Influence Learning and Maximization

George Panagopoulos
École Polytechnique
Institut Polytechnique de Paris

Fragkiskos D. Malliaros
CentraleSupélec, Inria
Paris-Saclay University

ICWE 2021
Biarritz, France
21ST INTERNATIONAL CONFERENCE ON WEB ENGINEERING

May 18, 2021

Slides at: http://fragkiskos.me/projects/influence_learning_tutorial/

Acknowledgements

- Partially based on material by
  - Jure Leskovec (Stanford University)
  - B Aditya Prakash and Naren Ramakrishnan (Virginia Tech)
  - Cigdem Aslay (Aarhus Univ.), Laks V.S. Lakshmanan (UBC), Wei Lu (LinkedIn), Xiaokui Xao (NUS)
  - David Kempe (USC)
  - Amit Goyal (University of British Columbia)
  - Manuel Gomez Rodriguez (MPI-SWS), Le Song (Georgia Tech)

Thank you!

What We Will Cover

- Motivate the problem with real-life applications
- Go through different approaches
- Describe the most prominent/representative solution of each approach
- Briefly outline similar solutions
- Mention theory but won't dive deep into it
- Focus on the advantages and disadvantages

Let’s make interactive,
so please ask questions whenever you want!

Outline of the Tutorial

Part I. Introduction

Part II. Traditional influence maximization

Part III. Influence and diffusion learning

Part IV. Learning influence maximization

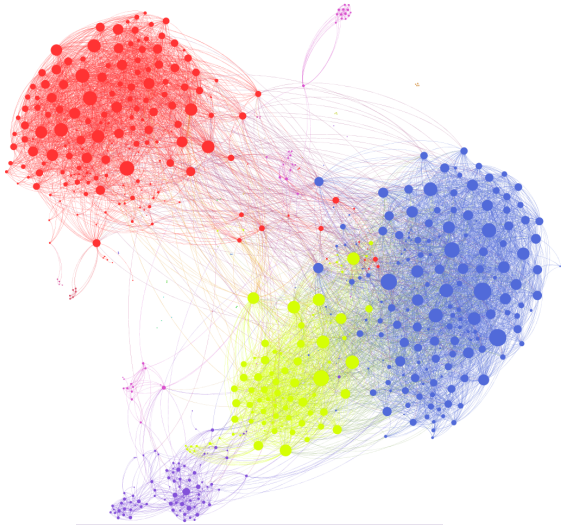
Part V. Online influence maximization

Part VI. Summary and open challenges

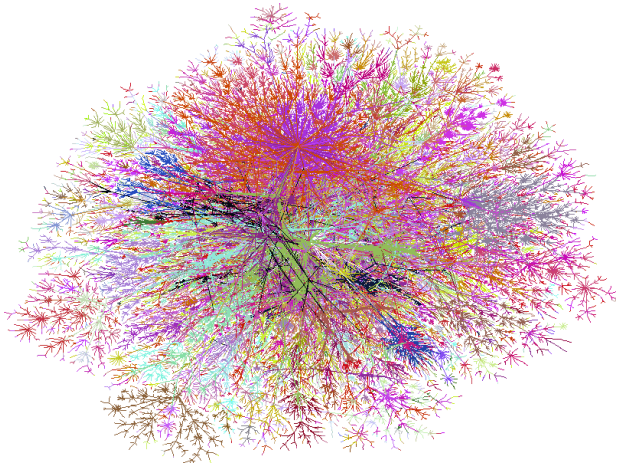
Part I. Introduction

- What is influence
- Exemplary applications
- Models of information diffusion
- Influencer identification

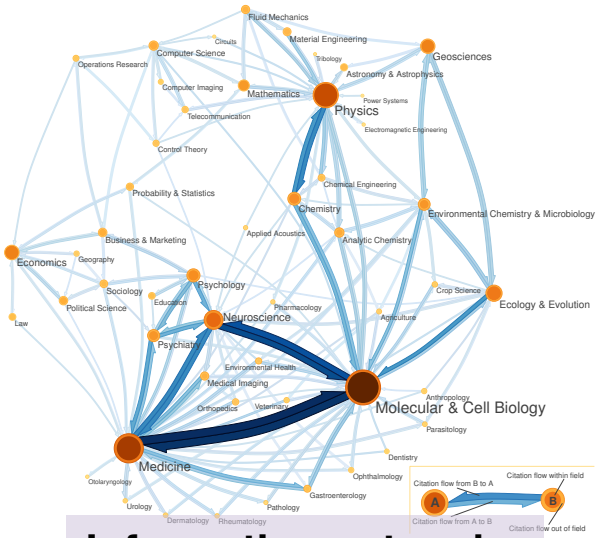
Activity-Rich Interconnected World



Social networks

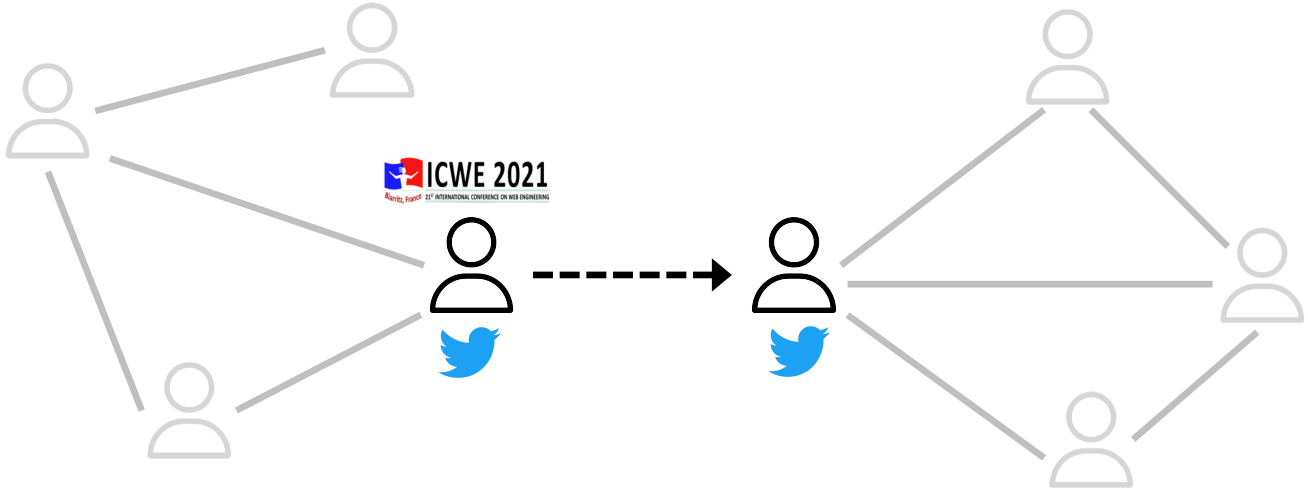


The Web graph



Information networks

Influence and Information Propagation



Individuals are **connected**, performing **activities**

friends
collaborators
followers
subscribers
…

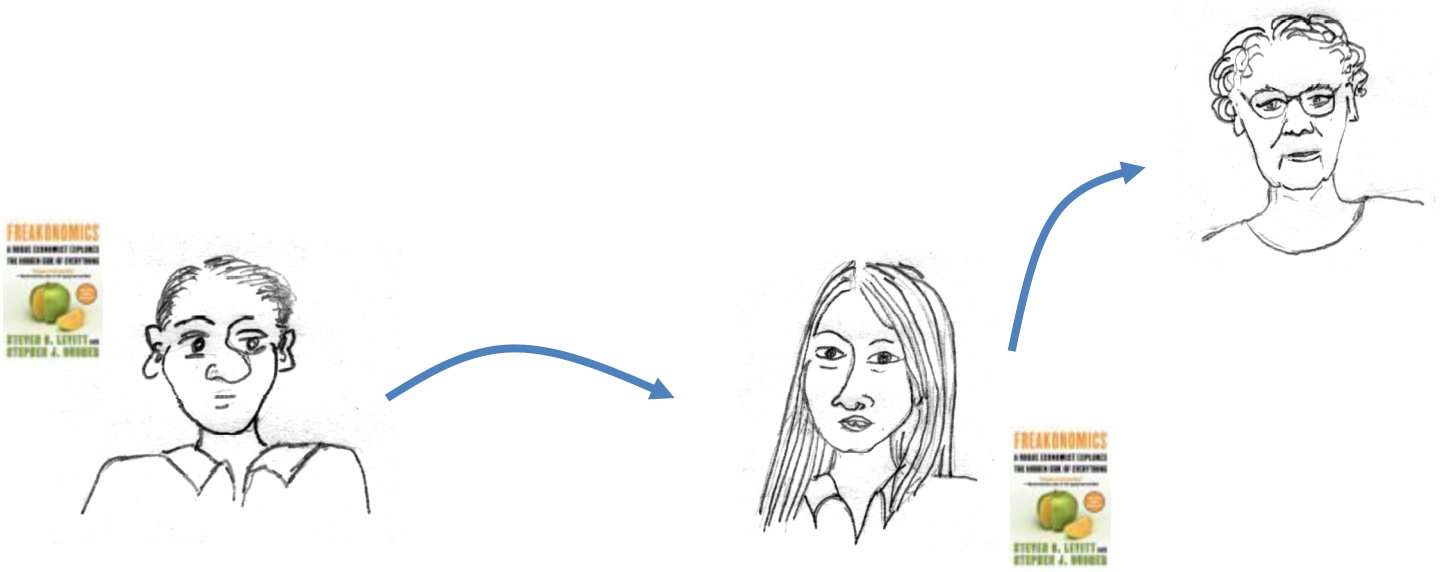
post
comment
share, retweet
like
…

Influence aims to measure the impact of interactions on the actions of individuals

Twitter & Facebook Post Sharing

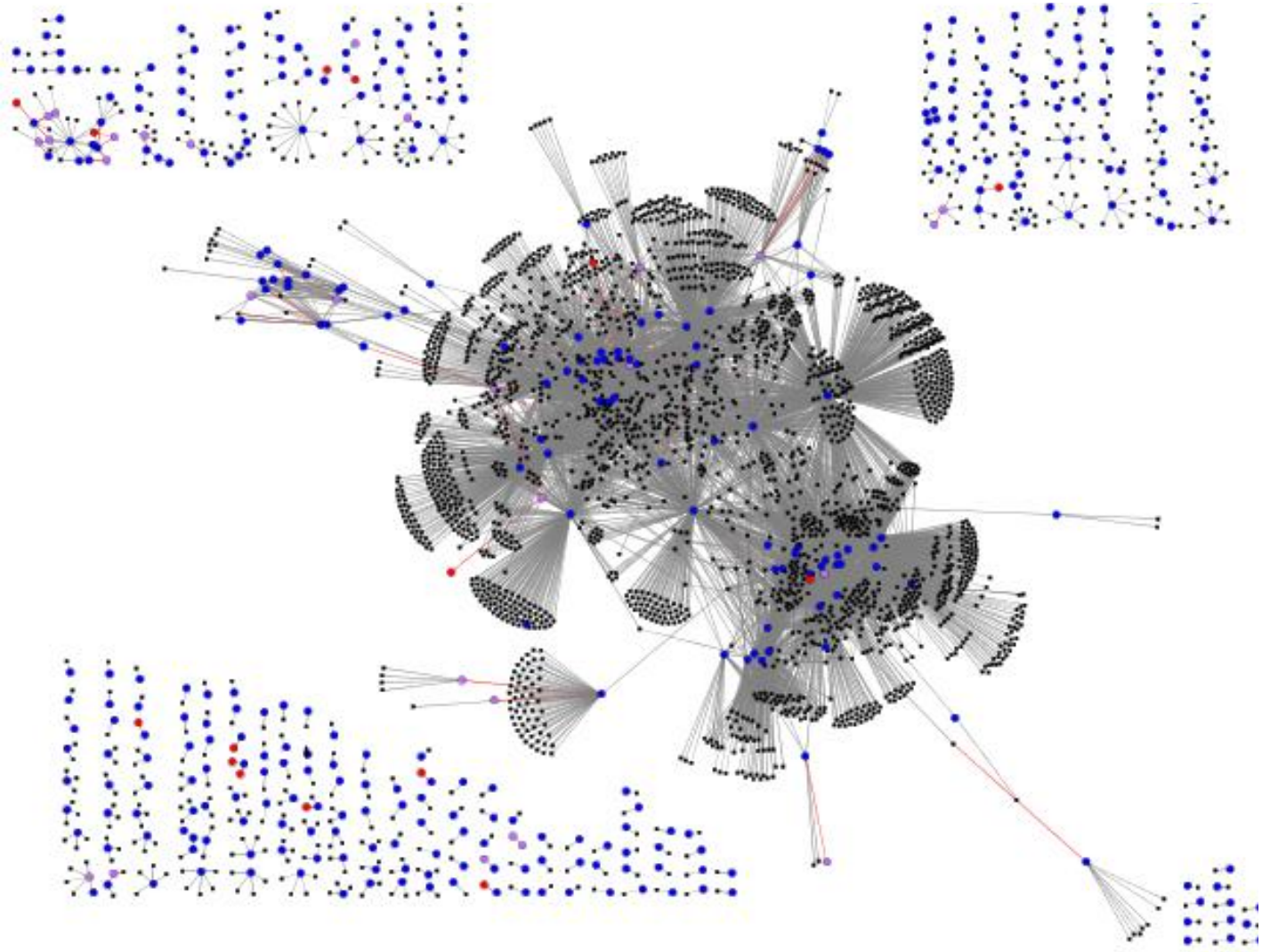
Influence and Diffusion in Viral Marketing (1/2)

- Product adoption by the "word of mouth" effect
  - Senders and followers of recommendations



[Leskovec et al., ACM TWeb '07], [Aral and Walker, Management Science '11]

Influence and Diffusion in Viral Marketing (2/2)



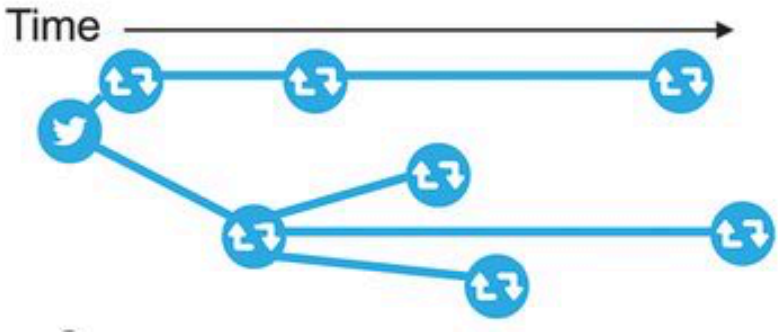
[Leskovec et al., ACM TWeb '07], [Aral and Walker, Management Science '11]

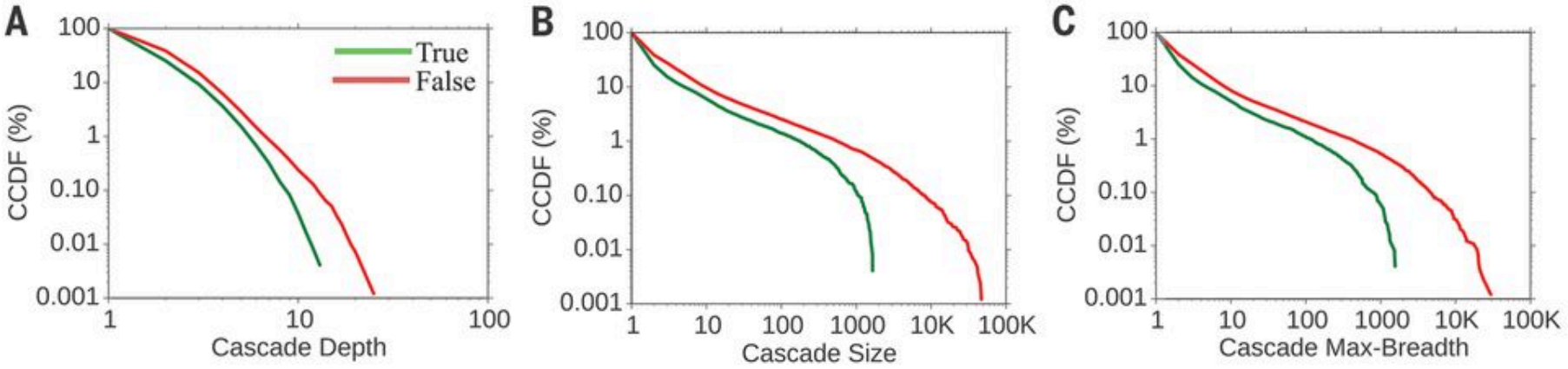
Social Media Influence and Marketing

Social Media Influence

[Prakash and Ramakrishnan, KDD '16]

News and Rumor Spreading





False news spread faster than the truth

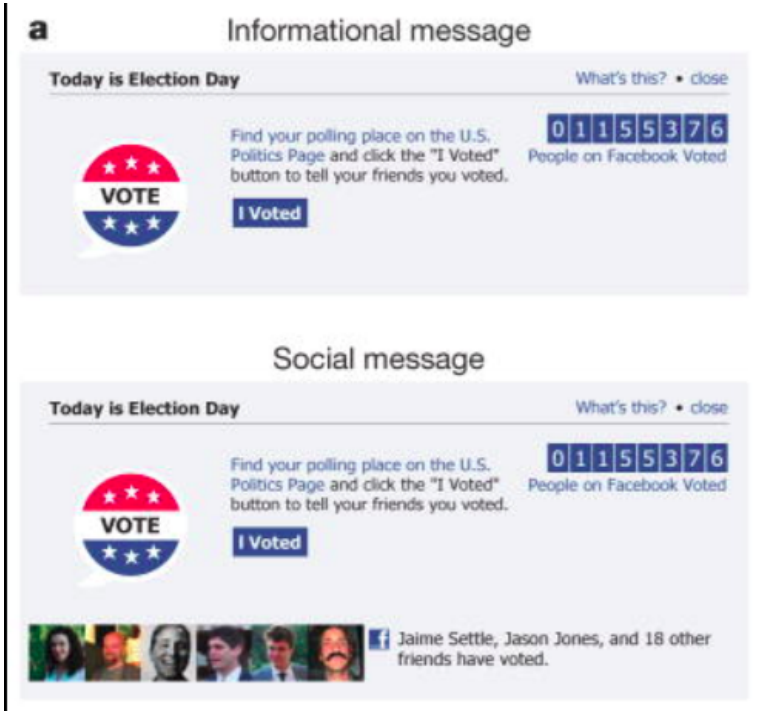
[Vosoughi et al., Science '18]

Facebook's 'Intent to Vote' Experiment (1/2)

Can someone use online social networks to shape a massive behavioral change?

- A simple news feed post was used to motivate voting
- Users could claim they voted by pressing "I voted" and access further information about the elections

- 61 million FB users split in three randomized groups:
  - Simple message (611K): the message included the count of FB users who reported voting
  - Social message (60M): The message included a list of the users' friends that have also voted appears
  - Control group (613K): no message



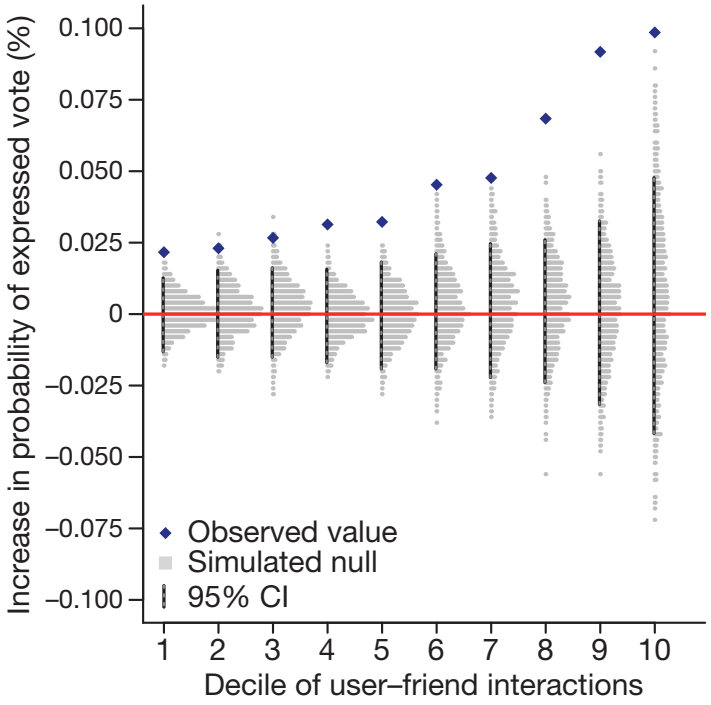
[Bond et al., Nature '12]

Facebook's 'Intent to Vote' Experiment (2/2)

- Users who received the social message were more likely to click "I voted" than users with the simple message
- Friends of users who received the social message were more likely to vote than friends of users with no message

- As the interaction increases, so does the observed per-friend effect of friend's treatment on a user's expressed voting

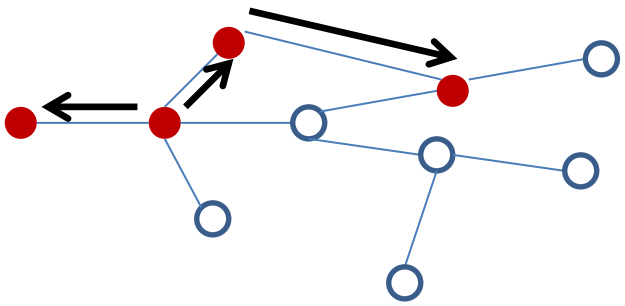
Online messages might influence a variety of offline behaviours

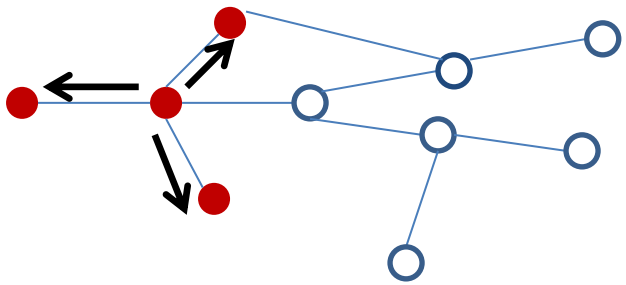


[Bond et al., Nature '12]

Information Propagation is Almost Everywhere

- Social collaboration
- Information Diffusion
- Viral Marketing
- Epidemiology and Public Health
- Cyber Security
- Human mobility
- Games and Virtual Worlds
- Bioinformatics
- …





Research Theme

ANALYSIS

Understanding

DATA

Large real-world networks & propagations

POLICY/ ACTION

Managing

[Prakash and Ramakrishnan, KDD '16]

Research Theme – Social Media

ANALYSIS

cascades in future?

DATA

Modeling Tweets spreading

POLICY/ ACTION

How to do better marketing?

