Perfect sampling, Part II

Mark Jerrum

Queen Mary, University of London

ADYN Summer School
27th June – 1st July, 2022

Joint work with Konrad Anand (QMUL), Heng Guo (Edinburgh) and Jingcheng Liu (Nanjing)

Beyond extremal instances

So far, we have seen partial rejection sampling for extremal instances.

We cannot expect many sampling problems to correspond to extremal instances.

The way we can extend the range of partial rejection sampling is suggested by the following example.

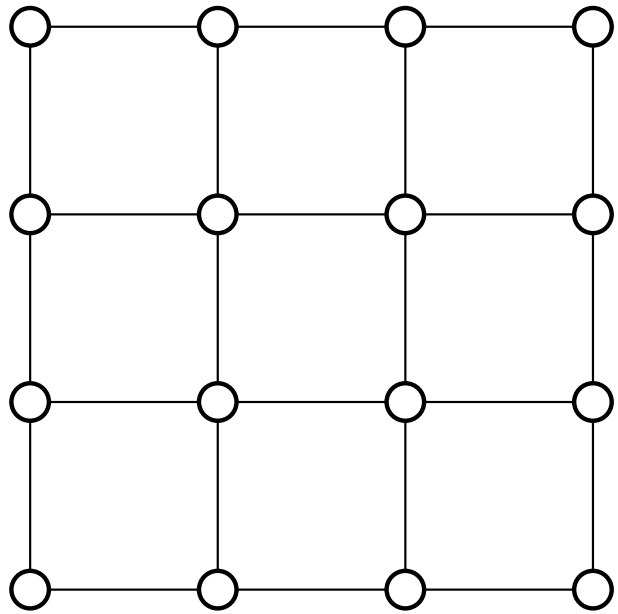
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

4. Resample variables in this set.
   Check independence.



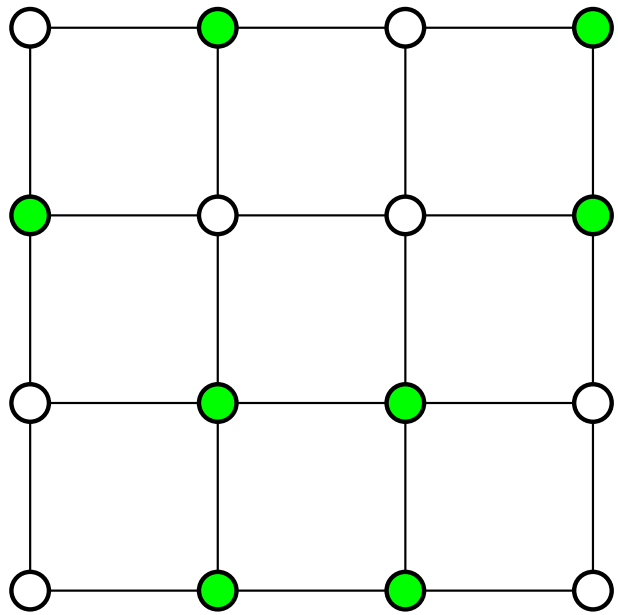
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

4. Resample variables in this set.
Check independence.



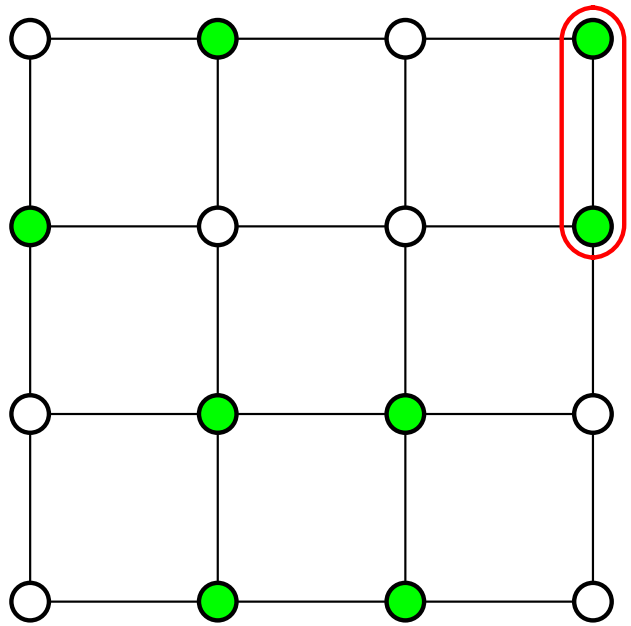
Sampling independent sets (Hardcore model)

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

→ 2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

4. Resample variables in this set.
   Check independence.



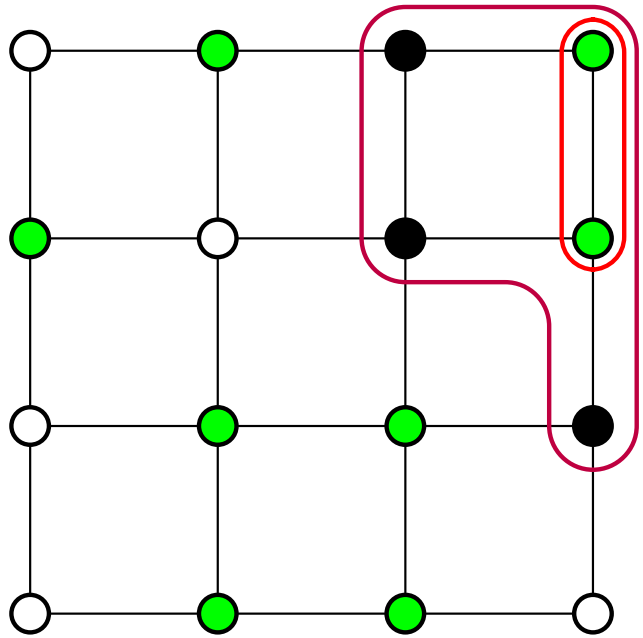
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

→ 3. Add the boundary (outlined in purple).

4. Resample variables in this set.
   Check independence.



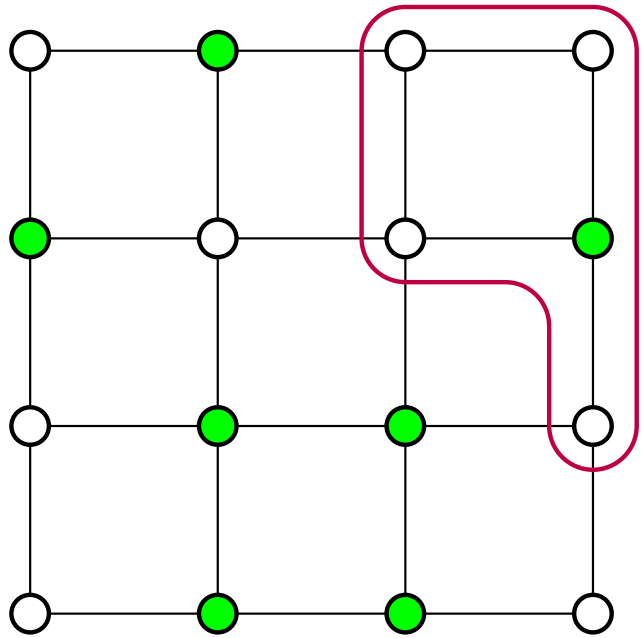
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

→ 4. Resample variables in this set.
   Check independence.



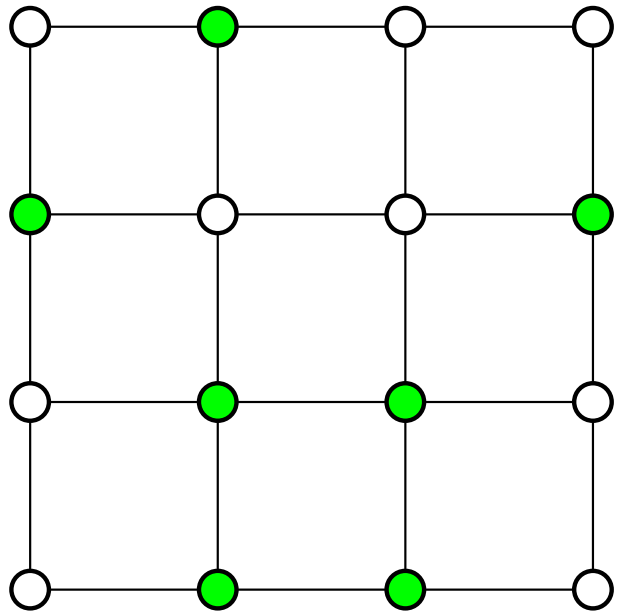
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

→ 4. Resample variables in this set.
   Check independence.



