Exact Computation of Influence Spread by Binary Decision Diagrams

Takanori Maehara[1], Hirofumi Suzuki[2], Masakazu Ishihata[2]

1) Riken Center for Advanced Intelligence Project
2) Hokkaido University

August 4, 2017 @ ERATO 感謝祭

Introduction

Viral Marketing [Richardson–Domingos'02]

Viral Marketing

Marketing strategy that uses social networks to promote a product, i.e.,

Giving products for some individuals (seeds) to propagate information over the social network by *word-of-mouth effect*

Objective

Estimate (and Maximize) the *size of influence spread*

Mathematical Model (Independent Cascade Model)

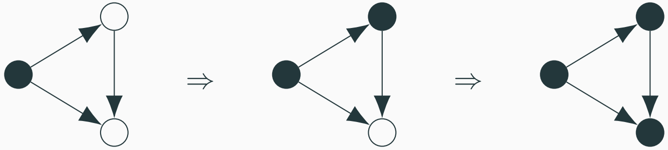
A social network is modeled as a directed graph $G = (V, E; p)$
(V: users, E: communications, $p(e)$: activation probability)

Independent Cascade Model [Goldenberg+'01]

Information propagates from $S \subseteq V$ (seeds):

1. At the first step, the seeds $s \in S$ become *active*
2. When $u \in V$ becomes *active*, its neighbor $v \in N(u)$ becomes *active* with probability $p(u, v)$

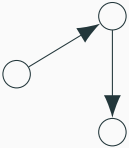
Influence spread $\sigma(S)$ = expected number of activated users u



In the IC model, each edge is examined once. Therefore it is equivalent to the following "pre coin-flipping" model:

Independent Cascade Model (Coin-Flipping)

$\sigma(S)$ is the expected number of reachable vertices from S in a random graph where $e \in E$ appears with probability $p(e)$



Influence Maximization Problem [Kempe+'03]

maximize $\sigma(S)$ subject to $|S| \leq k$

- (Good News) $\sigma(S)$ is a *monotone submodular function*
  $\Rightarrow$ The greedy algorithm attains $1 - 1/e$ approximation by polynomial number of $\sigma(\cdot)$ evaluations
- (Bad News) Evaluation of $\sigma(\cdot)$ is #P-Hard

Existing Work: Monte-Carlo simulation (e.g., [Borgs+'14])

- Simulate the model $\Omega(1/\epsilon^2)$ times to obtain $(1 \pm \epsilon)$ approximation of $\sigma(S)$
- Suitable for *rough estimation* of $\sigma(S)$ in large networks

This Study: Exact Computation of Influence Spread

Our Goal

Compute $\sigma(S)$ exactly on small networks (e.g., $m \approx 100$)
(Naive method cannot be used because of 2^{100} configurations)

Why exact computation?

- Precise analysis on small networks is important because social networks consist from many small communities
- Exact method can be used to evaluate the practical accuracy of Monte-Carlo simulations
- Practical algorithm for #P-Hard problem is itself interesting topic in Computer Science

Our Contributions

- We proposed the first (non-trivial) algorithm for computing $\sigma(S)$ exactly
- It runs on $m \approx 100$ networks in a reasonable time
- It can be used to solve another influence-spread related problems (e.g., sampling, conditional expectation, ...)

Our Approach

- Construct the **binary decision diagrams (BDDs)** for S-t connected subgraphs
- Perform **dynamic programming** to compute the influence spread

Proposed Algorithm (Outline)

Reformulation of Influence Spread

Let us define

- $p(F) = \prod_{e \in F} p(e) \prod_{e' \notin F} (1 - p(e'))$
- $\mathcal{R}(S, t) := \{F \subseteq E : G[F] \text{ has a } S\text{-}t \text{ path}\}$

Then the influence spread is given by

$$\sigma(S) = \sum_{t \in V} \sigma(S, t)$$

where

$$\sigma(S, t) = \sum_{F \in \mathcal{R}(S,t)} p(F) =: p(\mathcal{R}(S, t))$$

We compute $\sigma(S)$ by using the above formulas
… An efficient algorithm to enumerate $\mathcal{R}(S, t)$ is required

Binary Decision Diagram (BDD)

We maintain $\mathcal{R}(S, t)$ by a *binary decision diagram* (BDD)

