2014/7/30 AAAI-14 @ Québec, Canada

Fast and Accurate Influence Maximization on Large Networks with Pruned Monte-Carlo Simulations

Naoto Ohsaka (UTokyo)

Takuya Akiba (UTokyo)

Yuichi Yoshida (NII & PFI)

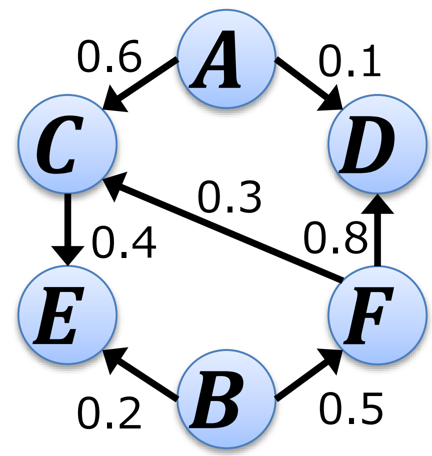
Ken-ichi Kawarabayashi (NII)

JST, ERATO, Kawarabayashi Large Graph Project

Influence Maximization

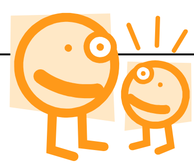
[Kempe, Kleinberg, Tardos. KDD'03]

- Input
  - Directed graph $G = (V, E)$
  - Edge probability $p_e \ (e \in E)$
  - Size of seed set k
- Problem
  - maximize $\sigma(S) \qquad (|S| \leq k)$
    - $\sigma(\cdot)$: the spread of influence



mathematically formalizing

- Motivation



  - Viral (word-of-mouth) Marketing

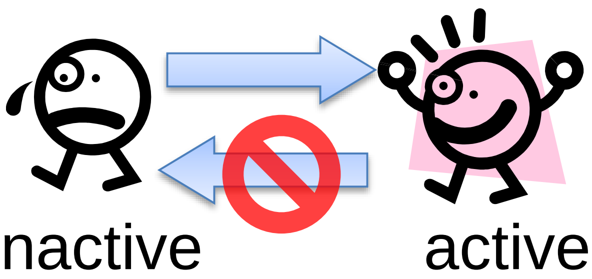
    [Domingos, Richardson. KDD'01], [Richardson, Domingos. KDD'02]

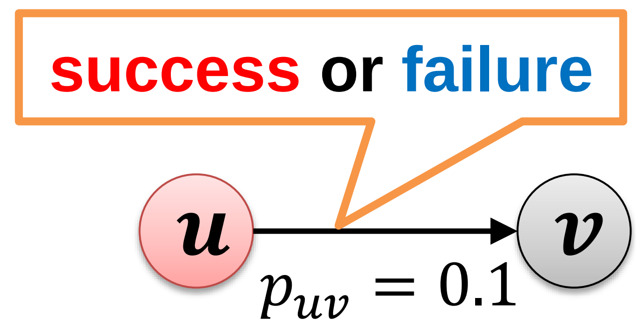
Q. How to find a small group of influential individuals?

Independent Cascade Model

[Goldenberg, Libai, Muller. Marketing Letters'01]

- Each vertex has 2 states (**inactive** / **active**)





