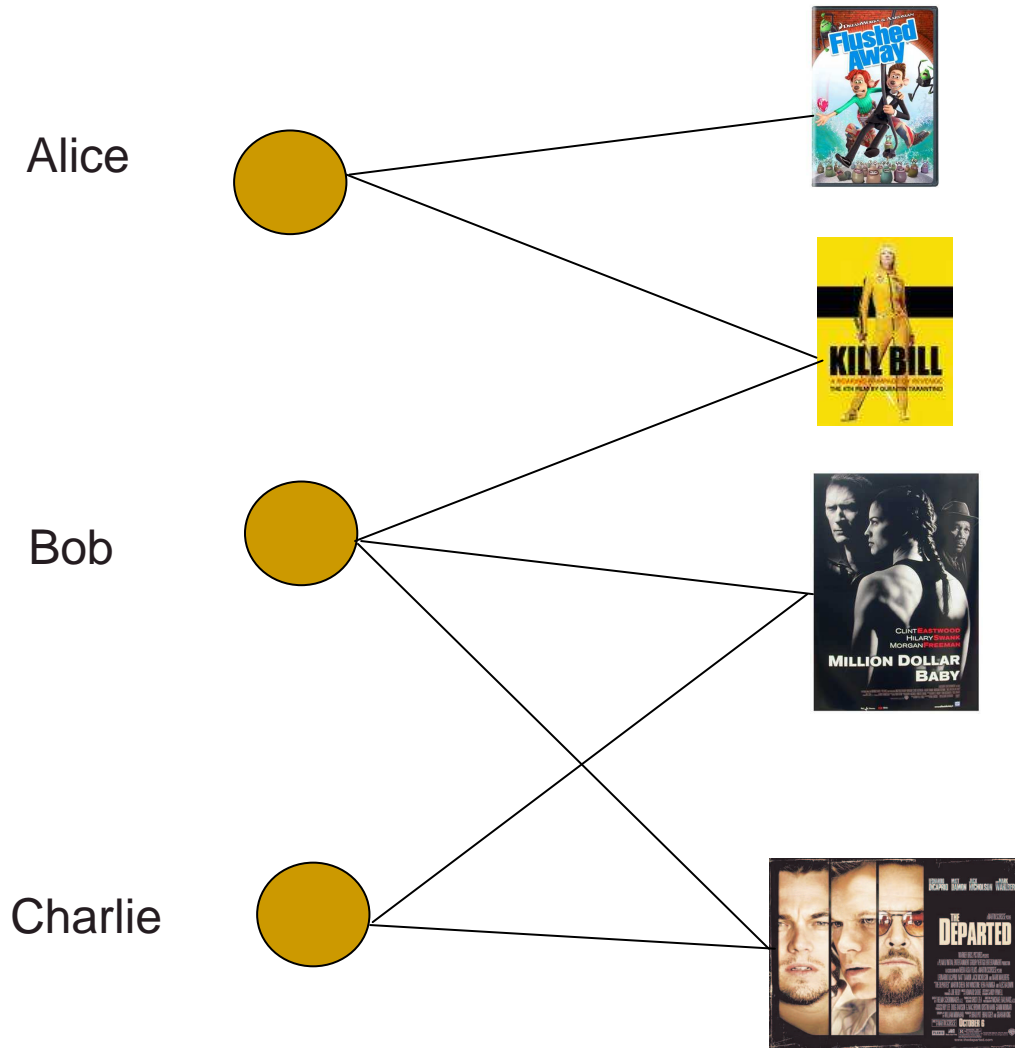

Fast Incremental Proximity Search in Large Graphs

Purnamrita Sarkar

Andrew W. Moore

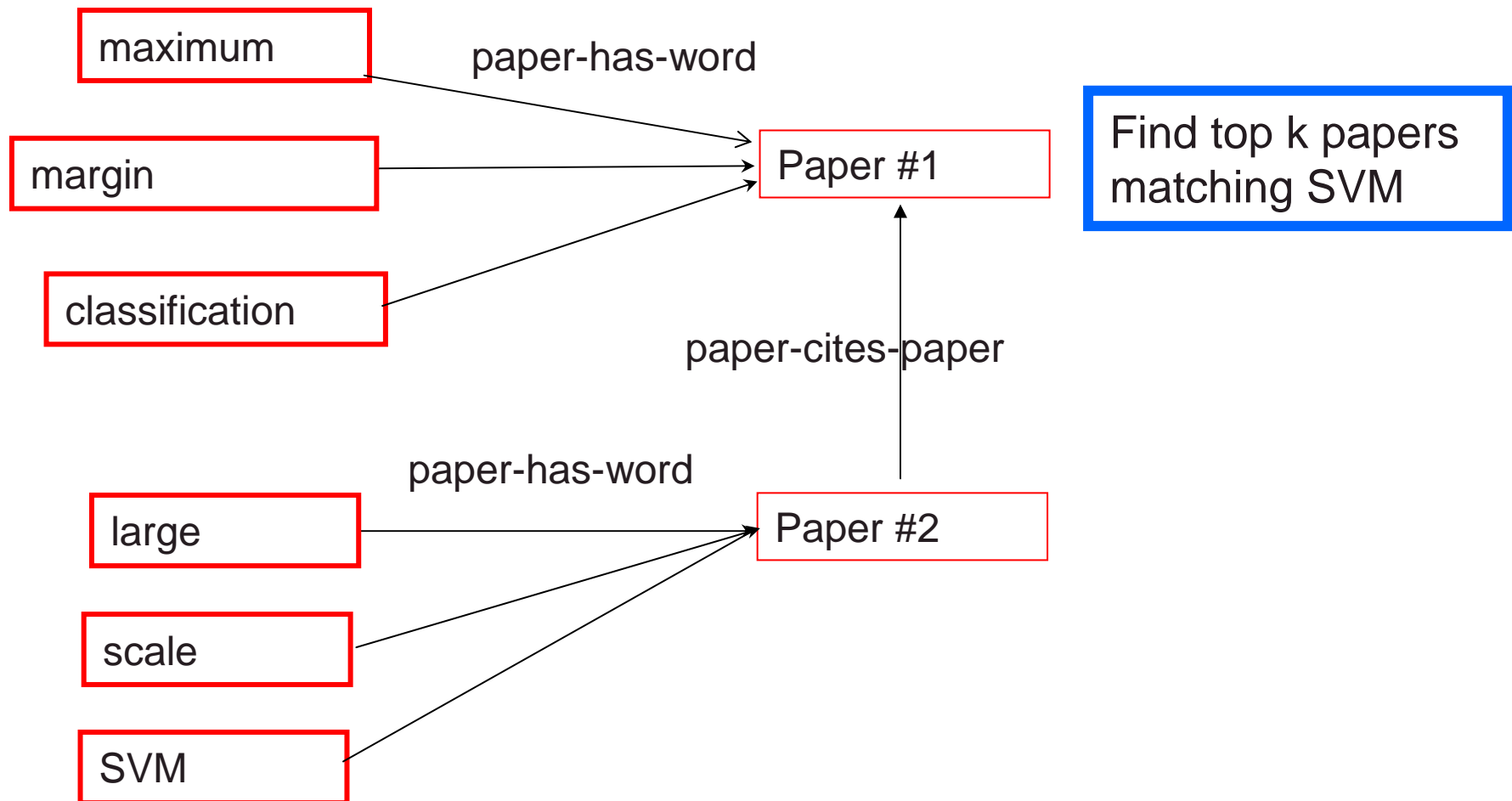
Amit Prakash

Recommender systems



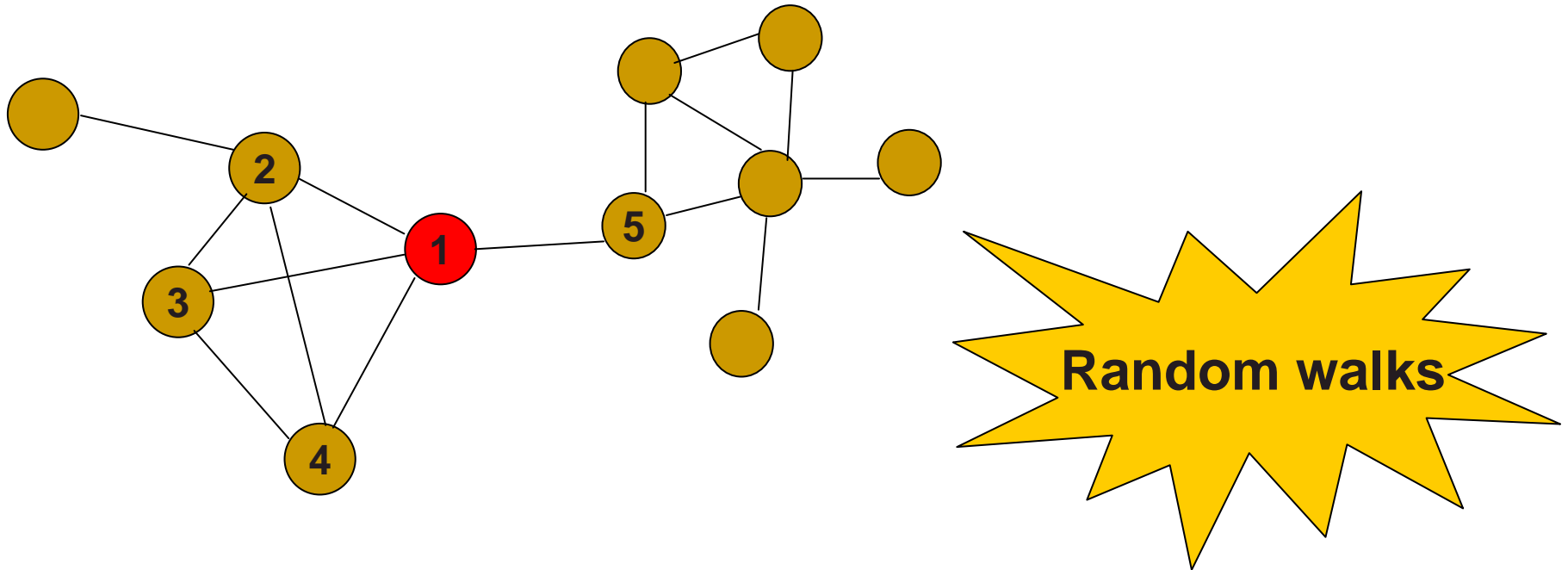
What are the top k movie recommendations for Alice?