[Prakash and Ramakrishnan, KDD '16]

Opportunities for Data Mining and Maching Learning

Algorithmic tools

and

Machine Learning models

to

(understand
maximize
predict)

influence spreading

in

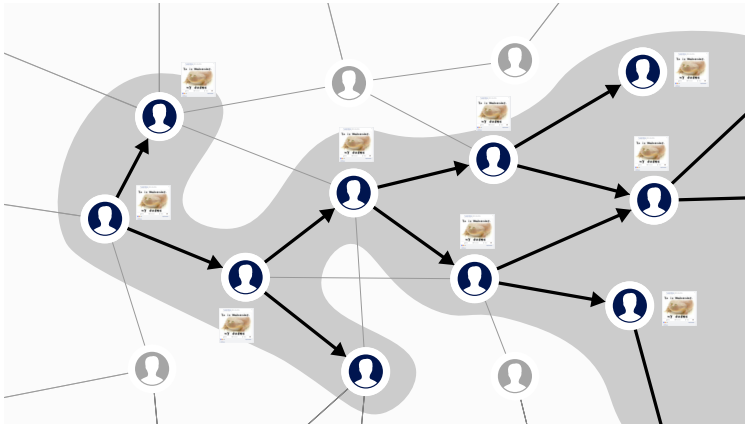
social and information networks

Part I
Part II

Part III
Part IV
Part V

What is it All About?

Part I. Basic models
for information diffusion and detection of influential spreaders



Part II. Influence maximization
find a seed set S of size k that maximizes the infuence spread

Part III. Influence and diffusion learning
analyze real information cascades to boost influence maximization

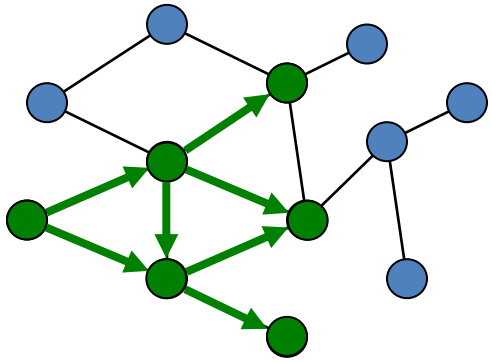
Part IV. Machine learning models
learn how to predict and maximize influence

Part V. Online influence maximization
learning influence probabilities and maximizing influence simultaneously

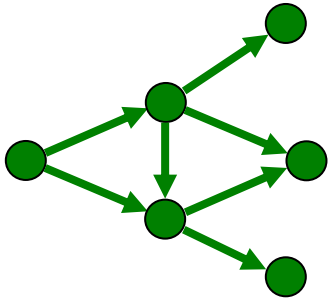
Information diffusion models

Information Cascades

- "Contagion" that spreads over the edges of the network
- It creates a propagation tree, i.e., a diffusion cascade



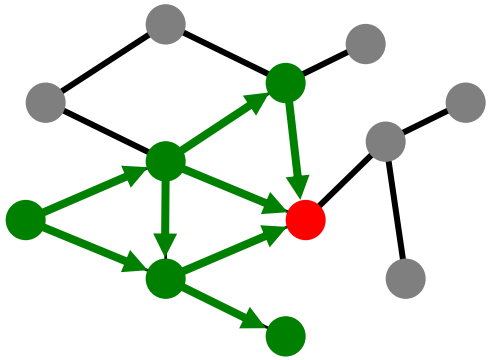
Network



Information cascade
(propagation graph)

How Do We Model Influence/Diffusion?

- Decision based models
  - Models of product adoption, decision making
    - A node observes decisions of its neighbors and makes its own decision
  - Example:
    - You attend this tutorial if k of your friends do so too
- Probabilistic models
  - Models of influence or disease spreading
    - An infected node tries to "push" the contagion to an uninfected node
  - Example:
    - You are influenced with some probability from each active (i.e., influenced) neighbor in the network

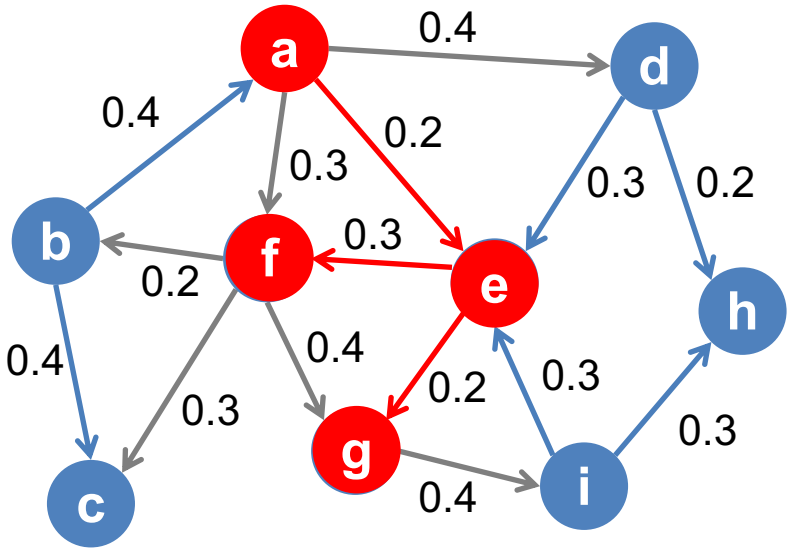


[Easley and Kleinberg, Networks, Crowds, and Markets '10]

Independent Cascade (IC) model

Independent Cascade Model (1/2)

- Initially some nodes S are active
- Each edge (u,v) has probability (weight) p_{uv}



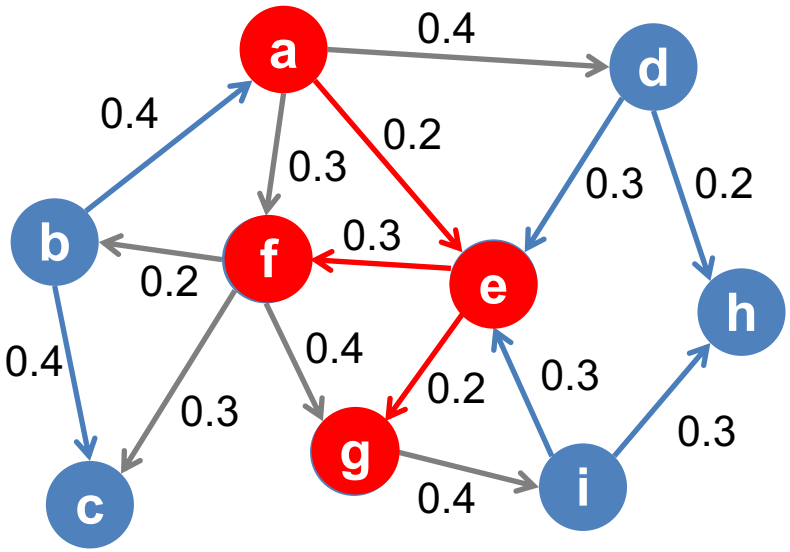
Each activated node has a single chance to activate its neighbors

- When node u becomes influenced
  - It activates each out-neighbor v with probability p_{uv}
- Influence spreads through the network

[Kempe et al., KDD '03]

Independent Cascade Model (2/2)

- The model is simple but requires parameters proportional to the # of edges
  - Estimating influence weights from the data
  - Next part of the tutorial



- **Solution 1:** Make all edges have the same weight
  - Simplistic and unrealistic approach is many practical applications
  - Similar to the SIR (Susceptible-Infected-Recovered) model in epidemiology
- **Solution 2:** Degree-based probabilities or similar
  - E.g., edge from u to v has probability $1/k_v$ of activating v (k_v : degree of v)

Linear threshold model (LT)

Linear Threshold Model

- A node v has a random threshold $\theta_v \sim U[0,1]$
- A node v is influenced by each neighbor w according to a weight b_{vw} such that

$$\sum_{w \in Nb(v)} b_{vw} \leq 1$$

- A node v becomes active when at least θ_v fraction of its neighbors are active

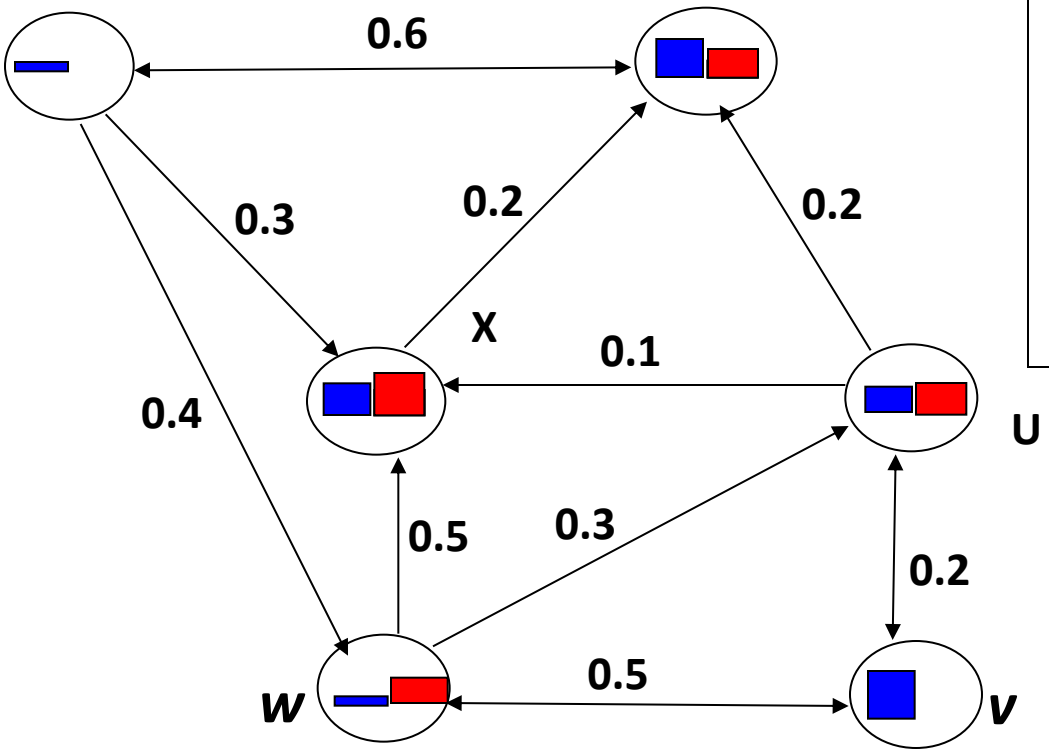
$$\sum_{w \in X(v)} b_{vw} \geq \theta_v$$

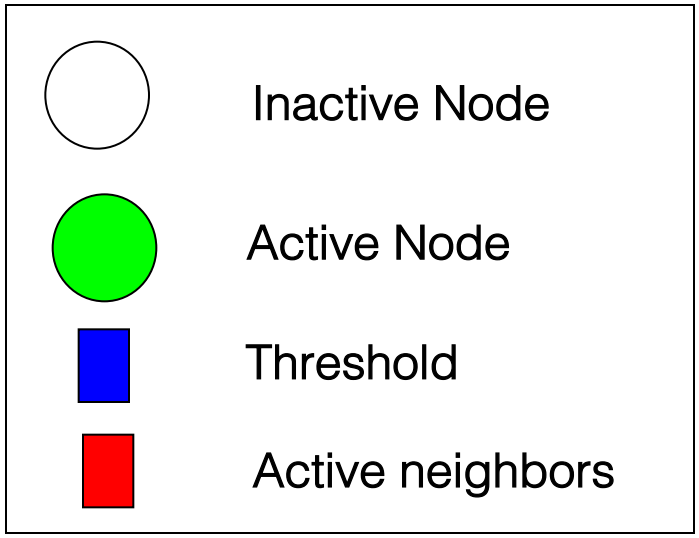
θ_v: fraction of neighbors of v that should be active in order for v to become active

X(v): set of active neighbors of v

[Kempe et al., KDD '03]

Example





Stop!

Become active if: $\sum_{w \in X(v)} b_{vw} \geq \theta_v$

Identification of influential spreaders

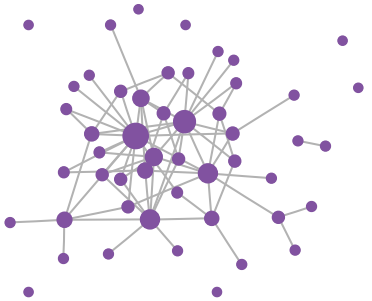
Identification of Influential Nodes: the Process (1/2)

- Typically, a two-step approach:
  1. Consider a topological or centrality criterion of the nodes of the network
  2. Rank the nodes accordingly
  3. The top-ranked nodes are candidates for the most influential ones
  4. Simulate the spreading process over the network to examine the performance of the chosen nodes
     - E.g., using the IC or the LT model

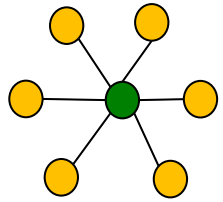
[Sei and Makse '13]

Identification of Influential Spreaders

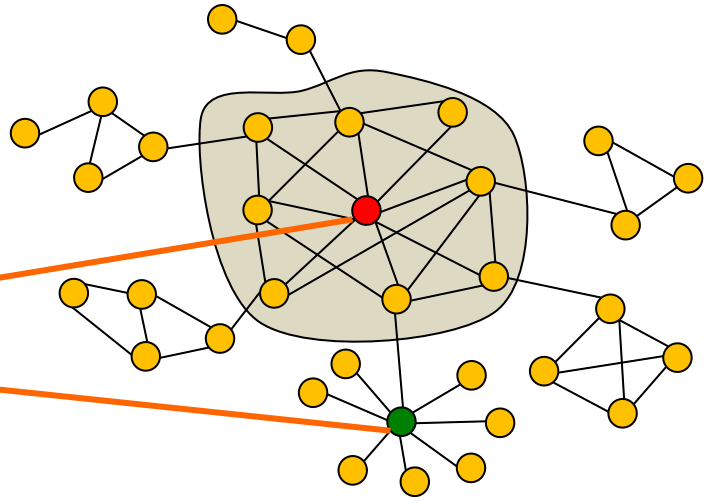
- **Straightforward approach:** consider degree centrality
  - High degree nodes are expected to be good spreaders
  - Hub nodes can trigger big cascades



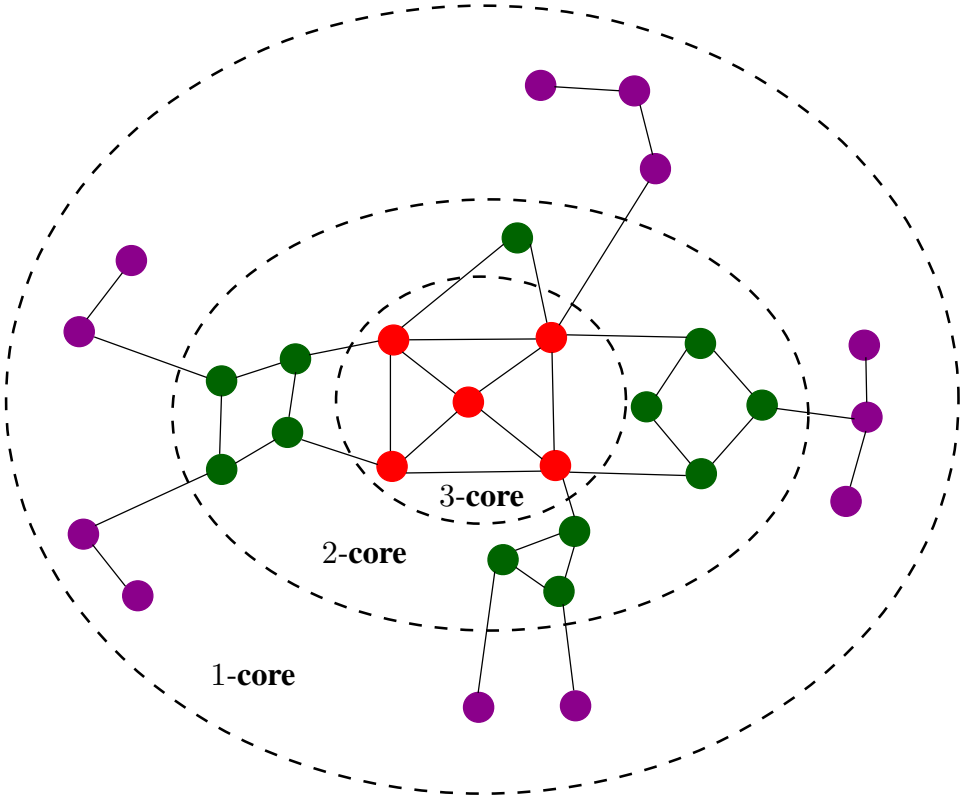
- However: degree is a **local criterion**
  - Bad instance: **star** subgraph



- **Core-periphery structure** of real-world networks



Degeneracy and k-Core Decomposition



Core number $\mathbf{c_i = 1}$

Core number $\mathbf{c_i = 2}$

Core number $\mathbf{c_i = 3}$

Graph Degeneracy $\delta^*(\mathbf{G}) = \mathbf{3}$

$\mathbf{G_0 = G}$

$\mathbf{G_1}$ = 1-core of $\mathbf{G}$

$\mathbf{G_2}$ = 2-core of $\mathbf{G}$

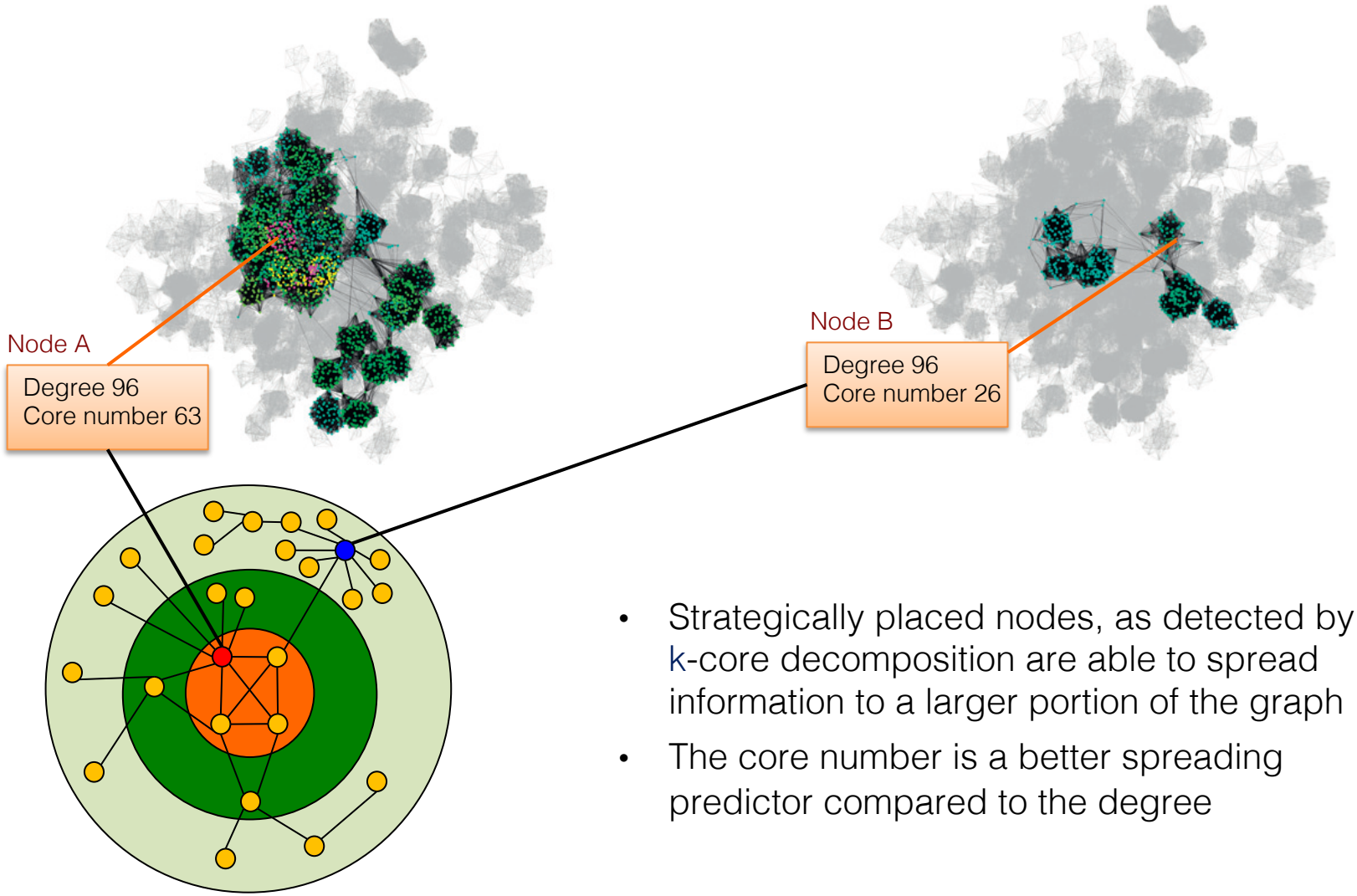
$\mathbf{G_3}$ = 3-core of $\mathbf{G}$

$\mathbf{G_0 \supseteq G_1 \supseteq G_2 \supseteq G_3}$

Fast detection of **dense** and **cohesive** subgraphs

[Seidman, Social Networks '83] [Malliaros et al., VLDB Journal '20]

The k-core Decomposition Finds Good Spreaders



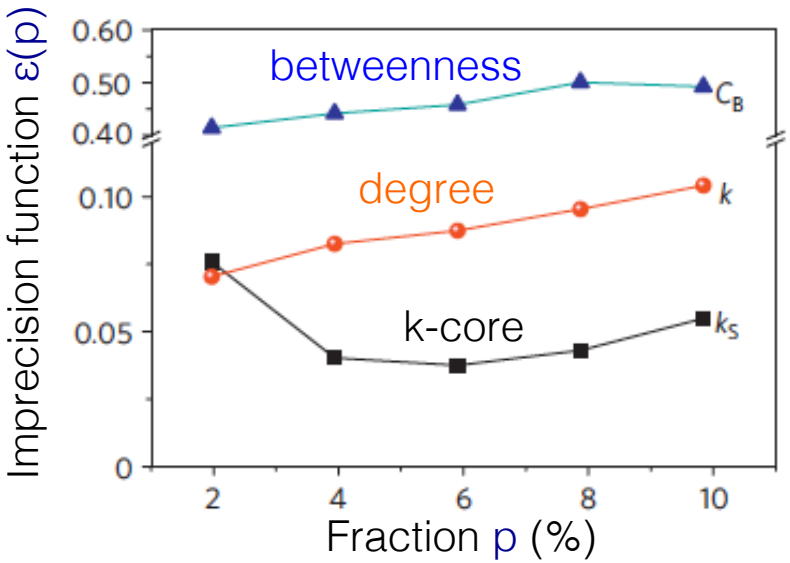
- Strategically placed nodes, as detected by the k-core decomposition are able to spread information to a larger portion of the graph
- The core number is a better spreading predictor compared to the degree

[Kitsak et al., Nature Physics '10]

How Close to the Optimal Spreading?

