On sampling and approximate counting

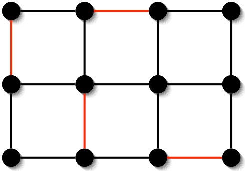
Mark Jerrum

Queen Mary, University of London

Collège de France, 9th January 2018

Example 1: Matchings (monomer-dimer)

Instance: a graph $G = (V, E)$.



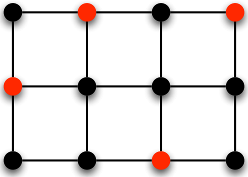
A *matching* is a collection $M \subseteq E$ of vertex-disjoint edges.

$$\pi(M) = \lambda^{|M|}/Z_{\mathsf{match}}, \quad \text{where } Z_{\mathsf{match}} = Z_{\mathsf{match}}(G, \lambda) = \sum_{M} \lambda^{|M|}.$$

Task: Sample from π, efficiently.

Example 2: Independent sets (hard-core gas)

Instance: a graph $G = (V, E)$.



An *independent set* is a subset $S \subseteq V$ of non-adjacent vertices.

$$\pi(S) = \lambda^{|S|}/Z_{\mathsf{IS}}, \quad \text{where } Z_{\mathsf{IS}} = Z_{\mathsf{IS}}(G, \lambda) = \sum_S \lambda^{|S|}.$$

Task: As before.

Estimating the partition function

A related computational task is estimating the *partition functions* $Z_{\mathsf{match}}(G, \lambda)$ and $Z_{\mathsf{IS}}(G, \lambda)$ for a specified graph G and *activity* λ, within specified relative error.

Estimating the partition function

A related computational task is estimating the *partition functions* $Z_{\mathrm{match}}(G, \lambda)$ and $Z_{\mathrm{IS}}(G, \lambda)$ for a specified graph G and *activity* λ, within specified relative error.

Metatheorem

Sampling combinatorial structures and estimating the associated partition function are of equivalent computational difficulty.

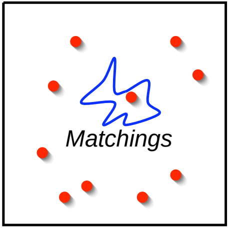
Computational complexity

Viewpoint: A computational problem is *tractable* if it can be solved in a number of steps that scales as a polynomial in the size of the instance (e.g., number of vertices).
The study of counting problems was initiated by Leslie Valiant.

Leslie Valiant

- Exact evaluation of the partition function is intractable (#P-complete) in both examples.
- Sampling configurations is tractable in one example and intractable in the other.
- Estimation of the partition function is tractable in one example and intractable in the other.

Approach 1: Rejection sampling (Dart throwing)

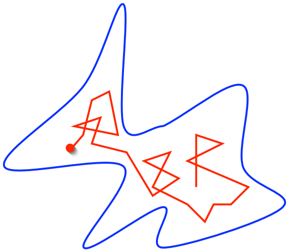


All subsets of E

- Until success:
  - Choose $M \subseteq E$ uniformly at random.
  - If M is a matching, output M.

Correct distribution (for $\lambda = 1$), but exponential running time.

Approach 2: Markov chain Monte Carlo (MCMC)



- $M := \emptyset$.
- Repeat for sufficiently many steps:
  - Choose $e \in E$ uniformly at random, and let $M' := M \oplus \{e\}$.
  - If M' is a matching then $M := M'$; otherwise, do nothing.
- Return M.

Mixing time

The trial just described defines the transition probabilities P of a *Markov chain* on state space

$$\Omega = \{\text{All matchings in } G\}.$$

This Markov chain converges to a stationary distribution π that is uniform on Ω.

Andrei A. Markov

We are interested in the *mixing time* τ of the Markov chain, i.e., the time to convergence to near stationarity.

