Subproblem pseudomarginal reversible jump MCMC in probabilistic programming languages

Marco Cusumano-Towner Alexander K. Lew Vikash K. Mansinghka
MIT Probabilistic Computing Project

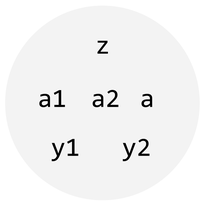
www.gen.dev

1. Address and traces in Gen

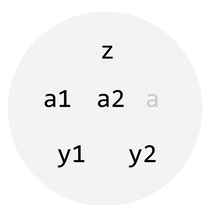
A probabilistic program

```
@gen function foo()
    if ({:z} ~ bernoulli(0.5))
        a1 = ({:a1} ~ gamma(1, 1))
        a2 = ({:a2} ~ gamma(1, 1))
    else
        a = ({:a} ~ gamma(1, 1))
        (a1, a2) = (a, a)
    end
    {:y1} ~ normal(a1, 0.1)
    {:y2} ~ normal(a2, 0.1)
end
```

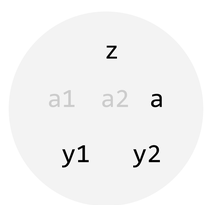
All random choice addresses



Addresses sampled if z = true

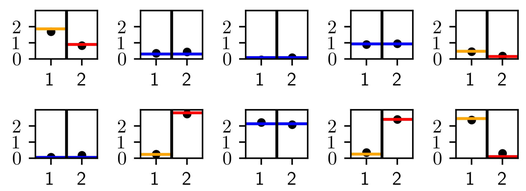


Addresses sampled if z = false



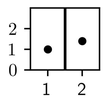
Some traces of the probabilistic program

$$
\begin{array}{llll}
(\texttt{z} = \text{true}, & \texttt{a1} = 1.87, \texttt{a2} = 0.89, & \texttt{y1} = 1.69, & \texttt{y2} = 0.82) \\
(\texttt{z} = \text{false}, & \texttt{a} = 0.32, & \texttt{y1} = 0.36, & \texttt{y2} = 0.44) \\
(\texttt{z} = \text{false}, & \texttt{a} = 0.09, & \texttt{y1} = -0.03, & \texttt{y2} = 0.08) \\
(\texttt{z} = \text{false}, & \texttt{a} = 0.93, & \texttt{y1} = 0.90, & \texttt{y2} = 0.96) \\
(\texttt{z} = \text{true}, & \texttt{a1} = 0.48, \texttt{a2} = 0.15, & \texttt{y1} = 0.47, & \texttt{y2} = 0.19) \\
(\texttt{z} = \text{false}, & \texttt{a} = 0.04, & \texttt{y1} = 0.05, & \texttt{y2} = 0.18) \\
(\texttt{z} = \text{true}, & \texttt{a1} = 0.22, \texttt{a2} = 2.81, & \texttt{y1} = 0.24, & \texttt{y2} = 2.73) \\
(\texttt{z} = \text{false}, & \texttt{a} = 2.13, & \texttt{y1} = 2.20, & \texttt{y2} = 2.07) \\
(\texttt{z} = \text{true}, & \texttt{a1} = 0.25, \texttt{a2} = 2.40, & \texttt{y1} = 0.34, & \texttt{y2} = 2.39) \\
(\texttt{z} = \text{true}, & \texttt{a1} = 2.45, \texttt{a2} = 0.09 & \texttt{y1} = 2.36, & \texttt{y2} = 0.29)
\end{array}
$$



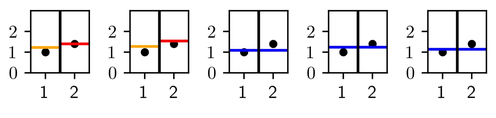
Observed data

$(\texttt{y1} = 1.0, \texttt{y2} = 1.4)$



Posterior samples (traces)