- Compute the imprecision function ε(p) that tests how close is each metric to the optimal spreading
  - For a given fraction p:
  1. Find p% of the most efficient spreaders and compute the average total spreading M_{eff}
  2. Find p% nodes with the highest core number and compute the average total spreading M_{core}
  3. Repeat the same for degree and betweenness centrality

$$\varepsilon(p) = 1 - \frac{M_{core}(p)}{M_{eff}(p)}$$

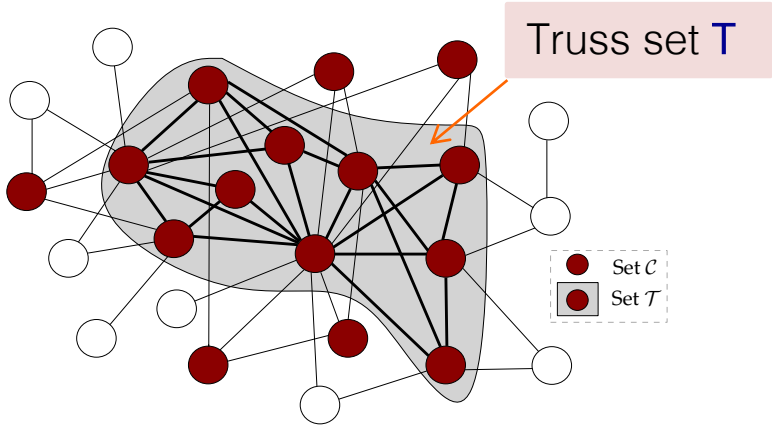


- The k-core is the most accurate spreading predictor
- Degree outperforms the global betweenness centrality

K-truss Decomposition (1/2)

- The k-core decomposition often returns a relatively large number of candidate influential spreaders
  - Only a small fraction corresponds to highly influential nodes
- How to further refine the set of the most influential nodes?
- Apply the K-truss decomposition [Cohen, TR '08], [Wang and Cheng, VLDB '12]
  - **Triangle-based** extension of the k-core decomposition
  - Each edge of the K-truss subgraph participates in at least K-2 triangles

C: set of nodes in the maxima k-core subgraph
T: set of nodes in the maximal K-truss subgraph



[Malliaros, Rossi & Vazirgiannis, Scientific Reports '16]

K-truss Decomposition (2/2)

	Method	Time Step 2	4	6	8	10	*Final step*	σ	*Max step*
EMAIL-ENRON	**truss**	8.44	46.66	204.08	418.77	355.84	2,596.52	136.7	33
	core	4.78	31.97	152.55	367.28	364.13	2,465.60	199.6	37
	top degree	6.89	34.13	155.48	360.89	357.08	2,471.67	354.8	36
EPINIONS	**truss**	4.17	19.70	75.04	204.14	329.08	2,567.69	227.8	37
	core	3.45	14.72	55.27	158.56	280.03	2,325.37	327.2	43
	top degree	4.22	16.03	58.84	166.23	289.49	2,414.99	331.7	47
WIKI-VOTE	**truss**	2.92	6.92	15.27	28.73	42.46	560.66	114.9	52
	core	1.92	4.78	10.65	20.66	32.40	466.01	104.5	57
	top degree	2.43	5.46	12.05	23.05	35.55	502.88	104.5	62
EMAIL-EUALL	**truss**	11.62	62.25	240.97	584.87	725.42	5,018.52	487.94	36
	core	9.85	40.82	158.72	433.81	644.76	4,579.84	498.71	38
	top degree	17.96	39.93	144.69	503.18	548.25	4,137.56	1,174.84	39

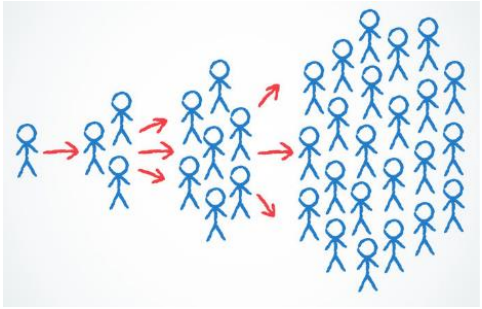
- **truss**: avg. spreading of the nodes of T
- **core**: avg. spreading of the nodes of C – T
- **top deg**: avg. spreading of the |C – T| top degree nodes

- The **truss** method achieves higher infection rate during the first steps
- Better and faster spreading

[Malliaros, Rossi & Vazirgiannis, Scientific Reports '16]

Part I Summary

- Combine centrality criteria with models of information propagation
- The core decomposition provides an effective way to detect influential spreaders
  - Outperforms more ‘complex’ centrality criteria
  - How to select multiple influential spreaders?
- Heuristic methods
  - No theoretical guarantees about the performance of the metrics
  - E.g., what will happen if all the nodes of a graph have ~ the same core number?



Outline of the tutorial

Part I. Introduction

Part II. Traditional influence maximization

Part III. Influence and diffusion learning

Part IV. Learning influence maximization

Part V. Online influence maximization

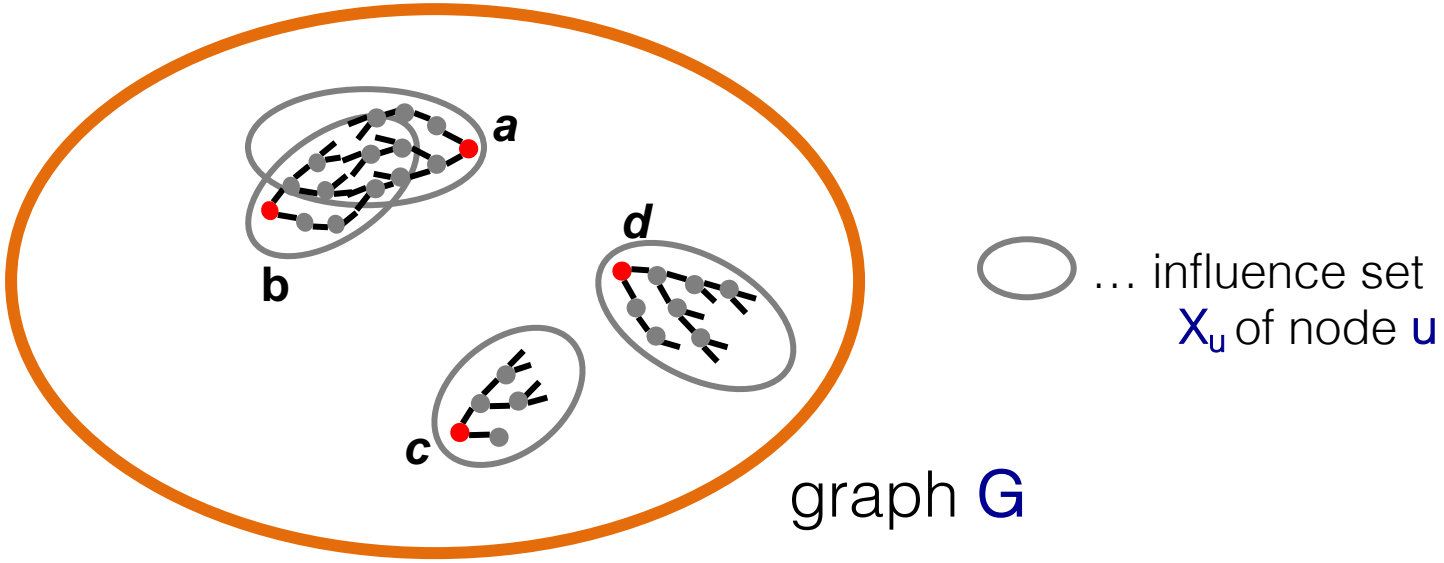
Part VI. Summary and open challenges

Part II. Traditional IM

- The IM problem
- The Greedy algorithm
- Scalable algorithms (CELF, Reverse Influence Sampling, Sketch-based Influence Maximization)
- Overview of scalable heuristics

Most Influential Set of Nodes

- S: is the initial active set
- σ(S): The expected size of final active set (expected influence)



- Set S is more influential if σ(S) is larger

$$\sigma(\{a,b\}) < \sigma(\{a,c\}) < \sigma(\{a,d\})$$

[Domingos and Richardson, KDD '01]

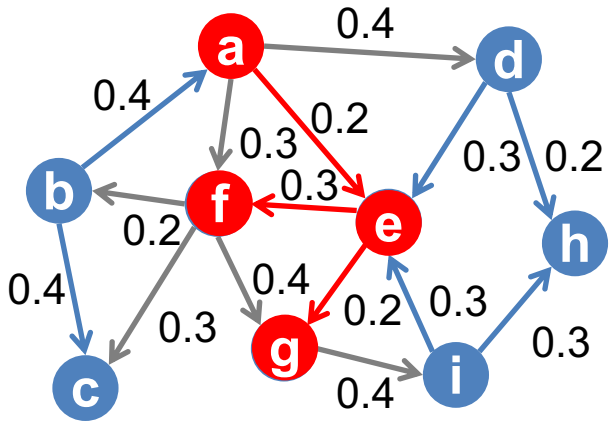
The Influence Maximization Problem

Problem (k is user-specified parameter - budget)

- Given a graph G=(V, E) and a diffusion model to simulate spreading
- Find a seed set S of k nodes that maximizes spreading σ(S)
  - The **expected** influence spread $\sigma(S) = \sum_{g \subseteq G} p_g \sigma_g(S)$

Complexity of IM

- The IM problem is NP-Hard under both the IC and LT models
  - Reduction from instances of the set cover problem



[Kempe et al., KDD '03]

Properties of the Spread Function

- Function σ(·) has with two properties:
  - σ(·) is monotone: (activating more nodes doesn't hurt)

$$\sigma(S \cup \{v\}) \geq \sigma(S), \forall v \in V, \forall S \subseteq V$$

  - σ(·) is submodular: (activating each additional node helps less)
    adding an element to a set gives less improvement than adding it to one of its subsets:

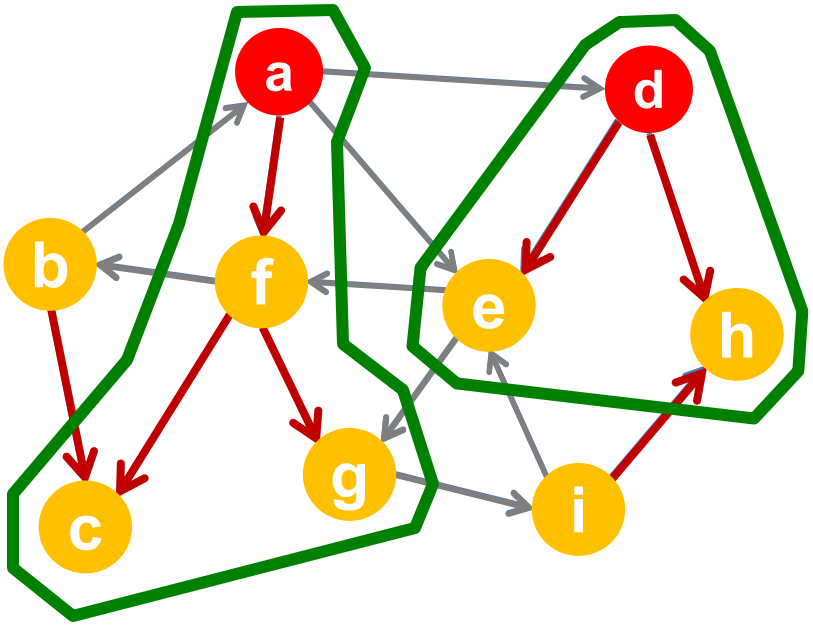
$$\underbrace{\sigma(S \cup \{v\}) - \sigma(S)}_{\text{Marginal gain of adding a node to a small set}} \geq \underbrace{\sigma(T \cup \{v\}) - \sigma(T)}_{\text{Marginal gain of adding a node to a large set}} \qquad S \subseteq T \subseteq V$$

If σ(·) is monotone and submodular, then we can have an approximate solution with theoretical guarantee

[Kempe et al., KDD '03]

Submodularity of Influence Spread σ(S)

- The live-edge model
  - Flip all the coins at the beginning and record which edges are activated successfully
    - Sample a random subgraph
  - Now we have a deterministic graph
  - Possible world of a probabilistic graph
    - $2^{|E|}$ possible outcomes



- What is the influence set X_u of node u?
  - The set of nodes reachable by **live-edge paths** from u

Influence sets for realization i :
$X_a^i = \{a,f,c,g\}$
$X_b^i = \{b,c\}$,
$X_c^i = \{c\}$
$X_d^i = \{d,e,h\}$
…

[Kempe et al., KDD '03]

The Greedy Approximation Algorithm

The Greedy Algorithm

Input: graph G=(V, E), parameter k and an influence model

Output: set S

- Start with $S \leftarrow \emptyset$
- While $|S| < k$
  - Take node u that $\arg\max_{u \in V \setminus S} \sigma(S \cup \{u\}) - \sigma(S)$
  - Let $S \leftarrow S \cup \{u\}$

Theorem: The Greedy algorithm is a (1 – 1/e) approximation

- The algorithm will find a set S for which σ(S) > 0.63*σ(OPT), where OPT is the globally optimal set
- The resulting set S activates at least (1- 1/*e*) > 63% of the number of nodes that any size-k set S could activate
- (e is the base of the natural logarithm, e ≈ 2.71)

[Kempe et al., KDD '03]

Evaluating σ(S) and Overall Complexity

- How to evaluate σ(S) for a set S?
  - Exact computation of σ(S) is #P-hard (class of counting problems)
- Very good estimation by simulation
  - Use Monte Carlo simulations (sampling-based approach)
  - Repeating the diffusion process often enough (polynomial in n)
  - Greedy algorithm is now a (1-1/e - ε)-approximation, ε>0
  - ε depends on the number of possible worlds (simulations)

- Complexity of the Greedy algorithm: O(n·k·R·m)
  - n: number of nodes
  - m: number of edges
  - k: size of the seed set
  - R: number of simulations (i.e., number of possible words)

Scalable Algorithms – CELF

CELF (Cost-Effective Lazy Forward Selection)

- In step i+1 of the Greedy algorithm

$$S_{i+1} = \arg\max_u \sigma(S_i \cup \{u\}) - \sigma(S_i)$$

  - Node u maximizes the marginal gain

$$\delta_i(u) = \sigma(S_i \cup \{u\}) - \sigma(S_i)$$

- From the submodularity property, we have that

$$\sigma(S_i \cup \{u\}) - \sigma(S_i) \geq \sigma\left(S_j \cup \{u\}\right) - \sigma\left(S_j\right) \text{ for } i < j$$

  - For every u: $\delta_i(u) \geq \delta_j(u)$, for $i < j$ since $S_i \subseteq S_j$

The marginal gains $\delta_i(u)$ only srink or remain the same as i increases

$\delta_i(u) \geq \delta_j(u)$

Activating node u in step i helps more than activating it at step j (j>i)

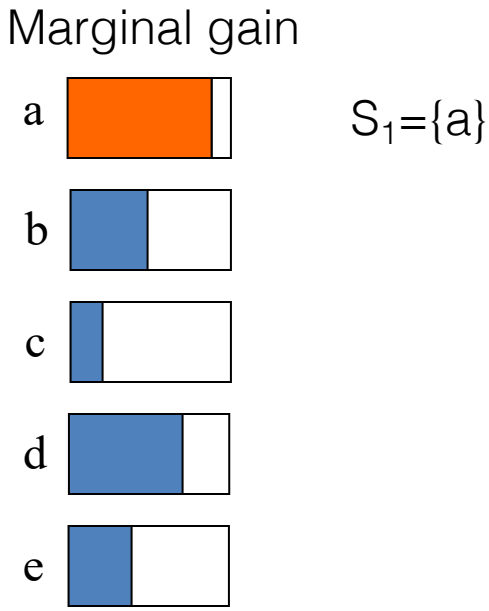
[Leskovec et al., KDD '07]

CELF – Lazy Hill Climbing

- Idea: use marginal gain δ_i as an upper-bound on δ_j (j > i)

CELF

1. Compute and sort the marginal gain δ_i of all nodes
2. Add the first node to the seed set
3. Compute the marginal gain from the top of the list
4. Re-sort every time you compute a node's new marginal gain



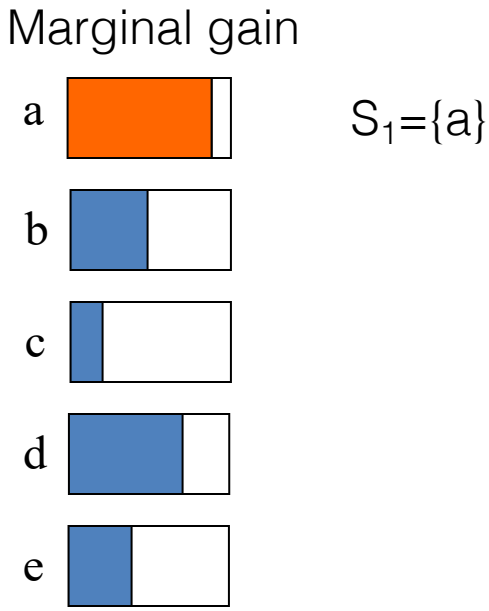
[Leskovec et al., KDD '07], [Leskovec, Stanford CS224W]

CELF – Lazy Hill Climbing

- Idea: use marginal gain δ_i as an upper-bound on δ_j (j > i)

CELF

1. Compute and sort the marginal gain δ_i of all nodes
2. Add the first node to the seed set
3. Compute the marginal gain from the top of the list
4. Re-sort every time you compute a node's new marginal gain



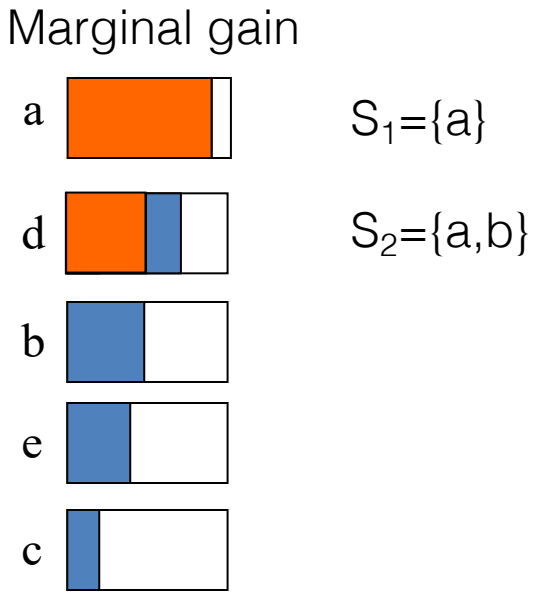
[Leskovec et al., KDD '07], [Leskovec, Stanford CS224W]

CELF – Lazy Hill Climbing

- Idea: use marginal gain δ_i as an upper-bound on δ_j (j > i)

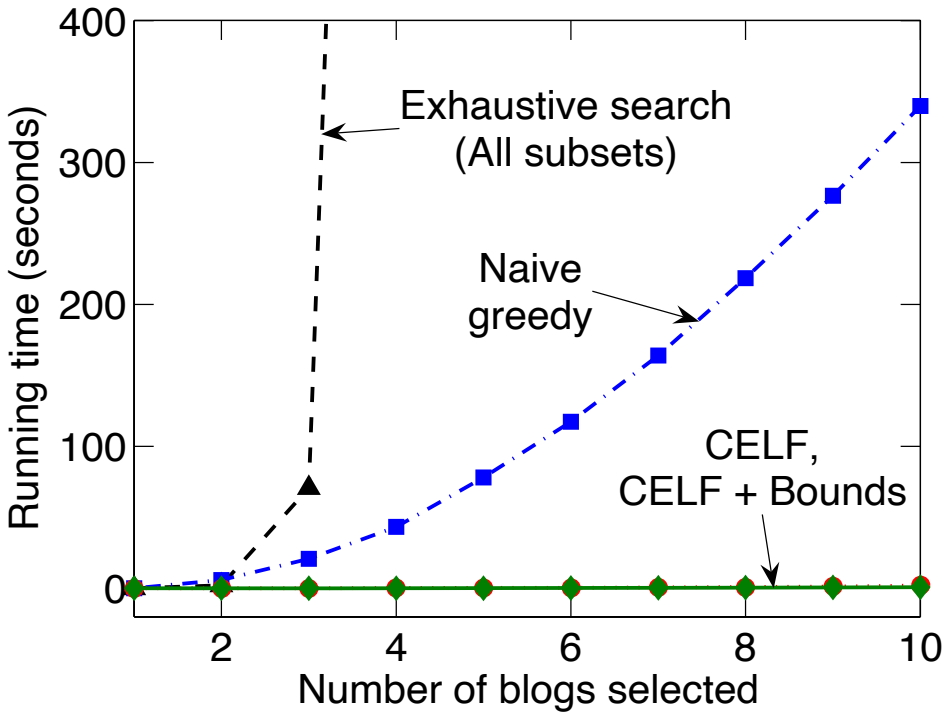
CELF

1. Compute and sort the marginal gain δ_i of all nodes
2. Add the first node to the seed set
3. Compute the marginal gain from the top of the list
4. Re-sort every time you compute a node's new marginal gain



[Leskovec et al., KDD '07], [Leskovec, Stanford CS224W]

Key Points for CELF



- Theoretical quarantess
- In the worst case, it performs as the Greedy
- In practice, 700x faster than the Greedy algorithm

[Leskovec et al., KDD '07]

Scalable Algorithms – Reverse Influence Sampling

Reverse Influence Sampling

- Algorithms that ensure $\left(1 - \frac{1}{e} - \epsilon\right)$ - approximation of the expected influence spread

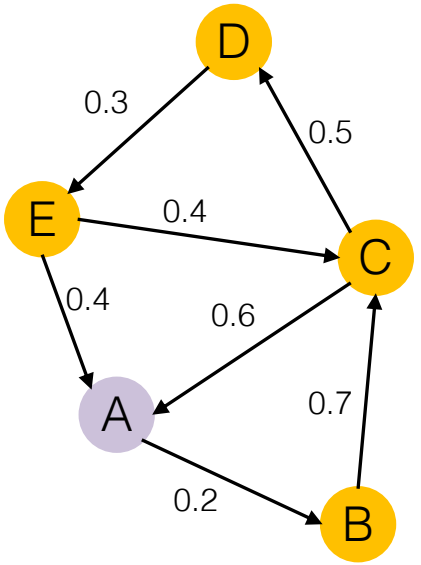
- Scalability (near-linear time) is achieved relying on the concept of Reverse Reachable Sets (RR set)

Reverse Reachable Sets (RR Sets)

- An RR set is a random sample of G
- Generation of RR sets by computing reachable nodes under the IC model

RR set = {A}

Start from a random node



[Borgs et al., SODA '14], [Aslay et al., Tutorial at WSDM '18]

Reverse Reachable Sets (RR Sets)

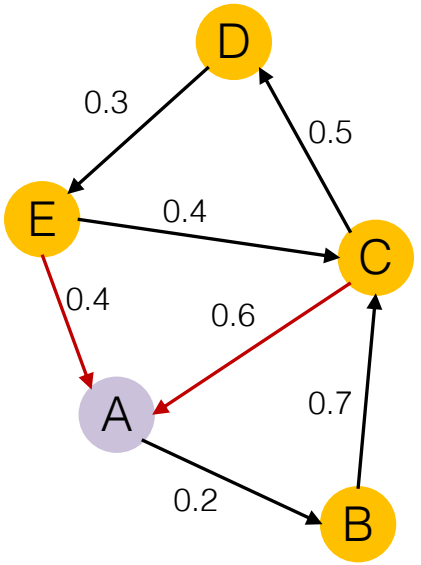
- An RR set is a random sample of G
- Generation of RR sets by computing reachable nodes under the IC model

