Perfect simulation for repulsive point processes

Why swapping at birth is a good thing

Mark Huber

Department of Mathematics
Claremont-McKenna College

20 May, 2009

In a world with limited resources...

Competition is everywhere!

Competition is everywhere

Towns compete for space

Trees compete for sunlight

Effects of competition

For spatial data...

- points look like they are repelling one another
- more regularly spaced than if locations independent

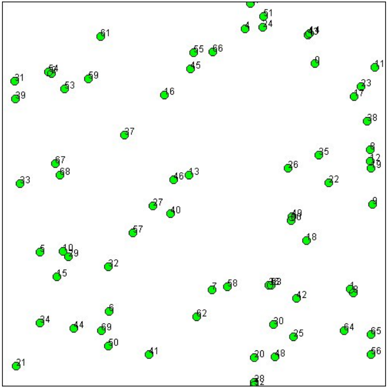
Outline

1. Modeling repulsive point processes
   - Spatial point processes
   - The Hard Core Gas Model
2. Birth Death Chains
   - Using Markov chains
   - Standard Birth Death Approach
   - New move: swapping at birth
3. Perfect Sampling
   - Dominated Coupling From the Past
   - dCFTP with standard chains
   - dCFTP with the swap
4. Results

Ways to model repulsion

Create Density with respect to Poisson point process

Poisson point process:



Space S

Intensity measure $\lambda \cdot \mu(\cdot)$

For $A \subseteq S$, $\mathbb{E}[A] = \lambda \cdot \mu(A)$

Generating from a Poisson point process

Two-step process:

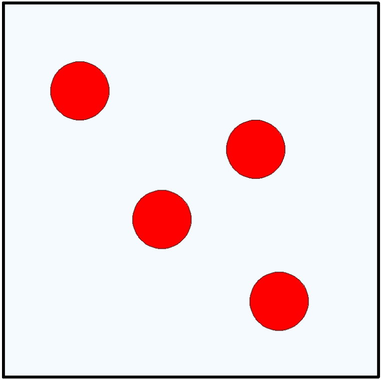
[1] Generate $N \sim \text{Poisson}(\mu)$

[2] Generate $X_1, X_2, \ldots, X_N$ independently on S using μ

Note when μ is Lebesgue measure, points are uniform on S

The Hard Core Gas Model

Point processes used to model gases
Each point center of hard core of molecule
“Hard” core means they do not overlap



Densities for Poisson point processes

Formally, use densities to force the constraints
Let $\#x$ denote number of points in a configuration

Hard Core:

$$f_{hardcore}(x) \propto \begin{cases} 1 & \text{dist}(x(i), x(j)) > R \text{ for all } i, j \in \{1, \ldots, \#x\} \\ 0 & \textit{otherwise} \end{cases}$$

Soft Core (Strauss Point Process)

$$\begin{aligned} f_{softcore}(x) &\propto \gamma^{n(x)} \\ n(x) &= \text{number of pairs } \{i, j\} \text{ with } \text{dist}(x(i), x(j)) \leq R \end{aligned}$$

Normalizing densities

What makes these problems difficult?

- Note $\propto$ in density descriptions
- Need to multiply by constant to make probability density
- Called the normalizing constant

The difficulty:

- Finding normalizing constant for general state spaces is a #P-complete problem
- Often referring to in literature as “intractable"

Simple perfect simulation: Acceptance/Rejection

Acceptance/Rejection for Strauss process

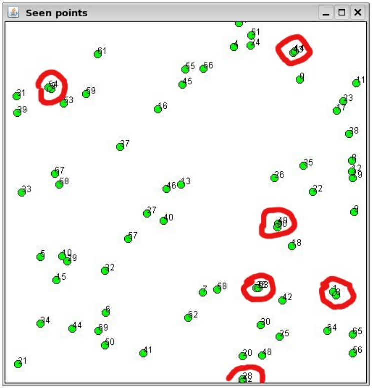
repeat
  draw X as Poisson point process on S
  draw U uniformly on $[0, 1]$
until $U \leq \gamma^{v(X)}$

The resulting X is a draw from the Strauss process density

Example

Strauss process (1975)

$$\gamma := \text{repulsion parameter in } (0,1)$$
$$R := \text{radius of interaction}$$
$$f(x) = \gamma^{\#\{(i,j):\text{dist}(x_i,x_j)<R\}}/Z$$



$R = .02$

$f(x) = \gamma^6 \,/\, Z$

Why not use A/R all the time?