Content-based search in databases^{1,2}



1. Dynamic personalized pagerank in entity-relation graphs. (Soumen Chakrabarti, WWW 2007)
2. Balmin, A., Hristidis, V., & Papakonstantinou, Y. (2004). ObjectRank: Authority-based keyword search in databases. *VLDB 2004*.

Proximity measures on a graph

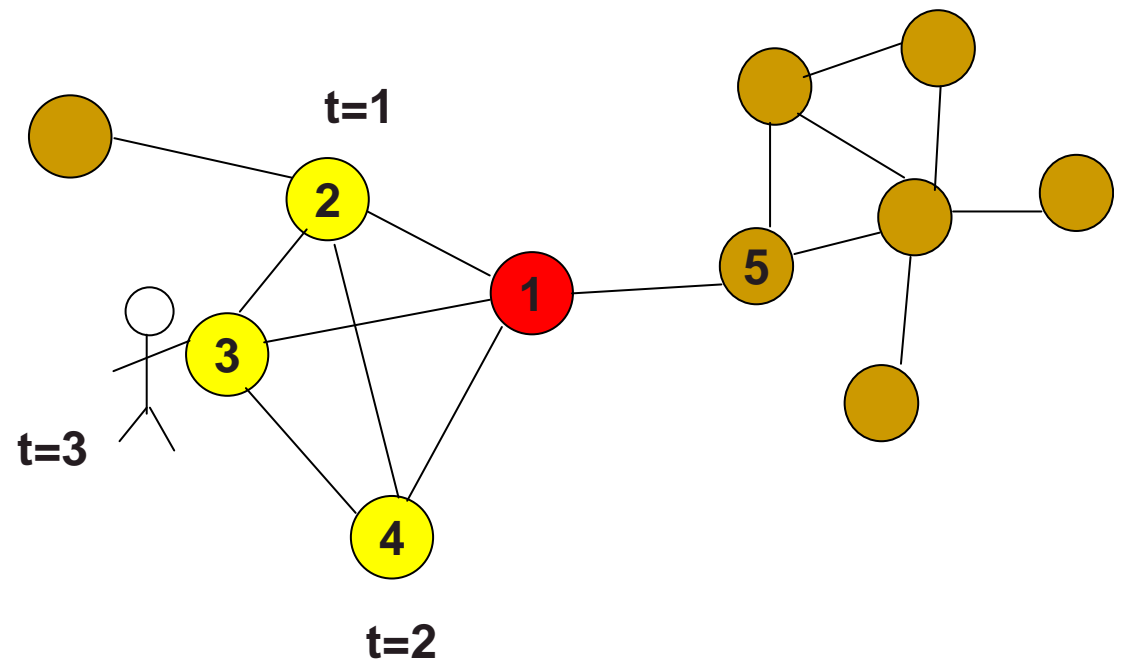


How similar are 1 and 2?

Many short paths from 1 to 2 → highly similar

Random walks

- Start at 1
- Pick a neighbor i uniformly at random
- Move to i
- Continue.



Outline

- Proximity measures on a graph
 - Standard measures: hitting/commute times
 - Improved measures: truncated versions
- Ranking nearby nodes
- Theoretical results
- Algorithm sketch
- Experimental results

Proximity between nodes i and j using random walks

- If we start the random walk at i what is the probability of hitting j .
 - *Personalized pagerank^{1,2,3}*
 - *Fast random walk with restart (Tong, Faloutsos 2006)*
 - *Fast direction-aware proximity for graph mining (Tong et al, 2007)*
- If we start the random walk at i what is the expected time to hit j
 - *Hitting and commute times*

1. *Scaling Personalized Web Search*, Jeh, Widom. 2003.

2. *Topic-sensitive PageRank*, Haveliwala, 2001

3. *Towards scaling fully personalized pagerank*, D. Fogaras and B. Racz, 2004

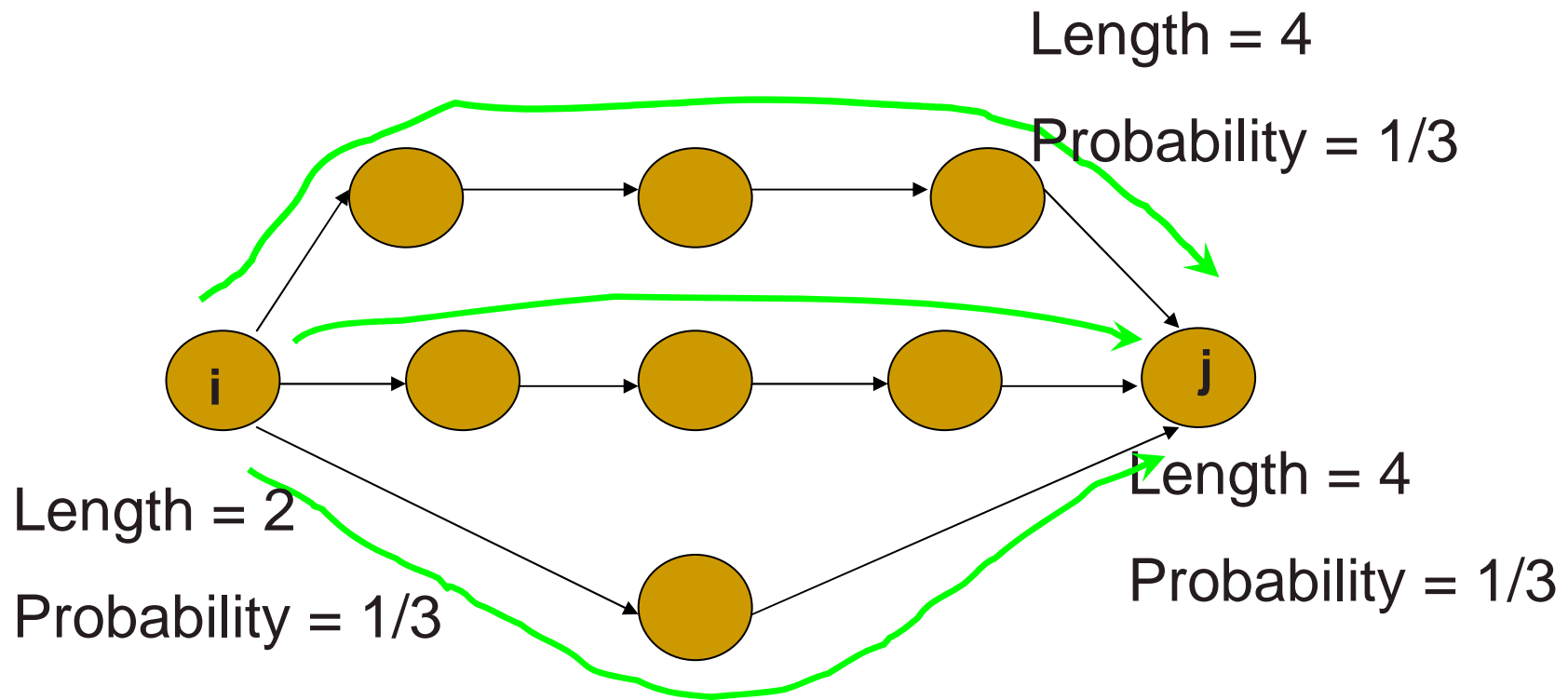
Personalized pagerank

- Personalized pagerank for node i
 - Start from node i
 - At any timestep jump to node i with probability c
 - Stop when the distribution does not change.
 - Solve for v such that

$$v = (1 - c)vP + cr$$

- r is a distribution

Random walk based proximity measures



$$\text{Hitting time } h^T(i,j) = \text{Expected path length} = 4 * \frac{1}{3} + 4 * \frac{1}{3} + 2 * \frac{1}{3} = 3.33$$

Random walk-based proximity measures

- Hitting time can be defined recursively

$$h(i, j) = \begin{cases} 1 + \sum_k P(i, k)h(k, j) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

- Symmetric version: commute time

$$c(i, j) = h(i, j) + h(j, i)$$

Drawbacks of Hitting & Commute times: Predictive power

- Often take into account very long paths, making them sensitive to changes in any part of the graph
- Rankings based on them tend to pick up popular entities
 - ❑ Bad for personalization
 - ❑ e.g. if Alice likes cartoons her top 10 recommendations should not be the 10 most popular movies.
- As a result these do not perform well for link prediction^{1,2}

1. Liben-Nowell, D., & Kleinberg, J. The link prediction problem for social networks CIKM '03.
2. Brand, M. (2005). A Random Walks Perspective on Maximizing Satisfaction and Profit. SIAM '05.

Drawbacks of Hitting & Commute times: Computational complexity

- Recent “efficient” approximation algorithm for computing pair wise commute times in undirected graphs¹.
- However, hard to compute these measures for directed graphs
- For many real world applications:
 - underlying graphs are directed
 - need fast **on the fly** algorithms for computing nearest neighbors of a query node

Outline

- Proximity measures on a graph
 - Standard measures: hitting/commute times
 - Improved measures: truncated versions
- Ranking using truncated measures
- Theoretical results
- Algorithm sketch
- Experimental results