Canonical paths/Multi-commodity flow

For every pair of states $x, y \in \Omega$, define a *canonical path* γ_{xy} from x to y using valid transitions of the MC.
"Congestion constant" ρ:

$$\sum_{\gamma_{xy} \ni (z,z')} \pi(x)\pi(y)\,|\gamma_{xy}| \leqslant \rho\,\pi(z)P(z,z'), \quad \forall z, z'.$$

Canonical paths/Multi-commodity flow

For every pair of states $x, y \in \Omega$, define a *canonical path* γ_{xy} from x to y using valid transitions of the MC.
"Congestion constant" ρ:

$$\sum_{\gamma_{xy} \ni (z,z')} \pi(x)\pi(y)\,|\gamma_{xy}| \leqslant \rho\,\pi(z)P(z,z'), \quad \forall z, z'.$$

"flow" through a transition

Canonical paths/Multi-commodity flow

For every pair of states $x, y \in \Omega$, define a *canonical path* γ_{xy} from x to y using valid transitions of the MC.
"Congestion constant" ρ:

$$\sum_{\gamma_{xy} \ni (z,z')} \pi(x)\pi(y)\,|\gamma_{xy}| \leqslant \rho\, \pi(z)P(z,z'), \quad \forall z, z'.$$

"flow" through a transition   capacity of the transition

Canonical paths/Multi-commodity flow

For every pair of states $x, y \in \Omega$, define a *canonical path* γ_{xy} from x to y using valid transitions of the MC.
"Congestion constant" ρ:

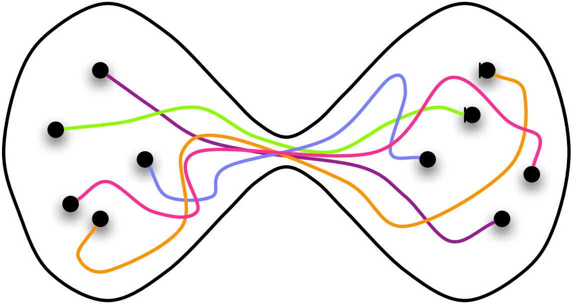
$$\sum_{\gamma_{xy} \ni (z,z')} \pi(x)\pi(y)\,|\gamma_{xy}| \leqslant \rho\,\pi(z)P(z,z'), \quad \forall z, z'.$$

Theorem (Diaconis, Stroock; Sinclair)

$$\tau = O(\rho \log \pi_{\min}^{-1}).$$

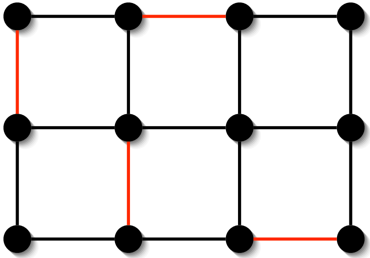
Alistair Sinclair

Low congestion implies no "bottleneck"



Richer set of transitions

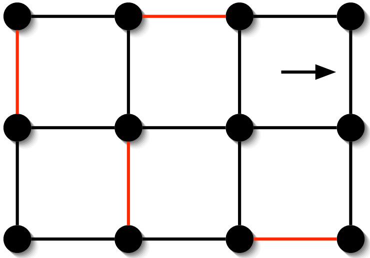
Convenient to augment existing "add" and "delete" transitions with a "slide":



[Broder, 1986; J. & Sinclair, 1988]

Richer set of transitions

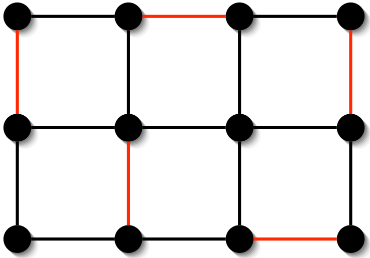
Convenient to augment existing “add” and “delete” transitions with a “slide”:



[Broder, 1986; J. & Sinclair, 1988]

Richer set of transitions

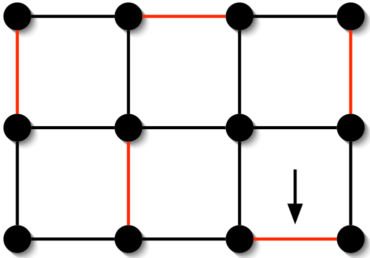
Convenient to augment existing "add" and "delete" transitions with a "slide":



[Broder, 1986; J. & Sinclair, 1988]

Richer set of transitions

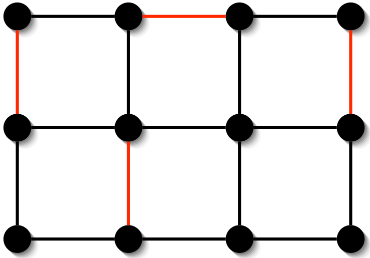
Convenient to augment existing "add" and "delete" transitions with a "slide":



