An FPRAS for two terminal reliability in directed acyclic graphs

Heng Guo (University of Edinburgh)

Based on joint work with Weiming Feng (ETH Zürich)

A&C workshop, University of Cambridge, Apr 15, 2024

Complexity class #**P** by Valiant (1979):

a counting analogue of **NP**

E.g. counting the number of solutions to CNF formulas

Other examples:

determinants / permanents of matrices;
evaluation of probabilities;
partition functions in statistical physics;
counting discrete structures ...

The complexity of approximate counting

What about (multiplicatively) approximating #**P**-complete problems?

- at most **NP**-hard (Stockmeyer 1983; Valiant and Vazirani, 1986);
- for many problems, polynomial approximation can be amplified into ε-approximation paying only a polynomial runtime overhead.

Typically, an approximate counting problem is either **NP**-hard or FPRASable.

FPRASes do exist. Famous examples include:

- the number of solutions to DNF formulae (Karp and Luby, 1983);
- all-terminal network (un)reliability (Karger 1999);
- the volume of convex bodies (Dyer, Frieze, and Kannan, 1991);
- the permanent of non-negative matrices (Jerrum, Sinclair, and Vigoda, 2004).

However, there are still many open problems in approximate counting!

The complexity of approximate counting

What about (multiplicatively) approximating #**P**-complete problems?

- at most **NP**-hard (Stockmeyer 1983; Valiant and Vazirani, 1986);
- for many problems, polynomial approximation can be amplified into ε-approximation paying only a polynomial runtime overhead.

Typically, an approximate counting problem is either **NP**-hard or FPRASable.

FPRASes do exist. Famous examples include:

- the number of solutions to DNF formulae (Karp and Luby, 1983);
- all-terminal network (un)reliability (Karger 1999);
- the volume of convex bodies (Dyer, Frieze, and Kannan, 1991);
- the permanent of non-negative matrices (Jerrum, Sinclair, and Vigoda, 2004).

However, there are still many open problems in approximate counting!

Network reliability

Network reliability

Given a directed or undirected graph (a.k.a. network) $G = (V, E)$, define a random subgraph $G(p)$ by removing each edge independently with probability p.

Two-terminal reliability: given $s, t \in V$,

$$\Pr\left[s \xrightarrow{G(p)} t\right]$$

Directed and undirected Two-terminal reliability (and a few other variants) are featured in the original list of 13 #**P**-complete problems by Valiant (1979).

One may ask the probability of other properties of $G(p)$, such as whether $G(p)$ is connected (All-terminal reliability).

TWO-TERMINAL RELIABILITY: $\Pr\left[s \xrightarrow{G(p)} t\right]$

In other words, we want to compute

$$Z_{\mathrm{rel}}(G,p) := \sum_{R \subseteq E : s \xrightarrow{(V,R)} t} p^{|E \setminus R|}(1-p)^{|R|}$$

For example:

$Z_{\mathrm{rel}}($  $, p) =$  $= (1-p)^{n-1}$

$Z_{\mathrm{rel}}($  $, p) =$  $+$  $+$  $+$  $+$  $+$  $+$ 

$$= (1-p)^4 + 4p(1-p)^3 + 2p^2(1-p)^2$$

Computational complexity of reliability

Exact evaluation of almost all variants of reliability is #**P**-complete, shown by the pioneer work of Valiant (1979), Jerrum (1981), Provan and Ball (1983), etc.

Much less is known about their approximation complexity.

A famous breakthrough is by Karger (1999), who gave an FPRAS for All-terminal unreliability.

(However, good approximation of *unreliability* is not necessarily a good approximation for *reliability* when reliability is exponentially small.)

All-terminal reliability is shown to have an FPRAS by G. and Jerrum (2019), resolving positively conjectures by Provan and Ball (1983), Welsh (1993), Karger (1999), etc.

Main results

We gave an FPRAS for Two-terminal reliability in directed acyclic graphs (DAGs).

Theorem (Feng and G., 2023)

Let G be a DAG and $\mathbf{p}$ denote failure probabilities. There is a randomized algorithm that takes $(G, \mathbf{p}, s, t, \varepsilon)$ as inputs and outputs a $(1 \pm \varepsilon)$-approximation to $s-t$ reliability with probability at least $3/4$ in time $\widetilde{O}(n^6 m^4 \max\{m^4, \varepsilon^{-4}\})$.

This answers positively a conjecture by Zenklusen and Laumanns (2011).

On the flip side, the corresponding unreliability problem is unlikely to have an FPRAS.

Theorem (Feng and G., 2023)

There is no FPRAS to estimate $s-t$ unreliability in DAGs unless there is an FPRAS for #BIS.

Here #BIS is the problem of counting independent sets in bipartite graphs. It is conjectured to have no FPRAS.

Simultaneous work

Independently, Amarilli, van Bremen, and Meel (2023) reduce $s-t$ *reliability* in DAGs to counting the number of accepting paths of a given length for non-deterministic automata (#NFA). The latter problem has an FPRAS by Arenas, Croquevielle, Jayaram, and Riveros (2021).

ACJR21's algorithm runs in time $O\left(\left(\frac{n\ell}{\varepsilon}\right)^{17}\right)$ for an n-state NFA and strings of length ℓ.

AvBM23 reduces a DAG with n vertices and m edges to a #NFA instance counting length m accepting strings for an NFA with $O(m^2)$ states. Thus their running time is like $O\left(\frac{m^{51}}{\varepsilon^{17}}\right)$.