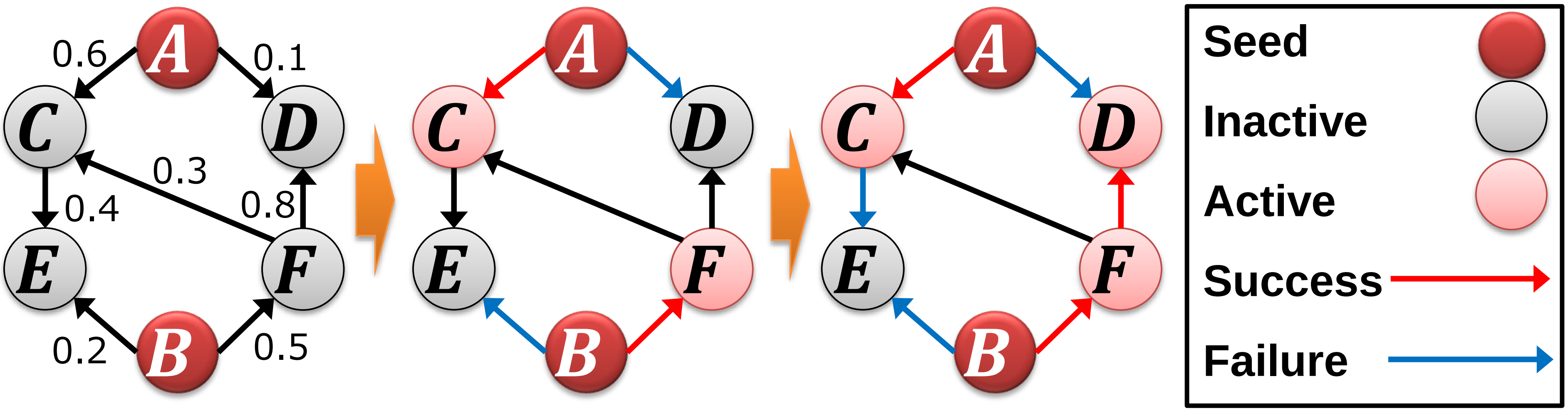
Diffusion Process

0. Activate vertices in $S \subseteq V$ called ***seed set***

1. **Active** vertex u activates **inactive** vertex v **with probability** p_{uv} (single trial)

2. Repeat ***1*** while new activations occur

Example of Independent Cascade Model



- ***Influence spread*** $\sigma(S)$
  - **Expected number** of active vertices given a seed set S

Previous Results

Hardness

Influence Maximization is
NP-hard
[Kempe, Kleinberg, Tardos. KDD'03]

Exact Computation of
$\sigma(\cdot)$ is
#P-hard
[Chen, Wang, Wang. KDD'10]

Original Greedy Approach

<u>Greedy Algorithm</u>
[Kempe, Kleinberg, Tardos. KDD'03]
Approx. ratio ≈ **63%**

<u>Monte-Carlo Simulations</u>
Good approximation

Original Greedy Approach

- Greedy Algorithm [Kempe, Kleinberg, Tardos. KDD'03]

$S \leftarrow \emptyset$
while $|S| < k$ **do**
  $t \leftarrow \arg\max_{v \in V} \sigma(S \cup \{v\}) - \sigma(S)$
  $S \leftarrow S \cup \{t\}$

Due to **submodularity** of $\sigma(\cdot)$

$$\sigma(S) \geq \left(1 - \frac{1}{e}\right) \mathrm{OPT} \geq 0.63\ \mathrm{OPT}$$

[Nemhauser, Wolsey, Fisher. Mathematical Programming'78]

- Monte-Carlo Simulations ($1 \pm \varepsilon$ approximation)
  [Kempe, Kleinberg, Tardos. KDD'03]
  - Simulating diffusion process repeatedly
  - Averaging # of active vertices

Produces **near-optimal** $\left(1 - \frac{1}{e} - \varepsilon'\right)$ solutions

Issue: Original Greedy Approach Suffers from Scalability

Greedy Algorithm

\# of Evaluating $\sigma(\cdot)$:

$\boldsymbol{nk}$

Monte-Carlo Simulations

Computation Time of $\sigma(\cdot)$:

$\boldsymbol{O(mR)}$

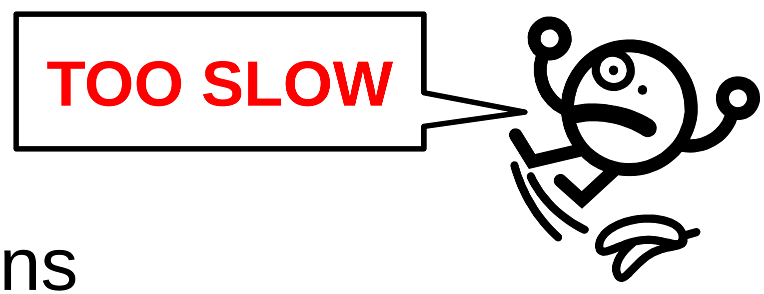
Total Time: $\boldsymbol{O(knmR)}$ $(R \approx 10{,}000)$

$n = |V|$ >10^6

$m = |E|$ >10^7

k: # of seeds

$R = \mathrm{poly}(\varepsilon^{-1})$: # of simulations



Previous Methods for Influence Maximization

	Low Quality	High Quality
Slow		Greedy Approach [Kempe, Kleinberg, Tardos. KDD'03] CELF [Leskovec, Krause, Guestrin, Faloutsos, VanBriesen, Glance. KDD'07] StaticGreedyDU [Cheng, Shen, Huang, Zhang, Cheng. CIKM'13]
Fast	DegreeDiscount [Chen, Wang, Yang. KDD'09] PMIA [Chen, Wang, Wang. KDD'10] SAEDV [Jiang, Song, Cong, Wang, Si, Xie. AAAI'11] IRIE [Jung, Heo, Chen. ICDM'12]	**CHALLENGE**

Simulation-based

Heuristic-based

Our Contribution

- Propose a **simulation-based fast** algorithm
  - **Fast**
    - Comparable to heuristics
    - Can handle graphs
      with **60M** edges in **20** min.
  - **Accurate**
    - Has a theoretical guarantee
    - Better than heuristics

