Maximizing the Spread of Influence through a Social Network

David Kempe

University of Washington

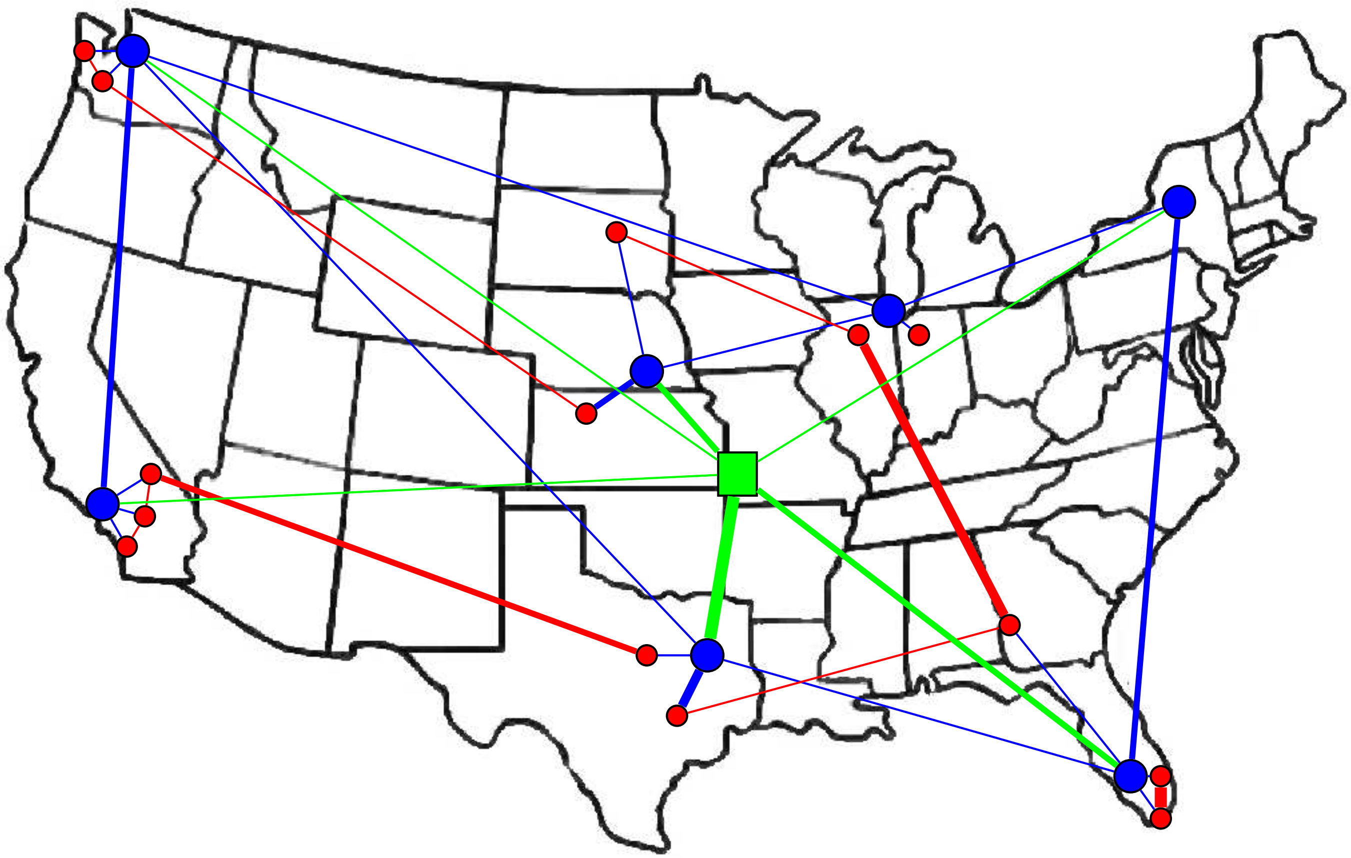
(joint work with Jon Kleinberg and Éva Tardos)

University of Washington Colloquium
Thursday, January 15, 2004

An Illustrative Example

Example: Adoption of a new drug by doctors and patients.

How do we reach many individuals?



Problem Outline

Goal: Use budget to reach many individuals

Examples: Market a product, spread an innovation, propagate a behavior.

☞ Individuals interact and influence each other in complex ways. They may do "word-of-mouth marketing" for us.

- Form models of influence in social networks.
- Obtain data about particular network.
- Devise algorithm to maximize spread of product.

Optimization problem first introduced by Domingos/Richardson [KDD '01/KDD '02]

Outline of Talk

- Models of influence
- Algorithm
- Outline of analysis
- A more general model
- Loose ends
- Experiments
- Conclusions

Models of Influence

- Collective behavior of individuals well-studied area of sociology.
- First mathematical models:
  [Schelling '70/'78, Granovetter '78]
- Large body of subsequent work:
  [Rogers '95, Valente '95, Wasserman/Faust '94]
- Two classes of models: threshold and cascade

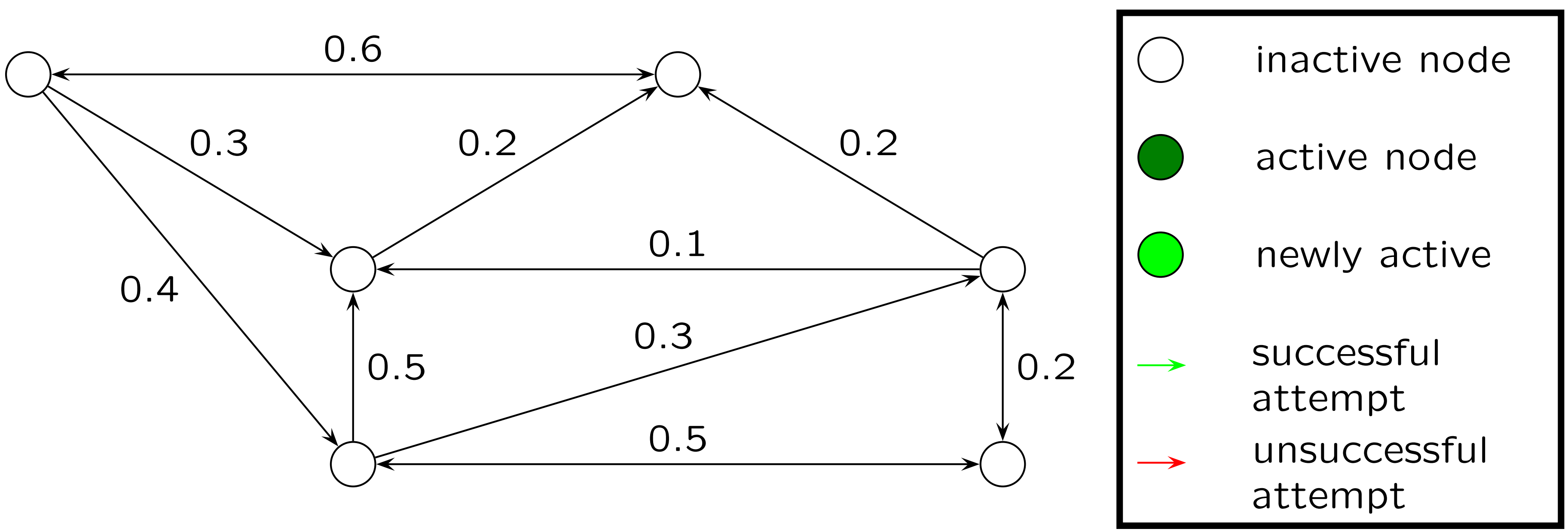
General operational view:

- Some nodes start active (bought the product).
- Active nodes may cause others to activate, etc.
- Monotonicity: active nodes never deactivate.

Independent Cascade Model

Independent Cascade, e.g. [GLM '01]:

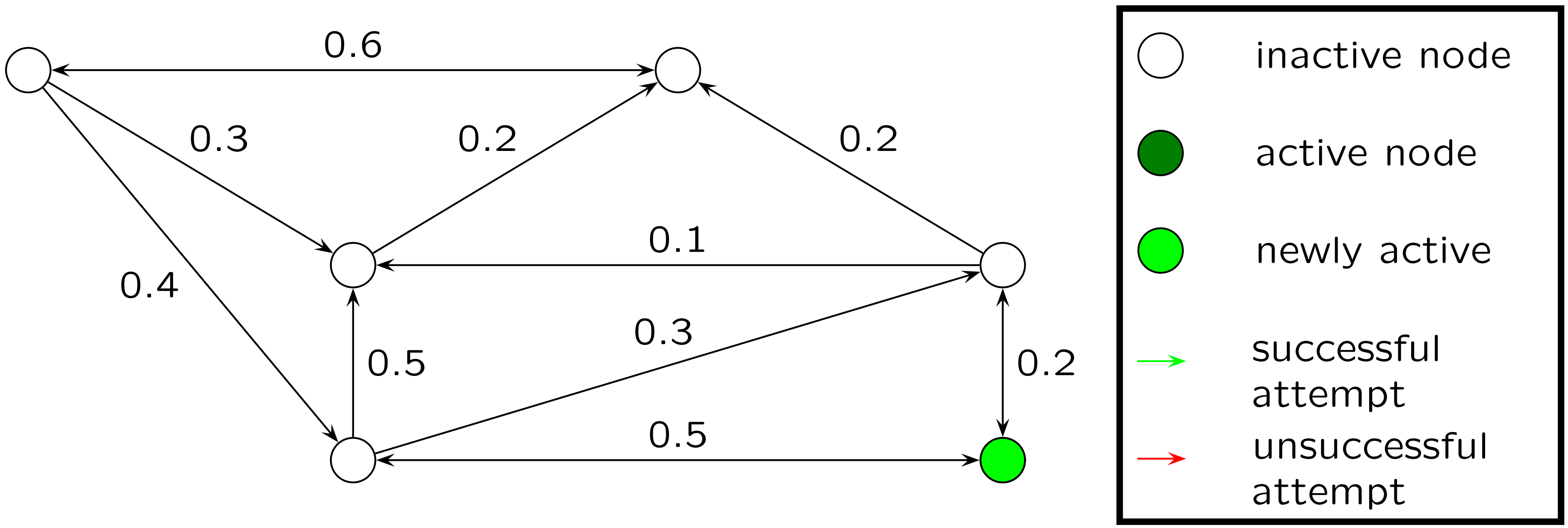
When u becomes active, it has **one** chance of activating each inactive neighbor v with probability p_{uv}.