Main drawbacks

- Only works when density is bounded
- Running time usually exponential in λ

Solution

- Use Markov chains
- Small random changes
- Add up over time

Outline

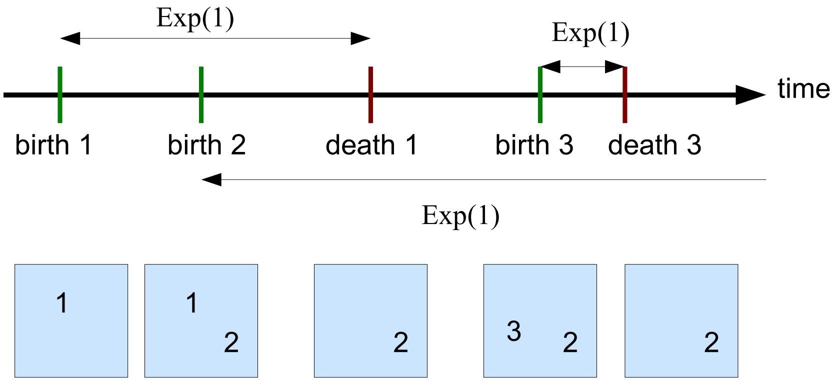
Birth Death chains

Jump processes:

- Points born at times given by 1-dimensional Poisson process
- The rate of births is $\lambda \cdot \mu(S)$
- When point born, decide “lifetime" that is exp(1)
- After lifetime, the point dies and is removed from process

Stationary distribution Poisson point process

Illustration of birth death chain



Purpose of birth death chains

Metropolis-Hastings

- Birth Death process plays role of proposal chain
- Preston's [3] approach: always accept deaths
- Only sometimes accept briths

By only accepting some births...

- Ensures jump process equivalent of reversibility
- Works for locally stable densities

Locally stable

Definition (Locally stable)

Call a density *locally stable* if there is a constant K such that for all sets of points x and points v we have $f(x + v) \leq Kf(x)$.

Preston's method to construct jump process:

- Death rate always 1, birth rate K
- Accept births with probability $[f(x + v)/f(x)]/K$

Reversibility:

$$f(x)b(x, v) = f(x + v)d(v)$$

Examples

Hard core gas model (μ is Lebesgue)

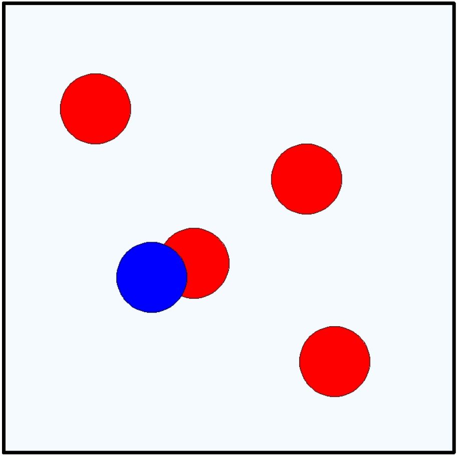
- $f(x+v)/f(x) \in \{0, \lambda\}$, so $K = \lambda$
- Birth of v to x accepted if no point of x is within R distance of v

Strauss process (μ is Lebesgue)

- $f(x+v)/f(x) \in \{0, \lambda\}$, so $K = \lambda$
- Let $n(v,x)$ be the # of points in x distance R of point v
- Probability accept birth of v to x: $\gamma^{n(v,x)}$

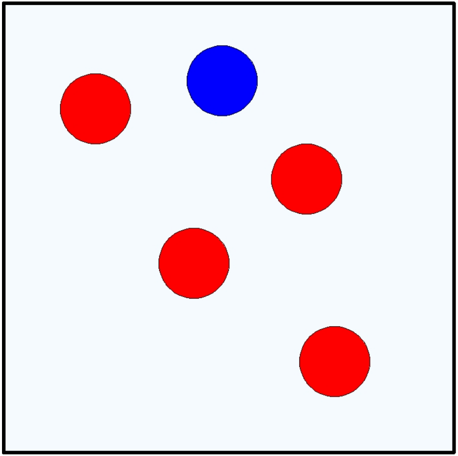
Hard core model: example of rejected birth

