An FPRAS for two-terminal reliability in directed acyclic graphs

Weiming Feng (ETH Zürich)

Joint work with Heng Guo (University of Edinburgh)

Summers School on Algorithms, Dynamics, and Information Flow in Networks
5th July 2024
Möhnesee Germany

All-terminal network reliability

All-terminal network reliability

Input: an undirected graph $G = (V, E)$ and parameters $q_e \in (0,1)$

Output: the *probability* that G is ***connected*** if each edge $e \in E$ fails independently with prob. q_e

- #P-complete for exact computing [Valiant 79; Colbourn 87]
- FPRAS (fully poly-time approx. algorithm) exists [Guo and Jerrum18]

All-terminal network unreliability

Input: an undirected graph $G = (V, E)$ and parameters $q_e \in (0,1)$

Output: the *probability* that G is ***not connected*** if each edge $e \in E$ fails independently with prob. q_e

- #P-complete for exact computing [Valiant 79; Colbourn 87]
- FPRAS (fully poly-time approx. algorithm) exists [Karger 99; Cen, He, Li and Panigrahi 23]

s-t network reliability

Two-terminal network reliability

Input: a graph $G = (V, E)$ and parameters $q_e \in (0,1)$

a source node s and a sink node t

Output: the *probability* that $s \rightarrow_{G(p)} t$ if each edge $e \in E$ fails independently with prob. q_e

s can reach t in the remaining graph

$$\Pr[s \rightarrow_{G(p)} t] = \sum_{\substack{R \subseteq E: \\ s \to t \text{ in graph } (V,R)}} \prod_{e \in R} (1 - q_e) \prod_{e \notin R} q_e$$

Exact computing is #P-complete for ***directed*** / ***undirected*** / ***DAG*** / ***planar DAG*** graphs

Approximate s-t network reliability in DAGs

Two-terminal network reliability approximation

Input: a ***DAG (direct acyclic graph)*** $G = (V, E)$ and parameters $q_e \in (0,1)$

a source node s and a sink node t

an error bound $\varepsilon > 0$

Output: a ***random*** number $\hat{p}$ approximating s-t network reliability $p = \Pr\left[s \to_{G(p)} t\right]$

$$\Pr[\hat{p} \in (1 \pm \varepsilon)p] \geq \frac{2}{3}$$

Two-terminal network unreliability approximation

Output: a ***random*** number $\hat{q}$ approximating s-t network unreliability $1 - p$

$$\Pr[\hat{q} \in (1 \pm \varepsilon)(1 - p)] \geq \frac{2}{3}$$

Previous works

Direct Monte Carlo method (sample random subgraph and check reachability)

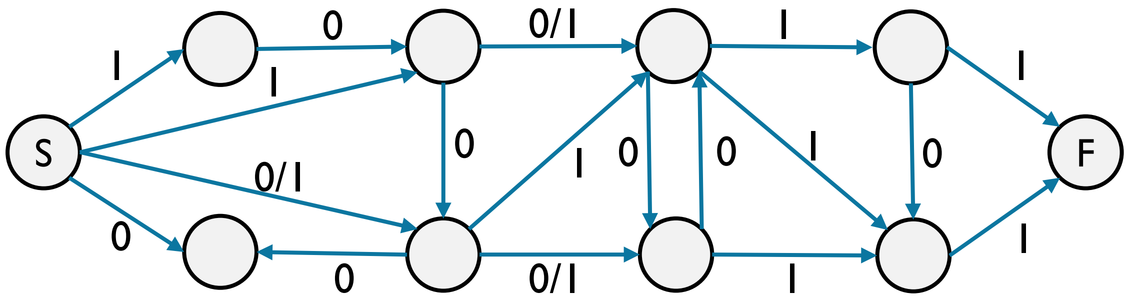
- the estimation is efficient if $p = \frac{1}{\mathrm{poly}(m)}$ (e.g. $q_e = O\left(\frac{\log m}{m}\right)$ is small, where $m = |E|$)

Improved analysis on lower bound of s-t reliability [Zenklusen and Laumanns 10]

- Improve the lower bound of reliability for some special DAGs

Approximate count accepting strings of **NFA** (nondeterministic finite automaton)

- Given n-state NFA, count #{distinct accepting strings} of length ℓ



Accepting String: 100011

Previous works

Direct Monte Carlo method (sample random subgraph and check reachability)

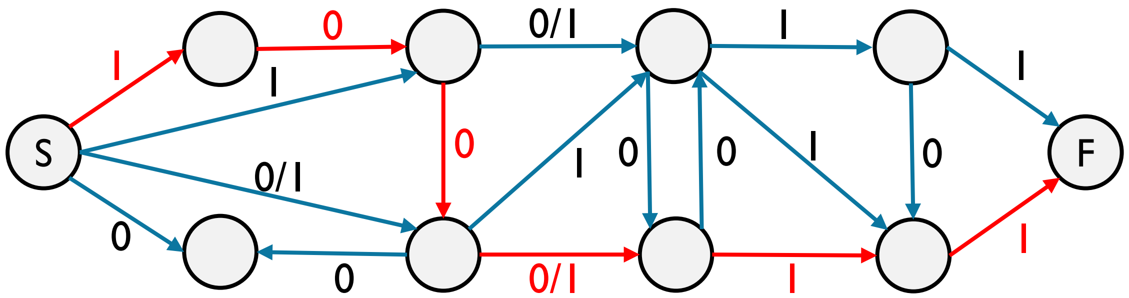
- the estimation is efficient if $p = \frac{1}{\mathrm{poly}(m)}$ (e.g. $q_e = O\left(\frac{\log m}{m}\right)$ is small, where $m = |E|$)

Improved analysis on lower bound of s-t reliability [Zenklusen and Laumanns 10]

- Improve the lower bound of reliability for some special DAGs

Approximate count accepting strings of **NFA** (nondeterministic finite automaton)

- Given n-state NFA, count #{distinct accepting strings} of length ℓ



Accepting String: 100011

Previous works

Direct Monte Carlo method (sample random subgraph and check reachability)

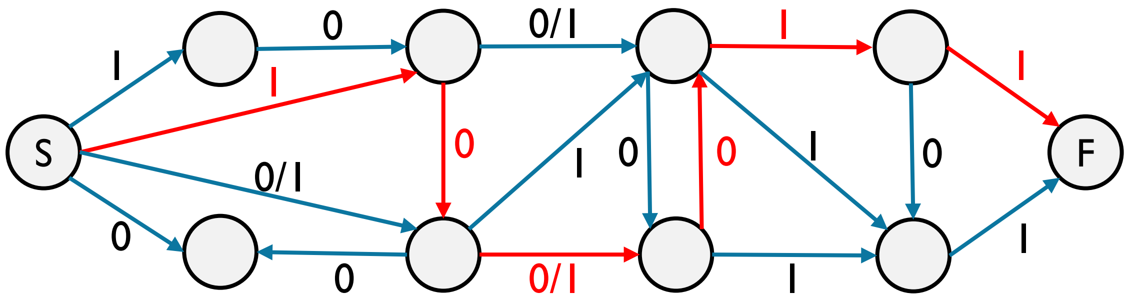
- the estimation is efficient if $p = \frac{1}{\mathrm{poly}(m)}$ (e.g. $q_e = O\left(\frac{\log m}{m}\right)$ is small, where $m = |E|$)