Computational complexity of approximating reliability

Terminal	Graphs	Type	Complexity	Best run-time
All	Undirected	Unrel	FPRAS (K99)	$\frac{m^{1+o(1)}}{\varepsilon^2} + \widetilde{O}\left(\frac{n^{1.5}}{\varepsilon^3}\right)$ (CHLP23)
All	Undirected	Rel	FPRAS (GJ19)	$\widetilde{O}\left(\frac{mn}{\varepsilon^2}\right)$ (CGZZ23)
$s-t$	DAG	Rel	FPRAS (FG23, AvBM23)	$\widetilde{O}\left(n^6 m^4 \max\{m^4, \varepsilon^{-4}\}\right)$ (FG23)
$s-t$	DAG	Unrel	#BIS-hard (FG23)	—
$S-t$	DAG	Rel	**NP**-hard (upcoming)	—

CHLP23: Cen, He, Li, and Panigrahi (2023)

CGZZ23: Chen, G., Zhang, and Zou (2023)

Some natural attempts

(and why they do not succeed)

Naive Monte Carlo

A natural unbiased estimator $\widetilde{Z}$ of Z_{rel}:

1. Draw k independent subgraphs $(S_i)_{i \in [k]}$ of $G(p)$.
2. Let

$$\widetilde{Z} := \frac{1}{k} \sum_{i \in [k]} \mathbb{1}_{s \to t}(S_i),$$

where $\mathbb{1}_{s \to t}(S)$ is the indicator variable whether $s \xrightarrow{(V,S)} t$.

It is easy to see that $\mathbb{E}\,\widetilde{Z} = Z_{rel}$.

However, if Z_{rel} is exponentially small (e.g. $Z_{rel}(P_n, p) = (1-p)^{n-1}$), then we will almost never see a connected S_i.

In that case, the relative variance of $\mathbb{1}_{s \to t}(S)$ is exponentially large, and k has to be exponentially large to yield a good approximation.

Naive Monte Carlo

A natural unbiased estimator $\widetilde{Z}$ of Z_{rel}:

1. Draw k independent subgraphs $(S_i)_{i\in[k]}$ of $G(p)$.
2. Let
$$\widetilde{Z} := \frac{1}{k}\sum_{i\in[k]} \mathbb{1}_{s\to t}(S_i),$$
where $\mathbb{1}_{s\to t}(S)$ is the indicator variable whether $s \xrightarrow{(V,S)} t$.

It is easy to see that $\mathbb{E}\,\widetilde{Z} = Z_{rel}$.

However, if Z_{rel} is exponentially small (e.g. $Z_{rel}(P_n, p) = (1-p)^{n-1}$), then we will almost never see a connected S_i.

In that case, the relative variance of $\mathbb{1}_{s\to t}(S)$ is exponentially large, and k has to be exponentially large to yield a good approximation.

Naive Monte Carlo

A natural unbiased estimator $\widetilde{Z}$ of Z_{rel}:

1. Draw k independent subgraphs $(S_i)_{i \in [k]}$ of $G(p)$.
2. Let

$$\widetilde{Z} := \frac{1}{k} \sum_{i \in [k]} \mathbb{1}_{s \to t}(S_i),$$

   where $\mathbb{1}_{s \to t}(S)$ is the indicator variable whether $s \xrightarrow{(V,S)} t$.

It is easy to see that $\mathbb{E}\,\widetilde{Z} = Z_{\mathrm{rel}}$.

However, if Z_{rel} is exponentially small (e.g. $Z_{\mathrm{rel}}(P_n, p) = (1-p)^{n-1}$), then we will almost never see a connected S_i.

In that case, the relative variance of $\mathbb{1}_{s \to t}(S)$ is exponentially large, and k has to be exponentially large to yield a good approximation.

Previous techniques

Karger's method for (undirected) All-terminal unreliability:

Naive Monte Carlo + random contraction

(Undirected) All-terminal reliability:

via the counting to sampling reduction (Jerrum, Valiant, and Vazirani, 1986) and then either

- Partial rejection sampling (G. and Jerrum, 2019); or
- Markov chain (Anari, Liu, Oveis Gharan, and Vinzant, 2019).

All these techniques rely on some nice structure of the solution space, which Two-terminal reliability (even in DAG) does not have.

Previous techniques

Karger's method for (undirected) All-terminal unreliability:

Naive Monte Carlo + random contraction

(Undirected) All-terminal reliability:

via the counting to sampling reduction (Jerrum, Valiant, and Vazirani, 1986) and then either

- Partial rejection sampling (G. and Jerrum, 2019); or
- Markov chain (Anari, Liu, Oveis Gharan, and Vinzant, 2019).

All these techniques rely on some nice structure of the solution space, which Two-terminal reliability (even in DAG) does not have.

Previous techniques

Karger's method for (undirected) All-terminal unreliability:

Naive Monte Carlo + random contraction

(Undirected) All-terminal reliability:

via the counting to sampling reduction (Jerrum, Valiant, and Vazirani, 1986) and then either

- Partial rejection sampling (G. and Jerrum, 2019); or
- Markov chain (Anari, Liu, Oveis Gharan, and Vinzant, 2019).

All these techniques rely on some nice structure of the solution space, which Two-terminal reliability (even in DAG) does not have.

Markov chain Monte Carlo

Markov chains are the "off the shelf" approach to sampling from complicated distributions.

Here we want to sample $R \sim G(p)$ conditioned on $s \xrightarrow{R} t$.

There is a natural Markov chain converging to the desired distribution:

1. Let $C_0 = E$.
2. Given C_t, randomly pick an edge $e \in E$.
   If $\neg \left(s \xrightarrow{C_t \setminus \{e\}} t \right)$ then $C_{t+1} = C_t$. Otherwise,

$$C_{t+1} = \begin{cases} C_t \cup \{e\} & \text{with prob. } 1 - p; \\ C_t \setminus \{e\} & \text{with prob. } p. \end{cases}$$

This chain is slow for s and t connected via two length $n/2$ paths.

