Divide-and-Conquer Sequential Monte Carlo
for inference in probabilistic graphical models

“D&C-SMC generalize the classical SMC framework from sequences (or chains) to trees.”

Thomas Schön
Division of Systems and Control
Department of Information Technology
Uppsala University.

Email: thomas.schon@it.uu.se,
www: user.it.uu.se/~thosc112

Joint work with: John Aston (University of Cambridge), Alexandre Bouchard-Côté (University of British Columbia), Adam M. Johansen (University of Warwick), Brent Kirkpatrick (Miami University), Fredrik Lindsten (Uppsala University) and Christian A. Naesseth (Linköping University).
Background – what we do in the team

We automate the extraction of knowledge and understanding from data.

Both basic research and applied research (with companies).

Create probabilistic models for dynamical systems and their surroundings.

Develop methods to learn models from data.

The models can then be used by machines (or humans) to understand or take decisions about what will happen next.
Problem background and our idea

Performing inference in statistical models involving a large number of random variables and nonlinear interactions is a hard problem.

Exploiting problem structure is vital.

Probabilistic graphical models are a natural way to represent and make use of underlying structure.

Key idea: Introduce a tree (or sequential) decomposition of the graphical model. The decomposition opens up for solving the inference problems using sequential Monte Carlo.
A **probabilistic graphical model** (PGM) is formalism where a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is used to **encode the dependencies** between random variables in a probabilistic model. It consists of:

1. a set of **vertices** $\mathcal{V}$ (nodes) represents the random variables
2. a set of **edges** $\mathcal{E}$ containing elements $(i, j) \in \mathcal{E}$ connecting a pair of nodes $(i, j) \in \mathcal{V} \times \mathcal{V}$
Sequential Monte Carlo (SMC)

The distribution of interest $\pi(x)$ is called target distribution. SMC methods are a class of sampling (simulation) algorithms capable of addressing our two problems:

1. Approximate $\pi(x)$ and compute integrals $\int h(x)\pi(x)dx$.
2. Approximate the normalizing constant $Z$.

(Abstract) problem formulation: Sample from a sequence of probability distributions $\{\pi_t(x_{1:t})\}_{t \geq 1}$ (where $x_{1:t} = (x_1, \ldots, x_t)$) defined on a sequence of spaces of increasing dimension,

$$\pi_t(x_{1:t}) = \frac{\gamma_t(x_{1:t})}{Z_t},$$

such that $\gamma_t(x_{1:t}) : X^t \to \mathbb{R}^+$ is known pointwise and $Z_t = \int \gamma(x_{1:t})dx_{1:t}$ is often computationally challenging.

Important question: How general is this formulation?
Outline

1. Introduction

2. Probabilistic state space models
   3. Classical SMC
      a) Particle filter for state space models
      b) Introductory example
      c) Algorithm
      d) SMC - general “static” formulation

4. SMC on trees (D&C-SMC)

5. Decomposing general PGMs

6. Snap-shots of some current research (if there is time)

7. Conclusion

D&C-SMC generalize the classical SMC framework from sequences (or chains) to trees.
Probabilistic state space models

Probabilistic modeling allow for **representing and manipulating uncertainty** in data, models, decisions and predictions.

A **parametric** state space model is given by:

\[
\begin{align*}
    x_{t+1} &= f_\theta(x_t, u_t) + v_{\theta,t}, \\
    y_t &= g_\theta(x_t, u_t) + e_{\theta,t}, \\
    x_0 &\sim p_\theta(x_0), \\
    \theta &\sim p(\theta).
\end{align*}
\]

\[
\begin{align*}
    x_{t+1} | x_t &\sim p_\theta(x_{t+1} | x_t, u_t), \\
    y_t | x_t &\sim p_\theta(y_t | x_t, u_t), \\
    x_0 &\sim p_\theta(x_0), \\
    \theta &\sim p(\theta).
\end{align*}
\]

The **full probabilistic model** is given by

\[
p(x_{1:T}, \theta, y_{1:T}) = p(y_{1:T} | x_{1:T}, \theta) p(x_{1:T}, \theta)
\]

\[= \underbrace{p(y_{1:T} | x_{1:T}, \theta)}_{\text{data distribution}} \underbrace{p(x_{1:T}, \theta)}_{\text{prior}}
\]
Probabilistic state space models