Independent Cascade Model

Independent Cascade, e.g. [GLM '01]:

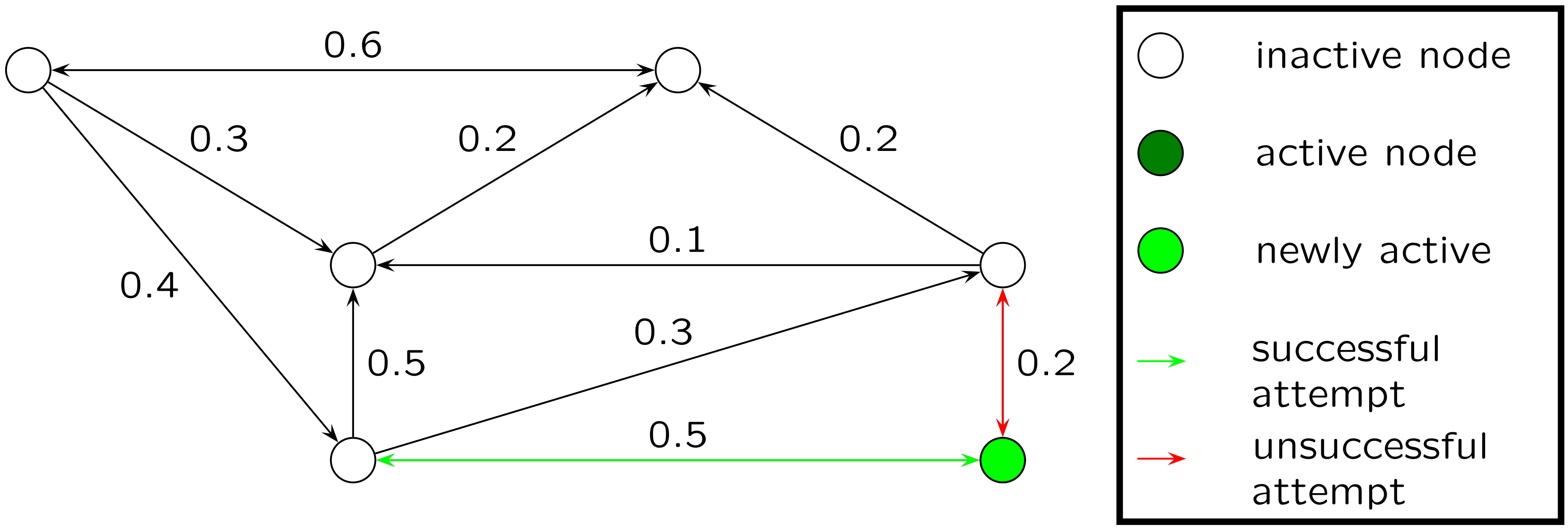
When u becomes active, it has **one** chance of activating each inactive neighbor v with probability p_{uv}.



Independent Cascade Model

Independent Cascade, e.g. [GLM '01]:

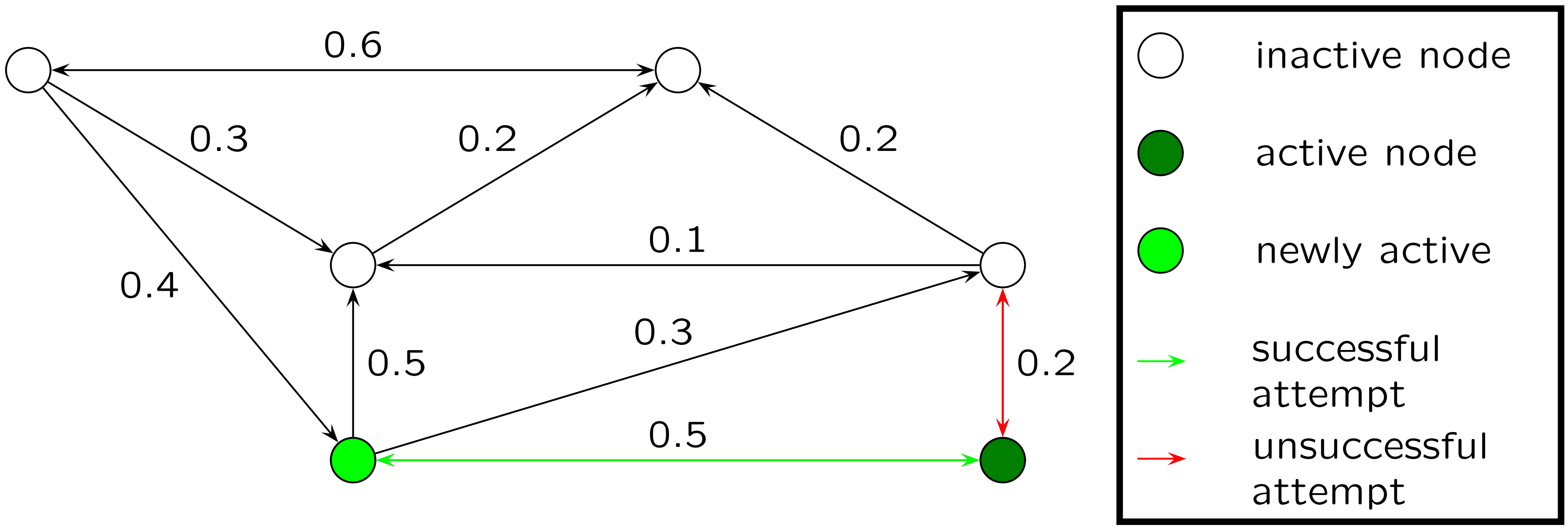
When u becomes active, it has **one** chance of activating each inactive neighbor v with probability p_{uv}.



Independent Cascade Model

Independent Cascade, e.g. [GLM '01]:

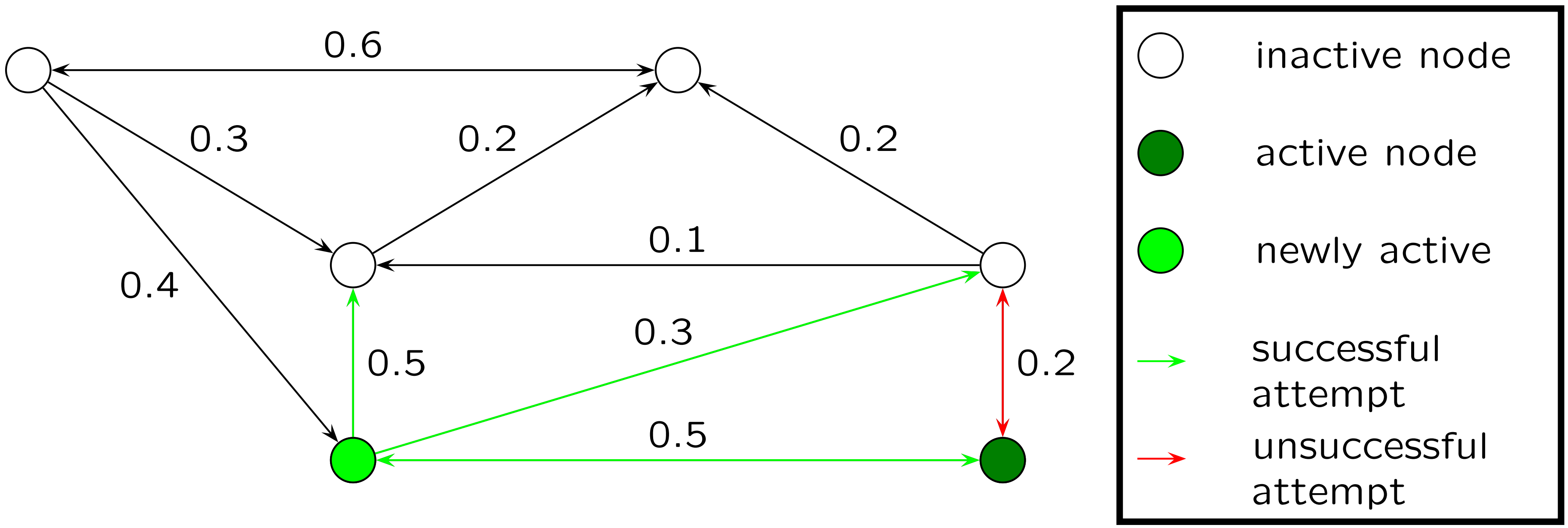
When u becomes active, it has **one** chance of activating each inactive neighbor v with probability p_{uv}.



