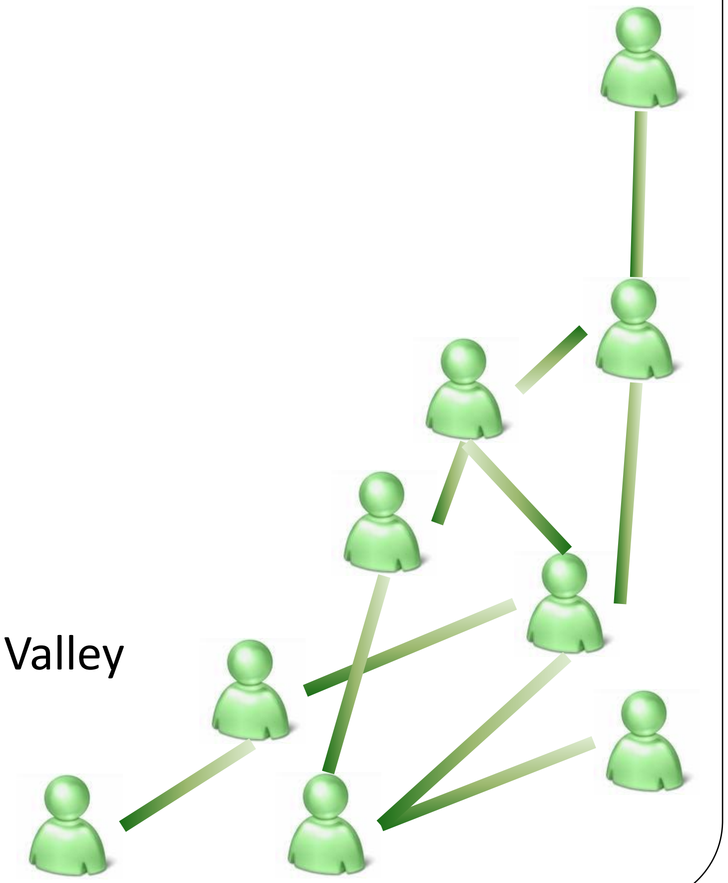
Scalable Influence Maximization in Social Networks under the Linear Threshold Model

Wei Chen
Microsoft Research Asia

In collaboration with

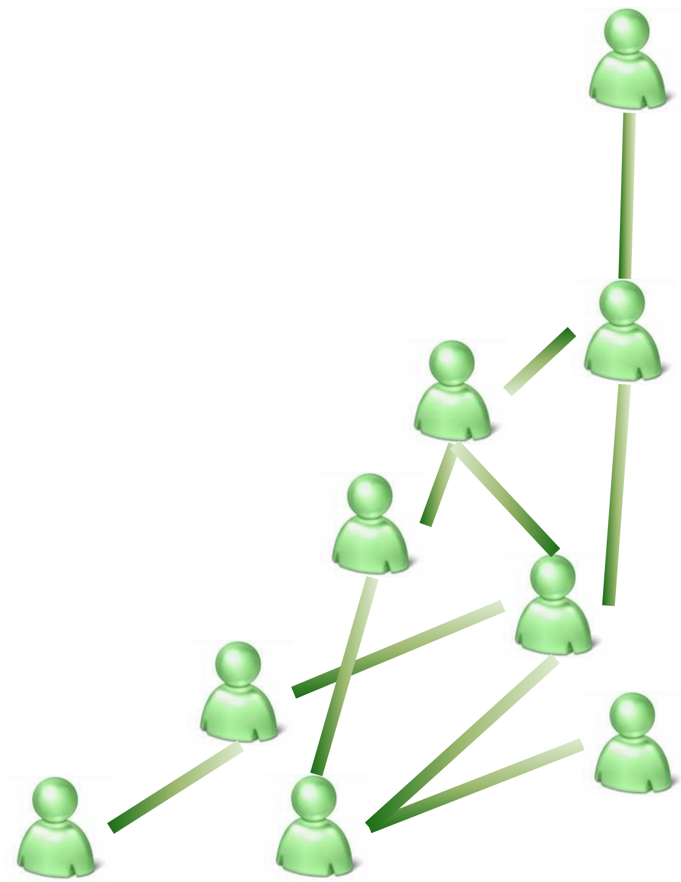
Yifei Yuan — University of Pennsylvania
Li Zhang — Microsoft Research Silicon Valley



ICDM'10, Dec. 15, 2010

Outline

- Background and problem definition
- Influence computation in the linear threshold model
- Local directed acyclic graph (LDAG) heuristic
- Experimental evaluations
- Future directions