Distribution describing a parametric nonlinear state space model:

\[
p(x_{1:T}, \theta, y_{1:T}) = \prod_{t=1}^{T} p(y_t | x_t, \theta) \prod_{t=1}^{T-1} p(x_{t+1} | x_t, \theta) p(x_1 | \theta) p(\theta)
\]

- \(p(x_1 | \theta)\): state prior
- \(p(\theta)\): parameter prior
- \(p(y_t | x_t, \theta)\): observation data distribution
- \(p(x_{t+1} | x_t, \theta)\): dynamics

Model = probability distribution!

Compute the posterior distribution

\[
p(x_{1:T}, \theta | y_{1:T}) = \underbrace{p(x_{1:T} | \theta, y_{1:T})}_\text{state} \underbrace{p(\theta | y_{1:T})}_\text{parameter}.
\]
Example – “what are $x_t$, $\theta$ and $y_t$”? 

**Aim (motion capture):** Compute $x_t$ (position and orientation of the different body segments) of a person ($\theta$ describes the body shape) moving around indoors using measurements $y_t$ (accelerometers, gyroscopes and ultrawideband).


Classical SMC – what it does

Model expressed using equations:

\[
\begin{align*}
x_{t+1} &= f(x_t, u_t) + v_t, \\
y_t &= g(x_t, u_t) + e_t.
\end{align*}
\]

SMC provides a way to sequentially approximate the filtering distributions \( \{\pi_t(x_t) = p(x_t | y_{1:t})\}_{t \geq 1} \) and it also provides an unbiased estimate of the normalization constants \( \{Z_t\}_{t \geq 1} \).
Relationship to abstract formulation

(Abstract) problem formulation: **Sample from a sequence** of probability distributions \( \{\pi_t(x_{1:t})\}_{t \geq 1} \) (where \( x_{1:t} = (x_1, \ldots, x_t) \)) defined on a sequence of spaces of increasing dimension,

\[
\pi_t(x_{1:t}) = \frac{\gamma_t(x_{1:t})}{Z_t},
\]

such that \( \gamma_t(x_{1:t}) : X^t \to \mathbb{R}^+ \) is known pointwise and \( Z_t = \int \gamma(x_{1:t}) \, dx_{1:t} \) is often computationally challenging.

\[
p(x_{1:t} | y_{1:t}) = \frac{\gamma_t(x_{1:t})}{p(y_{1:t})} \frac{p(x_{1:t}, y_{1:t})}{p(y_{1:t})} = \frac{\gamma_t(x_{1:t})}{Z_t}
\]
The particle filter approximates \( p(x_{1:t} \mid y_{1:t}) \) for

\[
x_{t+1} = f(x_t, u_t) + v_t, \\
y_t = g(x_t, u_t) + e_t,
\]

by maintaining an empirical distribution made up of \( N \) samples (particles) \( \{x_{1:t}^i\}_{i=1}^N \) and corresponding weights \( \{w_{1:t}^i\}_{i=1}^N \)

\[
\hat{p}(x_{1:t} \mid y_{1:t}) = \sum_{i=1}^{N} \frac{w_{t}^i}{\sum_{j=1}^{N} w_{t}^j} \delta_{x_{1:t}^i} (x_{1:t}).
\]

“The particle filter provides a systematic way of exploring the state space.”
Consider a toy 1D localization problem.

**Dynamic model:**

\[ x_{t+1} = x_t + u_t + v_t, \]

where \( x_t \) denotes position, \( u_t \) denotes velocity (known), \( v_t \sim \mathcal{N}(0, 5) \) denotes an unknown disturbance.

**Measurements:**

\[ y_t = h(x_t) + e_t. \]

where \( h(\cdot) \) denotes the world model (here the terrain height) and \( e_t \sim \mathcal{N}(0, 1) \) denotes an unknown disturbance.