[Broder, 1986; J. & Sinclair, 1988]

Richer set of transitions

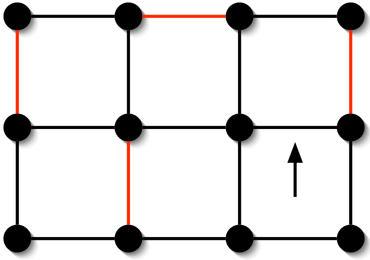
Convenient to augment existing "add" and "delete" transitions with a "slide":



[Broder, 1986; J. & Sinclair, 1988]

Richer set of transitions

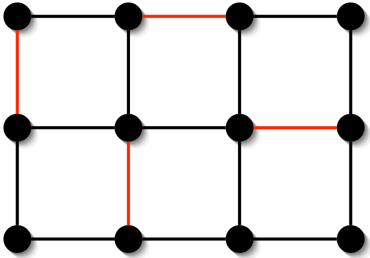
Convenient to augment existing "add" and "delete" transitions with a "slide":



[Broder, 1986; J. & Sinclair, 1988]

Richer set of transitions

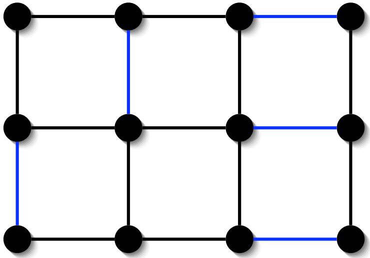
Convenient to augment existing "add" and "delete" transitions with a "slide":



[Broder, 1986; J. & Sinclair, 1988]

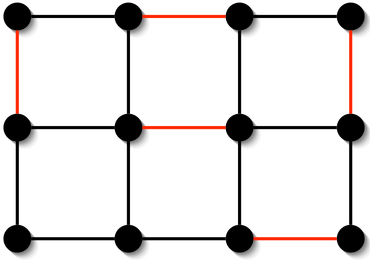
Canonical paths for matchings

To get from the blue matching. . .



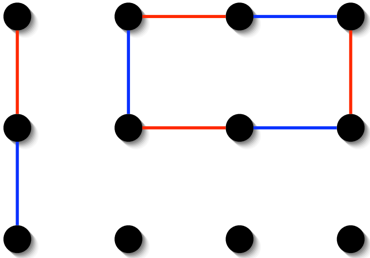
Canonical paths for matchings

. . . to the red matching. . .



Canonical paths for matchings

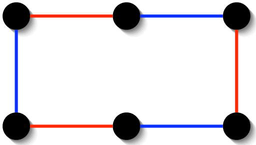
. . . first superimpose red and blue (symmetric difference). . .



and then "unwind" each component (path or cycle).

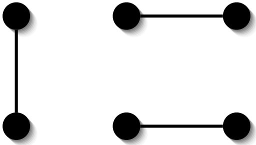
"Unwinding" a cycle

The cycle:



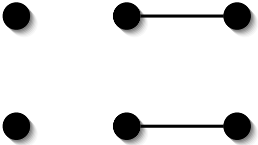
“Unwinding” a cycle

Initial matching:



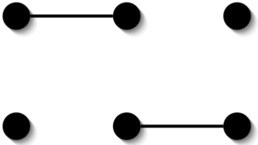
"Unwinding" a cycle

After 1 step:



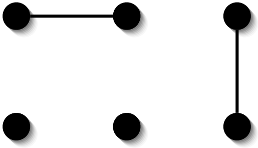
"Unwinding" a cycle

After 2 steps:



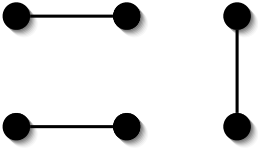
“Unwinding” a cycle

After 3 steps:



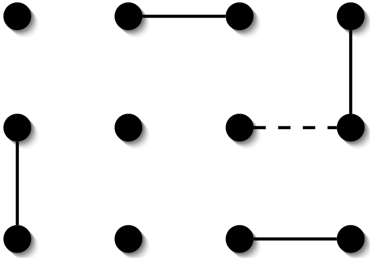
“Unwinding” a cycle

After 4 steps (final matching):