Improved proximity measures based on truncated random walks

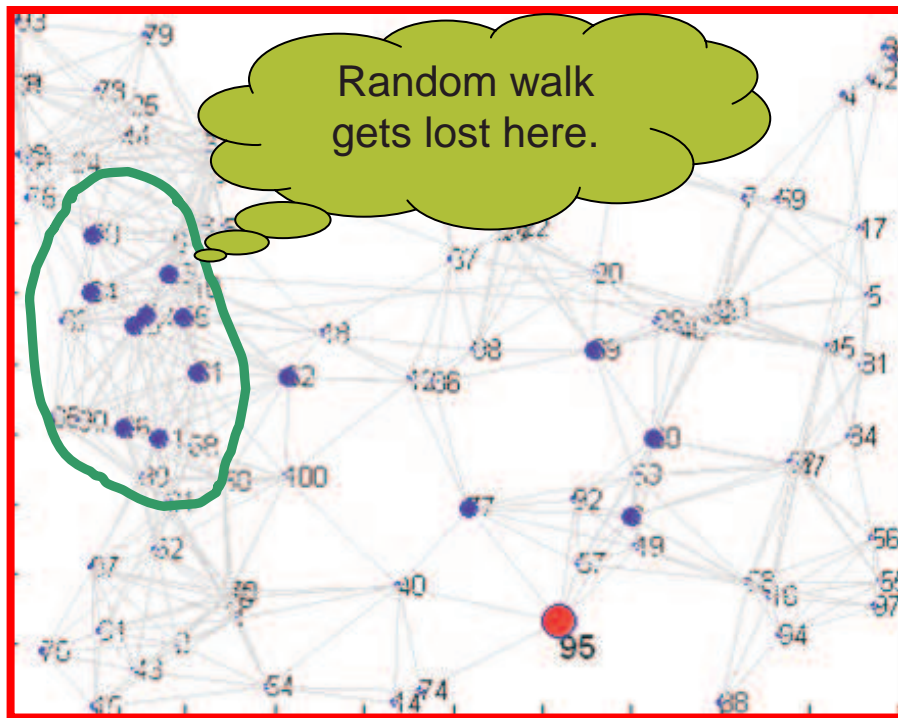
We use a truncated version of hitting and commute times, which only considers paths of length at most T

$$h^T(i, j) = \begin{cases} 1 + \sum_k P(i, k) h^{T-1}(k, j) & \text{if } i \neq j \text{ \& } T > 0 \\ 0 & \text{otherwise} \end{cases}$$

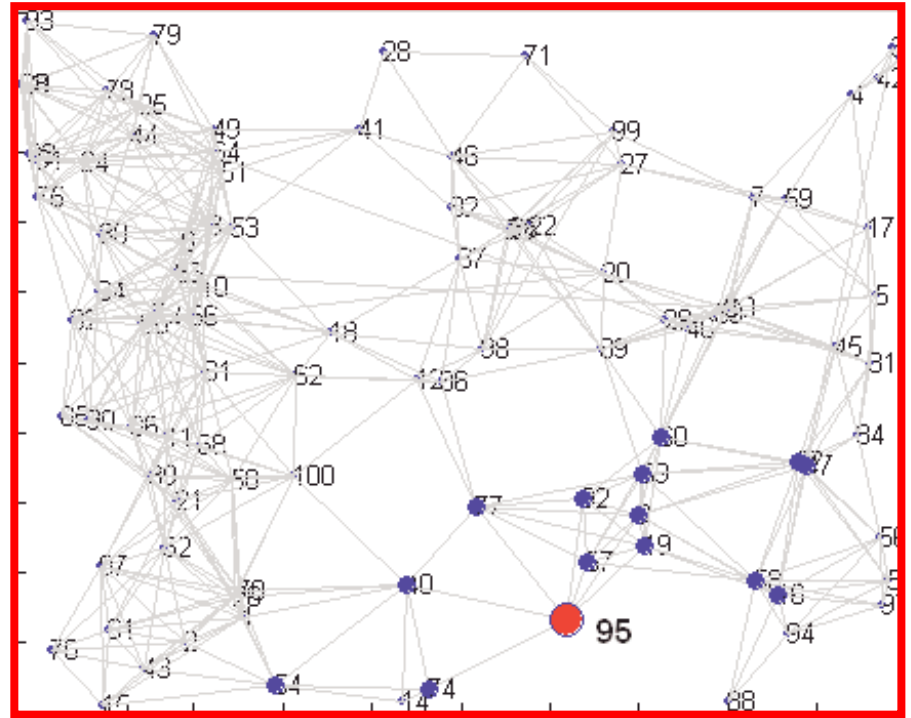
$$c^T(i, j) = h^T(i, j) + h^T(j, i)$$

Exact vs. truncated hitting time from a node

Exact hitting time



Truncated hitting time



15 nearest neighbors of node 95 (in red)

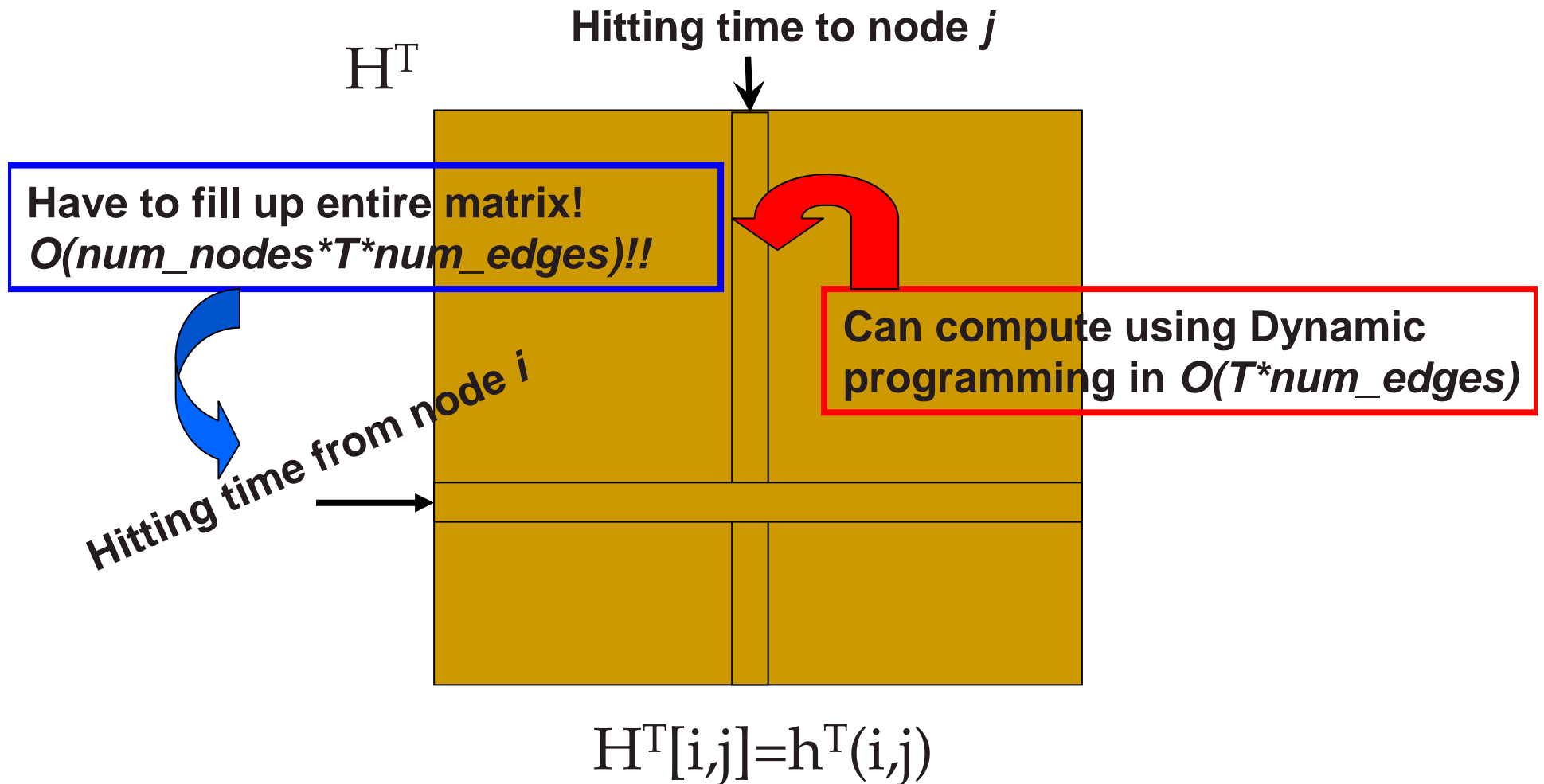
Properties of Truncated Hitting & Commute times

- For small T :
 - Are not sensitive to long paths
 - Do not favor high degree nodes
 - For a randomly generated undirected graph, average correlation coefficient (R_{avg}) with the degree-sequence is
 - R_{avg} with truncated hitting time is -0.087
 - R_{avg} with untruncated hitting time is -0.75



**Important for
personalized search!**

Nearest neighbors of a node: Computational complexity of truncated measures



Our goal : running time $\ll O(\text{num_nodes}^2)$

Outline

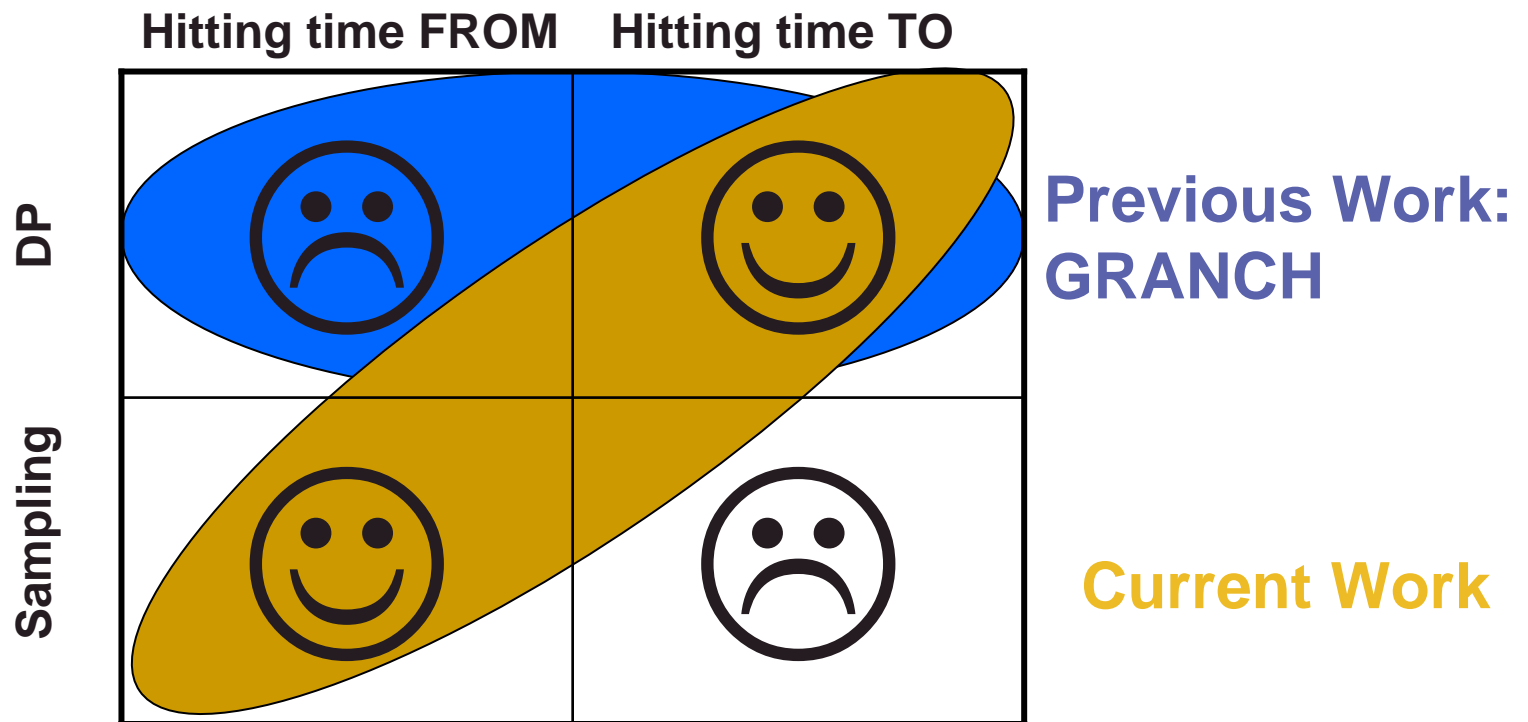
- Proximity measures on a graph
 - Standard measures: hitting/commute times
 - Improved measures: truncated versions