Dynamic programming meets Monte Carlo

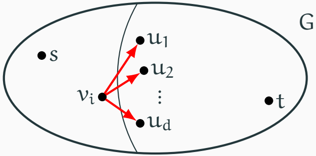
Dynamic programming

Our main inspiration is the #NFA algorithm of Arenas, Croquevielle, Jayaram, and Riveros (2021), which in turn builds upon a quasi-polynomial time algorithm for generating words in context-free grammars (#CFG) by Gore, Jerrum, Kannan, Sweedyk, and Mahaney (1997).

Given the topological ordering $s = v_1, v_2, \ldots, v_n = t$ of the input DAG, we estimate $R_i := \Pr\left[v_i \xrightarrow{G(p)} t\right]$ from $i = n$ to 1.

Note that for v_i, we can safely ignore all vertices unreachable from v_i. Call the resulting graph G_{v_i}, and we only need to consider $G_{v_i}(p)$.

The inductive step



Let v_i's neighbours be $u_1, \ldots, u_d$. Then,

$$\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t\right] = \Pr\left[\exists j \in [d],\ ((v_i, u_j) \in G_{v_i}(p)) \wedge (u_j \xrightarrow{G_{u_j}(p)} t)\right].$$

This reminds us to use the DNF counting technique of Karp and Luby (1983).

Karp-Luby in action

Given DNF $\Phi = C_1 \vee C_2 \vee \cdots \vee C_m$, counting the number of solutions to Φ is equivalent to evaluating

$$\Pr[\sigma \models \Phi] = \Pr[\exists i \in [m],\ \sigma \models C_i],$$

where σ is a uniformly at random assignment.

Let Ω_i be the set of solutions to C_i, and $p_i := \Pr[\sigma \models C_i] = \frac{|\Omega_i|}{2^n} = 2^{-|C_i|}$.

Karp-Luby goes as follows:

1. Draw i with probability proportional to p_i;
2. draw uniformly σ such that $\sigma \models C_i$ (namely uniformly from Ω_i);
3. Let t_σ be the number of clauses σ satisfies. Output $Z := \frac{1}{t_\sigma}$.

Then $\mathbb{E}[Z] = \frac{1}{\sum_{i\in[m]} p_i} \cdot \Pr[\exists i \in [m],\ \sigma \models C_i]$.

Also, $\mathbb{E}[Z] \geqslant \frac{1}{m}$, implying that $\frac{\mathrm{Var}(Z)}{\mathbb{E}[Z]^2} \leqslant m^2$.

Karp-Luby in action (for reliability)

We want to estimate $\Pr\left[\exists j \in [d],\ ((v_i, u_j) \in G_{v_i}(p)) \wedge (u_j \xrightarrow{G_{u_j}(p)} t)\right] = \Pr[\exists j \in [d],\ \mathcal{E}_j]$.

Let $p_j := \Pr[\mathcal{E}_j] = \Pr[(v_i, u_j) \in G_{v_i}(p)] \times \Pr\left[u_j \xrightarrow{G_{u_j}(p)} t\right] = (1-p)R_{u_j}$.

Then we can

1. draw $j \in [d]$ with probability proportional to p_j;
2. draw $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$; draw other edges in G_{v_i} independently with probability $1-p$; then add (v_i, u_j) to get a sample S';
3. let $t_{S'}$ be the number of events $\mathcal{E}_j$ occurring under S'; output $Z := \frac{1}{t_{S'}}$.

The same analysis implies that the expectation of this estimator is what we want and the relative variance is small.

Karp-Luby in action (for reliability)

We want to estimate $\Pr\left[\exists j \in [d],\ ((v_i, u_j) \in G_{v_i}(p)) \wedge (u_j \xrightarrow{G_{u_j}(p)} t)\right] = \Pr[\exists j \in [d],\ \mathcal{E}_j]$.

Let $p_j := \Pr[\mathcal{E}_j] = \Pr[(v_i, u_j) \in G_{v_i}(p)] \times \Pr\left[u_j \xrightarrow{G_{u_j}(p)} t\right] = (1-p)R_{u_j}$.

Then we can

1. draw $j \in [d]$ with probability proportional to p_j;
2. draw $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$; draw other edges in G_{v_i} independently with probability $1-p$; then add (v_i, u_j) to get a sample S';
3. let $t_{S'}$ be the number of events $\mathcal{E}_j$ occurring under S'; output $Z := \frac{1}{t_{S'}}$.

The same analysis implies that the expectation of this estimator is what we want and the relative variance is small.

Karp-Luby in action (for reliability)

We want to estimate $\Pr\left[\exists j \in [d],\ ((v_i, u_j) \in G_{v_i}(p)) \wedge (u_j \xrightarrow{G_{u_j}(p)} t)\right] = \Pr[\exists j \in [d],\ \mathcal{E}_j]$.

Let $p_j := \Pr[\mathcal{E}_j] = \Pr[(v_i, u_j) \in G_{v_i}(p)] \times \Pr\left[u_j \xrightarrow{G_{u_j}(p)} t\right] = (1-p)R_{u_j}$.

Then we can

1. draw $j \in [d]$ with probability proportional to p_j;
2. draw $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$; draw other edges in G_{v_i} independently with probability $1-p$; then add (v_i, u_j) to get a sample S';
3. let $t_{S'}$ be the number of events $\mathcal{E}_j$ occurring under S'; output $Z := \frac{1}{t_{S'}}$.

The same analysis implies that the expectation of this estimator is what we want and the relative variance is small.

Karp-Luby in action (for reliability)

We want to estimate $\Pr\left[\exists j \in [d],\ ((v_i, u_j) \in G_{v_i}(p)) \land (u_j \xrightarrow{G_{u_j}(p)} t)\right] = \Pr[\exists j \in [d],\ \mathcal{E}_j]$.