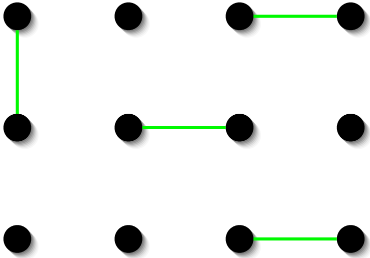
Encoding a canonical path through a transition

A transition:



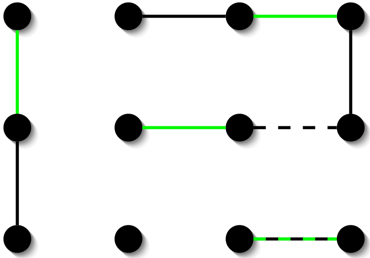
Encoding a canonical path through a transition

An encoding (matching):



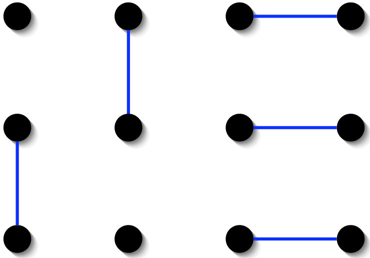
Encoding a canonical path through a transition

Superposition reveals the initial and final matching:



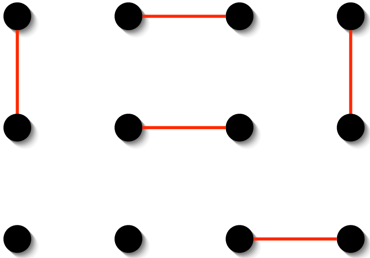
Encoding a canonical path through a transition

Superposition reveals the initial and final matching:



Encoding a canonical path through a transition

Superposition reveals the initial and final matching:



Calculating the congestion

The encoding argument shows that the number of canonical paths passing through a given transition is roughly equal to the size of the state space.

Pursuing the calculation in more detail yields:

Theorem (J. & Sinclair)

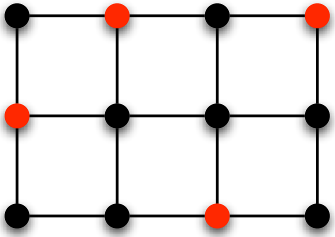
$\rho = O(nm\bar{\lambda}^2)$, *where* $n = |V|$, $m = |E|$ *and* $\bar{\lambda} = \max\{\lambda, 1\}$.

Corollary

$\tau = O(nm^2\bar{\lambda}^2)$.

Return to independent sets

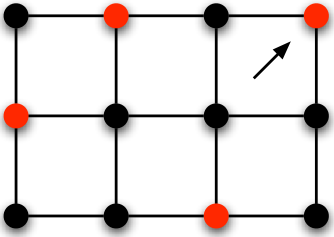
Consider an analogous set of transitions.



E.g., [Luby and Vigoda, Bubley and Dyer].

Return to independent sets

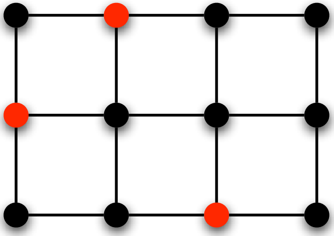
Consider an analogous set of transitions.



E.g., [Luby and Vigoda, Bubley and Dyer].

Return to independent sets

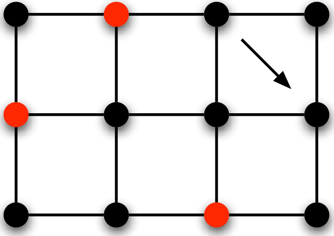
Consider an analogous set of transitions.



E.g., [Luby and Vigoda, Bubley and Dyer].

Return to independent sets

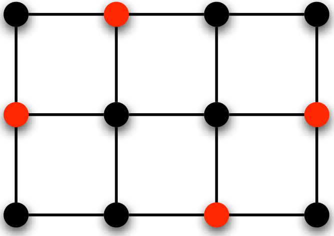
Consider an analogous set of transitions.