BDD

A compact representation of a Boolean function

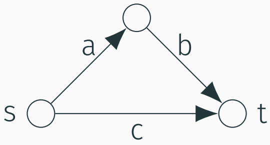
- A DAG with single root and two terminals (0 and 1)
- Each node has two children (0-child and 1-child)
- Each node is associated with a variable $e \in E$

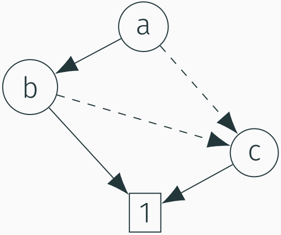
A path from the root to 1-child = a term of the function
(descend 0-arc/1-arc $\iff$ exclude/include e)

A Boolean function represents a set family as

$$\mathcal{R}(S, t) = \{F \in 2^E : r(F) = \texttt{True}\}$$

Example





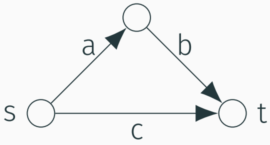
Left: Social Network   Right Associated BDD
(dotted/solid = 0-/1-arc, 0-terminal is omitted)

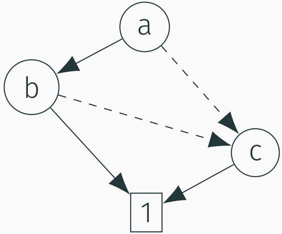
$\mathcal{R}(s,t) = \{\{c\},\{a,b\},\{a,c\},\{b,c\},\{a,b,c\}\}$,
Boolean function $= c + ab + ac + bc + abc = a(b + \bar{b}c) + \bar{a}c$

Influence Computation via BDD

Once the BDD of $\mathcal{R}(S,t)$ is obtained, $\sigma(S,t)$ is efficiently computed by the bottom-up dynamic programming:

$$\mathcal{B}(0) = 0,\ \mathcal{B}(1) = 1,\ \mathcal{B}(\alpha) = (1 - p(e))\mathcal{B}(\alpha_0) + p(e)\mathcal{B}(\alpha_1)$$





$\mathcal{B}(1) = 1, \mathcal{B}(c) = p, \mathcal{B}(b) = p + (1-p)p,$
$\mathcal{B}(a) = p(p + (1-p)p) + (1-p)p = p + p^2 - p^3$

Construction of BDD: Two stage method

Base Case: BDD of $\mathcal{R}(s,t)$ ($s,t \in V$)

Constructed by new *frontier-based search* procedure

General Case: BDD of $\mathcal{R}(S,t)$ ($S \subseteq V$, $t \in V$)

Constructed by the union of base-case BDDs:

$$\mathcal{R}(S,t) = \bigcup_{s \in S} \mathcal{R}(s,t)$$

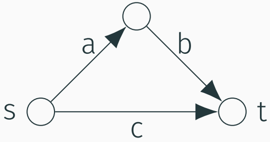
If the set families are represented by BDDs,
these union/intersection/... are efficiently obtained

Note: The 1st step in the greedy algorithm computes all $\mathcal{R}(s,t)$
Therefore this method is always faster than the naive method

Frontier-Based Search
— Main Procedure —

Exhaustive enumeration then node merging

$\mathcal{N}_0 = \emptyset$
for $k = 1, 2, \ldots, m$ **do**
    Generate $\mathcal{N}_k$ from $\mathcal{N}_{k-1}$ by including/excluding e_k
end for



