## *The Problem to be Solved*

We wish to sample from some distribution over states, so that we can make a Monte Carlo estimate of the expectation of some function of state.

We have an ergodic Markov Chain whose equilibrium distribution is the one of interest. If we run it for long enough, the final state will be from a distribution close to what we want.

But how long is long enough? We can:

1. Run it until is seems like it's reached equilibrium, and pray that we're right. This is the most common approach.

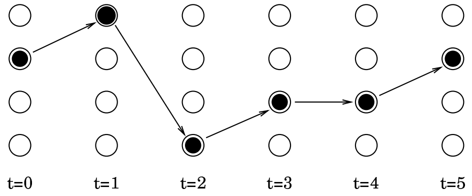
2. Prove that after $M$ steps, the distribution will be close (to a given tolerance) to the right one. This is usually very difficult.

Can we somehow obtain a sample from the *exact* distribution of interest?

# Exact Sampling Using Coupling From the Past

Propp, J. G. and Wilson, D. B. (1996) "Exact sampling with coupled Markov chains and applications to statistical mechanics", *Random Structures and Algorithms*, vol. 9, pp. 223-252.

## *Standard Forward Simulation of a Markov Chain*



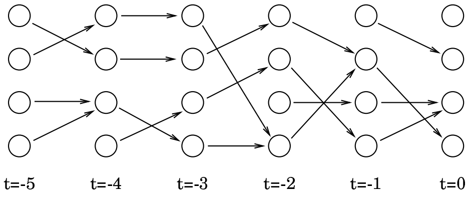
We start at $t = 0$ with a state drawn from some initial distribution (perhaps degenerate).

We hope that by $t = M$ the state comes from close to the right distribution, but we aren't too sure about this.

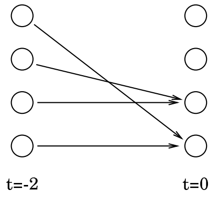
Note that we keep track of only a single state, so it may work even when the number of possible states is huge.

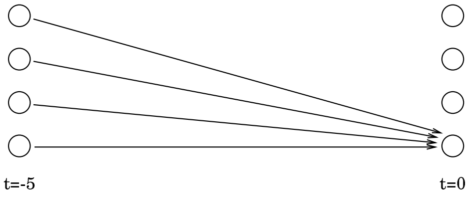
## *Tracking Mappings From Past States*

Heading back from $t = 0$, we generate random mappings from each state to its successor:



We concatenate these mappings, going back until we get one that is constant. They stay constant from then on.





## *Is This Useful?*

To find these random mappings, we need to keep track of a mapping from every state to a state at $t = 0$. This will be infeasible if the number of states is large.

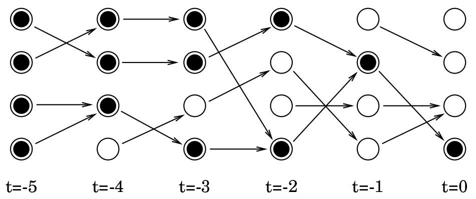
Hence, this isn't practical for most problems in statistics or statistical physics.

Neither will it be useful when the number of states is small, if we can feasibly compute the relative state probabilities — easier to just sample or compute expectations directly.

It might be useful for a system that has a small number of states, but whose state transitions are defined by a complex simulation with a huge number of intermediate states.

## *Forward Simulation From the Past*

Instead of tracking these mappings, we go forwards from some time, $t = -M$, in the past:



We keep track of the set of successors of every state we might have started from at $t = -M$.

If the set of successor states coalesces to a single state at $t = 0$, we know that starting the chain from the equilibrium distribution at $t = -\infty$ would, with these random numbers, have lead to that unique state at $t = 0$.

If we don't see coalescence at $t = 0$, we try again from further back, *using the same random numbers at times already simulated*.

## *Letting the Chains be Dependent*

To maintain validity, random mappings at different times must be independent, but it is *not* necessary for transitions from different states at the same time to be independent.

Introducing dependencies may help the chain coalesce faster, or make it easier to keep track of sets of states.

We express possible dependencies as follows:

- Let $U_t$ be the entire set of random numbers used at time $t$.
- Let the state that $i$ goes to at time $t$ be given by $\phi(i, U_t)$.

The $\phi$ function must give the right transition probabilities, but may introduce dependencies.