Independent Cascade Model

Independent Cascade, e.g. [GLM '01]:

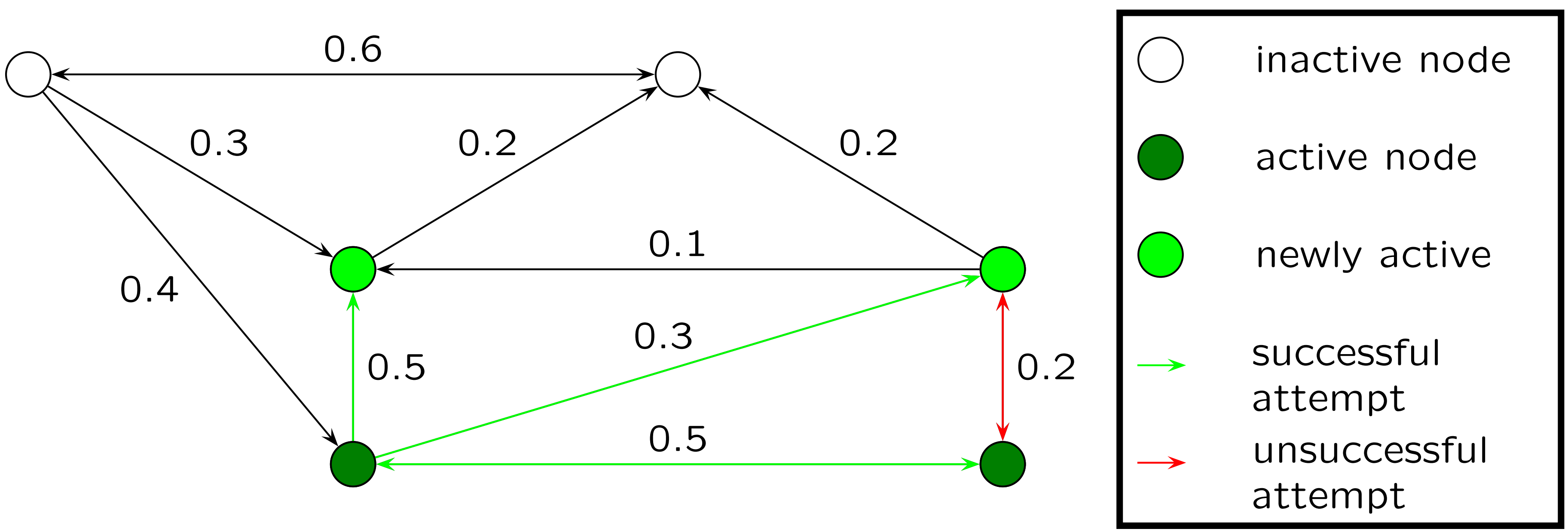
When u becomes active, it has **one** chance of activating each inactive neighbor v with probability p_{uv}.



Independent Cascade Model

Independent Cascade, e.g. [GLM '01]:

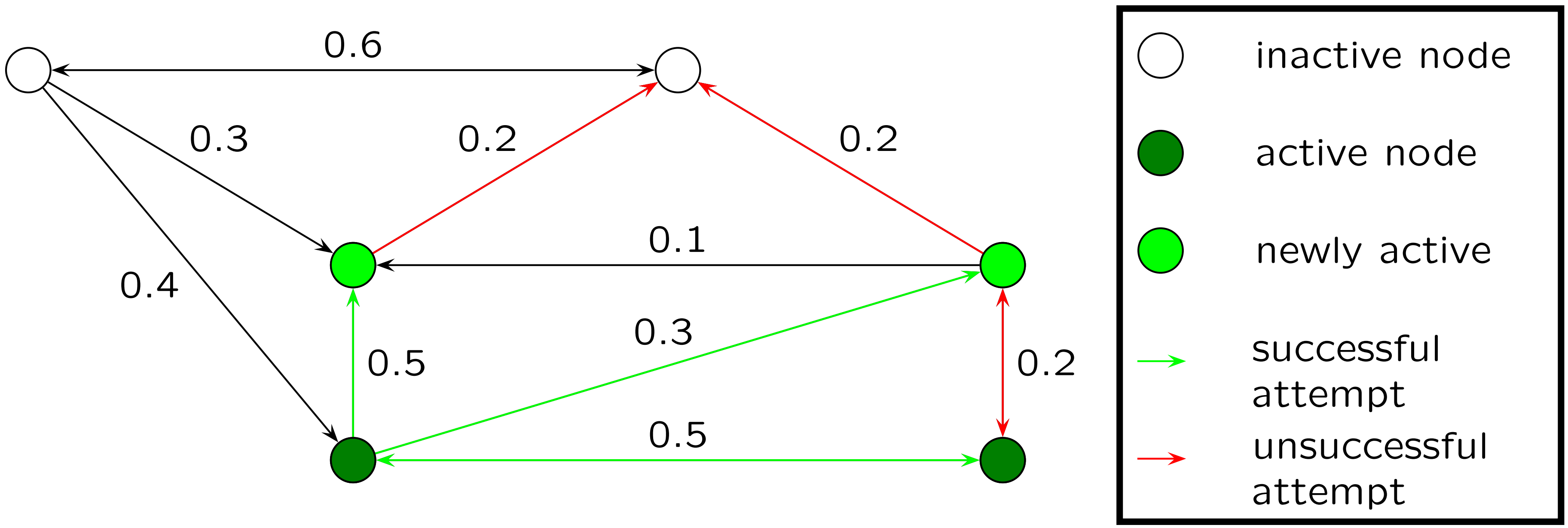
When u becomes active, it has **one** chance of activating each inactive neighbor v with probability p_{uv}.



Independent Cascade Model

Independent Cascade, e.g. [GLM '01]:

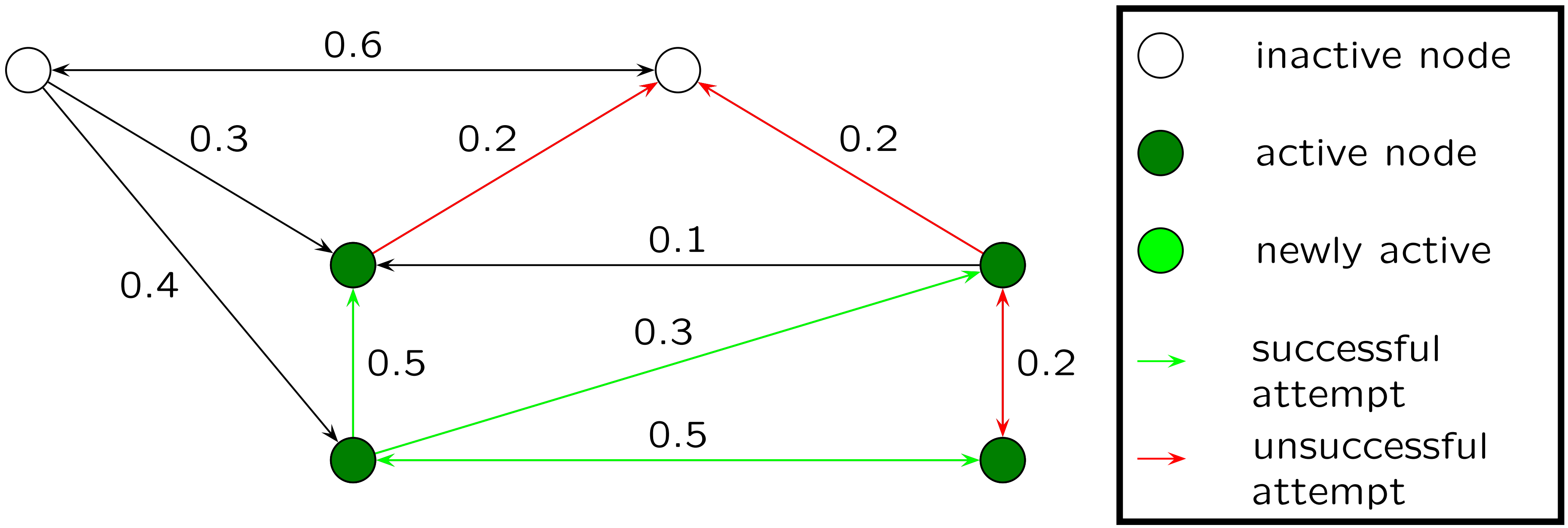
When u becomes active, it has **one** chance of activating each inactive neighbor v with probability p_{uv}.



Independent Cascade Model

Independent Cascade, e.g. [GLM '01]:

When u becomes active, it has **one** chance of activating each inactive neighbor v with probability p_{uv}.

