A Bernoulli factory using the Fundamental Theorem of Perfect Simulation

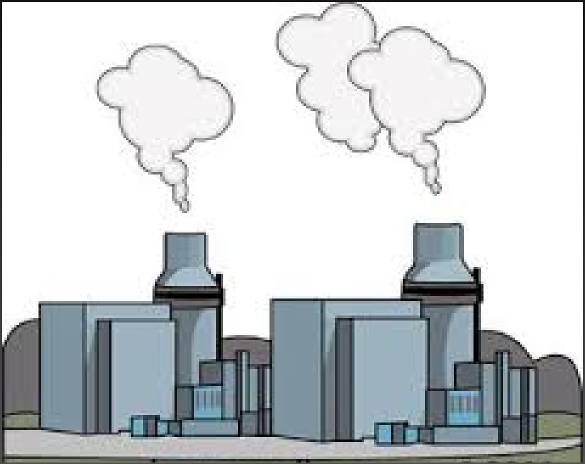
Mark Huber
Fletcher Jones Foundation Associate Professor of Mathematics and Statistics and George R. Roberts Fellow
Department of Mathematical Sciences
Claremont McKenna College
8 July, 2016

Recursion

Zen computing:

In order you understand recursion, you must first understand recursion.

Bernoulli Factory



What is a Bernoulli factory?

Suppose that I have an iid sequence of coin flips of heads or tails

H H H T H $\cdots$

Make heads $= 1$, tails $= 0$.

1 1 1 0 1 $\cdots$

Probability p that a coin flip is 1 is unknown

Bernoulli process

Mathematically,

$$B_1, B_2, \ldots \overset{\text{iid}}{\sim} \text{Bern}(p)$$

Where

$$B \sim \text{Bern}(p) \Rightarrow \mathbb{P}(B = 1) = p \text{ and } \mathbb{P}(B = 0) = 1 - p$$

An example Bernoulli Factory:

Question: Can I use these coin flips to build a new random variable

$$B \sim \text{Bern}(p(1-p))?$$

Answer: Sure! Just use

$$B = X_1(1 - X_2)$$
$$\mathbb{P}(B = 1) = \mathbb{P}(B_1 = 1)\mathbb{P}(B_2 = 0) = p(1 - p)$$

Extra randomness

$$B_1, B_2, \ldots \overset{\text{iid}}{\sim} \text{Bern}(p)$$

Question: Can I use these coin flips to build a new random variable $B \sim \text{Bern}(p/3)$?

Answer: Helpful to have some extra randomness.
Let $U \sim \text{Unif}([0,1])$ be independent of the $\{B_i\}$. Then

$$B = \mathbb{1}(U \leq 1/3)X_1$$

does the job, where $\mathbb{1}(\cdot)$ is the indicator function that is 1 if the argument is true and 0 otherwise

Bernoulli factory (informal)

Definition

A Bernoulli factory takes an iid sequence of coin flips with parameter p together with some extra randomness and builds a single coin flip with parameter $f(p)$ for a function f.

Definition

If T is the (possibly random) number of coin flips needed, then call T the running time or number of flips taken by the algorithm.

Bernoulli factory (formal)

Definition

Given $p^* \in (0,1]$ and a function $f : [0,p^*] \to [0,1]$, a Bernoulli Factory is a computable function $\mathcal{A}$ that takes as input $X_1, X_2, \ldots$ and U and returns Y such that if $X_i \stackrel{\text{iid}}{\sim} \text{Bern}(p)$ and $U \sim \text{Unif}([0,1])$, then $\mathcal{A}(U, X_1, X_2, \ldots) \sim \text{Bern}(f(p))$.

Definition

If T is a stopping time with respect to the natural filtration created by $U, X_1, X_2, \ldots$, and for all values of y_i,

$$\mathcal{A}(U, X_1, X_2, \ldots, X_T, y_{T+1}, y_{T+2}, \ldots)$$

has the same value, call T the running time or number of flips taken by the algorithm.

Bernoulli factory: origins

S. Asmussen, P. W. Glynn, and H. Thorisson, Stationarity Detection in the Initial Transient Problem, *ACM Trans. Modeling and Computer Simulation*, 2(2):130–157, 1992.