RR set = {A}

Start from a random node

↓

Sample its incoming edges



[Borgs et al., SODA '14], [Aslay et al., Tutorial at WSDM '18]

Reverse Reachable Sets (RR Sets)

- An RR set is a random sample of G
- Generation of RR sets by computing reachable nodes under the IC model

RR set = {A}

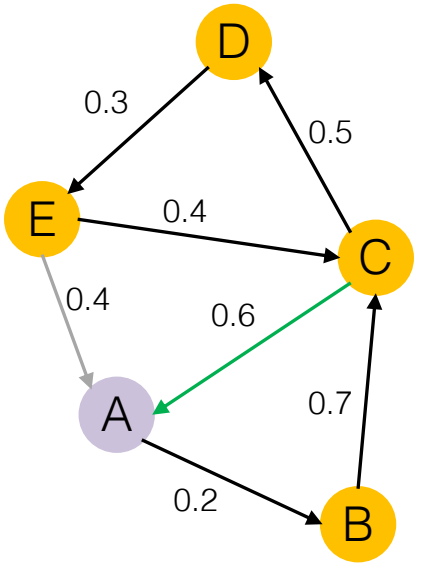
Start from a random node

↓

Sample its incoming edges

↓

Add the sampled neighbors

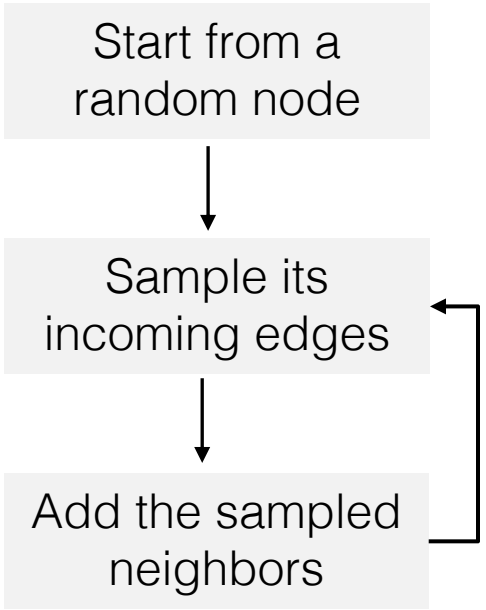


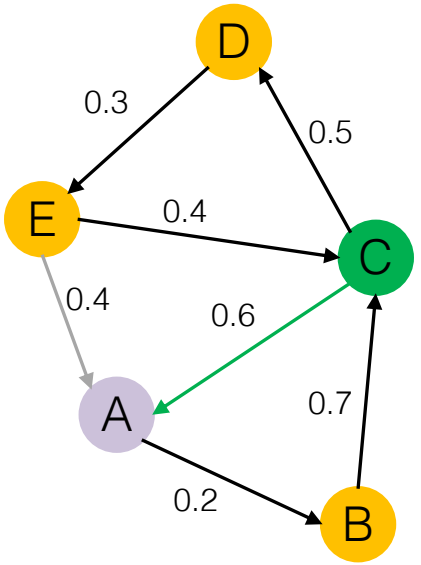
[Borgs et al., SODA '14], [Aslay et al., Tutorial at WSDM '18]

Reverse Reachable Sets (RR Sets)

- An RR set is a random sample of G
- Generation of RR sets by computing reachable nodes under the IC model

RR set = {A, C}



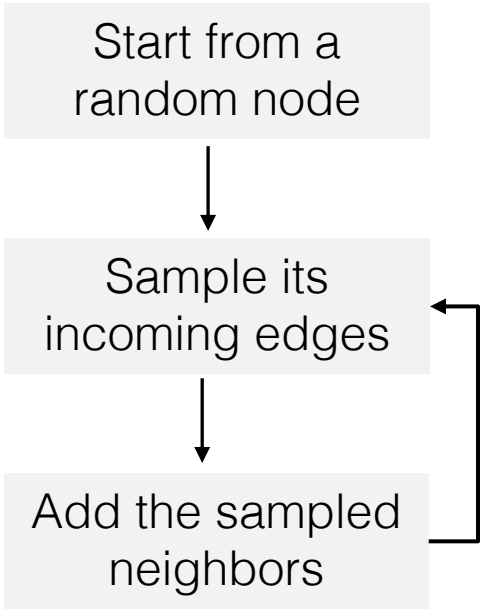


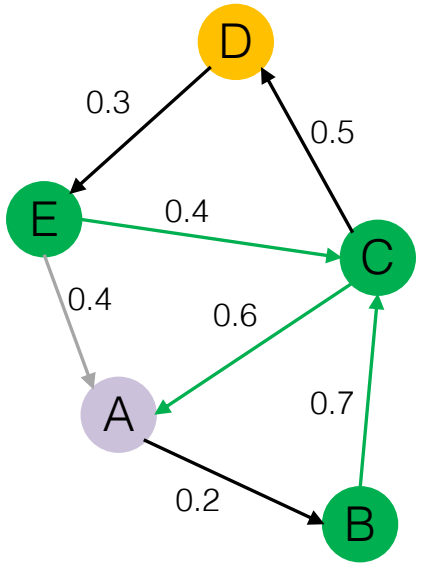
[Borgs et al., SODA '14], [Aslay et al., Tutorial at WSDM '18]

Reverse Reachable Sets (RR Sets)

- An RR set is a random sample of G
- Generation of RR sets by computing reachable nodes under the IC model

RR set = {A, C, B, E}





The RR set is a sample set of nodes that can influence node A

Influence Estimation with RR Sets (1/2)

- Suppose that we randomly generate a lot of RR sets

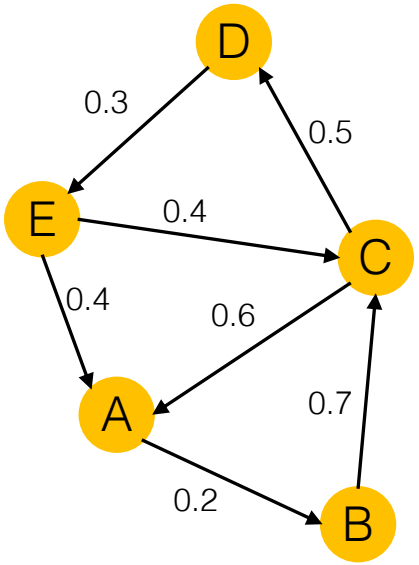
R1 = {A, C, B}

R2 = {B, A, E}

R3 = {C}

R4 = {D, C}

R5 = {E}



- Node C appears very frequently
  - C has a large influence

[Borgs et al., SODA '14], [Aslay et al., Tutorial at WSDM '18]

Influence Estimation with RR Sets (2/2)

$$\Pr(\text{node } v \text{ appears in a random RR set}) = \frac{1}{n} \cdot (v\text{'s expected influence})$$

- Example
  - Node C appears in $\frac{3}{5}$ RR sets
  - C's expected influence is roughly $\frac{3}{5}$ n

R1 = {A, C, B}

R2 = {B, A, E}

R3 = {C}

R4 = {D, C}

R5 = {E}

[Borgs et al., SODA '14], [Aslay et al., Tutorial at WSDM '18]

Influence Estimation with RR Sets (2/2)

$$\Pr(\text{node set } S \text{ overlaps a random RR set}) = \frac{1}{n} \cdot (S\text{'s expected influence})$$

- Example
  - {A, E} overlaps $\frac{3}{5}$ RR sets
  - {A, E}'s expected influence is roughly $\frac{3}{5}$ n

R1 = {A, C, B}

R2 = {B, A, E}

R3 = {C}

R4 = {D, C}

R5 = {E}

Computational benefit of RR sets

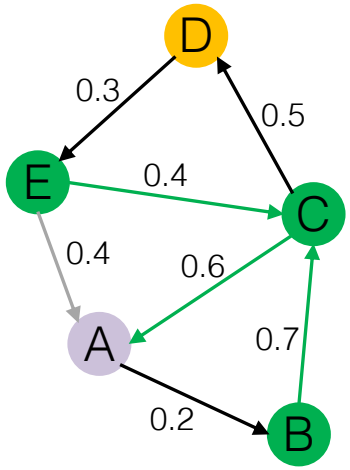
- The simulations to estimate influence are not repeated for each candidate seed
- Use the same RR sets to estimate the influence of all nodes

[Borgs et al., SODA '14], [Aslay et al., Tutorial at WSDM '18]

Reverse Influence Sampling – The Algorithm

1. Generate a number of RR sets
   - A seed set's influence spread is analogous to the number of RR sets it covers
   - Prob of a node being influenced by a seed = Prob of the seed existing in its RR set
2. Apply the Greedy algorithm to find a k-set that overlaps the most number of RR-sets
   - Maximum cover problem

- How many RR sets to sample?
  - Count the total cost of RR set construction
  - Stop when the cost > a threshold
- Example graph: Cost = 7
  - Cost = 1 for adding A
  - Cost = 2 for adding C
  - Cost = 2 for adding E and B
  - Cost = 2 for cheching the last two edges (D, E), (A, B)



[Borgs et al., SODA '14], [Aslay et al., Tutorial at WSDM '18]

Summary of Reverse Influence Sampling

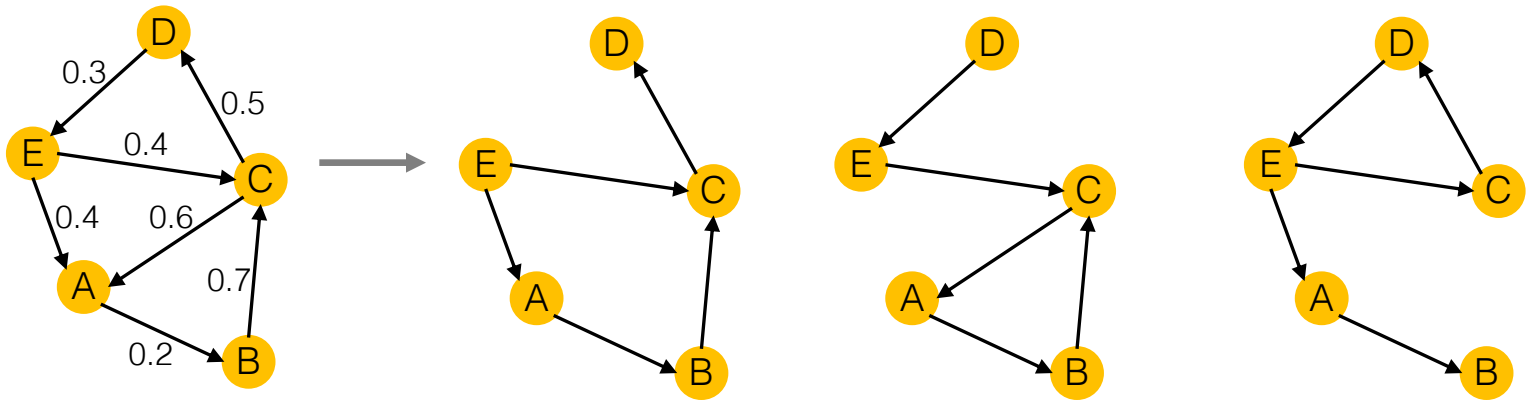
- Advantage of the [Borgs et al., SODA '14] algorithm
  - The first near-linear time algorithm
  - $\left(1 - \frac{1}{e} - \epsilon\right)$ - approximation
  - Time complexity: $\mathcal{O}\left((m+n)k\epsilon^{-2}\log n\right)$
- Drawbacks
  - Cost-driven selection of the # of RR sets

- Other ideas based on RR sets
  - TIM
    - Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency, Tang et al., SIGMOD '14
  - IMM
    - Influence Maximization in Near-Linear Time: A Martingale Approach, Tang at al., SIGMOD '15

Scalable Algorithms – Sketch-based IM (SKIM)

Sketch-based Influence Maximization

- Overview of the idea
  1. Take different realizations, producing a number of possible words



  2. Idenify set S of k nodes with the largest influence in these instances
     - σ(S): the average number of nodes reached by S in the instances

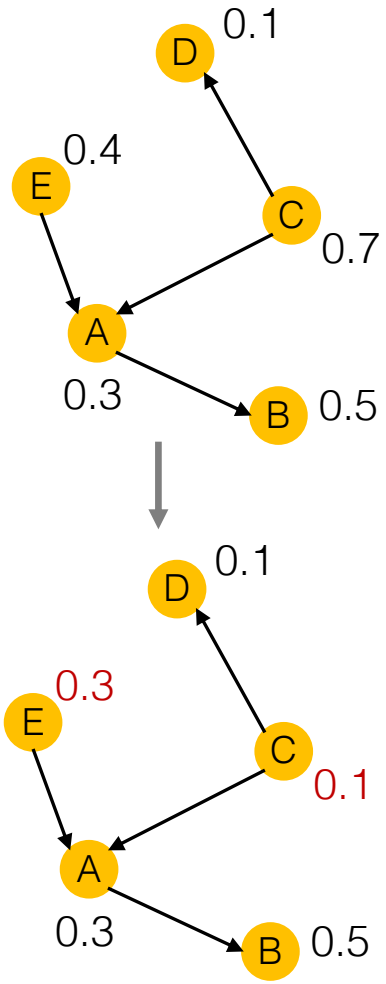
Evaluating influence in a possible world takes O(m) time
Use sketches (signatures) to reduce the estimation time

[Cohen et al., CIKM '14], [Aslay et al., Tutorial at WSDM '18]

Reachability Sketches (1/2)

Overview of the idea:

- Take a possible world G_i and assign a random number in [0, 1] to each node
- Compute the rank of each node v
  - The minimum number among the nodes that v can reach



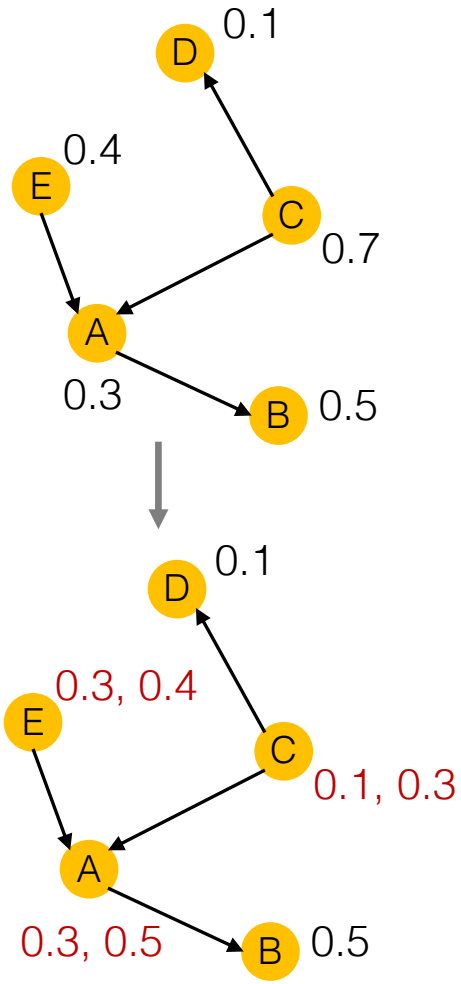
- If v can reach many nodes, then its rank is likely to be small
- Use the rank of v to estimate the influence of v in G_i

[Cohen et al., CIKM '14], [Aslay et al., Tutorial at WSDM '18]

Reachability Sketches (2/2)

Problem: influence estimation based on one rank would be inaccurate

- Keep multiple ranks to have a better estimate of the influence
  - E.g., the smallest c values among the nodes that v can reach
- Keep the smallest c values among the nodes that v can reach in all possible worlds extracted



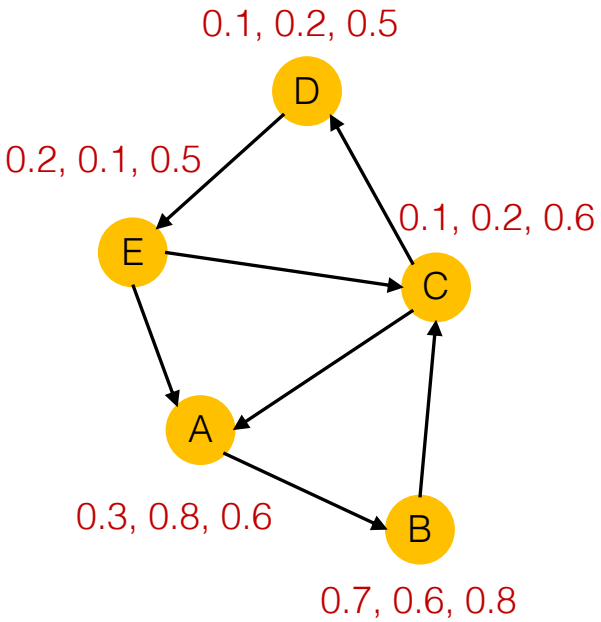
Idea

- Use the Greedy algorithm
- Substitute the influence spread estimation using the ranks of the candidate seed set

[Cohen et al., CIKM '14], [Aslay et al., Tutorial at WSDM '18]

Sketch-based IM (SKIM)

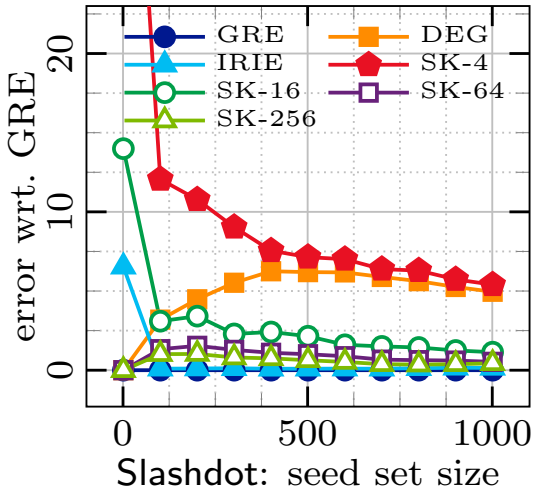
1. Generate a number of possible words
2. Construct reachability sketches for all nodes
   - Each node will obtain c ranks
3. Use the Greedy algorithm
   - To evaluate the influence of a seed set S, check the ranks to derive the estimation
   - Smaller rank ≈ higher influence, for node v

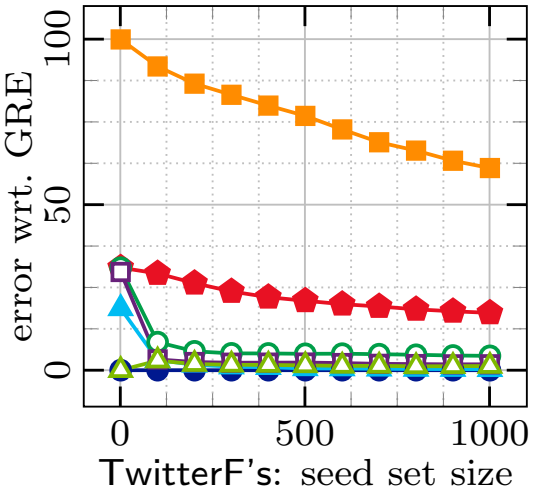


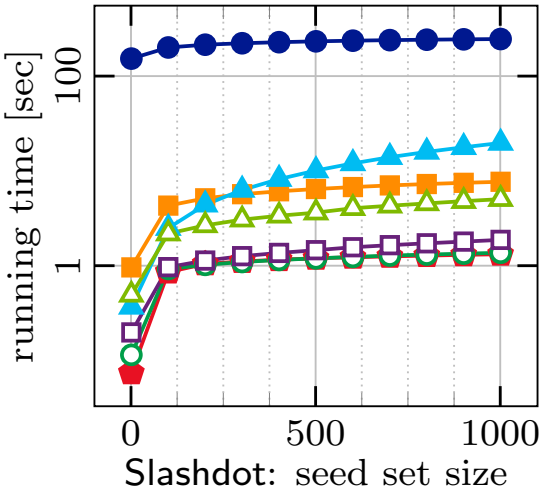
Theoretical guarantees

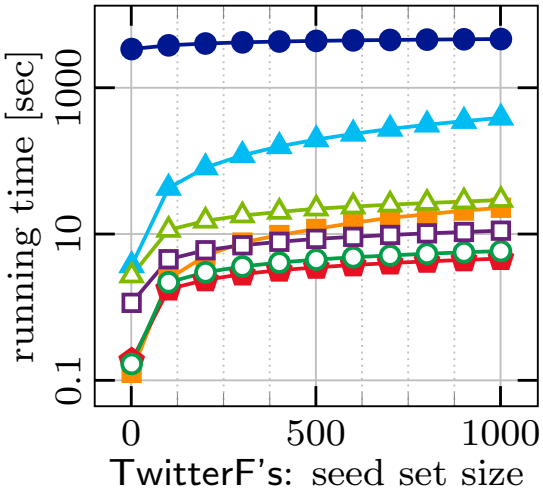
- Expected running time: near-linear to the total number of possible worlds
- When c is large enough, $\left(1 - \frac{1}{e} - \epsilon\right)$ - approximation w.r.t. the number of possible worlds