New point (in blue) is rejected
Too close to existing points



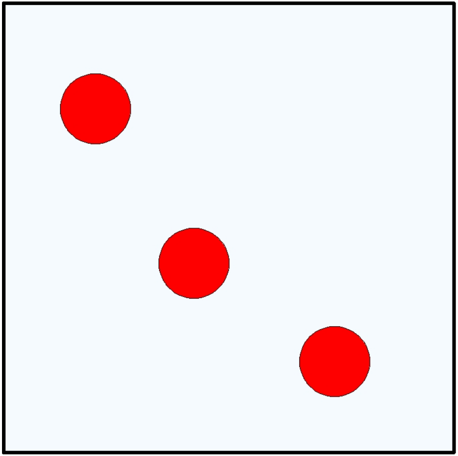
Example of accepted birth

New point (in blue) is accepted an added to configuration
Too close to existing points



Example of death

Deaths are always accepted
(Removing point never violates hard core constraint)



To speed up chain, add a move

Old moves

- Birth: addition of point
- Death: removal of point

New move

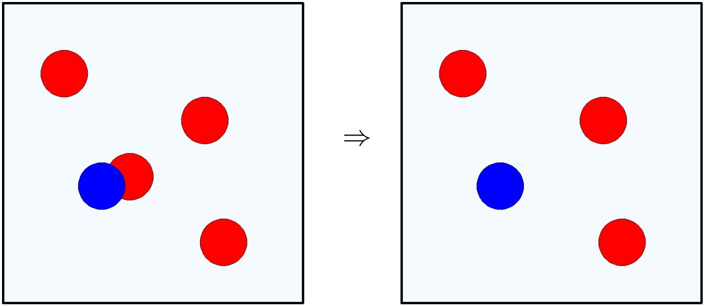
- Swap: addition and removal happen simulataneously

History

- Used in discrete context by Broder (1986) for perfect matchings
- Used for discrete hard core processes by Luby & Vigoda (1999)

Example of swap for hard core gas model

When blocked by exactly one point, “swap” with blocking point:



Some details

Things to consider:

- Does swapping give correct distribution?
- Does it improve performance in a theoretical way?
- Does the swap move generalize?

Preprint: Huber [2]

- Can set up probability of swapping to give correct distribution
- Current state x, add v remove w at rate $s(x, w, v)$

$$f(x)s(x, w, v) = f(x + v - w)s(x + v - w, v, w)$$

- Only swaps when reject birth
- Does work faster than original chain
- Speeds up perfect simulation algorithm

Some details

Things to consider:

- Does swapping give correct distribution?
- Does it improve performance in a theoretical way?
- Does the swap move generalize?

Preprint: Huber [2]

- Can set up probability of swapping to give correct distribution
- Current state x, add v remove w at rate $s(x, w, v)$

$$f(x)s(x, w, v) = f(x + v - w)s(x + v - w, v, w)$$

- Only swaps when reject birth
- Does work faster than original chain
- Speeds up perfect simulation algorithm

Example: Strauss process

One method to build swaps is to swap exactly when one point is “blocking" the birth point:

Birth for Strauss (no swap)

draw $v \leftarrow \mu$
draw U_w iid from $[0, 1]$ for each $w \in x$ with $\text{dist}(v, w) \leq R$
If $U_w \leq \gamma$ for all w, add v to x

Birth for Strauss (swap allowed)

draw $v \leftarrow \mu$
draw U_w iid from $[0, 1]$ for each $w \in x$ with $\text{dist}(v, w) \leq R$
If $U_w \leq \gamma$ for all w, add v to x
If $U_w > \gamma$ and $U_{w'} \leq \gamma$ for all $w' \neq w$, then add v and remove w from x

Why use such a complex swap move?

Several benefits

- Better analysis of mixing time through coupling
- Faster simulation from density using perfect simulation techniques

Outline

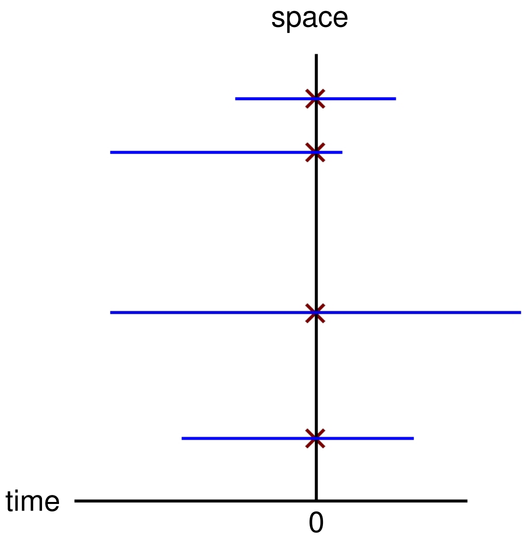
Perfect Sampling

"Practice makes perfect, but nobody's perfect, so why practice?"