- Ranking using truncated measures

- Theoretical results
- Algorithm sketch
- Experimental results

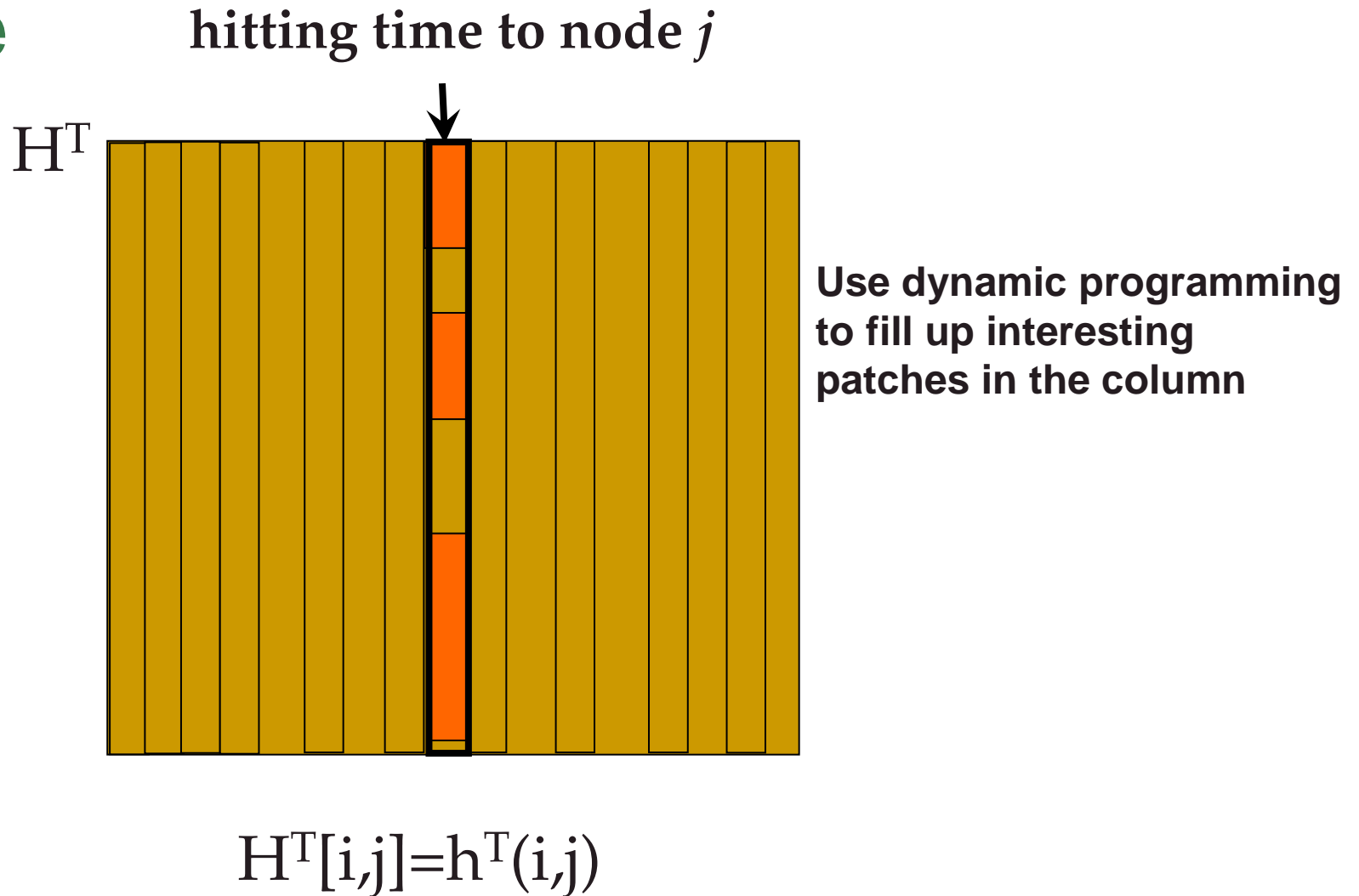
Problem

Return k approximate nearest neighbors of the query node in truncated commute time without looking at entire graph.



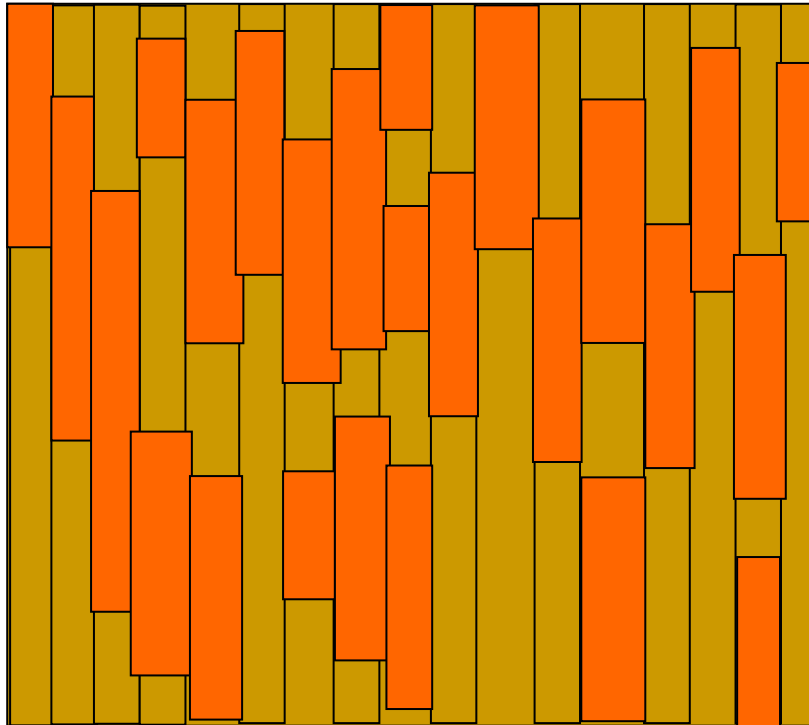
Sarkar, P., & Moore, A. (2007). A tractable approach to finding closest truncated-commute-time neighbors in large graphs. Proc. UAI.

GRANCH: computing hitting time to a node



GRANCH: computing hitting time from a node

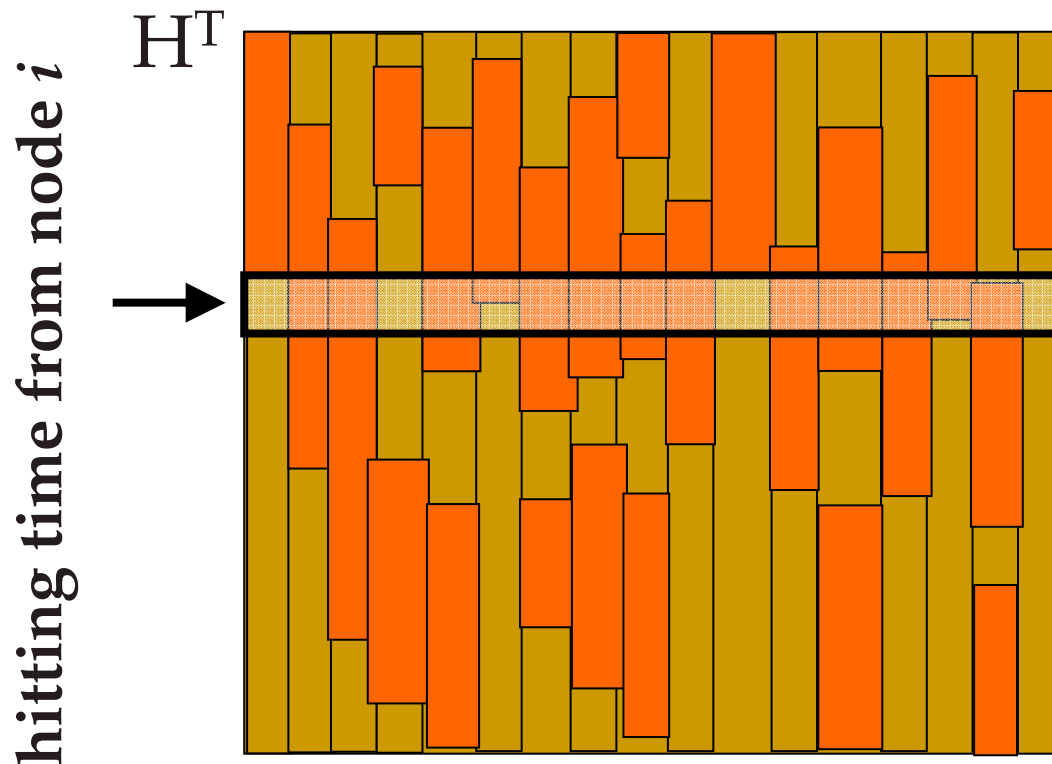
H^T



Fill up all columns

$$H^T[i,j]=h^T(i,j)$$

GRANCH: computing hitting time from a node



$$H^T[i,j]=h^T(i,j)$$

GRANCH: Pros & Cons

- Pros:

- The amortized time and space per node is small.
- Great for computing nearest neighbors for all nodes!