When simulating forward from further and further back in the past, we need to save the $U_t$ for times we've already simulated.

## *Tracking Sets of States with Monotone Markov Chains*

For forward simulation from the past to be practical when the number of states is huge, we must be able to keep track of a huge set of states with a small amount of work.

This can be done if the Markov chain is *monotone* with respect to a partial order $\leq$. That is, if

$$i \;\leq\; j \quad \text{implies} \quad \phi(i, U_t) \;\leq\; \phi(j, U_t)$$

We also assume that there are states $\hat{0}$ and $\hat{1}$ such that $\hat{0} \leq i \leq \hat{1}$ for all states $i$.

For such a chain, we can keep track of a set of states through the transitions by keeping track of just two "upper" and "lower" states:

```
upper ← $\hat{1}$
lower ← $\hat{0}$
for $t = -M$ to $-1$
    upper ← $\phi(\text{upper}, u_t)$
    lower ← $\phi(\text{lower}, u_t)$
```

## *The Heatbath (Gibbs Sampling) Method for the Ising Model*

In the Ising model, a state is a collection of "spins", $\sigma$, arranged in an $n$-dimensional array, with $\sigma_i = \pm 1$.

The probability of a state at temperature $T$ is

$$\pi(\sigma) \;=\; \frac{1}{Z}\, \text{exp}\Big( \sum_{\substack{i \text{ and } j \\ \text{neighbors}}} J\, \sigma_i\, \sigma_j \,/\, T \Big)$$

The heathbath (aka, Gibbs Sampling) method updates single spins (randomly or in a scan) by drawing from the spin's conditional distribution given other spins:

$$\begin{aligned} &\pi(\sigma_i = +1 \mid \{\sigma_j\}_{j\neq i}) \\ &\quad = \Big[ 1 + \text{exp}\Big( -2 \sum_{\substack{i \text{ and } j \\ \text{neighbors}}} J\, \sigma_j \,/\, T \Big) \Big]^{-1} \end{aligned}$$

## *Applying Propp & Wilson's Method to the Ising Model*

We can define a partial order on spin states by

$$\sigma \leq \tau \quad \text{iff} \quad \text{for all } i,\ \sigma_i \leq \tau_i$$

If $J > 0$, we can introduce dependencies to make the heatbath method monotone with respect to this order. We update $\sigma_i$ to $\sigma_i'$ using a random number $u_t \in [0, 1)$:

$$\sigma_i' \;=\; \begin{cases} +1 & \text{if } u_t < \pi(\sigma_i = +1 \mid \{\sigma_j\}_{j\neq i}) \\ -1 & \text{if } u_t \geq \pi(\sigma_i = +1 \mid \{\sigma_j\}_{j\neq i}) \end{cases}$$

This works because the system is "attractive". That is, if $\tau$ has some spins of $+1$ where $\sigma$ has spins of $-1$, but not vice versa, then the probability of an update setting $\tau_i'$ to $+1$ can't be less than that of setting $\sigma_i'$ to $+1$.

Hence we can do forward simulation starting with all $-1$ and all $+1$ states, from further and further back in time, until we see coalescence.

## *Generalizing Propp & Wilson's Method*

Use of a monotone Markov chain is not essential to the method. We require only

- A compact representation for a set of states,
- A way of finding the set of states after a transition (based on given random numbers) from the set of states before.

Perfect solutions may not always be possible, but we can use imperfect solutions that

- May not find the exact set of successor states, only a superset guaranteed to contain all the successors,
- Use a representation that we cannot prove will remain compact, but which we hope will turn out to be compact enough when we actual run things.

## *Coupling From the Past in Practice*

We could use Propp & Wilson's method to estimate an expectation in the following way:

1. Independently simulate quite a few (say 100) chains from successively further back in time, until each has coalesced at $t = 0$.
2. Do standard forward simulation for some predetermined period of time from the $t = 0$ state of all chains that coalesced.
3. Estimate the expectation of a function by its average value for states with $t \geq 0$ in all these chains.

But what if the chains don't coalesce in the time allowed?

### *Dealing with Lack of Coalescence*

Some or all of the chains we simulate may not coalesce when started back to some limit, $t = -L$. What does that mean?