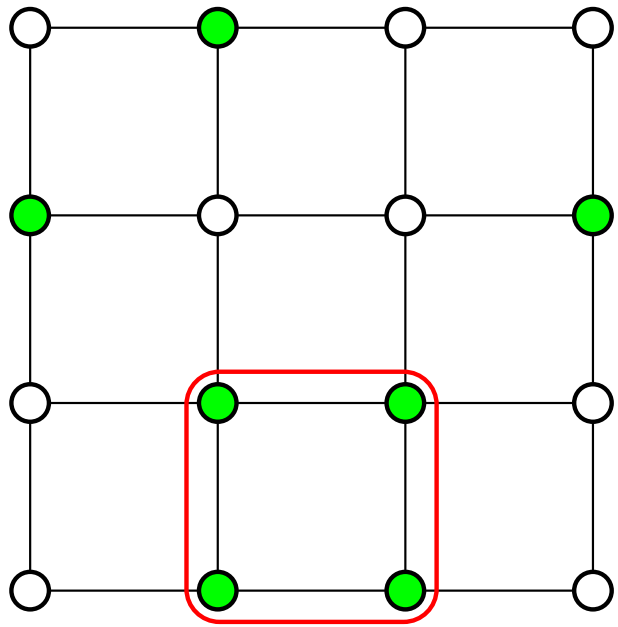
Sampling independent sets (Hardcore model)

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

→ 2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

4. Resample variables in this set.
   Check independence.



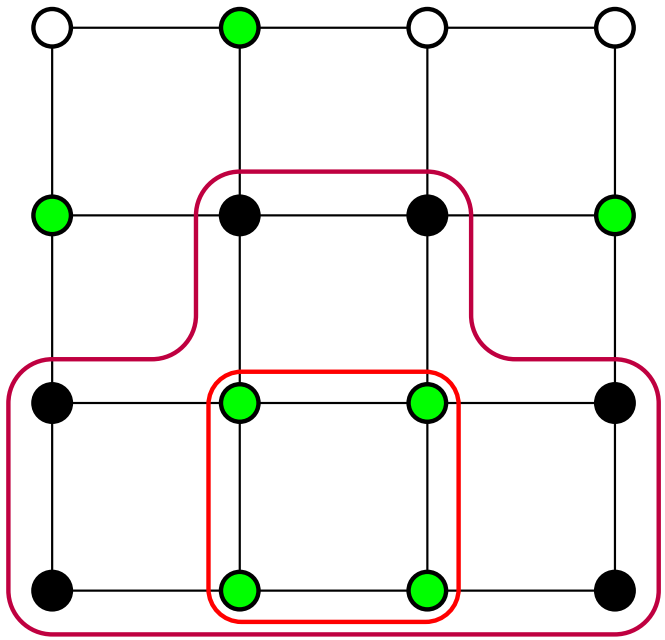
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

→ 3. Add the boundary (outlined in purple).

4. Resample variables in this set.
   Check independence.



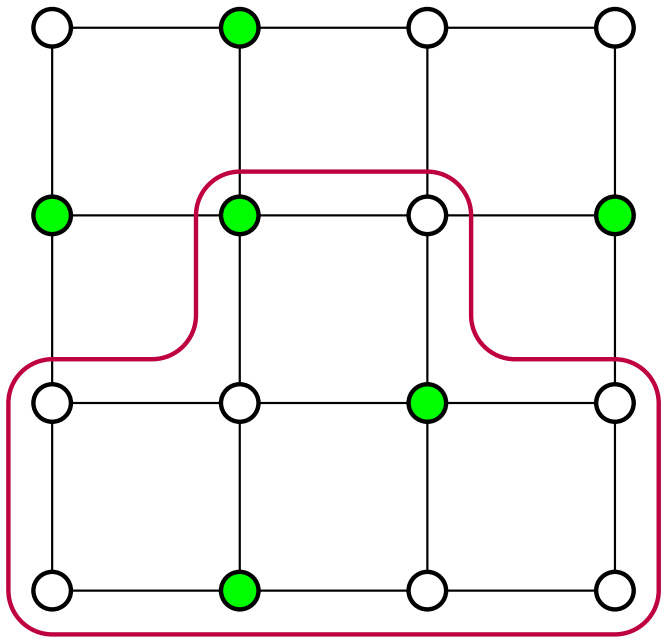
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

→ 4. Resample variables in this set.
   Check independence.



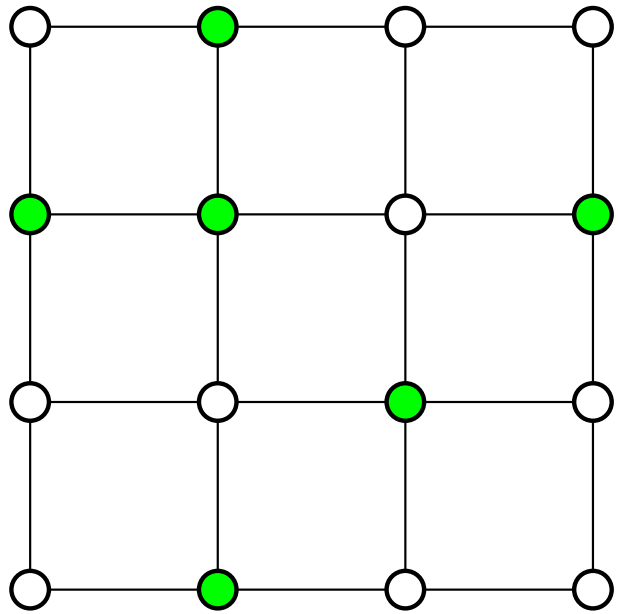
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

→ 4. Resample variables in this set.
   Check independence.



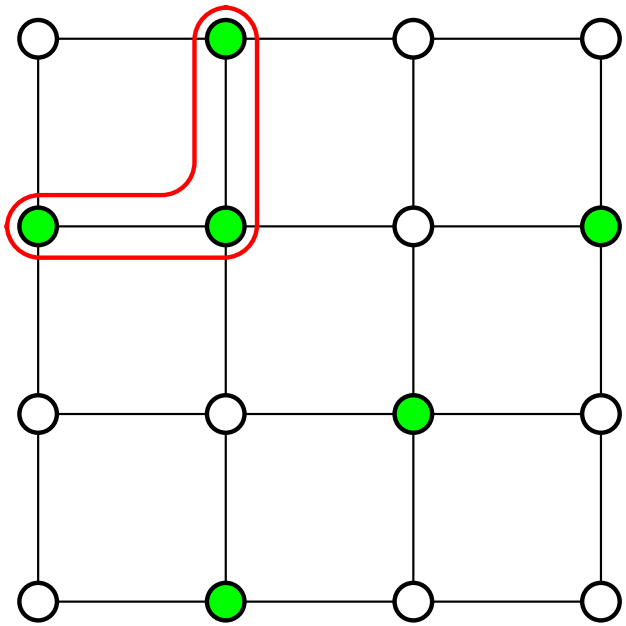
Sampling independent sets (Hardcore model)

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

→ 2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

4. Resample variables in this set.
   Check independence.



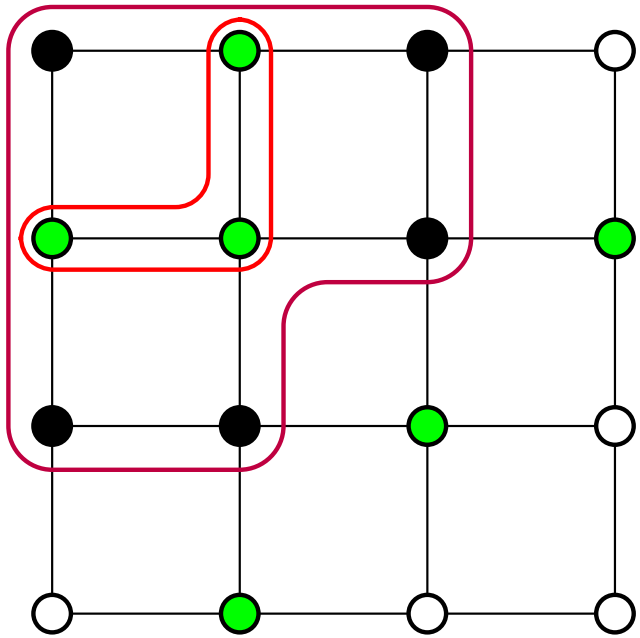
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

→ 3. Add the boundary (outlined in purple).

