A polynomial-time approximation algorithm for all-terminal network reliability

Heng Guo (University of Edinburgh)
Joint with Mark Jerrum (Queen Mary, University of London)

Queen Mary Algorithm Day, Jul 17, 2018

Random sampling strikes back

The complexity of computing quantities

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

a counting analogue of **NP**.

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 (Valiant and Vazirani, 1986);
- any polynomial approximation can be amplified into an ε-approximation with polynomial overhead.

Efficient approximation algorithms do exist! Famous examples include

- the volume of a convex body (Dyer, Frieze, and Kannan, 1991);
- the partition function of ferromagnetic Ising models (Jerrum and Sinclair, 1993);
- the permanent of a non-negative matrix (Jerrum, Sinclair, and Vigoda, 2004).

There are still many open problems in approximate counting!

Network reliability

Network reliability

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

(All-terminal) reliability is the probability such that $G(p)$ is connected.

One may ask the probability of other properties of $G(p)$, such as whether two distinct vertices s and t are connected (s-t reliability), or whether $G(p)$ is acyclic (counting weighted forests), etc.

Network reliability

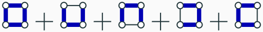
(ALL-TERMINAL) RELIABILITY: The probability that $G(p)$ is connected.

In other words, we want to compute

$$Z_{rel}(G, p) := \sum_{R \subseteq E : (V, R) \text{ is connected}} p^{|E \setminus R|}(1-p)^{|R|}.$$

For example:

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

$Z_{rel}($  $, p) =$ 

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

$$Z_{rel}(G, 1/2) = \frac{|\{R \subseteq E : (V, R) \text{ is connected}\}|}{2^{|E|}}.$$

Computational complexity of reliability

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

Exact evaluation of all-terminal reliability is shown to be #**P**-complete by Jerrum (1981), and independently Provan and Ball (1983).

What about approximation? Open since 80s.

Karger (1999) has given a famous FPRAS for Unreliability (namely $1-Z_{\mathsf{rel}}$). However, approximating $1 - Z_{\mathsf{rel}}$ does not yield a good approximation for Z_{rel} when Z_{rel} is exponentially small.

The Tutte polynomial

For a connected undirected graph $G = (V, E)$,

$$Z_{\mathrm{Tutte}}(G; x, y) := \sum_{R \subseteq E} (x-1)^{\kappa(R)-1} (y-1)^{\kappa(R)+|R|-|V|},$$

where $\kappa(R)$ is the number of connected components of (V, R).

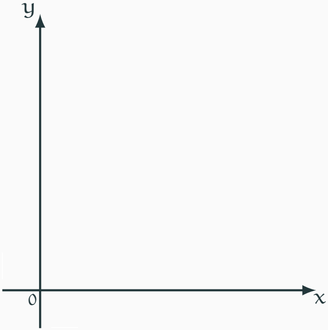
$(1, 1)$: counting spanning trees;

$(1, 1/p)$: network reliability;

$(x, 1)$: counting weighted forests;

$(x-1)(y-1) = 2$:
ferromagnetic Ising model;

and many more ...



The Tutte polynomial

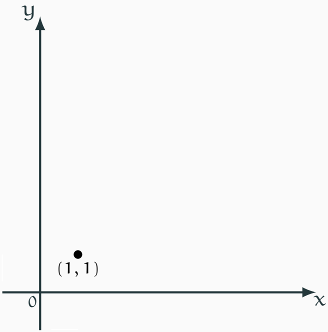
For a connected undirected graph $G = (V, E)$,

$$Z_{\mathrm{Tutte}}(G; x, y) := \sum_{R \subseteq E} (x-1)^{\kappa(R)-1} (y-1)^{\kappa(R)+|R|-|V|},$$

where $\kappa(R)$ is the number of connected components of (V, R).

- $(1, 1)$: counting spanning trees;
- $(1, 1/p)$: network reliability;
- $(x, 1)$: counting weighted forests;
- $(x-1)(y-1) = 2$:
  ferromagnetic Ising model;

and many more ...



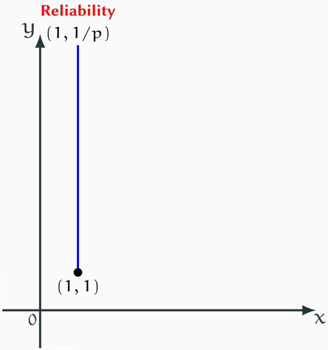
The Tutte polynomial

For a connected undirected graph $G = (V, E)$,

$$Z_{\mathrm{Tutte}}(G; x, y) := \sum_{R \subseteq E} (x-1)^{\kappa(R)-1} (y-1)^{\kappa(R)+|R|-|V|},$$

where $\kappa(R)$ is the number of connected components of (V, R).

- $(1, 1)$: counting spanning trees;
- $(1, 1/p)$: network reliability;
- $(x, 1)$: counting weighted forests;
- $(x-1)(y-1) = 2$:
  ferromagnetic Ising model;
- and many more ...



The Tutte polynomial

For a connected undirected graph $G = (V, E)$,

$$Z_{\mathrm{Tutte}}(G; x, y) := \sum_{R \subseteq E} (x-1)^{\kappa(R)-1} (y-1)^{\kappa(R)+|R|-|V|},$$

where $\kappa(R)$ is the number of connected components of (V, R).

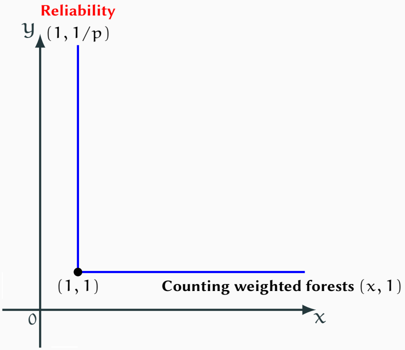
$(1,1)$: counting spanning trees;

$(1, 1/p)$: network reliability;

$(x, 1)$: counting weighted forests;

$(x-1)(y-1) = 2$:
ferromagnetic Ising model;

and many more ...



The Tutte polynomial

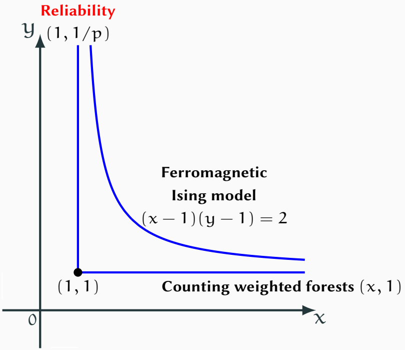
For a connected undirected graph $G = (V, E)$,

$$Z_{\mathrm{Tutte}}(G; x, y) := \sum_{R \subseteq E} (x-1)^{\kappa(R)-1}(y-1)^{\kappa(R)+|R|-|V|},$$

where $\kappa(R)$ is the number of connected components of (V, R).

- $(1, 1)$: counting spanning trees;
- $(1, 1/p)$: network reliability;
- $(x, 1)$: counting weighted forests;
- $(x-1)(y-1) = 2$: ferromagnetic Ising model;

and many more ...



The Tutte polynomial

For a connected undirected graph $G = (V, E)$,

$$Z_{\mathrm{Tutte}}(G; x, y) := \sum_{R \subseteq E} (x-1)^{\kappa(R)-1} (y-1)^{\kappa(R)+|R|-|V|},$$

where $\kappa(R)$ is the number of connected components of (V, R).

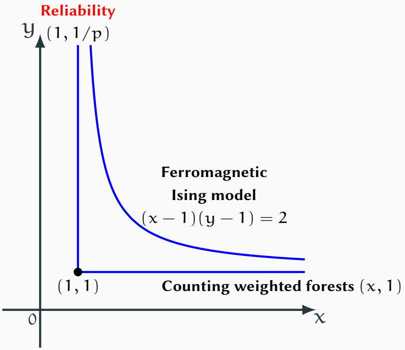
$(1, 1)$: counting spanning trees;

$(1, 1/p)$: network reliability;

$(x, 1)$: counting weighted forests;

$(x-1)(y-1) = 2$:
ferromagnetic Ising model;

and many more ...



Main result

Let $m := |E|$ and $n := |V|$.