- If *none* of the chains (or perhaps just one) have coalesced, the Markov chain is not close to convergence after time $L$ (or at least it doesn't couple). The method tells us this rather than giving a wrong answer.
- If *some* of the chains (a non-negligible fraction) have coalesced, then convergence is not too bad. If we go back in time a bit further, the others will coalesce too (unless we're plain unlucky, of course).

Still, we might not like to commit to an unbounded amount of computation time to get an answer.

### *What Happens if We Just Stop?*

We might consider stopping runs that have taken more time than we like, using only other runs in our estimate.

*This gives biased results.*

We *do* get an unbiased estimate if we stop after using up total time $A$, except that we continue long enough to get $S$ points regardless of how long this takes. $A$ and $S$ must be predetermined. If we stop in the middle of a run, it is discarded.

(This is a slight generalization of a result of Wilson and Fill, also discovered earlier in the operations research literature.)

We still have to continue long runs, even though they will "wastefully" have to go through many steps to get to $t = 0$.

## An Interruptible Algorithm

Fill, J. A. (1998) "An interruptible algorithm for perfect sampling via Markov chains", *Annals of Applied Probability* vol. 8, pp. 131-162.

### *Requirements*

Suppose we can simulate a Markov chain $P$, converging to $\pi$, and that we can also simulate from the *reversed* chain, $\tilde{P}$:

$$\tilde{P}(x, x') \;=\; P(x', x)\,\pi(x')/\pi(x)$$

We express $\tilde{P}$ in terms of random numbers $\tilde{U}$, which determine the path taken from each starting state (for some number of iterations).

We assume that this setup has a monotonicity property. (As with coupling from the past, generalization is possible.)

We also assume that given a starting state and a path produced by $\tilde{P}$, we can sample from the conditional distribution of $\tilde{U}$ given that path.

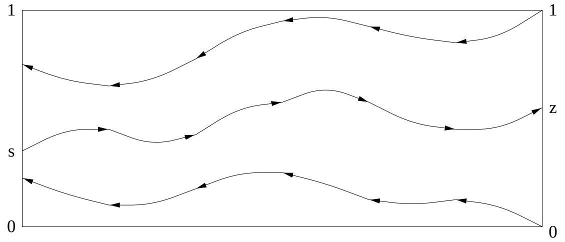
## The Algorithm

Here is the core of the algorithm, based on running for $N$ iterations:

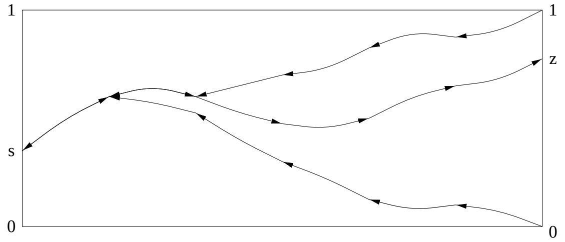
1. Run $P$ for $N$ iterations from some state $s$. Remember the path taken. Suppose it ends in state $z$.
2. Look at this path as one produced by $\tilde{P}$, starting at $z$.
3. Sample $\tilde{U}$ from its distribution given this path.
4. Run $\tilde{P}$ from $\hat{0}$ and from $\hat{1}$ using $\tilde{U}$.
5. If these two chains coalesce (at $s$), then return $z$ as a point drawn from $\pi$. If they don't, discard everything and try again.

Letting $s = \hat{0}$ saves one simulation. In practice, one would increase $N$ if the rejection rate is high.

## Picture of the Algorithm



Rejected Point



Accepted Point

## Why Does it Work?

Here's my explanation. Define:

**Distribution D1**: $\tilde{U}$ and $z$ are independent. $z \sim \pi$, $\tilde{U} \sim$ our random number generator.

Note that under D1, the end-state $s$ from apply $\tilde{P}$ from $z$ using $\tilde{U}$ also has distribution $\pi$.

**Distribution D2**: Distribution D1 conditional on $\tilde{U}$ causing $\tilde{P}$ to coalesce at a given $s$.

Note that $z \sim \pi$ under D2 still, since it's independent of $\tilde{U}$.

We can sample from D1 by running $P$ from a start state chosen from $\pi$, then sampling $\tilde{U}$ from its conditional distribution given this path.

To sample from D2, we (1) fix the start state to $s$, (2) reject if $\tilde{U}$ does not lead $\tilde{P}$ to coalesce at $s$. The $z$ obtained from an accepted point has distribution $\pi$.