E.g., [Luby and Vigoda, Bubley and Dyer].

Return to independent sets

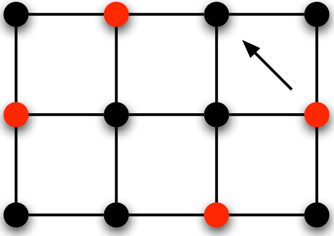
Consider an analogous set of transitions.



E.g., [Luby and Vigoda, Bubley and Dyer].

Return to independent sets

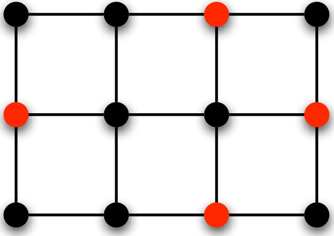
Consider an analogous set of transitions.



E.g., [Luby and Vigoda, Bubley and Dyer].

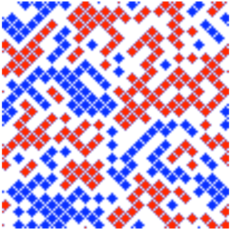
Return to independent sets

Consider an analogous set of transitions.



E.g., [Luby and Vigoda, Bubley and Dyer].

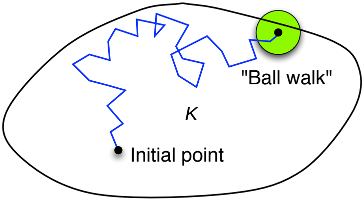
Clustering at higher activities



- The diamonds are the vertices in the independent set; the colour red/blue indicates parity.
- At high enough λ there is a bias towards red or blue. (E.g., red "islands" in a blue "sea".) This leads to a constriction in the state space.

Example 3: sampling from a convex body

[Dyer, Frieze & Kannan, 1991], [Lovász & Simonovits, 1997].



Exploit the geometry directly (c.f. Cheeger inequality).

Mixing is rapid, provided $K \subseteq \mathbb{R}^n$ is not "long and thin", specifically, $\tau = \mathrm{poly}(n, \mathrm{diam}\, K)$.

Technically, surprisingly hard!

Some other successes

- Proper colourings of a bounded degree graph, a.k.a. antiferromagnetic Potts model [Various].
- Partition function of the Ferromagnetic Ising model [J. and Sinclair].
- Bases of a "balanced" matroid [Feder and Mihail].
- Linear extensions of a partial order. [Khachiyan and Karzanov], [Bubley and Dyer].
- Feasible solutions to an instance of the knapsack problem [Morris and Sinclair].
- Perfect matchings in a bipartite graph [J., Sinclair and Vigoda].

A renaissance

MCMC was to a close approximation the only game in town. That situation has changed in the last ten years with a rapid expansion of techniques. Can be explored in the context of the independent sets model.

Started with Weitz [2006] who exploited decay of correlations to produce a deterministic algorithm. Uses $O(\log n)$ depth explorations from a vertex v to estimate the probability that v is in a random independent set.

Theorem (Weitz, 2006)

If $\lambda < \lambda_c(d)$ then there is an FPRAS for $Z_{IS}(\lambda)$ on graphs of maximum degree d.

A renaissance (continued)

Here $\lambda_c(d)$ is the location of the phase transition (unique versus multiple Gibbs measures) in an infinite regular tree of degree d. This result prompts the daring conjecture that the phase transition in the usual physical sense coincides with a phase transition for computational tractability. Indeed:

Theorem (Sly, 2010; Galanis, Ge, Štefankovič, Vigoda and Yang, 2012; Sly and Sun, 2012)

If $\lambda > \lambda_c(d)$ then $Z_{\mathsf{IS}}(\lambda)$ is NP-hard to approximate, even for graphs of maximum degree d.

So, remarkably, for graphs of bounded degree d, the critical value $\lambda_c(d)$ is also the boundary between computational tractability and intractability.

A renaissance (continued)

Another significant line of attack was introduced by Barvinok, and taken forward by Patel and Regts.

The idea is to consider a Taylor expansion of $\log Z$ about some point where the partition function is easy to evaluate. Then by enumerating small substructures, one can estimate the first few coefficients in the expansion. This in turn gives a good estimate for Z at a hard-to-evaluate point.