Let $p_j := \Pr[\mathcal{E}_j] = \Pr[(v_i, u_j) \in G_{v_i}(p)] \times \Pr\left[u_j \xrightarrow{G_{u_j}(p)} t\right] = (1-p)R_{u_j}$.

Then we can

1. draw $j \in [d]$ with probability proportional to p_j;
2. draw $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$; draw other edges in G_{v_i} independently with probability $1-p$; then add (v_i, u_j) to get a sample S';
3. let $t_{S'}$ be the number of events $\mathcal{E}_j$ occurring under S'; output $Z := \frac{1}{t_{S'}}$.

The same analysis implies that the expectation of this estimator is what we want and the relative variance is small.

Karp-Luby in action (for reliability)

We want to estimate $\Pr\left[\exists j \in [d],\ ((v_i, u_j) \in G_{v_i}(p)) \wedge (u_j \xrightarrow{G_{u_j}(p)} t)\right] = \Pr[\exists j \in [d],\ \mathcal{E}_j]$.

Let $p_j := \Pr[\mathcal{E}_j] = \Pr[(v_i, u_j) \in G_{v_i}(p)] \times \Pr\left[u_j \xrightarrow{G_{u_j}(p)} t\right] = (1-p)R_{u_j}$.

Then we can

1. draw $j \in [d]$ with probability proportional to p_j;
2. draw $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$; draw other edges in G_{v_i} independently with probability $1-p$; then add (v_i, u_j) to get a sample S';
3. let $t_{S'}$ be the number of events $\mathcal{E}_j$ occurring under S'; output $Z := \frac{1}{t_{S'}}$.

The same analysis implies that the expectation of this estimator is what we want and the relative variance is small.

Karp-Luby in action (for reliability)

We want to estimate $\Pr\left[\exists j \in [d],\ ((v_i, u_j) \in G_{v_i}(p)) \wedge (u_j \xrightarrow{G_{u_j}(p)} t)\right] = \Pr[\exists j \in [d],\ \mathcal{E}_j]$.

Let $p_j := \Pr[\mathcal{E}_j] = \Pr[(v_i, u_j) \in G_{v_i}(p)] \times \Pr\left[u_j \xrightarrow{G_{u_j}(p)} t\right] = (1-p)R_{u_j}$.

Then we can

1. draw $j \in [d]$ with probability proportional to p_j;
2. draw $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$; draw other edges in G_{v_i} independently with probability $1-p$; then add (v_i, u_j) to get a sample S';
3. let $t_{S'}$ be the number of events $\mathcal{E}_j$ occurring under S'; output $Z := \frac{1}{t_{S'}}$.

The same analysis implies that the expectation of this estimator is what we want and the relative variance is small.

Karp-Luby in action (for reliability)

We want to estimate $\Pr\left[\exists j \in [d],\ ((v_i, u_j) \in G_{v_i}(p)) \wedge (u_j \xrightarrow{G_{u_j}(p)} t)\right] = \Pr[\exists j \in [d],\ \mathcal{E}_j]$.

Let $p_j := \Pr[\mathcal{E}_j] = \Pr[(v_i, u_j) \in G_{v_i}(p)] \times \Pr\left[u_j \xrightarrow{G_{u_j}(p)} t\right] = (1-p)R_{u_j}$.

Then we can

1. draw $j \in [d]$ with probability proportional to p_j;
2. draw $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$; draw other edges in G_{v_i} independently with probability $1-p$; then add (v_i, u_j) to get a sample S';
3. let $t_{S'}$ be the number of events $\mathcal{E}_j$ occurring under S'; output $Z := \frac{1}{t_{S'}}$.

The same analysis implies that the expectation of this estimator is what we want and the relative variance is small.

Karp-Luby in action (for reliability)

We want to estimate $\Pr\left[\exists j \in [d],\ ((v_i, u_j) \in G_{v_i}(p)) \wedge (u_j \xrightarrow{G_{u_j}(p)} t)\right] = \Pr[\exists j \in [d],\ \mathcal{E}_j]$.

Let $p_j := \Pr[\mathcal{E}_j] = \Pr[(v_i, u_j) \in G_{v_i}(p)] \times \Pr\left[u_j \xrightarrow{G_{u_j}(p)} t\right] = (1-p)R_{u_j}$.

Then we can

1. draw $j \in [d]$ with probability proportional to p_j;
2. draw $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$; draw other edges in G_{v_i} independently with probability $1-p$; then add (v_i, u_j) to get a sample S';
3. let $t_{S'}$ be the number of events $\mathcal{E}_j$ occurring under S'; output $Z := \frac{1}{t_{S'}}$.

The same analysis implies that the expectation of this estimator is what we want and the relative variance is small.

Are we there yet?

To implement Karp-Luby, we need to do two things:

- calculate $p_j = (1-p)R_{u_j}$, which can be done as R_{u_j} is known in previous steps;
- sample $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$.

How do we sample?

The sampling to counting reduction *a la* Jerrum, Valiant, and Vazirani (1986) to the rescue!

We maintain a set of samples S_{v_i} in addition to R_{v_i}.

Are we there yet?

To implement Karp-Luby, we need to do two things:

- calculate $p_j = (1 - p)R_{u_j}$, which can be done as R_{u_j} is known in previous steps;
- sample $S \sim G_{u_j}(p)$ conditioned on $u_j \xrightarrow{S} t$.

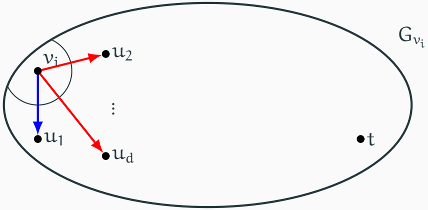
How do we sample?

The sampling to counting reduction *a la* Jerrum, Valiant, and Vazirani (1986) to the rescue!