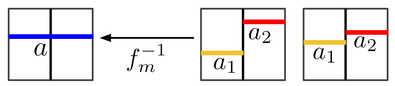
$$
\begin{array}{lll}
(\texttt{z} = \text{true}, & \texttt{a1} = 1.22, \texttt{a2} = 1.40, & \texttt{y1} = 1.0, \texttt{y2} = 1.4) \\
(\texttt{z} = \text{true}, & \texttt{a1} = 1.27, \texttt{a2} = 1.53, & \texttt{y1} = 1.0, \texttt{y2} = 1.4) \\
(\texttt{z} = \text{false}, & \texttt{a} = 1.09, & \texttt{y1} = 1.0, \texttt{y2} = 1.4) \\
(\texttt{z} = \text{false}, & \texttt{a} = 1.23, & \texttt{y1} = 1.0, \texttt{y2} = 1.4) \\
(\texttt{z} = \text{false}, & \texttt{a} = 1.13, & \texttt{y1} = 1.0, \texttt{y2} = 1.4)
\end{array}
$$



Cusumano-Towner, Marco. Gen: A High-Level Programming Platform for Probabilistic Inference. PhD thesis. MIT 2020. Link to PDF

2. Involutive MCMC and custom reversible jump MCMC in Gen

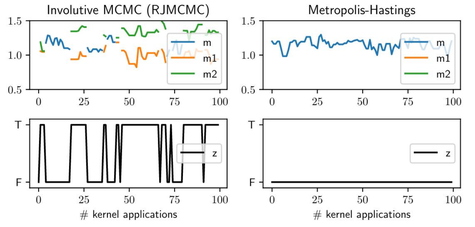
A custom kernel for switching branches efficiently



$$f_m^{-1}(a_1, a_2) = \left(\sqrt{a_1 a_2}, \frac{a_1}{a_1 + a_2}\right)$$

Geometric mean — Degree of freedom

Custom kernel is more efficient



Auxiliary probabilistic program

```
@gen function q(trace)
  if !trace[:z]
    # switch from one to two segments
    {:dof} ~ uniform_continuous(0, 1)
  end
end
```

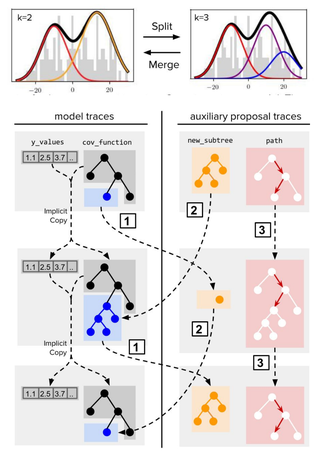
Trace transform program (involution)

```
@transform h (model_in, aux_in) to (model_out, aux_out)
  if @read(model_in[:z], :disc)
    # currently two segments, switch to one
    @write(model_out[:z], false)
    m1 = @read(model_in[:m1], :cont)
    m2 = @read(model_in[:m2], :cont)
    (m, dof) = merge_means(m1, m2)
    @write(model_out[:m], m, :cont)
    @write(aux_out[:dof], dof, :cont)
  else
    # currently one segment, switch to two
    @write(model_out[:z], true, :disc)
    m = @read(model_in[:m], :cont)
    dof = @read(aux_in[:dof], :cont)
    (m1, m2) = split_mean(m, dof)
    @write(model_out[:m1], m1, :cont)
    @write(model_out[:m2], m2, :cont)
  end
end
```

```
function merge_means(m1, m2)
    m = sqrt(m1 * m2)
    dof = m1 / (m1 + m2)
    return (m, dof)
end

function split_mean(m, dof)
    m1 = m * sqrt((dof / (1 - dof)))
    m2 = m * sqrt(((1 - dof) / dof))
    return (m1, m2)
end
```