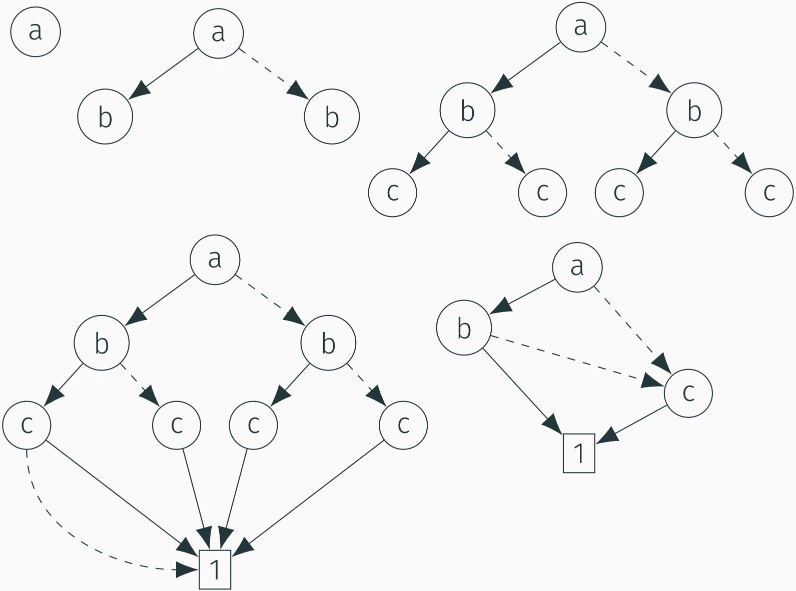
$\mathcal{N}_1 = \{\{a\}, \emptyset\}$

$\mathcal{N}_2 = \{\{a, b\}, \{a\}, \{b\}, \emptyset\}$

$\mathcal{N}_3 = \{\{a, b, c\}, \{a, b\}, \{a, c\}, \{a\}, \{b, c\}, \{b\}, \{c\}, \emptyset\}$

...

Exhaustive enumeration then node merging



Issue in exhaustive enumeration

Issue

It enumerates 2^m nodes (too expensive)

We do not want to enumerate the nodes that

- … must reach to the terminals (1 or 0)
- … are "equivalent" to some other node

Frontier-Based search solves these issues!

Frontier-Based Search

Exhaustive enumeration with *node sharing*

$\mathcal{N}_0 = \emptyset$
for $k = 1, 2, \ldots, m$ **do**
    Generate $\mathcal{N}_k$ from $\mathcal{N}_{k-1}$ by including/excluding e_k
    Terminality check
    Merge the equivalent nodes if possible
end for

Each node $\alpha \in \mathcal{N}_k$ represents a family of *equivalent* sets $R(\alpha)$

- $F, F' \subseteq \{e_1, \ldots, e_k\}$ are equivalent iff $\forall H \subseteq \{e_{k+1}, \ldots, e_m\}$, reachability of $G[F \cup H]$ and $G[F' \cup H]$ are the same
- α, α' are equivalent iff for all elements represented by these nodes are equivalent

Configuration (aka. Mate Array)

A *configuration* ϕ is used to check the equivalence:

$$\phi(\alpha) = \phi(\alpha') \Rightarrow \alpha \equiv \alpha'$$

Our configuration for s-t connected subgraphs

Let $W \subseteq V$ be the *frontier vertices*, which is the set of vertices incident to both processed and unprocessed edges

Then we define

$$\begin{aligned}\phi(\alpha) = {}& \textit{reachability matrix} \text{ from } (W \cup \{s\}) \text{ to } (W \cup \{t\}) \\ & \text{on } G[F] \text{ for some } F \in R(\alpha)\end{aligned}$$

- $\phi(\alpha) = \phi(\alpha')$ then the feature reachabilities are the same. This also proves the well-definedness
- The size is $O(|W|^2)$. Therefore the number nodes having distinct configurations is $2^{O(|W|^2)}$

Terminality Check

- The included edges contains s-t path $\Rightarrow$ Merge α to 1
  … Efficiently checked by the configuration: $O(1)$
- The excluded edges forms s-t cut $\Rightarrow$ Merge α to 0
  … Checked by the BFS: $O(m^2)$ preproc. + $O(|W|^2)$ time

"Performing BFS in the frontier-based search" is expensive, but necessarily to deal with larger networks

Frontier-Based Search
— Additional Techniques —

Pruning Redundant Edges

We remove all edges that do not contribute s-t connectivity

- Checking existence of s-t path containing $e \in E$ is NP-hard (equivalent with the two commodity flow)
- However, since the network is small, we can solve this problem by the enumeration by frontier-based search! (Knuth's SimPath algorithm)

"Using frontier-based search as a preprocessing for the frontier-based search" seems a technically interesting idea

Edge Ordering

The complexity of the frontier-based search depends on the edge ordering ($\because W \Rightarrow$ fast enumeration)

The same ordering should be used for all $\mathcal{R}(s,t)$ because we perform set family manipulations

$\Rightarrow$ We used path decomposition + beam-search heuristics proposed by [Inoue and Minato'16]

Node Sharing among Different BDDs

The algorithm construct $O(n^2)$ BDDs $\mathcal{R}(s, t)$ for all $s, t \in V$, which may have many similar sub-structures

