Resampling on parallel architectures

Pierre Jacob with P. Del Moral, A. Lee, L. Murray, G. Peters

September 20th, 2012
Recent Advances in Sequential Monte Carlo, CRiSM

Outline

Outline

Resampling on parallel architectures

Discrete-time model

Markov chain (X_n) for $n \in \mathbb{N}$ with laws $(\mathbb{P}_n)$ on the path space:

$$\mathbb{P}_n(x_{0:n}) = \mu_0(x_0) \prod_{0 \leq p < n} M_{p+1}(x_p, x_{p+1}).$$

For potential functions (G_n), introduce Feynman–Kac measures:

$$\mathbb{Q}_n(x_{0:n}) = \frac{1}{\mathcal{Z}_n} \left\{ \prod_{0 \leq p < n} G_p(x_p) \right\} \mathbb{P}_n(x_{0:n}).$$

Resampling on parallel architectures

Sequential Monte Carlo

To approximate some integrals w.r.t. Feynman–Kac measures, introduce a system of particles $(\xi_n^{1:N})$ as follows:

- At time 0: $\forall i \in \{1, \dots, N\} \quad \xi_0^i \sim \mu_0$ and set $w_0^i = \frac{1}{N}$.

Resampling on parallel architectures

Sequential Monte Carlo

To approximate some integrals w.r.t. Feynman–Kac measures, introduce a system of particles $(\xi_n^{1:N})$ as follows:

- At time 0: $\forall i \in \{1, \dots, N\} \quad \xi_0^i \sim \mu_0$ and set $w_0^i = \frac{1}{N}$.
- Then at time $n > 0$:
- $\forall i \in \{1, \dots, N\} \quad a_{n-1}^i \sim r(\cdot | i, w_{n-1}^{1:N})$
- $\forall i \in \{1, \dots, N\} \quad \xi_n^i \sim M_n(\xi_{n-1}^{a_{n-1}^i}, \cdot)$
- $\forall i \in \{1, \dots, N\} \quad w_n^i = G_n(\xi_n^i)$

where $r(\cdot | i, w_{n-1}^{1:N})$ is a distribution on $\{1, \dots, N\}$.

Resampling on parallel architectures

Computational effort

- Most of the effort usually lies in
  - drawing from the transition M_n
  - and evaluating the potential G_n.

Resampling on parallel architectures

Computational effort

- Most of the effort usually lies in
  - drawing from the transition M_n
  - and evaluating the potential G_n.
- This can be done independently for all particles $(\xi_n^{1:N})$.

Resampling on parallel architectures

Computational effort

- Most of the effort usually lies in
  - drawing from the transition M_n
  - and evaluating the potential G_n.
- This can be done independently for all particles ($\xi_n^{1:N}$).
- The remaining task is the resampling step.

Resampling on parallel architectures

Computational effort

- Most of the effort usually lies in
  - drawing from the transition M_n
  - and evaluating the potential G_n.
- This can be done independently for all particles $(\xi_n^{1:N})$.
- The remaining task is the resampling step.

Problem

Drawing from r typically requires computing $\sum_{1 \leq i \leq N} w_n^i$ or some other "Reduce" step. This can take time because of memory transfer and asynchronicity.

Resampling on parallel architectures

Asynchronicity

Asynchronicity can occur for various reasons:

- heterogeneous architecture,

Resampling on parallel architectures

Asynchronicity

Asynchronicity can occur for various reasons:

- heterogeneous architecture,
- drawing from M_n is more or less expensive depending on x_n, e.g. if it implies numerically solving a differential equation,

Resampling on parallel architectures

Asynchronicity

Asynchronicity can occur for various reasons:

- heterogeneous architecture,
- drawing from M_n is more or less expensive depending on x_n, e.g. if it implies numerically solving a differential equation,
- evaluating G_n is more or less expensive depending on x_n.

Resampling on parallel architectures

Asynchronicity

Asynchronicity can occur for various reasons:

- heterogeneous architecture,
- drawing from M_n is more or less expensive depending on x_n, e.g. if it implies numerically solving a differential equation,
- evaluating G_n is more or less expensive depending on x_n.

Asynchronicity has terrible consequences on the resampling step: all particles wait for one "slow" particle instead of going on to the next steps.

Resampling on parallel architectures

Example: SMC^2

- Each particle corresponds to a vector of parameters in a state space model context.