4. Resample variables in this set.
   Check independence.



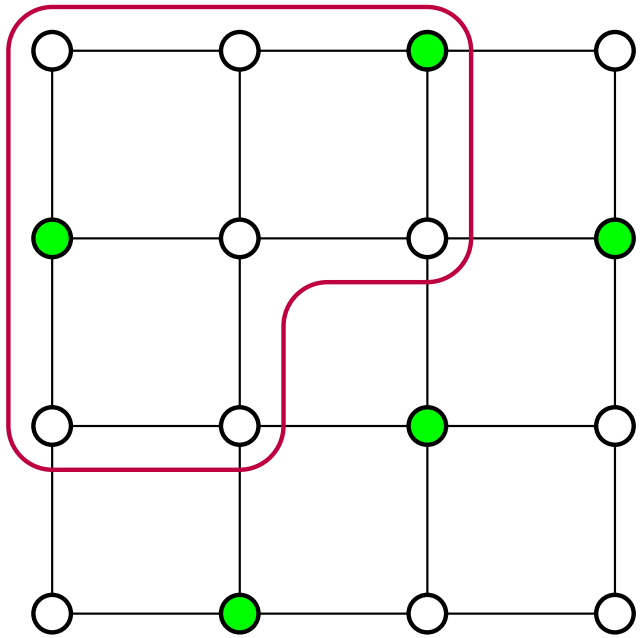
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

→ 4. Resample variables in this set.
   Check independence.



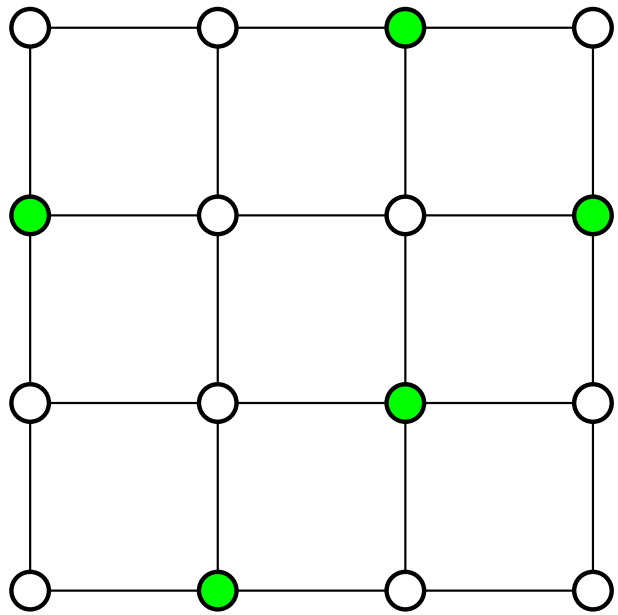
Sampling independent sets (Hardcore model)

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

2. Find a connected component of size at least 2 (outlined in red).

3. Add the boundary (outlined in purple).

→ 4. Resample variables in this set.
   Check independence.



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

Return to the PRS framework

- We have variables $X_1, \ldots, X_n$ corresponding to vertices of G.
- The most direct encoding as a formula is

$$\Phi_G(\mathbf{X}) = \bigwedge_{\{i,j\} \in E(G)} (\neg X_i \vee \neg X_j),$$

but we have seen that this doesn't work: Φ is not extremal.

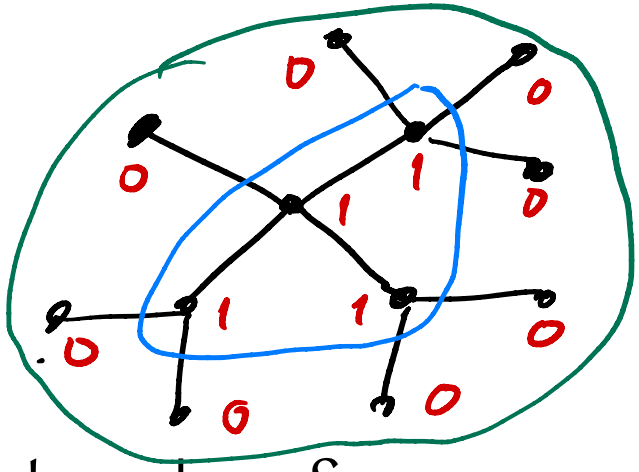
- Let $S \subseteq V(G)$ be any subset of vertices that induces a connected subgraph of G. Let ∂S be the boundary of S (vertices that are not in S but that are adjacent to a vertex in S). We say that S is a *cluster* with respect to $\mathbf{X}$ iff

$|S| \geq 2$

$$X_i = 1 \text{ for all } i \in S \quad \text{and} \quad X_i = 0 \text{ for all } i \in \partial S.$$

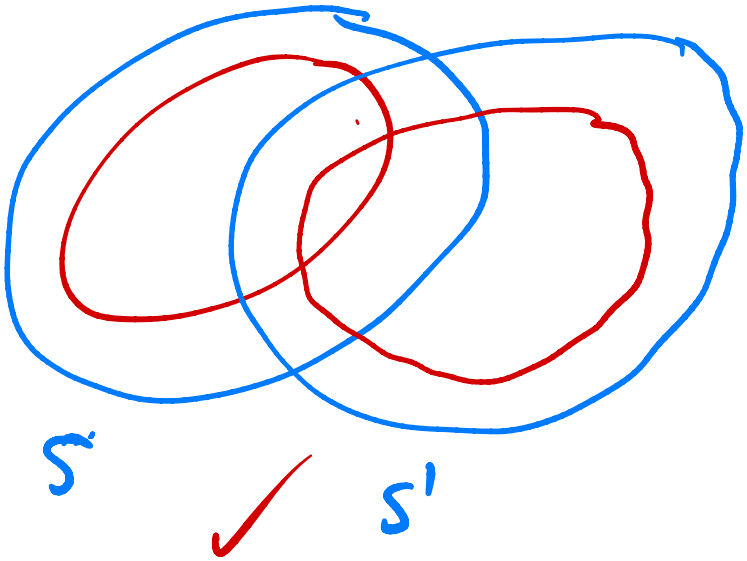
The formula $\phi_S(\mathbf{X})$ expresses the condition that S is a cluster.