$\Rightarrow$ Share them to reduce the total number of nodes [Minato'90] (reduced about factor 2 in practice)

Other Influence-Spread Related Problems

Other Influence-Spread Related Problems

Our algorithm constructs the BDDs for influence spread
$\Rightarrow$ many other problems can be solved using the BDDs

1. Random Sampling without Rejection
2. Conditional Influence Spread
3. Dynamic Probability Update
4. Optimization (Computing $\partial\sigma(S)/\partial p(e)$)

Details are omitted (perhaps well-known in BDD community)
These are important in viral marketing

Experiments

Experimental Setting

- C++ (g++5.4.0 with -O3 option)
- BDD implementation: TdZdd library
  https://github.com/kunisura/TdZdd
- 64-bit Ubuntu 16.04 LTS with an Intel Core i7-3930K 3.2 GHz CPU and 64 GB RAM
- Real Networks: Koblenz Network Collection
  http://konect.uni-koblenz.de/

Basic Results on Typical Networks

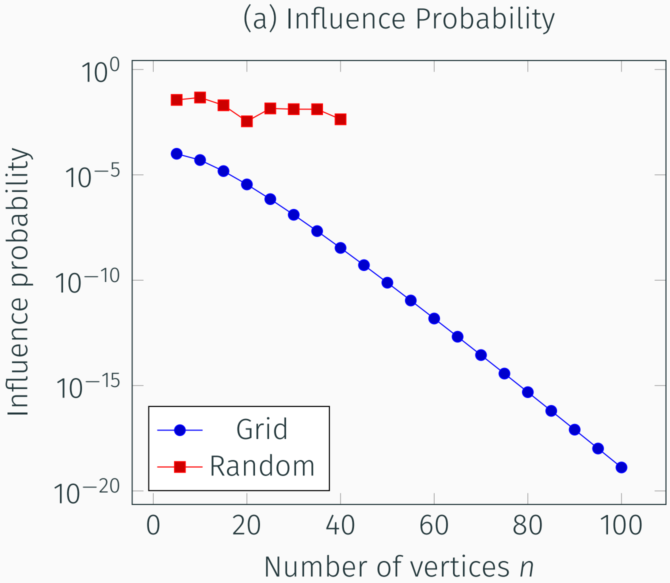
Network	n	m	Time [ms]	BDD Size	Shared Size	Cardinality
South-African-Companies	11	26	0.1	12.1	472	2.2e+07
Southern-women-2	20	28	0.3	54.7	2,266	1.3e+08
Taro-exchange	22	78	4.1	1,119.2	277,756	1.6e+23
Zachary-karate-club	34	156	24.9	7,321.8	4,988,148	6.4e+46
Contiguous-USA	49	214	117.9	30,599.8	41,261,047	1.6e+64
American-Revolution	141	320	2.2	120.0	1,530,677	5.7e+95
Southern-women-1	50	178	—	—	—	—
Club-membership	65	190	—	—	—	—
Corporate-Leadership	64	198	—	—	—	—

- Fast in solvable instances
- Very slow in unsolvable instances

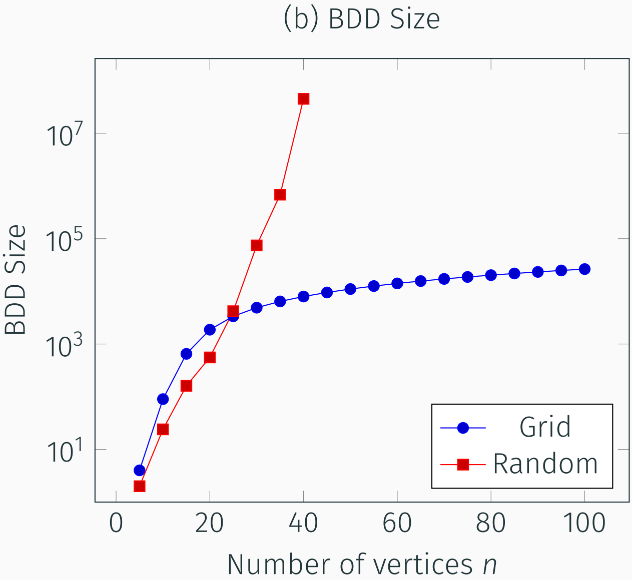
Experiments on Grid/Random Network

- Construct the following networks
  1. $n \times 5$ grid network
  2. random network with the same n, m
- Compute the exact influence spread from the corner