We maintain a set of samples S_{v_i} in addition to R_{v_i}.

JVV'86 self-reduction:

sample edges one-by-one, using marginal probabilities conditioned on previous outcomes

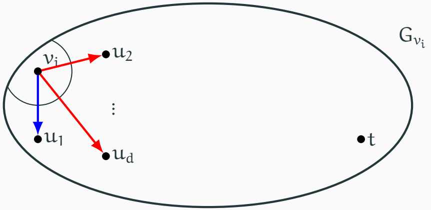


The marginal of $e = (v_i, u_1)$ is

$$\frac{\Pr[e \in G_{v_i}(p)] \Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right]}{\Pr[e \in G_{v_i}(p)] \Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right] + \Pr[e \notin G_{v_i}(p)] \Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \notin G_{v_i}(p)\right]}$$

JVV'86 self-reduction:

sample edges one-by-one, using marginal probabilities conditioned on previous outcomes



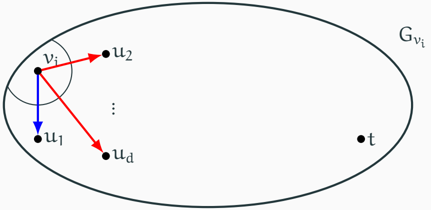
The marginal of $e = (v_i, u_1)$ is

$$\frac{\Pr\left[e \in G_{v_i}(p)\right] \Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right]}{\Pr\left[e \in G_{v_i}(p)\right] \Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right] + \Pr\left[e \notin G_{v_i}(p)\right] \Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \notin G_{v_i}(p)\right]}$$

Sampling to counting reduction

JVV'86 self-reduction:

sample edges one-by-one, using marginal probabilities conditioned on previous outcomes

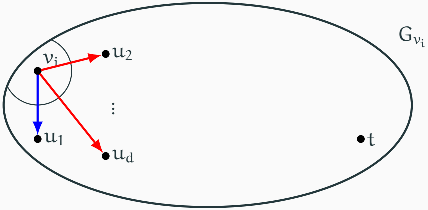


The marginal of $e = (v_i, u_1)$ is

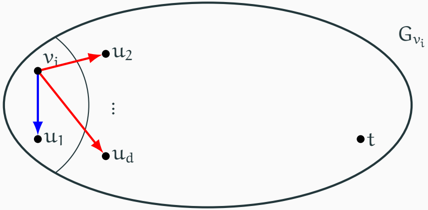
$$\frac{(1-p)\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right]}{(1-p)\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right] + p\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \notin G_{v_i}(p)\right]}$$

Computing the marginal

How do we estimate $\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right]$?



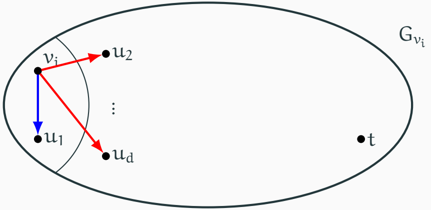
How do we estimate $\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right]$?



Contract v_i and u_1. Call it v_i'. Note that we would have already computed (R_u, S_u) for u that is a neighbour of v_i or u_1 (as it comes after v_i in the topological order).

Computing the marginal

How do we estimate $\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right]$?

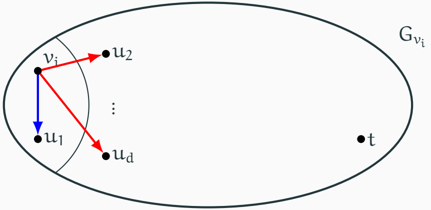


Contract v_i and u_1. Call it v_i'. Note that we would have already computed (R_u, S_u) for u that is a neighbour of v_i or u_1 (as it comes after v_i in the topological order).

That's all Karp-Luby needs to estimate $\Pr\left[v_i' \xrightarrow{G_{v_i'}(p)} t\right]$!

Computing the marginal

How do we estimate $\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right]$?



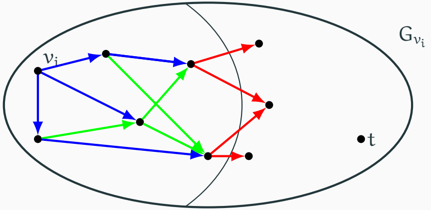
Contract v_i and u_1. Call it v_i'. Note that we would have already computed (R_u, S_u) for u that is a neighbour of v_i or u_1 (as it comes after v_i in the topological order).

That's all Karp-Luby needs to estimate $\Pr\left[v_i' \xrightarrow{G_{v_i'}(p)} t\right]$!

The case for $\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \notin G_{v_i}(p)\right]$ is similar. Then move on to the next edge and repeat.

Computing the marginal

How do we estimate $\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right]$?



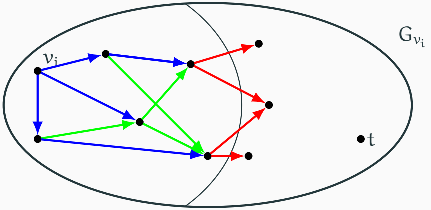
Contract v_i and u_1. Call it v_i'. Note that we would have already computed (R_u, S_u) for u that is a neighbour of v_i or u_1 (as it comes after v_i in the topological order).

That's all Karp-Luby needs to estimate $\Pr\left[v_i' \xrightarrow{G_{v_i'}(p)} t\right]$!

The case for $\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \notin G_{v_i}(p)\right]$ is similar. Then move on to the next edge and repeat.

Computing the marginal

How do we estimate $\Pr\left[v_i \xrightarrow{G_{v_i}(p)} t \mid e \in G_{v_i}(p)\right]$?



- Blue edges: those chosen using the marginal by Karp-Luby, forming a tree.
- Green edges: conditioned on the blue edges, they have no affect on reaching t, and thus have marginal $1 - p$.
- Red edges: the current frontier, whose tails have been processed (namely (R_u, S_u) has been computed already).