- Simulation from stationary distribution of regenerative Markov processes
- Required as subroutine ability to generate from Bernoulli factory with $f(p) = Cp$ for constant C

Bernoulli factory: next steps

M. S. Keane and G. L. O'Brien, A Bernoulli factory, *ACM Trans. Modeling and Computer Simulation*, 4:213–219, 1994.

- Introduced term Bernoulli factory
- Gave necessary and sufficient conditions on f for a Bernoulli factory to exist
- Mathematical construct rather than algorithm.
- Unknown if expected run time finite or tails heavy or light

Bernoulli factory: Bernstein connection

S. Nacu and Y. Peres, Fast simulation of new coins from old, *Ann. Appl. Probab.*, 15(1A):93–115, 2005.

- Gave method with exponential tails (so unknown if expected run time finite)
- Used Bernstein polynomials to approximate $f(p)$:

$$\sum_{i=0}^{n} a_i p^i (1-p)^i \leq f(p) \leq \sum_{i=0}^{n} b_i p^i (1-p)^i$$

- Algorithm, but required exponential time to implement
- Showed $f(p) = 2p$ sufficient to get any real analytic f

Bernoulli factory: first practical algorithm

K. Łatuszyński, I. Kosmidis, O. Papaspiliopoulos, and G. O. Roberts. Simulation events of unknown probability via reverse time Martingales, *Random Structures Algorithms*, 38:441–452, 2011.

- Practical implementation of Nacu & Peres
- Introduced reverse time Martingales technique for perfect simulation
- Numerical experiments indicated run time not linear in C

Bernoulli factory: small improvement

J. Fegal and R. Herbei, Exact sampling for intractable probability distributions via a Bernoulli factory, *Electron. J. Stat.*, 6:10–37,2012

- Changed target function slightly to improve Nacu & Peres analysis

A. C. Thomas and J. Blanchet, A practical implementation of the Bernoulli factory, arXiv:1105.2508, 2011.

Why Cp hard: Needs unbounded random number of flips

Fact

For $C > 1$, no Bernoulli factory exists for Cp that uses a finite number of flips over any nontrivial interval of p values.

Proof

After n flips there are 2^n possible outcomes. If outcome i yields a 1 (using U) with probability p_i, and has $n(i)$ heads and $n - n(i)$ tails, then the output function $g(p)$ has the form:

$$g(p) = \sum_{i=1}^{2^n} p^{n(i)}(1-p)^{n-n(i)}.$$

This is a polynomial in p, but only one polynomial equals Cp over a nontrivial interval of p values, and that is Cp. But $g(p) \in [0,1]$, so cannot equal Cp over all $p \in [0,1]$. □

Why $2p$ *hard for* $p = 1/2$

Suppose have a $2p$ Bernoulli factory

- Suppose for $X_1, X_2, \overset{\text{iid}}{\sim}$ Bern(p), $Y \sim$ Bern$(2p)$.
- Estimate p by $\hat{p}_Y = Y/2$
- If $p = 1/2$, then $\mathbb{P}(Y = 1) = 1$, $\mathbb{V}(\hat{p}_Y) = 0$!
- Not possible! (Proof: Wald's sequential ratio probability test)

Restrict domain

- Only allow $2p \in [0, 1 - \epsilon]$ so $p \in [0, 1/2 - \epsilon/2]$

General variance argument

Unbiased minimum variance estimate for p:

$$\hat{p}_n = \frac{B_1 + \cdots + B_n}{n}, \quad \mathbb{V}(\hat{p}_n) = \frac{p(1-p)}{n}$$

Suppose $Y \sim$ Bern(Cp). Then unbiased estimate for p:

$$\hat{p} = \frac{Y}{c}, \quad \mathbb{V}(\hat{p}) = \frac{p(1-Cp)}{Cn}$$

One draw of Y counts as

$$\frac{C(1-p)}{1-Cp}$$

draws from B_i

Why Cp is hard

Therefore, for p small and $1 - Cp > \epsilon$, one draw of Y should require at least