Results on Grid/Random Network (1/3)

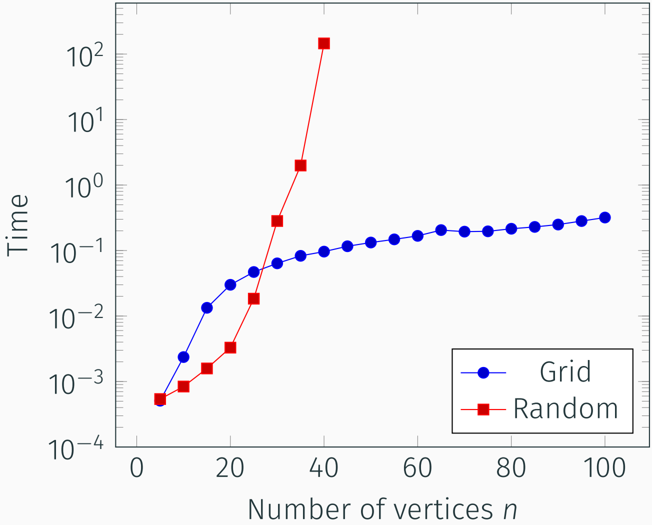


Results on Grid/Random Network (2/3)



Results on Grid/Random Network (3/3)

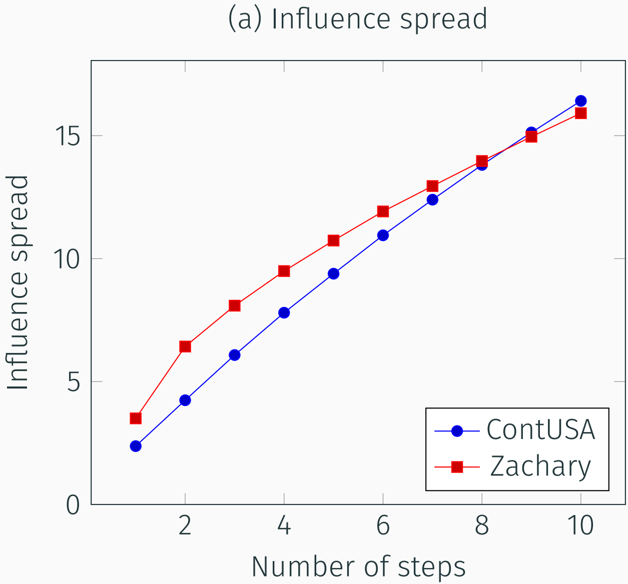
(c) Time [s]



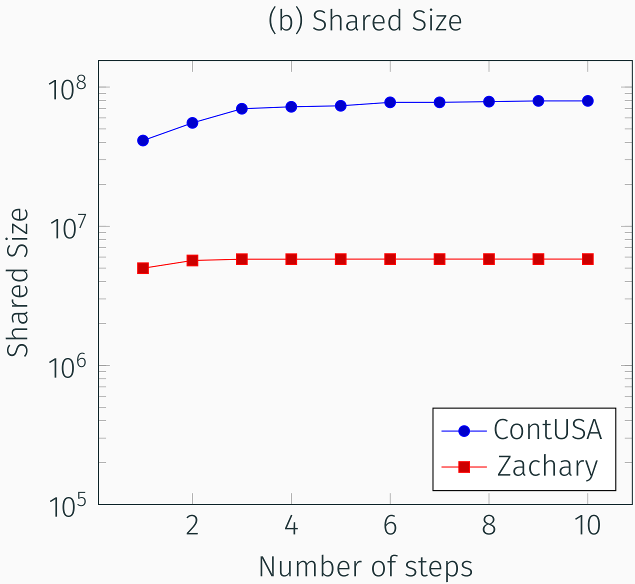
Example 3: Greedy Algorithm

- We used the Contiguous USA Network
- We performed the greedy algorithm to seek the most influencial seeds of size $k = 10$.