Theorem (G. and Jerrum, 2018)

There is a randomised algorithm approximating Z_{rel} *within multiplicative factor* $(1 \pm \varepsilon)$*, with expected running time* $O\left(\varepsilon^{-2}(1-p)^{-3}m^2n^3\right)$.

Theorem (G. and Jerrum, 2018)

There is an exact sampler to draw (edge-weighted) connected subgraphs with expected running time $O((1-p)^{-1}m^2n)$.

Spoiler: sampling can be done in $O(mn)$ time and approximate counting in $O(mn^2 \log n)$ time (G. and He, 2018+).

Main result

Let $m := |E|$ and $n := |V|$.

Theorem (G. and Jerrum, 2018)

There is a randomised algorithm approximating Z_{rel} *within multiplicative factor* $(1 \pm \varepsilon)$*, with expected running time* $O\left(\varepsilon^{-2}(1-p)^{-3}m^2n^3\right)$.

Theorem (G. and Jerrum, 2018)

There is an exact sampler to draw (edge-weighted) connected subgraphs with expected running time $O((1-p)^{-1}m^2n)$.

Spoiler: sampling can be done in $O(mn)$ time and approximate counting in $O(mn^2 \log n)$ time (G. and He, 2018+).

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 $(R_i)_{i \in [k]}$ of $G(p)$.
2. Let

$$\widetilde{Z} := \frac{1}{k} \sum_{i \in [k]} \mathbb{1}_{\mathrm{conn}}(R_i),$$

where $\mathbb{1}_{\mathrm{conn}}(R)$ is the indicator variable of (V, R) being connected.

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 R_i.

In that case, the variance of $\mathbb{1}_{\mathrm{conn}}(R)$ 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 $(R_i)_{i \in [k]}$ of $G(p)$.
2. Let

$$\widetilde{Z} := \frac{1}{k} \sum_{i \in [k]} \mathbb{1}_{conn}(R_i),$$

where $\mathbb{1}_{conn}(R)$ is the indicator variable of (V, R) being connected.

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 R_i.

In that case, the variance of $\mathbb{1}_{conn}(R)$ 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 $(R_i)_{i \in [k]}$ of $G(p)$.
2. Let

$$\widetilde{Z} := \frac{1}{k} \sum_{i \in [k]} \mathbb{1}_{conn}(R_i),$$

   where $\mathbb{1}_{conn}(R)$ is the indicator variable of (V, R) being connected.

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 R_i.

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

Unreliability

Nonetheless, naive Monte Carlo (NMC) is the basic building block of the FPRAS by Karger (1999) for UNRELIABILITY (namely $1 - Z_{rel}$).

Karger's algorithm has been subsequently refined by Harris and Srinivasan (2014), Karger (2016, 2017).

Karger (2017) is a recursive algorithm using NMC running in $O(n^{2.87})$. Nonetheless, these ideas does not seem to help with approximating Z_{rel}.

Unreliability

Nonetheless, naive Monte Carlo (NMC) is the basic building block of the FPRAS by Karger (1999) for UNRELIABILITY (namely $1 - Z_{rel}$).

Karger's algorithm has been subsequently refined by Harris and Srinivasan (2014), Karger (2016, 2017).

Karger (2017) is a recursive algorithm using NMC running in $O(n^{2.87})$. Nonetheless, these ideas does not seem to help with approximating Z_{rel}.

Unreliability

Nonetheless, naive Monte Carlo (NMC) is the basic building block of the FPRAS by Karger (1999) for UNRELIABILITY (namely $1 - Z_{rel}$).

Karger's algorithm has been subsequently refined by Harris and Srinivasan (2014), Karger (2016, 2017).

Karger (2017) is a recursive algorithm using NMC running in $O(n^{2.87})$. Nonetheless, these ideas does not seem to help with approximating Z_{rel}.

Reducing counting to sampling

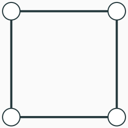
Let $\pi_G(\cdot)$ be the distribution of $G(p)$, conditioned on being connected.

We can approximate Z_{rel} using an oracle drawing from π_G.

Reducing counting to sampling

Let $\pi_G(\cdot)$ be the distribution of $G(p)$, conditioned on being connected.

We can approximate Z_{rel} using an oracle drawing from π_G.

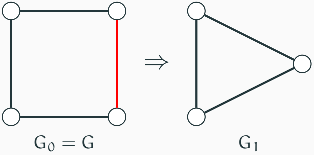


$G_0 = G$

Reducing counting to sampling

Let $\pi_G(\cdot)$ be the distribution of $G(p)$, conditioned on being connected.

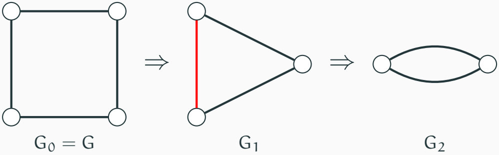
We can approximate Z_{rel} using an oracle drawing from π_G.



Reducing counting to sampling

Let $\pi_G(\cdot)$ be the distribution of $G(p)$, conditioned on being connected.

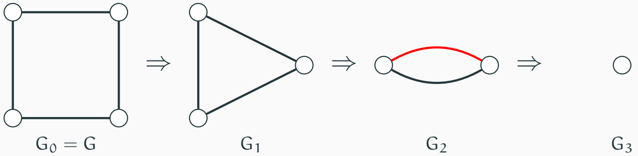
We can approximate Z_{rel} using an oracle drawing from π_G.



Reducing counting to sampling

Let $\pi_G(\cdot)$ be the distribution of $G(p)$, conditioned on being connected.

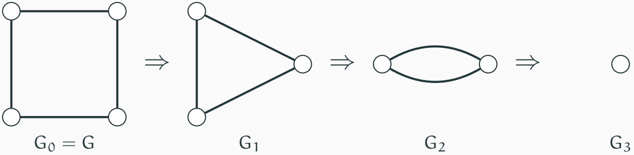
We can approximate Z_{rel} using an oracle drawing from π_G.



Reducing counting to sampling

Let $\pi_G(\cdot)$ be the distribution of $G(p)$, conditioned on being connected.

We can approximate Z_{rel} using an oracle drawing from π_G.



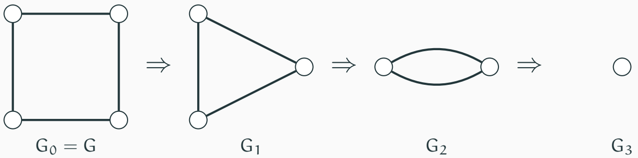
Rewrite

$$Z_{rel}(G) = \frac{Z_{rel}(G_0)}{Z_{rel}(G_1)} \cdot \frac{Z_{rel}(G_1)}{Z_{rel}(G_2)} \cdot \frac{Z_{rel}(G_2)}{Z_{rel}(G_3)} \cdot Z_{rel}(G_3).$$

Reducing counting to sampling

Let $\pi_G(\cdot)$ be the distribution of $G(p)$, conditioned on being connected.

We can approximate Z_{rel} using an oracle drawing from π_G.



To estimate $\frac{Z_{rel}(G_i)}{Z_{rel}(G_{i+1})}$, draw $C \sim \pi_{G_{i+1}}(\cdot)$ and let

$$C' := \begin{cases} C & \text{with prob. } p; \\ C \cup \{e\} & \text{otherwise,} \end{cases} \quad \text{and} \quad X := \mathbb{1}_{\mathrm{conn},\, G_i}(C').$$

Then $\mathbb{E}\, X = \frac{Z_{rel}(G_i)}{Z_{rel}(G_{i+1})}$ and its variance is bounded by a constant.

Markov chain Monte Carlo

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

There is a natural Markov chain converging to $\pi_G(\cdot)$:

1. Let $C_0 = E$.
2. Given C_t, randomly pick an edge $e \in E$.

   If $C_t \setminus \{e\}$ is disconnected then $C_{t+1} = C_t$. Otherwise,

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

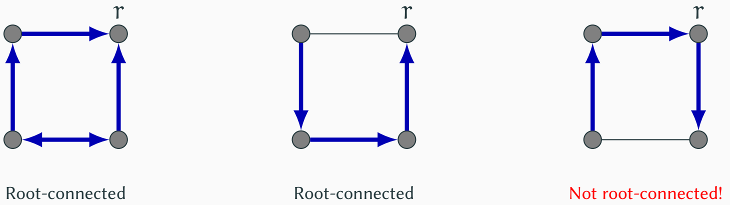
Unfortunately, no polynomial upper bound (nor exponential lower bound) is known about its mixing time (rate of convergence).