Improved analysis on lower bound of s-t reliability [Zenklusen and Laumanns 10]

- Improve the lower bound of reliability for some special DAGs

Approximate count accepting strings of **NFA** (nondeterministic finite automaton)

- Given n-state NFA, count #{distinct accepting strings} of length ℓ
- FPRAS in time $\tilde{O}((n\ell)^{17})$ [Arenas, Croquevielle, Jayaram, Riveros, 21]



Accepting String: 100011

Previous works

Direct Monte Carlo method (sample random subgraph and check reachability)

- the estimation is efficient if $p = \frac{1}{\mathrm{poly}(m)}$ (e.g. $q_e = O\left(\frac{\log m}{m}\right)$ is small, where $m = |E|$)

Improved analysis on Lower bound of s-t reliability [Zenklusen and Laumanns 10]

- Improve the lower bound of reliability for some special DAGs

Approximate count accepting strings of **NFA** (nondeterministic finite automaton)

- Given n-state NFA, count #{distinct accepting strings} of length ℓ
- FPRAS in time $\tilde{O}((n\ell)^{17})$ [Arenas, Croquevielle, Jayaram, Riveros, 21]
- s-t reliability in DAGs can be reduced to #NFA [implied by Burtschick 95] [explicitly in Amarilli, Bremen and Meel 24]

  ➡ An FPRAS for s-t reliability in DAG (running time is a huge polynomial)

Very recently, improved FPRAS for #NFA [Meel, Chakraborty and Mathur 24]

- An FPRAS for s-t reliability in time $\tilde{O}(m^{19})$ for $q_e = \frac{1}{2}$ via **standard black-box reduction**

Our results

There is an ***FPRAS*** for two-terminal network ***reliability*** in DAGs in time

$$\tilde{O}(n^6 m^4 \max\{m^4, \varepsilon^{-4}\}) \text{ , where } n = |V| \text{ and } m = |E|$$

- technique inspired by [Arenas, Croquevielle, Jayaram and Riveros 21]
- running time can be further reduced by combining our technique with very recent technique for #NFA [Meel, Chakraborty and Mathur 24]

There is ***no FPRAS*** for two-terminal network ***unreliability*** in DAGs unless there is an FPRAS for ***#BIS*** problem

#BIS: counting the number of independent sets in bipartite graphs conjectured to has no FPRAS

Some basic settings for this talk

Two-terminal network reliability approximation

Input: a ***DAG (direct acyclic graph)*** $G = (V, E)$ and parameters $q_e \in (0,1)$

a source node s and a sink node t

an error bound $\varepsilon > 0$

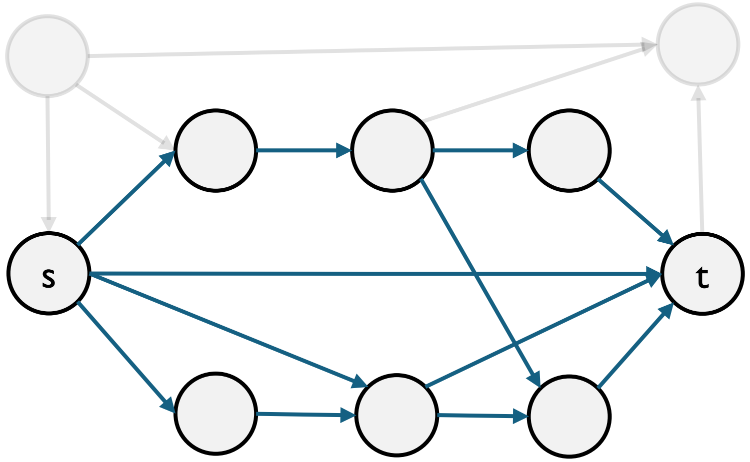
Output: a ***random*** number $\hat{p}$ approximating s-t network reliability $p = \Pr[s \to_{G(p)} t]$

1) For any vertex v in DAG $G = (V, E)$ assume that
   v can be reached from s and v can reach t

2) For any edge $e \in E, q_e = \frac{1}{2}$
   general $q_e \in (0,1)$ can be solved with ***very small tweaks***

$$p = \frac{\#\{\text{subgraphs such that } s \text{ can reach } t\}}{2^m}$$

counting problem

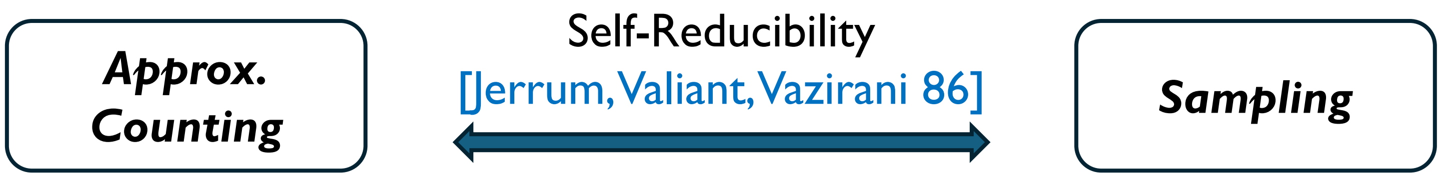


Sampling and counting

Two-terminal network reliability in DAGs

$$\Omega = \{\text{subgraphs of } G \text{ such that } s \text{ can reach } t\}$$

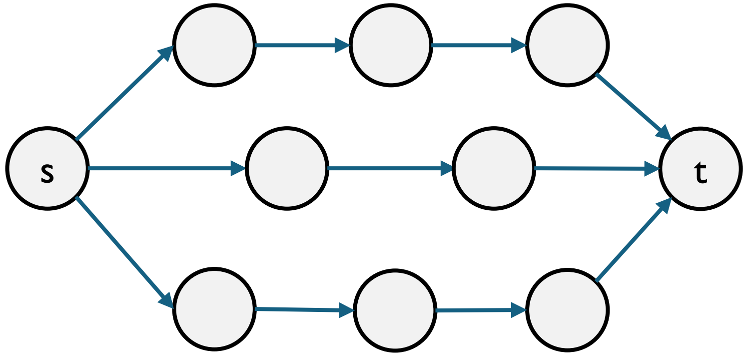
- ***Counting Problem (network reliability)***: estimate the size $\#\Omega$
- ***Sampling Problem***: draw random subgraphs from Ω uniformly at random



Sampling algorithm: MCMC (Metropolis chain)

Maintain a subset of edges $R \in \Omega$ (s can reach t in subgraph (V, R))

- Pick an edge $e \in E$ uniform at random
- If $e \in R$ and $R - e \in \Omega$, let $R \leftarrow R - e$
- If $e \notin R$, let $R \leftarrow R + e$