Outline of Proposed Method

- Preprocessing: Generating random graphs

⇧ **Coin Flip Technique**

- Greedy Strategy

$S \leftarrow \emptyset$

while $|S| < k$ **do**

$$t \leftarrow \arg\max_{v \in V} \underline{\underline{\sigma(S \cup \{v\}) - \sigma(S)}}$$

$S \leftarrow S \cup \{t\}$ ⇧ **Our Speed-up Techniques**

Preprocessing: Generating Random Graphs

Coin Flip Technique

[Kempe, Kleinberg, Tardos. KDD'03]

Computing influence spread $\sigma(S)$

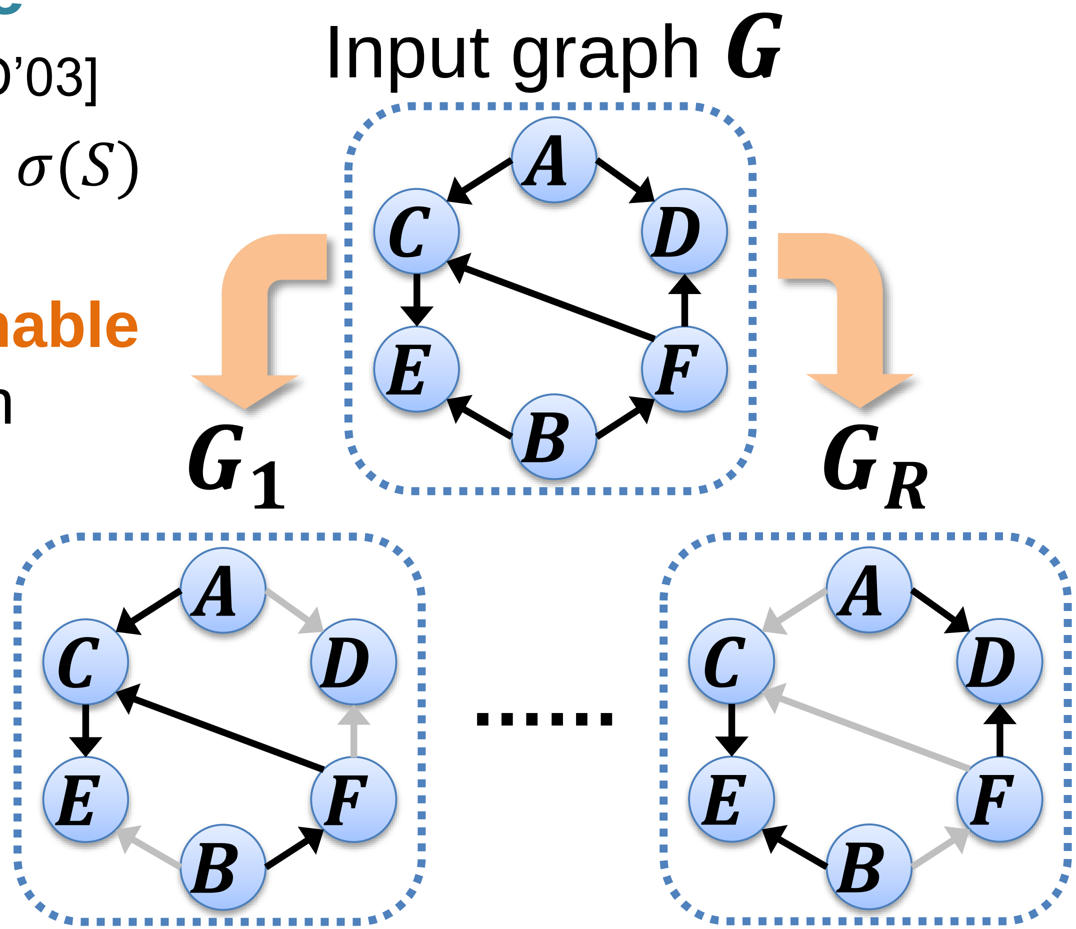
||

Counting # of vertices **reachable** from S on random graph

Edge e lives w.p. p_e

live edge: **success**

blocked edge: **failure**



R random graphs

How to Approximate $\sigma(S)$

$$\sigma(S) \approx \frac{1}{R}\sum_{i=1}^{R} \sigma_{G_i}(S)$$

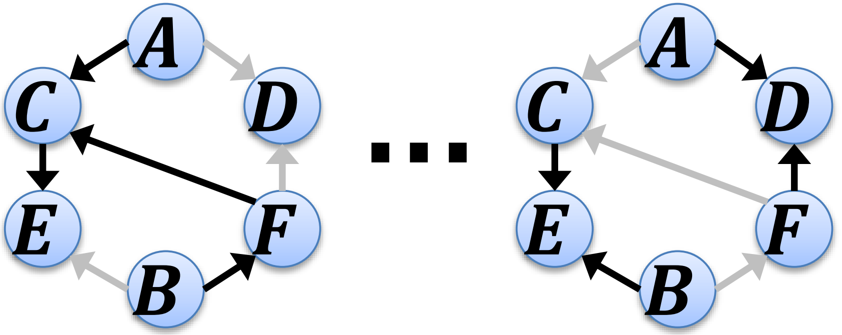
$\sigma_{G_i}(S)$ = # of vertices **reachable** from S on G_i

R = 200

v	$\sigma_{G_1}(\{v\})$	…	$\sigma_{G_R}(\{v\})$	$\sigma(\{v\})$
A	3	…	2	2.4
B	4	…	2	2.8
C	2	…	2	1.6
D	1	…	1	1
E	1	…	1	1
F	3	…	2	2.2

10^6

CHALLENGE
Computing this table as **fast** as possible