A surprising equivalence

(and an alternative way to sampling)

Reachability

We say a directed graph D with root r is *root-connected* if all vertices can reach r.



REACHABILITY: in a directed graph $D = (V, A)$ with root r, what's the probability that $D(p)$ is root-connected?

$$Z_{reach}(D, p) := \sum_{R \subseteq A : (V, R) \text{ is root-connected}} p^{|A \setminus R|}(1-p)^{|R|}.$$

A surprising equivalence

Ball (1980) showed that for any undirected graph $G = (V, E)$,

$$Z_{rel}(G, p) = Z_{reach}(\overrightarrow{G}, p),$$

where $\overrightarrow{G}$ is the directed graph obtained by replacing every $e \in E$ with a pair of anti-parallel arcs. (Called bi-directed).

Thus we just need to approximate REACHABILITY in bi-directed graphs.

A coupling proof

We have an alternative coupling proof of Ball's equivalence:

There is a coupling $\mathcal{C}$ under which

$$G(p) \text{ is connected} \Leftrightarrow \overrightarrow{G}(p) \text{ is root-connected.}$$

Explore G and $\overrightarrow{G}$ like a BFS, starting from r. Reveal $\overrightarrow{G}(p)$ and $G(p)$ as the process proceeds. Couple the arc going towards the current vertex in $\overrightarrow{G}(p)$ with the corresponding edge in $G(p)$.

When both exploration processes end, the sets of vertices that can reach r are exactly the same.

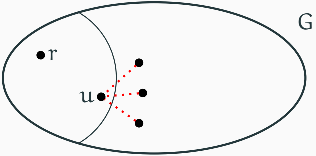
A coupling proof

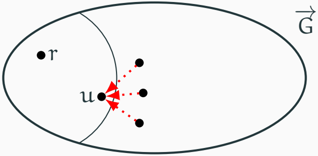
We have an alternative coupling proof of Ball's equivalence:

There is a coupling $\mathcal{C}$ under which

$$G(p) \text{ is connected} \Leftrightarrow \overrightarrow{G}(p) \text{ is root-connected.}$$

Explore G and $\overrightarrow{G}$ like a BFS, starting from r. Reveal $\overrightarrow{G}(p)$ and $G(p)$ as the process proceeds. Couple the arc going towards the current vertex in $\overrightarrow{G}(p)$ with the corresponding edge in $G(p)$.





When both exploration processes end, the sets of vertices that can reach r are exactly the same.

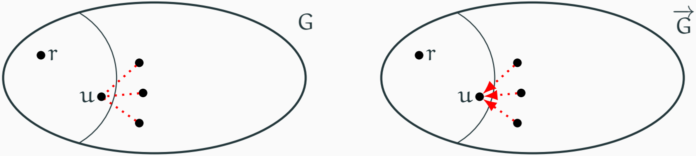
A coupling proof

We have an alternative coupling proof of Ball's equivalence:

There is a coupling $\mathcal{C}$ under which

$$G(p) \text{ is connected} \Leftrightarrow \overrightarrow{G}(p) \text{ is root-connected.}$$

Explore G and $\overrightarrow{G}$ like a BFS, starting from r. Reveal $\overrightarrow{G}(p)$ and $G(p)$ as the process proceeds. Couple the arc going towards the current vertex in $\overrightarrow{G}(p)$ with the corresponding edge in $G(p)$.



When both exploration processes end, the sets of vertices that can reach r are exactly the same.

Sample in at least two ways

Goal: sample uniform (or edge-weighted) root-connected subgraphs.

It is straightforward to apply Markov chains, but again analysing the mixing time does not seem easy.

Sample in at least two ways

Goal: sample uniform (or edge-weighted) root-connected subgraphs.

It is straightforward to apply Markov chains, but again analysing the mixing time does not seem easy.

Sample in at least two ways

Goal: sample uniform (or edge-weighted) root-connected subgraphs.

It is straightforward to apply Markov chains, but again analysing the mixing time does not seem easy.

However, another hope is to do (exact) sampling in expected polynomial-time, based on rejections.

Cluster-popping

Goal: sample uniform (or edge-weighted) root-connected subgraphs.

Gorodezky and Pak (2014) proposed the "cluster-popping" algorithm:
(Cluster: a subset of vertices not including r and with no arc going out.)

1. Let R be a subset of arcs by choosing each arc e with probability $1 - p$ independently.
2. **While** there is at least one cluster in (V, R):
   - Let $C_1, \ldots, C_k$ be all minimal clusters in (V, R), and $C = \bigcup_{i=1}^{k} C_i$.
   - Re-randomize all arcs whose heads are in C to get a new R.

Gorodezky and Pak (2014) showed that this algorithm draws from the correct distribution, and they also conjectured that cluster-popping runs in expected polynomial time in bi-directed graphs.

Cluster-popping

Goal: sample uniform (or edge-weighted) root-connected subgraphs.

Gorodezky and Pak (2014) proposed the "cluster-popping" algorithm:
(Cluster: a subset of vertices not including r and with no arc going out.)

1. Let R be a subset of arcs by choosing each arc e with probability $1 - p$ independently.
2. While there is at least one cluster in (V, R):
   - Let $C_1, \ldots, C_k$ be all minimal clusters in (V, R), and $C = \bigcup_{i=1}^{k} C_i$.
   - Re-randomize all arcs whose heads are in C to get a new R.

Gorodezky and Pak (2014) showed that this algorithm draws from the correct distribution, and they also conjectured that cluster-popping runs in expected polynomial time in bi-directed graphs.

Cluster-popping

Goal: sample uniform (or edge-weighted) root-connected subgraphs.

Gorodezky and Pak (2014) proposed the "cluster-popping" algorithm:
(Cluster: a subset of vertices not including r and with no arc going out.)

1. Let R be a subset of arcs by choosing each arc e with probability $1 - p$ independently.
2. While there is at least one cluster in (V, R):
    - Let $C_1, \dots, C_k$ be all minimal clusters in (V, R), and $C = \bigcup_{i=1}^{k} C_i$.
    - Re-randomize all arcs whose heads are in C to get a new R.

Gorodezky and Pak (2014) showed that this algorithm draws from the correct distribution, and they also conjectured that cluster-popping runs in expected polynomial time in bi-directed graphs.

Cluster-popping

Goal: sample uniform (or edge-weighted) root-connected subgraphs.

Gorodezky and Pak (2014) proposed the "cluster-popping" algorithm:
(Cluster: a subset of vertices not including r and with no arc going out.)

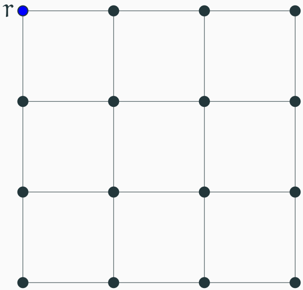
1. Let R be a subset of arcs by choosing each arc e with probability $1 - p$ independently.
2. While there is at least one cluster in (V, R):
   - Let $C_1, \dots, C_k$ be all minimal clusters in (V, R), and $C = \bigcup_{i=1}^{k} C_i$.
   - Re-randomize all arcs whose heads are in C to get a new R.

Gorodezky and Pak (2014) showed that this algorithm draws from the correct distribution, and they also conjectured that cluster-popping runs in expected polynomial time in bi-directed graphs.