Sampling and counting

Two-terminal network reliability in DAGs

$$\Omega = \{\text{subgraphs of } G \text{ such that } s \text{ can reach } t\}$$

- ***Counting Problem (network reliability)***: estimate the size $\#\Omega$
- ***Sampling Problem***: draw random subgraphs from Ω uniformly at random

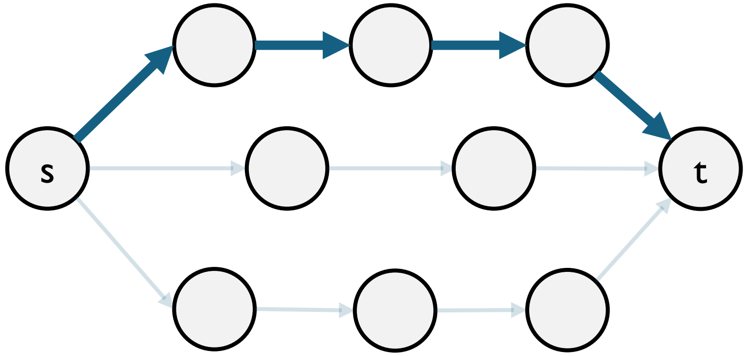
Approx. Counting $\longleftrightarrow$ ***Sampling***

Self-Reducibility
[Jerrum, Valiant, Vazirani 86]

Sampling algorithm: MCMC (Metropolis chain)

Maintain a subset of edges $R \in \Omega$ (s can reach t in subgraph (V, R))

- Pick an edge $e \in E$ uniform at random
- If $e \in R$ and $R - e \in \Omega$, let $R \leftarrow R - e$
- If $e \notin R$, let $R \leftarrow R + e$



Sampling and counting

Two-terminal network reliability in DAGs

$$\Omega = \{\text{subgraphs of } G \text{ such that } s \text{ can reach } t\}$$

- ***Counting Problem (network reliability)***: estimate the size $\#\Omega$
- ***Sampling Problem***: draw random subgraphs from Ω uniformly at random

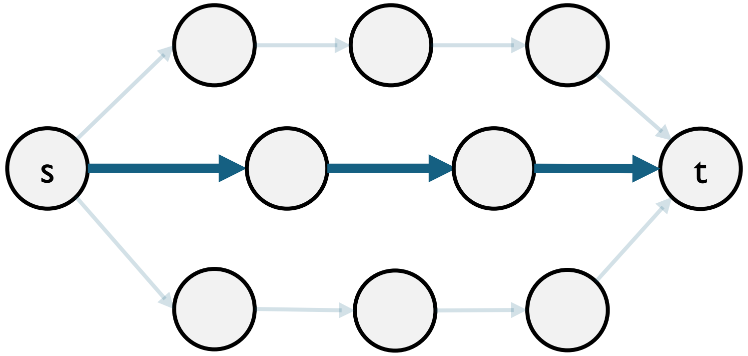
Approx. Counting ⟷ ***Sampling***

Self-Reducibility
[Jerrum, Valiant, Vazirani 86]

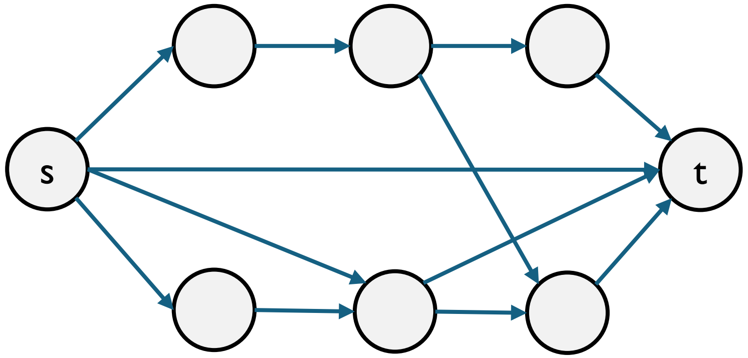
Sampling algorithm: MCMC (Metropolis chain)

Maintain a subset of edges $R \in \Omega$ (s can reach t in subgraph (V, R))

- Pick an edge $e \in E$ uniform at random
- If $e \in R$ and $R - e \in \Omega$, let $R \leftarrow R - e$
- If $e \notin R$, let $R \leftarrow R + e$

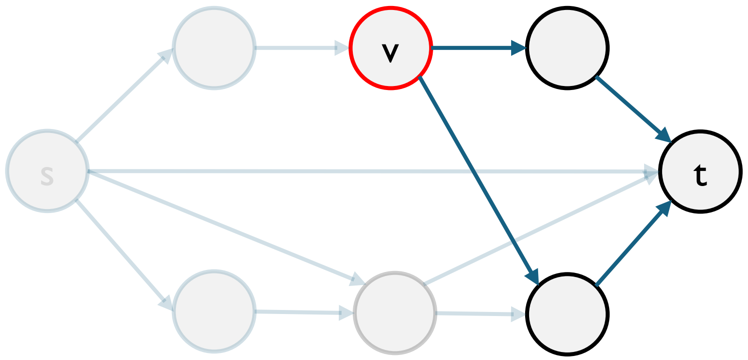


Our Algorithm



sort all vertices according to topological order $s = v_n \succ v_{n-1} \succ \cdots \succ v_1 = t$

Our Algorithm



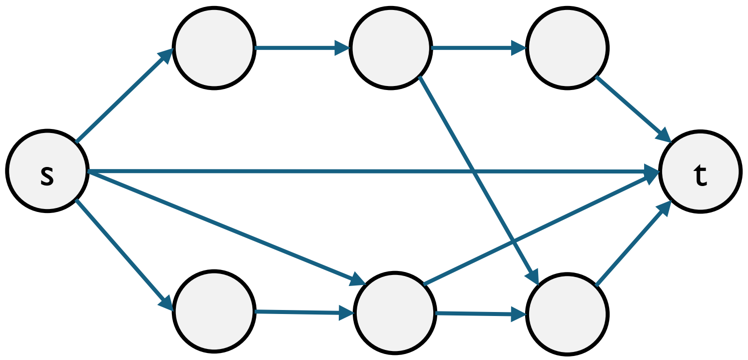
sort all vertices according to topological order $s = v_n \succ v_{n-1} \succ \cdots \succ v_1 = t$

- G_v: subgraph containing all vertices that can be reached from $v \in V$

$$\Omega_v = \{\text{subgraphs of } G_v \text{ such that } v \text{ can reach } t\}$$

- Z_v: a **counting estimate** of the size $\#\Omega_v$
- S_v: a set of **samples**, where each $H \in S_v$ is ***uniform random sample*** from Ω_v

Our Algorithm



sort all vertices according to topological order $s = v_n \succ v_{n-1} \succ \cdots \succ v_1 = t$