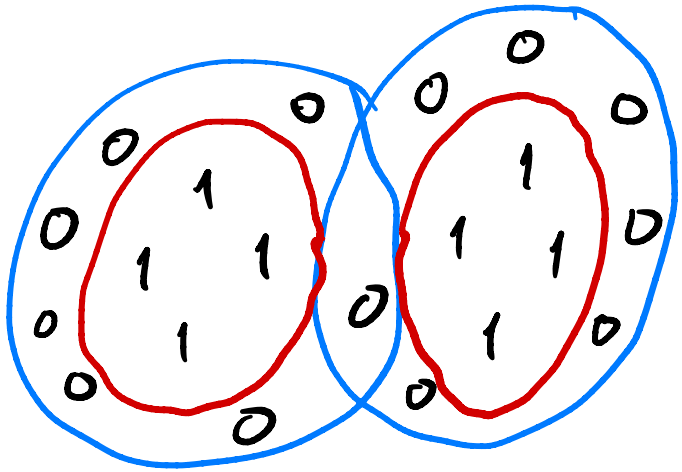
A modified formula

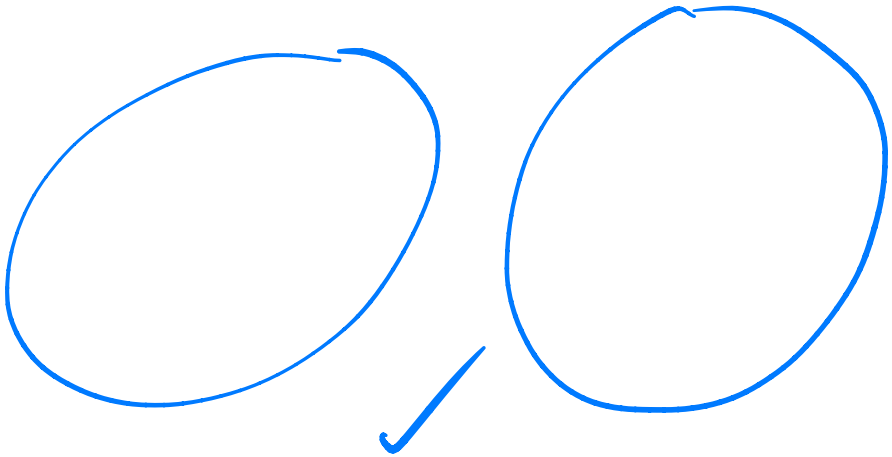


- Consider the formula $\Phi = \bigwedge_S \phi_S$, where S ranges over all vertex sets described above.
- $\Phi(\mathbf{X})$ expresses that $\mathbf{X}$ encodes an independent set.
- The formula Φ is not extremal either, but it is sufficiently close.
- We indicate what 'sufficiently close' means in the context of independent sets...

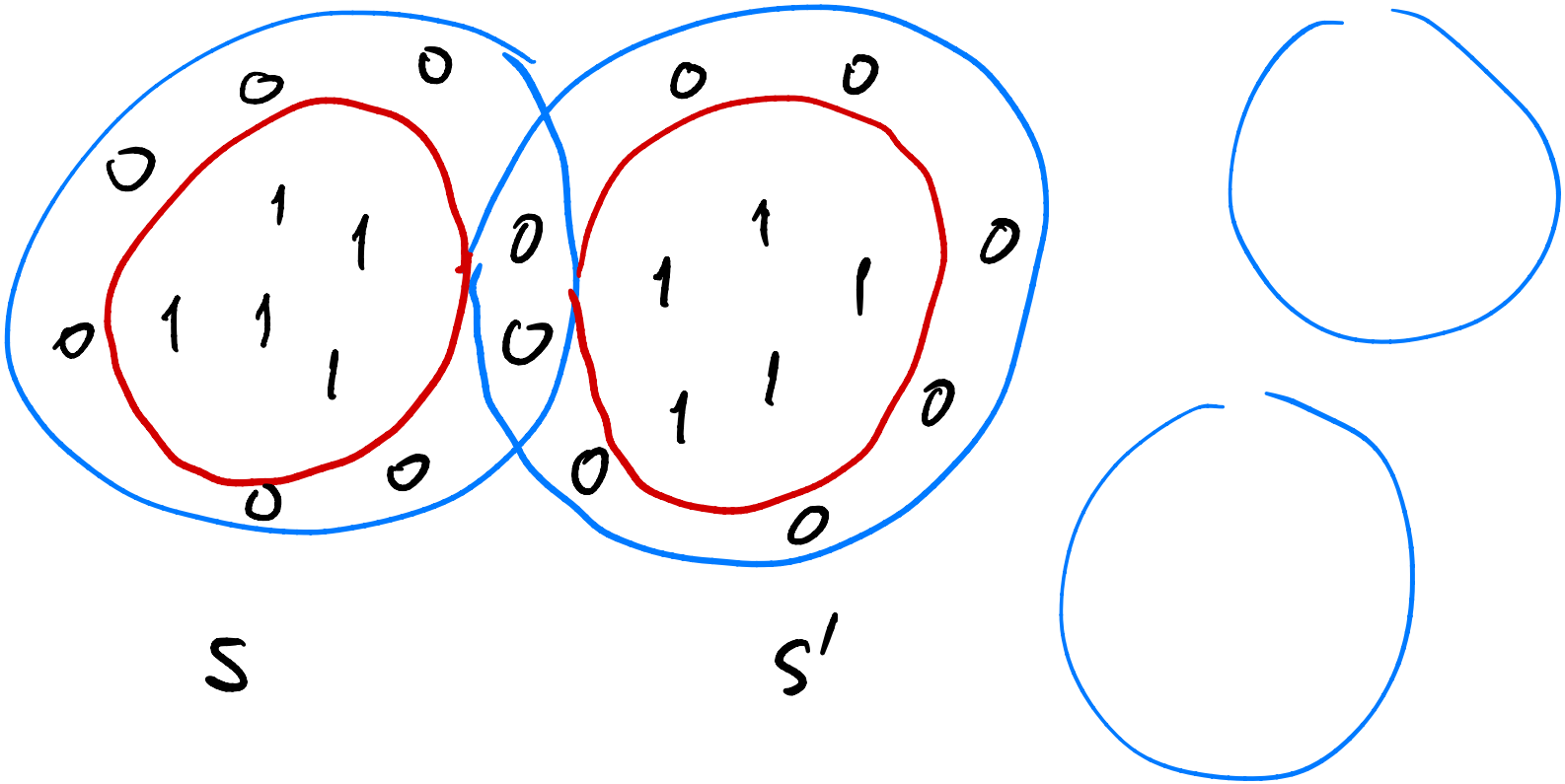
A relaxation of 'extremal' in pictures







A relaxation of ‘extremal’ in pictures



Correctness and efficiency

- If a formula Φ meets the relaxed definition just sketched, then PRS produces a satisfying assignment with the desired distribution.
- It is no longer true that clauses can be resampled in any order. However, any choice rule that is based purely on which clauses are false will work. (Peeking at the variables will spoil the output distribution.)
- Unfortunately, there is no longer a neat expression for the expected number of resamplings.
- However, assuming a certain commutativity condition (to be sketched), a significant body of work on 'Lopsided Lovász Local Lemma' can be brought to bear.

Commutativity and run time analysis

- The resulting time bounds are optimal (linear in n, the number of variables).
- However the range of validity can be a little disappointing. E.g., Markov chain simulation can sample independent sets with substantially higher density.

Hard discs model

Poisson point process of rate $\lambda_r = \lambda/(\pi r^2)$ in a unit square, conditioned on no pair of points being closer than $2r$.