Experimental Evaluation







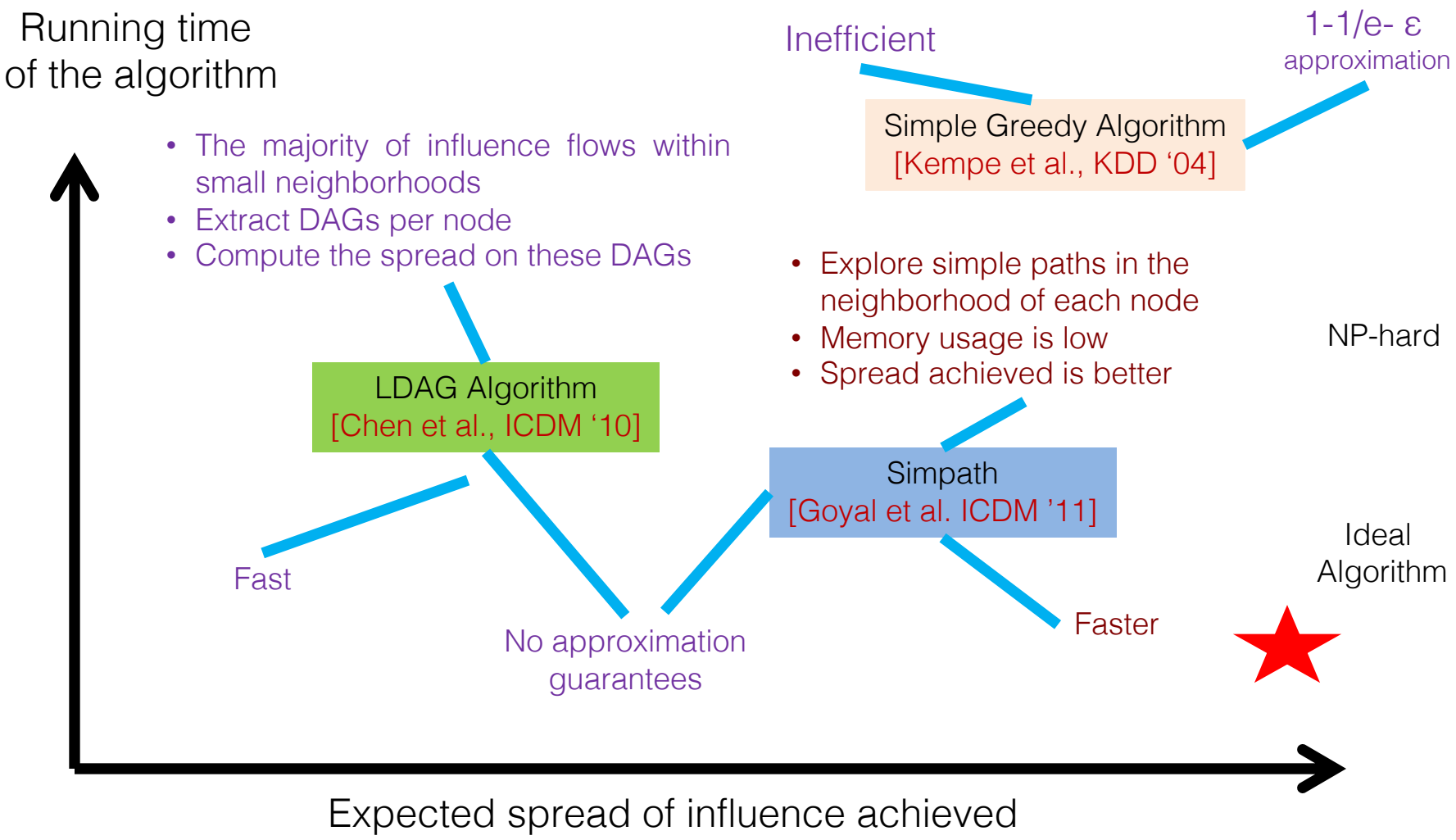


- GRE: Greedy
- IRIE: scalable heuristic
- DEG: degree-based
- SK: Sketch-based IM

[Cohen et al., CIKM '14]

Scalable Heuristics – MIA, LDAG, SimPath

Scalable Heuristics for IM



[Goyal et al. ICDM '11]

Part II Summary

- The Greedy algorithm
  - Approximation guarantees
  - Running time is high
- Main research focus on scalabilility
  - With or without guarantees
- Mainly rely on the structure of the graph
  - Real diffusion cascades (e.g., retweets) are not taken into account
  - Next part of the tutorial

Outline of the Tutorial

Part I. Introduction

Part II. Traditional influence maximization

Part III. Influence and diffusion learning

Part IV. Learning influence maximization

Part V. Online influence maximization

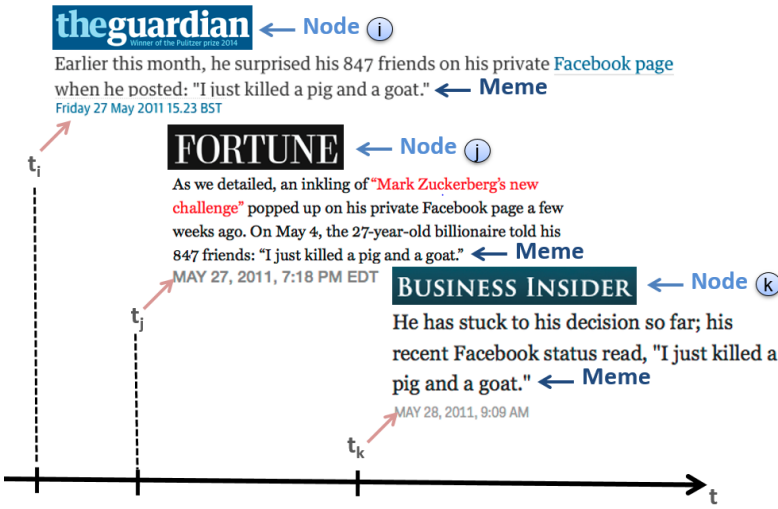
Part VI. Summary and open challenges

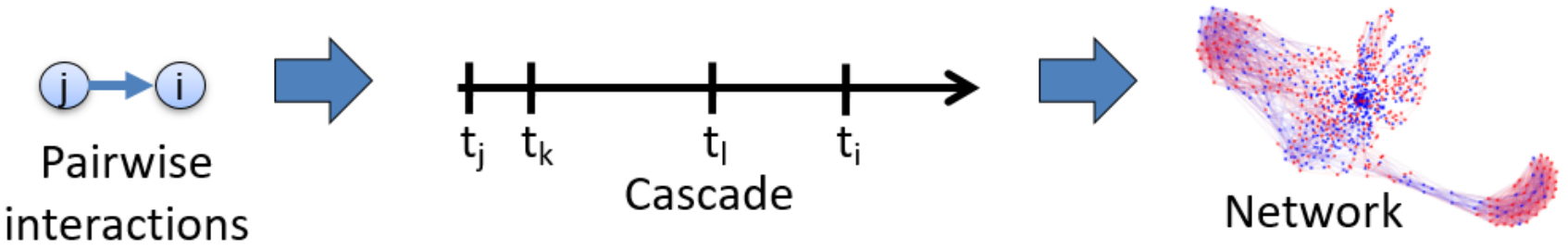
Part III. Influence and diffusion learning

- Learning influence
- Predicting diffusion
  - Recurrent Neural Networks
  - Point-processes
- Learning influence for IM

Learn from Diffusion Cascades

- Network inference:
  - Social networks have an underlying network, but the web of media doesn't
  - Use the cascades to infer the actual edges between nodes in the web.



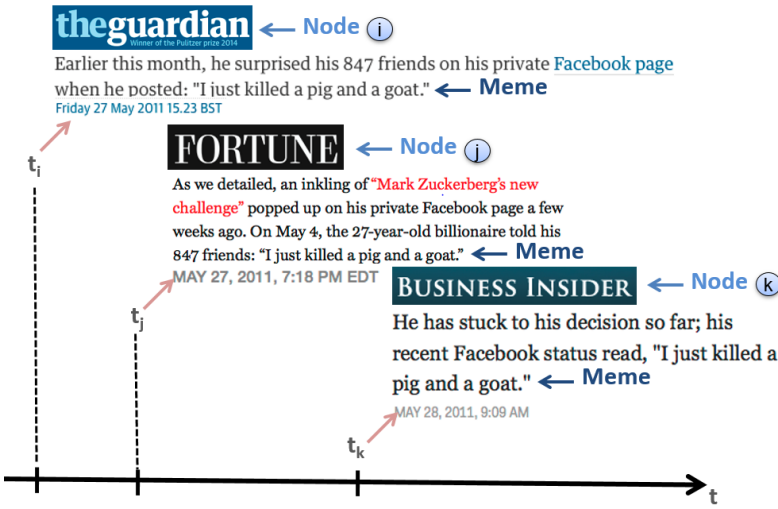


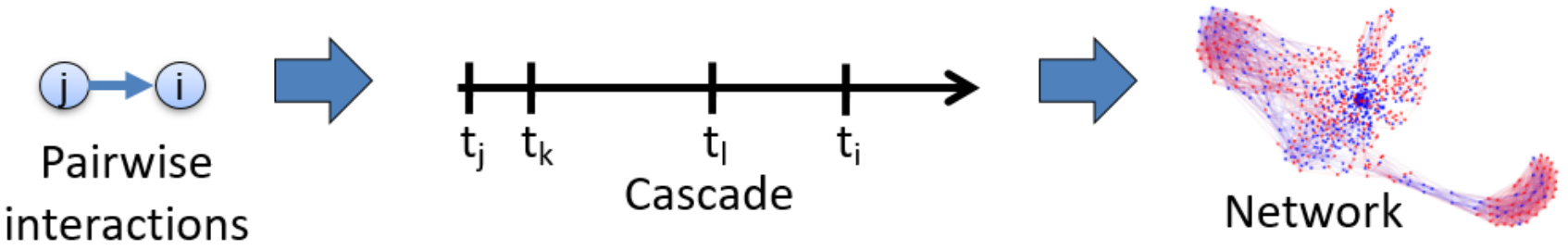
- Over a social network, learn how users influence each other to:
  - Predict the diffusion accurately
  - Use it for influence maximization

[Gomez Rodriguez et al., Tutorial at KDD '15]

Learn from Diffusion Cascades

- Network inference:
  - Social networks have an underlying network, but the web of media doesn't
  - Use the cascades to infer the actual edges between nodes in the web.



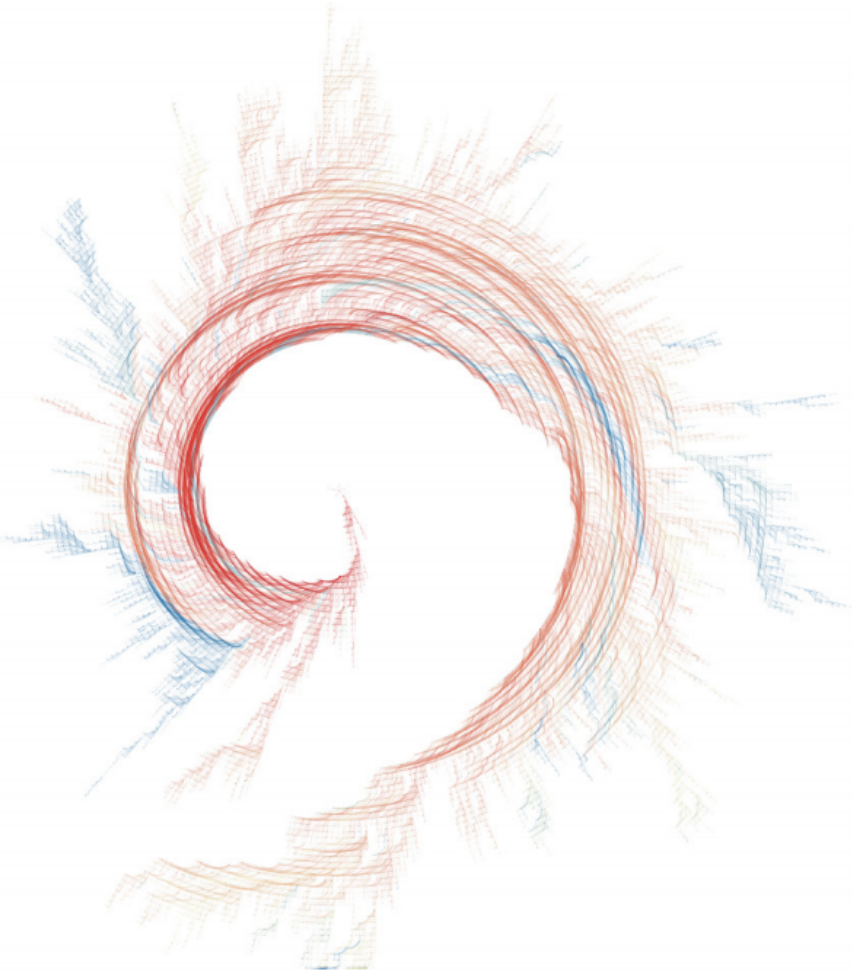


- Over a social network, learn how users influence each other to:
  - Predict the diffusion accurately
  - Use it for influence maximization

[Gomez Rodriguez et al., Tutorial at KDD '15]

Diffusion Cascades Example

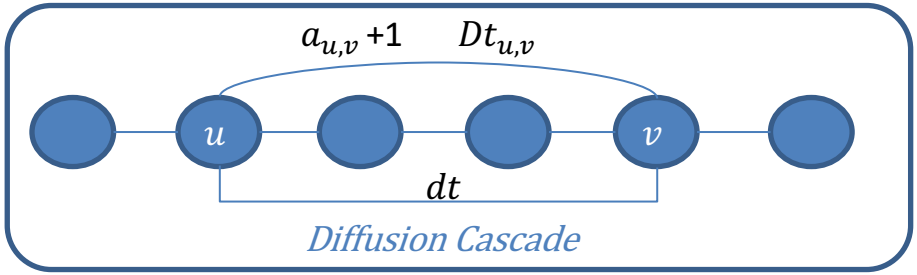
- Individuals posted music from an artist whose name matched the letter they were assigned by a friend
- Each node starts a new line when their friends adopt the cascade
- Each friend increases radius by starting their own cascade
- Edges are colored from red (early) to blue (late)

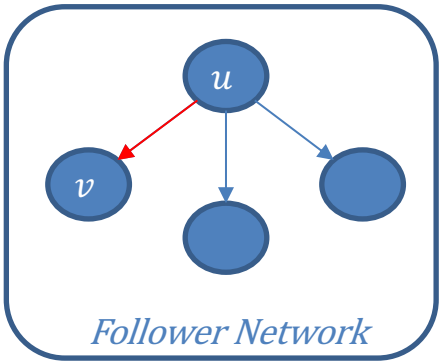


The diffusion tree of a cascade.
[Cheng et al. ICWSM '18]

Estimate Influence Weights

- Using a log of activities (diffusion cascades) and the structure of the network
- When v retweets in a cascade at time t
  - Increase $a_{u,v}$, where u are all the previous nodes that v follows at t
  - Augment $Dt_{u,v}$ with the time passed between u's and v's occurrence





$$p_{u,v}^{t} = \frac{a_{u,v}}{a_u} e^{-\left(\frac{t-t_u}{Dt_{u,v}}\right)}$$

[Goyal et al. WSDM '10]

Estimate Influence Weights

- The exponential decay of influence is an empirically observed phenomenon
- Under IC, the probability of $\mathbf{v}$ getting influenced in a diffusion cascade at time t is the opposite of the probability of surviving from all of its neighbors:

$$p_v^t = 1 - \prod_{u \in N(v)} (1 - p_{u,v}^t)$$

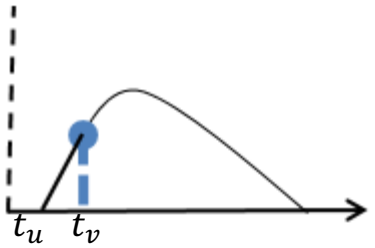
- For prediction, use the above to define if a node will be influenced based on a diffusion C and measure the
- For Influence Maximization, use probabilities as weights and run SimPath

[Goyal et al. WSDM '10]

Learning Continuous Time Influence

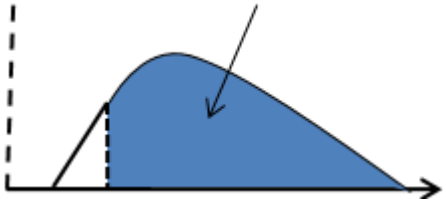
- Influence probability is a function of the transmission delay

$$P(t_v|t_u; a_{u,v}) := a_{u,v}\, e^{-a_{u,v}(t_v - t_u)}$$



- Infer the transmission rate $a_{u,v}$, which may reveal an edge
- The survival function is the probability that v is not activated by node u.

$$S(t_v|t_u; a_{u,v}) := 1 - P(t_v|t_u; a_{u,v})$$



[Gomez Rodriguez and Song, Tutorial at KDD '15] [Gomez Rodriguez et al., ICML '11]

Learning Continuous Time Influence

- Likelihood of activations **and** non-activations of a cascade C by time T

$$P(C|A) = \prod_{\substack{v \in C, \\ t_v \leq T}} \left[\prod_{t_m > T} S(T|t_v; a_{v,m}) * \left(\prod_{t_u < t_v} P(t_v|t_u ; a_{u,v}) \prod_{\substack{k \neq u, \\ t_k < t_v}} S(t_v|t_k ; a_{k,v}) \right) \right]$$

Not having infected m by time T

Getting infected by u

Staying uninfected by other nodes

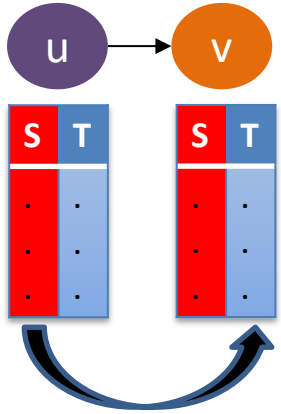
- Convex log-likelihood
- Inferred network needs to be in a very small scale (thousands of nodes)
- The inference accuracy increases as the number of cascades increases
- Run PMIA, an IC-based heuristic for IM

[Gomez Rodriguez and Song, Tutorial at KDD '15] [Gomez Rodriguez et al., ICML '11]

Embedded Independent Cascade

- Need to learn less parameters to battle overfitting
- Define the influence probability based on a pair of *influence & susceptibility*

$$p_{u,v} = f(S_u, T_v) = \frac{1}{1+e^{\left(S_u^0 + T_v^0 + \sum_{i=1}^{m-1}\left(S_u^i - T_v^i\right)^2\right)}}$$



- Instead of |E| parameters, learn |N|$\bar{d}$ embeddings in R^f

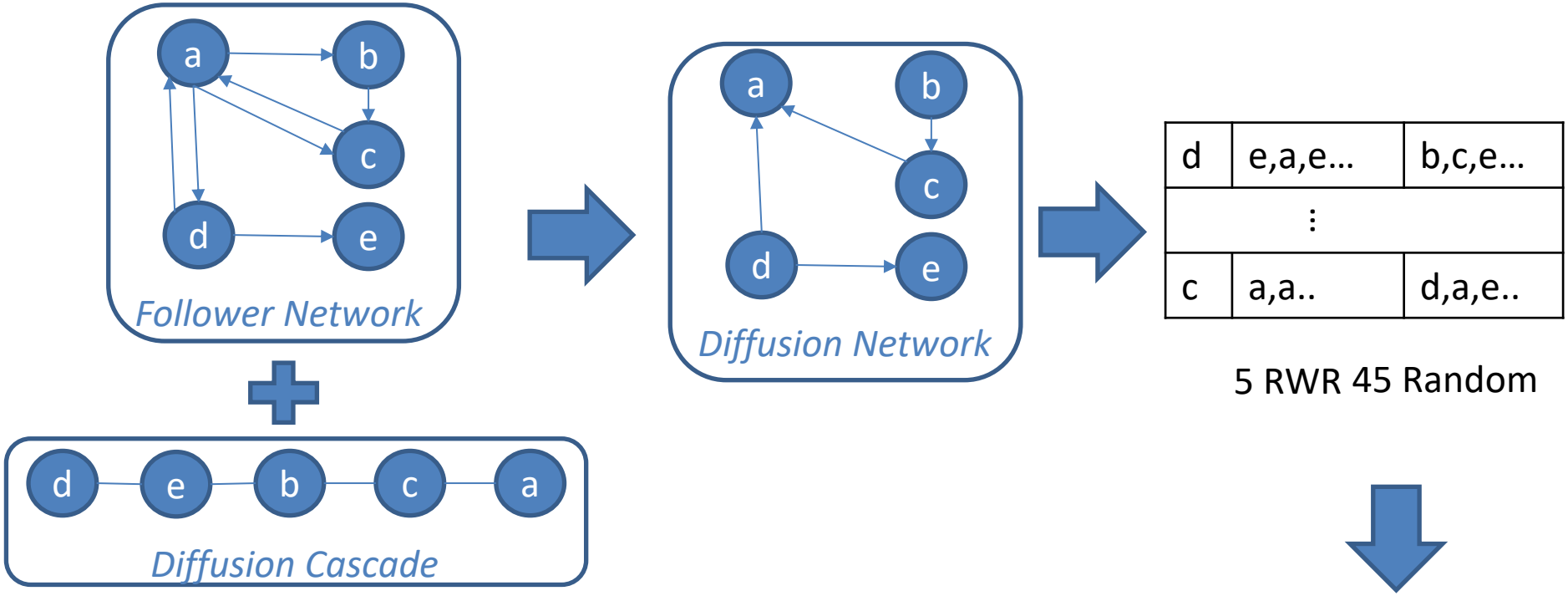
$$L(P, D) = \sum_{d \in D}\left(\sum_{v \in d} \log\left(P_v^d\right) + \sum_{v' \in \bar{d}} \log\left(1 - P_{v'}^d\right)\right)$$

- Optimize using EM
- Evaluate on predicting the diffusion of real cascade.