- G_v: subgraph containing all vertices that can be reached from $v \in V$

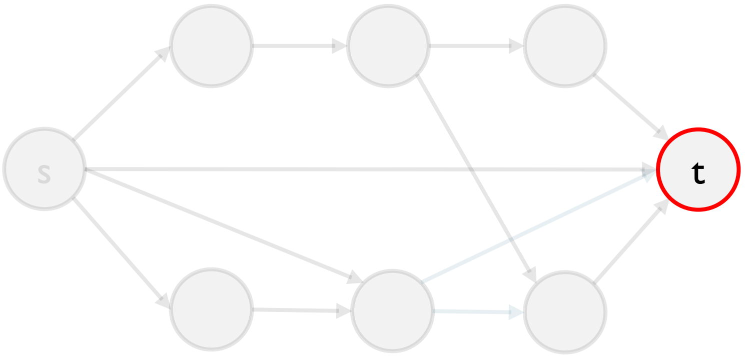
$$\Omega_v = \{\text{subgraphs of } G_v \text{ such that } v \text{ can reach } t\}$$

- Z_v: a **counting estimate** of the size $\#\Omega_v$
- S_v: a set of **samples**, where each $H \in S_v$ is ***uniform random sample*** from Ω_v

Compute (Z_v, S_v) for v from v_1 to v_n via ***dynamic programming*** + ***Monte Carlo***

Framework appeared in [Gore, Jerrum, Kannan, Sweedyk, and Mahaney 97] [Arenas, Croquevielle, Jayaram and Riveros 21]

Our Algorithm



base case $v = t$ is trivial

sort all vertices according to topological order $s = v_n \succ v_{n-1} \succ \cdots \succ v_1 = t$

- G_v: subgraph containing all vertices that can be reached from $v \in V$

$$\Omega_v = \{\text{subgraphs of } G_v \text{ such that } v \text{ can reach } t\}$$

- Z_v: a **counting estimate** of the size $\#\Omega_v$
- S_v: a set of **samples**, where each $H \in S_v$ is ***uniform random sample*** from Ω_v

Compute (Z_v, S_v) for v from v_1 to v_n via ***dynamic programming*** + ***Monte Carlo***

Framework appeared in [Gore, Jerrum, Kannan, Sweedyk, and Mahaney 97] [Arenas, Croquevielle, Jayaram and Riveros 21]

Estimate the count via Karp-Luby

- A vertex $v \in V$ and a set of ***neighbors*** $v_1, v_2, v_3, \dots, v_d$

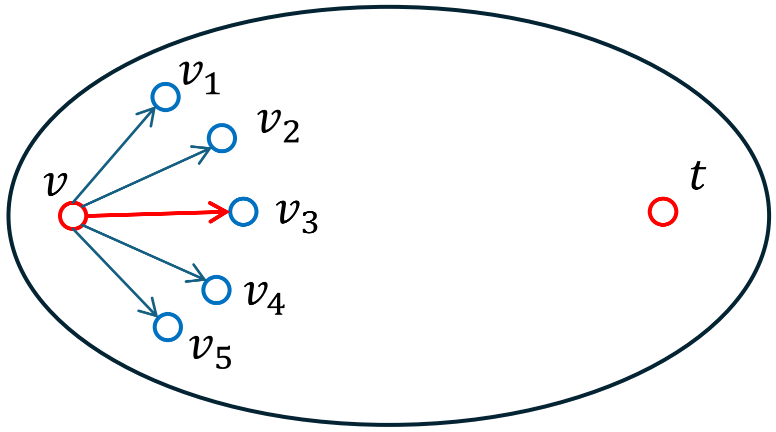
$$\Omega_v = \{\text{subgraphs of } G_v \text{ s.t. } v \to t\}$$

- For any neighbor v_i,

$$\Omega_i = \{\text{subgaphs of } G_v \text{ s.t. } v \to v_i \to t\}$$

- Vertex v must reach t through some neighbors

$$\Omega_v = \cup_{i=1}^{d} \Omega_i$$



Estimate the count via Karp-Luby

- A vertex $v \in V$ and a set of ***neighbors*** $v_1, v_2, v_3, \dots, v_d$

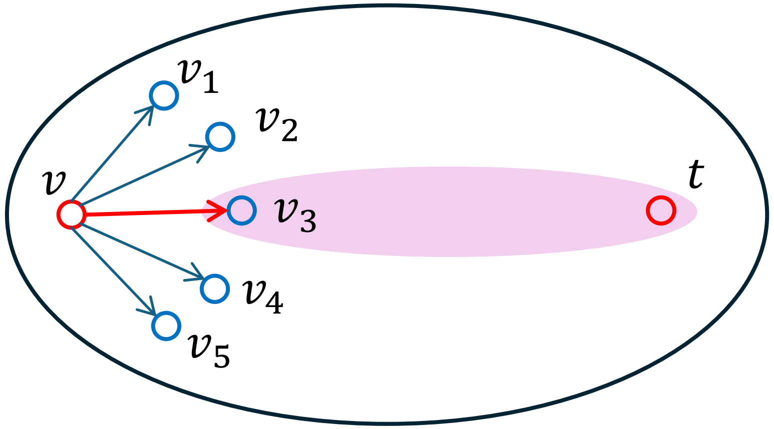
$$\Omega_v = \{\text{subgraphs of } G_v \text{ s.t. } v \to t\}$$

- For any neighbor v_i,

$$\Omega_i = \{\text{subgaphs of } G_v \text{ s.t. } v \to v_i \to t\}$$

- Vertex v must reach t through some neighbors

$$\Omega_v = \cup_{i=1}^{d} \Omega_i$$



Karp-Luby's method for estimating the size of union [Karp and Luby 83]

- Sample an index $i \in [d]$ with prob $\propto |\Omega_i|$ ➡ estimate $|\Omega_i|$ using $Z_{v_i} \approx |\Omega_{v_i}|$
- Sample a subgraph H from Ω_i uniformly at random ➡ take a sample $H' \in S_{v_i}$ and modify $H' \to H$
- $X \in \{0,1\}$ indicate whether i is the first set containing H ➡ do DFS search in H

$$\mathbb{E}[X] = \frac{|\Omega_v|}{\sum_{i=1}^{d} |\Omega_i|} \geq \frac{1}{d} \qquad |\Omega_v| = \left(\sum_{i=1}^{d} |\Omega_i|\right) \mathbb{E}[X]$$

generate ind. X and take average

Estimate the count via Karp-Luby

- A vertex $v \in V$ and a set of ***neighbors*** $v_1, v_2, v_3, \dots, v_d$

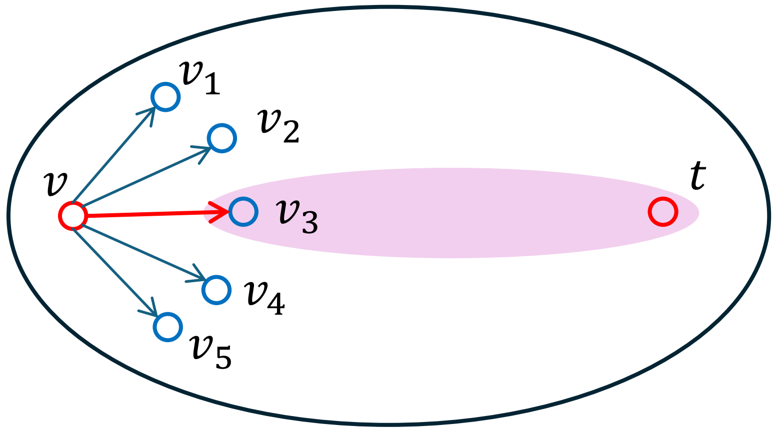
$$\Omega_v = \{\text{subgraphs of } G_v \text{ s.t. } v \to t\}$$