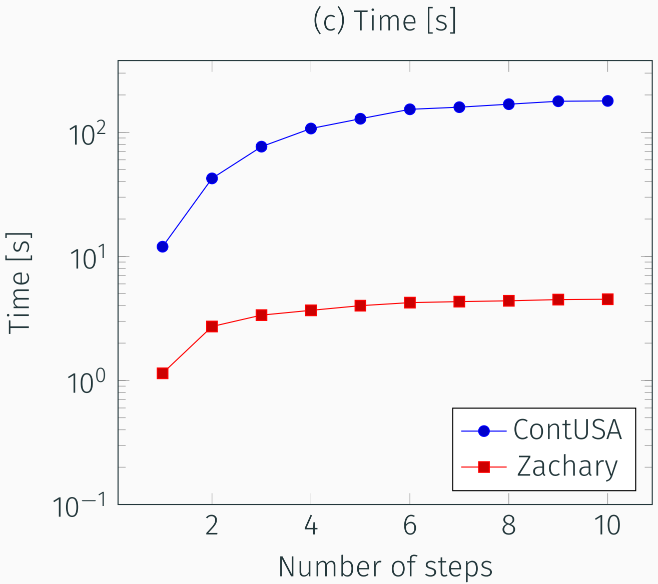
Results on Greedy Algorithm (1/3)



Results on Greedy Algorithm (2/3)



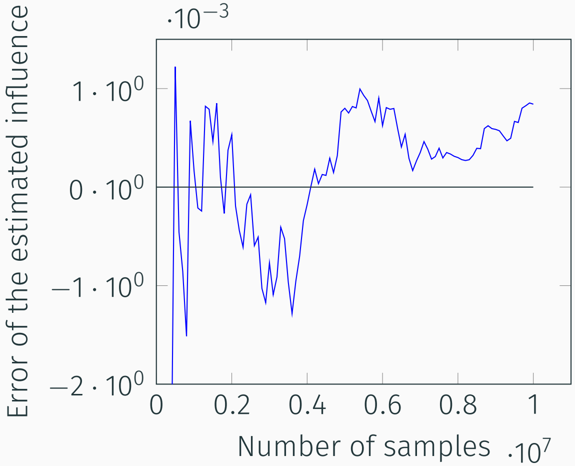
Results on Greedy Algorithm (3/3)



Experiment 4: Exact vs Monte-Carlo

- We used the Contiguous USA Network
- Fix a set and compute influence spread by exact method and Monte-Carlo method
- Compare the results for various number of MC samples

Results for Exact vs Monte-Carlo



- Fluctuates in 10^{-3}, which is consistent with the theory

Related Work

Related Work

- This is the first attempt for exact influence spread computation
- No existing method for enumerating *s*-*t* connected subgraphs efficiently
  (An algorithms for undirected case is known in the literature of “Network Reliability” (e.g., [Valiant’79]), but it cannot be extend to our directed case)

Conclusion

Conclusion / Future Work

Conclusion

1. We proposed the **first algorithm to compute the influence spread exactly**
2. It constructs the BDDs of S-u connected subgraphs by **new frontier-based search** and **set family manipulations**
3. **Real networks with** $m \approx 100$ **were solved**

Future Work

1. **We want to solve** $m \approx 200$ **networks**
2. **Exact influence maximization**
   NP-hard problem with #P-hard function evaluation

This slide is available at:
`http://www.prefield.com/slide/maehara2017exact_slide.pdf`

Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier.
Maximizing social influence in nearly optimal time.
In *SODA*, pages 946–957, 2014.

Jacob Goldenberg, Barak Libai, and Eitan Muller.
Talk of the network: A complex systems look at the underlying process of word-of-mouth.
Marketing Letters, 12(3):211–223, 2001.

Yuma Inoue and Shinichi Minato.
Acceleration of ZDD construction for subgraph enumeration via path-width optimization.
TCS-TR-A-16-80. Hokkaido University, 2016.

David Kempe, Jon Kleinberg, and Éva Tardos.

Maximizing the spread of influence through a social network.
In *KDD*, pages 137–146, 2003.

Shinichi Minato, Nagisa Ishiura, and Shuzo Yajima.
Shared binary decision diagram with attributed edges for efficient boolean function manipulation.
In *DAC*, pages 52–57, 1990.

Matthew Richardson and Pedro Domingos.
Mining knowledge-sharing sites for viral marketing.
In *KDD*, pages 61–70, 2002.

Leslie G Valiant.
The complexity of enumeration and reliability problems.
SIAM Journal on Computing, 8(3):410–421, 1979.