Resampling on parallel architectures

Example: SMC^2

- Each particle corresponds to a vector of parameters in a state space model context.
- Evaluating its weight implies estimating the associated likelihood value.

Resampling on parallel architectures

Example: SMC^2

- Each particle corresponds to a vector of parameters in a state space model context.
- Evaluating its weight implies estimating the associated likelihood value.
- Estimating the likelihood for a given parameter θ means running a particle filter given θ.

Resampling on parallel architectures

Example: SMC^2

- Each particle corresponds to a vector of parameters in a state space model context.
- Evaluating its weight implies estimating the associated likelihood value.
- Estimating the likelihood for a given parameter θ means running a particle filter given θ.
- Some (bad) values of θ are such that the particle filter takes longer, and eventually correspond to small likelihood values.

Resampling on parallel architectures

Example: SMC^2

- Each particle corresponds to a vector of parameters in a state space model context.
- Evaluating its weight implies estimating the associated likelihood value.
- Estimating the likelihood for a given parameter θ means running a particle filter given θ.
- Some (bad) values of θ are such that the particle filter takes longer, and eventually correspond to small likelihood values.
- Most particles wait for the "slowest" one that eventually gets killed.

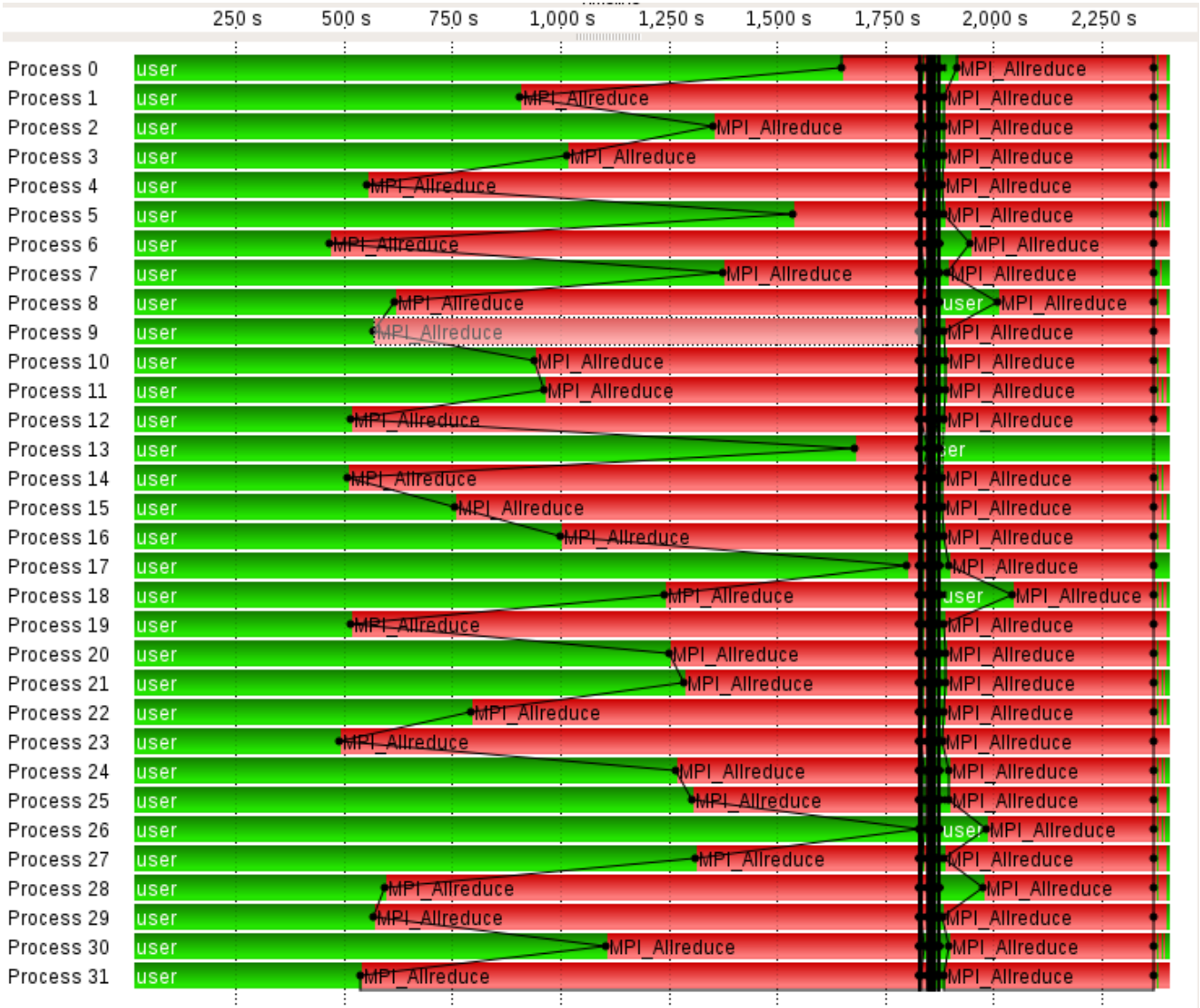
Resampling on parallel architectures

Example: SMC^2

- Each particle corresponds to a vector of parameters in a state space model context.
- Evaluating its weight implies estimating the associated likelihood value.
- Estimating the likelihood for a given parameter θ means running a particle filter given θ.
- Some (bad) values of θ are such that the particle filter takes longer, and eventually correspond to small likelihood values.
- Most particles wait for the "slowest" one that eventually gets killed.

This is very frustrating.

Resampling on parallel architectures



Resampling on parallel architectures

New resampling schemes

An example of such a scheme is proposed in:

GPU acceleration of the particle filter: the Metropolis resampler
Lawrence Murray (arXiv 1202.6163).

This is an approximate scheme: a bias is introduced compared to e.g. multinomial resampling.