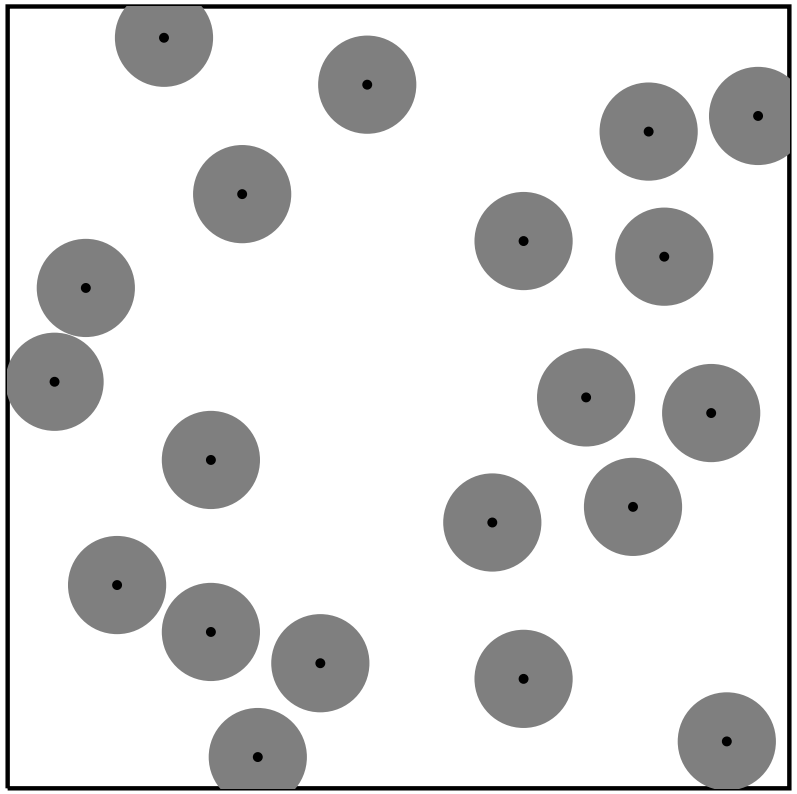
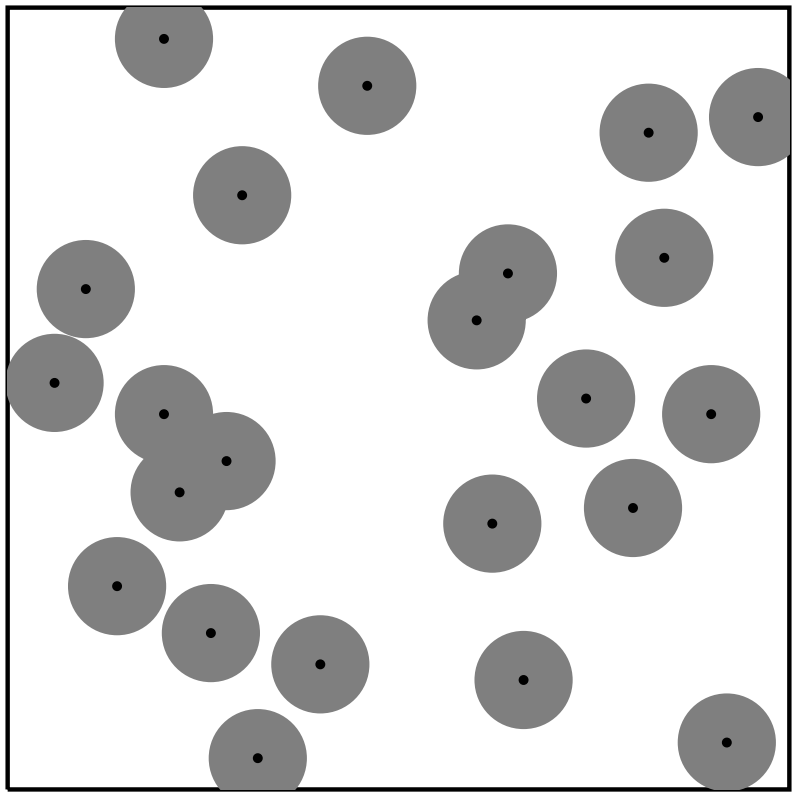


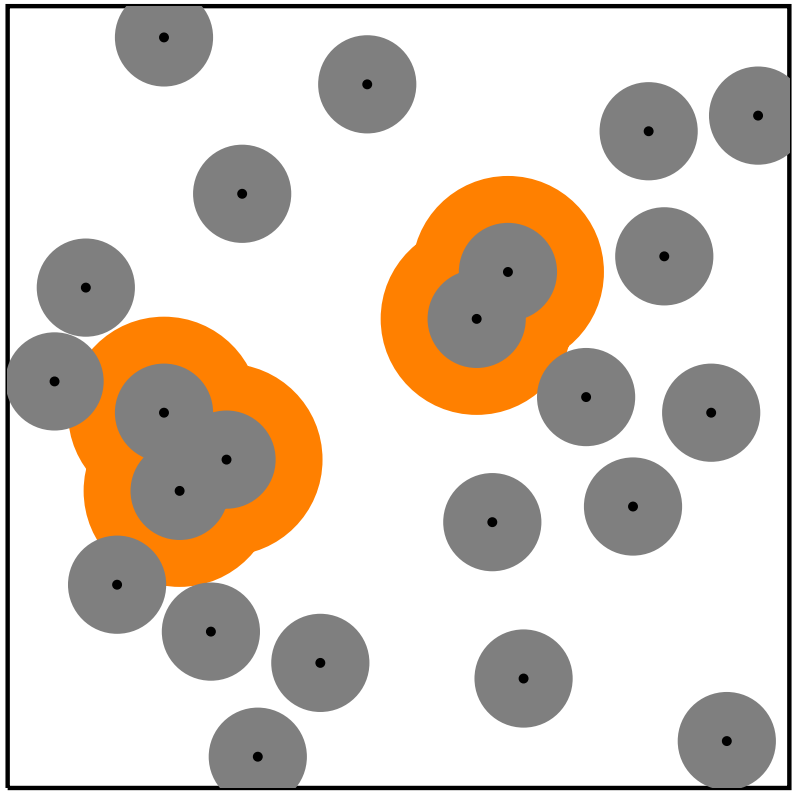
Figure: A typical configuration with disks of radius r.

Partial rejection sampling applied to hard disks



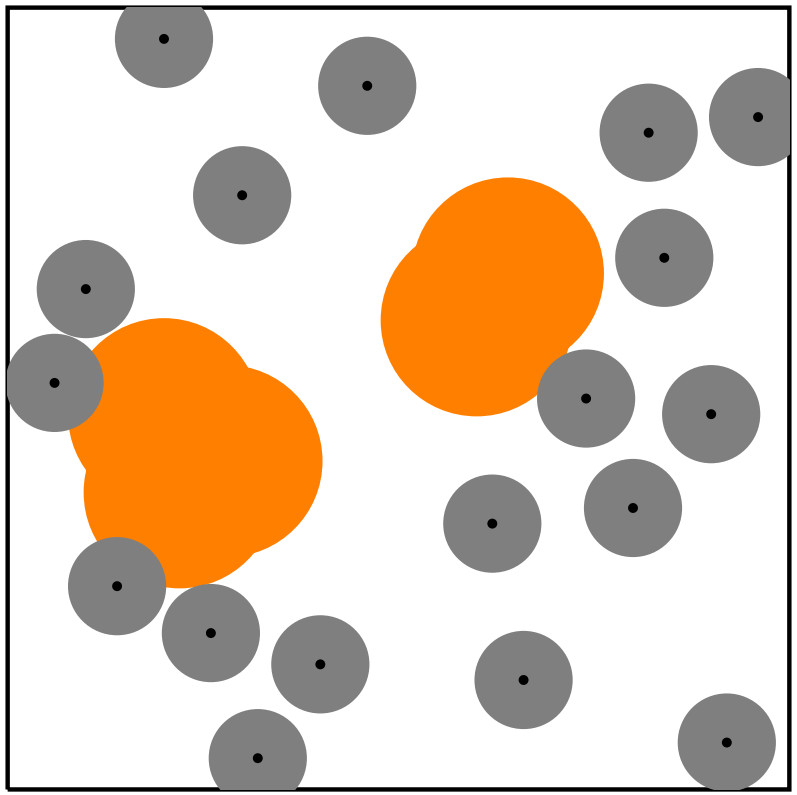
Initial configuration.