Efficiency of the algorithm

The self-reduction to generate one sample uses, say, k samples for each Karp-Luby step. If we generate fresh samples each time, in total at least k^n samples are required. The key to be efficient here is that samples can be reused!

Reusing samples introduces subtle correlation among R_{v_i}'s and the samples. However:

- accurate marginals $\xRightarrow{\text{union bound over all edges}}$ accurate samples;
- accurate samples $\xRightarrow{\text{union bound over all possibilities}}$ accurate marginals.

Accurate marginals come from accurate estimation of the R_{v_i}'s. The samples are subtly correlated, but they can be coupled with independent fresh samples with small error.

Efficiency of the algorithm

The self-reduction to generate one sample uses, say, k samples for each Karp-Luby step. If we generate fresh samples each time, in total at least k^n samples are required. The key to be efficient here is that samples can be reused!

Reusing samples introduces subtle correlation among R_{v_i}'s and the samples. However:

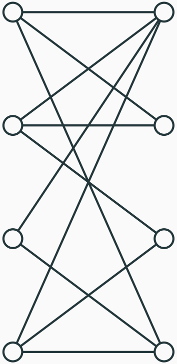
- accurate marginals $\xRightarrow{\text{union bound over all edges}}$ accurate samples;
- accurate samples $\xRightarrow{\text{union bound over all possibilities}}$ accurate marginals.

Accurate marginals come from accurate estimation of the R_{v_i}'s. The samples are subtly correlated, but they can be coupled with independent fresh samples with small error.

Hardness results

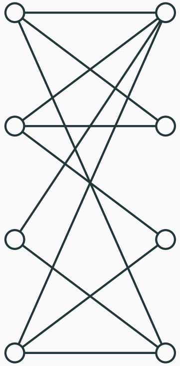
#BIS-hardness for Two-terminal vertex unreliability in DAGs

#BIS $\leqslant$ Two-terminal vertex unreliability



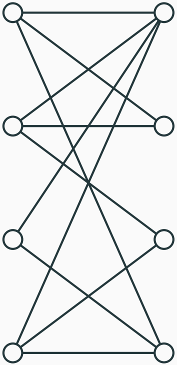
#BIS-hardness for Two-terminal vertex unreliability in DAGs

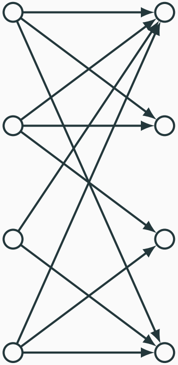
#BIS $\leqslant$ Two-terminal vertex unreliability



#BIS-hardness for Two-terminal vertex unreliability in DAGs

#BIS $\leqslant$ Two-terminal vertex unreliability

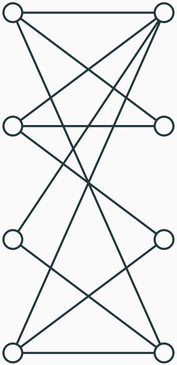


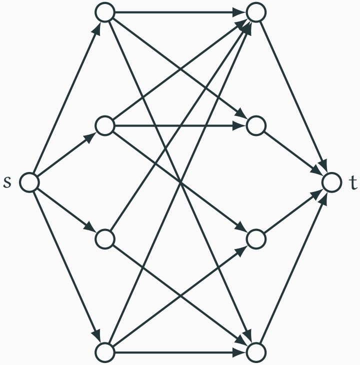


#BIS $\leqslant$ Two-terminal vertex unreliability

#BIS-hardness for Two-terminal vertex unreliability in DAGs

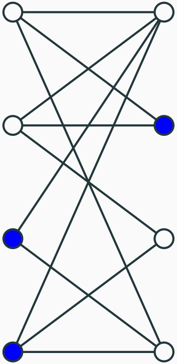
#BIS $\leqslant$ Two-terminal vertex unreliability

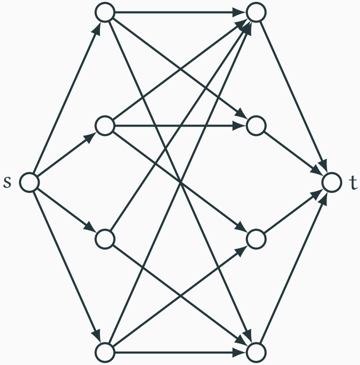




#BIS-hardness for Two-terminal vertex unreliability in DAGs

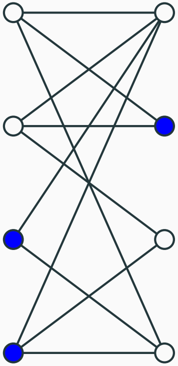
#BIS $\leqslant$ Two-terminal vertex unreliability

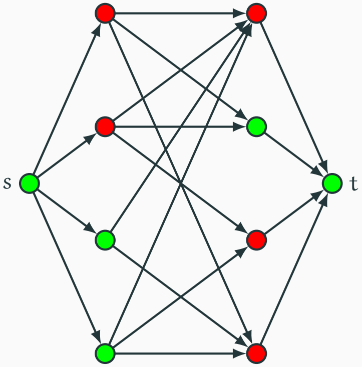




#BIS-hardness for Two-terminal vertex unreliability in DAGs

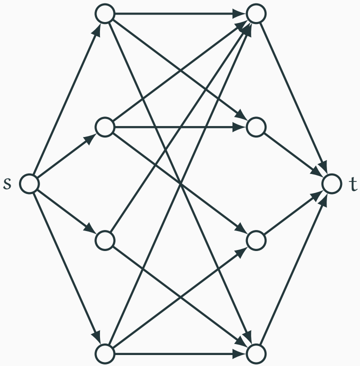
#BIS $\leqslant$ Two-terminal vertex unreliability