- For any neighbor v_i,

$$\Omega_i = \{\text{subgaphs of } G_v \text{ s.t. } v \to v_i \to t\}$$

- Vertex v must reach t through some neighbors

$$\Omega_v = \cup_{i=1}^{d} \Omega_i$$



- Apply ***Karp-Luby*** to compute an estimate Z_v such that

$$\Pr\left[Z_v \in \left(1 \pm \mathrm{poly}\left(\frac{\varepsilon}{m}\right)\right)|\Omega_v|\right] \geq \frac{2}{3}$$

- Apply ***median-trick*** to boost the successful prob.

$$2/3 \to 1 - \exp(-\mathrm{poly}(m))$$

- ***Higher accuracy*** and ***higher successful probability*** require ***more samples*** in set S_{v_i}

Estimate the count via Karp-Luby

- A vertex $v \in V$ and a set of ***neighbors*** $v_1, v_2, v_3, \dots, v_d$

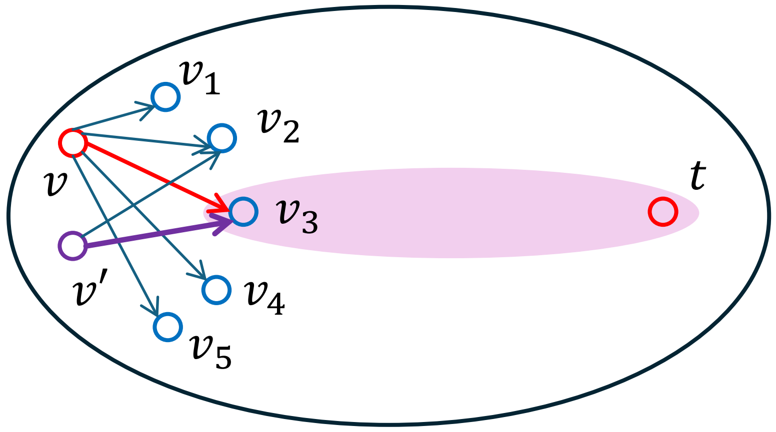
$$\Omega_v = \{\text{subgraphs of } G_v \text{ s.t. } v \to t\}$$

- For any neighbor v_i,

$$\Omega_i = \{\text{subgaphs of } G_v \text{ s.t. } v \to v_i \to t\}$$

- Vertex v must reach t through some neighbors

$$\Omega_v = \cup_{i=1}^{d} \Omega_i$$



- Apply ***Karp-Luby*** to compute an estimate Z_v such that

$$\Pr\left[Z_v \in \left(1 \pm \mathrm{poly}\left(\frac{\varepsilon}{m}\right)\right)|\Omega_v|\right] \geq \frac{2}{3}$$

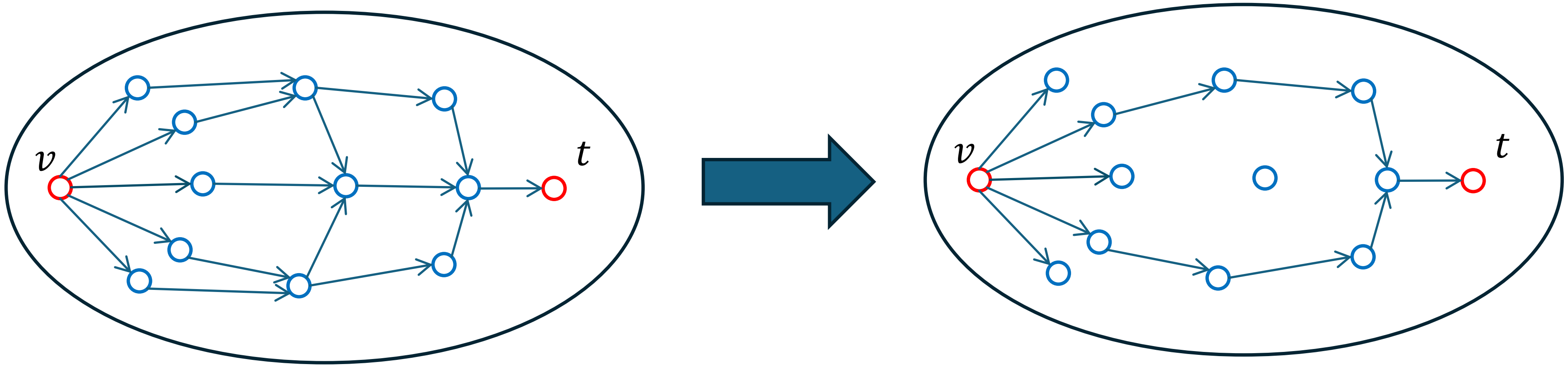
- Apply ***median-trick*** to boost the successful prob.

$$2/3 \to 1 - \exp(-\mathrm{poly}(m))$$

- ***Higher accuracy*** and ***higher successful probability*** require ***more samples*** in set S_{v_i}

- Different vertices v, v' may have the same neighbor v_i, we ***reuse*** S_{v_i} when computing Z_v and $Z_{v'}$

Generate the samples via self-reduction



graph G_v random subgraph of G_v s.t. $v \to t$

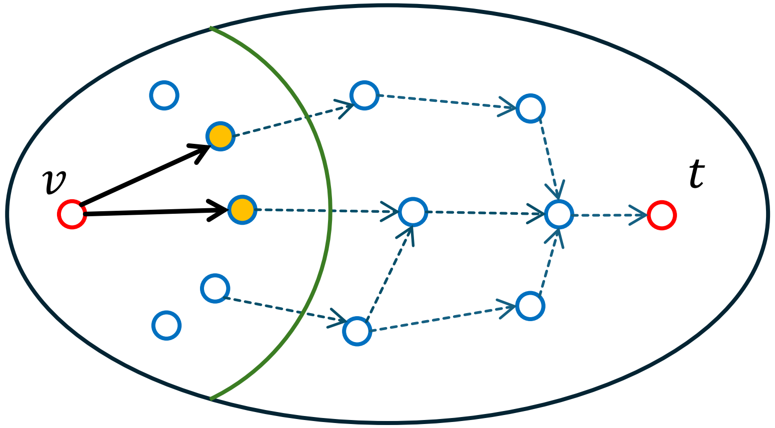
- Go through all edges $e_1, e_2, \ldots, e_m$ in graph G_v in some ordering
- For each edge e_i, sample $X(e_i) \in \{0,1\}$ that indicates whether e_i in random subgraph
- Compute the ***conditional marginal probability***

$$p(e_i) = \Pr[\, X(e_i) = 1 \mid X(e_1), X(e_2), \ldots, X(e_{i-1}) \,]$$

- Set $X(e_i) = 1$ with prob. $p(e_i)$ and set $X(e_i) = 0$ with prob. $1 - p(e_i)$
- Repeat the processing for $\mathrm{poly}(m/\epsilon)$ times to generate $\mathrm{poly}(m/\epsilon)$ samples