Partial rejection sampling applied to hard disks



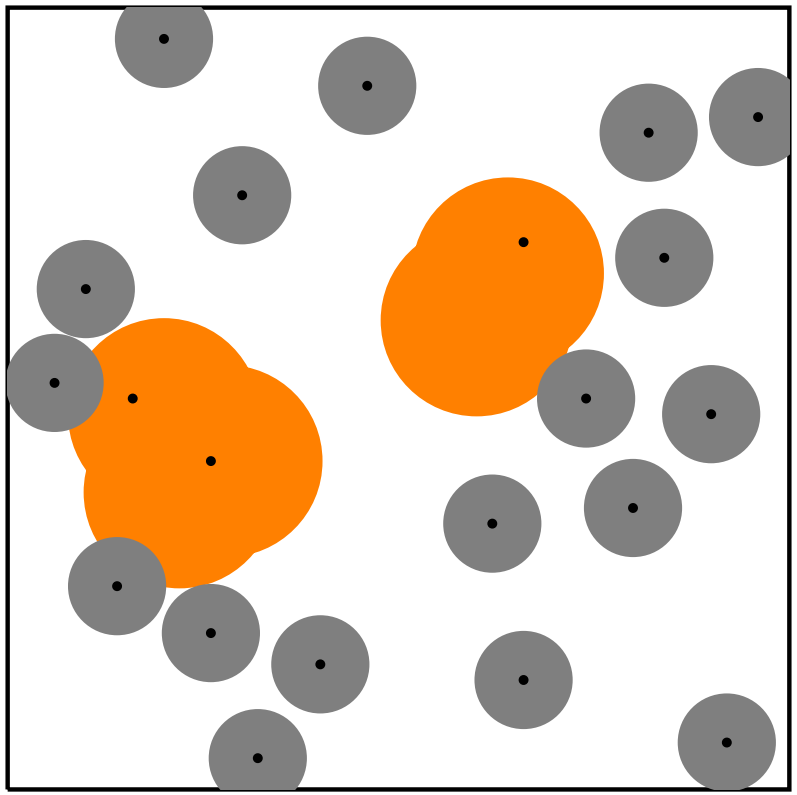
Compute the resampling set (orange).

Partial rejection sampling applied to hard disks



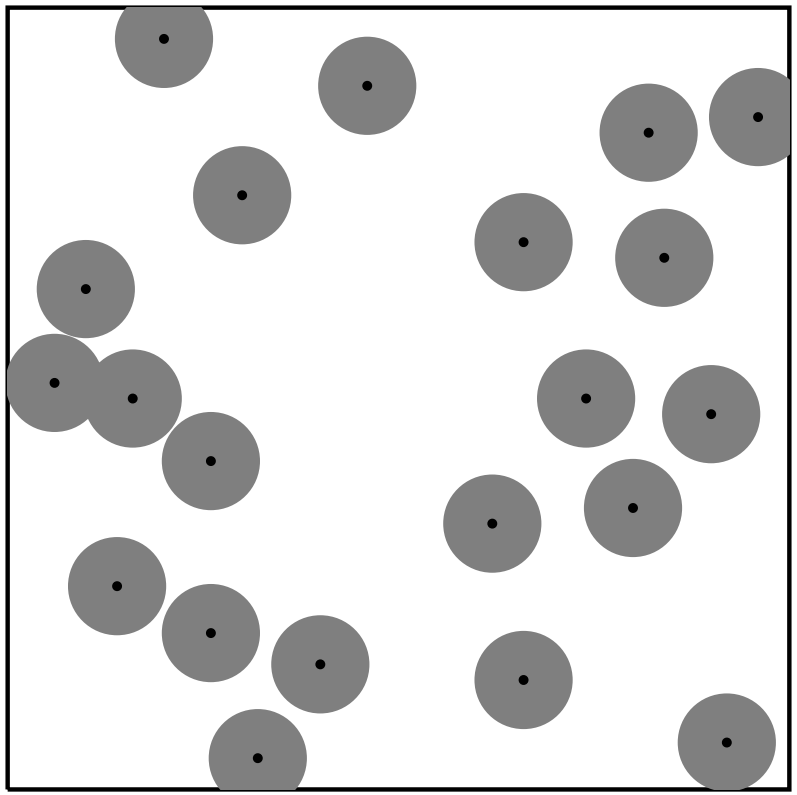
Compute the resampling set.

Partial rejection sampling applied to hard disks



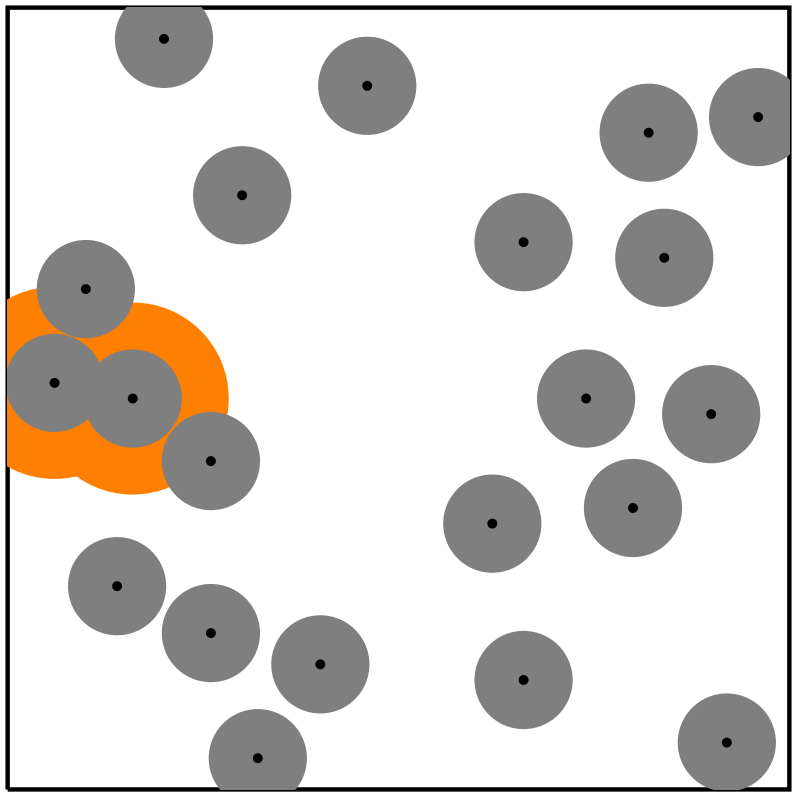
Rerandomise within the resampling set.

Partial rejection sampling applied to hard disks



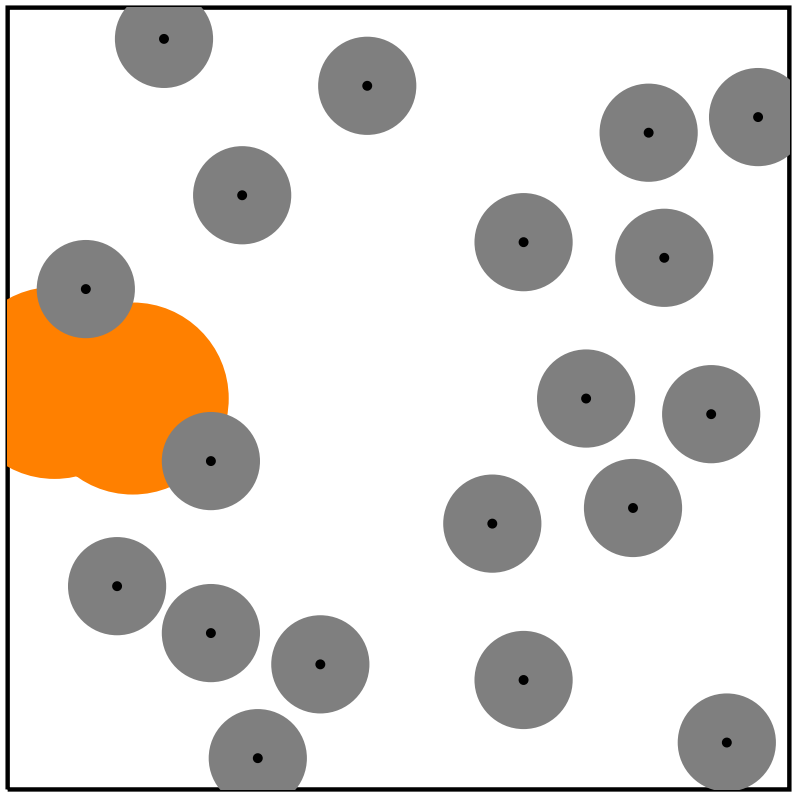
Updated configuration.