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

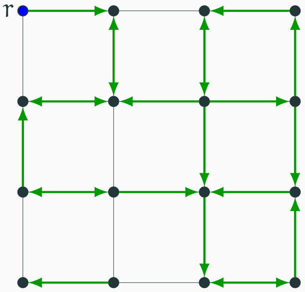
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

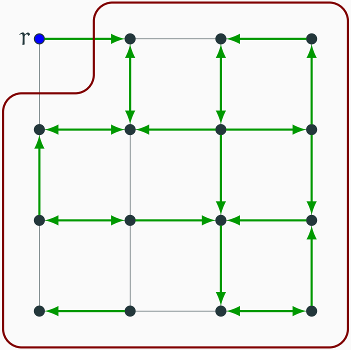
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

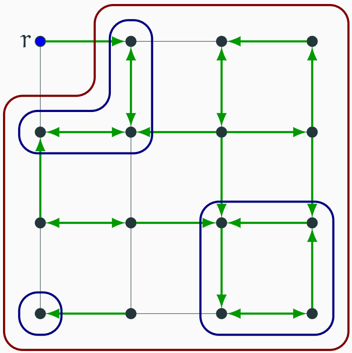
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

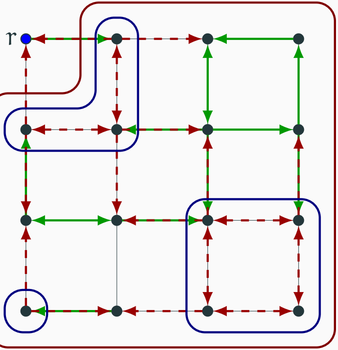
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

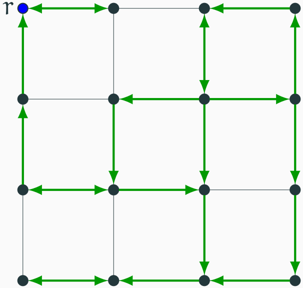
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

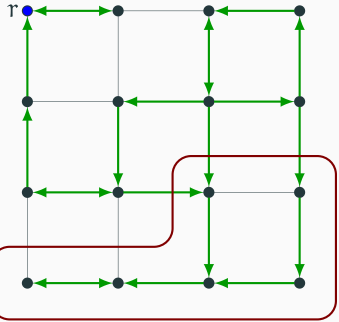
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

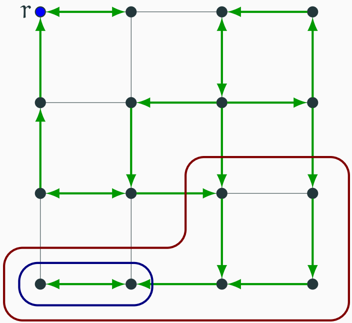
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

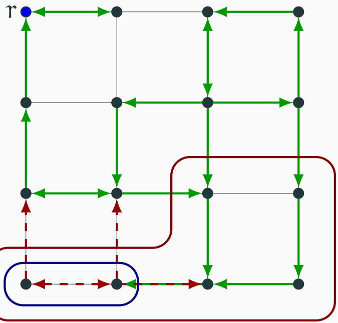
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

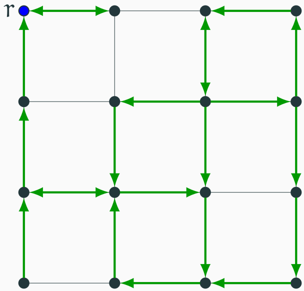
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

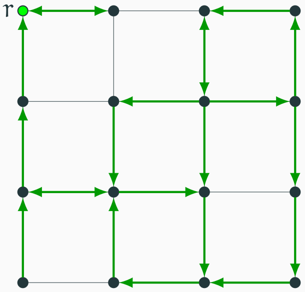
(Cluster: a subset of vertices not including r and with no arc going out.)



An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



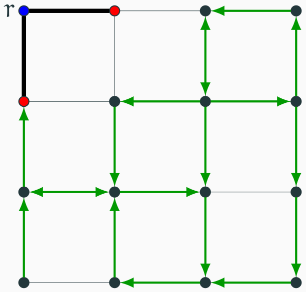
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



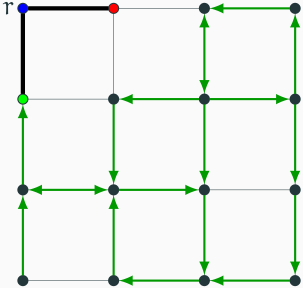
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



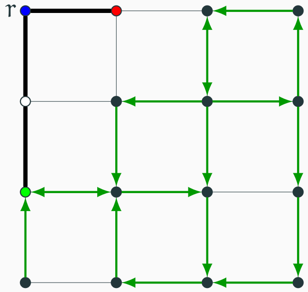
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



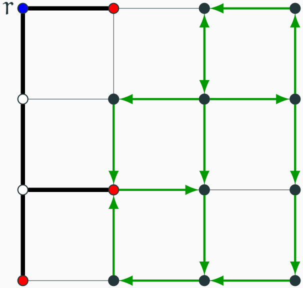
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



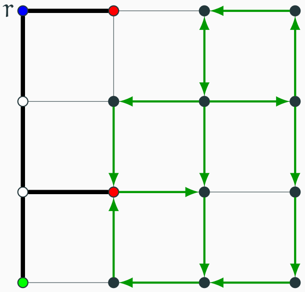
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



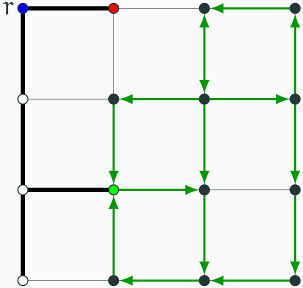
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



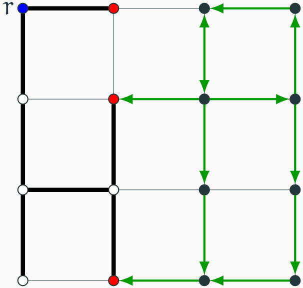
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



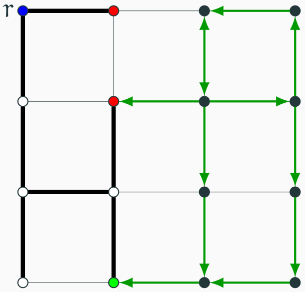
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



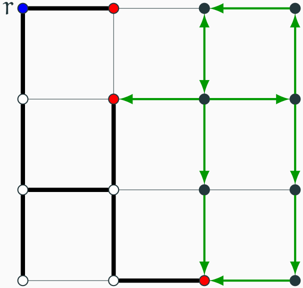
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



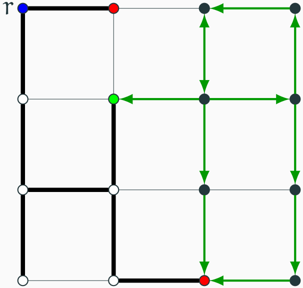
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



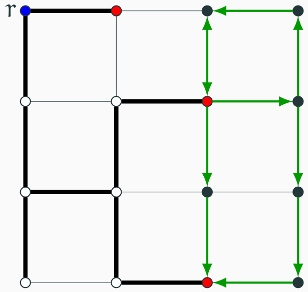
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



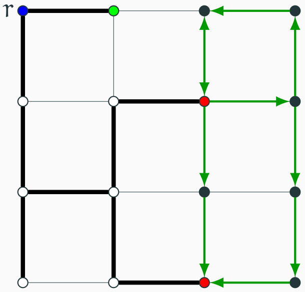
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



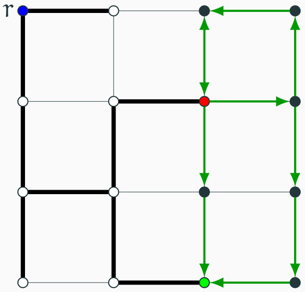
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



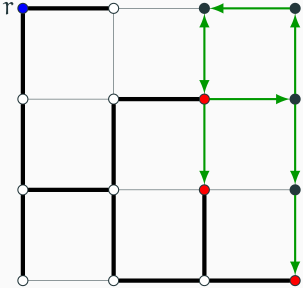
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including $\mathfrak{r}$ and with no arc going out.)



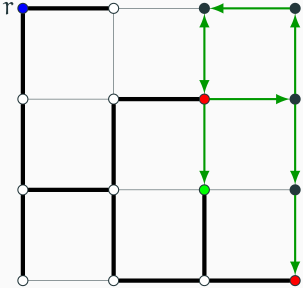
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



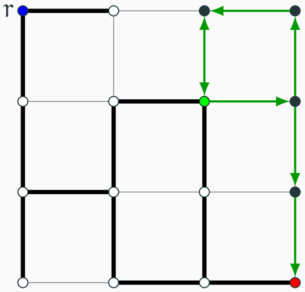
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



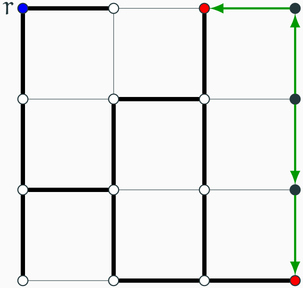
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including $\mathfrak{r}$ and with no arc going out.)