[Bourigault et al., WSDM '16]

Influence 2 vector

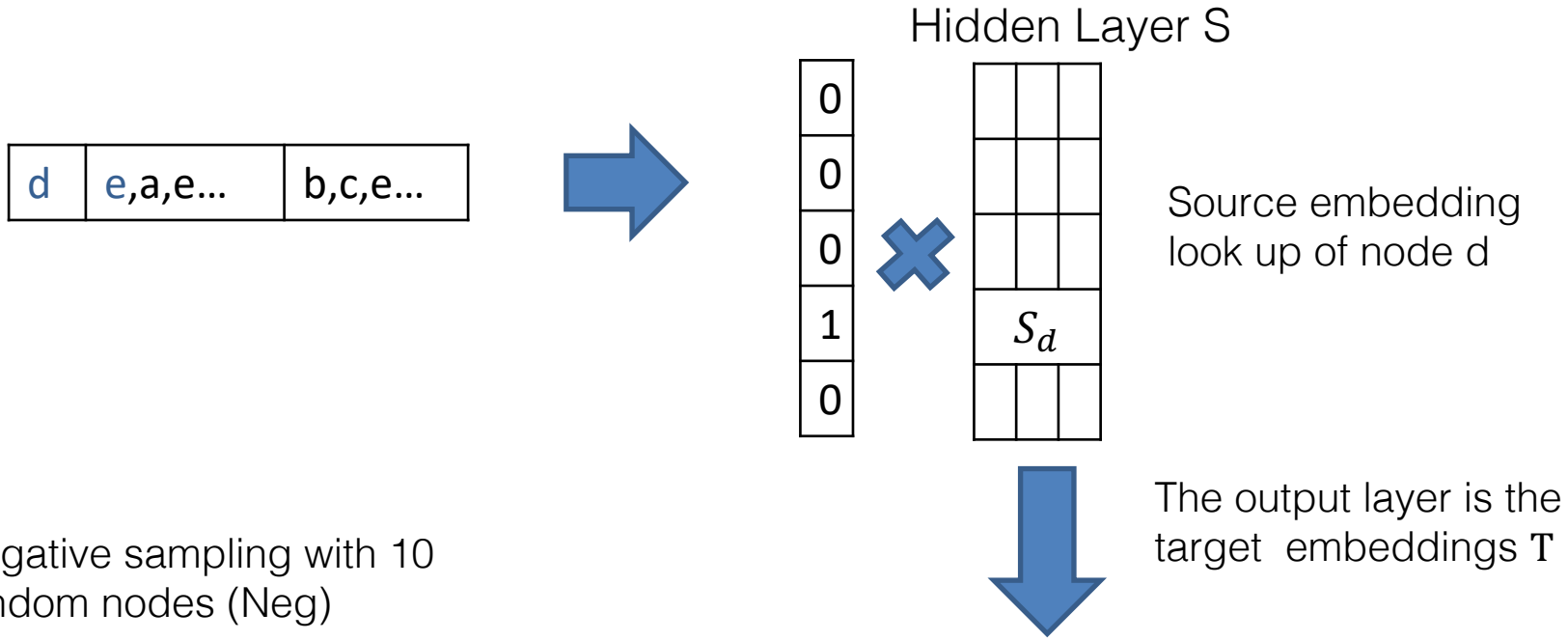
- Derive embedding's of influence using diffusion networks
- Use both, the network and the diffusion cascades
- For each node in a diffusion network, derive a context (similar to word2vec) based on RWR and random sampling



[Feng et al., ICDE '18]

Influence 2 vector

- Use the node-context pairs to train a shallow NN
- Predict the course of the cascade using these representations



[Feng et al., ICDE '18]

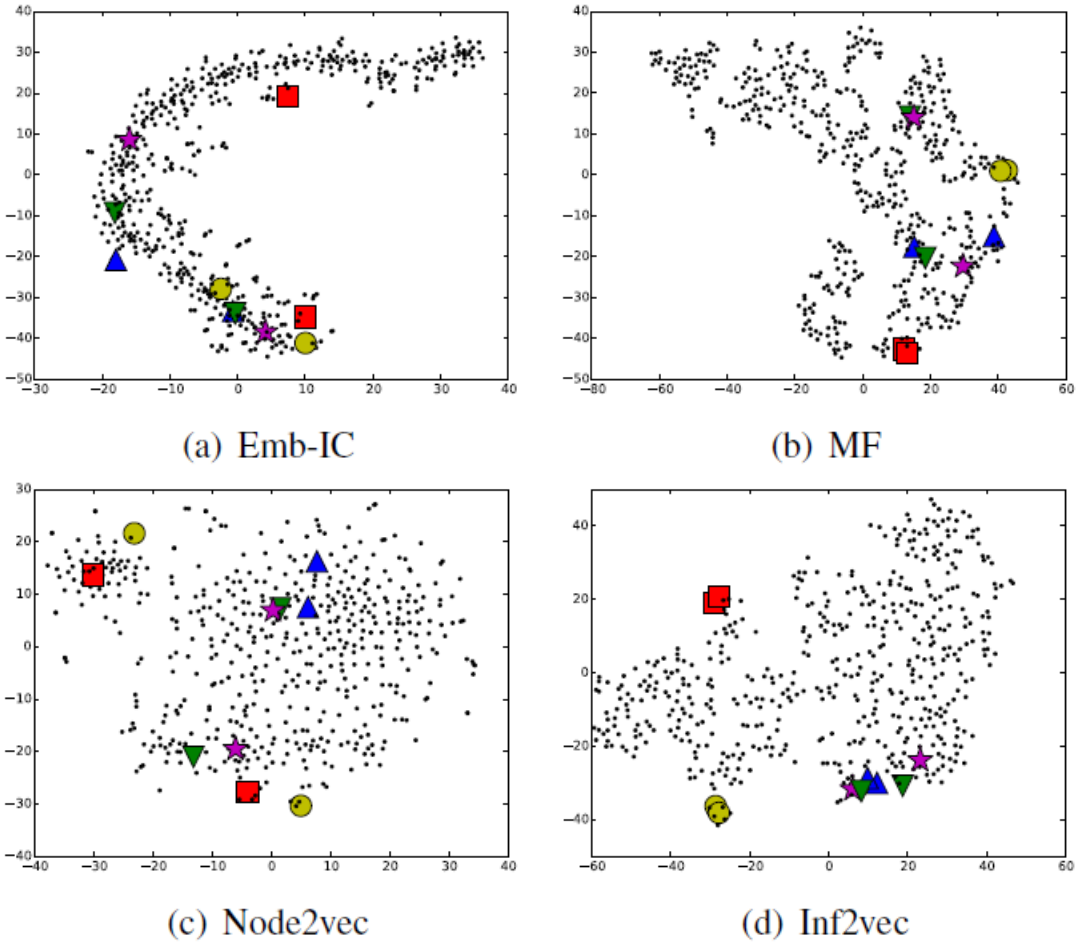
Influence 2 vector

Dataset	Method	AUC	MAP	$P@10$	$P@50$	$P@100$
Digg	DE	0.6183	0.0173	0.0121	0.0145	0.0132
	ST	0.6874	0.1064	0.6735	0.3841	0.3091
	EM	0.7095	0.1241	0.6261	0.4364	0.3572
	Emb-IC	0.6649	0.1047	0.5458	0.3912	0.3286
	MF	0.8677	0.1347	0.5087	0.4059	0.3389
	Node2vec	0.6606	0.0219	0.0810	0.0718	0.0556
	Inf2vec	**0.8904**	**0.1793**	**0.7386**	**0.4750**	**0.3932**
	(stdev σ)	(0.0002)	(0.0015)	(0.0214)	(0.0107)	(0.0049)
Flickr	DE	0.6177	0.0026	0.0025	0.0048	0.0041
	ST	0.6840	0.0242	0.1215	0.0871	0.0685
	EM	0.7479	0.0260	0.1115	0.0773	0.0636
	Emb-IC	0.7582	0.0199	0.0955	0.0754	0.0622
	MF	0.8699	0.0280	0.1044	0.0832	0.0703
	Node2vec	0.6233	0.0023	0.0010	0.0053	0.0048
	Inf2vec	**0.8778**	**0.0301**	**0.1254**	**0.0943**	**0.0759**
	(stdev σ)	(0.0011)	(0.0004)	(0.0054)	(0.0009)	(0.0007)

Results in predicting the diffusion

[Feng et al., ICDE '18]

Influence 2 vector



(a) Emb-IC (b) MF

(c) Node2vec (d) Inf2vec

Learned representations of pairs of nodes that appear frequently together in the cascades of the Digg dataset

[Feng et al., ICDE '18]

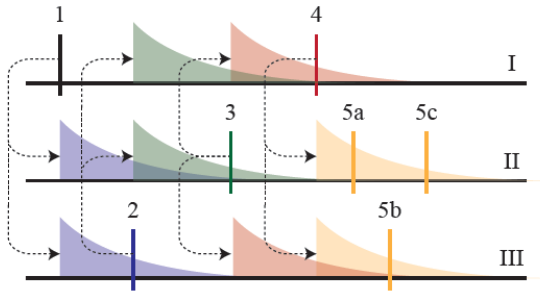
Neural Networks for Diffusions

- Extensive work on predicting diffusion with temporal neural networks:
  - TopoLSTM [Jia et al. ICDM 2017]
  - Cyan RNN [Wang et al. CIKM 2018]
  - DeepDiffuse [Islam et al. ICDM 2018]
  - FOREST [Yang et al. IJCAI 2019]

- The representation from these models can not be adapted for other tasks, such as influence maximization, in a straightforward manner

Hawkes Process for Diffusions

- Model the cascades as a set of interacting Poisson processes



- $\{S_n\}$ is a set of marked events
- Probability of an event happening at node k depends on its background rate λ_k and its interaction with other nodes h
- $p((s_n, c_n, z_n)|\lambda, h) =$
$\prod_{k=1} p(c_n = k, z_n = 0|\lambda_k) * \prod_{n\prime=1} \prod_{v=1} p(c_{n'} = v, z_n = n'|h_{v,k}(\Delta T_{n',n}))$

s_n caused by background rate of node k

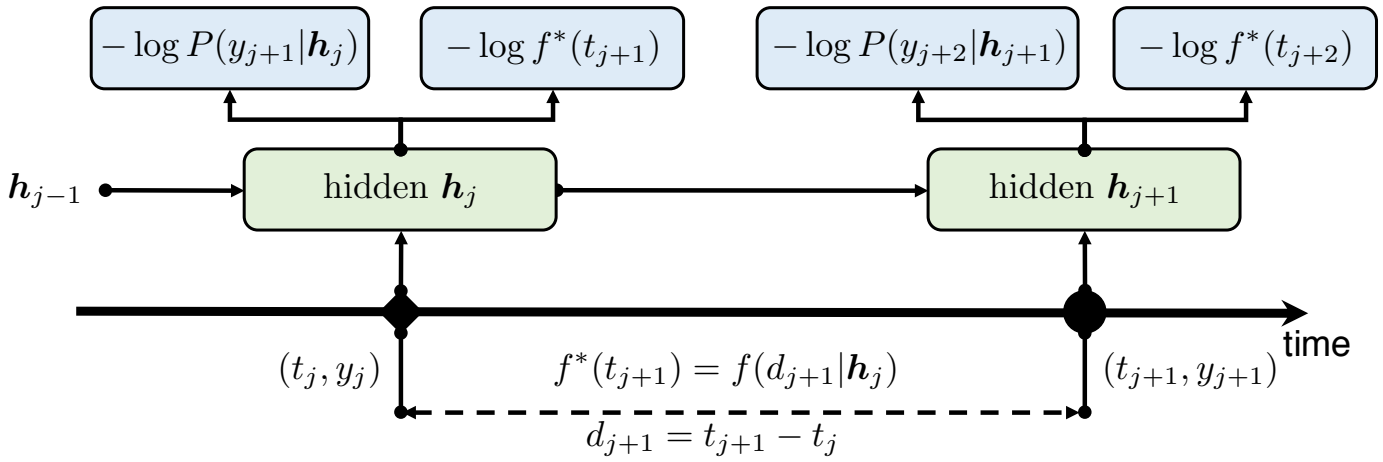
s_n was caused by spike at time n', which belonged to node v

- Learn $h_{v,k}(\Delta t) = W_{v,k} g_{\theta_{v,,k}}(\Delta t)$ using Stochastic Variational Inference

[Linderman et al., ICML '14]

Recurrent Marked Temporal Point Process

- Use RNN to predict which node will get influenced next and when, during a diffusion cascade
- The next node y_{j+1} and its time t_{j+1}, depends non-linearly on the history h_j (previously infected nodes and their times)
- Embed the history into a latent vector (hidden state of an RNN) and use it for prediction

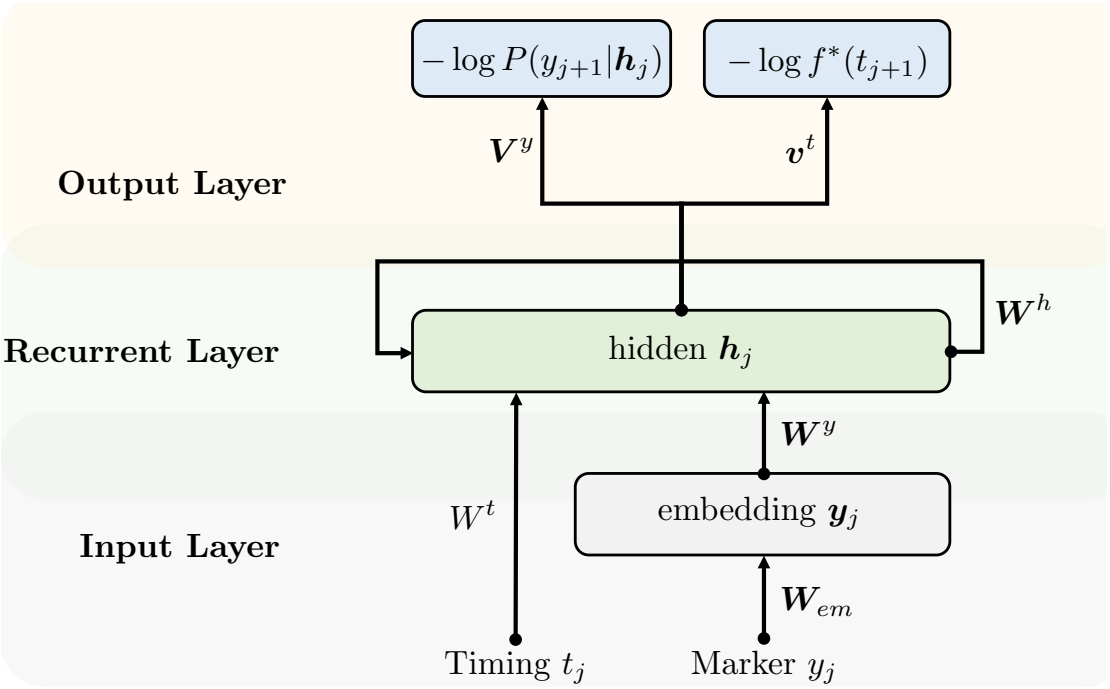


[Du et al., KDD '16]

Recurrent Marked Temporal Point Process

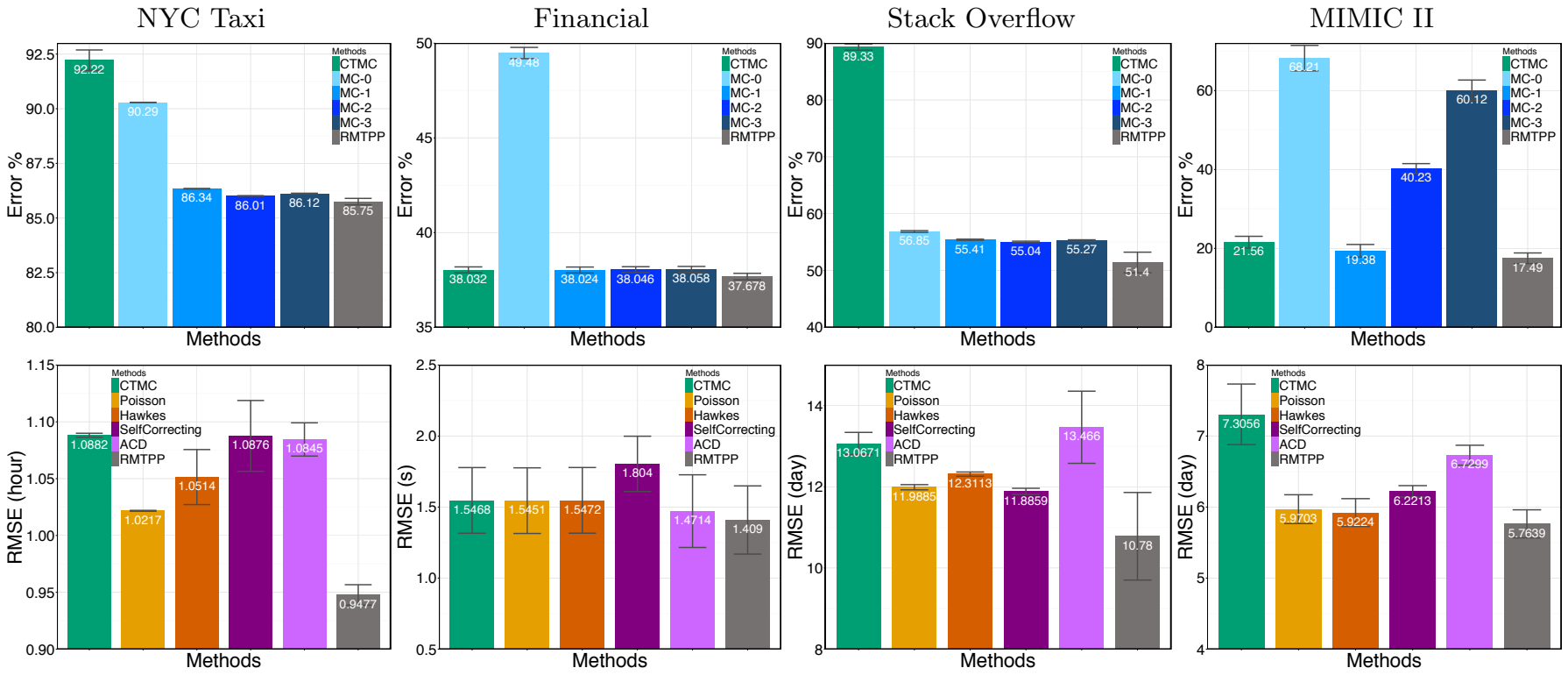
- To predict next node use a standard softmax output
- To predict next time use a point process with rate

$$\lambda^*(t) = \exp\left(u^t h_j + w^t\left(t - t_j\right) + b^t\right)$$



[Du et al., KDD '16]

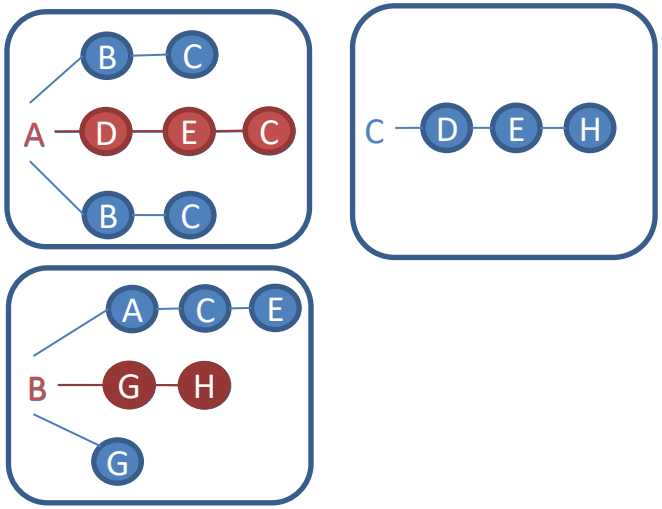
Recurrent Marked Temporal Point Process

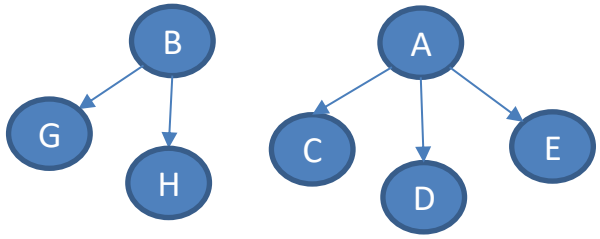


[Du et al., KDD '16]

DiffuGreedy

- Use the diffusion cascades to directly maximize influence
- Assumption: the candidate seeds have started a diffusion in the past
- Use greedy, but compute a seed's influence spread by its diffusion cascades:
  - Choose the diffusion cascade that provides the best marginal gain using DNI





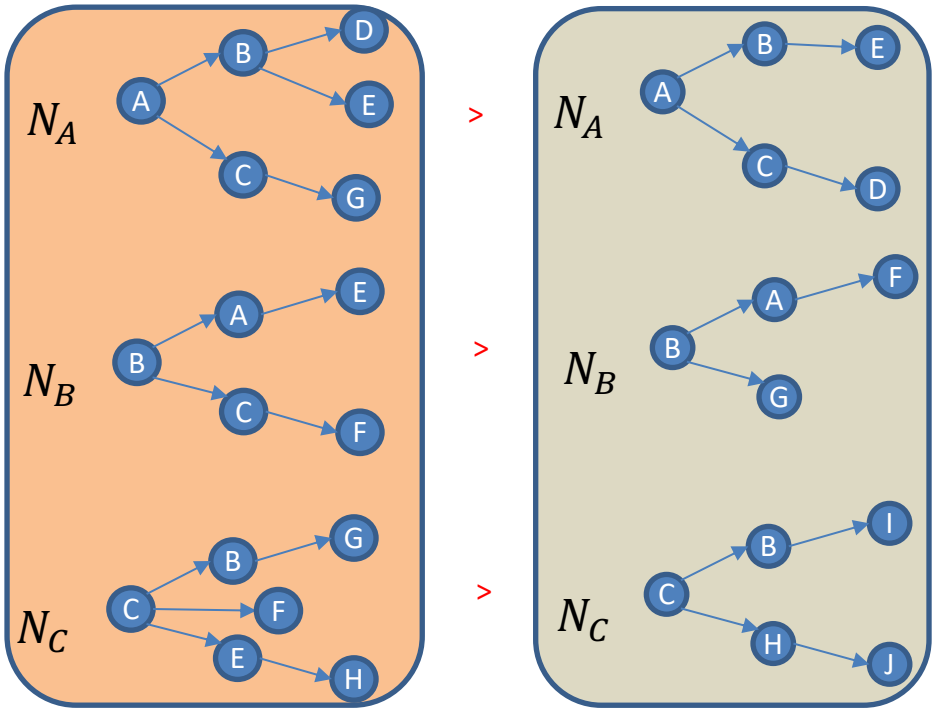
[Panagopoulos et al., Complex Networks '18]

DiffuGreedy

- Train-test split of the cascades based on time of the initial post

- Simpler (but wrong) evaluations:
  - Sum of seeds' average test cascade size
  - Sum of seed's follows, mentions or retweets

- Number of distinct nodes influenced



$\{A: H\} < \{A : \mathrm{J}\}$

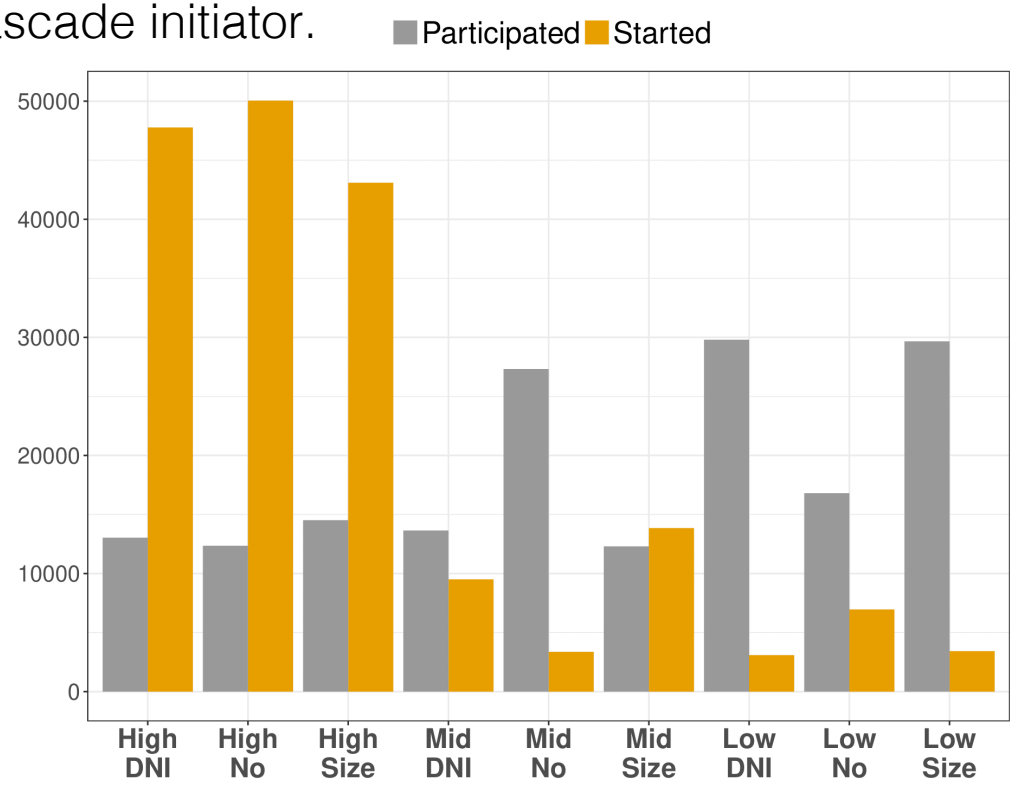
[Panagopoulos et al., Complex Networks '18]

Learning Influence for IM

- Influencers create or copy more?
  - Rank initiators in the test set based on success metrics.
  - Successful influencers are more prone to start than participate in train cascades
  - Derive only the context of the cascade initiator.