**Task:** Find the state \( x_t \) (position) based on the measurements \( y_{1:t} \triangleq \{y_1, \ldots, y_t\} \) by computing the filter density \( p(x_t | y_{1:t}) \).
Particle filter – introductory example (II/II)

Highlights two key capabilities of the PF:

1. Automatically handles an unknown and dynamically changing number of hypotheses.
2. Work with nonlinear/non-Gaussian models.
Sequential Monte Carlo – particle filter

**SMC = sequential importance sampling + resampling**

1. **Propagation:** $x_t^i \sim f(x_t \mid x_{1:t-1}^{a_i})$ and $x_{1:t}^i = \{x_{1:t-1}^{a_i}, x_t^i\}$.
2. **Weighting:** $\tilde{w}_t^i = W_t(x_t^i) = g(y_t \mid x_t^i)$.
3. **Resampling:** $\mathbb{P}(a_t^i = j) = \frac{\tilde{w}_t^j}{\sum_l \tilde{w}_{t-1}^l}$.

The **ancestor indices** $\{a_t^i\}_{i=1}^N$ are very **useful** auxiliary variables! They make the stochasticity of the resampling step explicit.
Application – indoor localization (I/III)

**Aim:** Compute the position of a person moving around indoors using sensors (inertial, magnetometer and radio) located in an ID badge and a map.

The sensors (IMU and radio) and the DSP are mounted inside an ID badge.

The inside of the ID badge.
pdf for an office environment, the bright areas are rooms and corridors (i.e., walkable space).

An estimated trajectory and the particle cloud visualized at a particular instance.
Application – indoor localization (III/III)

Show movie

Particle MCMC = SMC + MCMC

A systematic way of combining SMC and MCMC.

Builds on an extended target construction.

Intuitively: SMC is used as a high-dimensional proposal mechanism on the space of state trajectories $X^T$.

A bit more precise: Construct a Markov chain with $p(\theta, x_{1:T} \mid y_{1:T})$ (or one of its marginals) as its stationary distribution. Also used for parameter learning.

Exact approximations

Pioneered by the work

SMC – general “static” formulation

SMC is a class of sampling (simulation) methods capable of:

1. Approximating the target distribution $\pi(x)$ and computing integrals under $\pi(x)$ of some test function $h : X \rightarrow \mathbb{R}$,
   $$\int h(x)\pi(x)dx.$$

2. Computing an **unbiased** estimate of the normalization constant $Z$.

More precisely SMC can be used to simulate from a sequence of probability distributions $\pi_t(x_t)$, $x_t = (\tilde{x}_1, \ldots, \tilde{x}_t)$ defined on a sequence of spaces of increasing dimension

$$X_t = \tilde{X}_1 \times \tilde{X}_2 \times \cdots \times \tilde{X}_t.$$
SMC – general “static” solution

1. Propagation: $x^i_t \sim q_t(x_t | \tilde{x}^i_{1:t-1})$ and $x^i_{1:t} = \{\tilde{x}^i_{1:t-1}, x^i_t\}$.

2. Weighting: $w^i_t = W_t(x^i_{1:t}) = \frac{\gamma_t(x^i_{1:t})}{\gamma_{t-1}(x^i_{1:t-1})q_t(x^i_t | x^i_{1:t-1})}$.

3. Resampling: $\{x^i_{1:t-1}, w^i_{t-1}\}^N_{i=1} \rightarrow \{\tilde{x}^i_{1:t-1}, 1\}^N_{i=1}$.

Results in a new particle population $\{x^i_{1:t}, w^i_t\}^N_{i=1}$. 
Going from classical SMC fo D&C-SMC

The computational graph of classic SMC is a sequence (chain)

D&C-SMC generalize the classical SMC framework from sequences to trees.
Ex – hierarchical Bayesian network

There are 3 observations $y_{1:3}$ and 5 latent variables $\tilde{x}_{1:5}$.

Notation: $x_5 = \tilde{x}_{1:5}$.

Target distribution: $\pi(x_5) = \pi_5(x_5) = p(\tilde{x}_{1:5} \mid y_{1:3})$.