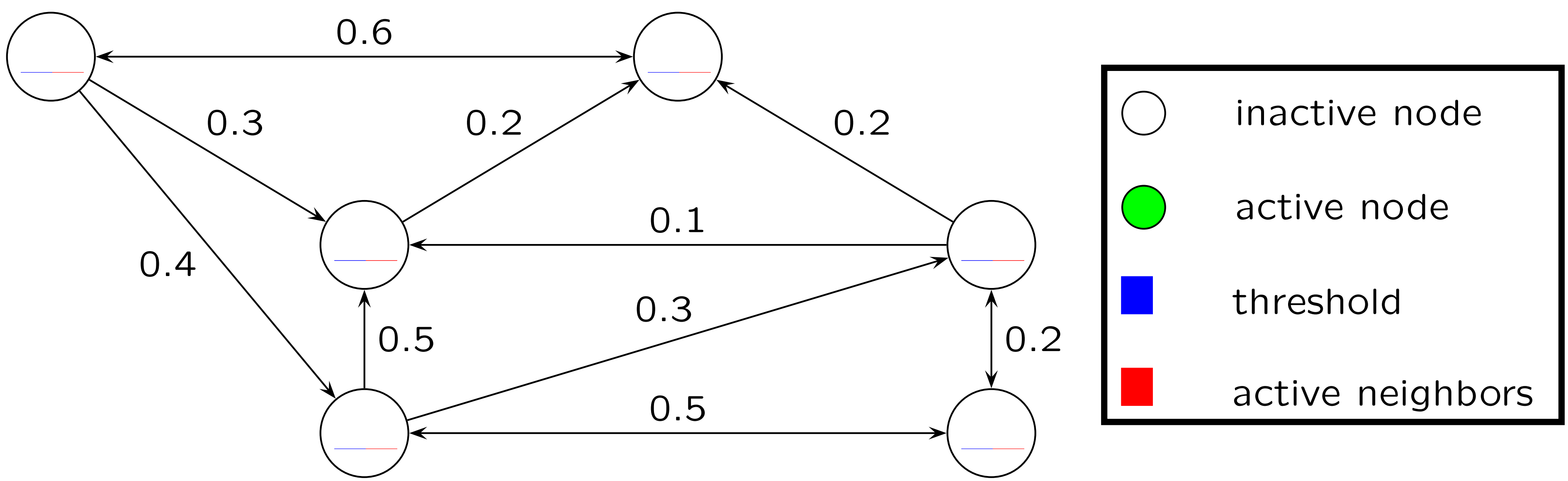


Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0, 1]$.

Node v becomes active when at least (weighted) θ_v fraction of its neighbors are active.



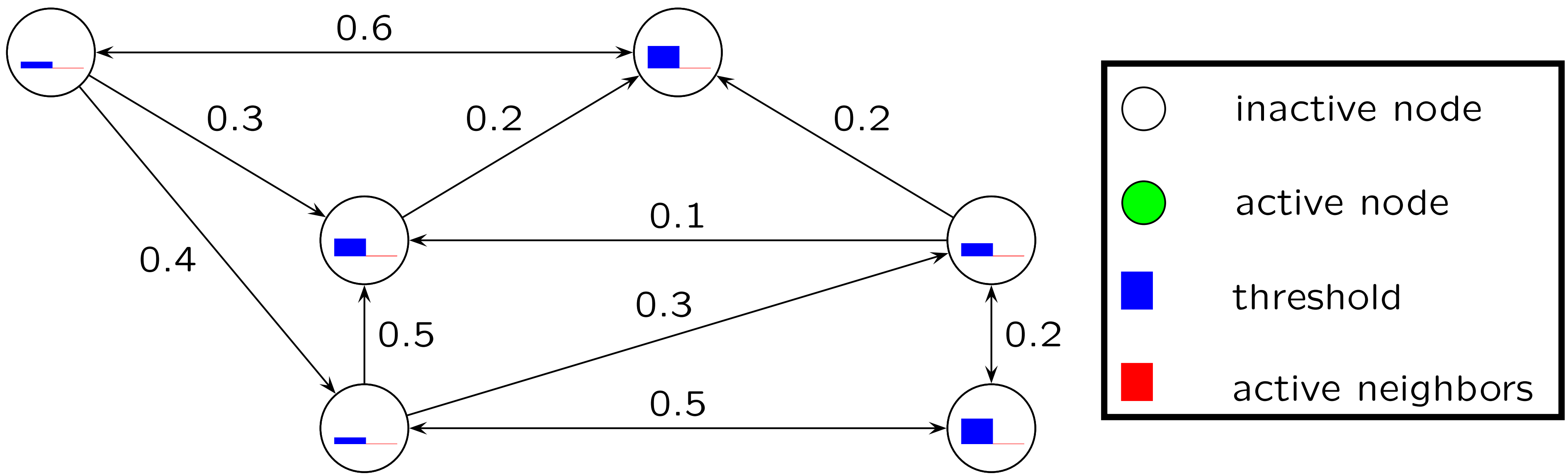
Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0, 1]$.

Node v becomes active when at least

(weighted) θ_v fraction of its neighbors are active.



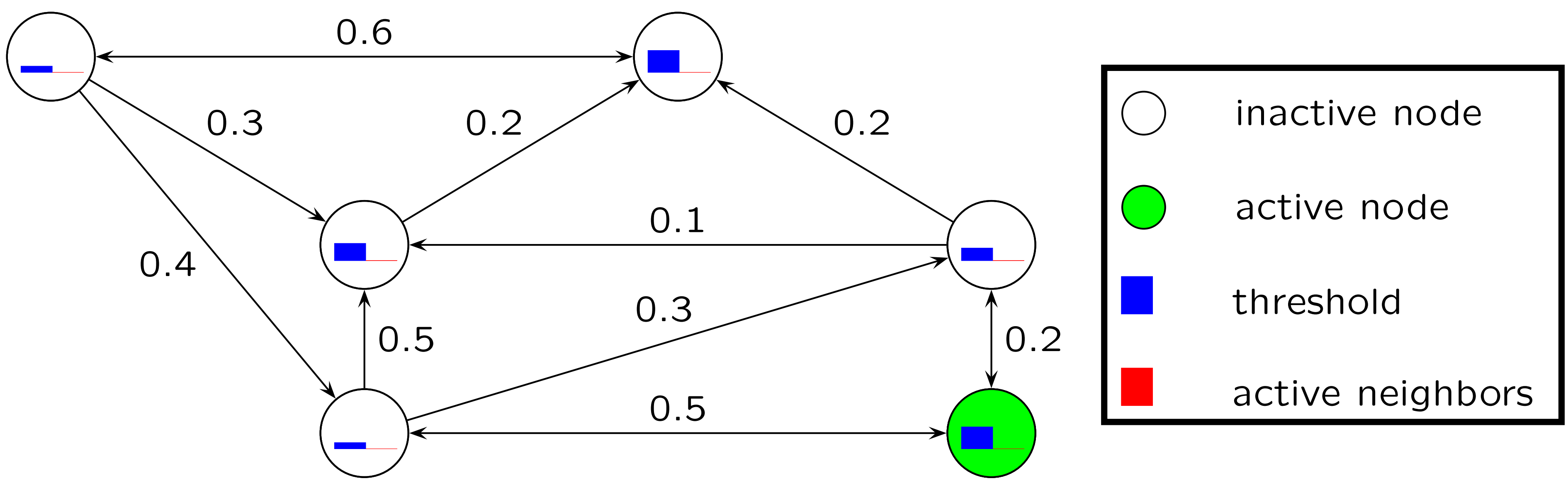
Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0, 1]$.

Node v becomes active when at least

(weighted) θ_v fraction of its neighbors are active.

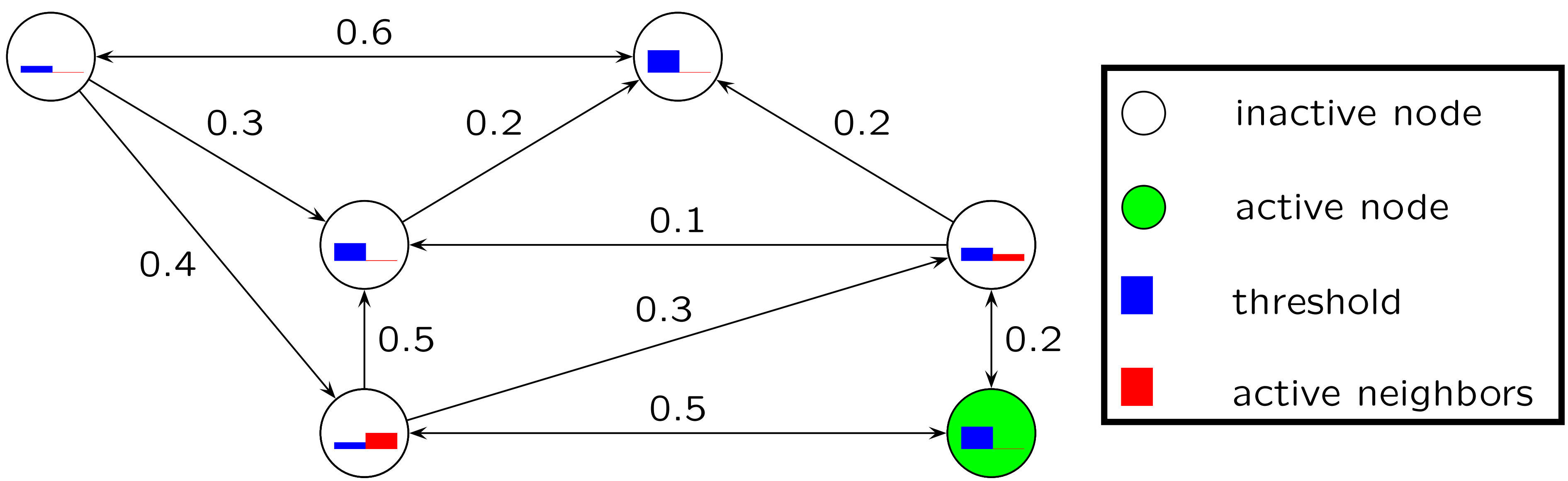


Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0, 1]$.

Node v becomes active when at least (weighted) θ_v fraction of its neighbors are active.