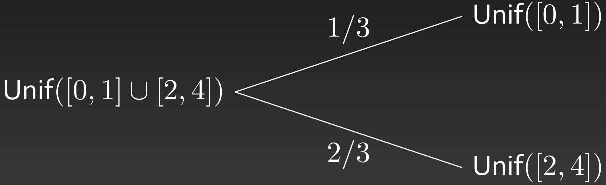
$$C\epsilon^{-1}$$

draws from original coin

Recursive Bernoulli Factories

Breaking simulations into pieces

Suppose I wish to simulate from $[0,1] \cup [2,4]$

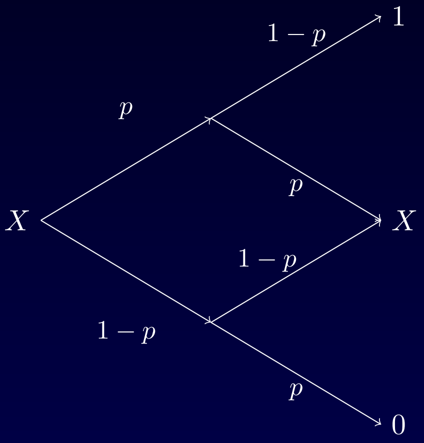


Works because

$$\text{Unif}([0,1] \cup [2,4]) \sim (1/3)\text{Unif}([0,1]) + (2/3)\text{Unif}([2,4])$$

Von Neumann's Bernoulli Factory

To flip a $X \sim \text{Bern}(1/2)$ coin



Proof of correctness

X might be 1, so let's find the probability:

$$\mathbb{P}(X = 1) = p(1-p) + (p^2 + (1-p)^2)\mathbb{P}(X = 1)$$

Solving for $\mathbb{P}(X = 1)$:

$$\mathbb{P}(X = 1) = \frac{p(1-p)}{1 - (p^2 + (1 - 2p + p^2))} = \frac{p(1-p)}{2(p)(1-p)} = \frac{1}{2}$$

Expected # of flips

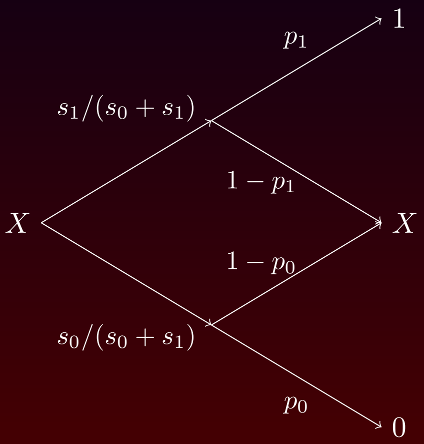
Recursive nature makes it easy to find expected # of flips:

$$\mathbb{E}[T] = 2 + [p \cdot p + (1-p)(1-p)]\mathbb{E}[T]$$

$$\mathbb{E}[T] = \frac{2}{2p(1-p)} = \frac{1}{p(1-p)}$$

Two coin algorithm [Gonçalves, Roberts, Łatuszyński. 2016]

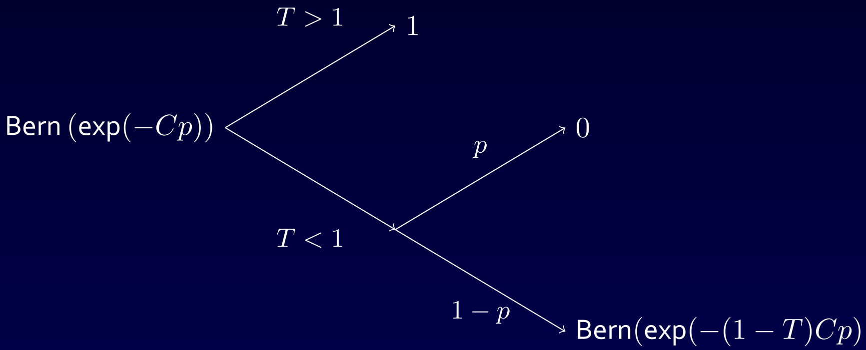
$$X \sim \text{Bern}\left(\frac{s_1 p_1}{s_0 p_0 + s_1 p_1}\right)$$