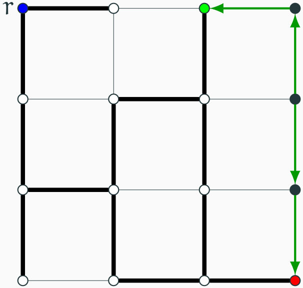
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



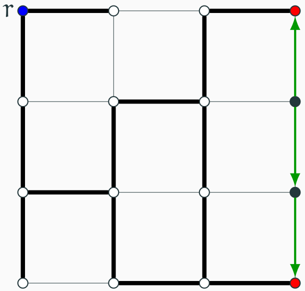
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



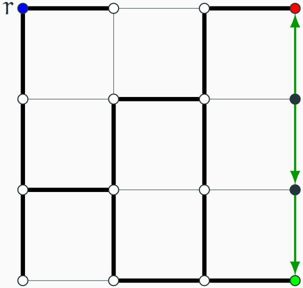
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



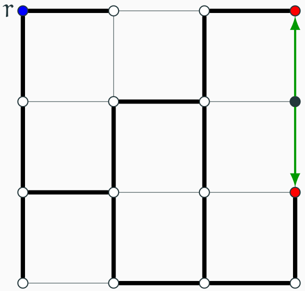
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



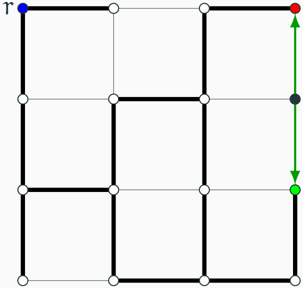
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



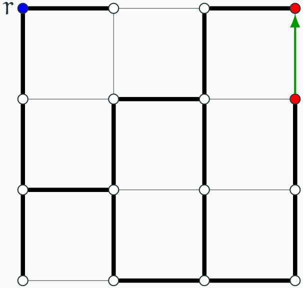
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



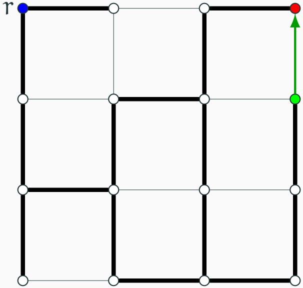
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



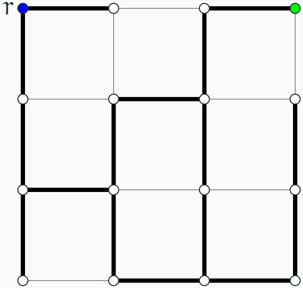
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



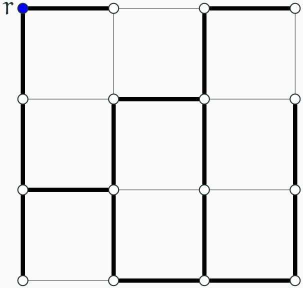
Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

An example run

Cluster-popping: repeatedly resample minimal clusters until none.

(Cluster: a subset of vertices not including r and with no arc going out.)



Mapping back to connected subgraph.

(Exploration order: left to right, bottom to top)

Partial rejection sampling

(A more general perspective for cluster-popping)

Partial rejection sampling

Cluster-popping is a special case of PARTIAL REJECTION SAMPLING framework (G., Jerrum, and Liu, 2017).

The goal is to sample from a product distribution, conditioned on a number of "bad" events not happening.

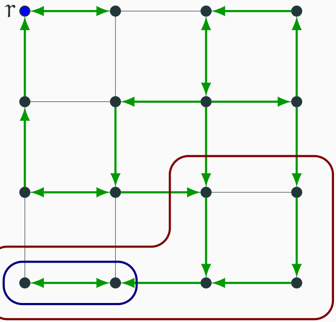
Rejection sampling throws away all variables.

Instead, we want to recycle some randomness while resampling the "bad" events (and hopefully not too much more).

Partial rejection sampling

Cluster-popping under partial rejection sampling:

Arcs are variables. Minimal clusters are "bad" events.

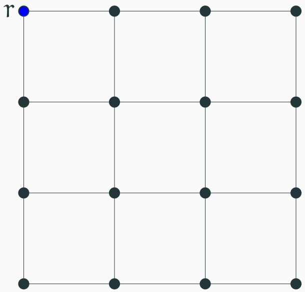


There can be exponentially many bad events.

Extremal instances

An instance is called extremal (in the sense of Shearer (1985) regarding non-uniform Lovász Local Lemma):

if any two “bad” events A_i and A_j are either independent or disjoint.

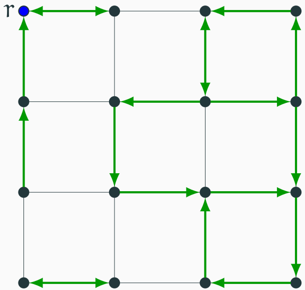


If the instance is extremal, then eliminating precisely the “bad” events in each iteration yields the correct distribution once the process halts (GJL’17)!

Extremal instances

An instance is called extremal (in the sense of Shearer (1985) regarding non-uniform Lovász Local Lemma):

if any two "bad" events A_i and A_j are either independent or disjoint.

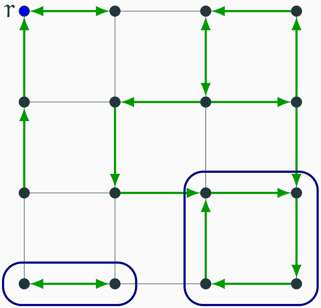


If the instance is extremal, then eliminating precisely the "bad" events in each iteration yields the correct distribution once the process halts (GJL'17)!

Extremal instances

An instance is called extremal (in the sense of Shearer (1985) regarding non-uniform Lovász Local Lemma):

if any two "bad" events A_i and A_j are either independent or disjoint.

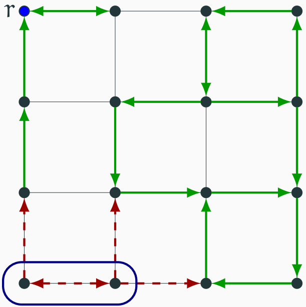


If the instance is extremal, then eliminating precisely the "bad" events in each iteration yields the correct distribution once the process halts (GJL'17)!

Extremal instances

An instance is called extremal (in the sense of Shearer (1985) regarding non-uniform Lovász Local Lemma):

if any two "bad" events A_i and A_j are either independent or disjoint.

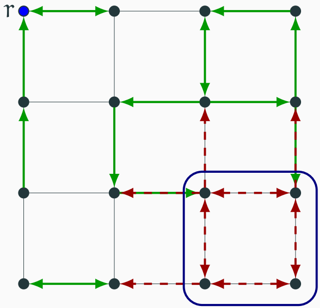


If the instance is extremal, then eliminating precisely the "bad" events in each iteration yields the correct distribution once the process halts (GJL'17)!

Extremal instances

An instance is called extremal (in the sense of Shearer (1985) regarding non-uniform Lovász Local Lemma):

if any two "bad" events A_i and A_j are either independent or disjoint.



If the instance is extremal, then eliminating precisely the "bad" events in each iteration yields the correct distribution once the process halts (GJL'17)!

Resampling table

Associate an infinite stack $X_{i,0}, X_{i,1}, \ldots$ to each random variable X_i. When we need to resample, draw the next value in the stack.

X_1	$X_{1,0}$	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	$\cdots$
X_2	$X_{2,0}$	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$\cdots$
X_3	$X_{3,0}$	$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$	$\cdots$
X_4	$X_{4,0}$	$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$	$\cdots$

Resampling table

Associate an infinite stack $X_{i,0}, X_{i,1}, \ldots$ to each random variable X_i. When we need to resample, draw the next value in the stack.