Straightforward to obtain a tree decomposition of the $\pi_5$, by making use of the hierarchical structure of the PGM.
By removing the root node \( \tilde{x}_5 \), we obtain 2 decoupled components for which we can define 2 auxiliary target distributions:

\[
\pi_4(x_4) = p(x_4 \mid y_{1,2}), \quad x_4 = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_4),
\]
\[
\pi_3(x_3) = p(x_3 \mid y_3), \quad x_3 = \tilde{x}_3.
\]

Repeating this procedure we can further decompose \( \pi_4(x_4) \) into 2 auxiliary target distributions:

\[
\pi_1(x_1) = p(x_1 \mid y_1), \quad x_1 = \tilde{x}_1.
\]
\[
\pi_2(x_2) = p(x_2 \mid y_2), \quad x_2 = \tilde{x}_2.
\]
Ex – hierarchical Bayesian network

Level 0:
Level 1:
Level 2:

We initialise the D&C-SMC with independent particle populations for each leaf in the tree decomposition. These are then merged, resampled and propagated as we move up the tree.

Iter 1: Initialise \((\tilde{x}_k^i, w_k^i)^N_{i=1}\) for \(k = 1, 2, 3\).

Iter 2: Merge populations 1 and 2 and propagate \(\Rightarrow (\tilde{x}_{1,2,4}^i, w_{4}^i)^N_{i=1}\)

Iter 3: Merge populations 3 and 4 and propagate \(\Rightarrow (\tilde{x}_{1,2,3,4,5}^i, w_{5}^i)^N_{i=1}\)
Computational graph (tree)

This computational process can be organized in a tree:

Each node is associated with a target distribution,

- $\pi_k(\tilde{x}_k) \propto p(y_k | \tilde{x}_k)p(\tilde{x}_k)$ for $k = 1, 2, 3$.
- $\pi_4(\tilde{x}_{1,2,4}) \propto p(y_{1,2}, \tilde{x}_{1,2} | \tilde{x}_4)p(\tilde{x}_4)$.
- $\pi_5(\tilde{x}_{1,2,3,4,5}) \propto p(y_{1,2,3}, \tilde{x}_{1,2,3,4} | \tilde{x}_5)p(\tilde{x}_5)$. 
Similarities and differences SMC, D&C SMC

As in classical SMC, D&C SMC approximates each $\pi_t$ by a collection of weighted samples, called particle populations.

Unlike classical SMC, D&C SMC maintains multiple independent populations of weighted particles ($\{x^i_t, w^i_t\}_{i=1}^N : t \in T_k$), which are

1. propagated and
2. merged

as the algorithm proceeds. Here $T_k \subset T$ is the set of “active” target distributions at iteration $k$, $1 \leq k \leq \text{height}(T)$.

All details are available in:

Divide-and-Conquer SMC

Formally:

- Let $T$ denote a tree with root $r \in T$.
- For $t \in T$, let $C(t)$ denote the children of $t$.
- Let $\{\pi_t : t \in T\}$ be a family of probability distributions.
- Let $\pi_t \propto \gamma_t$ where $\gamma_t$ can be evaluated point-wise.

If $\pi_t$ is defined on the space

$$X_t = \left( \bigotimes_{c \in C(t)} X_c \right) \times \tilde{X}_t,$$

then D&C-SMC is an algorithm for sampling from each $\pi_t$ bottom-up on the tree $T$.

$\tilde{X}_t$ denotes the “incremental” set.
Theoretical properties

Since D&C-SMC consists of classical SMC steps combined with merging of particle populations via resampling, it is possible to extend (with care) many of the results from classical SMC.

**Property 1:** The D&C-SMC estimate of the normalization constant is unbiased, i.e.

\[
E \left[ \hat{Z}_r \right] = Z_r = \int \gamma_r(x_r) \, dx_r.
\]

**Property 2:** The D&C-SMC algorithm is consistent,

\[
\sum_{i=1}^{N} \frac{\omega_r^i}{\sum_{j=1}^{N} \omega_r^j} f(x_r^i) \rightarrow \int f(x) \pi(x) \, dx, \quad \text{as } N \rightarrow \infty.
\]
Example - NY math test (model)

Dataset containing New York state mathematics test results.