- Cons:

- Large pre-processing time: Looks at entire graph.
- Significant space required: caches neighborhoods for all nodes.

- What if the graph changes between two consecutive queries?

➔ Need fast, incremental, space-efficient search algorithms!

Problem

- Given τ, k, ε for any node i , find k other nodes y within truncated commute time 2τ , s.t.

$$c_{iy}^T \leq c_{ix}^T(1+\varepsilon)$$

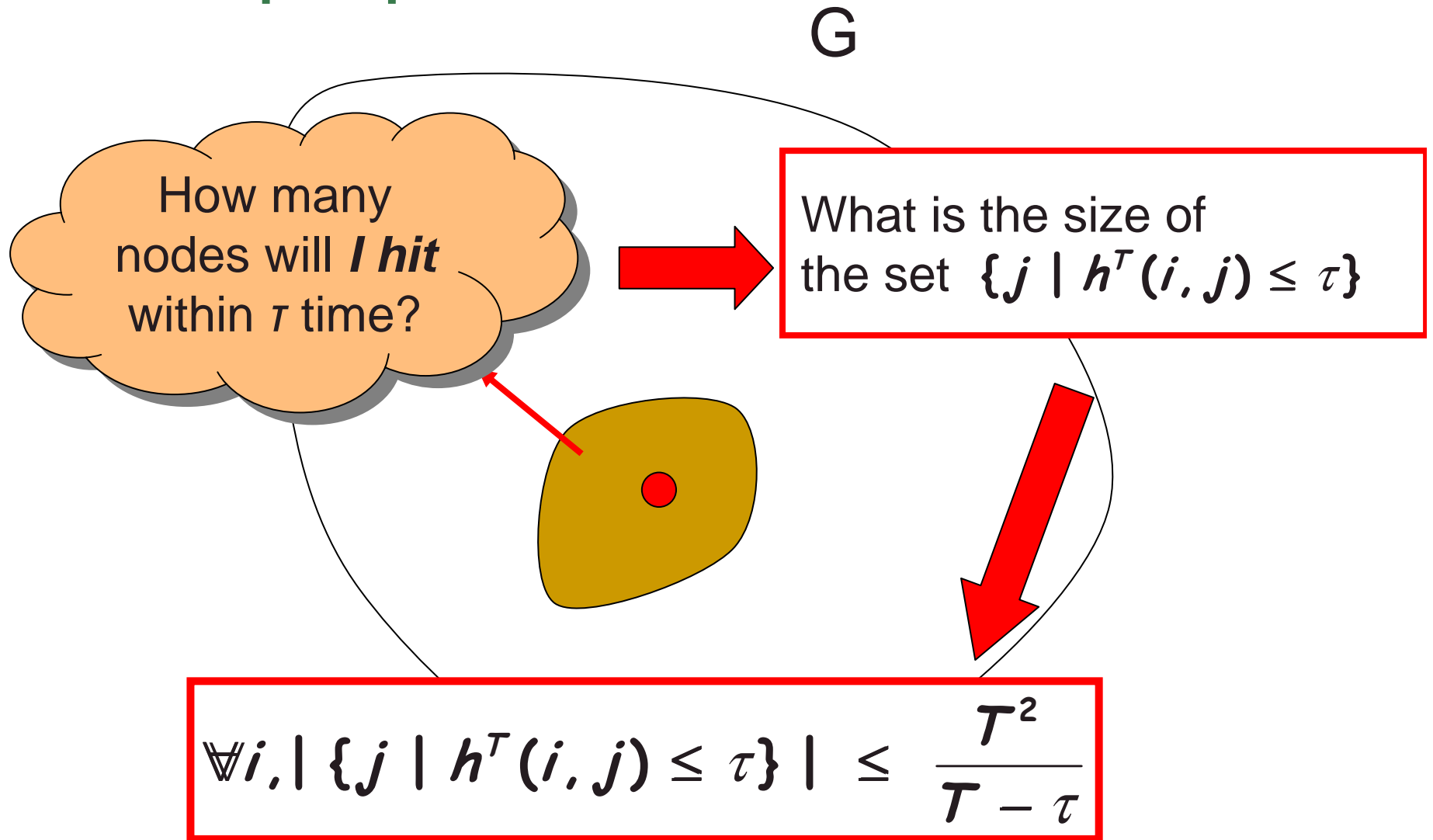
where x is the true k -th-nearest neighbor.

- How many nodes do we need to examine?
 - Number of nodes with hitting time $\leq \tau$ **from** the query
 - Number of nodes with hitting time $\leq \tau$ **to** the query
- Need theoretical justification that these two quantities are small.

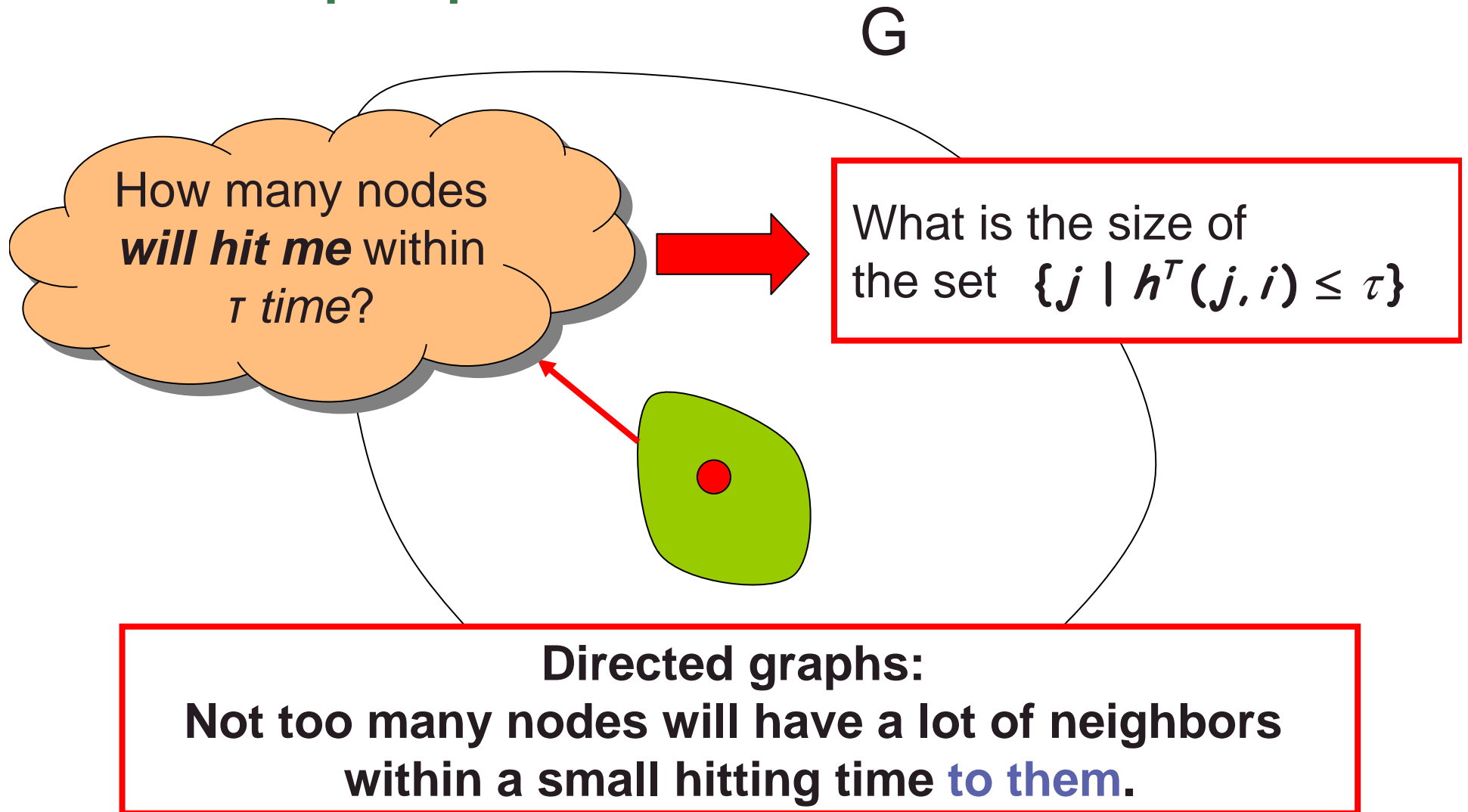
Outline

- Proximity measures on a graph
 - Standard measures: hitting/commute times
 - Improved measure: truncated versions
- Ranking using truncated measures
- Theoretical results
- Algorithm sketch
- Experimental results