Proposed Speed-up Techniques

(we apply each random graph)

1. **Pruned BFS** for reachability tests (on random graphs)
(We will focus on this)
[Akiba, Iwata, Yoshida. SIGMOD'13]
[Yano, Akiba, Iwata, Yoshida. CIKM'13]
[Akiba, Iwata, Kawarabayashi, Kawata. ALENEX'14]

} **CORE IDEA of our paradigm**

2. Reducing unnecessary influence recomputations

3. Reducing # of random graphs by
Sample Average Approximation approach
[Kimura, Saito, Nakano. AAAI'07], [Cheng, Shen, Huang, Zhang, Cheng. CIKM'13]
[Sheldon et al., UAI'10]

- We provide nice theoretical bound

These techniques do **NOT** affect the estimation of $\sigma(\cdot)$

Pruned BFS

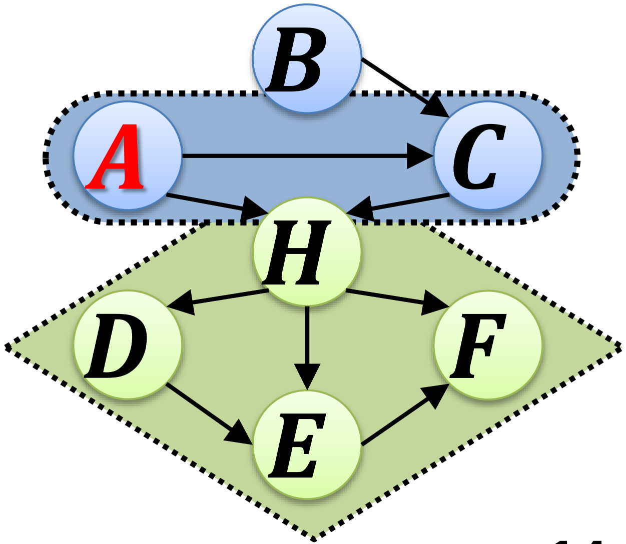
- Idea: Most BFSs are **redundant**
- **Preprocessing**: Compute **ancestors** and **descendants** of vertex H with max. deg.
- **Pruning** (BFS from v): If v is **ancestor** of H, we ignore **descendants** of H

(# of vertices visited during BFS) **2**

\+ **+**

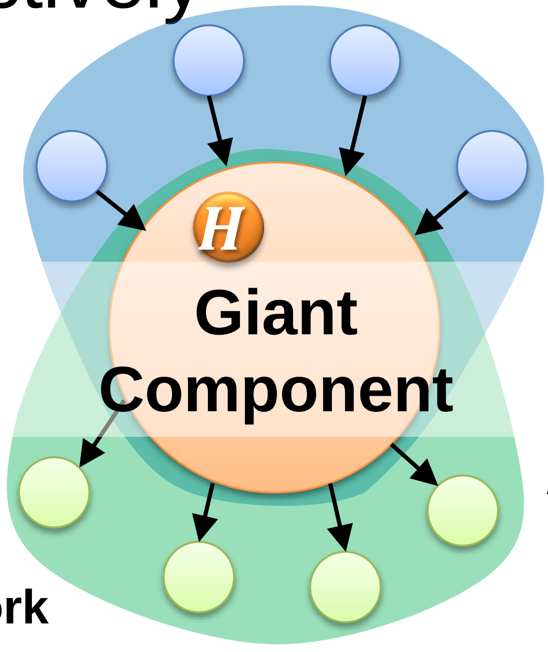
(# of descendants of H) **4**

⇧ Precomputed



Is Pruned BFS Really Effective?