Generate the samples via self-reduction

- Suppose we have sampled $X(e_i)$ for $i \le \ell$
- Let $\mathcal{E}$ be set of edges e_i s.t. $X(e_i) = 1$
- $\Lambda = \{w \in V: v \to w \text{ through edges in } \mathcal{E}\}$

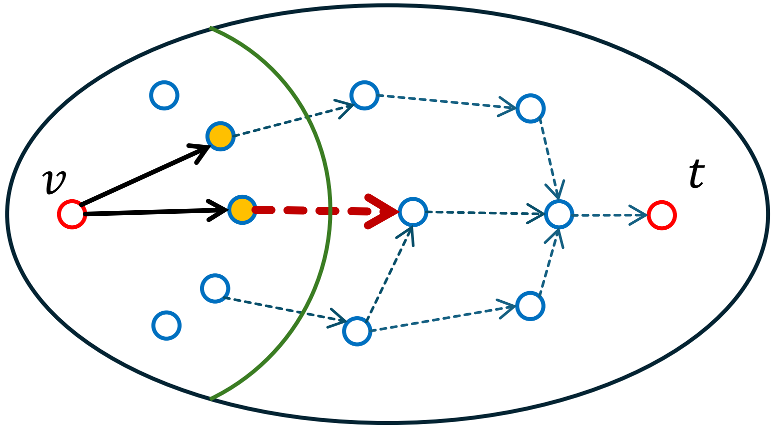


Λ the set of vertices v can reach

Generate the samples via self-reduction

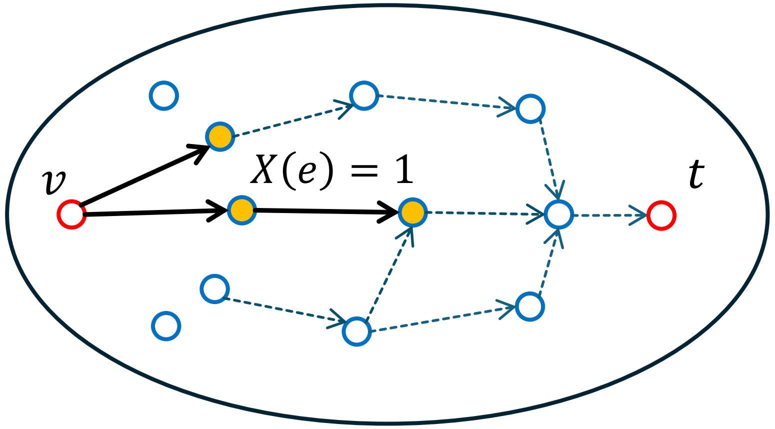
- Suppose we have sampled $X(e_i)$ for $i \le \ell$
- Let $\mathcal{E}$ be set of edges e_i s.t. $X(e_i) = 1$
- $\Lambda = \{w \in V: v \to w \text{ through edges in } \mathcal{E}\}$

Λ the set of vertices v can reach



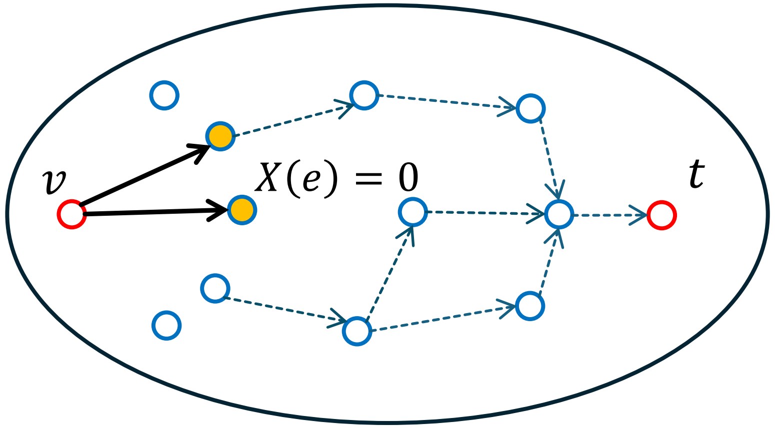
- Pick the edge $e = (u, w)$ from Λ to the ***out-boundary*** $\partial\Lambda$ where w has the largest topological order

$$\partial\Lambda = \{w \notin \Lambda: \exists u \in \Lambda\ s.t.\ (u, w) \in E\}$$