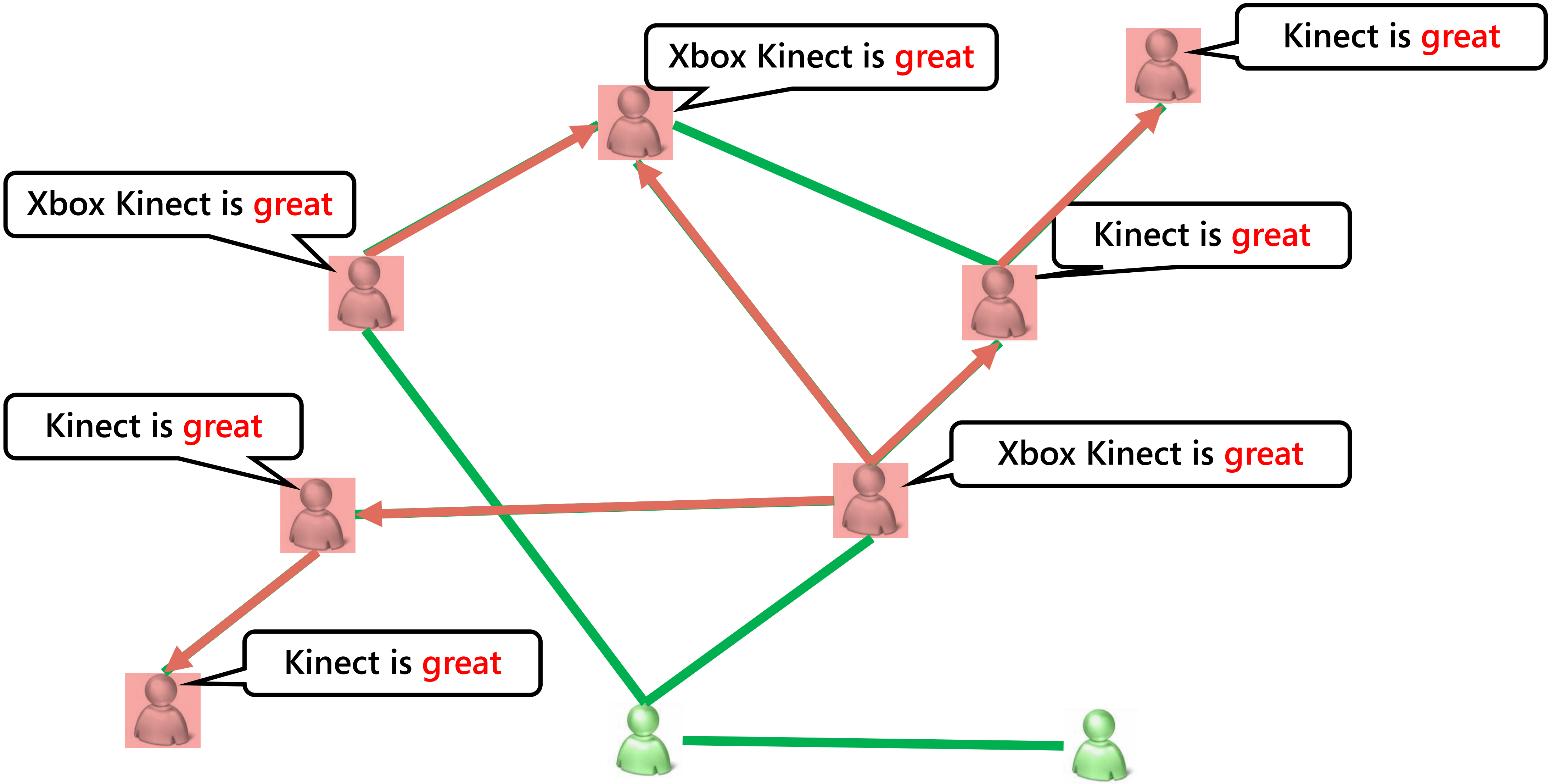
Explosion of Online Social Media

Social Influence

- **Social influence** is when the actions or thoughts of individual(s) are changed by other individual(s).
- Social influence is everywhere

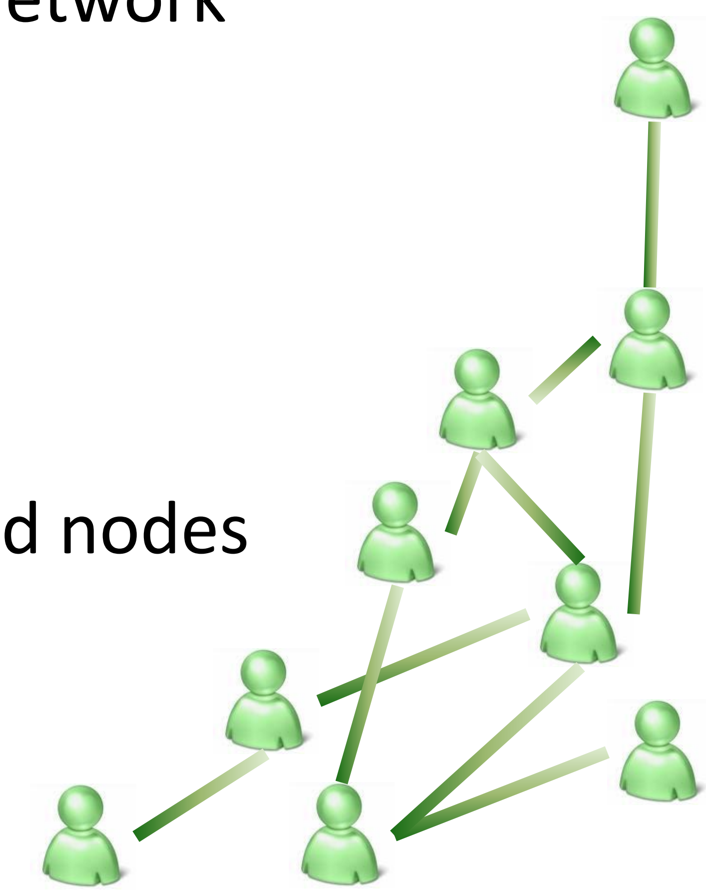
- How can we extract social influence pattern from rich online social media?
- How can we utilize social influence in online social media? --- focus of this paper