- For **Path Graphs**
  - Pruned BFS is **NOT** effective $\Theta(|V|^2)$
- But, for **Social Networks**
  - Pruned BFS works effectively
  - since there is a **hub** (or **giant component**)



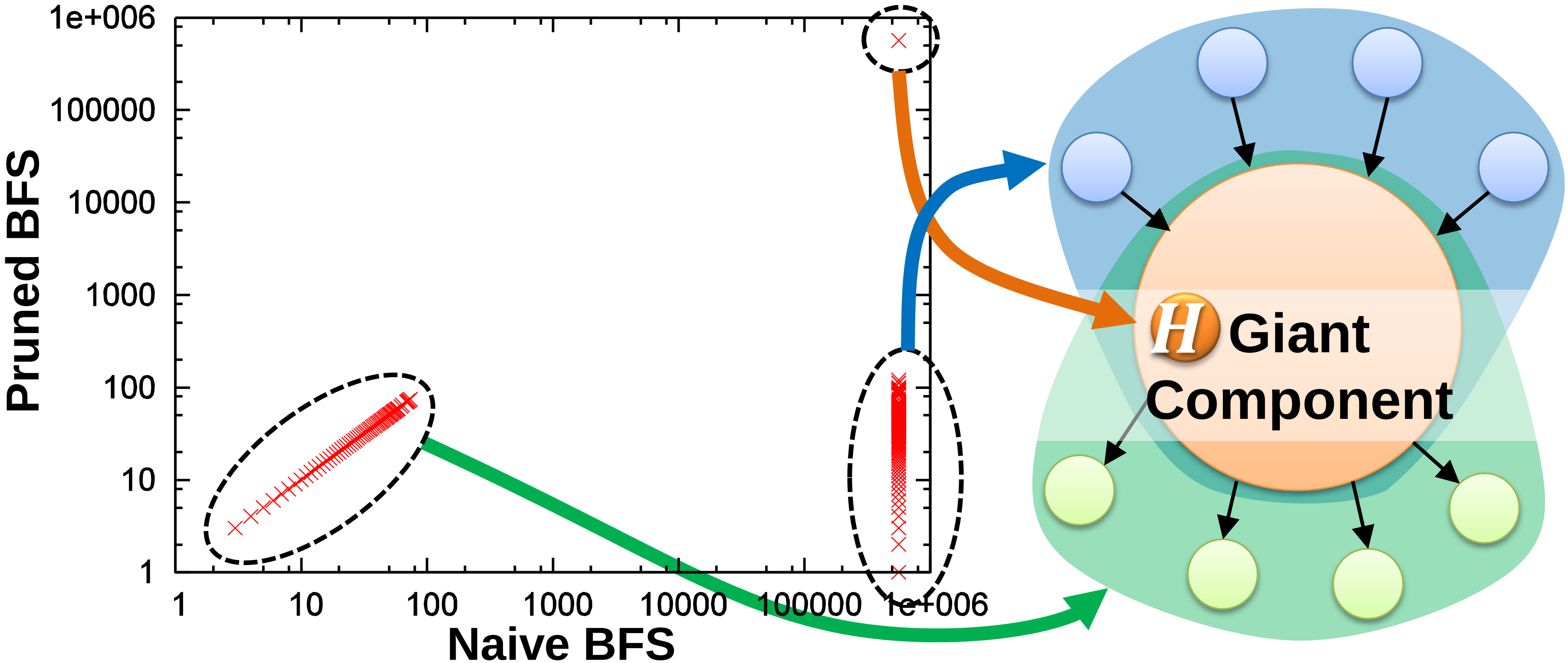
A social network

A path graph

Effect of Pruned BFS on Social Networks

(LiveJournal dataset, $|V| = 4.8$M, $|E| = 69$M, $p_e = 0.1 \ \forall e$)

- # of vertices visited during Naive & Pruned BFSs



- Average # of visited vertices (from each vertex):
  - **400,000** (Naive BFS) ⇨ **6** (Pruned BFS)

Experiments: Influence Spread

We set $p_e = P$ for every edge. Size of seed set = 50

Dataset	Ours (this work)	StaticGreedy DU [Cheng+'13]	IRIE [Jung+'12]	PMIA [Chen+'10]	SAEDV [Jiang+'11]
DBLP ($P = 0.01$)	**332**	330	323	317	76
DBLP ($P = 0.1$)	**100076**	--	99533	99505	99579
LiveJournal ($P = 0.01$)	**47527**	--	41906	40544	26066
LiveJournal ($P = 0.1$)	**1686629**	--	1682436	--	1682242

significantly better

Dataset	$\|V\|$	$\|E\|$
DBLP	655K	2.0M
Live Journal	4.8M	69M

- Ours & StaticGreedyDU give the best results

Experiments: Running Time [s]