X_1	$X_{1,0}$	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	$\cdots$
X_2	$X_{2,0}$	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$\cdots$
X_3	$X_{3,0}$	$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$	$\cdots$
X_4	$X_{4,0}$	$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$	$\cdots$

Resampling table

Associate an infinite stack $X_{i,0}, X_{i,1}, \ldots$ to each random variable X_i. When we need to resample, draw the next value in the stack.

X_1	$X_{1,0}$	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	$\cdots$
X_2	$X_{2,0}$	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$\cdots$
X_3	$X_{3,0}$	$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$	$\cdots$
X_4	$X_{4,0}$	$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$	$\cdots$

Resampling table

Associate an infinite stack $X_{i,0}, X_{i,1}, \ldots$ to each random variable X_i. When we need to resample, draw the next value in the stack.

X_1	$X_{1,0}$	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	$\cdots$
X_2	$X_{2,0}$	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$\cdots$
X_3	$X_{3,0}$	$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$	$\cdots$
X_4	$X_{4,0}$	$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$	$\cdots$

Resampling table

Associate an infinite stack $X_{i,0}, X_{i,1}, \ldots$ to each random variable X_i. When we need to resample, draw the next value in the stack.

X_1	$X_{1,0}$	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	$\cdots$
X_2	$X_{2,0}$	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$\cdots$
X_3	$X_{3,0}$	$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$	$\cdots$
X_4	$X_{4,0}$	$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$	$\cdots$

Resampling table

Associate an infinite stack $X_{i,0}, X_{i,1}, \ldots$ to each random variable X_i. When we need to resample, draw the next value in the stack.

X_1	$X_{1,0}$	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	$\cdots$
X_2	$X_{2,0}$	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$\cdots$
X_3	$X_{3,0}$	$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$	$\cdots$
X_4	$X_{4,0}$	$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$	$\cdots$

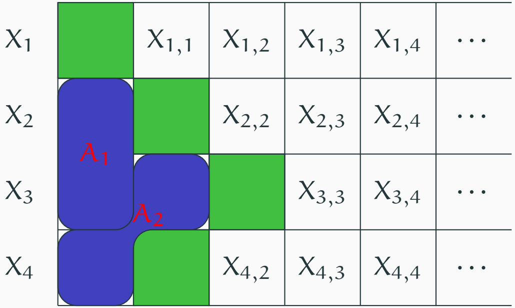
Change the future, not the history

For extremal instances, replacing a perfect assignment with another one will not change the resampling history!

X_1	$X_{1,0}$	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	$\cdots$
X_2	$X_{2,0}$	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$\cdots$
X_3	$X_{3,0}$	$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$	$\cdots$
X_4	$X_{4,0}$	$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$	$\cdots$

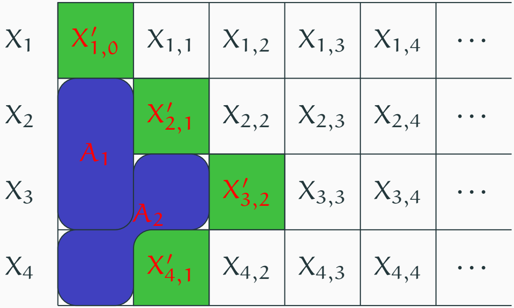
Change the future, not the history

For extremal instances, replacing a perfect assignment with another one will not change the resampling history!



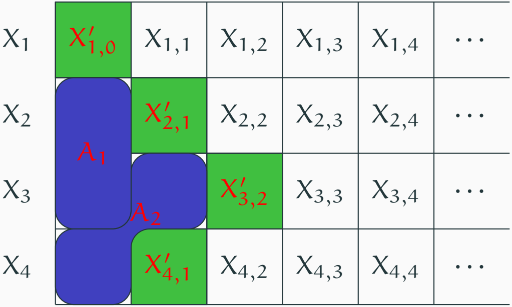
Change the future, not the history

For extremal instances, replacing a perfect assignment with another one will not change the resampling history!



Change the future, not the history

For extremal instances, replacing a perfect assignment with another one will not change the resampling history!

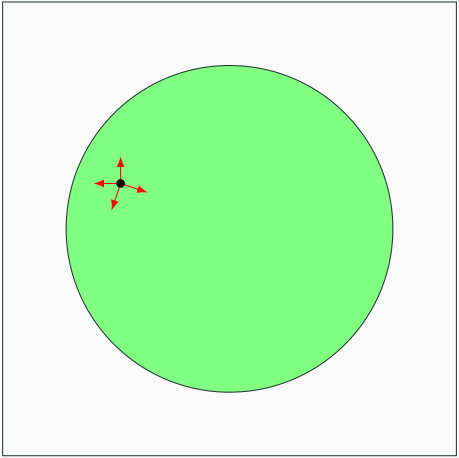


For any two outputs σ and τ, there is a bijection between trajectories leading to σ and τ.

Partial Rejection Sampling vs Markov chains

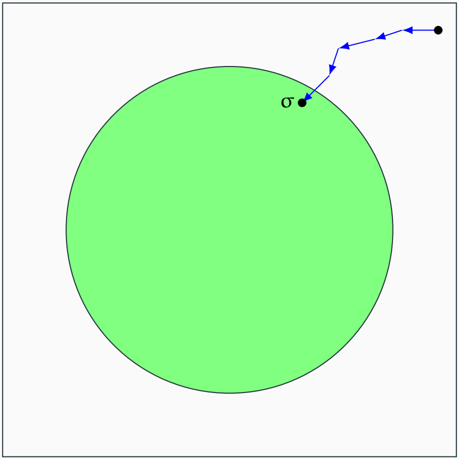
Markov chain is a random walk in the solution space.

(The solution space has to be connected,
and the mixing time is not easy to analyze.)



Partial Rejection Sampling vs Markov chains

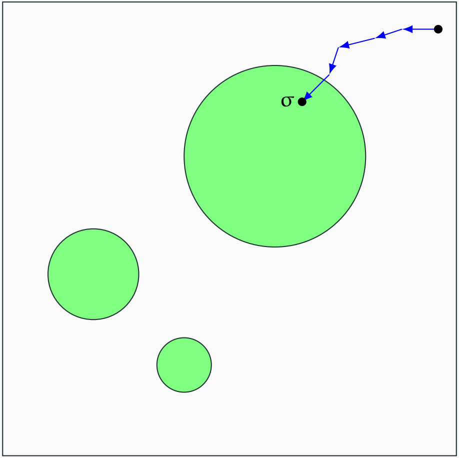
PRS is a local search on the whole space.



Partial Rejection Sampling vs Markov chains

PRS is a local search on the whole space.

(Ergodicity is not an issue.)

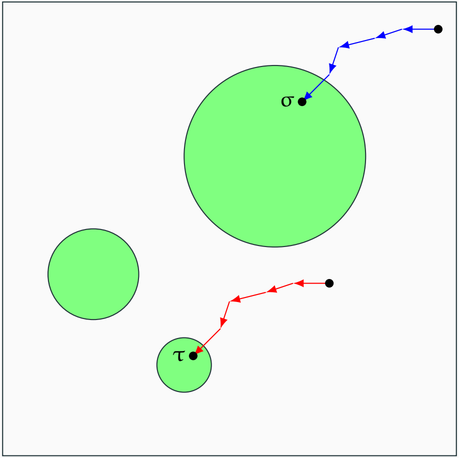


Partial Rejection Sampling vs Markov chains

PRS is a local search on the whole space.

(Correctness guaranteed by the bijection.

Exact formula for its running time on extremal instances.)



Run-time analysis

Theorem (G., Jerrum, and Liu, 2017)

Under Shearer's condition, for extremal instances,

$$\mathbb{E}\,T = \frac{\text{total weight of one-flaw assignments}}{\text{total weight of perfect assignments}}.$$

(Shearer (1985) has shown a sufficient condition to guarantee the existence of one perfect assignment, which is optimal for Lovász Local Lemma.)

The upper bound is shown by Kolipaka and Szegedy (2011).

Back to cluster-popping

Cluster-popping: repeatedly resample minimal clusters.