The same will apply for our proposed algorithm.

Outline

Avoiding global interactions

Back to Feynman–Kac

In the Feynman–Kac model, we have the following recursion for the time-marginal distributions μ_n of $\mathbb{Q}_n$:

$$\begin{aligned}\mu_{n+1} &= \Psi_{G_n}(\mu_n) M_{n+1} \\ &= \mu_n S_{n,\mu_n} M_{n+1}\end{aligned}$$

for Ψ_G the Boltzmann–Gibbs transformation: $\Psi_G(\mu)(f) = \frac{\mu(fG)}{\mu(G)}$ and $S_{n,\mu}$ a Markov transition that depends on μ.

Avoiding global interactions

Case of bounded potentials

We are going to consider the following transitions:

$$S_{n,\mu_n}(x, dy) = \alpha G_n(x)\delta_x(dy) + (1 - \alpha G_n(x))\Psi_{G_n}(\mu_n)(dy)$$

for $\alpha G_n \leq 1$.

Avoiding global interactions

Case of bounded potentials

We are going to consider the following transitions:

$$S_{n,\mu_n}(x, dy) = \alpha G_n(x)\delta_x(dy) + (1 - \alpha G_n(x))\Psi_{G_n}(\mu_n)(dy)$$

for $\alpha G_n \leq 1$.

Note that global interactions in the resampling step appear through $\Psi_{G_n}(\mu_n)$.

Avoiding global interactions

In SMC, the sequence of measures μ_n is approximated by particles:

$$\mu_n^N = \frac{1}{N} \sum_{i=1}^{N} \delta_{\xi_n^i}.$$

"Recycling-jump sampling"

Plugging μ_n^N in the previous expression, we obtain:

$$S_{n,\mu_n^N}(\xi_n^i, dy) = \alpha G_n(\xi_n^i) \delta_{\xi_n^i}(dy) + (1 - \alpha G_n(\xi_n^i)) \left(\sum_{j=1}^{N} \frac{G_n(\xi_n^j)}{\sum_{j=1}^{N} G_n(\xi_n^j)} \delta_{\xi_n^j}(dy) \right).$$

Avoiding global interactions

Getting rid of global interactions

If $G_n(\xi_n^i)$, $i = 1, \ldots, N$ were even, this scheme would be close to:

$$\tilde{S}_{n,\mu_n^N}(\xi_n^i, dy) = \alpha G_n(\xi_n^i)\delta_{\xi_n^i}(dy) + (1 - \alpha G_n(\xi_n^i)) \left(\frac{1}{N} \sum_{j=1}^{N} \delta_{\xi_n^j}(dy) \right)$$

but there is no reason why the weights should be even.

Avoiding global interactions

Intermediate time steps

For some integer m, consider the sequence $t_k = k/m$ for $k \in \mathbb{N}$.
Introduce the Feynman–Kac model $(M_{t_k}^{(m)}, G_{t_k}^{(m)})$ for $k \in \mathbb{N}$:

- $M_{t_k}^{(m)} = M_n$ if $t_k = n \in \mathbb{N}$ and $M_{t_k}^{(m)} = \delta$ otherwise.
- $G_{t_k}^{(m)} = G_{\lfloor t_k \rfloor}^{1/m}$

and denote by $\mu_{t_k}^{(m)}$ the time-marginal distribution of the associated Feynman–Kac measure.