From vertex to edge unreliability in DAGs

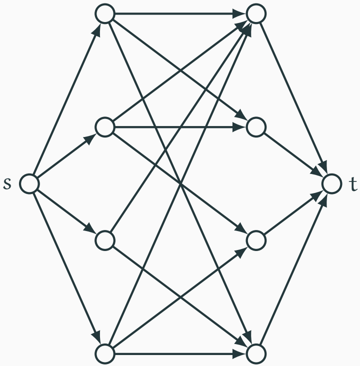
Vertex unreliability



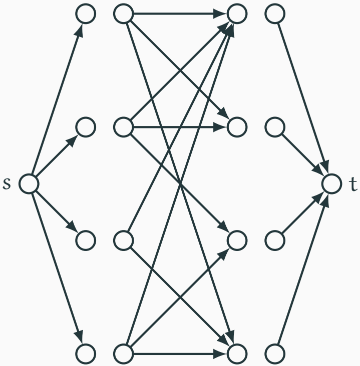
Edge unreliability

From vertex to edge unreliability in DAGs

Vertex unreliability

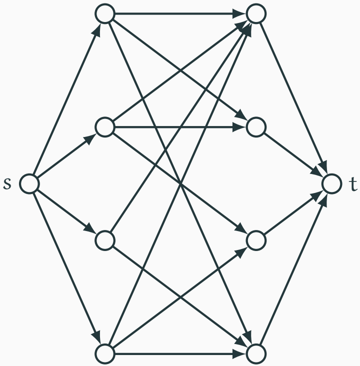


Edge unreliability

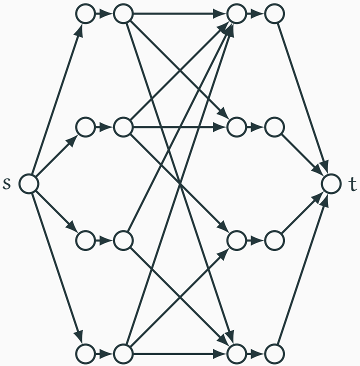


From vertex to edge unreliability in DAGs

Vertex unreliability

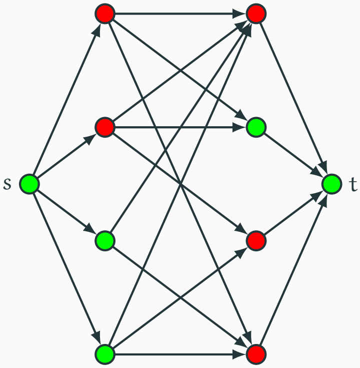


Edge unreliability

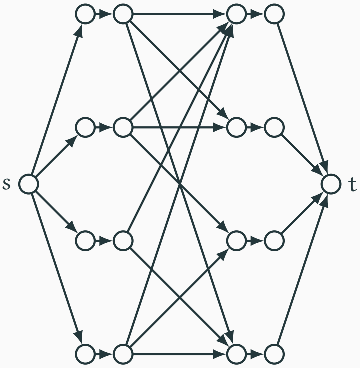


From vertex to edge unreliability in DAGs

Vertex unreliability

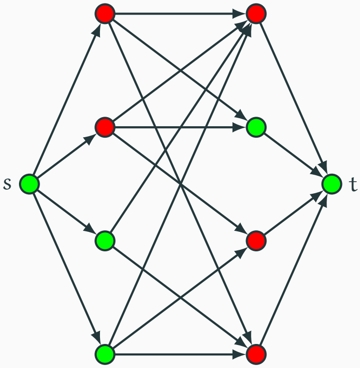


Edge unreliability

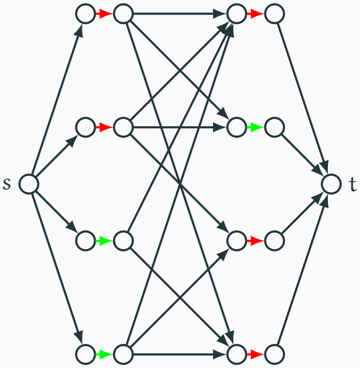


From vertex to edge unreliability in DAGs

Vertex unreliability



Edge unreliability



Black edges have failure probability 0. Colored edges have failure probability 1/2.

Concluding remarks

Open problems

- Faster than $O(m^4 n^6 \max\{m^4, \varepsilon^{-4}\})$?

- Efficient deterministic algorithms?

  This would require derandomise the Karp-Luby algorithm, which has been studied but a polynomial-time algorithm is still elusive.

- Approximation complexity for Two-terminal reliability in directed and undirected graphs?

- Improve the quasi-polynomial time algorithm for #CFG by Gore, Jerrum, Kannan, Sweedyk, and Mahaney (1997).

Thank you!

arXiv:2310.00938

The overall running time

We need union bounds over all edges at various points. Also they accumulate during the DP. Thus set the error $\delta := n^{-1}\min\{m^{-1}, \varepsilon\}$ in Karp-Luby, and we need $O(n\delta^{-2})$ samples each time. (In fact here we use a two-stage algorithm to be more efficient.) To compute the estimator, we need $O(m)$ time, so one Karp-Luby step takes $O(mn\delta^{-2})$ time.

One run of Karp-Luby succeeds with constant probability. To show that reusing samples is fine, we need a union bound over all possible scenarios in the sampling algorithm, which are exponentially many. Thus, we repeat Karp-luby $O(m)$ times to achieve $\exp(-O(m))$ failure probability.

How many samples do we need for each S_i?

On average, we need $\ell := \frac{O(mn\delta^{-2})}{n} = O(m\delta^{-2})$ samples. This in fact suffices.

At each step of the DP, we do approximate counting once and sampling ℓ times, and each sampling step, in turn, calls approximate counting $O(m)$ times.