For this to work we need there to be no zeros in the neighbourhood in which we are performing the Taylor expansion.

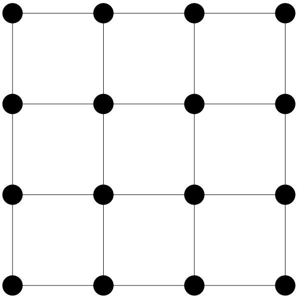
Zeros of the partition function Z are associated with phase transitions: another connection to the physics of spin systems!

Independent sets by Partial Rejection Sampling

A further recent approach is to efficient sampling is *Partial rejection sampling* [J. & Guo, 2017]:

Independent sets by Partial Rejection Sampling

1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.
2. Let Bad be the set of vertices in connected components of size at least 2.
3. Resample $=$ Bad $\cup\, \partial$Bad.
4. Resample variables in set Resample. Check independence.



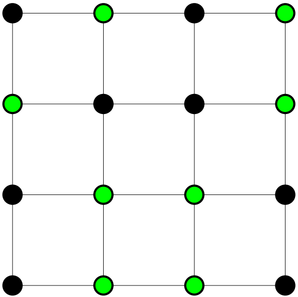
Independent sets by Partial Rejection Sampling

→ 1. Randomize each vertex (in/out).
Consider the connected components induced by the in-vertices.

2. Let Bad be the set of vertices in connected components of size at least 2.

3. Resample $=$ Bad $\cup\, \partial$Bad.

4. Resample variables in set Resample.
Check independence.



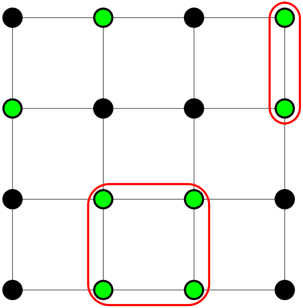
Independent sets by Partial Rejection Sampling

1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.

→ 2. Let Bad be the set of vertices in connected components of size at least 2.

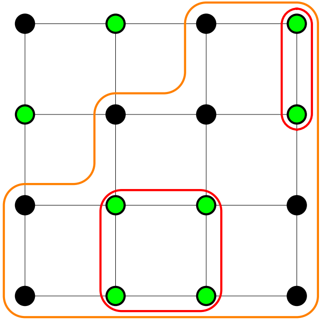
3. Resample $=$ Bad $\cup\, \partial$Bad.

4. Resample variables in set Resample. Check independence.



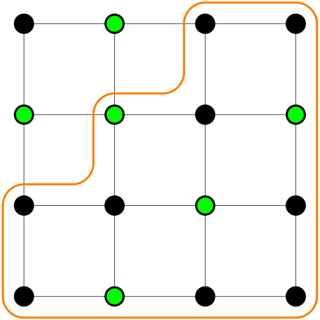
Independent sets by Partial Rejection Sampling

1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.
2. Let Bad be the set of vertices in connected components of size at least 2.
3. Resample $=$ Bad $\cup\, \partial$Bad.
4. Resample variables in set Resample. Check independence.



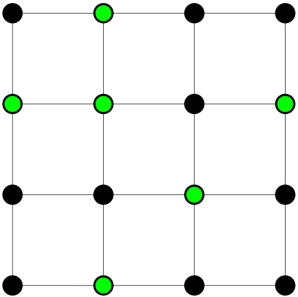
Independent sets by Partial Rejection Sampling

1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.
2. Let Bad be the set of vertices in connected components of size at least 2.
3. Resample = Bad $\cup$ ∂Bad.
4. Resample variables in set Resample. Check independence.



Independent sets by Partial Rejection Sampling

1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.
2. Let Bad be the set of vertices in connected components of size at least 2.
3. Resample $=$ Bad $\cup\, \partial$Bad.
4. → Resample variables in set Resample. Check independence.



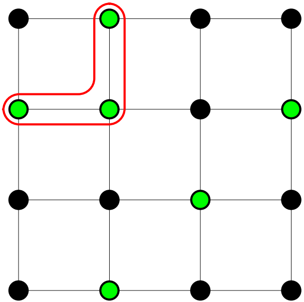
Independent sets by Partial Rejection Sampling

1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.