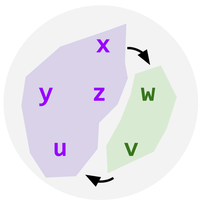
Other examples



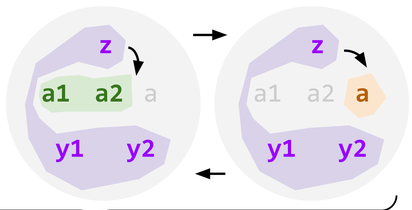
Cusumano-Towner, Marco, Alexander K. Lew, and Vikash K. Mansinghka.
"Automating Involutive MCMC using Probabilistic and Differentiable Programming."
arXiv preprint arXiv:2007.09871 (2020). Link to PDF

3. Use approx. samplers in the auxiliary probabilistic program to infer values of new addresses

(a) Gibbs sampling



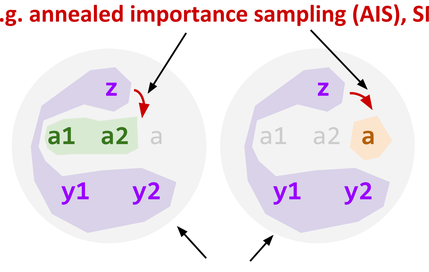
(b) Conditionally sample new choices within structure change



(Sampling from exact conditionals)

(c) This work: *Approximately* conditionally sample new *(+ optionally, some old)* choices within a structure change

Sampler e.g. annealed importance sampling (AIS), SMC, etc.



Algorithm 1 Subproblem pseudomarginal reversible jump (SPRJ) MCMC kernel template

procedure SPRJ-KERNEL(z, θ_{I_z})
- $m \sim q(\cdot; z)$ ▷ Sample a move type m which moves to new structure z'
- $u', \theta'_{(I_{z'} \setminus I_z) \cup S_m} \sim q(\cdot; m, z', \theta_{(I_z \cap I_{z'}) \setminus S_m}, y)$ ▷ Run sampler on forward subproblem
- $\theta'_{I_{z'}} \leftarrow \left(\theta_{(I_z \cap I_{z'}) \setminus S_m}, \theta'_{(I_{z'} \setminus I_z) \cup S_m}\right)$ ▷ Form complete set of parameter values for structure z'
- $\hat{p}(y|\theta_{(I_z \cap I_{z'}) \setminus S_m}; z') \leftarrow \text{MLEST}(m, z, z', u', \theta'_{I_{z'}})$ ▷ Forward marginal likelihood estimate
- $u \sim r(\cdot; m, z, \theta_{(I_z \cap I_{z'}) \setminus S_m}, y, \theta_{(I_z \setminus I_{z'}) \cup S_m})$ ▷ Run meta-inference for reverse subproblem sampler
- $\hat{p}(y|\theta_{(I_z \cap I_{z'}) \setminus S_m}; z) \leftarrow \text{MLEST}(m, z', z, u, \theta_{I_z})$ ▷ Reverse marginal likelihood estimate
- $a \leftarrow \frac{q(m; z) p(z') \hat{p}(y|\theta_{(I_z \cap I_{z'}) \setminus S_m}; z')}{q(m; z') p(z) \hat{p}(y|\theta_{(I_z \cap I_{z'}) \setminus S_m}; z)}$ ▷ Compute acceptance probability
- **return** ACCEPT-REJECT$(a, (z, \theta_{I_z}), (z', \theta_{I_{z'}}))$ ▷ Accept or reject proposed move to $(z', \theta'_{I_{z'}})$

end procedure

procedure MLEST(m, z, z', u', $\theta'_{I_{z'}}$)
- **return** $\frac{p(\theta'_{I_z}; z') p(y; z', \theta'_{I_z}) r(u'; m, z', \theta'_{(I_z \cap I_{z'}) \setminus S_m}, y, \theta'_{(I_{z'} \setminus I_z) \cup S_m})}{q(u', \theta'_{(I_{z'} \setminus I_z) \cup S_m}; m, z', \theta_{(I_z \cap I_{z'}) \setminus S_m}, y))}$

end procedure

procedure ACCEPT-REJECT(a, x, x')
- $w \sim \text{Uniform}(0, 1)$
- **if** $w \leq a$ **then return** x' **else return** x