We set $p_e = P$ for every edge. Size of seed set = 50

Dataset	Ours (this work)	StaticGreedy DU [Cheng+'13]	IRIE [Jung+'12]	PMIA [Chen+'10]	SAEDV [Jiang+'11]
DBLP ($P = 0.01$)	27	117	77	**4**	388
DBLP ($P = 0.1$)	**52**	OOM	77	289	388
LiveJournal ($P = 0.01$)	**327**	OOM	1622	500	1275
LiveJournal ($P = 0.1$)	**663**	OOM	1635	OOM	1294

- As fast as heuristics
- Robust against value of P

Dataset	$\|V\|$	$\|E\|$
DBLP	655K	2.0M
Live Journal	4.8M	69M

Environment: Intel Xeon X5670 (2.93GHz), 48GB, Language: C++

Future Work

- Applying other models
- Parallelization
- Analysis of Pruned BFS on social networks

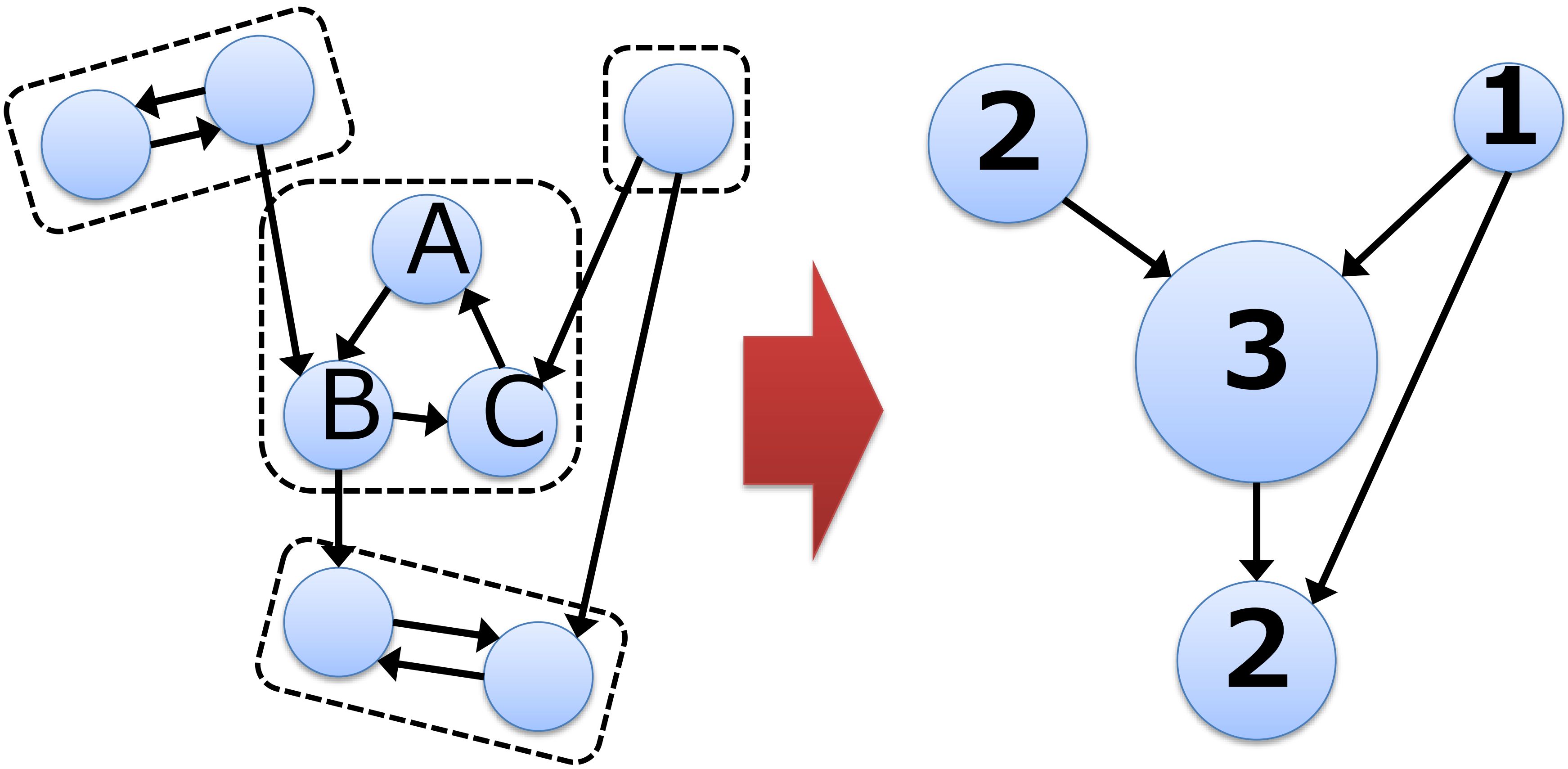