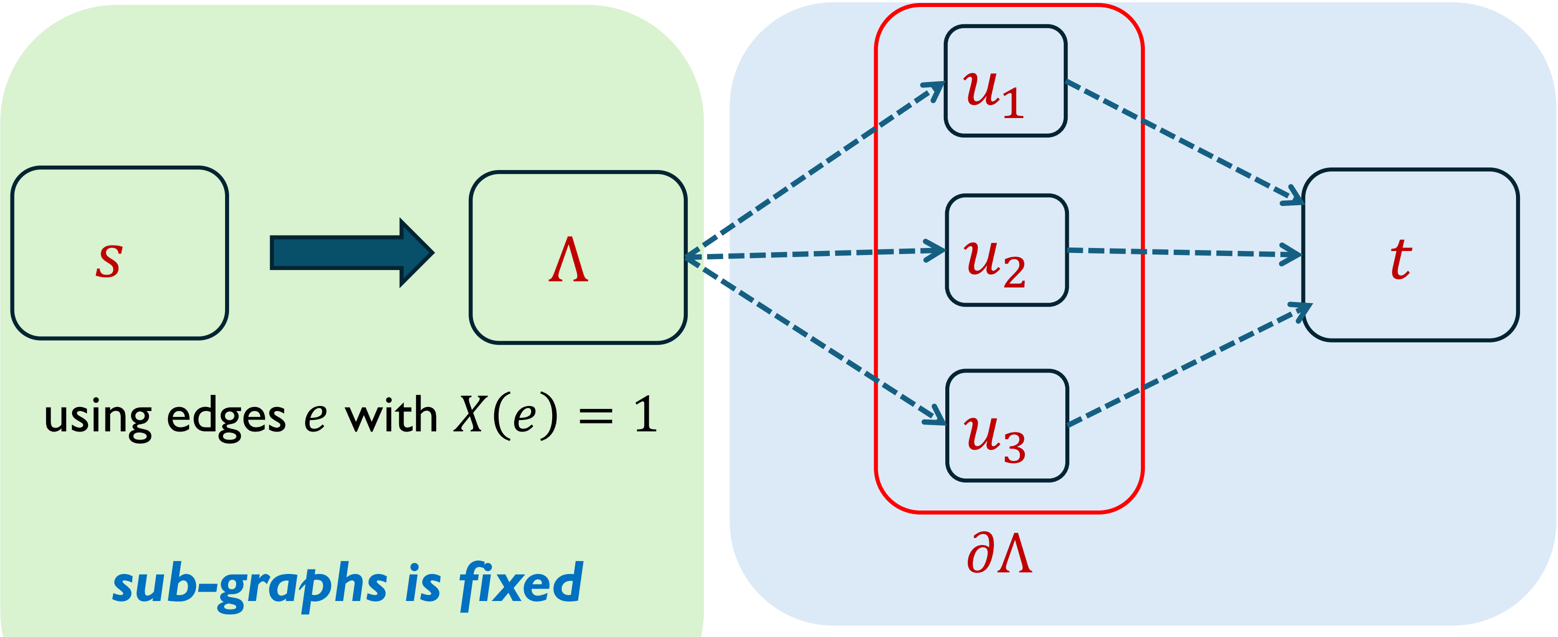


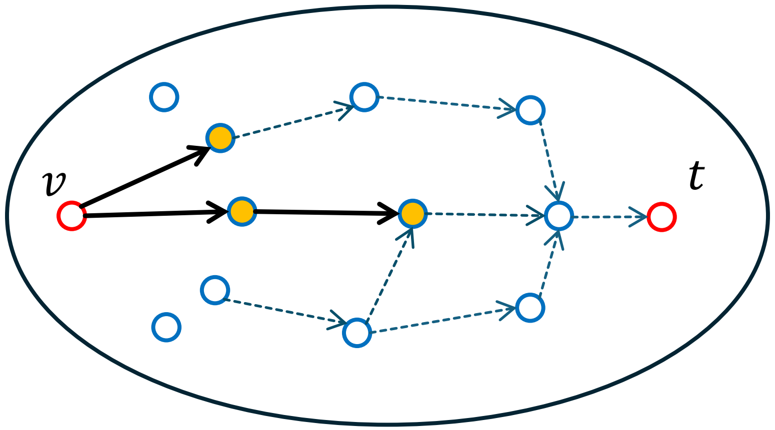
Prob ∝ # {subgraphs v can reach t}

OR



Prob ∝ # {subgraphs v cannot reach t}





Approx, count # {subgraphs s.t. Λ can reach t through $\partial\Lambda$}

$$\Omega = \bigcup_{u \in \partial\Lambda} \Omega_u$$

- $\Omega =$ {subgraphs s.t. Λ can reach t} is the set we want to count
- $\Omega_u =$ {subgraphs s.t. Λ can reach t through ***u***}
- Run ***Karp-Luby algorithm*** by ***reusing*** the samples in nodes at $\partial\Lambda$

Sampling Algorithm at Each Node

approximate counting via Karp-Luby (reusing samples) → estimate ***conditional marginal probability*** → Generate samples by sampling edges one-by-one

↑ also used for compute the ***count estimate Z_v*** at each node

- Sort all vertices in topological ordering $t = v_1 \prec v_2 \prec \cdots \prec v_n = s$
- G_v: subgraph containing all vertices that can be reached from $v \in V$

$$\Omega_v = \{\text{subgraphs of } G_v \text{ such that } v \text{ can reach } t\}$$

- Z_v: a **count estimate** of the size $\#\Omega_v$
- S_v: a set of **samples**, where each $H \in S_v$ is ***uniform random sample*** from Ω_v

High-level analysis of the algorithm

Challenge: ***reusing*** samples introduces complicated ***correlations***

Key Property: the algorithm only use samples for approx. counting (estimate marginal prob.)

approximate counting via Karp-Luby (reusing samples)	→	estimate *conditional marginal prob. p*	→	Generate samples by sampling edges one-by-one

use ***fresh randomness*** to sample $X(e)$ with $\Pr[X(e) = 1] = p$

- The algorithm is correct if random samples ***correctly*** estimate counts or marginal prob.

$\text{poly}(m)$ number samples	**Chernoff + median trick** →	estimation is correct with prob. $1 - \exp(-m)$

- Conditional on estimate is correctly only bias random samples with ***exp-small*** error
- Show the whole algorithm is correct by an **induction proof**

Analysis of the algorithm

When algorithm scans u, the following ***good event*** happens with ***high probability***

A. Random samples on all vertices $v \preccurlyeq u$ are ***accurate***

$$TV(\text{random samples in } S_v, \text{unifrom distribution of } \Omega_v) \leq \exp(-\Omega(m))$$

B. If we run Karp-Luby on any $\partial\Lambda \subseteq \{v \mid v \preccurlyeq u\}$, the answer is ***accurate***

$$\text{Estimate } \cup_{w\in\partial\Lambda} \Omega_w \text{ with } \left(1 \pm \text{poly}\left(\frac{\epsilon}{m}\right)\right) \text{ error}$$

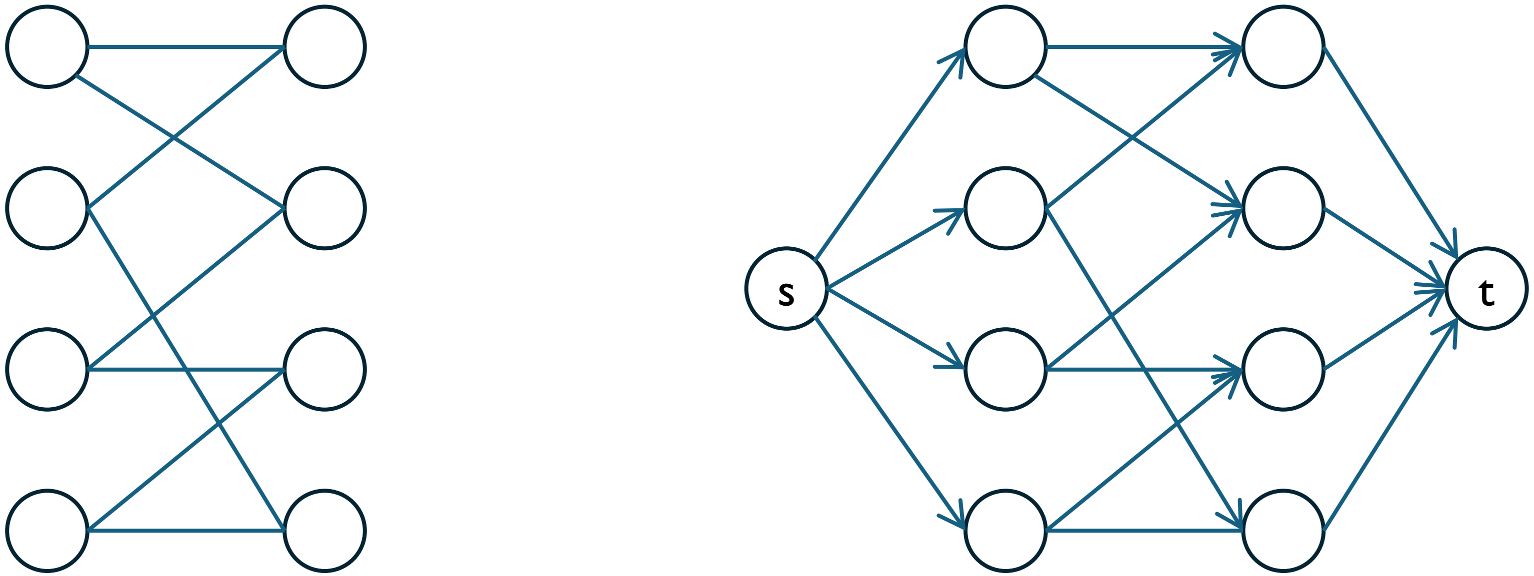
the error becomes worse during induction
the final error can be controlled