Exponential Bernoulli factory

Beskos et. al. 2006, for C a positive constant want
$X \sim \text{Bern}(\exp(-Cp))$

$$T \sim \text{Exp}(C)$$



Not a proof of correctness

Note that this tree is locally correct:

$$
\begin{aligned}
\mathbb{P}(X = 1) &= \mathbb{P}(T > 1)(1) + (1-p)\int_{t=0}^{1} C \operatorname{exp}(-Ct)(\operatorname{exp}(-(1-t)Cp))\,dt \\
&= \operatorname{exp}(-C) + (1-p)\int_{t=0}^{1} C \operatorname{exp}(-Cp)\operatorname{exp}(-tC(1-p))\,dt \\
&= \operatorname{exp}(-C) + \operatorname{exp}(-Cp) - \operatorname{exp}(-Cp)\cdot\operatorname{exp}(-C(1-p)) \\
&= \operatorname{exp}(-Cp)
\end{aligned}
$$

Had to assume that recursive call worked to prove correctness

Randomly Truncated Infinite Series

Connecting random truncation and recursion

Suppose

$$X = \sum_{i=1}^{N} X_i,$$

where $N \in \{1, 2, \ldots\}$ is a random variable

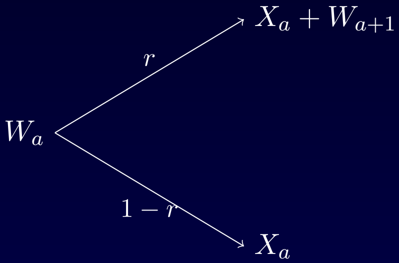
Let

$$W_a = \sum_{i=a}^{N} X_i$$

Note $X = W_1$

In recursion form…

$$r = \mathbb{P}(N \geq a+1)/\mathbb{P}(N \geq a)$$



Recursive Linear Bernoulli Factory

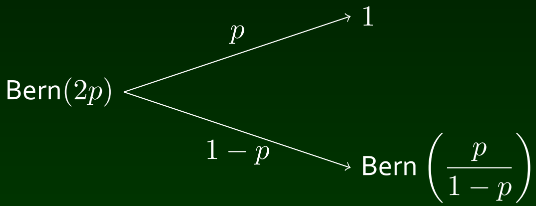
Can recursion aid in the $2p$-coin problem?

M. Huber, A Bernoulli mean estimate with known relative error distribution, *Random Structures & Algorithms*, arXiv:1309.5413, to appear

Idea:

- Break simulation problem into pieces using the p-coin
- Employ recursion to handle created subproblems

One flip of the coin

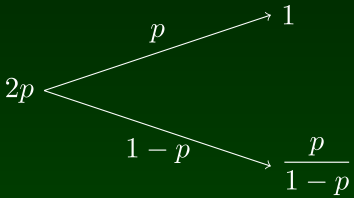


Works because (for $X \sim$ Bern$(2p)$,

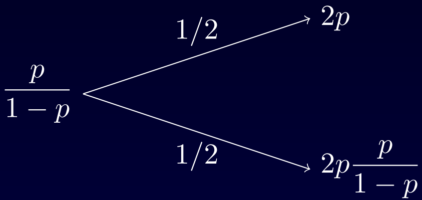
$$\mathbb{P}(X = 1) = 2p = (p)(1) + (1 - p)\left(\frac{p}{1-p}\right)$$

Shorthand

Since the only distributions we are interested here are Bernoulli, which are determined by their parameter, shorthand to write:



What to do with $p/(1-p)$



Here

$$\frac{p}{1-p} = \frac{1}{2} \cdot 2p + \frac{1}{2}(2p)\frac{p}{1-p}$$

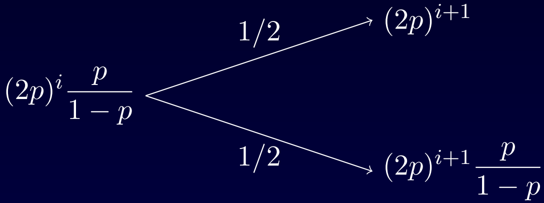
This is recursion!

We have reduced the problem of flipping a Bern$(2p)$ coin to flipping a Bern$(2p)$ coin!