Partial rejection sampling applied to hard disks



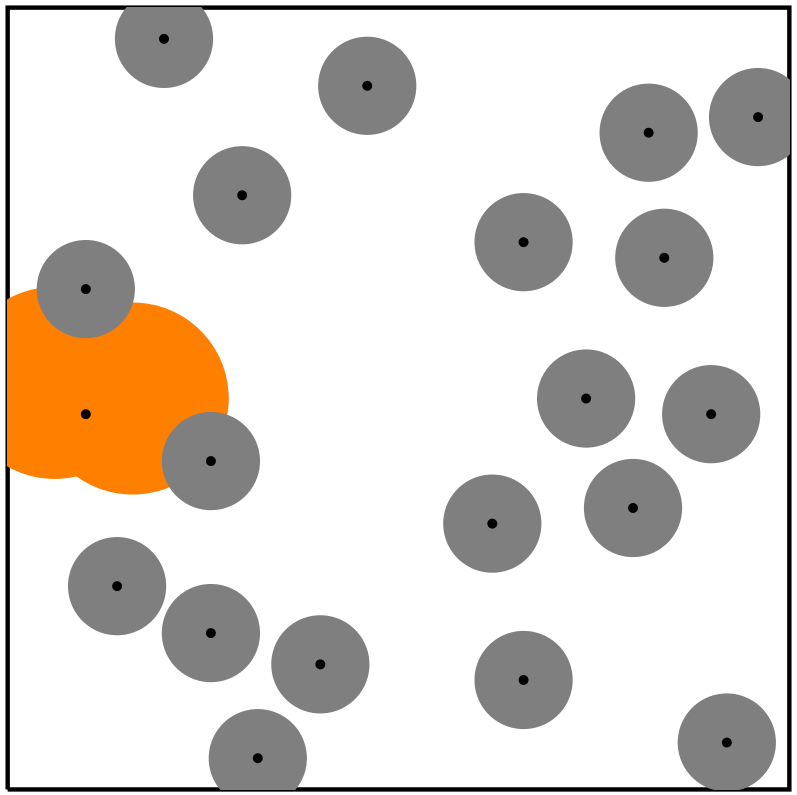
Compute the resampling set.

Partial rejection sampling applied to hard disks



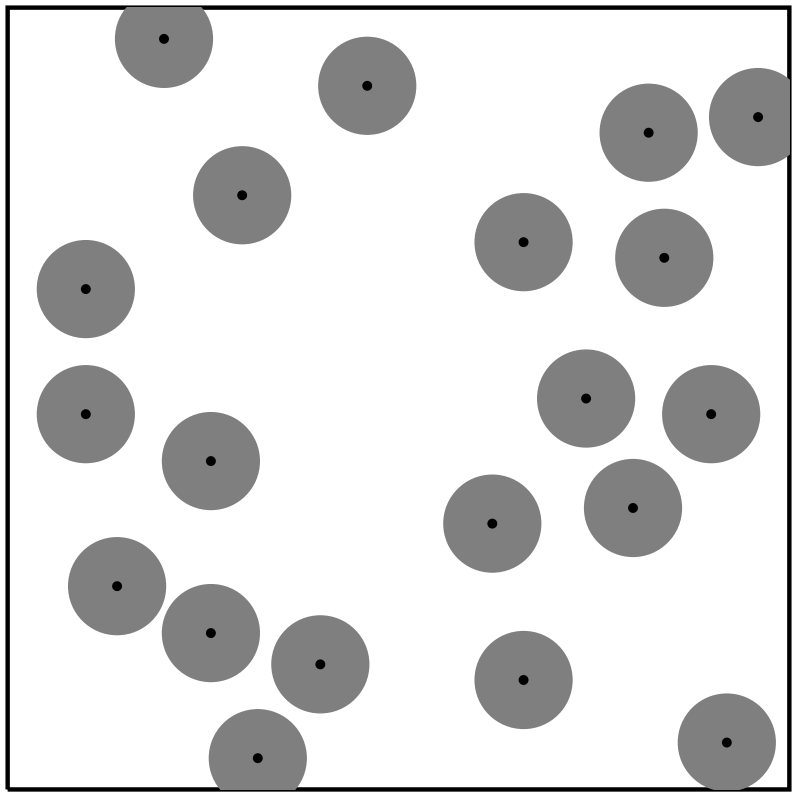
Compute the resampling set.

Partial rejection sampling applied to hard disks



Rerandomise within the resampling set.

Partial rejection sampling applied to hard disks



Final configuration.

Analysis

Theorem (Guo and Jerrum)

The above procedure is correct, and runs in $O(\log(r^{-1}))$ *iterations, provided* $\lambda < 0.21027$.

In d dimensions, the natural scaling is $\lambda_{rd} = \lambda/(v_d r^d)$ where v_d is the volume of a ball of radius 1 in $\mathbb{R}^d$.

Theorem (Guo and Jerrum)

For the hard spheres model in d *dimensions, the above procedure runs in* $O(\log(r^{-1}))$ *iterations, provided* $\lambda < 2^{-d-\frac{1}{2}}$.

New topic: Lazy Depth-First Sampling (LDFS)

Reference: Anand and Jerrum, Perfect sampling in infinite spin systems via strong spatial mixing, arXiv:2106.15992.

Suppose G is a graph. Consider some spin model on G, whose configurations are assignments $\sigma : V(G) \to Q$, where Q is a finite set of spins. Each edge contributes a weight that is a function of the spins at its endpoints. The weight of a configuration is the product of contributions from all the edges.

Example: in the case of independent sets, $Q = \{0, 1\}$ and the weight contributed by edge ij is 0 if $\sigma(i) = \sigma(j) = 1$, and 1 otherwise.

The goal

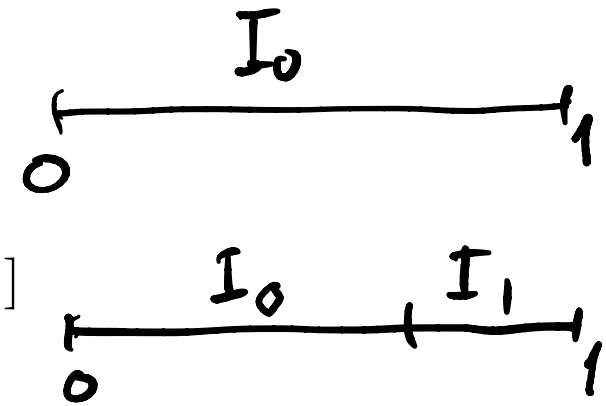
Suppose $v \in V(G)$ and (Γ, σ_Γ) is a *context*, composed of a set $\Gamma \subseteq V(G) \setminus \{v\}$ and a partial assignment $\sigma_\Gamma : \Gamma \to Q$. We wish to design a perfect sampler that will return a spin $s \in Q$ from the marginal distribution on v, conditioned on vertices in Γ having spins pinned to σ_Γ.

Towards a sampling scheme (for independent sets)

$\mathrm{LDFSample}(v, (\Gamma, \sigma_\Gamma))$
for all $u \sim v$ **do**
  if $u \notin \Gamma$ **then**
    $s := \mathrm{LDFSample}(u, (\Gamma, \sigma_\Gamma))$
    $\Gamma := \Gamma \cup \{u\}; \quad \sigma_\Gamma := \sigma_\Gamma \cup \{(u, s)\}$
  end if
end for
if $\sigma_\Gamma(u) = 1$ for some $u \sim v$ **then**
  return 0
else
  with probability $\lambda/(\lambda + 1)$ return 1 otherwise 0
end if

Towards a sampling scheme

$\mathrm{LDFSample}(\nu, (\Gamma, \sigma_\Gamma))$
for all $u \sim \nu$ **do**
  if $u \notin \Gamma$ **then**
    $s := \mathrm{LDFSample}(u, (\Gamma, \sigma_\Gamma))$
    $\Gamma := \Gamma \cup \{u\}; \quad \sigma_\Gamma := \sigma_\Gamma \cup \{(u, s)\}$
  end if
end for
if $\sigma_\Gamma(u) = 1$ for some $u \sim \nu$ **then**
  $I_0 := [0, 1]$ and $I_1 := \emptyset$
else
  $I_0 := [0, 1/(\lambda + 1)]$ and $I_1 := (1/(\lambda + 1), 1]$
end if
Let α be a realisation of a uniform $[0, 1]$ random variable
return s where $\alpha \in I_s$