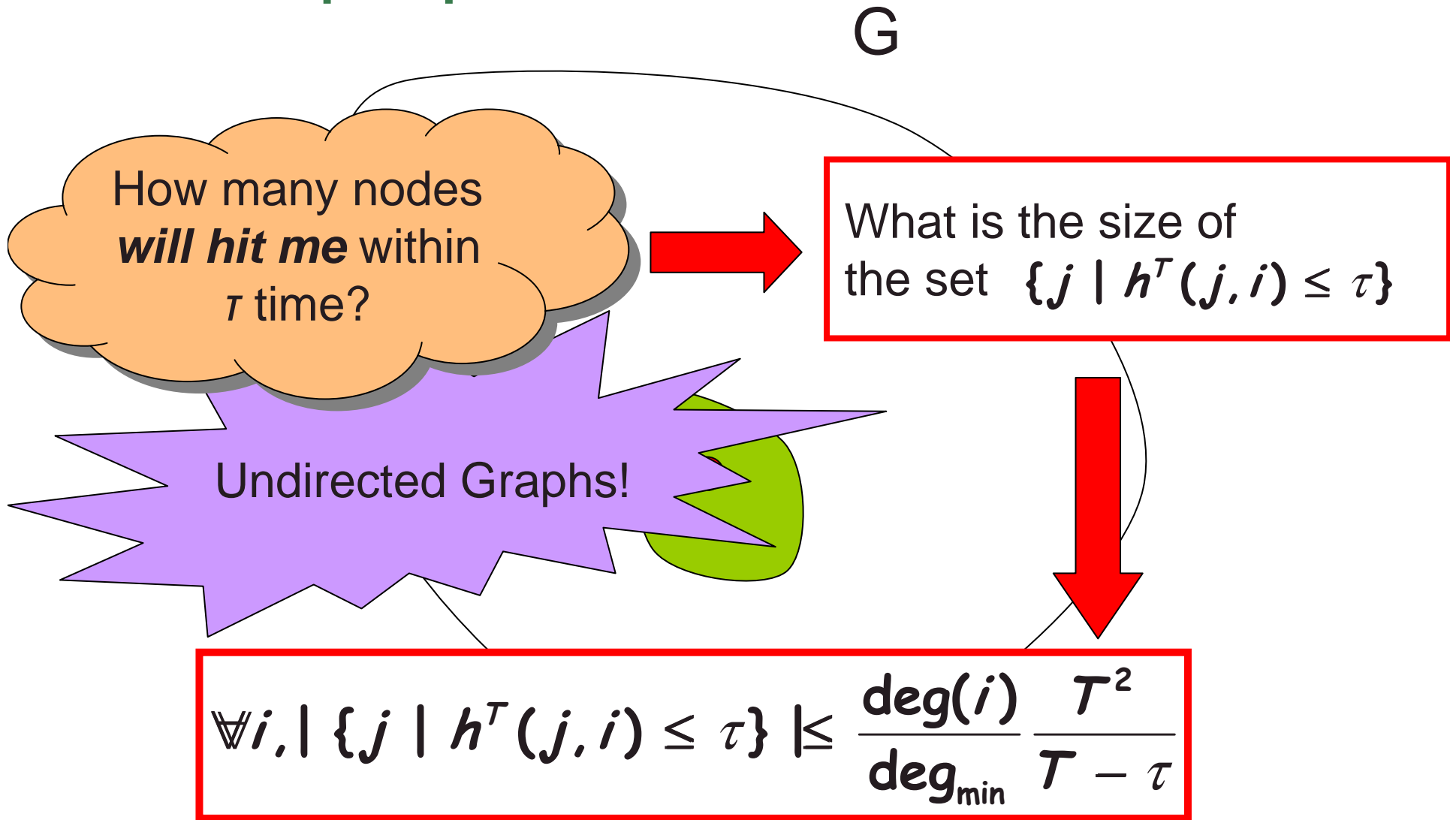
Local properties



Local properties



Local properties



Outline

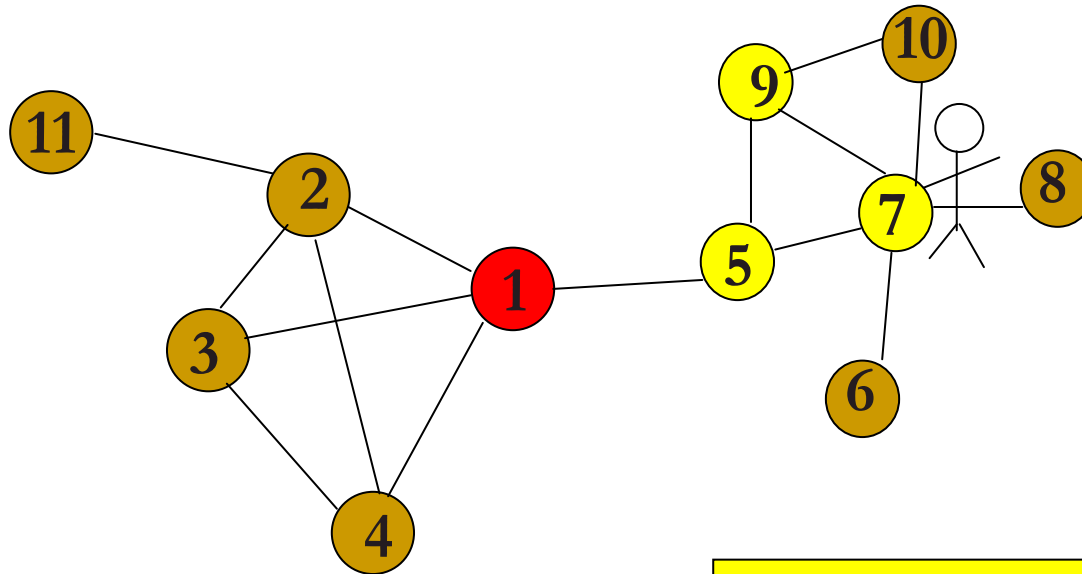
- Proximity measures on a graph
 - Standard measures: hitting/commute times
 - Improved measures: truncated versions
 - Ranking using truncated measures
 - Theoretical results
 - Algorithm sketch
 - Experimental results
-

Combining sampling and dynamic programming: Algorithm sketch

- Use sampling to estimate hitting time from node j
- Maintain a neighborhood $NB(i)$ around node j
 - Use estimated hitting times and dynamic programming to compute bounds on truncated commute time of nodes in $NB(j)$ and j .
 - Characterize truncated commute time to all nodes outside $NB(j)$ by a single lower and upper bound.
- As we expand the neighborhood these bounds get tighter.
- Perform ranking using bounds on commute times.

Sampling for computing hitting time from a node

$T=5$



$$h(1,5) = 1$$

$$h(1,9) = 2$$

$$h(1,7) = 4$$

$$h(1,-) = 5$$

1 5 9 5 7

Sampling for computing hitting time from a node

- Define $X(i,j)$ as the first hitting time at j .
- If the random walk never hits j , $X(i,j) = T$.

$$h^T(i,j) = E(X_{ij})$$

$$\hat{h}^T(i,j) = \bar{X}_{ij}$$

Sample complexity bounds

Estimating $h(i,j)$, $j=1,\dots,n$

$$\text{If } \mathcal{M} \geq \frac{1}{2\varepsilon^2} \log \frac{2n}{\delta}$$

$$\Pr(\exists j \in \{1 : n\}, |h^T(i, j) - \hat{h}^T(i, j)| \geq \varepsilon \mathcal{T}) \leq \delta$$

The number of samples needed to achieve low error with high probability is proportional to $\log(n)$.


Sample complexity bounds

Retrieving the top k neighbors

- Let $\{j_1 j_2 j_3 \dots j_k\}$ be the true k nearest neighbors of
- Also, let $\alpha = h(i, j_{k+1}) - h(i, j_k)$

$$\text{If } M \geq \frac{1}{2\alpha^2} \log \frac{nk}{\delta}$$

then we can retrieve the top k neighbors with high probability.

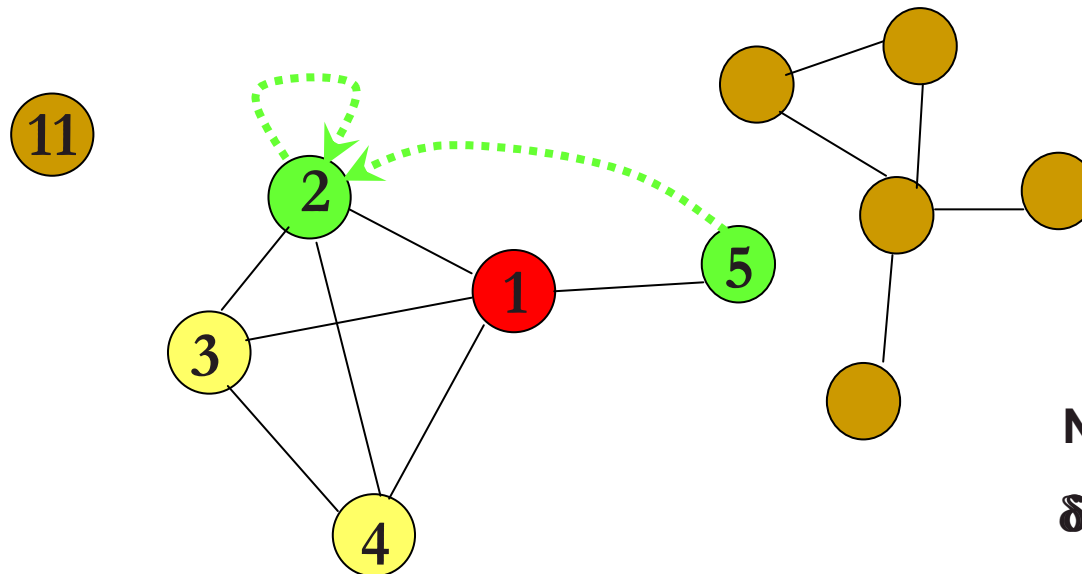


**Well separated:
easy to retrieve!**

Optimistic bounds (ho-values)

UAI'07

- What is the fastest way to go back ?
 - Jump to the boundary node that has smallest hitting time to the destination (2)