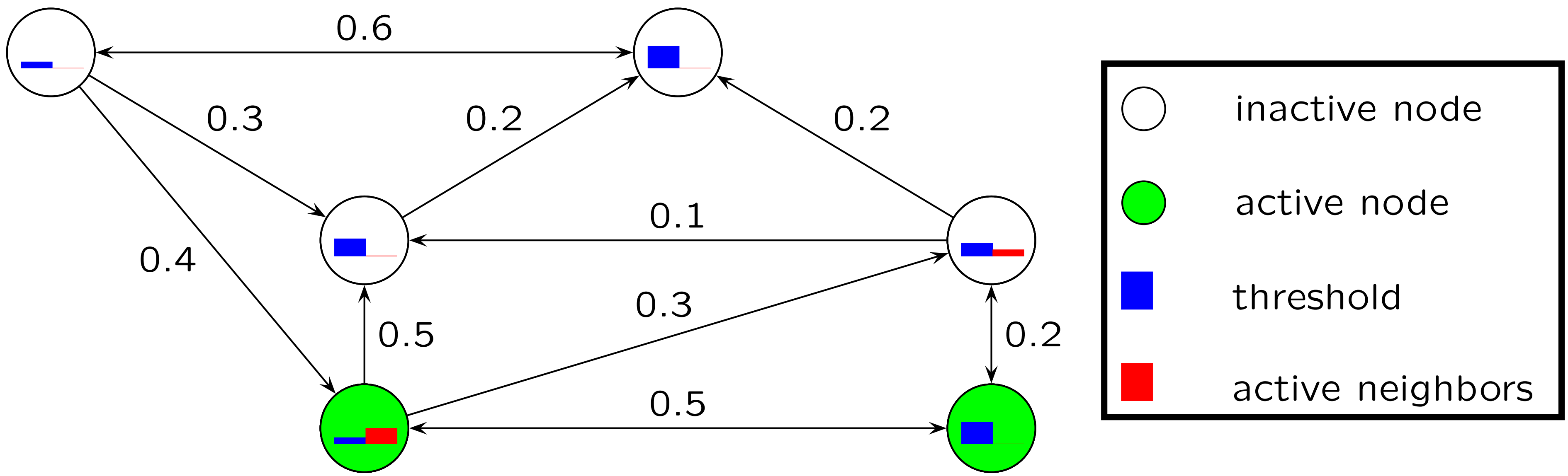


Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0, 1]$.

Node v becomes active when at least (weighted) θ_v fraction of its neighbors are active.



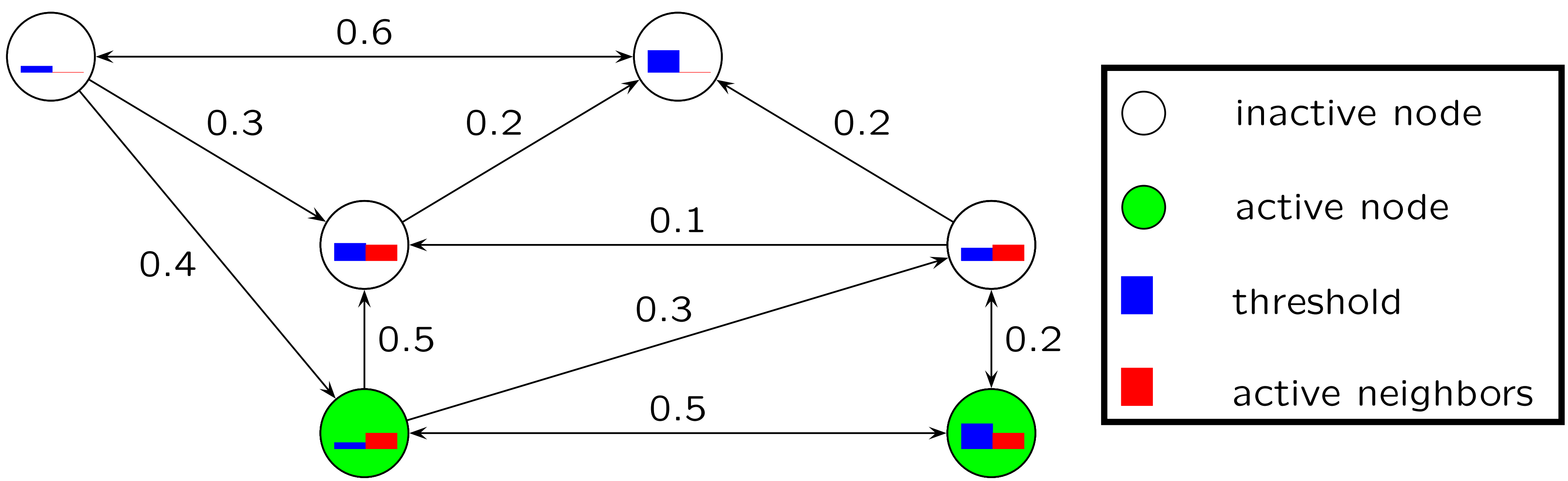
Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0, 1]$.

Node v becomes active when at least

(weighted) θ_v fraction of its neighbors are active.



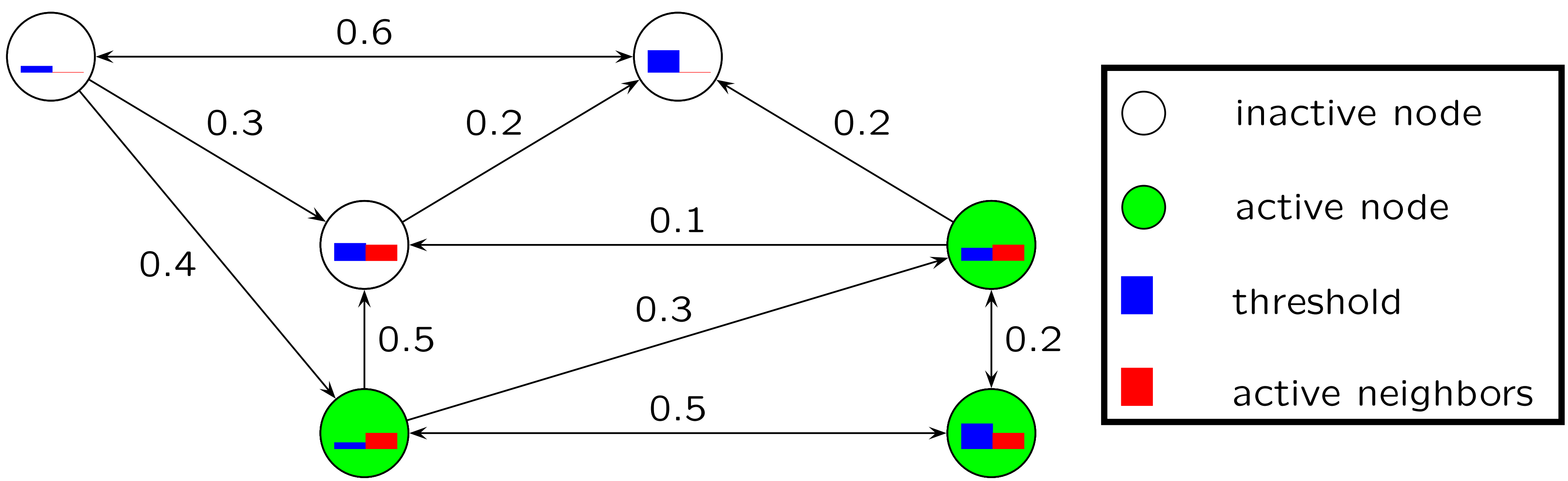
Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0,1]$.

Node v becomes active when at least

(weighted) θ_v fraction of its neighbors are active.

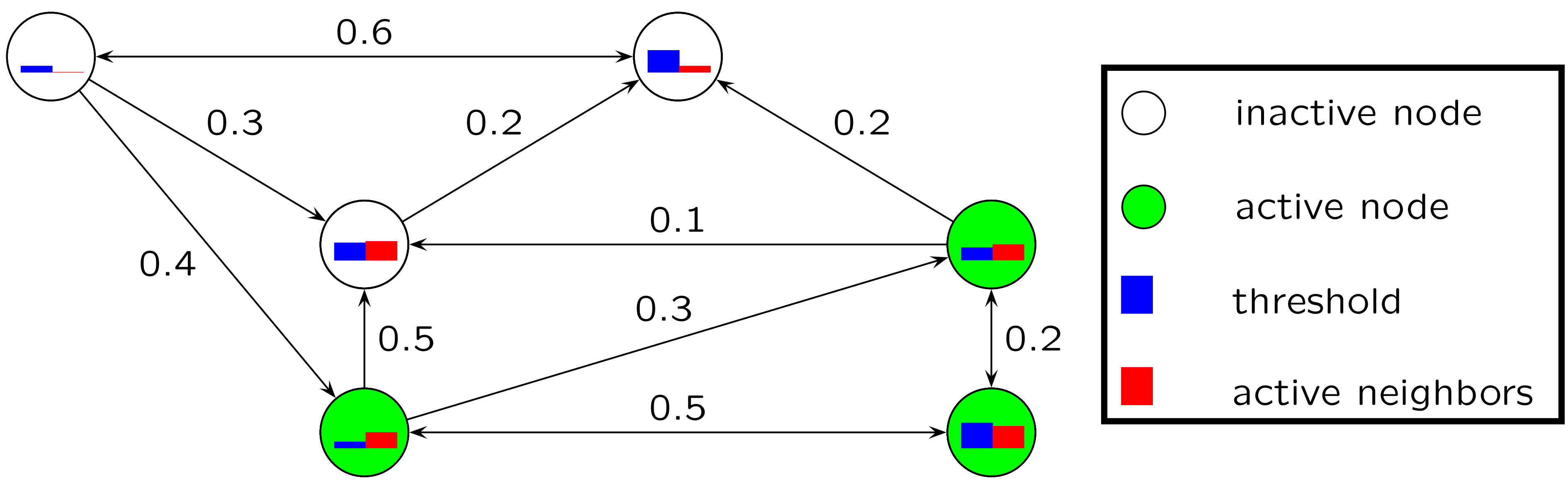


Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0, 1]$.