Problem with Markov chains

- How long should they be run?
- Perfect sampling algorithms share good properties of Markov chains...
- ...but terminate in finite time (with probability 1)

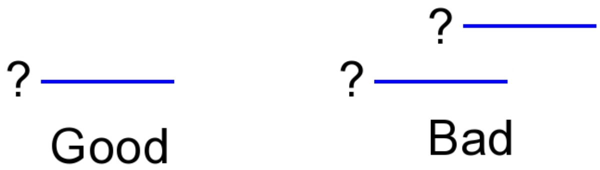
One dimensional Poisson process



Dominated Coupling From the Past

Kendall and Møller [1]: DCFTP for locally stable processes

- Say we don't know if a point should be in the set or not
- If it dies, great!
- If point born within range before it dies, bad



Dominated Coupling From the Past (part 2)

Start at fixed time in the past with some unknowns

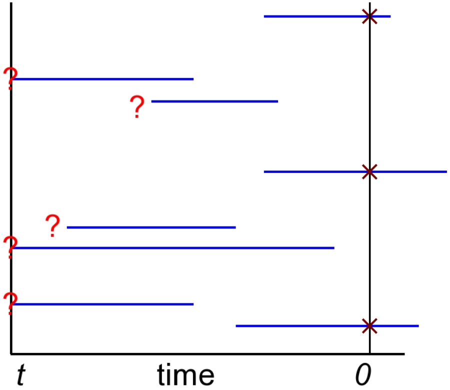
- Run forward up until time 0
- If no “?” points, quit, return sample
- Otherwise go farther back in time and begin again

Theorem

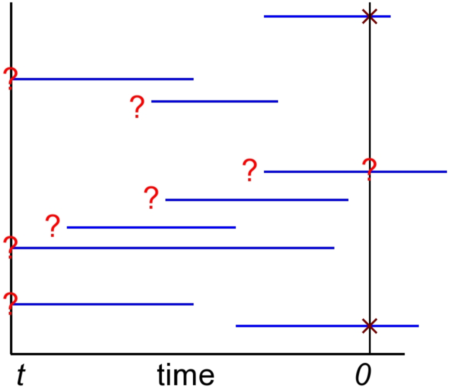
This actually works!

(Wilson called Coupling Into and From the Past)

Example where “?” go away



Example where "?" do not go away



How to update "?"

Notation:

$$
\begin{aligned}
L &= \{\text{ points definitely in process}\} \\
U &= L \cup \{\text{ ? points}\}
\end{aligned}
$$

Note that $L \subseteq U$ and when $L = U$ there are no ? points The hard part of DCFTP

- Updating the L and U processes correctly

Pseudocode for updating "?"

Updates for regular birth-death chain

Hard core bounding process update

Input: move, L, U, *Output:* L,U

1) **If** *move* $=$ death of point w
2) **Let** $L \leftarrow L - w$, **let** $U \leftarrow U - w$
3) **Else** (*move* $=$ birth of point v)
4) **Let** $N_U \leftarrow \{w \in U : \rho(w, v) \leq R\}$
5) **Let** $N_L \leftarrow \{w \in L : \rho(w, v) \leq R\}$
6) **Execute** one of the following cases:
7) **Case I:** $|N_U| = |N_L| = 0$, **let** $L \leftarrow L + v$, **let** $U \leftarrow U + v$
8) **Case II:** $|N_U| \geq 1$, $|N_L| = 0$, **let** $U \leftarrow U + v$

When is procedure fast?

The ? are like an infection

- Die out when average # of children < 1
- Let a be area of ball of radius R
- Average # children before death is λa

When are you guaranteed good performance?

Theorem (Huber [2])

Suppose that N events are generated backwards in time and then run forward to get $U_N(0)$ and $L_N(0)$. Let $B(v,R)$ denote the area within distance R of $v \in S$, let $a = \sup_{v \in S} \cdot B(v,R)$, and suppose $\mu \geq 4$. If $\lambda a < 1$, then for the chain without the swap move

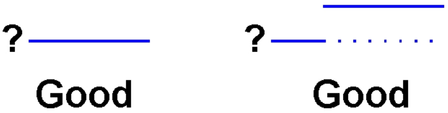
$$\mathbb{P}(U_N(0) \neq L_N(0)) \leq 2\mu(S)\exp(-N(1-\lambda a)/(10\mu(S)). \tag{1}$$

Corollary

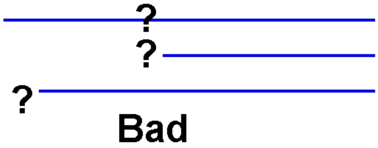
Running time of dCFTP is $\Theta(\mu(S)\ln\mu(S))$ for λ small.