$$\text{NB}(1) = \{2 \ 3 \ 4 \ 5\}$$

$$\delta(1) = \{2 \ 5\}$$

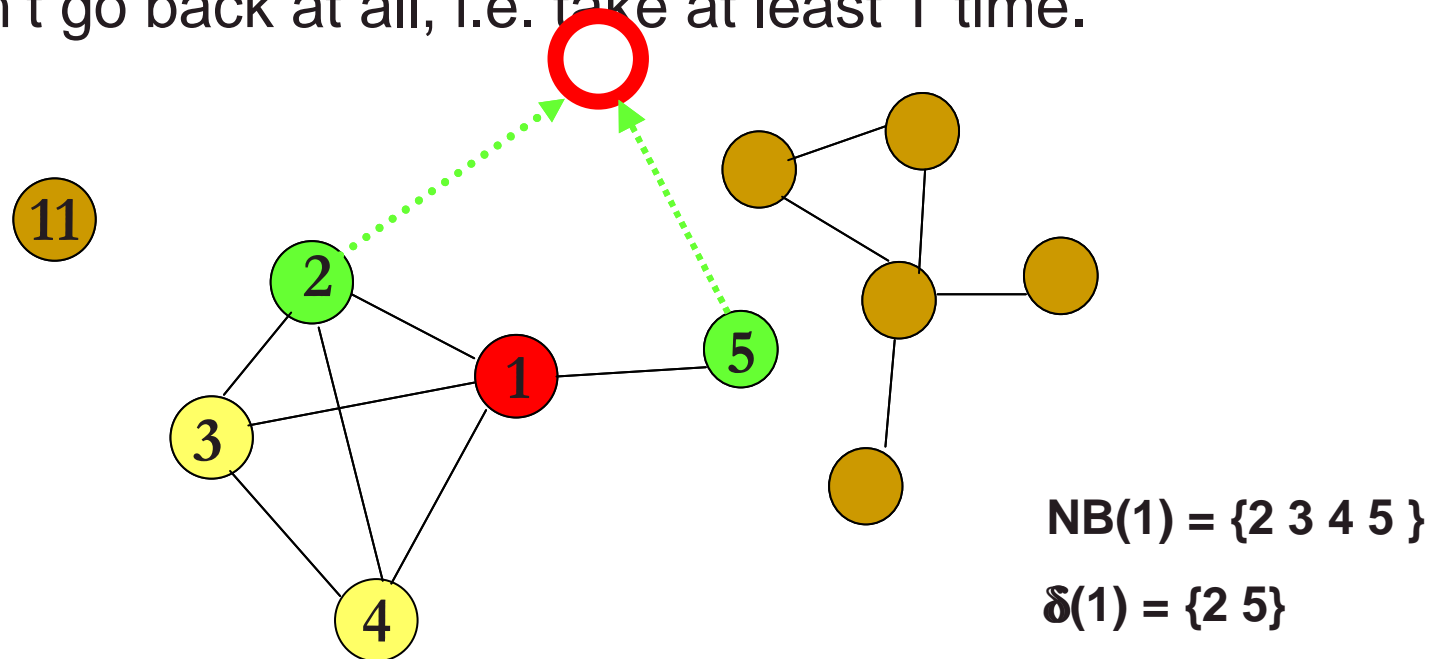
$$ho_{ij}^1 = 1$$

$$ho_{ij}^T = 1 + \sum_{\text{neighbors within NB}} p_{ik} ho_{kj}^{T-1} + (1 - \sum_{\text{neighbors within NB}} p_{ik}) (1 + \min_{m \in \delta(j)} ho_{mj}^{T-2})$$

Pessimistic bounds (hp-values)

UAI'07

- What is the worst case ?
 - Don't go back at all, i.e. take at least T time.



$$hp_{ij}^1 = 1$$

$$hp_{ij}^T = 1 + \sum_{\text{neighbors within NB}} p_{ik} hp_{kj}^{T-1} + (1 - \sum_{\text{neighbors within NB}} p_{ik})(T - 1)$$

New upper and lower bounds on commute times

$$\begin{array}{ccc} i \in NB(j) & \xrightarrow{\text{w.h.p}} & \begin{aligned} co(i, j) &= \hat{h}(j, i) + ho(i, j) \\ cp(i, j) &= \hat{h}(j, i) + hp(i, j) \end{aligned} \end{array}$$

$$\begin{array}{ccc} i \notin NB(j) & \xrightarrow{\text{w.h.p}} & \begin{aligned} co(i, j) &= lb_{ct}(j) \\ cp(i, j) &= 2T \end{aligned} \end{array}$$

Can estimate
from samples

Expanding the Neighborhood of j

- Estimate hitting time from j to all nodes by sampling.
- Start with all nodes within one or two hops of node j .
 - **Compute co^T, cp^T values of current neighborhood.**
 - **Compute $lb_{ct}(j)$**
 - **Expand neighborhood**
- Stop when $lb_{ct}(j) \geq 2\pi$

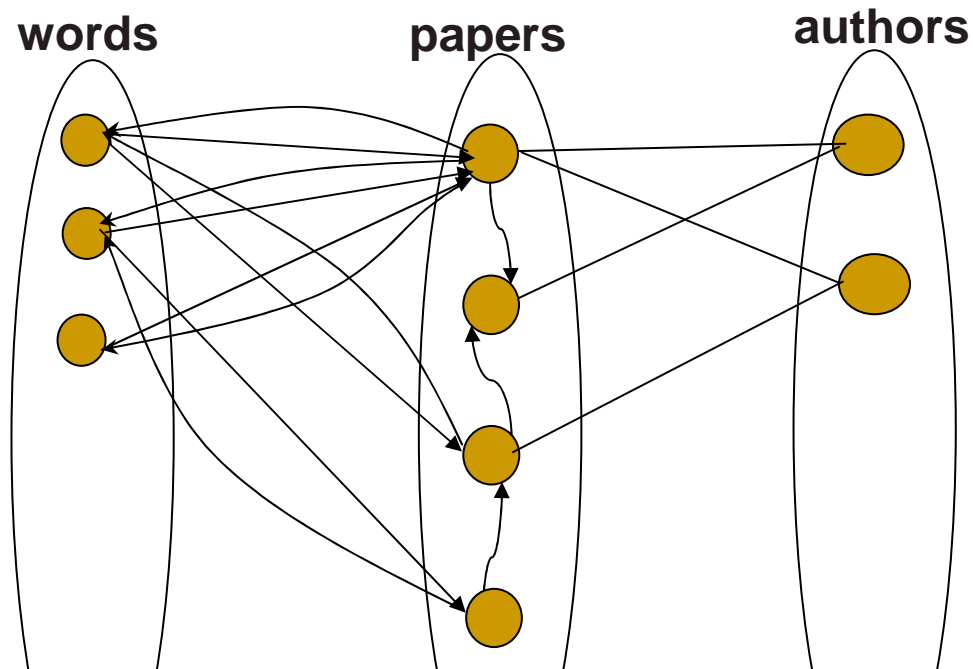
Outline

- Proximity measures on a graph
 - Standard measures: hitting/commute times
 - Improved measures: truncated versions
- Ranking using truncated measures
- Theoretical results
- Algorithm sketch
- Experimental results

Citeseer graph: Average query processing time

- Find k nearest neighbors of a query node in truncated hitting/commute times
- 628,000 nodes. 2.8 Million edges on a single CPU machine.
 - Sampling (1000 samples) 0.7 seconds
 - Exact truncated commute time: 88 seconds
 - Hybrid commute time: 4 seconds

Keyword-author-citation graphs

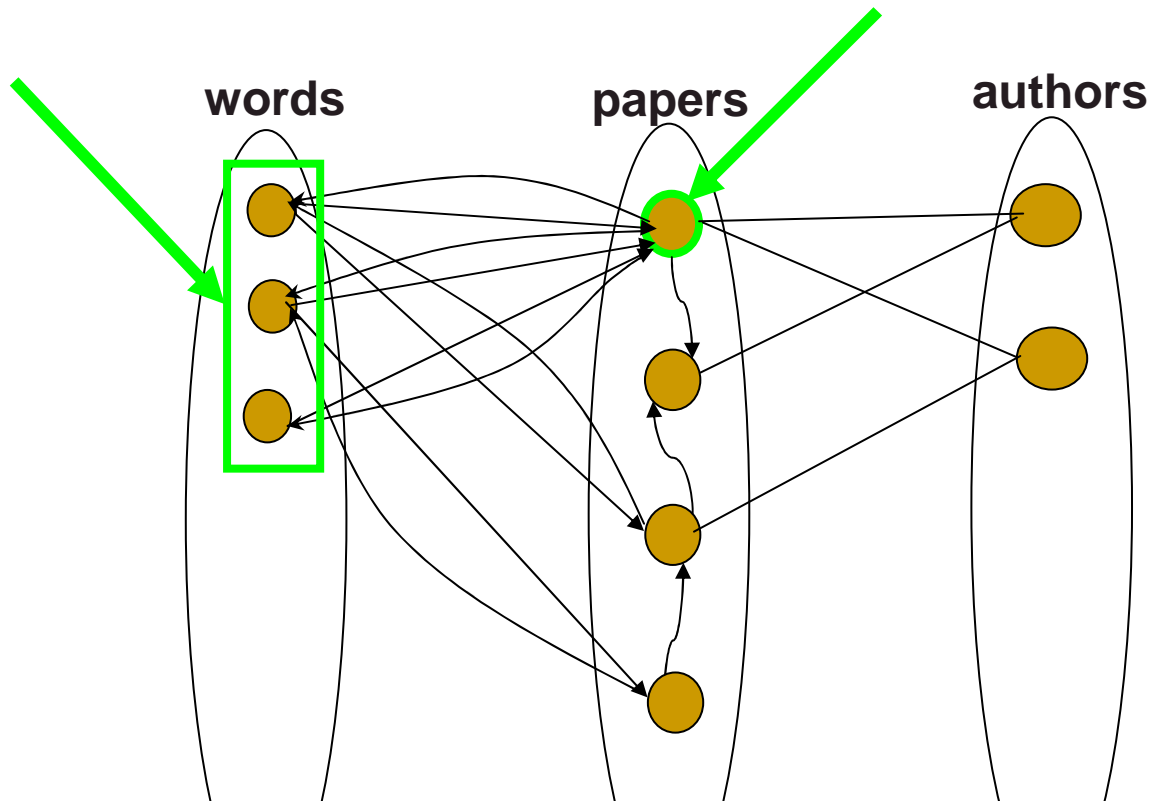


- Existing work use Personalized Pagerank (PPV) .
- We present quantifiable link prediction tasks
- We compare PPV with truncated hitting and commute times.