Node v becomes active when at least (weighted) θ_v fraction of its neighbors are active.



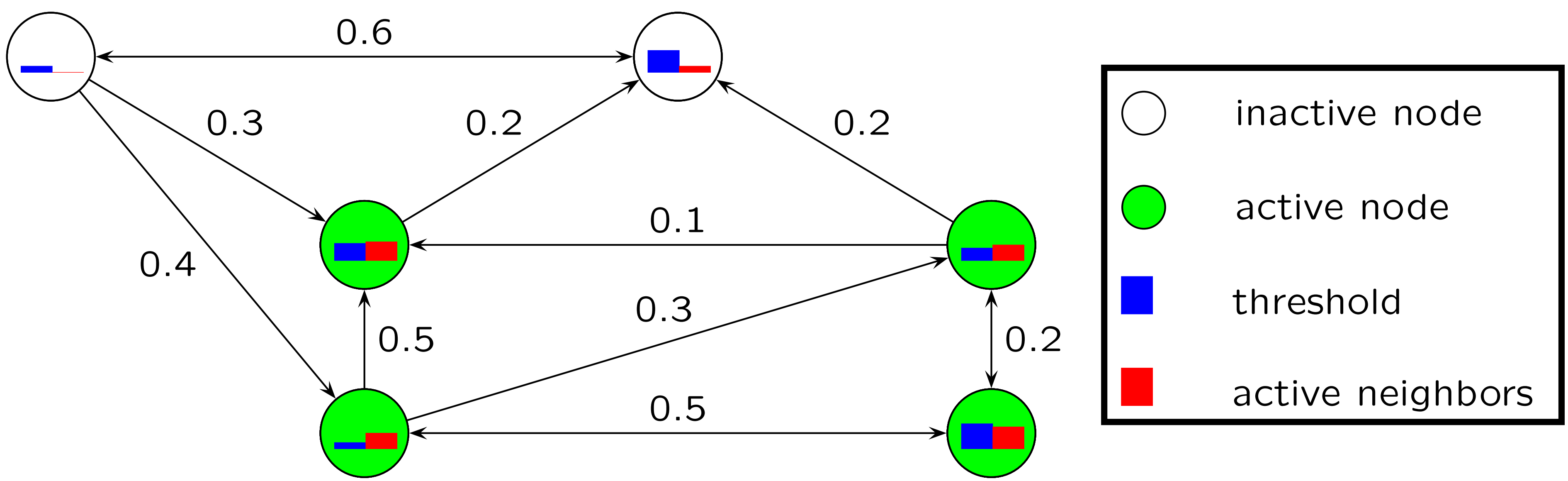
Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0, 1]$.

Node v becomes active when at least

(weighted) θ_v fraction of its neighbors are active.

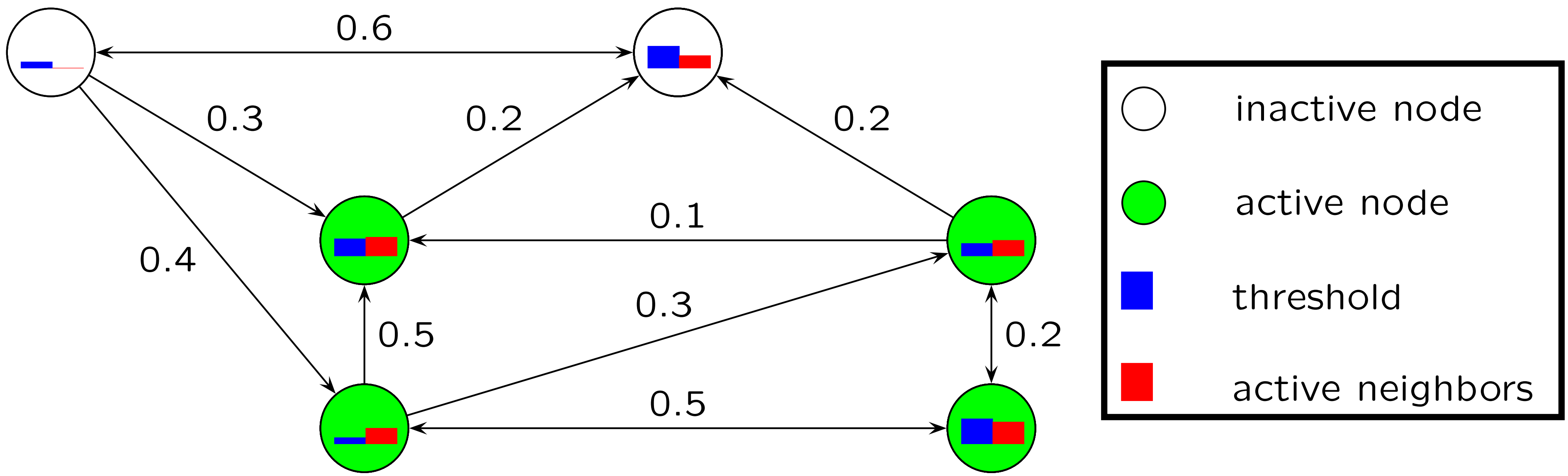


Linear Threshold Model

Threshold Model [Granovetter '78]:

Nodes have random thresholds $\theta_v \in [0, 1]$.

Node v becomes active when at least (weighted) θ_v fraction of its neighbors are active.



Optimization Problem

What is the most influential set of nodes?

Whom should we activate initially to reach many nodes?

(Or **try** to activate?)

$f(S)$: expected number of nodes active at the end,
if set S is targeted for initial activation.

> Given a budget B, select a set S of B nodes,
> so as to maximize $f(S)$.

☞ Problem is NP-hard. Look for approximate solutions.

Approximation Algorithm

Greedy Algorithm:

For B iterations:

Add node v to S that maximizes $f(S+v) - f(S)$.

Theorem:

The greedy algorithm is a $(1 - 1/e)$ approximation.

The set S found activates at least $(1 - 1/e) > 63\%$ of the number of nodes that any size-B set S^* could activate.

Analysis Outline

1. Prove that expected activation f at the end is:

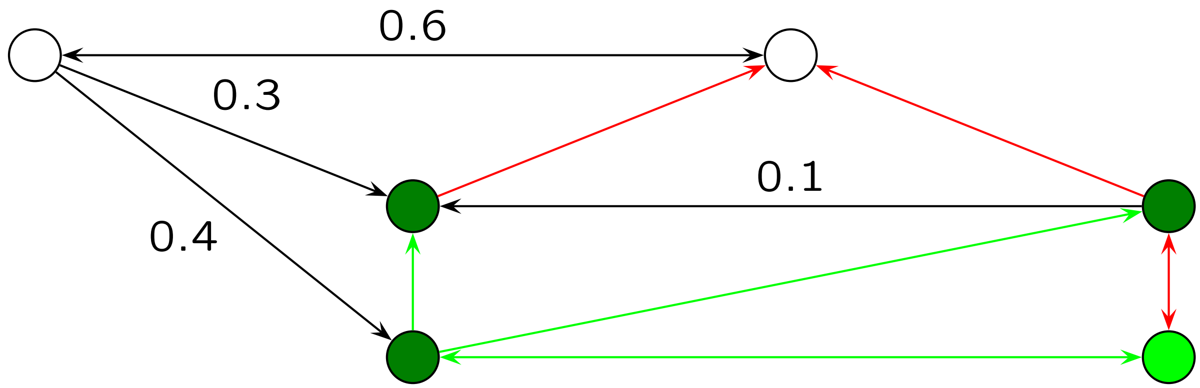
- Monotone: $f(S+v) \geq f(S)$
- Submodular (diminishing returns):
  $f(S+v) - f(S) \geq f(T+v) - f(T)$ whenever $S \subseteq T$.

  Need to understand dynamics of activation.

2. Use Theorem by Nemhauser, Wolsey, Fisher '78:

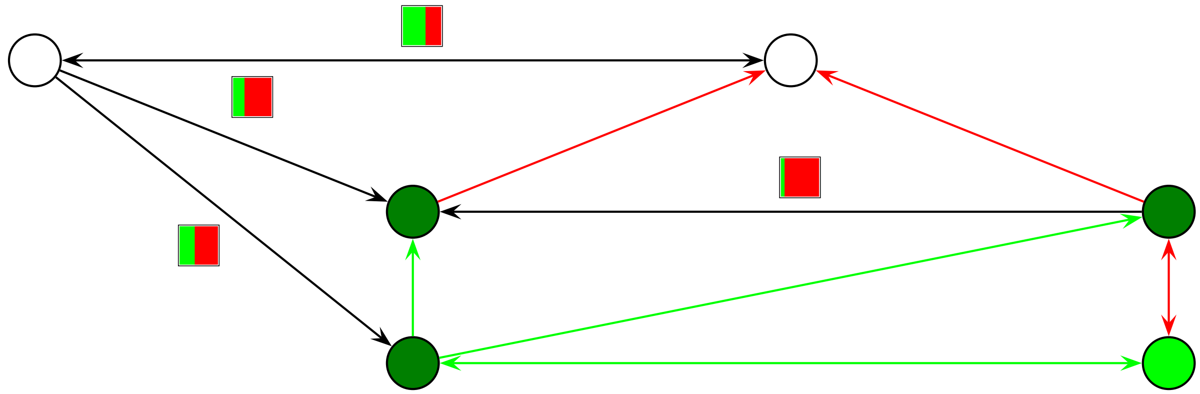
> Whenever f is monotone and submodular, the greedy algorithm for maximization is a $(1-1/e)$ approximation.