A Hypothetical Example of Viral Marketing



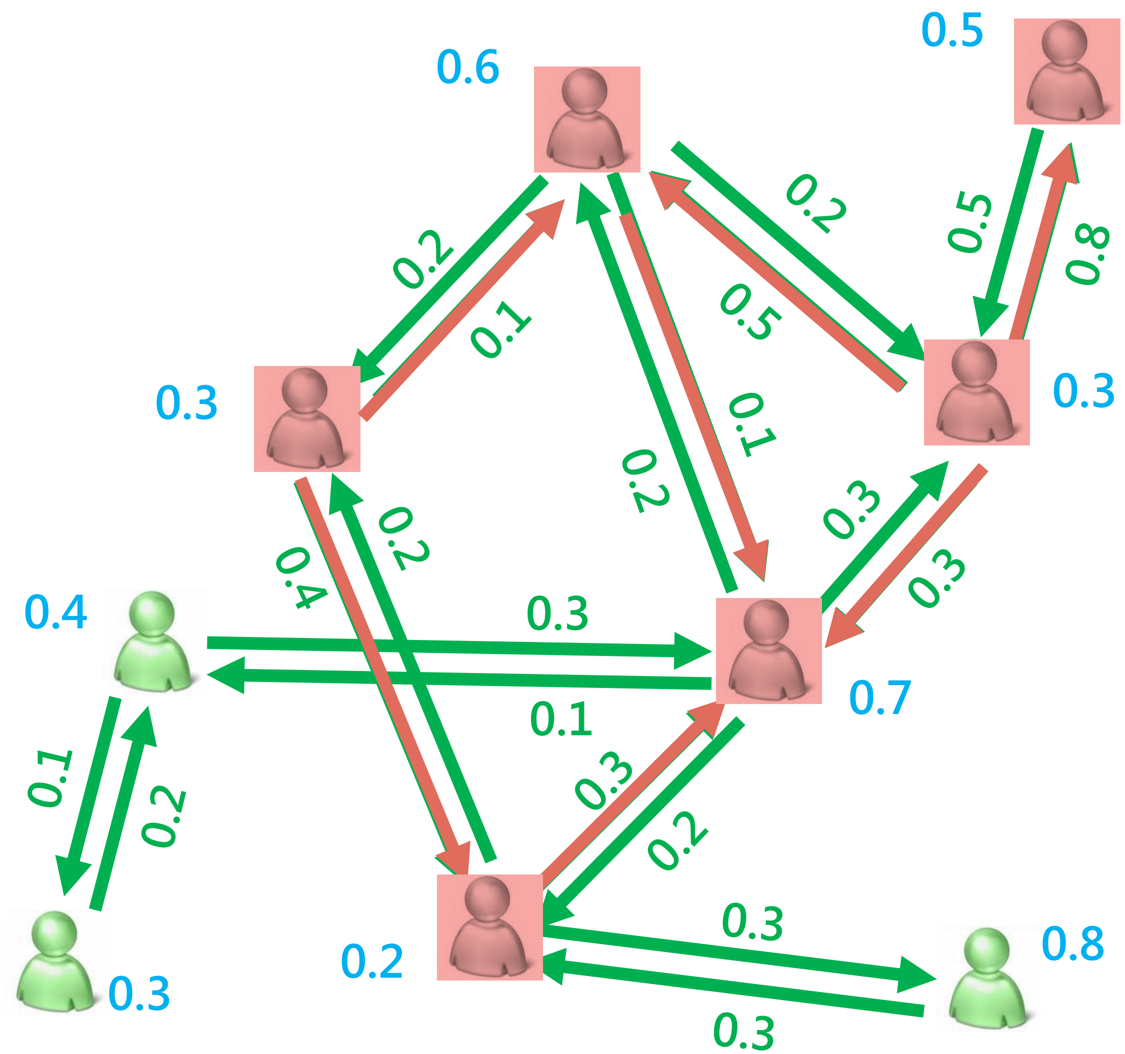
The Problem of Influence Maximization

- Given a social network
- Given a dynamic influence cascade model
  - From an initial seed set, a stochastic process propagates node activation (influence) to part of the network
  - independent cascade (IC) model
  - linear threshold (LT) model
- Influence maximization:
  - finding a seed set with size at most *k*,
  - such that the expected number of activated nodes (called influence spread) is the largest