Recursion: when an algorithm calls a version of itself

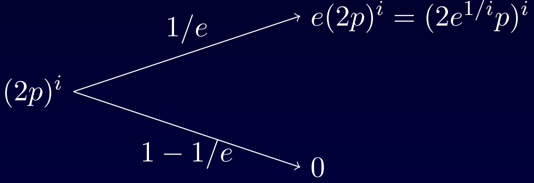
What to do with $(2p)^i p/(1-p)$*?*

For $i \in \{0, 1, \ldots\}$



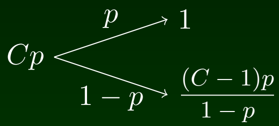
Large i

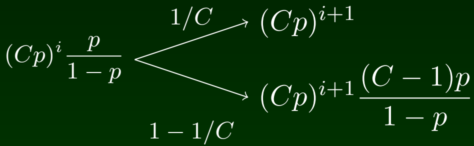
Since $2p \leq 1 - \epsilon$, $(2p)^i \to 0$ as $i \to \infty$:



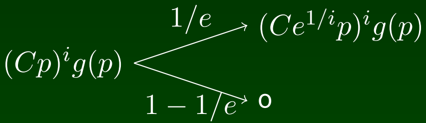
Total algorithm in pictures

To draw $f(p) = Cp$ for constant C, $Cp \leq 1 - \epsilon$





Run the above until terminates at 1 or $i > 4.6/\epsilon$. Then:



Update: $\epsilon \leftarrow 1 - e^{1/i}(1-\epsilon)$, $C \leftarrow Ce^{1/i}$, continue until halts

Is this algorithm correct?

Reasons to doubt algorithm

- Algorithm calls itself recursively with larger value of C
- C is unbounded
- Is that legal?

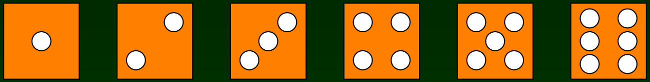
In original paper, proved correctness

- Repeated arguments made in other perfect simulation algorithms

Recursion and Perfect Simulation

An example

Suppose that $X \sim \text{Unif}(\{1, 2, 3, 4, 5, 6\})$.



I can roll as many dice (iid) as I'd like

I'd like $X \sim \text{Unif}(\{1, 2, 3, 4, 5\})$

Acceptance Rejection

The idea:

- Roll the die once
- If it falls in $\{1, 2, 3, 4, 5\}$, accept as draw from Unif$(\{1, 2, 3, 4, 5\}$
- Otherwise, start algorithm over again.

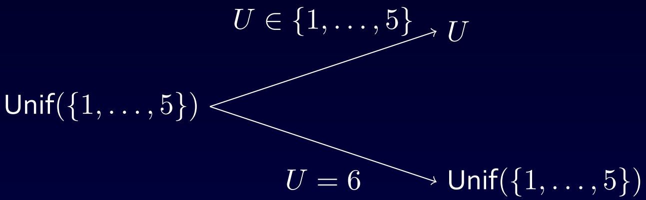
In pseudocode:

function `draw_x_5`

1. Draw $X \leftarrow$ Unif$(\{1, 2, 3, 4, 5, 6\})$
2. If $X \in \{1, 2, 3, 4, 5\}$, then return X and halt
   Else $X \leftarrow$ `draw_x_5`, return X and halt

Algorithm in pictures

Draw $U \leftarrow$ Unif$(\{1, \ldots, 6\})$



When an algorithm calls itself, call it **recursion**.

Proof that the algorithm works

Consider X the output of the algorithm. Then:

$$\mathbb{P}(X=3) = \underbrace{\frac{1}{6}}_{\mathbb{P}(U=3)} + \underbrace{\frac{1}{6}}_{\mathbb{P}(U=6)} \underbrace{\mathbb{P}(X=3)}_{\text{recursion}}$$

Solve to get

$$\mathbb{P}(X=3) = \frac{1}{5}$$

Definition

A **perfect simulation** *is an exact simulation that employs recursion.*

Coupling from the past

J. G. Propp and D. B. Wilson, Exact sampling with coupled Markov chains and applications to statistical mechanics, *Random Structures & Algorithms*, 9(1–2):223–252, 1996

Definition

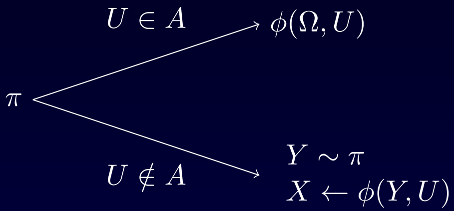
For a distribution ϕ, ay that $\phi : \Omega \times [0,1] \to \Omega$ is a stationary update function if for $X \sim \pi$ and $U \sim$ Unif$([0,1])$, $\phi(X, U) \sim \pi$.