Submodularity for Independent Cascade



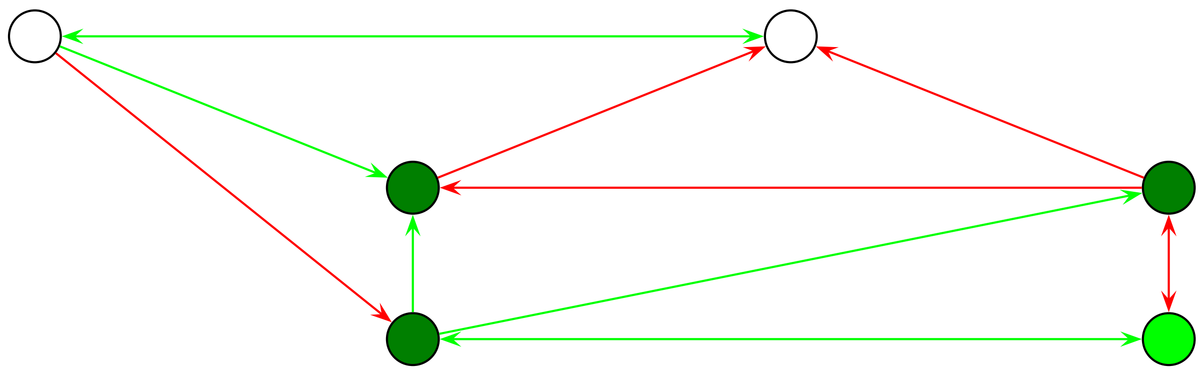
- Coins for edges are flipped during activation attempts.

Submodularity for Independent Cascade



- Coins for edges are flipped during activation attempts.
- Can pre-flip all coins and reveal results as needed.

Submodularity for Independent Cascade

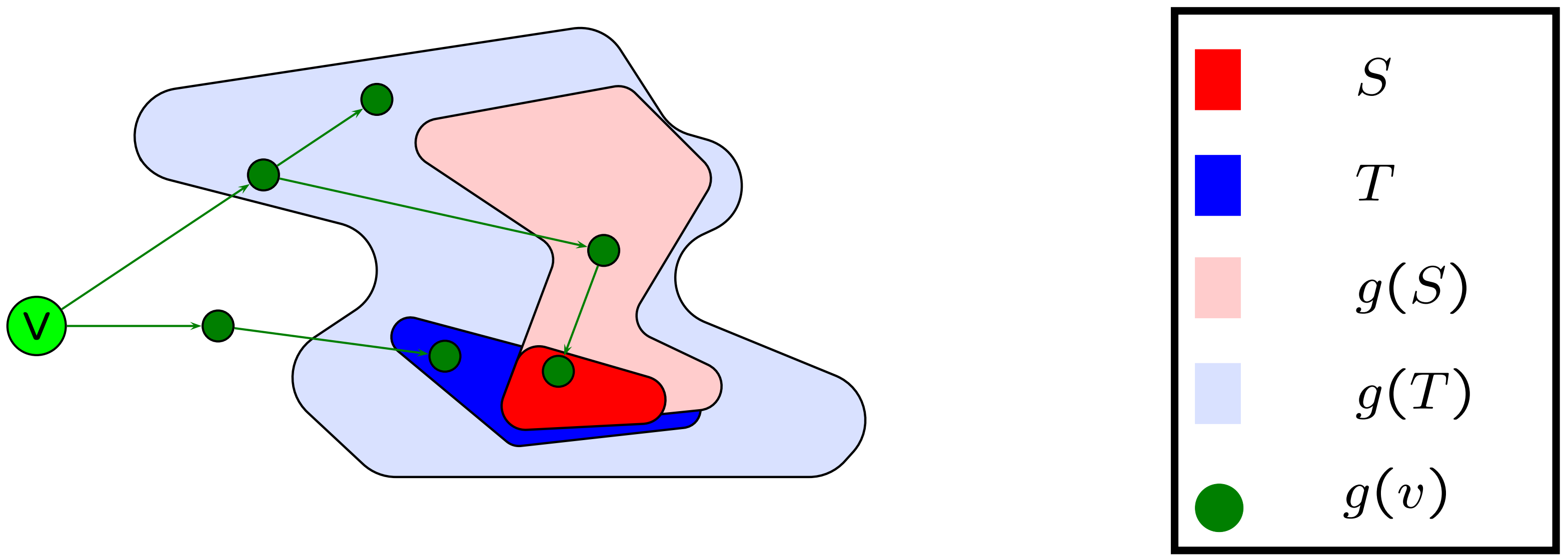


- Coins for edges are flipped during activation attempts.
- Can pre-flip all coins and reveal results as needed.
- Can pre-flip all coins and reveal results immediately.
- Active nodes in the end are reachable via green paths from initially targeted nodes.

➡ Study reachability in green graphs

Submodularity, Fixed Graph

- Fix "green graph" G. g(S) are nodes reachable from S in G.
- Submodularity: $g(T+v) - g(T) \subseteq g(S+v) - g(S)$ when $S \subseteq T$.
- $g(S+v) - g(S)$: nodes reachable from $S+v$, but not from S.
  - ☞ Exactly nodes reachable from v, but not from S.



- From the picture: $g(T+v) - g(T) \subseteq g(S+v) - g(S)$ when $S \subseteq T$.

Submodularity, Wrap-Up

Fact:

A non-negative linear combination of submodular functions is submodular.

$$f(S) \;=\; \sum_{G} \text{Prob}[G \text{ is green graph}] \cdot |g_G(S)|$$

- $g_G(S)$: nodes reachable from S in G.
- Each $g_G(S)$ is submodular (previous slide).
- Probabilities are non-negative.

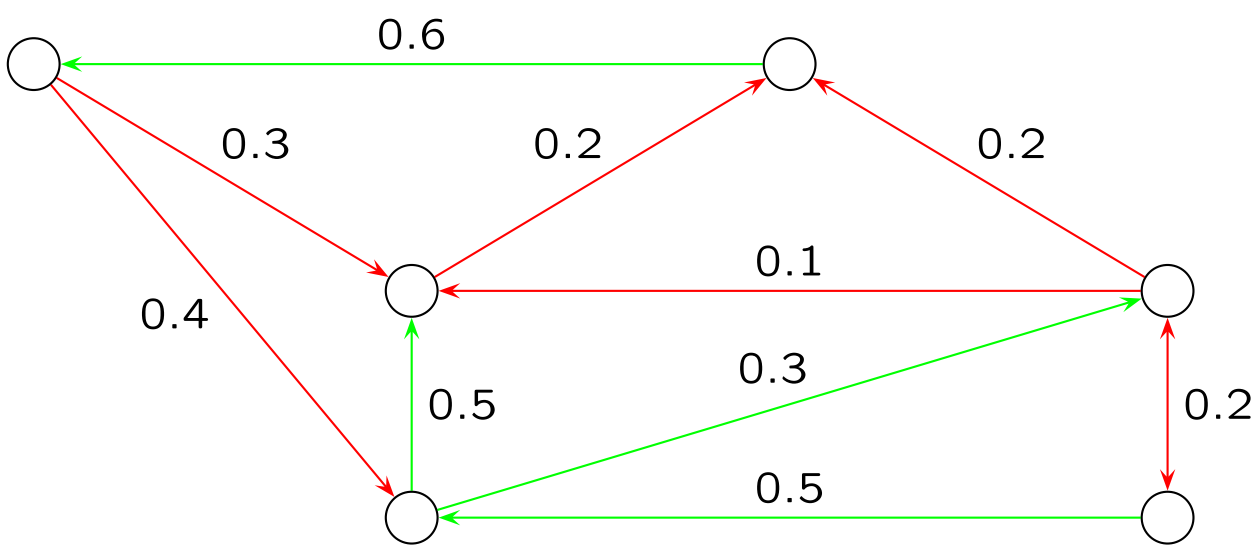
➡ f is submodular.

Submodularity for Linear Threshold

- Use similar “green graph” idea.
- Once a graph is fixed, “reachability” argument is identical.
- How do we fix a green graph now?

Submodularity for Linear Threshold

- Use similar "green graph" idea.
- Once a graph is fixed, "reachability" argument is identical.
- How do we fix a green graph now?
- Each node picks **at most one** incoming edge, with probabilities proportional to edge weights.



- Equivalent to linear threshold model (trickier proof).

A General Model

- Independent Cascade and Linear Threshold are two specific models.
- We would like algorithms for as large a class as possible.
- How to generalize these models?

A General Model

- Independent Cascade and Linear Threshold are two specific models.
- We would like algorithms for as large a class as possible.
- How to generalize these models?