Avoiding global interactions

Property

We have $\mu_{t_k}^{(m)} = \mu_n$ when $t_k = n \in \mathbb{N}$, so that the original model is embedded in a finer discrete-time model.

Avoiding global interactions

Property

We have $\mu_{t_k}^{(m)} = \mu_n$ when $t_k = n \in \mathbb{N}$, so that the original model is embedded in a finer discrete-time model.

Motivation

If m is large, the one-step weights are quite even, thus $\tilde{S}_{t_k,\mu_{t_k}}$ is close to $S_{t_k,\mu_{t_k}}$, therefore we can bypass global interactions.

Avoiding global interactions

Property

We have $\mu_{t_k}^{(m)} = \mu_n$ when $t_k = n \in \mathbb{N}$, so that the original model is embedded in a finer discrete-time model.

Motivation

If m is large, the one-step weights are quite even, thus $\tilde{S}_{t_k,\mu_{t_k}}$ is close to $S_{t_k,\mu_{t_k}}$, therefore we can bypass global interactions.

More formally...

...we can get results of the type

$$\|\Psi_{G_{t_k}^{(m)}}(\mu_{t_k}^N) - \mu_{t_k}^N\|_{TV} \leq \frac{\sup_x |G_{\lfloor t_k \rfloor}(x)|}{m}.$$

Outline

Proposed algorithm

We are going to follow the modified resampling step:

$$\tilde{S}_{n,\mu_{t_k}^N}(\xi_{t_k}^i, dy) = G_{t_k}^{(m)}(\xi_{t_k}^i)\delta_{\xi_{t_k}^i}(dy) + (1 - G_{t_k}^{(m)}(\xi_{t_k}^i)) \left(\frac{1}{N} \sum_{j=1}^{N} \delta_{\xi_{t_k}^j}(dy) \right)$$

using the so-called "Clock Resampling" algorithm.

Proposed algorithm

We are going to follow the modified resampling step:

$$\tilde{S}_{n,\mu^N_{t_k}}(\xi^i_{t_k}, dy) = G^{(m)}_{t_k}(\xi^i_{t_k})\delta_{\xi^i_{t_k}}(dy) + (1 - G^{(m)}_{t_k}(\xi^i_{t_k}))\left(\frac{1}{N}\sum_{j=1}^{N}\delta_{\xi^j_{t_k}}(dy)\right)$$

using the so-called "Clock Resampling" algorithm.

Clocks

Each particle ξ^i is associated with a uniform variable $U^i \sim \mathcal{U}_{[0,1]}$.
When $\prod_{0 \leq p \leq k} G^{(m)}_{t_p}(\xi^i_{t_p}) \leq U^i$, the particle jumps.
When it jumps, it goes uniformly to any location among $\xi^{1:N}_{t_k}$.

Proposed algorithm

Clock resampling algorithm, discrete-time

$\forall i \in \{1, \dots, N\}$ init $P_G^i \leftarrow 1$.
$\forall i \in \{1, \dots, N\}$ draw $U^i \sim \mathcal{U}_{[0,1]}$.
for $k = 0$ to $m - 1$ **do**
  for $i \in \{1, \dots, N\}$ **do**
    update product of potentials $P_G^i \leftarrow P_G^i \times G_{t_k}^{(m)}(\xi_{t_k}^i)$.
    if $P_G^i \leq U^i$ **then**
      draw new location $i \leftarrow j \sim \mathcal{U}_{1:N}$.
      copy $\xi_{t_{k+1}}^i \leftarrow \xi_{t_k}^j$.
      reset $P_G^i \leftarrow 1$.
      draw a new variable $U^i \sim \mathcal{U}_{[0,1]}$.
    end if
  end for
end for

Proposed algorithm

Extension: continuous-time version

We can allow any intermediate time between integer times: the particle jumps exactly at time t such that

$$\left(G_{\lfloor t \rfloor}(\xi_t^i)\right)^{(t-\lfloor t \rfloor)/m} = U^i.$$

Extension: unbounded potentials

Unbounded potentials can be dealt with "birth" mechanisms.

No numerical results yet...