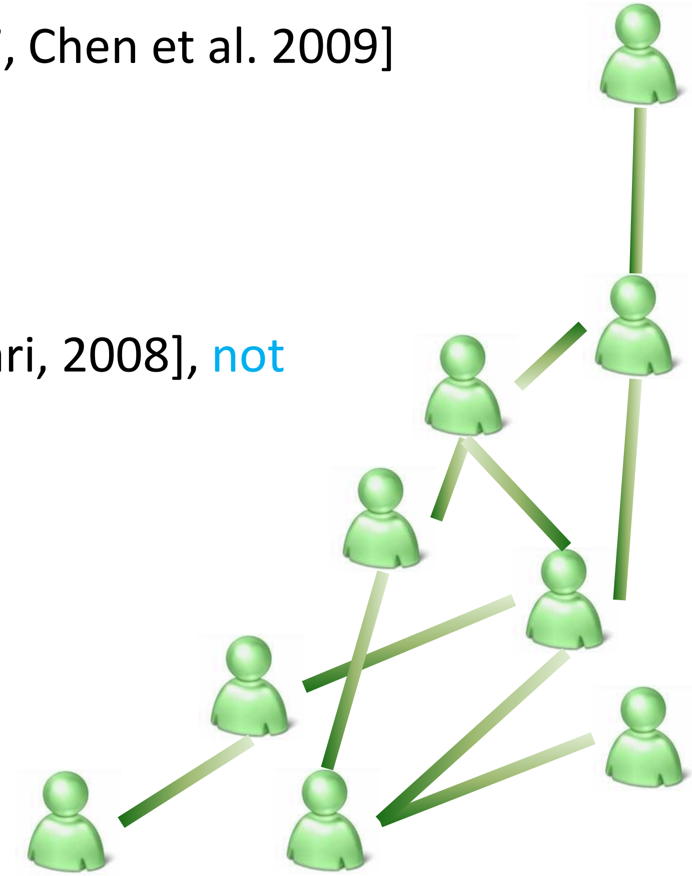
Linear Threshold Model

- Social influence graph
  - vertices are individuals
  - links are social relationships
  - link (u, v) has weight $w(u, v)$: $\sum_u w(u, v) \leq 1$
- Linear threshold model
  - each node v selects a threshold $\lambda_v \in [0,1]$ uniformly at random
  - initially some *seed* nodes are activated
  - At each step, node v checks if the weighted sum of its active neighbors is greater than its threshold λ_v, if so v is activated



Research Background

- Influence maximization as a discrete optimization problem proposed by Kempe, Kleinberg, and Tardos, in KDD'2003
- Approximation algorithms
  - Greedy approximation algorithm in [KKT'03], 63% approximation of the optimal solution
  - Several improvements on running time [Leskovec, et al. 2007, Chen et al. 2009]
  - very slow, not scalable: > 3 hrs on a 30k node graph for 50 seeds
- Heuristic algorithms
  - SPIM [Kimura and Saito, 2006], SPIN [Narayanam and Narahari, 2008], not scalable
  - PMIA [Chen et al. 2010] scalable and good performance, but only for IC model
- Lack of scalable solution for the LT model



Our Work

- Influence spread computation in the LT model
  - Computing exact influence spread in a general graph given a seed set is #P-hard (counting hardness)
    - Reduced from counting the number of simple paths in a graph
    - resolve an open problem in [KKT'03]
    - indicate the intrinsic difficulty of computing influence spread
  - Computing exact influence spread in a DAG (directed acyclic graph) can be done in linear time
- Influence maximization heuristic for the LT model
  - LDAG (local directed acyclic graph) heuristic
    - specifically designed for the LT model
    - 10^3 speedup --- from hours to seconds (or days to minutes)
    - influence spread close to that of the greedy algorithm of [KKT'03]

Computing Influence Spread in a DAG

- Setup
  - DAG $D = (V, E, w)$
  - Seed set S
  - activation probability of node v: $ap(v)$
  - Influence spread = $\sum_{v \in V} ap(v)$
- Computing activation probability in D:

$$ap(v) = \sum_{u \in V \setminus \{v\}} ap(u) \cdot w(u, v)$$

- Follow the DAG partial order to compute all $ap(v)$'s

Main Structure of the LDAG Heuristic

- Compute local DAGs (LDAGs) surrounding every node
  - idea 1: restrict influence computation at local region
- Compute incremental influence of every node based on LDAGs
  - idea 2: influence computation in DAGs is fast
- Select k seeds one by one with largest incremental influence
  - select new seed s with the largest incremental influence
  - update incremental influence of all nodes u sharing LDAGs with s
  - idea 3: batch updates reducing running time from $O(|D|^2)$ to $O(D)$

Finding an LDAG Influencing Node v

- Want:
  - a DAG D surrounding v, with v as a sink in D --- LDAG rooted at v
  - the LDAG is local
    - the influence of every node u in D to v is above some threshold θ
  - the LDAG covers a significant portion of influence
    - the sum of influence of all nodes u in D to v is as large as possible
- Exact maximization problem is NP-hard
- Greedy approach (similar to Dijkstra's shortest-path algorithm)
  - select a node x with the largest influence to v

$$x = \operatorname{argmax}_u Inf(u, v)$$

  - after x is selected, update the influence of the in-neighbors u of x, based on the linear relationship