Dynamic personalized pagerank in entity-relation graphs. (Chakrabarti, S., WWW 2007)

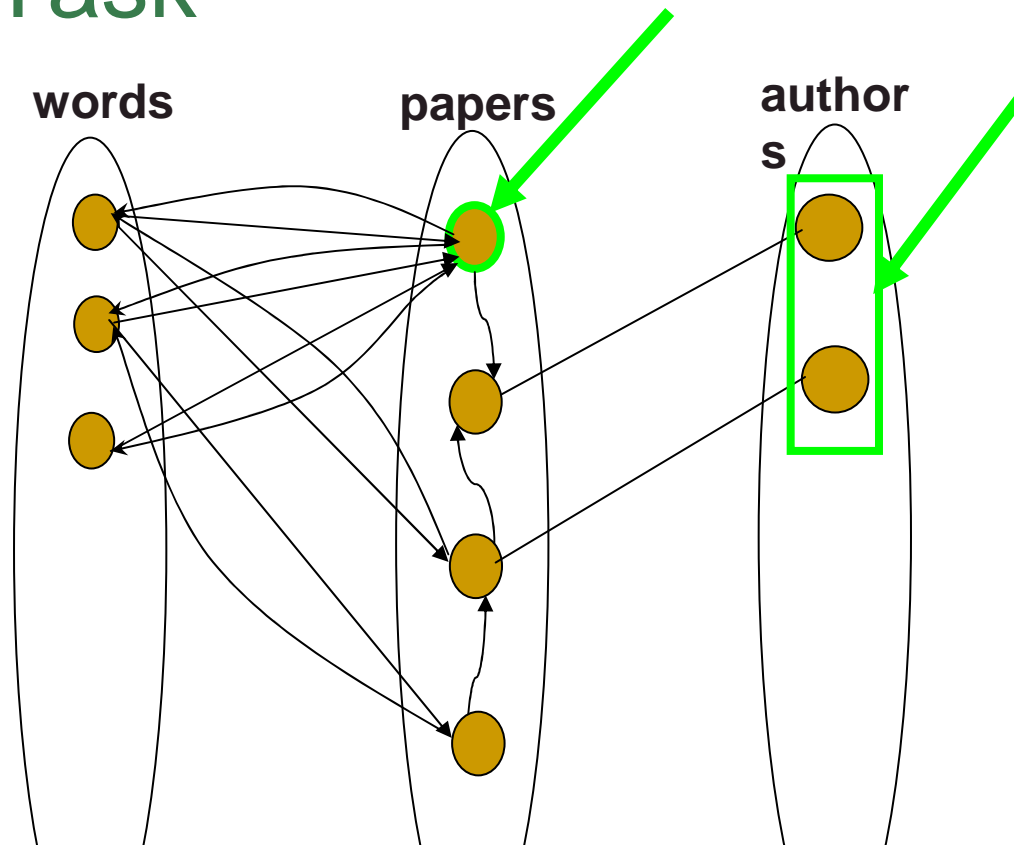
Balmin, A., Hristidis, V., & Papakonstantinou, Y. (2004). ObjectRank: Authority-based keyword search in databases. *VLDB 2004*.

Word Task



Remove the links between the words and the paper
Rank the papers for these words.
See if the paper comes up in top 1,3,5,...

Author Task

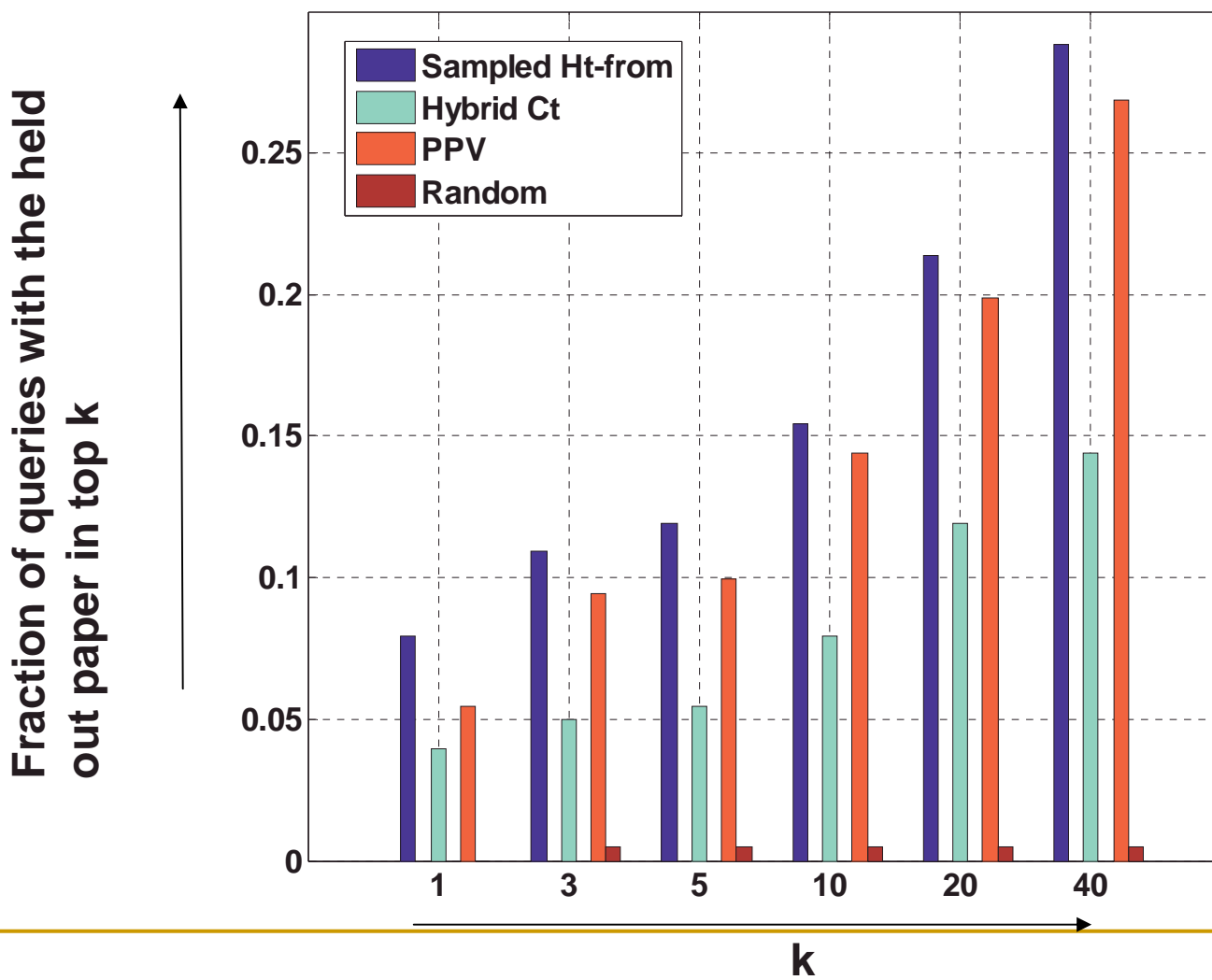


Remove the links between the authors and the paper

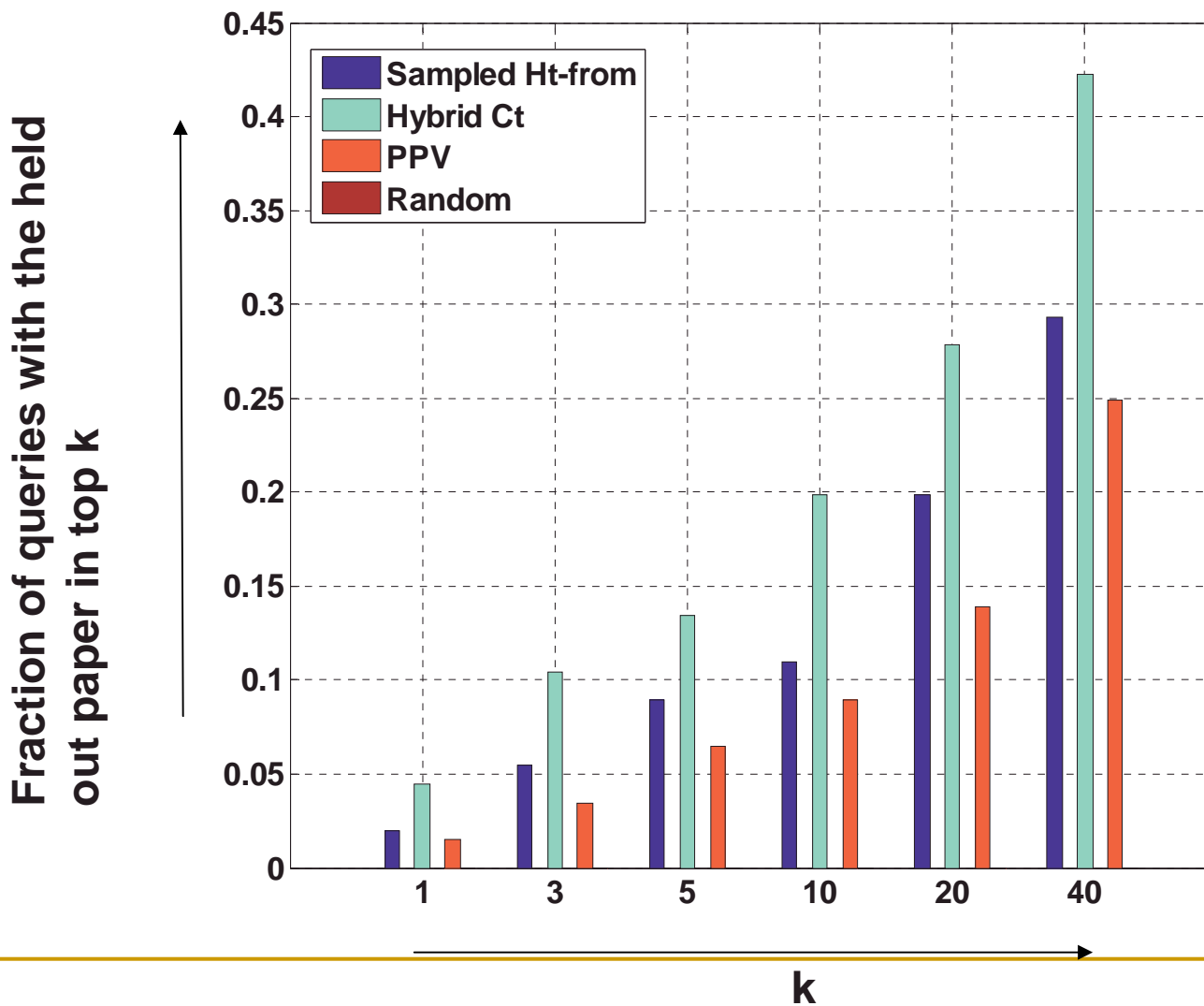
Rank the papers for these authors.

See if the paper comes up in top 1,3,5,...

Word Task



Author Task



Conclusion

- *On the fly* algorithm to compute approximate nearest neighbors in truncated hitting and commute time.
- If the graph changes between consecutive queries, our algorithm is fast.
- On most quantifiable link prediction tasks our measures outperform personalized pagerank.
- On a single CPU machine, we can process queries in 4 seconds on average in a graph with 600,000 nodes and 3 million edges.

Acknowledgement

- Soumen Chakrabarti
- Anupam Gupta
- Geoffrey Gordon
- Tamás Sarlós
- Martin Zinkevich

Thank you