The data is organized into a tree $T$, where a path from the root to the leaf has the form:

1. NYC
2. Borough (5)
3. School district (32)
4. school (710)
5. year (1)

278 399 test instances and each leaf $t \in T$ has an observation of $m_t$ exam successes out of $M_t$ trials.
Example - NY math test (model)

Parameters:

- Observations at the leaf (binomial $p_t = \text{logistic}(\theta_t)$).
- Internal nodes: $\theta_t' = \theta_t + \Delta_e$, with $\Delta_e \sim \mathcal{N}(0, \sigma_{e}^2)$.
- Hyperparameters $\sigma_{e}^2 \sim \text{Exp}(1)$.

After marginalization of internal $\theta$-parameters, the dimensionality of the remaining parameters in the model is 3 555.
Example - NY math test (results)

We compare our D&C-SMC (implemented in Java) to Hamiltonian Monte Carlo (Stan, implemented in C++).

Similar posterior approximation accuracy.

<table>
<thead>
<tr>
<th>Method</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>D&amp;C-SMC</td>
<td>39 s</td>
</tr>
<tr>
<td>HMC (Stan)</td>
<td>3860 s (64 min)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node</th>
<th>Stan</th>
<th>D&amp;C-SMC</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manhattan</td>
<td>0.17</td>
<td>15.96</td>
<td>93.89</td>
</tr>
<tr>
<td>Bronx</td>
<td>0.05</td>
<td>8.12</td>
<td>165.69</td>
</tr>
<tr>
<td>Brooklyn</td>
<td>0.18</td>
<td>6.52</td>
<td>36.22</td>
</tr>
<tr>
<td>Queens</td>
<td>0.07</td>
<td>14.01</td>
<td>209.05</td>
</tr>
<tr>
<td>Staten Island</td>
<td>0.05</td>
<td>25.50</td>
<td>481.17</td>
</tr>
</tbody>
</table>

The effective samples (ESS) per second and speedup.
D&C-SMC (dc) outperforms classical SMC (std) for all computational budgets.
1. Introduction
2. Probabilistic state space models
3. Classical SMC
   a) Particle filter for state space models
   b) Introductory example
   c) Algorithm
   d) SMC - general “static” formulation
4. SMC on trees (D&C-SMC)
5. Decomposing general PGMs
6. Snap-shots of some current research (if there is time)
7. Conclusion

D&C-SMC generalize the classical SMC framework from sequences (or chains) to trees.
Classical SMC methods are used to approximate a sequence of probability distributions on a sequence of spaces of increasing dimension.

Key idea:
1. Introduce a sequential decomposition of the PGM.
2. Each subgraph induces an intermediate target dist.
3. Apply SMC to the sequence of intermediate target dist.

Using an artificial sequence of intermediate target distributions for an SMC method is a powerful (quite possibly underutilized) idea.
Example – latent MRF

Illustrating possible decompositions of a latent Markov random field (MRF).

Tree decomposition follows naturally when the graphical model is a tree. However, the idea is more generally applicable.

Example: Lattice Markov random field (MRF)

The subgraphs can be organized on a tree!
The bootstrap PF suffers from weight collapse in high-dimensional settings.

This degeneracy can be reduced by using so-called **fully adapted** proposals.

We can mimic the efficient fully adapted proposals for arbitrary latent spaces and structures in high-dimensional models.

**Approximations the proposal distribution** and use a **nested coupling** of multiple SMC samplers and backward simulators.

Gaussian process state space model

“Inspired by the Gaussian process, enabled by the particle filter”

\[ x_{t+1} = f(x_t) + v_t, \quad \text{s.t.} \quad f(x) \sim \mathcal{GP}(0, \kappa \eta, f(x, x')) , \]

\[ y_t = g(x_t) + e_t, \quad \text{s.t.} \quad g(x) \sim \mathcal{GP}(0, \kappa \eta, g(x, x')) . \]

The model functions \( f \) and \( g \) are assumed to be realizations from Gaussian process priors and \( v_t \sim \mathcal{N}(0, Q) \), \( e_t \sim \mathcal{N}(0, R) \).

We can now find the posterior distribution

\[ p(f, g, Q, R, \eta \mid y_{1:T}) , \]

via some approximation (we use particle MCMC).