$$Inf(u, v) \mathrel{+}= w(u, x) \cdot Inf(x, v)$$

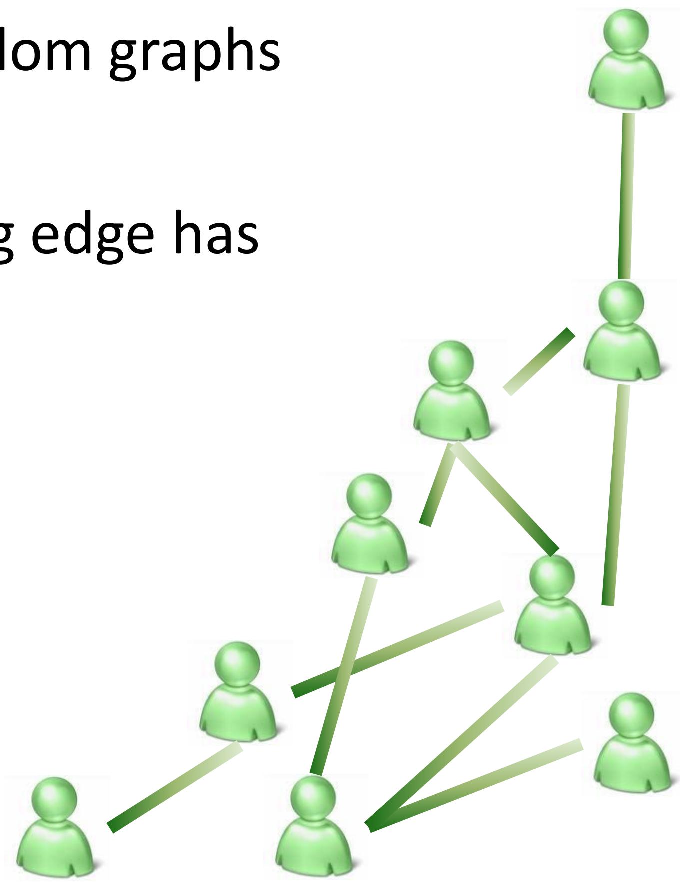
  - Repeat above two steps until no node has influence greater than θ

Efficient Batch Updates on Activation Probabilities

- For an LDAG D rooted at v, if a node u in D is selected as an additional seed, the incremental influence of u to v in D is
$$(1 - ap(u)) \cdot \alpha_v(u)$$
  - $\alpha_v(u)$'s for all u in D can be computed in linear time
  - time reduced from $O(|D|^2)$ to $O(D)$
- After selecting seed s, update $\alpha_v(w)$ for all w's that are in the same LDAGs as s

Experimental Evaluation

- Networks
  - Real-world datasets:
    - collaboration networks: arXiv (31K), DBLP (2M)
    - trust network: Epinions (509K)
    - product co-purchasing network: Amazon (1.2M)
  - Synthetic datasets: generated from power-law random graphs
- Influence weights
  - uniform: for node v with degree d_v, every incoming edge has weight $1/d_v$
  - random
- Algorithms tested: LDAG, Greedy, SPIN, PageRank, DegreeDiscount