Swap moves can help or hurt...

Situation 1: When only a single ? in range, swapping helps:



Situation 2: When more than one neighbor in range, swapping hurts:



Sometimes swapping

When given opportunity to swap:

- Execute swap with probability p_{swap}
- Otherwise no swap

Set $p_{\text{swap}} = 1/4$**:**

- Situation 1: +1 ?'s with prob 3/4, -1 ?'s with proba $1/4$
- Situation 2: 0 ?'s with prob 3/4, +2 ?'s with prob $1/4$
- Either way: rate of ?'s is $(+1)(3/4) + (-1)(1/4) = 2(1/4) = 1/2$

Effectively, ?'s born at half the rate they were with no swap move

Pseudocode for updating “?” with swap

Updates for regular birth-death chain

Hard-core bounding process update

Input: move, L, U, *Output:* L,U

1) **If** $move =$ death of point w
2) **Let** $L \leftarrow L - w$, **let** $U \leftarrow U - w$
3) **Else** ($move =$ birth of point v)
4) **Let** $N_U \leftarrow \{w \in U : \rho(w, v) \leq R\}$
5) **Let** $N_L \leftarrow \{w \in L : \rho(w, v) \leq R\}$
6) **Execute** one of the following cases:
7) **Case I:** $|N_U| = |N_L| = 0$, **let** $L \leftarrow L + v$, **let** $U \leftarrow U + v$
8) **Case II:** $|N_U| = 1$, **let** $L \leftarrow L + v - N_L$, **let** $U \leftarrow U + v - N_U$
9) **Case III:**$|N_U| > 1$, $|N_L| = 0$ **let** $U \leftarrow U + v$
10) **Case IV:** $|N_U| > 1$, $|N_L| = 1$, **let** $L \leftarrow L - N_L$, $U \leftarrow U + v + N_L$
11) **Case V:** $|N_U| > 1$, $|N_L| > 1$ (do nothing)

When are you guaranteed good performance?

Theorem (Huber [2])

Suppose that N events are generated backwards in time and then run forward to get $U_N(0)$ and $L_N(0)$. Let $B(v, R)$ denote the area within distance R of $v \in S$, let $a = \sup_{v \in S} \lambda \cdot B(v, R)$, and suppose $\mu \geq 4$. If $\mu a < 2$, then for the chain where a swap is executed with probability $1/4$,

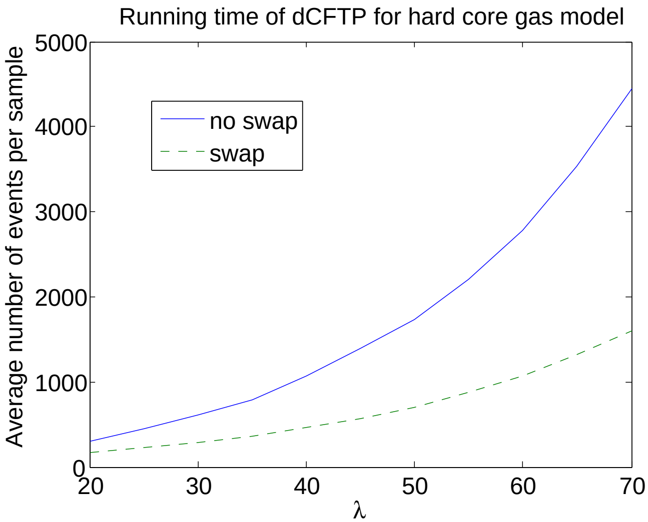
$$\mathbb{P}(U_N(0) \neq L_N(0)) \leq 2\mu \exp(-N(1 - .5a)/(30\mu). \tag{2}$$

Corollary

Running time of dCFTP is $\Theta(\mu \ln \mu)$ for λ twice as large as without the swap.

Outline

Running time results



Conclusions about swap move

What is known:

- Swap move easy to add to point processes
- Also can be used in dCFTP to get perfect sampling algorithm
- Results in about a 4-fold speedup for hard-core gas model

Future work:

- Running time comparison for Strauss process
- Improvement near phase transition
- Experiment better than theory–can theory be improved?

References

W.S. Kendall and J. Møller.
Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes.
Adv. Appl. Prob., 32:844–865, 2000.

M. L. Huber.
Spatial Birth-Death-Swap Chains
preprint, 2007

C.J. Preston.
Spatial birth-and-death processes.
Bull. Inst. Int. Stat., 46(2):371–391, 1977.