Definition

Call $A \subseteq [0,1]$ coalescent if for all $u \in A$, $\phi(\Omega, u)$ is a single element set.

Algorithm in pictures

Draw $U \leftarrow$ Unif$([0,1])$



Let $[\pi|U]$ be the distribution of $\phi(X,U)$ where $X \sim \pi$ and $U \sim$ Unif$([0,1])$. Then this works because

$$\pi \sim [\pi|U \in A]\mathbb{P}(U \in A) + [\pi|U \notin A]\mathbb{P}(U \notin A)$$

What do these have in common?

Acceptance/rejection, CFTP, recursive Bernoulli factory

- All use recursion
- All easy to prove correct if can assume recursive call is correct
- All actually are correct (if halt with probability 1)

Properties of a fundamental theorem

- Should explain a wide range of phenomenon
- Should be obvious when looked at in the right way
- Does not cover everything in area

Some examples

Fundamental Theorem of Simulation

Most problems reduce to uniform random variables.

Fundamental Theorem of Markov chains

Under mild conditions, as you take more steps in a Markov chain, you approach the stationary distribution of the chain.

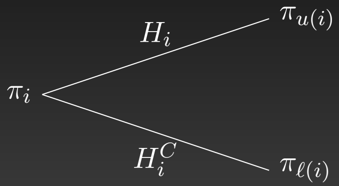
Informal version 1

Theorem (Fundamental Theorem of Perfect Simulation)

In proving an algorithm's correctness, you can assume that your recursive call to your probabilistic algorithm gives the correct result, assuming that the algorithm halts with probability 1.

Another way of viewing recursion

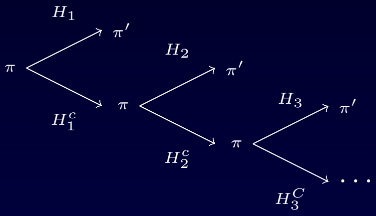
Each level of algorithm splits target into two possibilities



With recursion, gives rise to an infinite tree

Infinite tree for acceptance/rejection example

Let $\pi \sim$ Unif$(\{1,\ldots,6\})$, $\pi' \sim$ Unif$(\{1,\ldots,5\})$.
Let H_i be event $U_i \in \{1,2,3,4,5\}$



The π' nodes are halting nodes

Infinite tree version of FTPS

Theorem (Fundamental Theorem of Perfect Simulation)

Suppose for all nodes i, that

$$\pi_i \sim \mathbb{P}(H_i)\pi_{u(i)} + \mathbb{P}(H_i^C)\pi_{\ell(i)},$$

and that the probability of reaching a halting node is 1. The output of the algorithm is the distribution of the starting node.

Proof of FTPS

- Call the original algorithm $\mathcal{A}$, and its output X.
- Suppose algorithm $\mathcal{A}_i$ is just like $\mathcal{A}$, but if you get to node i, just output $\perp$ and quit. Call its output X_i.
- Then use local correctness to show by induction that for all measurable D,

$$\mathbb{P}(X_i \in D) \leq \pi(D) \leq \mathbb{P}(X_i \in D) + \mathbb{P}(\text{reach node } i)$$

- The probability that the algorithm halts with probability 1 gives that $\mathbb{P}(X_i \in D)$ coverges to $\mathbb{P}(X \in D)$ and the inequality above gives that it converges to $\pi(D)$. Hence $\mathbb{P}(X \in D) = \pi(D)$.