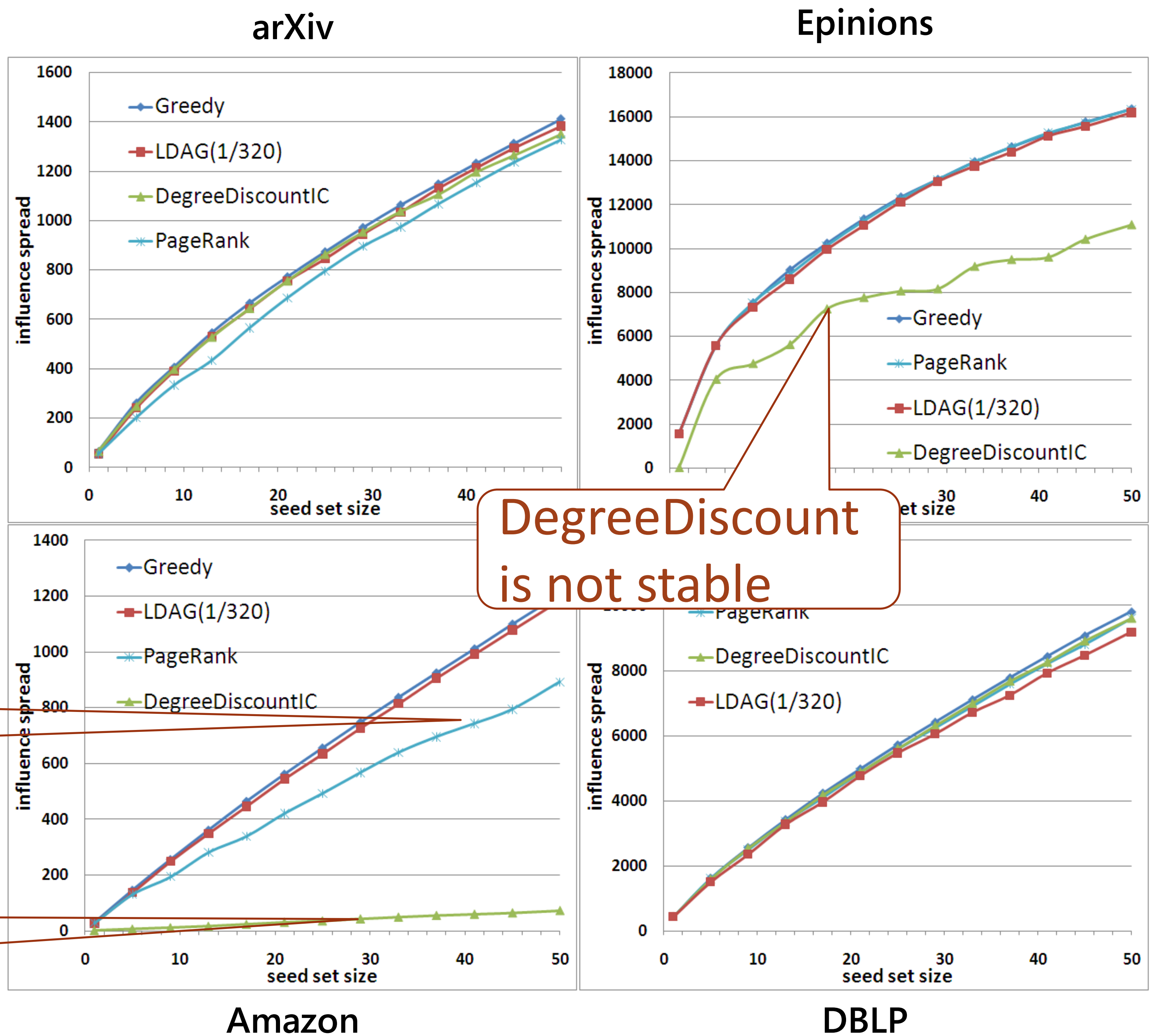


Experiment Results on Influence Spread

- LDAG always among the best
- matches with Greedy
- PageRank / DegreeDiscount not stable

