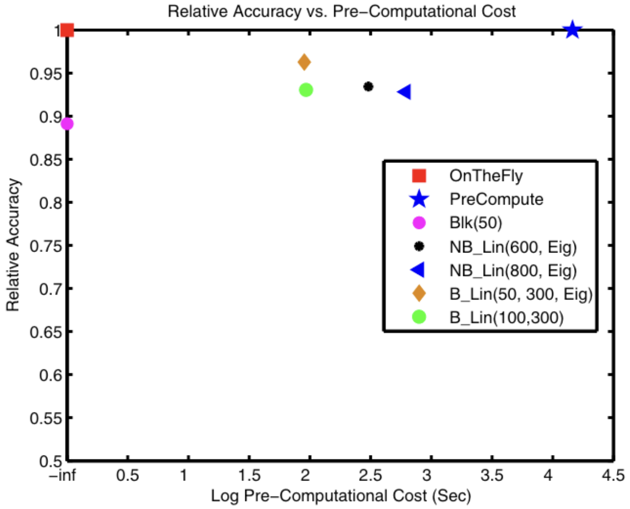
Experiments: CoIR



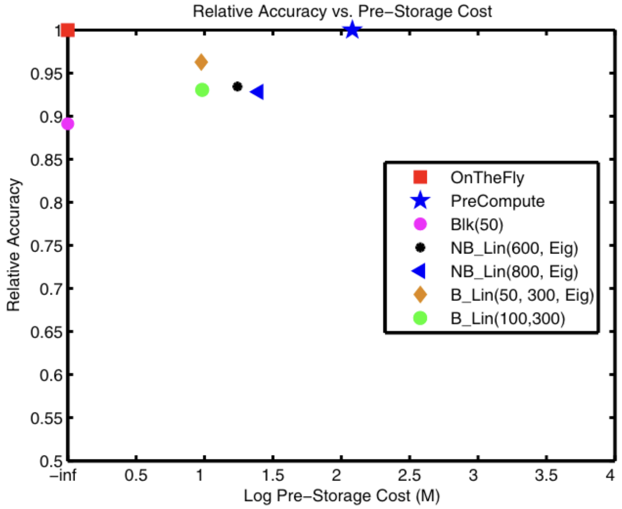
(a) Accuracy (Initial) vs. Log QT



(b) Accuracy (RF) vs. Log QT

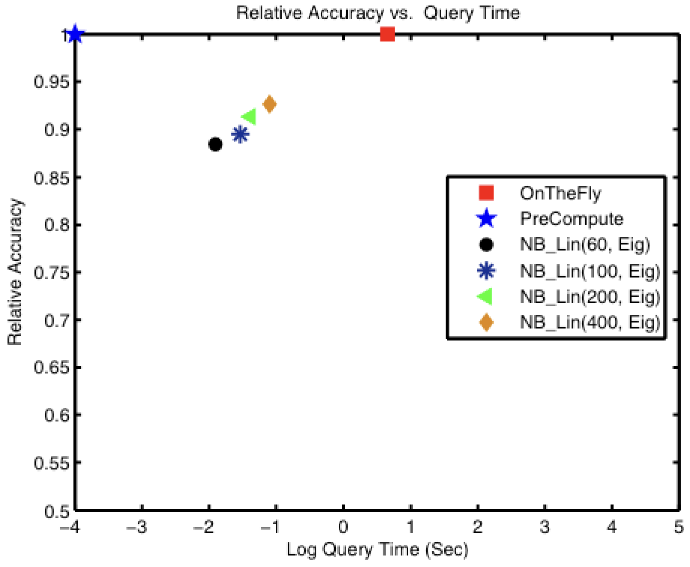


(c)Accuracy (Initial) vs. Log PT

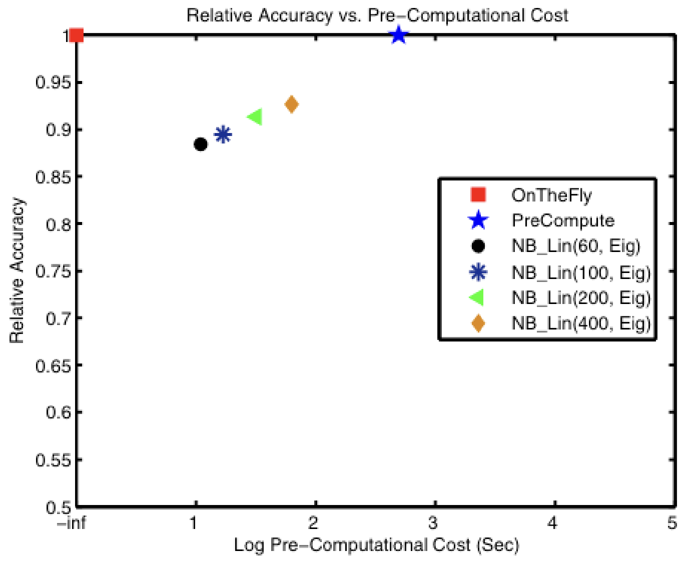


(d) Accuracy (Initial) vs. Log PS

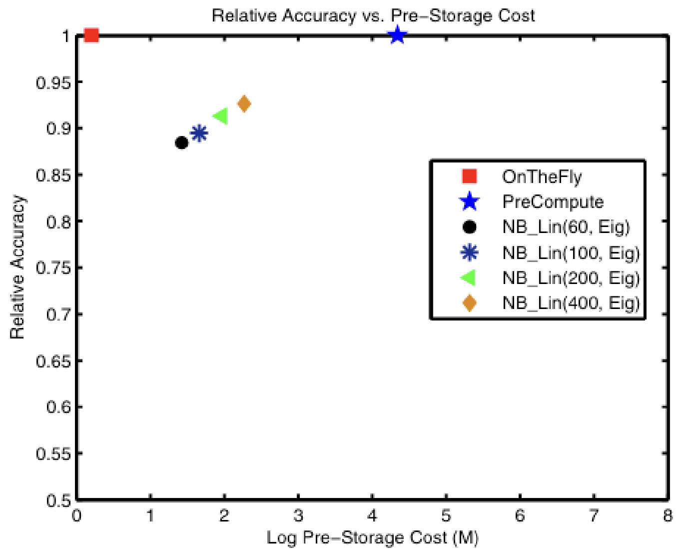
Experiments: COMMG



(a) Accuracy vs. Log QT

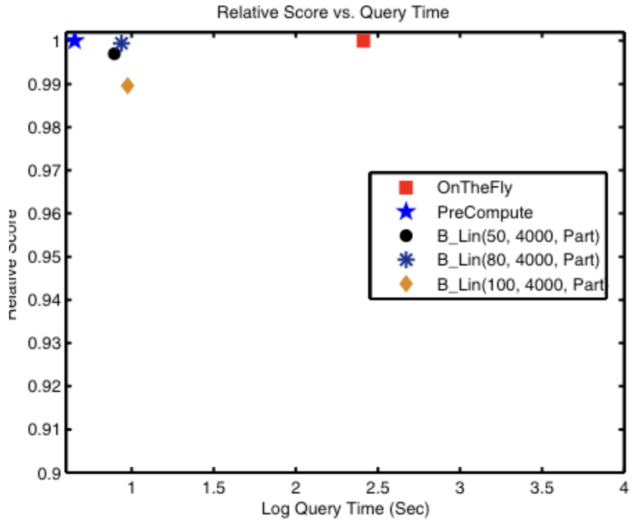


(b) Accuracy vs. Log PT

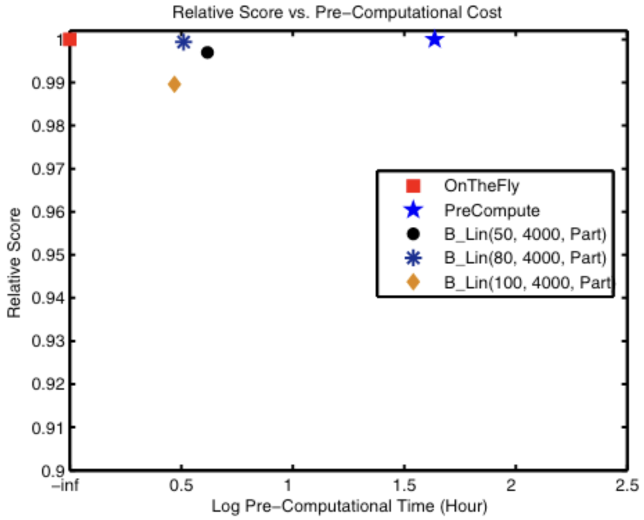


(c) Accuracy vs. Log PS

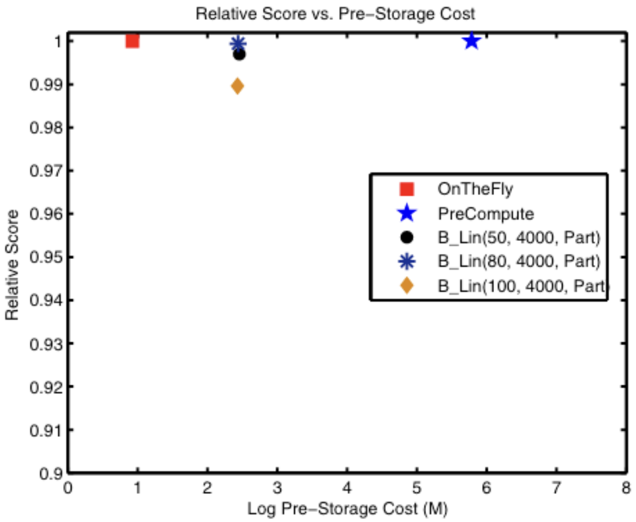
Experiments: CoIR



(a) Accuracy vs. Log QT



(b) Accuracy vs. Log PT



(c) Accuracy vs. Log QS

Discussion

Contributions

- Propose a fast and accurate approximation for Random Walk with Restart (RWR) by exploiting structural properties of real-world graphs

Discussion

- Low-rank approximation works well because nodes share similar connectivity patterns
  - But where does this correlation occur?
  - Likely within communities: Nodes in same group (e.g. same topic, organization) have high redundancy
  - Does it also appear across communities?

My Opinion

- A interesting attempt to translate real-world characteristics into technology