Regularization in nonlinear state spaces

Results in a **flexible** non-parametric model where the GP prior on $f$ takes on the role of a regularizer.

Provides a data-driven way of **tuning** the model flexibility.

Toy example:

$$x_{t+1} = -10 \frac{x_t}{1 + 3x_t^2} + v_t,$$

$$y_t = x_t + e_t.$$

For all details on the new GP-SSM and more examples, see

Coupling of particle filters

New coupled resampling schemes can be used to improve a variety of particle-based algorithms.

Log-likelihood estimates obtained by particle filters, in a hidden AR(1).

New particle smoother, easy to parallelize and with analysis.

**ASSEMBLE project**

**Aim:** Automate probabilistic modeling of dynamical systems (and their surroundings) via a formally defined **probabilistic modeling language**.

**Probabilistic Modeling Research**
- Application Model

**Inference Methods Research**
- Inference Methods

**Modeling Language Research**
- Probabilistic Model Compiler

**Demonstrators**
- Smart Meters (Greenely)
- Cell Tracking (Karolinska Institute)
- Energy-Aware Computing
- Container Crane Automation (ABB)
- Smart Automotive Safety (Autoliv)

**Application Specific Machine Learning Solution**

Feedback from demonstrators enables:
- Improved modeling techniques
- Improved inference methods
- Enhanced modeling language

We will create a **market place** for inference and learning algorithms and probabilistic model libraries.

**Project URL:** [www.it.uu.se/research/assemble](http://www.it.uu.se/research/assemble)
SMC conference and PhD course

We are hosting the SMC conference in Uppsala during the time Aug. 30 – Sep. 1, 2017.

More information and registration is available here:

www.it.uu.se/conferences/smc2017

Welcome!!

During the week leading up to the SMC conference (Aug. 24 – Aug. 29) we are offering an intensive PhD course on SMC.

More information and registration is available here:

http://www.it.uu.se/conferences/smc2017/course

Welcome!!
Conclusion

D&C-SMC generalize the classical SMC framework from sequences (or chains) to trees.

Key insight: We exploit various decompositions of the graphical model to design efficient SMC samplers.

D&C-SMC = Classical SMC steps + merging of particle populations via resampling.

- Does not require a tree structure in the model.
- Produce an approximation of the full joint distribution.
- Produce an unbiased estimate of the normalization constant.
- Can be used within MCMC in a plug-and-play manner.

Huge need for automation (ASSEMBLE project).
References to some of our work

Divide-and-conquer SMC


SMC for graphical models


Financial support from the Swedish Research Council (VR) and the Swedish Foundation for Strategic research (SSF) is gratefully acknowledged.
Divide-and-Conquer SMC

Algorithm \texttt{dc\_smc}(t) – D&C-SMC for node $t \in T$

1. For $c \in \mathcal{C}(t)$:
   1.1 $(x^i_c, w^i_c)_{i=1}^N \leftarrow \texttt{dc\_smc}(c)$.
   1.2 Resample $(x^i_c, w^i_c)_{i=1}^N$ to obtain the equally weighted particle system $(\tilde{x}^i_c, 1)_{i=1}^N$.

2. For particle $i = 1, \ldots, N$:
   2.1 Simulate $\tilde{x}^i_t \sim q_t(\cdot \mid \tilde{x}^i_{c_1}, \ldots, \tilde{x}^i_{c_C})$ from some proposal kernel on $\tilde{X}_t$, and where $(c_1, c_2, \ldots, c_C) = \mathcal{C}(t)$.
   2.2 Set $x^i_t = (\tilde{x}^i_{c_1}, \ldots, \tilde{x}^i_{c_C}, \tilde{x}^i_t)$.
   2.3 Compute $w^i_t = \frac{\gamma_t(x^i_t)}{\prod_{c \in \mathcal{C}(t)} \gamma_c(\tilde{x}^i_c)} \frac{1}{q_t(\tilde{x}^i_t \mid \tilde{x}^i_{c_1}, \ldots, \tilde{x}^i_{c_C})}$.

3. Return $(x^i_t, w^i_t)_{i=1}^N$.

Generalises the SMC framework (std SMC recovered if $T$ is a chain).