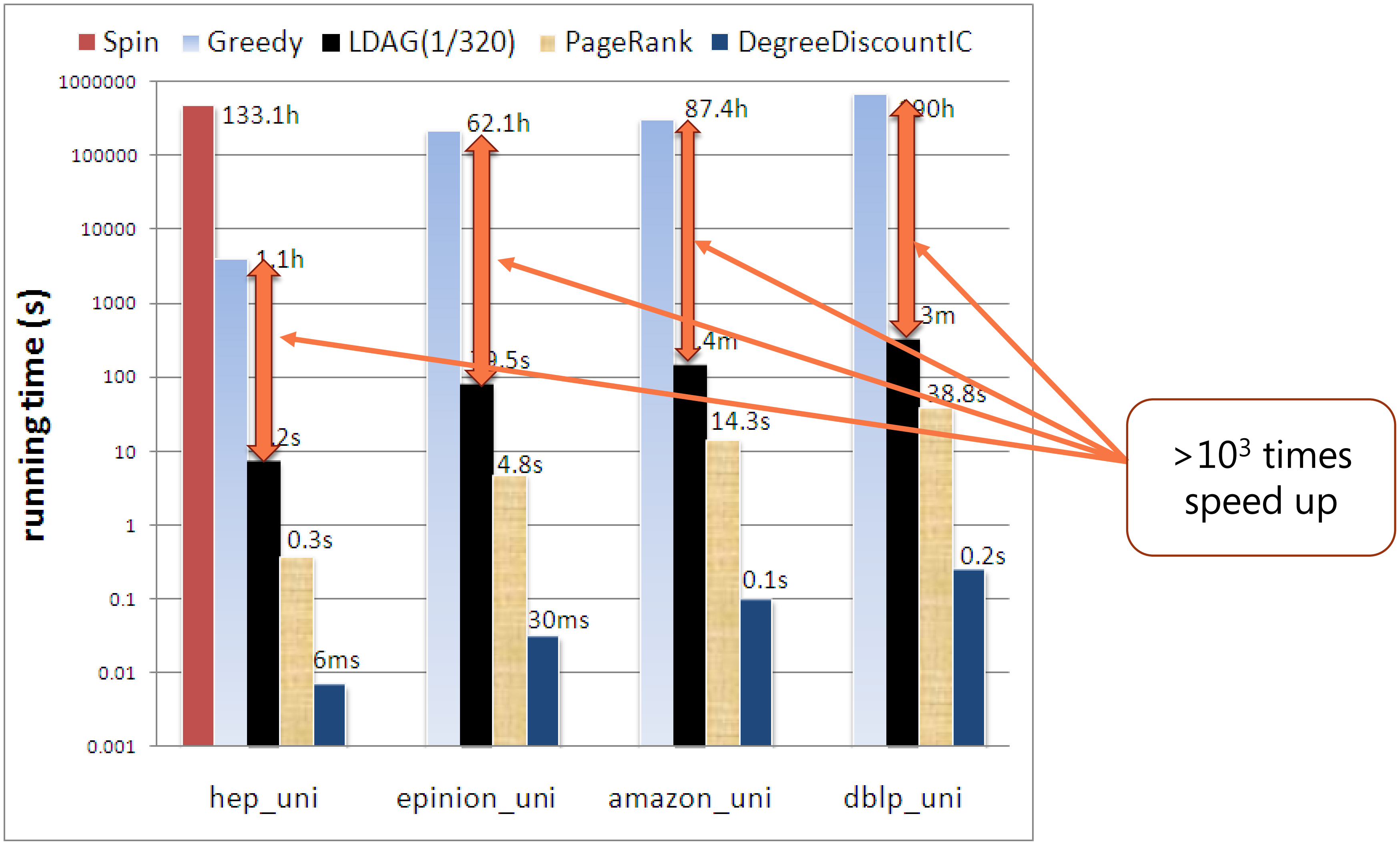
PageRank is not stable

DegreeDiscount is not stable



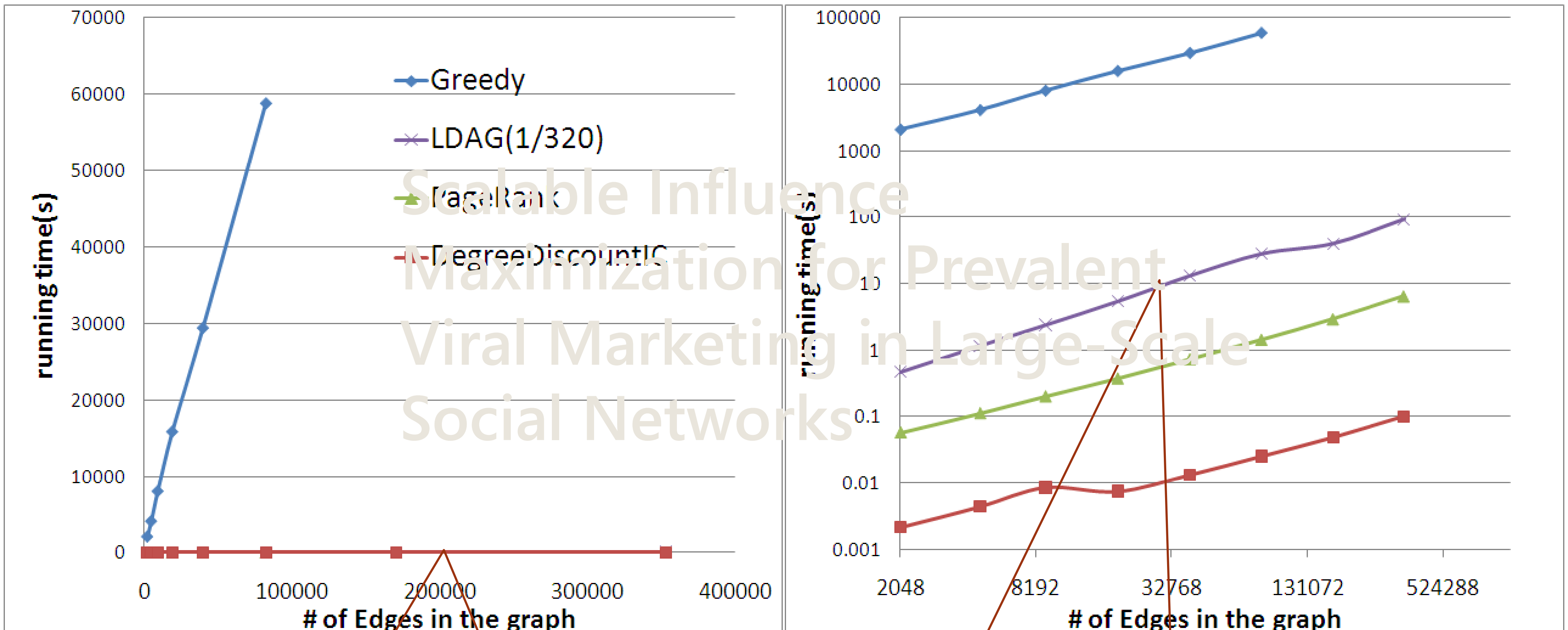
random weights

Running Time on Real-World Networks



Running time is for selecting 50 seeds

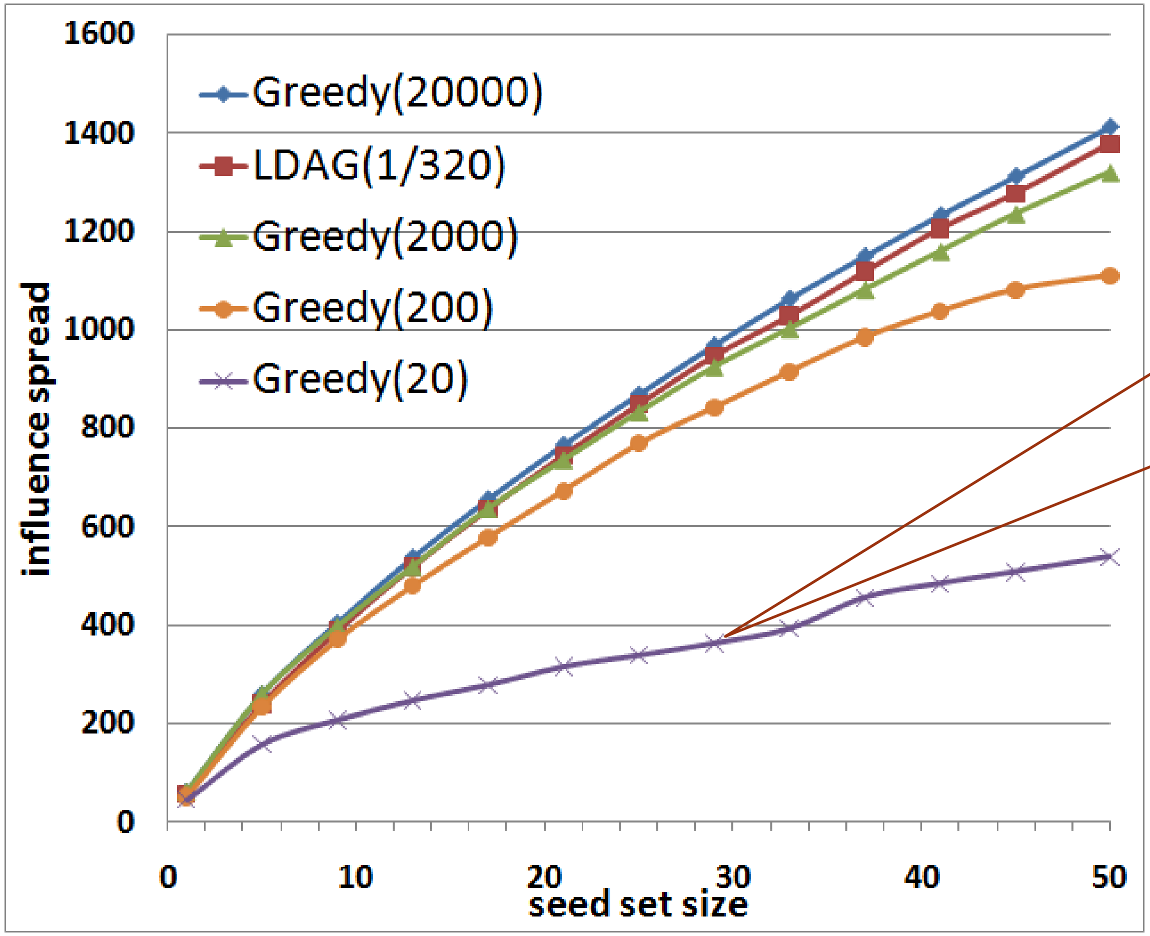
Scalability of LDAG on Synthetic Graphs



LDAG is scalable

$y \propto x^{1.02}$, almost linear

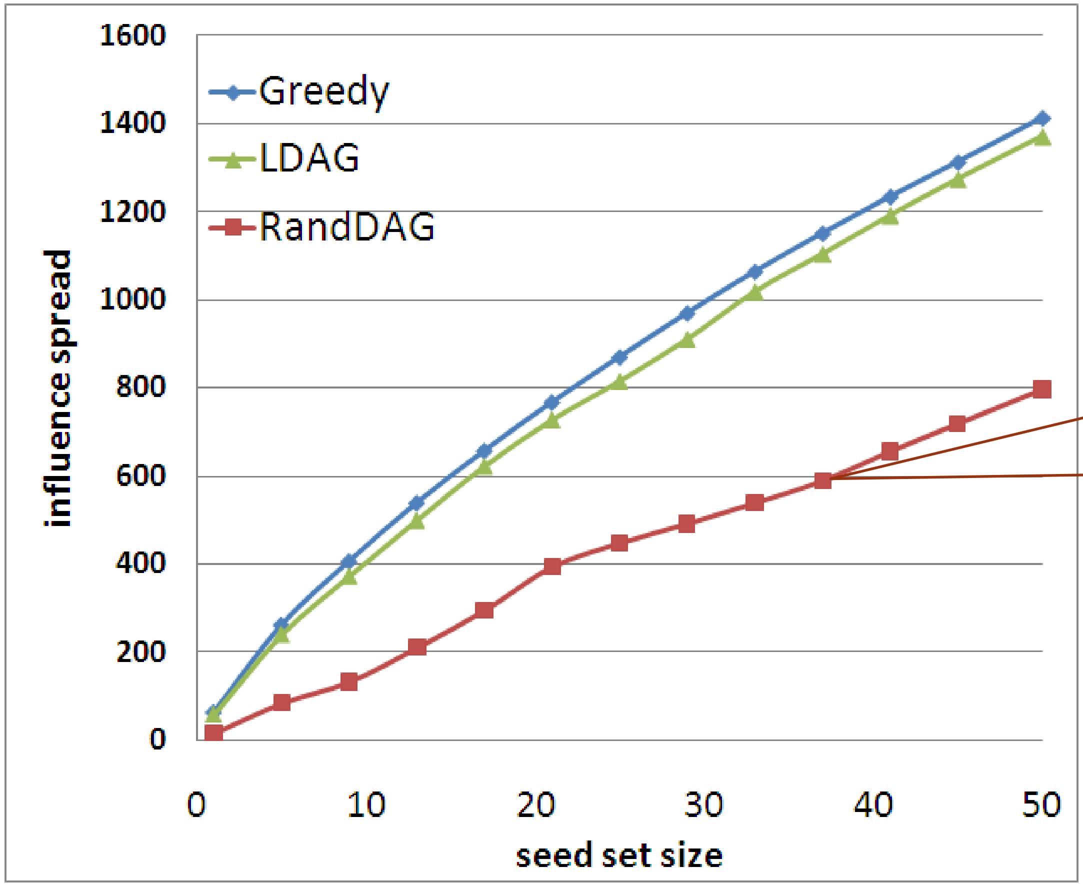
Compare with Greedy with Different Number of Simulated Cascade Runs



running time comparable with LDAG, but influence significantly worse

- Greedy cannot maintain high influence spread when reducing the number of simulations

Compare with Random LDAG Construction



random LDAG construction is significantly worse

Future Directions

- Theoretical problem: efficient approximation algorithms:
  - How to efficiently approximate influence spread given a seed set?
- Practical problem:
  - Influence analysis from online social media: How to mine the influence graph?
  - Influence maximization in other settings

Thanks!
and
questions?