Perfect simulation pseudocode

Instead of infinite tree, can use recursion to describe:

$\texttt{PS}(\pi)$

1. Draw $U \leftarrow$ Unif$([0, 1])$
2. If $U \in A$ return $g(U)$ and halt
3. Else recursively draw $Y \leftarrow \texttt{PS}(\pi')$, return $g(Y, U)$ and halt

Recursion point of view

Theorem (Fundamental Theorem of Perfect Simulation)

Suppose that for all measurable sets B*,*

$$\mathbb{P}(X \in B) = \mathbb{P}(U \in A)\mathbb{P}(g(U) \in B | U \in A) \\ + \mathbb{P}(U \notin A)\mathbb{P}(g(Y, U) \in B | U \notin A).$$

where $X \sim \pi$ *and* $Y \sim \pi'$*.*
If the probability that $\mathtt{PS}(\pi)$ *eventually halts is 1, then the output of* $\mathtt{PS}$ *has distribution* π*.*

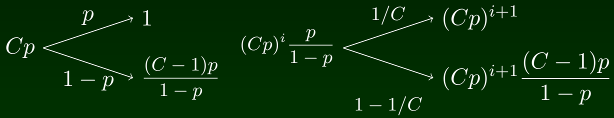
Back to Bernoulli Factory!

Local correctness

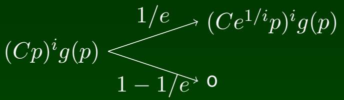
Recall the recursive Bernoulli factory...

Total algorithm in pictures

To draw $f(p) = Cp$ for constant C, $Cp \leq 1 - \epsilon$



Run the above until terminates at 1 or $i > 4.6/\epsilon$. Then:



Update: $\epsilon \leftarrow 1 - e^{1/i}(1 - \epsilon)$, $C \leftarrow Ce^{1/i}$, continue until halts

Applying FTPS

Recursive BF has nice properties

- Local correctness comes from design
- Local correctness also means parameter is a martingale
- Bounded martingales are uniformly integrable, so Martingale Convergence Theorem says it converges with probability 1
- Only way parameter converges is when it moves to 0 or 1, that is, convergence of martingale = algorithm terminates

Recursive view also helps in analyzing running time

By making algorithm recursive, aids in bounding $\mathbb{E}[T]$.

Theorem

The expected number of flips for the recursive Bernoulli factory is bounded above by

$$9.5C\epsilon^{-1}$$

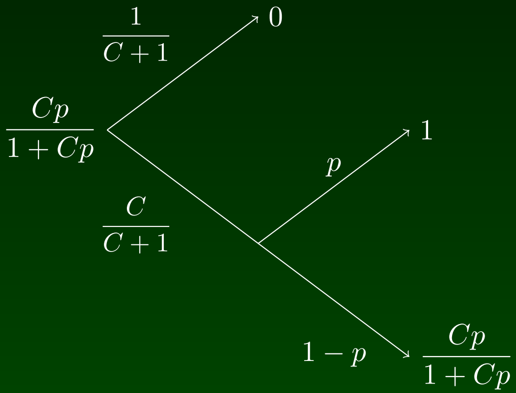
Order of the run time is correct, constant is not.

Small Cp Bernoulli Factory

M. Huber, Optimal Bernoulli factories for small mean problems
arXiv:1407.00843

Getting close to optimal for small Cp

Can use recursion to get $Cp/(1+Cp)$ coin:



Correctness

Use FTPS

$$\frac{Cp}{1+Cp} = (0)\frac{1}{C+1} + \frac{C}{C+1}\left[p + (1-p)\frac{Cp}{1+Cp}\right] \quad \checkmark$$

Also, $1/(C+1)$ branch ensures that algorithm terminates in finite time with probability 1

Run time

Recursive form makes it easy to determine run time

$$\mathbb{E}[T] = \frac{C}{C+1}\left[1 + (1-p)\mathbb{E}[T]\right]$$

$$\vdots$$

$$\mathbb{E}[T] = \frac{C}{1+Cp}$$

Using this new coin

Taking advantage of small mean coins

- If Cp is small, then $Cp/(1 + Cp)$ is just slightly smaller than Cp
- Say $Cp \leq M$
- Then if $\beta \leq (1 - 2M)^{-1}$, then $Cp\beta(1 + \beta Cp)^{-1} \geq Cp$

The small mean algorithm

Input: M which is an upper bound on Cp

1. $\beta \leftarrow (1 - 2M)^{-1}$
2. Draw $Y \leftarrow \text{Bern}(\beta Cp(1 + \beta Cp)^{-1})$
3. Draw $B \leftarrow \text{Bern}(1/\beta)$
4. If $Y = 0$ then $X \leftarrow 0$
5. Elseif $Y = 1$ and $B = 1$, then $X \leftarrow 1$
6. Else $X \leftarrow \text{Bern}([\beta C(\beta - 1)^{-1}]p)$

The last line can be accomplished using our original method

The running time

Theorem

For $Cp \leq M < 1/2$, it requires at most (on average)

$$\frac{C}{(1-2M)(1+Cp)} + Cp \cdot \left[19C \frac{1}{1-2M+Cp}\right]$$

coin flips.