The induction proof

- If (B) holds for all $v \prec u$, then the algorithm at u generates ***prefect samples*** (via a filter)
- If (A)(B) holds for all $v \prec u$, then (B) holds for u with prob. $1 - O(\mathrm{e}^{-\Omega(m)})$
- Conditional on (B) holds for u only bias the sample with $O(e^{-\Omega(m)})$ TV-distance error

Proof of hardness

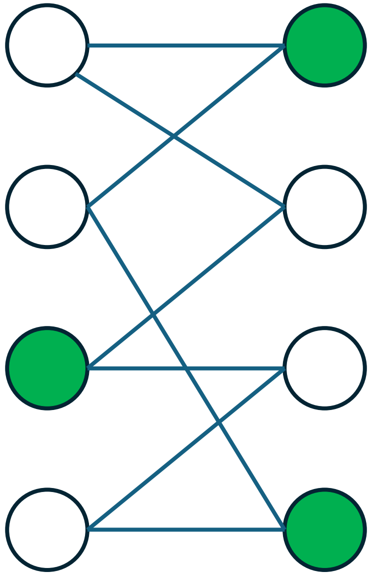
There is ***no FPRAS*** for two-terminal network ***unreliability*** in DAGs
unless there is an FPRAS for ***#BIS*** problem

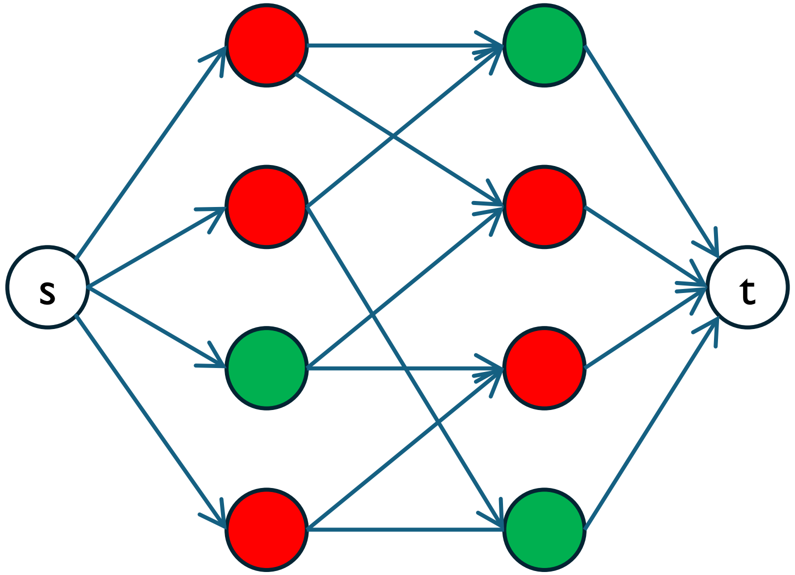


#BIS $\leq$ vertex unreliability

Proof of hardness

There is ***no FPRAS*** for two-terminal network ***unreliability*** in DAGs
unless there is an FPRAS for ***#BIS*** problem

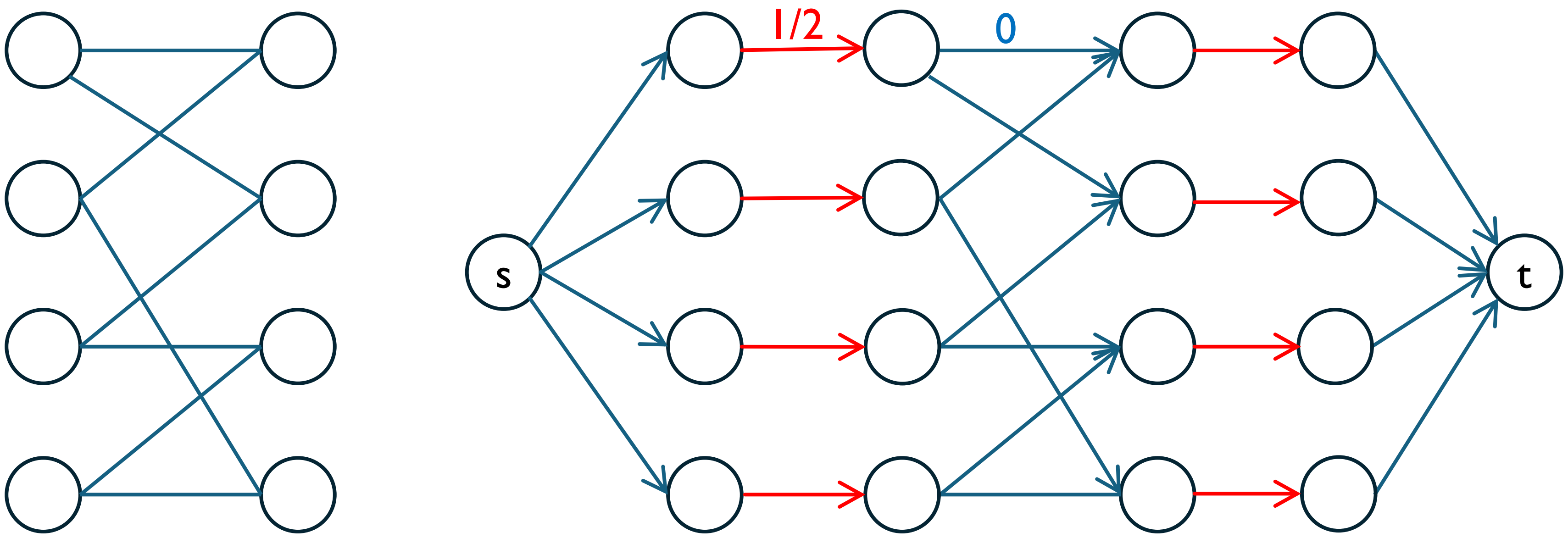




#BIS $\leq$ vertex unreliability

Proof of hardness

There is ***no FPRAS*** for two-terminal network ***unreliability*** in DAGs
unless there is an FPRAS for ***#BIS*** problem



#BIS $\leq$ edge unreliability

Open Problems

- ***Faster*** algorithm for s-t reliability in DAGs
- ***Simple*** algorithm for s-t reliability in DAGs
- FPTAS (***deterministic***) algorithm for s-t reliability in DAGs
- Algorithm or hardness for approximating s-t reliability in ***undirected / directed*** graphs
- Simple or faster FPRAS/FPTAS for ***#NFA***

Thank you
Q&A