→ 2. Let Bad be the set of vertices in connected components of size at least 2.

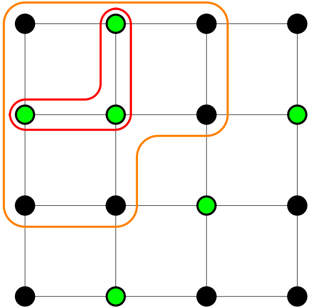
3. Resample $=$ Bad $\cup\, \partial$Bad.

4. Resample variables in set Resample. Check independence.



Independent sets by Partial Rejection Sampling

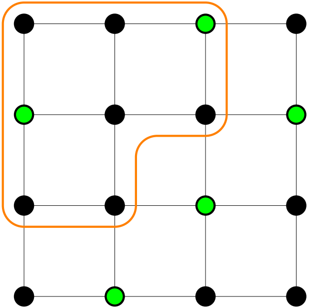
1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.
2. Let Bad be the set of vertices in connected components of size at least 2.
3. → Resample $=$ Bad $\cup\ \partial$Bad.
4. Resample variables in set Resample. Check independence.



Independent sets by Partial Rejection Sampling

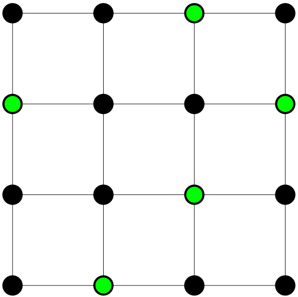
1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.
2. Let Bad be the set of vertices in connected components of size at least 2.
3. Resample $=$ Bad $\cup\, \partial$Bad.

$\rightarrow$ 4. Resample variables in set Resample. Check independence.



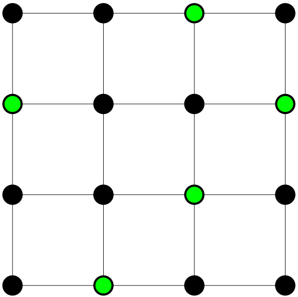
Independent sets by Partial Rejection Sampling

1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.
2. Let Bad be the set of vertices in connected components of size at least 2.
3. Resample = Bad $\cup$ ∂Bad.
4. Resample variables in set Resample. Check independence.



Independent sets by Partial Rejection Sampling

1. Randomize each vertex (in/out). Consider the connected components induced by the in-vertices.
2. Let Bad be the set of vertices in connected components of size at least 2.
3. Resample $=$ Bad $\cup\, \partial$Bad.
4. Resample variables in set Resample. Check independence.



When the algorithm stops, it yields a uniform independent set.

Unblocking sets

The key property of the set Resample is that it is unblocking under the current assignment σ to the variables.

Definition

A set $\mathcal{U}$ of variables is unblocking under σ if $\sigma[\mathcal{U}]$ determines the truth value of all clauses that *share* variables with $\mathcal{U}$ (and not just the clauses containing *only* variables from $\mathcal{U}$

The resampling set from the independent set example was unblocking.

In applications we also require that $\mathcal{U}$ is "adapted" to σ.

Partial rejection sampling

Algorithm 1 Partial Rejection Sampling

PRS(V, Φ) // Φ is a formula on variable set V
Sample, from the product distribution, an assignment σ to the variables in V
while Bad$(\sigma) \neq \emptyset$ **do**
  $S \leftarrow \bigcup\{\mathrm{var}(C) : C \in \mathrm{Bad}(\sigma)\}$
  Resample all variables in $U = \mathrm{Resample}(S; \sigma)$
end while

Note 1. The procedure Resample must not probe variables outside of U while computing U. This is the condition of being adapted.

Note 2. A previous approach to rejection sampling that tries to preserve randomness is the "Randomness Recycler" of Fill and Huber.

A selection of open problems

- Is there a polynomial-time algorithm for sampling perfect matchings in a *general* graph?
- Is there an algorithm for sampling perfect matchings in a bipartite graph that is efficient in practice?
- Is there a polynomial-time algorithm for sampling contingency tables?
- Can one sample proper colourings efficiently when $q \geqslant (1+\varepsilon)d$? ($q$ is the number of colours and d the maximum degree.)
- Is the bases-exchange walk rapidly mixing for all matroids?