Supplement

Running Time [s] for Each Variant of Our Method

Dataset	Pruned BFS + Technique 2	Naive BFS + Technique 2	Pruned BFS	Naive BFS
DBLP ($\boldsymbol{P = 0.01}$)	27	26	149	158
DBLP ($\boldsymbol{P = 0.1}$)	54	3036	306	3275
LiveJournal ($\boldsymbol{P = 0.01}$)	327	1934	2176	3820
LiveJournal ($\boldsymbol{P = 0.1}$)	634	272518	2426	272973

Construct a Vertex-weighted DAG from a Random Graph

Strongly Connected Component Decomposition



Other Models for Information Diffusion

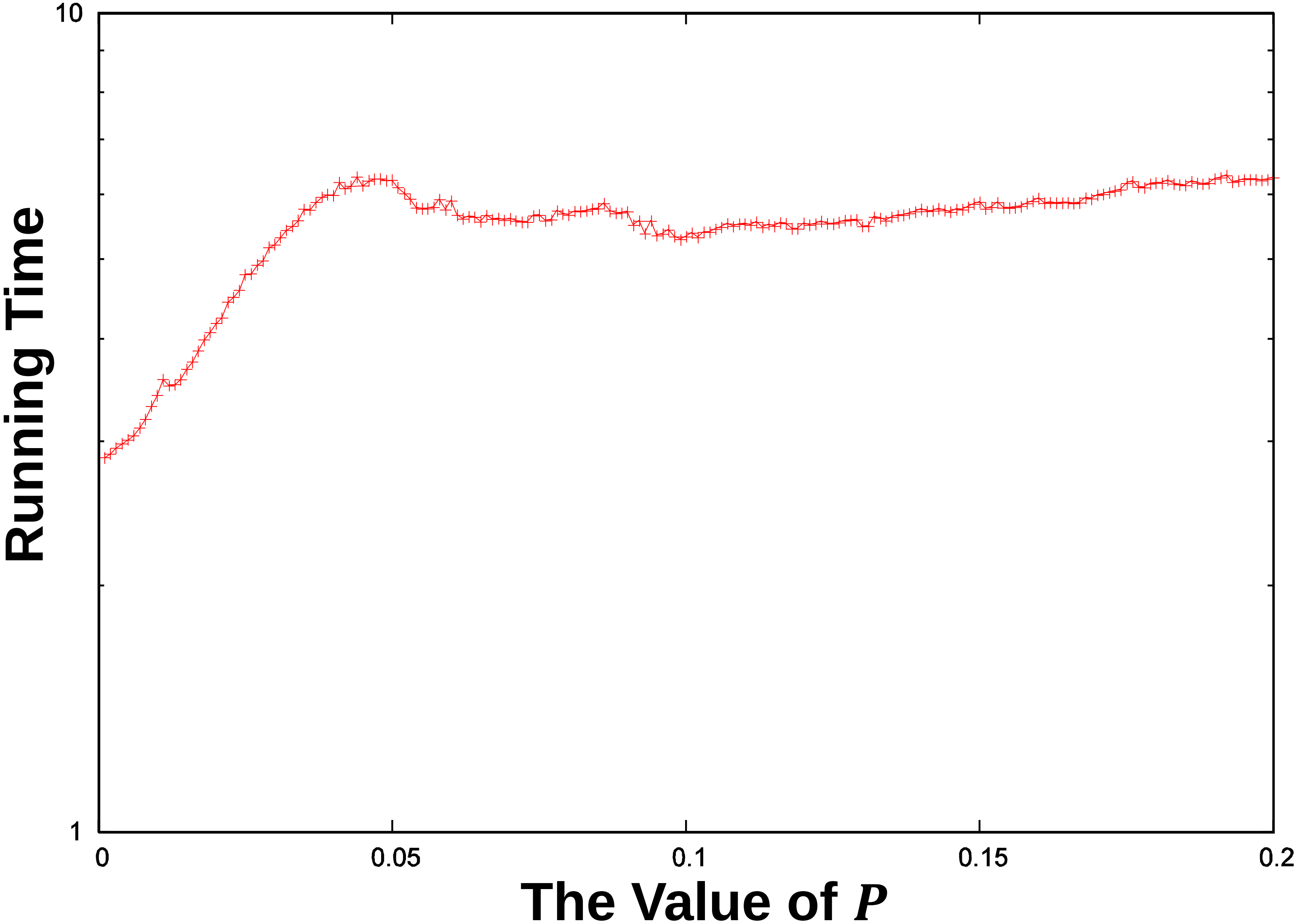
- Linear Threshold Model [Kempe, Kleinberg, Tardos. KDD'03]
  - Inactive vertex v becomes active if