Let Ω_k be the set of subgraphs with k minimal clusters, and

$$Z_k := \sum_{S \in \Omega_k} p^{|E \setminus S|}(1-p)^{|S|}. \qquad \text{Then, } \mathbb{E}\, T = \frac{Z_1}{Z_0}.$$

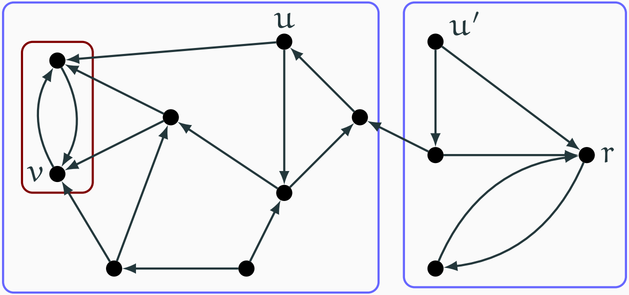
Lemma (G. and Jerrum, 2018)

For bi-directed graphs, $Z_1 \leqslant \frac{p}{1-p} \cdot mnZ_0$.

We show this by designing an injective mapping $\Omega_1 \to \Omega_0 \times V \times E$.

Injective mapping

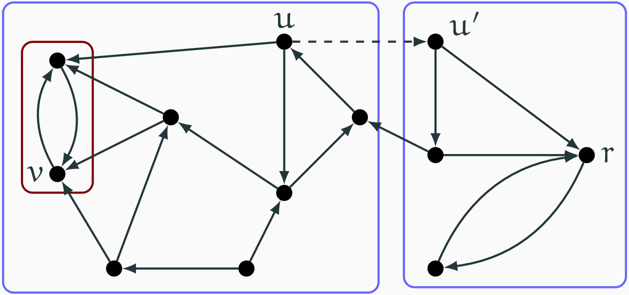
Given $R \in \Omega_1$, we map it to $R_0 \in \Omega_0$ by "repairing" the unique minimal cluster.



Conversely, given $R_0 \in \Omega_0$, (u, u') and v, we can recover $R \in \Omega_1$.

Injective mapping

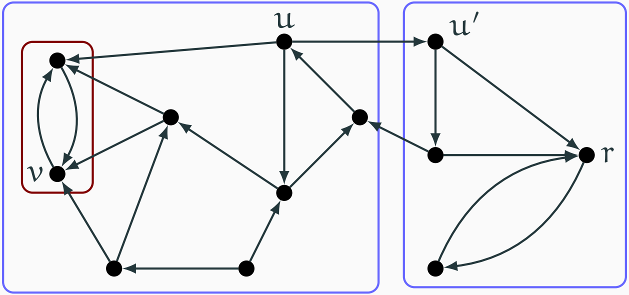
Given $R \in \Omega_1$, we map it to $R_0 \in \Omega_0$ by "repairing" the unique minimal cluster.



Conversely, given $R_0 \in \Omega_0$, (u, u') and v, we can recover $R \in \Omega_1$.

Injective mapping

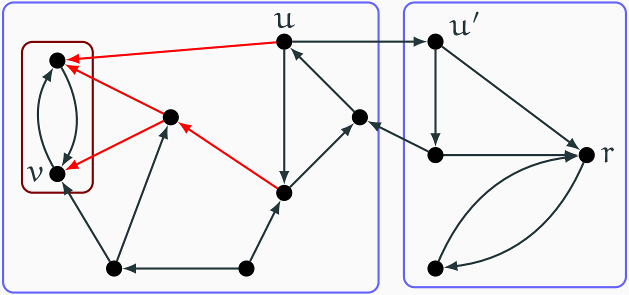
Given $R \in \Omega_1$, we map it to $R_0 \in \Omega_0$ by "repairing" the unique minimal cluster.



Conversely, given $R_0 \in \Omega_0$, (u, u') and v, we can recover $R \in \Omega_1$.

Injective mapping

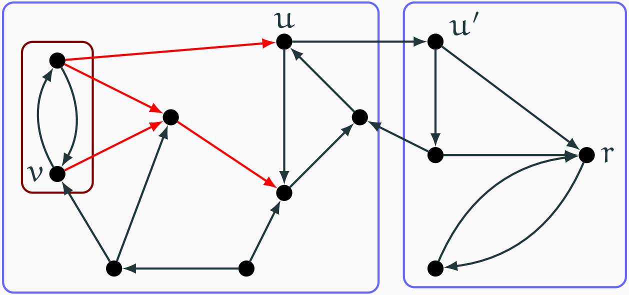
Given $R \in \Omega_1$, we map it to $R_0 \in \Omega_0$ by "repairing" the unique minimal cluster.



Conversely, given $R_0 \in \Omega_0$, (u, u') and v, we can recover $R \in \Omega_1$.

Injective mapping

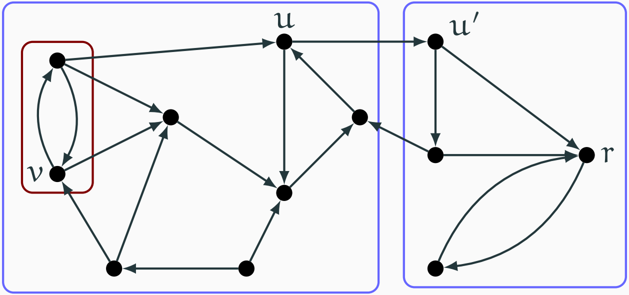
Given $R \in \Omega_1$, we map it to $R_0 \in \Omega_0$ by "repairing" the unique minimal cluster.



Conversely, given $R_0 \in \Omega_0$, (u, u') and v, we can recover $R \in \Omega_1$.

Injective mapping

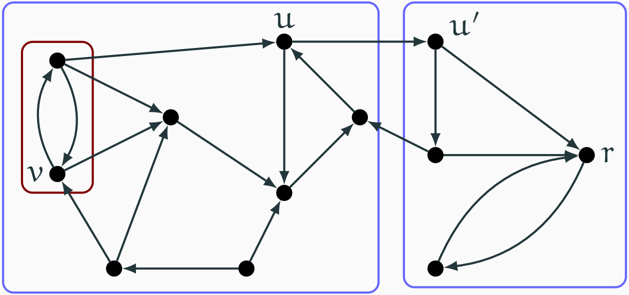
Given $R \in \Omega_1$, we map it to $R_0 \in \Omega_0$ by "repairing" the unique minimal cluster.



Conversely, given $R_0 \in \Omega_0$, (u, u') and v, we can recover $R \in \Omega_1$.

Injective mapping

Given $R \in \Omega_1$, we map it to $R_0 \in \Omega_0$ by "repairing" the unique minimal cluster.



Conversely, given $R_0 \in \Omega_0$, (u, u') and v, we can recover $R \in \Omega_1$.

Recap for reliability

Approximate $Z_{rel}(G)$ via a sequence of contractions $G_0, \ldots, G_{n-1}$, and estimate each $\frac{Z_{rel}(G_i)}{Z_{rel}(G_{i+1})}$ using the following sampling oracle:

1. run cluster-popping to sample a root-connected subgraph in $\overrightarrow{G}$;
2. use the coupling to get a random connected subgraph.

To bound the running time of cluster-popping, we use a result of (GJL'17) and design an injective mapping.

Counting connected subgraphs of fixed size

Let S_t be the set of connected subgraph of size t where $n-1 \leqslant t \leqslant m$ and $N_t = |S_t|$. Then a result of Huh and Katz (2012) implies that the sequence $(N_t)_t$ is log-concave, namely,

$$N_{t-1}N_{t+1} \leqslant N_t^2.$$

(The complements of connected subgraphs are independent sets of the co-graphic matroid associated with G, and co-graphic matroids are representable. So HK'12 applies. Similar log-concavity in general matroids is resolved by Adiprasito, Huh, and Katz (2015).)

Given the sampler for connected subgraphs and log-concavity, we can set $p = \frac{N_t}{N_{t-1}+N_t}$ so that subgraphs in S_t show up frequently enough. There is a standard approach (Jerrum and Sinclair, 1989) to estimate each individual N_t.