end procedure

Algorithm 2 SPRJ kernel using annealed importance sampling for subproblem inference (AIS-SPRJ)

procedure SPRJ-KERNEL-AIS(z, θ_{I_z})
- $m \sim q(\cdot; z)$ ▷ Sample a move type m which moves to new structure z'
- $\left(\theta'_{(I_{z'} \setminus I_z) \cup S_m}, \hat{p}(y|\theta_{(I_z \cap I_{z'}) \setminus S_m}; z')\right) \sim \text{AIS}(m, z', \theta_{(I_z \cap I_{z'}) \setminus S_m}, y)$ ▷ AIS on forward subproblem
- $\hat{p}(y|\theta_{(I_z \cap I_{z'}) \setminus S_m}; z) \sim \text{R-AIS}(m, z, \theta_{(I_z \cap I_{z'}) \setminus S_m}, y, \theta_{(I_z \setminus I_{z'}) \cup S_m})$ ▷ Reverse AIS on rev. subproblem
- $a \leftarrow \frac{q(m; z) p(z') \hat{p}(y|\theta_{(I_z \cap I_{z'}) \setminus S_m}; z')}{q(m; z') p(z) \hat{p}(y|\theta_{(I_z \cap I_{z'}) \setminus S_m}; z)}$ ▷ Compute acceptance probability
- **return** ACCEPT-REJECT$(a, (z, \theta_{I_z}), (z', \theta_{I_{z'}}))$ ▷ Accept or reject proposed move to $(z', \theta'_{I_{z'}})$

end procedure

"Subproblem" = Any previously nonexistent addresses, and any addressed in the (optional) "shared subproblem"

Generalizes "subproblem" MCMC (Mansinghka et al., 2014):
(1) Can use approximate instead of exact samplers
(2) Applies generally to structure-changing moves

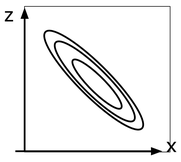
Related work: Georgios Karagiannis and Christophe Andrieu. 2013. Annealed importance sampling reversible jump MCMC algorithms. Journal of Computational and Graphical Statistics 22, 3 (2013), 623–648.

4. Approximates standard Gibbs sampling when there is no structure change

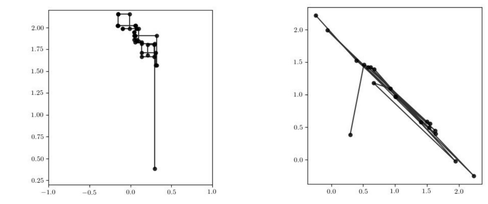
A model with fixed structure

```
@gen function model(noise::Float64)
  x = @trace(normal(0, 1), :x)
  z = @trace(normal(0, 1), :z)
  y = @trace(normal(x + z, noise), :y)
end
```

Conditional distribution on x, z given y



Iterates from two different AIS-SPRJ kernels

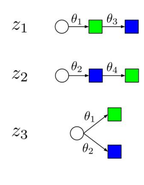


Cycle of two AIS-SPRJ kernels with subproblems {x} and {z}

An AIS-SPRJ kernel with subproblem {x,z}

5. Application: Inferring geometric scene graph structure (tree-based pose parametrization)

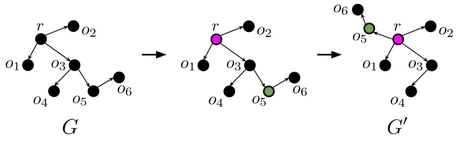
Tree of coordinate frames in a scene



Three possible trees for a scene with two objects

- Root node is the world coordinate frame
- Other nodes are objects' coordinate frames (poses)
- Edges from one object to another indicate sliding (3 degree-of-freedom) contact between objects
- Edges from root to to object indicate the object has a full 6 degree-of-freedom pose

MCMC move on the tree (prune and regraft)



Use AIS to infer new continuous pose parameters during each move