$$\sum_{u\text{: active neighbor of } v} q_{uv} \geq \theta_v$$

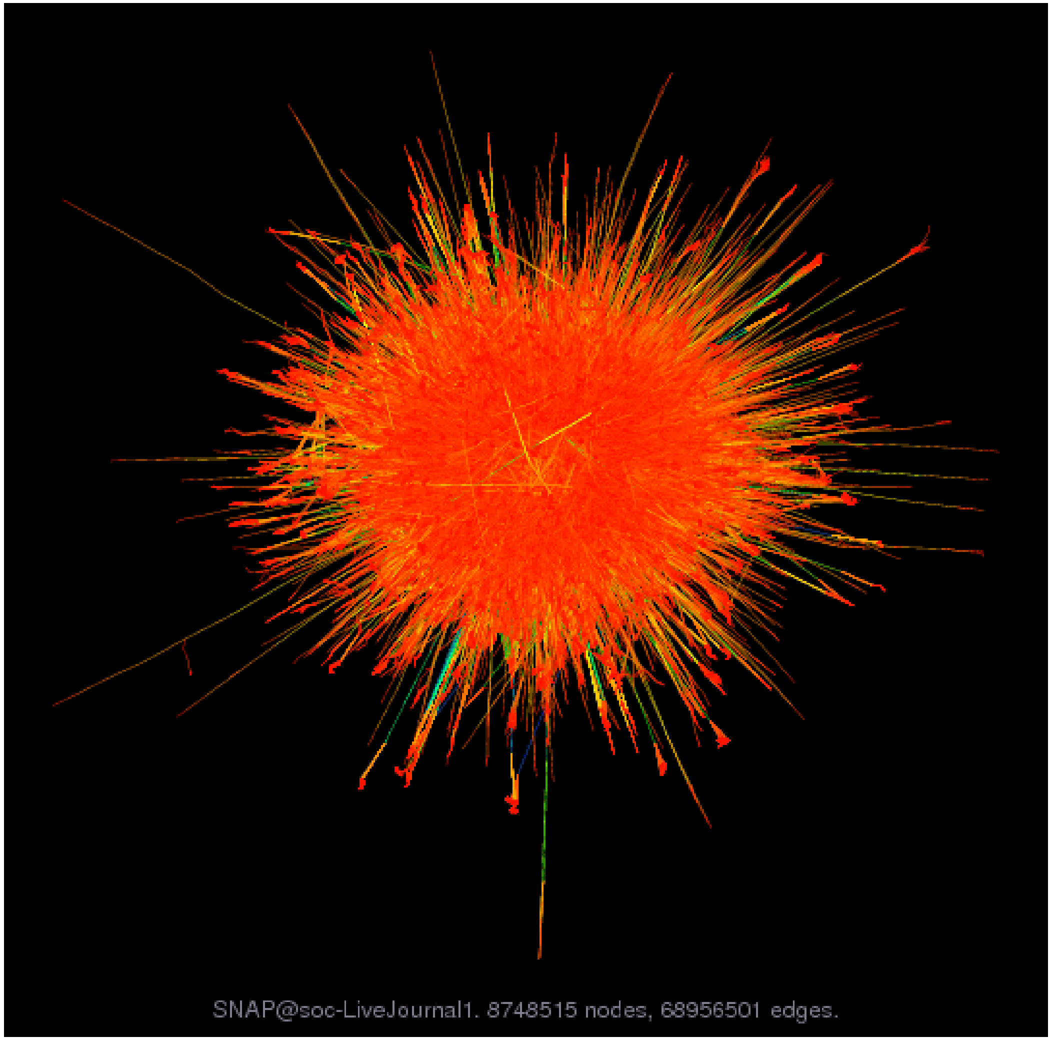
    - θ_v: Threshold chosen from $[0,1]$ uniformly at random
  - Equivalent to reachability tests on random graphs

- Independent Cascade with Meeting Events [Chen, Lu, Zhang. AAAI'12]
  - Maximizing the influence spread within a given **deadline**
  - We have to consider **shortest paths** (not only reachability)

Running Time for Each Value of *P*



A Social Network



http://www.cise.ufl.edu/research/sparse/matrices/SNAP/soc-LiveJournal1.html