- Utilize their cascades and inf2vec to learn influence and susceptibility embeddings between them and the rest of the network

- Use these embeddings to perform influence maximization



[Panagopoulos et al., ICWSM '20]

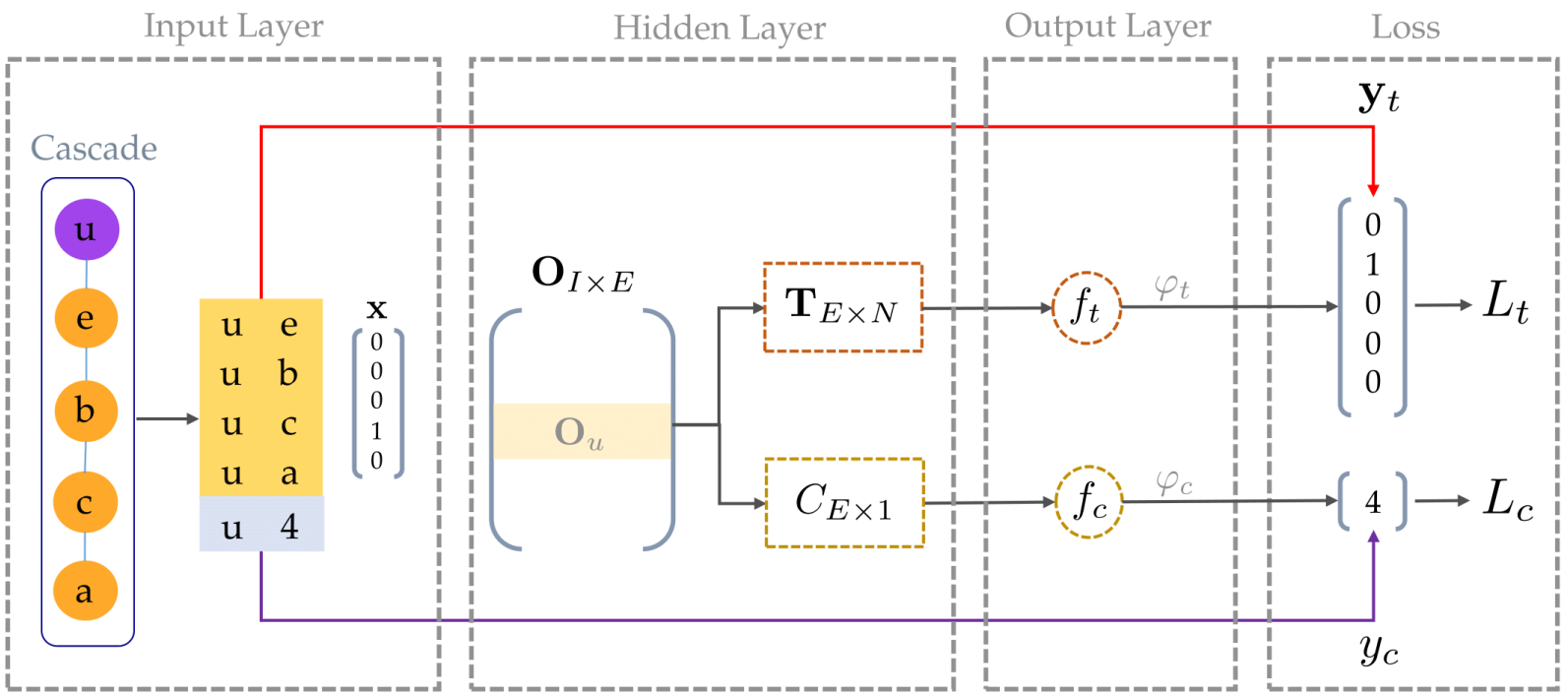
Learning INFluencer Vector (INFECTOR)

- Embed at the same vectors:
  - The probability of influencing a node
  - The initiator's aptitude to create lengthy cascades
- Hidden layer $\mathbf{S}$ is updated by both inputs, in an alternating manner
  - $\mathbf{S}$ and $\mathbf{T}$ form the influence likelihood between nodes
  - $|\mathbf{S}|$ captures the nodes' cascade size

	Classify influenced node	**Regress cascade size**
Hidden	$z_{t,u} = S_u T + b_t$	$z_{c,u} = S_u C + b_c$
Output	$\varphi_t(S_u) = \frac{e^{-(z_{t,u})}}{\sum_{u' \in G} e^{-(z_{t,u'})}}$	$\varphi_c(S_u) = 1/(1 + e^{-(z_{c,u})})$
Loss	$L_t = y_t \log(\varphi_t(S_u))$	$L_c = (y_c - \varphi_c(S_u))^2$

[Panagopoulos et al., TKDE '20]

INFECTOR

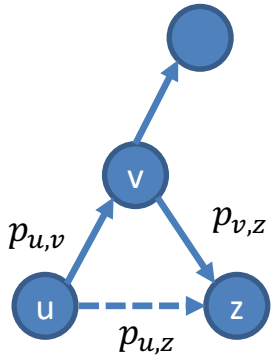


[Panagopoulos et al., TKDE '20]

Reformulation of the Problem

- Use diffusion probability matrix to compute influence spread:

$$D = \begin{bmatrix} \varphi_t(O_1\, T) \\ \vdots \\ \varphi_t(O_I T) \end{bmatrix}$$



- Difference between influence and diffusion probabilities
  - $\mathbf{D}$ does not require the existence of an edge in the network

- Advantage: Captures higher order correlations that IM techniques fail to
  - If $\mathbf{v}$ appears in the diffusions of $\mathbf{u}$ and $\mathbf{z}$ appears in the diffusions of $\mathbf{v}$ but not in $\mathbf{u}$'s

[Panagopoulos et al., TKDE '20]

Reformulation of the Problem

- Disadvantage: Too many edges (essentially a fully connected influence network)
- Define an expectation of a candidate's seed influence spread:

$$\Lambda_u = \left\lceil N \frac{||O_u||_2}{\sum_{u' \in \mathcal{I}} ||O_{u'}||_2} \right\rceil$$

- Use it to diminish the pool of candidate nodes
- For a seed s its influence spread is the total edge weight of the nodes it influences and is given by

$$\sigma'(s) = \sum_{j}^{\Lambda_s} \hat{D}_{s,j},$$

[Panagopoulos et al., TKDE '20]

Influence Maximization with INFECTOR

- Optimize $\sigma'(s)$ in a greedy manner.
- Since there are no higher order paths, remove the node added in each iteration

Seed	*N1*	*N2*	*N3*	*N4*	*N5*	Λ	σ'
S1	0.1	0.3	0.2	0.2	0.2	2	0.5
S2	0.4	0.2	0.2	0.1	0.2	2	0.6
S3	0.5	0.1	0.2	0.2	0	3	0.9

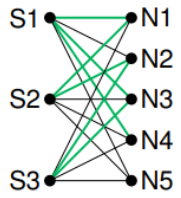


Figure: Step 1: S=[S3]

Seed	*N2*	*N5*	Λ	σ'
S1	0.3	0.2	2	0.5
S2	0.2	0.2	2	0.4

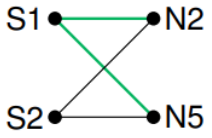
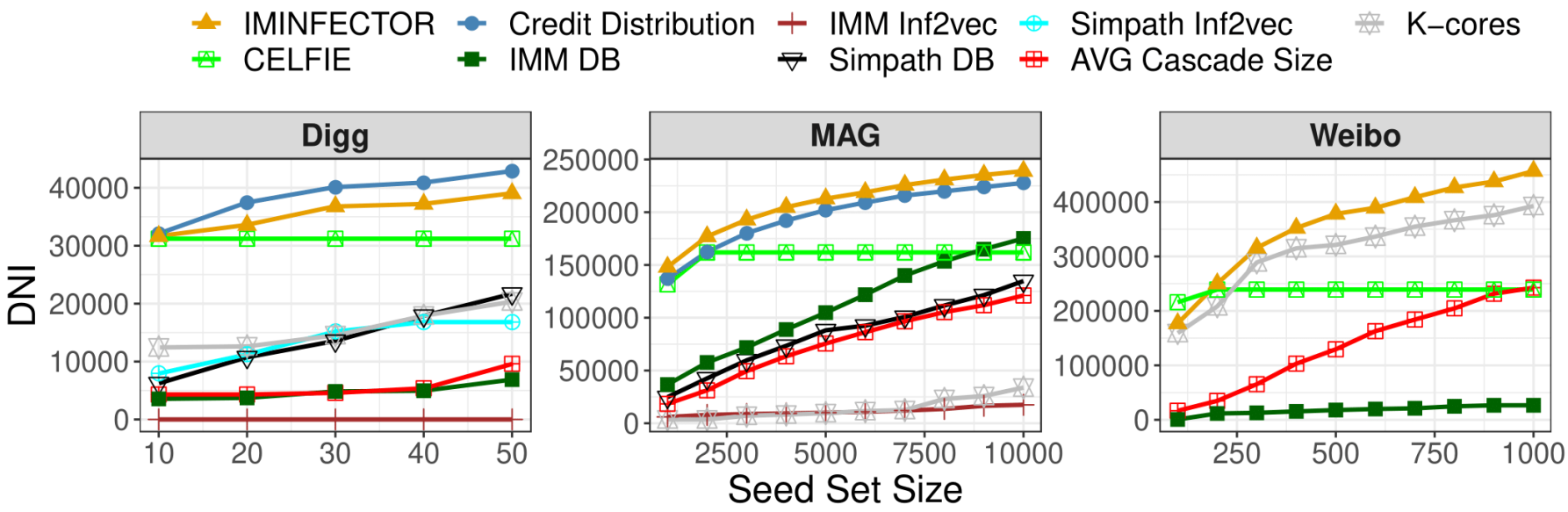


Figure: Step 2: S=[S3,S1]

- The spread is submodular & monotonic -> Use CELF for optimization

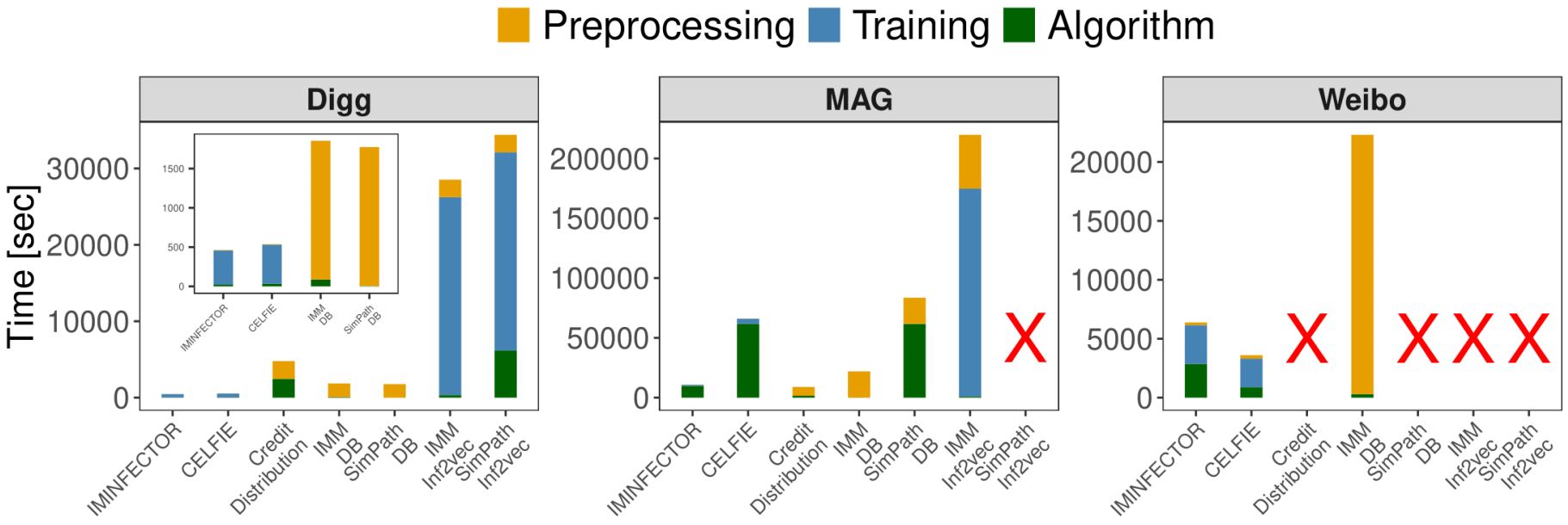
[Panagopoulos et al., TKDE '20]

IMINFECTOR – Experimental Results



[Panagopoulos et al., TKDE '20]

IMINFECTOR – Experimental Results



[Panagopoulos et al., TKDE '20]

Part III Summary

- Learning influence
  - Data mining and probabilistic perspective
- Diffusion prediction
  - Recurrent neural networks
  - Point Processes
  - Combination
- Influence maximization with learnt parameters
  - Improve efficiency and effectiveness using the cascades

Outline of the Tutorial

Part I. Introduction

Part II. Traditional influence maximization

Part III. Influence and diffusion learning

Part IV. Learning influence maximization

Part V. Online influence maximization

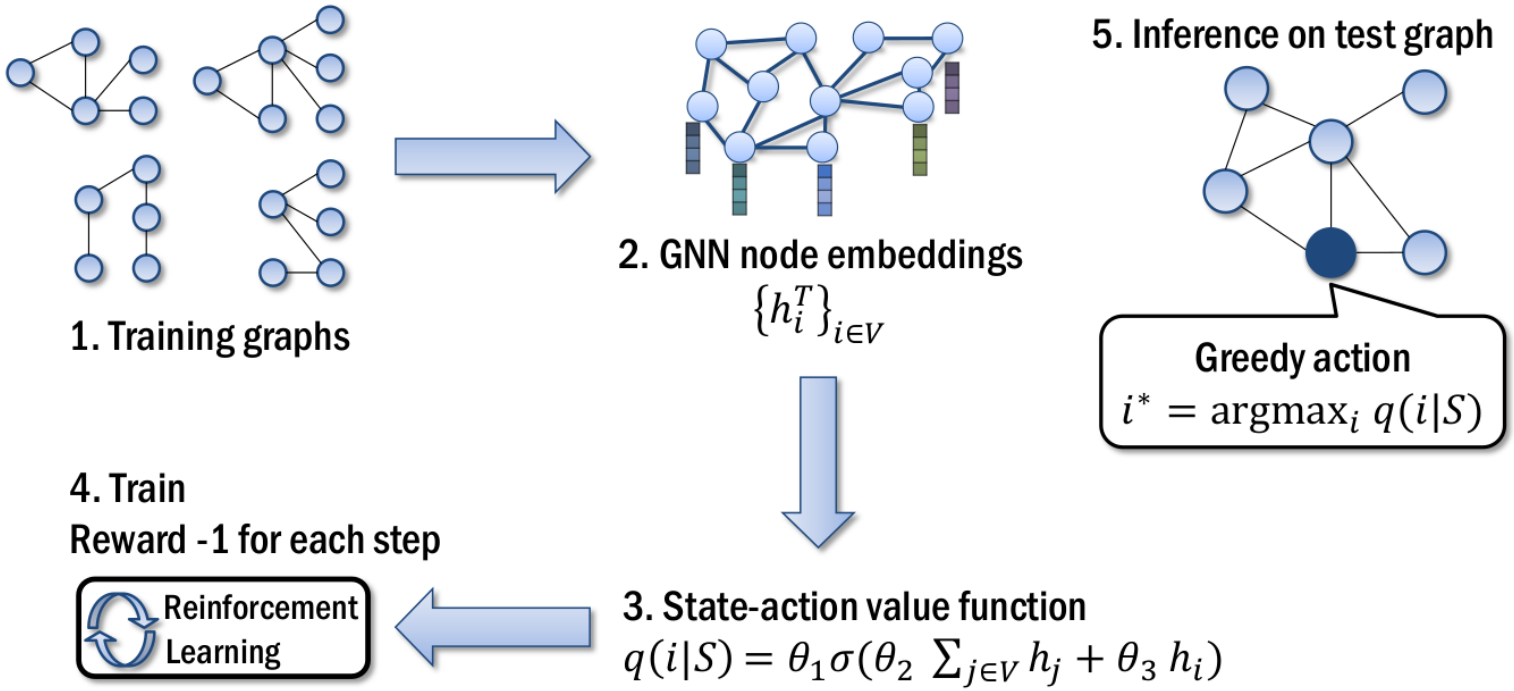
Part VI. Summary and open problems

Part IV. Learning IM

- Learning combinatorial optimization problems
- Learning Influence Maximization
  - Graph Neural Networks
  - Deep Reinforcement Learning

Learning Combinatorial Optimization

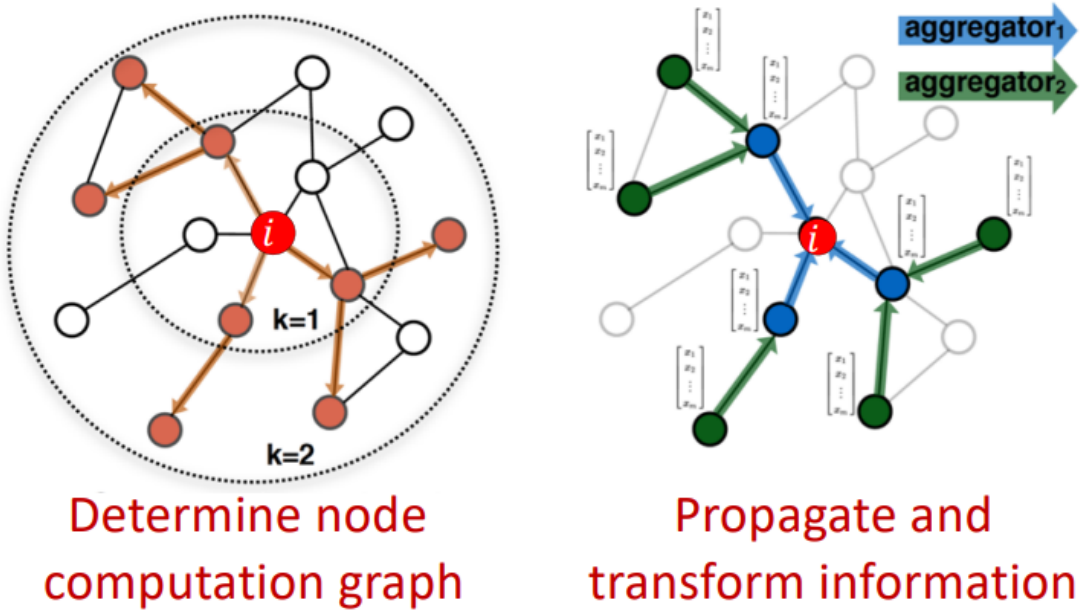
- Can we learn heuristics for combinatorial optimization?
  - Use graph representation learning to capture the state of a graph.
  - Use reinforcement learning to learn how to make sequential decisions.



[Dai et al., NeurIPS '17]

Graph Neural Networks Basics

- Learn how to represent nodes using a weighted combination of their features and their neighbors' features



Determine node computation graph

Propagate and transform information

- Mostly used for semi-supervised learning and graph classification

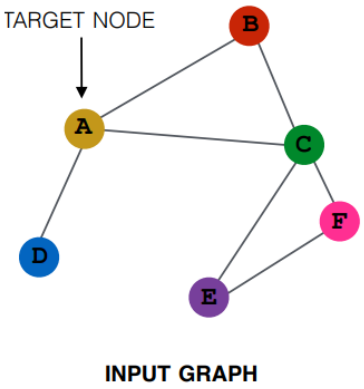
[Hamilton et al., NeurIPS '17]

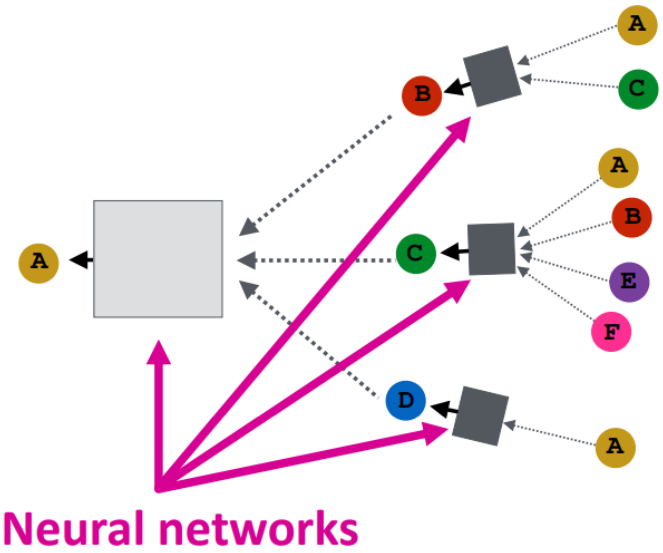
Graph Neural Networks Basics

- Each node aggregates the features of her neighbors using a parameterized non linear combination, i.e. a neural network
- Train the model to adjust the $\mathbf{W_o}\&\mathbf{W_t}$ parameters such that the outcome is optimized

- Each layer's node representation is used as input to the next layer

- The node's representation in the final layer is utilized for the end task





$$h_v^{l+1} = \sigma\left(W_o^l\left(\frac{1}{|N(v)|}\sum_{u\in N(v)} h_u^l\right)+W_t^l h_v^l\right)$$

[Jure Leskovec, Stanford CS224W: Machine Learning with Graphs, http://cs224w.stanford.edu]

Learn CO with DQL

- Given graph learning embeddings, we learn the greedy sequential algorithm for the minimum vertex cover problem using Q-learning:
- In every step, choose a node at random or based on the Q function (e-greedy)
  - The nodes used up to now are tagged with a 1 in the initial representation $\mathbf{h_0}$
  - The Q function uses the node embedding and the sum of node embeddings derived from $\mathbf{l}$ layer to represent the action and the graph state:

$$Q(h(S), v|W) = W_f \sigma([W_S \sum_{u \in V} h_u^l, W_n h_v^l])$$

- The policy is greedy and deterministic: always the node with the highest Q-value is used
- Keep choosing nodes until all edges are covered
- One episode corresponds to solving the problem i.e. repeating steps 1-4 for one graph

[Dai et al., NeurIPS '17]