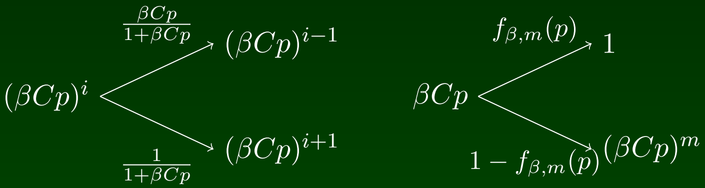
Note for Cp is small and M bounded away from $1/2$, the second term is small

Current fastest all ϵ Bernoulli Factory

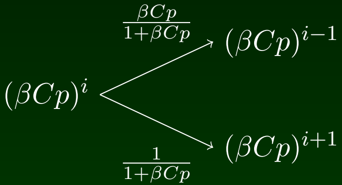
Subroutines

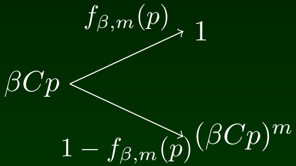
Let $\beta \in [1, 1/(1-\epsilon))$, then can make coins with mean

$$\frac{\beta C p}{1 + \beta C p}$$



Use to find $f_{\beta,m}(p)$





$$\beta Cp = f_{\beta,m}(p)(1) + (1 - f_{\beta,m}(p))(\beta Cp)^m$$

$$f_{\beta,m}(p) = \beta Cp \frac{1 - (\beta Cp)^{m-1}}{1 - (\beta Cp)^m}$$

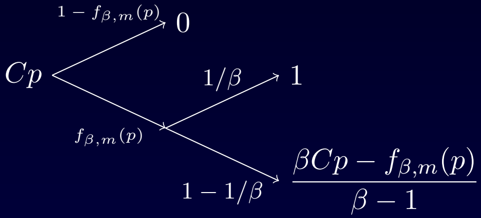
Why is this helpful?

For $\beta > 1$:

$$\frac{f_{\beta,m}(p)}{\beta} \leq Cp \leq f_{\beta,m}(p)$$

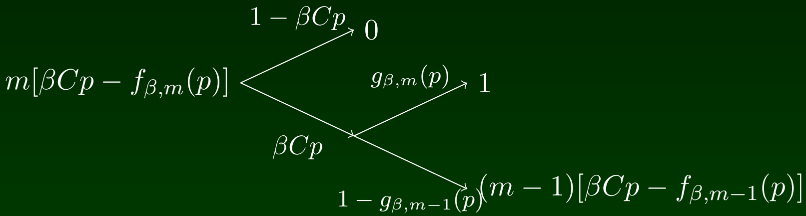
This inequality can then be turned into an algorithm

Turning the inequality into an algorithm



Let $\beta = 1 + 1/m$ so $1/(\beta - 1) = m$.

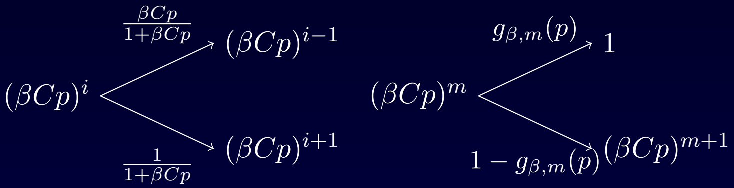
Reducing m



where

$$g_{\beta,m}(p) = \frac{(\beta C p)^m}{1 + \cdots + (\beta C p)^m}$$

Breaking the last coin apart



The result

Theorem

The mean number of coin flips used by the r based algorithm is bounded above by

$$7.57C\epsilon^{-1}$$

What is Retrospective Monte Carlo?

Narrow view

Special case of acceptance/rejection where only part of the random variate need be generated to determine if acceptance or rejection occurs.

Broad view

By rearranging the order in which you utilize randomness, sometimes recursion is made unnecessary.