Other examples of PRS

Extremal instances:

- Uniform spanning trees — cycle-popping (Wilson, 1996)
- Uniform sink-free orientations — sink-popping (Bubley and Dyer, 1997) (Cohn, Pemantle, and Propp, 2002)
- Uniform bases of bicircular matroids (G. and Jerrum, 2018+)

General instances (G., Jerrum, and Liu, 2017):

- Weighted independent set (Hardcore gas model)
- Hard disks / hard spheres model (G. and Jerrum, 2018)
- Solutions to k-CNF formulas with bounded variable degrees

Results for general instances are far from optimal.

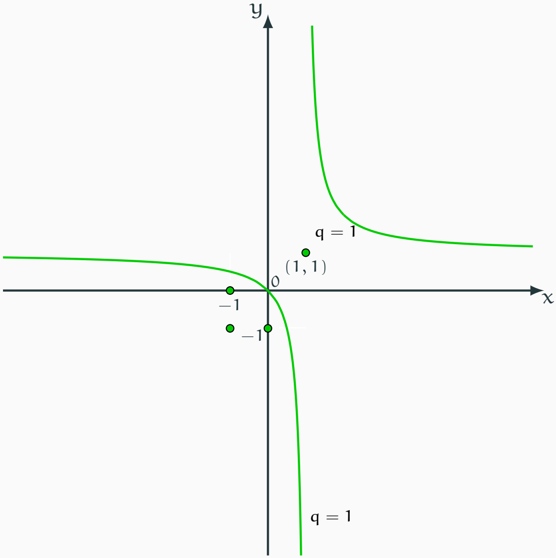
Can we do this for colourings?

Concluding remarks

Approximating the Tutte polynomial

$q = (x-1)(y-1)$

Poly-time



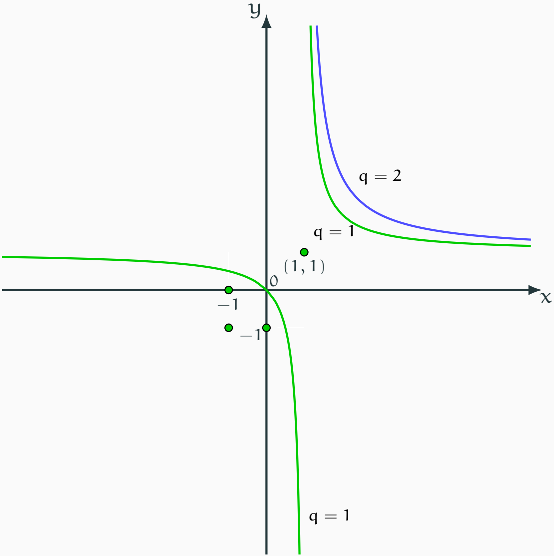
Ref:
Jaeger, Vertigan, and Welsh (1990);
Jerrum and Sinclair (1993);
Goldberg and Jerrum (2008, 2012, 2014)

Approximating the Tutte polynomial

$q = (x-1)(y-1)$

Poly-time

FPRAS



Ref:
Jaeger, Vertigan, and Welsh (1990);
Jerrum and Sinclair (1993);
Goldberg and Jerrum (2008, 2012, 2014)

Approximating the Tutte polynomial

$q = (x-1)(y-1)$

Poly-time

FPRAS

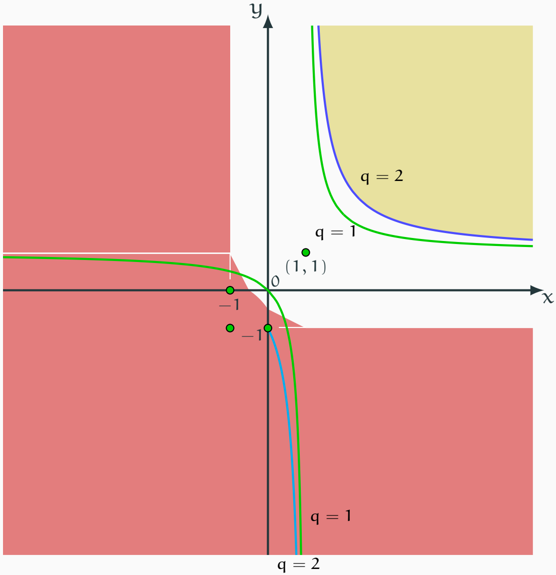
NP-hard to approximate
(#**P**-hard mostly)

#PM-equivalent

#BIS-hard

Open: white area

Ref:
Jaeger, Vertigan, and Welsh (1990);
Jerrum and Sinclair (1993);
Goldberg and Jerrum (2008, 2012, 2014)



Approximating the Tutte polynomial

$q = (x - 1)(y - 1)$

Poly-time

FPRAS

NP-hard to approximate

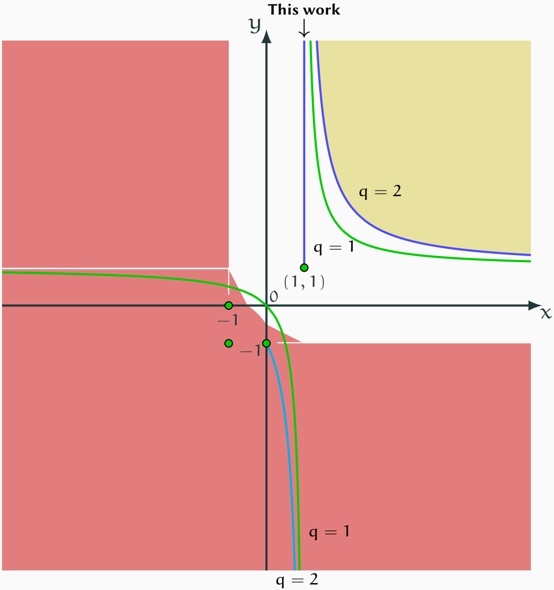
(#**P**-hard mostly)

#PM-equivalent

#BIS-hard

Open: white area

Ref:
Jaeger, Vertigan, and Welsh (1990);
Jerrum and Sinclair (1993);
Goldberg and Jerrum (2008, 2012, 2014)



Approximating the Tutte polynomial

$q = (x-1)(y-1)$

Poly-time

FPRAS

NP-hard to approximate

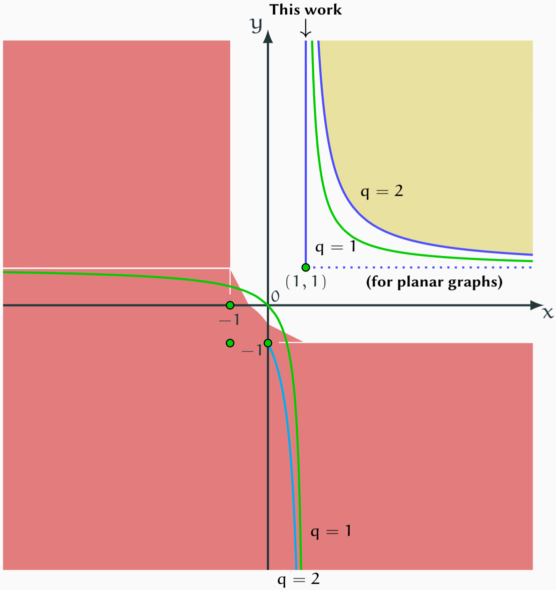
(#**P**-hard mostly)

#PM-equivalent

#BIS-hard

Open: white area

Ref:
Jaeger, Vertigan, and Welsh (1990);
Jerrum and Sinclair (1993);
Goldberg and Jerrum (2008, 2012, 2014)



A common paradigm

Both our result and the previous positive result on the Tutte plane (Jerrum and Sinclair, 1993) follow the same pattern:

1. Transform the problem into an equivalent one:
   - Ferromagnetic Ising model $\to$ even subgraphs (JS'93);
   - Reliability $\to$ bi-directed reachability.
2. Exploit some nice properties of the new solution space.

Are there other equivalences we have not discovered yet?

Open problems

- Is the Markov chain for connected subgraphs rapidly mixing?
- Approximating s-t RELIABILITY, and other variants?
  (The natural Markov chain is exponentially slow for s-t version.)
- Approximating $Z_{\mathrm{Tutte}}(G; x, 1)$ for $x > 1$ (edge-weighted forests)?

Thank you!

`arXiv:1611.01647`

(Partial rejection sampling)

`arXiv:1709.08561`

(Network reliability)

`arXiv:1807.01680`

(Tight bounds)