Towards a sampling scheme

$\mathrm{LDFSample}(\nu, (\Gamma, \sigma_\Gamma))$
for all $u \sim \nu$ **do**
  if $u \notin \Gamma$ **then**
    $s := \mathrm{LDFSample}(u, (\Gamma, \sigma_\Gamma))$
    $\Gamma := \Gamma \cup \{u\}; \quad \sigma_\Gamma := \sigma_\Gamma \cup \{(u, s)\}$
  end if
end for
if $\sigma_\Gamma(u) = 1$ for some $u \sim \nu$ **then**
  $I_0 := [0, 1]$ and $I_1 := \emptyset$
else
  $I_0 := [0, 1/(\lambda + 1)]$ and $I_1 := (1/(\lambda + 1), 1]$
end if
Let α be a realisation of a uniform $[0, 1]$ random variable
return s where $\alpha \in I_s$

Towards a sampling scheme

LDFSample$(\nu, (\Gamma, \sigma_\Gamma))$

Let α be a realisation of a uniform $[0,1]$ random variable

for all $u \sim \nu$ **do**

  if $u \notin \Gamma$ **then**

    $s := \text{LDFSample}(u, (\Gamma, \sigma_\Gamma))$

    $\Gamma := \Gamma \cup \{u\}; \quad \sigma_\Gamma := \sigma_\Gamma \cup \{(u, s)\}$

  end if

end for

if $\sigma_\Gamma(u) = 1$ for some $u \sim \nu$ **then**

  $I_0 := [0,1]$ and $I_1 := \emptyset$

else

  $I_0 := [0, 1/(\lambda+1)]$ and $I_1 := (1/(\lambda+1), 1]$

end if

return s where $\alpha \in I_s$

$\alpha \leq \frac{1}{\lambda+1}$

Lazy Depth-First Sampler (LDFS)

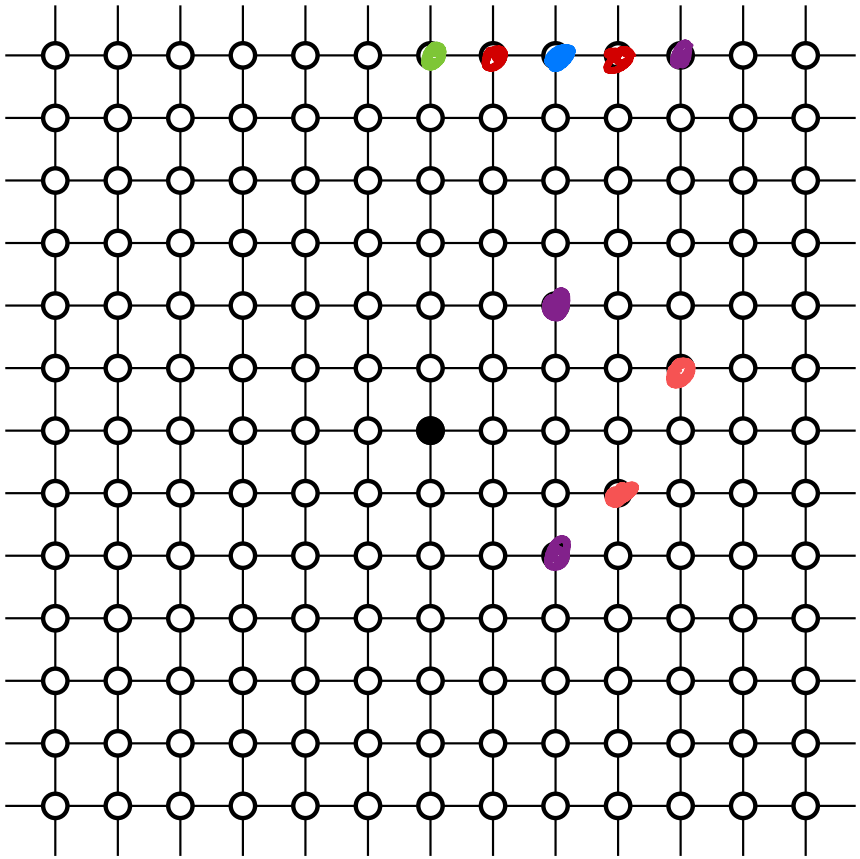
- Now observe that if $\alpha \leqslant 1/(\lambda+1)$ then the output is necessarily $0\ldots$ so we don't need to make the recursive calls.
- Suppose we apply LDFSample to a graph of maximum degree Δ. With probability $1/(\lambda+1)$ we make no recursive calls. With probability $\lambda/(\lambda+1)$ we make at most Δ recursive calls. So the expected number of recursive calls is bounded by a branching process. The mean number of offspring at a node of the tree $\lambda\Delta/(\lambda+1)$. So the expected number of recursive calls is bounded if $\lambda\Delta/(\lambda+1) < 1$, i.e., $\lambda < 1/(\Delta-1)$.
- This analysis is crude and can easily be improved. For example, if some recursive call generates a 1 we can omit the succeeding recursive calls.

Radius $r > 1$ Lazy Depth-First Sampler

- The same approach can be applied to many other sampling problems, but there are limits. For example, it does not apply to q-colouring, as any putative colour for v is eliminated by some colouring of its neighbours.
- However, we can generalise the sampling scheme just considered to radius r. In this, we recursively call the procedure on vertices at distance r. What we have seen with independent sets is the special case $r = 1$.
- Even at $r = 2$, the approach has something to say about sampling q-colourings, provided q is sufficiently large in relation to Δ.

Sampling on infinite lattices

Suppose we want to sample a uniform random 6-colouring of an infinite square lattice (or, say, a large $\mathrm{L} \times \mathrm{L}$ square region, to avoid discussing what we mean by a random colouring of an infinite graph).



Zone of indecision

- When r is large, so the influence of the vertices at radius r is small.
- Suppose that, when $r = 6$, the marginal probability at the origin v is such that $\Pr(v \text{ is red}) > 0.165$ and similarly for the other colours.
- The 'zone of indecision' then has length $1 - 6 \times 0.165 = 0.01$.
- There are just 48 vertices in the boundary, so the branching process is subcritical ($48 \times 0.01 < 1$).
- We thus would have a sampling algorithm that requires expected constant time per site.

Weak spatial mixing

- In fact, in the case of 6-colourings of a square lattice, it *is* the case that the influence of vertices at distance r declines exponentially fast with r. So the zone of indecision contracts exponentially fast with r.
- At the same time, the number of vertices at distance r is $8r$.
- So the branching process will become subcritical at *some* value of r, and we obtain a constant-time per site sampling algorithm, that works even for infinite square lattices.
- But hold on....

Strong spatial mixing

- As the sampling algorithm progresses, the environment of frozen colours grows.
- So to get things to work, we need exponential decay of correlations, even when some vertices are 'pinned' to certain colours. This is 'strong spatial mixing' and is a more elusive concept than weak spatial mixing.
- Strong spatial mixing holds for 7-colourings of the square lattice (e.g., Goldberg, Martin and Paterson), but at the time of writing this slide I don't know the status of 6-colourings.
- It is possible to get away with weak spatial mixing in this and some other similar situations (work in progress).

The ferromagnetic Ising model

An interesting example is the ferromagnetic Ising model.

Ding, Song and Sun recently (2021) settled a long-standing conjecture in the affirmative: in a strong sense, weak spatial mixing implies strong spatial mixing for the ferromagnetic Ising model. (The ferromagnetic Ising model may be essentially unique in having this strong property.)

As a consequence we can sample Ising configurations on the cubic lattice $\mathbb{Z}^{\mathrm{d}}$ whenever weak spatial mixing holds, i.e., whenever the Ising measure is defined on those lattices.

Work in progress

- When can we weaken the constraint of strong spatial mixing?
- Do we have to pay (in terms of range of validity of the sampling algorithm) for perfect sampling?
- Do we have to pay (in terms of range of validity of the sampling algorithm) for linear time (constant time per site) sampling?