General Threshold Model:

Each node v has activation function $h_v : V \to [0, 1]$.
v becomes active when $h_v(A)$ exceeds v's threshold θ_v.
(A: active neighbors of v)

☞ Linear Threshold: special case where $h_v(A) = \sum_{u \in A} c_{uv}$.

A General Model

- Independent Cascade and Linear Threshold are two specific models.
- We would like algorithms for as large a class as possible.
- How to generalize these models?

- Threshold: general activation functions h_v.

General Cascade Model:

Activation probabilities p_{uv} change as a function of who has already tried and failed (now: $p_{uv}(F)$).

Order-independence: order of attempts does not matter.

☞ Independent Cascade: special case (p_{uv} independent of F).

A General Model

- Independent Cascade and Linear Threshold are two specific models.
- We would like algorithms for as large a class as possible.
- How to generalize these models?
- Threshold: general activation functions h_v.
- Cascade: activation probabilities change.

Theorem:

These two general models are equivalent.

- Can we solve the problem for the general model?

Alas

In general, any non-trivial approximation of f is NP-hard.

How general a model can we handle?

Alas

In general, any non-trivial approximation of f is NP-hard.

How general a model can we handle?

Decreasing Cascade Model: $p_{uv}(F)$ are non-increasing in F.

Theorem:

For the Decreasing Cascade Model, the greedy algorithm is a $(1 - 1/e)$-approximation.

Conjecture: If each activation function $h_v(A)$ is submodular, then $f(S)$ is submodular.

Evaluating f

- To run greedy algorithm, we need to determine most profitable node to target next. That requires evaluating function $f(S)$.
- How to evaluate $f(S)$?

Evaluating f

- To run greedy algorithm, we need to determine most profitable node to target next. That requires evaluating function $f(S)$.
- How to evaluate $f(S)$?
- We don't know! Do you?

Evaluating f

- To run greedy algorithm, we need to determine most profitable node to target next. That requires evaluating function $f(S)$.
- How to evaluate $f(S)$?
- We don't know! Do you?
- By repeating experiment often enough (polynomial in $n, \frac{1}{\epsilon}$), obtain $(1 \pm \epsilon)$-approximation to $f(S)$.
- From this, obtain $(1 - \epsilon)$-approximate best element to add.
- Generalization of Nemhauser/Wolsey proof shows: Greedy algorithm is now a $(1 - 1/e - \epsilon')$-approximation.

☞ Can get arbitrarily close to $(1 - 1/e)$.

Realistic Marketing

- So far: deterministically targeted node set S.
- More realistic: different marketing actions **increase** likelihood of initial activation, for **several** nodes at once.
- Goal: Find optimal investments of budget into marketing actions.

- m different marketing actions.
- Nodes have non-decreasing response function $h_v : \mathbb{R}^m \to [0, 1]$, satisfying diminishing returns in all coordinates.
- With investments $x_1, \ldots, x_m$, nodes become active initially with probability $h_v(x_1, \ldots, x_m)$, independently.
- Then, run process as before. Expectation is now over **both** sources of randomness.

Realistic Marketing, II

Greedy Hill Climbing Algorithm:

Repeat until all of budget is used up:

Add small amount δ of budget to marketing action with largest marginal gain.

The greedy algorithm is a $(1 - 1/e - \epsilon)$-approximation.

Two proof steps:

- Expected activation $f(x_1, \ldots, x_m)$ satisfies diminishing returns condition and monotonicity.
- Hill Climbing is $(1 - 1/e - \epsilon)$ approximation for **any** monotone function with diminishing returns.
(Analogue of Nemhauser/Wolsey Theorem)

Monotonicity

- So far: active nodes never deactivate.
- If they can deactivate, there is no quiescent final state. What are we trying to optimize then?

Monotonicity

- So far: active nodes never deactivate.
- If they can deactivate, there is no quiescent final state. What are we trying to optimize then?
- Sum, over all time steps, of number of active nodes.
  Or weighted by time step (earlier revenue is worth more).
- Marketing actions can affect individuals at different times.

Monotonicity

- So far: active nodes never deactivate.
- If they can deactivate, there is no quiescent final state. What are we trying to optimize then?
- Sum, over all time steps, of number of active nodes.
  Or weighted by time step (earlier revenue is worth more).
- Marketing actions can affect individuals at different times.

Reduction to monotone case:

- One copy of each node for each time. Earlier copies may influence later ones.
- Maximizing number of nodes now corresponds to maximizing sum over all time steps.

Weights

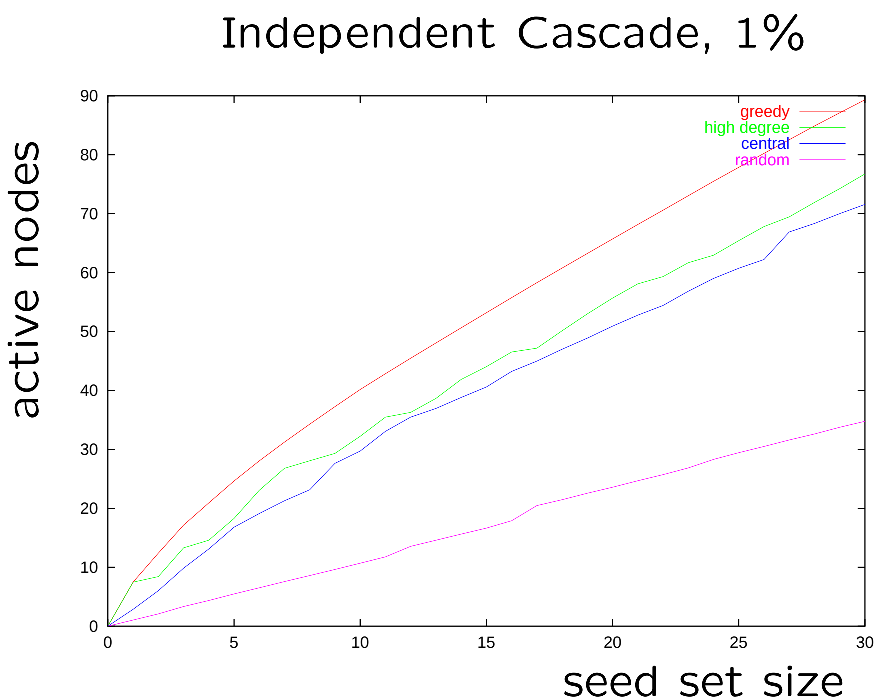
- Nodes may have different values v_i (e.g. order sizes).
- Function f is still submodular, so all of the proof stays the same.
- Alternatively, can replace each node by v_i copies forming a clique.

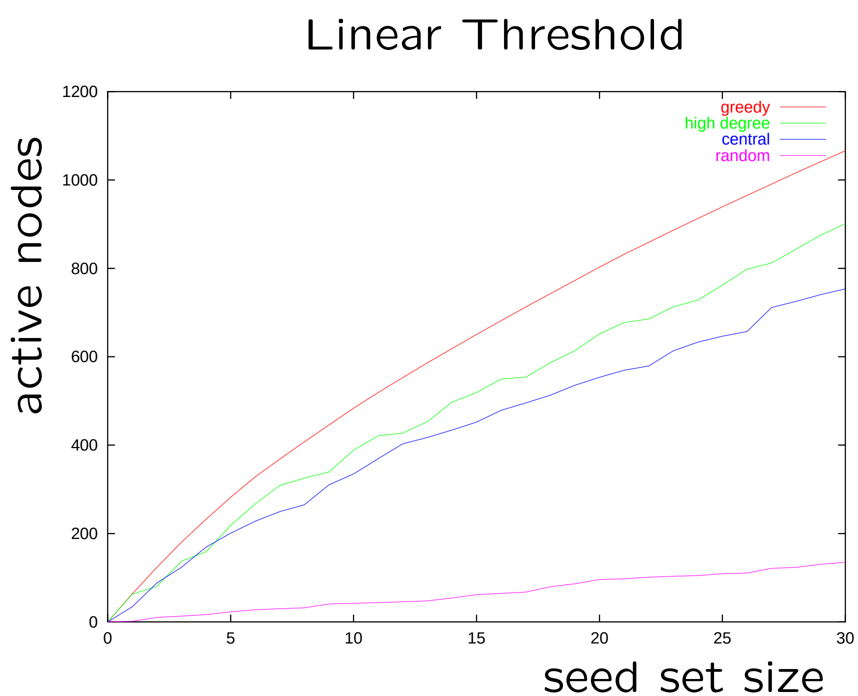
Weights

- Nodes may have different values v_i (e.g. order sizes).
- Function f is still submodular, so all of the proof stays the same.
- Alternatively, can replace each node by v_i copies forming a clique.
- Evaluating f may take time pseudo-polynomial in the v_i. (Example: one very important node, reached with very small probability.)
- Can we evaluate f in strongly polynomial time?

Experiments

- Used arXiv high-energy physics collaboration graph.
- Compared greedy algorithm, degree centrality heuristic, distance centrality heuristic, random nodes.





Conclusions

- Word-of-mouth effect play a crucial role in collective behavior, marketing. Use them!
- Studied sociology models.
- Obtained provable approximation guarantees and good behavior in practice for simple algorithm.

Open Questions:

- Study more general influence models. Find trade-offs between generality and feasibility.
- Deal with negative influences.
- Model competing products.
- Obtain more data about how activations occur in real social networks.