Thus, the overall running time is

$$n \times m\ell \times m \times O(mn\delta^{-2}) = O(m^4 n^2 \delta^{-4}) = O(m^4 n^6 \max\{m^4, \varepsilon^{-4}\}).$$

The overall running time

We need union bounds over all edges at various points. Also they accumulate during the DP. Thus set the error $\delta := n^{-1}\min\{m^{-1}, \varepsilon\}$ in Karp-Luby, and we need $O(n\delta^{-2})$ samples each time. (In fact here we use a two-stage algorithm to be more efficient.) To compute the estimator, we need $O(m)$ time, so one Karp-Luby step takes $O(mn\delta^{-2})$ time.

One run of Karp-Luby succeeds with constant probability. To show that reusing samples is fine, we need a union bound over all possible scenarios in the sampling algorithm, which are exponentially many. Thus, we repeat Karp-luby $O(m)$ times to achieve $\exp(-O(m))$ failure probability.

How many samples do we need for each S_i?

On average, we need $\ell := \frac{O(mn\delta^{-2})}{n} = O(m\delta^{-2})$ samples. This in fact suffices.

At each step of the DP, we do approximate counting once and sampling ℓ times, and each sampling step, in turn, calls approximate counting $O(m)$ times.

Thus, the overall running time is

$$n \times m\ell \times m \times O(mn\delta^{-2}) = O(m^4 n^2 \delta^{-4}) = O(m^4 n^6 \max\{m^4, \varepsilon^{-4}\}).$$

The overall running time

We need union bounds over all edges at various points. Also they accumulate during the DP. Thus set the error $\delta := n^{-1} \min\{m^{-1}, \varepsilon\}$ in Karp-Luby, and we need $O(n\delta^{-2})$ samples each time. (In fact here we use a two-stage algorithm to be more efficient.) To compute the estimator, we need $O(m)$ time, so one Karp-Luby step takes $O(mn\delta^{-2})$ time.

One run of Karp-Luby succeeds with constant probability. To show that reusing samples is fine, we need a union bound over all possible scenarios in the sampling algorithm, which are exponentially many. Thus, we repeat Karp-luby $O(m)$ times to achieve $\exp(-O(m))$ failure probability.

How many samples do we need for each S_i?

On average, we need $\ell := \frac{O(mn\delta^{-2})}{n} = O(m\delta^{-2})$ samples. This in fact suffices.

At each step of the DP, we do approximate counting once and sampling ℓ times, and each sampling step, in turn, calls approximate counting $O(m)$ times.

Thus, the overall running time is

$$n \times m\ell \times m \times O(mn\delta^{-2}) = O(m^4 n^2 \delta^{-4}) = O(m^4 n^6 \max\{m^4, \varepsilon^{-4}\}).$$

The overall running time

We need union bounds over all edges at various points. Also they accumulate during the DP. Thus set the error $\delta := n^{-1}\min\{m^{-1}, \varepsilon\}$ in Karp-Luby, and we need $O(n\delta^{-2})$ samples each time. (In fact here we use a two-stage algorithm to be more efficient.) To compute the estimator, we need $O(m)$ time, so one Karp-Luby step takes $O(mn\delta^{-2})$ time.

One run of Karp-Luby succeeds with constant probability. To show that reusing samples is fine, we need a union bound over all possible scenarios in the sampling algorithm, which are exponentially many. Thus, we repeat Karp-luby $O(m)$ times to achieve $\exp(-O(m))$ failure probability.

How many samples do we need for each S_i?

On average, we need $\ell := \frac{O(mn\delta^{-2})}{n} = O(m\delta^{-2})$ samples. This in fact suffices.

At each step of the DP, we do approximate counting once and sampling ℓ times, and each sampling step, in turn, calls approximate counting $O(m)$ times.

Thus, the overall running time is

$$n \times m\ell \times m \times O(mn\delta^{-2}) = O(m^4 n^2 \delta^{-4}) = O(m^4 n^6 \max\{m^4, \varepsilon^{-4}\}).$$

The overall running time

We need union bounds over all edges at various points. Also they accumulate during the DP. Thus set the error $\delta := n^{-1}\min\{m^{-1}, \varepsilon\}$ in Karp-Luby, and we need $O(n\delta^{-2})$ samples each time. (In fact here we use a two-stage algorithm to be more efficient.) To compute the estimator, we need $O(m)$ time, so one Karp-Luby step takes $O(mn\delta^{-2})$ time.

One run of Karp-Luby succeeds with constant probability. To show that reusing samples is fine, we need a union bound over all possible scenarios in the sampling algorithm, which are exponentially many. Thus, we repeat Karp-luby $O(m)$ times to achieve $\exp(-O(m))$ failure probability.

How many samples do we need for each S_i?

On average, we need $\ell := \frac{O(mn\delta^{-2})}{n} = O(m\delta^{-2})$ samples. This in fact suffices.

At each step of the DP, we do approximate counting once and sampling ℓ times, and each sampling step, in turn, calls approximate counting $O(m)$ times.

Thus, the overall running time is

$$n \times m\ell \times m \times O(mn\delta^{-2}) = O(m^4 n^2 \delta^{-4}) = O(m^4 n^6 \max\{m^4, \varepsilon^{-4}\}).$$

NP-hardness for $S - t$ reliability in DAGs

We reduce from Minimum directed Steiner tree in DAGs with unit weights, which is **NP**-hard by reducing from Set Covers.

Note that minimum Steiner trees are the extremal configurations for $S - t$ reliability. Thus, if the failure probability of edges is $1 - \exp(-O(n))$, then, drawing from the corresponding distribution, we would only see the minimizers.

Exponentially small success probability can be simulated by gadgets, such as replacing a normal edge by a path (sometimes called a stretching).