FINDER

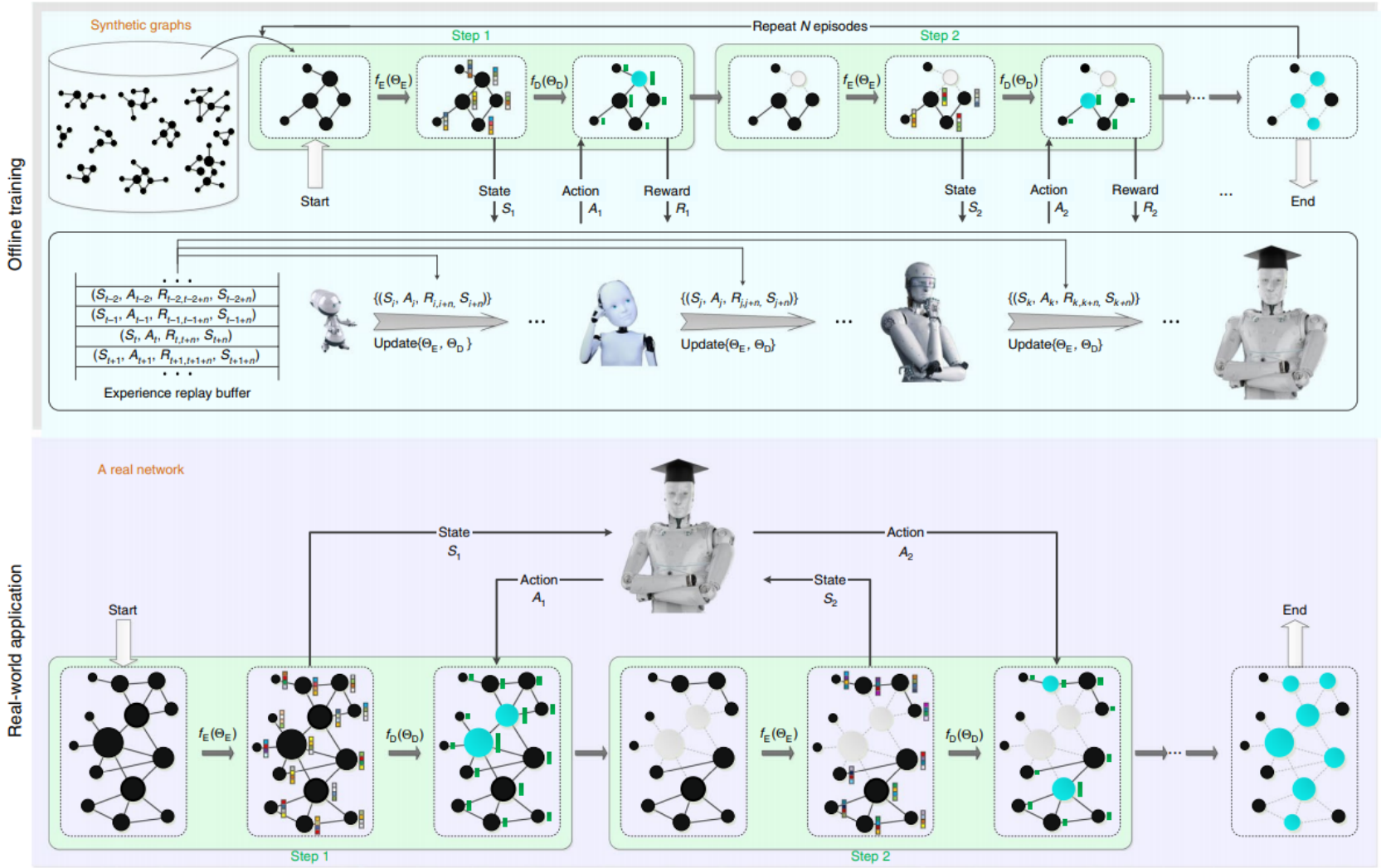
- Learn how to maximize influence through dismantling the network:
  - Iteratively find the nodes that would decrease the size of the giant connected component

- This is equivalent to influence maximization in the linear threshold model [Morone and Makse, Nature 2015]

- Use GraphSage to compute the graph into node and graph representations.
- In each step, DQN chooses an action, i.e. a node to remove
- The new state is the graph after removing the chosen seed and its edges.
- The reward is the accumulated network connectivity of chosen seed set S

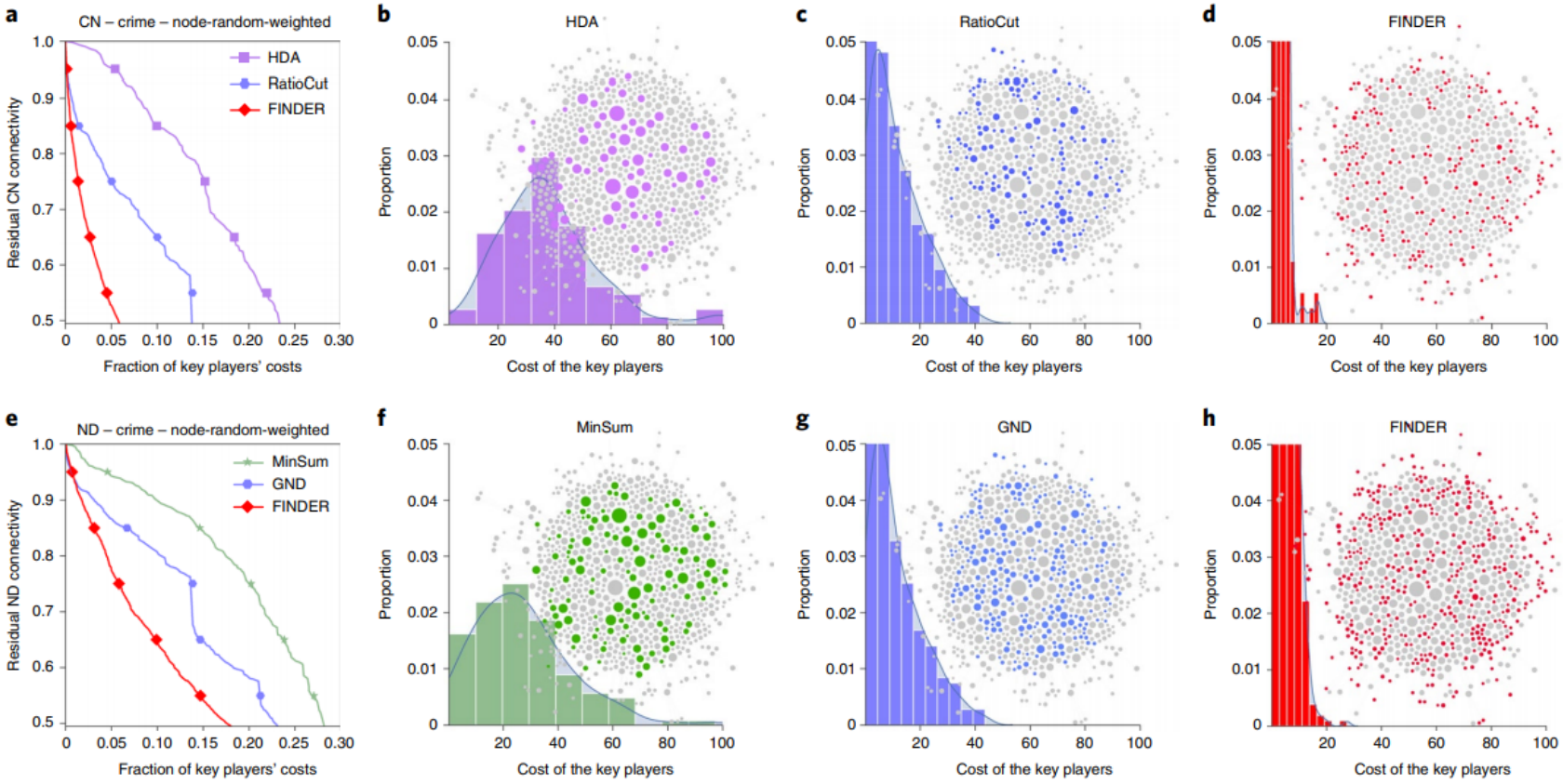
[Fan et al., Nature Machine Intelligence '20]

FINDER



[Fan et al., Nature Machine Intelligence '20]

FINDER



[Fan et al., Nature Machine Intelligence '20]

Part IV Summary

- Learning heuristics for combinatorial optimization
  - Graph and node embedding for encoding
  - Q learning for decoding
  - Used for minimum vertex cover, maximum cut etc.

- Learn influence maximization
  - Optimize for the network dismantling process

Outline of the Tutorial

Part I. Introduction

Part II. Traditional influence maximization

Part III. Influence and diffusion learning

Part IV. Learning influence maximization

Part V. Online influence maximization

Part VI. Summary and open problems

Part IV. Online Influence Maximization

- Repetitive Campaigns
- Multi-armed bandits with edge feedback

Online Influence Maximization

- What if we need to estimate the probabilities of influence without having previous diffusion cascades?
  - Learn while generating data
- Perform multiple rounds of IM. Use the influence spread in each round to estimate the network probabilities

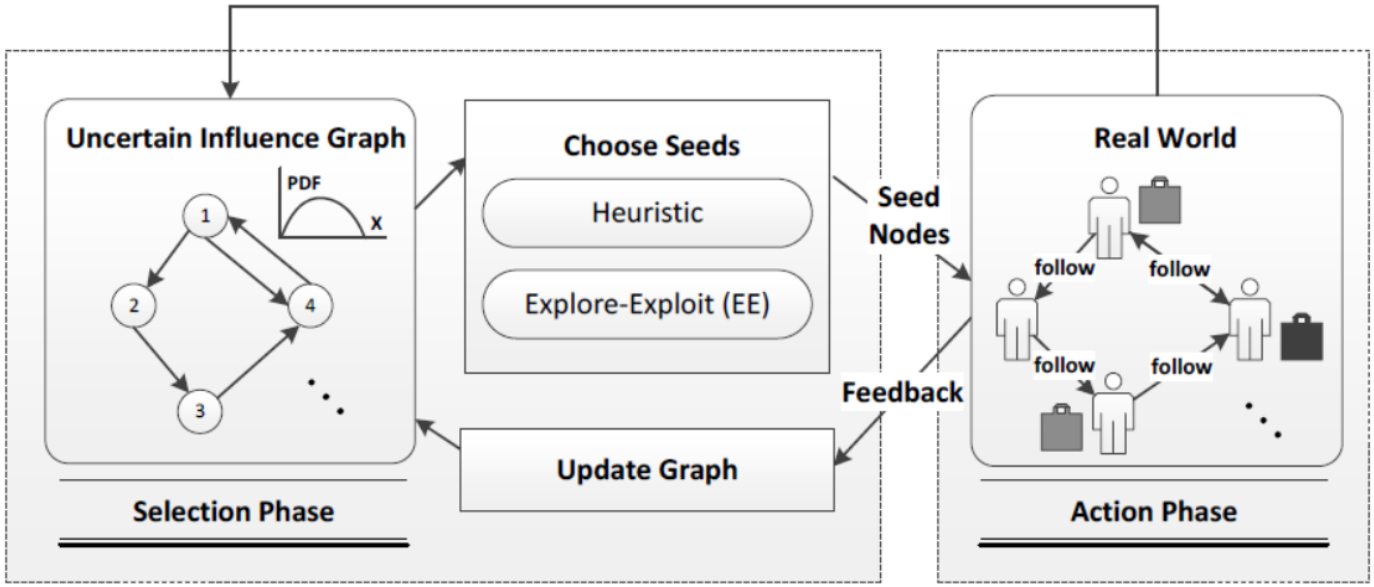
- Given a budget of N trials with k activated nodes, find the seed set of size n that maximizes the influence spread throughout all trials

- Exploitation vs. Exploration:
  - Maximize the influence spread of the algorithm in each round
  - In the same time, estimate the influence probabilities in the edges

[Lei et al., KDD '15]

Online Influence Maximization

- Assign to each edge a Beta distribution
- Run an IM algorithm on the graph using the current edge probabilities
- Receive a feedback based on the influence spread estimated
- Update the edge probabilities



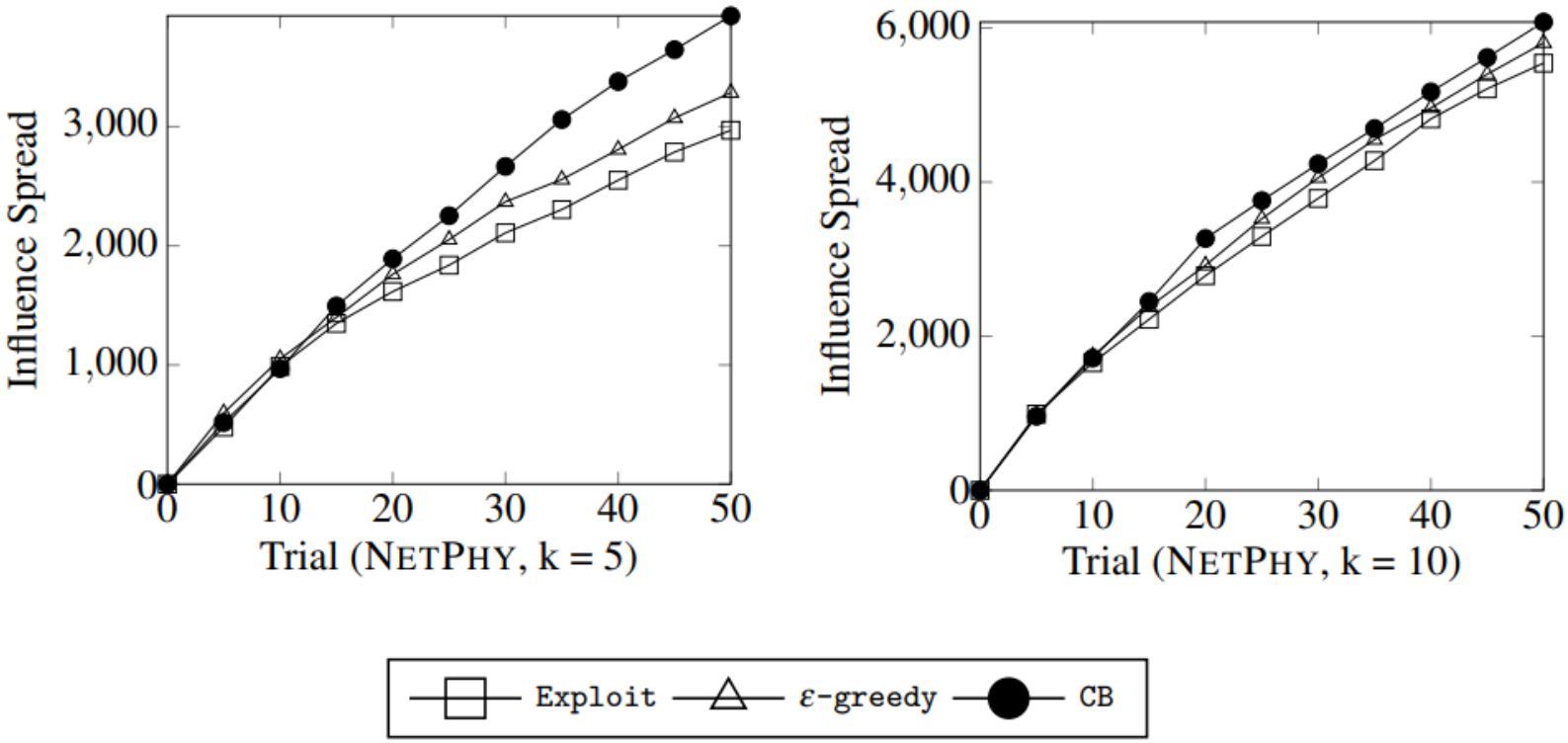
[Lei et al., KDD '15]

Online Influence Maximization

- Assign a beta distributions in each edge e:
  - $\mu_e = \frac{a_e}{a_e+b_e}$, $\sigma_e = \frac{1}{a_e+b_e}\sqrt{\frac{a_e b_e}{a_e+b_e+1}}$
  - $p_e = \mu_e + \theta\sigma_e$
  - θ=0 in exploitation, θ=1 exploration to increase the variance
- Local update: Increment $\mathbf{a_e}$ when the edge activates and $\mathbf{b_e}$ when it does not
- A θ used in a successful round, should be more likely to be reused in the future
  - Learn θ using exponentiated gradient [Cesa-Bianchi et al. 2006]

[Lei et al., KDD '15]

Online Influence Maximization



[Lei et al., KDD '15]

Combinatorial Multi-Armed Bandits

- Each node is an arm in a multi-armed bandit
  - M arms with one reward each
  - Pull arms in T rounds and receive reward
  - Arm pull is binary, hence MLB is suitable for the task

- Begin with uniform priors and choose seeds at each step

- Minimize the difference between choosing the best arm and the chosen algorithm i.e. the regret R

$$R_t = E[\sum_{i=1}^{t} R^*(i)] - E[\sum_{i=1}^{t} R(i)]$$

[Cautis et al., Tutorial KDD '19]

Multi-Armed Bandits for IM

- Arms are the edges and have an unknown expectation
- In each round, a super-arm consisting of a subset of the e arms $S \subseteq 2^e$ is selected outgoing from at most k nodes
- Oracle: Use the greedy using the current probabilities in each step to find the best set of nodes
- The diffusion is run and the outcomes of all edges is revealed.
- The reward of a super-arm depends on the expected reward of all arms and the arms in S

[Chen et al., JMLR '16] [Cautis et al., Tutorial KDD '19]

Multi-Armed Bandits for IM

ALGORITHM 3: – CUCB

Input: Arms $[m]$, Oracle algorithm

1: Maintain T_i – total number of times arm i has been played, the estimated mean $\hat{\mu}_i$
2: For each arm i, play an arbitrary super-arm $S \in \mathcal{S}$ such that $i \in S$ and update T_i and $\hat{\mu}_i$
3: $t \leftarrow m$
4: **while** true **do**
5: $\quad t \leftarrow t + 1$
6: $\quad$ Set each $\bar{\mu}_i = \hat{\mu}_i + \sqrt{\frac{3 \ln t}{2 T_i}}$
7: $\quad S = \text{Oracle}(\bar{\mu}_1, \ldots, \bar{\mu}_m)$
8: $\quad$ Play S and update each T_i and $\hat{\mu}_i$
9: **end while**

[Chen et al., JMLR '16] [Cautis et al., Tutorial KDD '19]

Part V Summary

- Influence maximization in multiple rounds
- Learn the influence parameters while finding the best seeds
  - Observe in each round which edges are activated by the seed set

- An upper confidence bound on edge probabilities
- MAB with edge-based feedback

- Left out semi-bandit feedback [Wen et al NeurIPS 2017] and model-independent online IM [Vaswani et al ICML 2017]

Outline of the Tutorial

Part I. Introduction

Part II. Traditional influence maximization

Part III. Influence and diffusion learning

Part IV. Learning influence maximization

Part V. Online influence maximization

Part VI. Summary and open challenges

Part V. Summary

- Overview
- Research directions
- List of references

Summary

Algorithmic tools

and

Machine Learning models

to

(understand
maximize
predict)

influence spreading

in

social and information networks

Open Research Challenges

- Topic-aware IM using information-rich cascades
  - Content, user profiles, locations, time
- Online and adaptive IM
- Learn IM for the independent cascade model
- Use influence for efficient/scalable GNNs
- Use IM to find nodes for adversarial attacks

Thank You! Questions?

George Panagopoulos

École Polytechnique, France

george.panagopoulos@polytechnique.edu

https://geopanag.github.io

Fragkiskos D. Malliaros

CentraleSupélec, Inria, Paris-Saclay University

fragkiskos.malliaros@centralesupelec.fr

http://fragkiskos.me

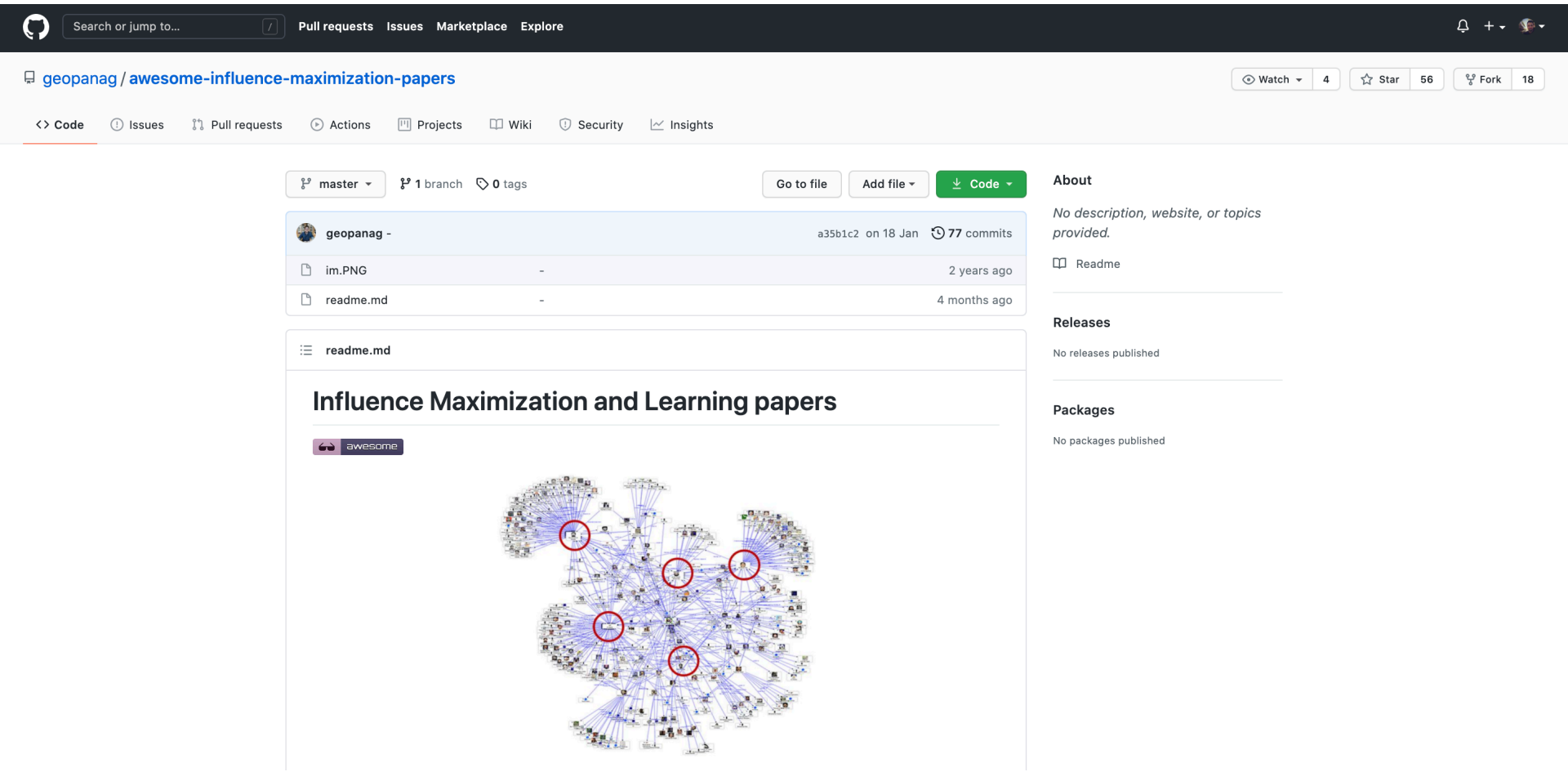
Supported by:

PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR

ANR

Tutorial material at: http://fragkiskos.me/projects/influence_learning_tutorial/

References



https://github.com/geopanag/awesome-influence-maximization-papers