Divide-and-Conquer with Sequential Monte Carlo

F. Lindsten[1], A. M. Johansen[2], C. A. Naesseth[3], B. Kirkpatrick[4], T. B. Schön[1], J. A. D. Aston[5], and A. Bouchard-Côté[6]

October 11, 2016

Abstract

We propose a novel class of Sequential Monte Carlo (SMC) algorithms, appropriate for inference in probabilistic graphical models. This class of algorithms adopts a divide-and-conquer approach based upon an auxiliary tree-structured decomposition of the model of interest, turning the overall inferential task into a collection of recursively solved sub-problems. The proposed method is applicable to a broad class of probabilistic graphical models, *including* models with loops. Unlike a standard SMC sampler, the proposed Divide-and-Conquer SMC employs multiple independent populations of weighted particles, which are resampled, merged, and propagated as the method progresses. We illustrate empirically that this approach can outperform standard methods in terms of the accuracy of the posterior expectation and marginal likelihood approximations. Divide-and-Conquer SMC also opens up novel parallel implementation options and the possibility of concentrating the computational effort on the most challenging sub-problems. We demonstrate its performance on a Markov random field and on a hierarchical logistic regression problem. Supplementary material including proofs and additional numerical results is available online.

Keywords: Bayesian methods, Graphical models, Hierarchical models, Particle filters

1 Introduction

Sequential Monte Carlo (SMC) methods are a popular class of algorithms for approximating some sequence of probability distributions of interest, $(\pi_t(\mathbf{x}_t) : t = 1, \ldots, n)$. This is done by simulating, for each distribution in the sequence, a collection of N particles $\{\mathbf{x}_t^i\}_{i=1}^N$ with corresponding nonnegative importance weights $\{\mathbf{w}_t^i\}_{i=1}^N$, such that the weighted empirical

[*]1 Uppsala University, 2 University of Warwick, 3 Linköping University, 4 University of Miami, 5 University of Cambridge, 6 University of British Columbia. Address for Correspondence: Fredrik Lindsten, Division of Systems and Control, Department of Information Technology, Uppsala University, Box 337, 751 05 Uppsala, Sweden. Email: `fredrik.lindsten@it.uu.se`.

distribution $\widehat{\pi}_t^N(\mathrm{d}\mathbf{x}_t) := (\sum_j \mathbf{w}_t^j)^{-1} \sum_i \mathbf{w}_t^i \delta_{\mathbf{x}_t^i}(\mathrm{d}\mathbf{x}_t)$ approximates π_t. The weighted particles are generated sequentially, in the sense that the particles generated at iteration t depends on the particles generated up to iteration $t-1$.

The most well-known application of SMC is to the filtering problem in general state-space hidden Markov models, see e.g., Doucet and Johansen (2011) and references therein. However, these methods are much more generally applicable and there has been much recent interest in using SMC for sampling from probability distributions that do not arise from chain-shaped probabilistic graphical models (PGMs). This typically involves using SMC to target a sequence of auxiliary distributions which are constructed to admit the original distribution as an appropriate marginal (Del Moral et al., 2006). Examples include likelihood tempering (Del Moral et al., 2006), data tempering (Chopin, 2002), and sequential model decompositions (Bouchard-Côté et al., 2012; Naesseth et al., 2014), to mention a few.

For many statistical models of interest, however, a *sequential* decomposition might not be the most natural, nor computationally efficient, way of approaching the inference problem. In this contribution we propose an extension of the classical SMC framework, Divide-and-Conquer SMC (D&C-SMC), which we believe will further widen the scope of SMC samplers and provide efficient computational tools for Bayesian inference within a broad class of probabilistic models.

The idea underlying D&C-SMC is that an approximation can be made to any multivariate distribution by splitting the collection of model variables into *disjoint* sets and defining, for each of these sets, a suitable auxiliary target distribution. Sampling from these distributions is typically easier than sampling from the original distribution and can be done in parallel, whereafter the results are merged to provide a solution to the original problem of interest (correcting for the discrepancy between the approximating and exact distributions by importance sampling techniques). Using the divide-and-conquer methodology, we recurse and repeat this procedure for each of the components. This corresponds to breaking the overall inferential task into a collection of simpler problems. At any intermediate iteration of the D&C-SMC algorithm we maintain multiple independent sets of weighted particles, which are subsequently merged and propagated as the algorithm progresses, using rules similar to those employed in standard SMC. The proposed method inherits some of the theoretical guarantees of standard SMC methods. In particular, our simulation scheme can be used to provide *exact approximations* of costly or intractable MCMC algorithms, via the particle MCMC methodology (Andrieu et al., 2010).

Furthermore, we introduce a method for constructing the aforementioned decompositions for a broad class of PGMs of interest, which we call *self-similar graphical models*. To construct auxiliary distributions, we remove edges and nodes in a PGM of interest, creating smaller connected components as sub-graphical models. These sub-graphical models are then recursively decomposed as well. Note that this decomposition does not assume that the PGM of interest is tree-shaped. Indeed, we demonstrate that the proposed methodology is effective not only when the model has an obvious hierarchical structure (for example, Figure 3), but also in cases where the hierarchical decomposition is artificial (Figure 5). In either case, one iteratively exploits solutions to easier sub-problems as a first step in the solution of a more complex problem.

We conclude this section with a summary of the structure of the remainder of the paper. Section 2 provides details on the background of the work presented here: algorithms upon which it builds and those to which it is related. The basic D&C-SMC and decomposition methodology is presented in Section 3, including a discussion of its theoretical properties. A number of methodological extensions are presented in Section 4—these comprise fundamental components of the general strategy introduced in this paper, and may be required to obtain good performance in challenging settings. In Section 5 two numerical illustrations are presented. The paper concludes with a discussion. Online Supplementary material contains proofs and additional numerical examples.

2 Background and problem formulation

2.1 Problem formulation

We let π denote a probability distribution of interest, termed the *target distribution.* With a slight abuse of notation, we also denote its density by $\pi(\mathbf{x})$, $\mathbf{x} \in \mathsf{X}$ (with respect to an anonymous reference measure). The set X is called the *state space*, and could be discrete, continuous or mixed (we assume throughout the paper that all spaces are Polish and equipped with Borel σ-algebras). We assume that the density π can be written as $\pi(\mathbf{x}) = \gamma(\mathbf{x})/Z$, where the unnormalized density $\gamma(\mathbf{x})$ can be computed point-wise, whereas evaluating the *normalization constant* $Z = \int \gamma(\mathbf{x})\mathrm{d}\mathbf{x}$ may be computationally challenging. The two problems with which we are concerned are *(1)* approximating the normalization constant Z, and *(2)*, computing integrals under π of some *test function* $f : \mathsf{X} \to \mathbb{R}$, $\int f(\mathbf{x})\pi(\mathbf{x})\mathrm{d}\mathbf{x}$, where $f(\mathbf{x})$ can be computed point-wise. In a Bayesian context, *(1)* corresponds to approximating the marginal likelihood of the observed data, and *(2)*, computing the posterior expectation of some function, f, of the parameters and latent variables, $\mathbf{x}$.

2.2 Probabilistic graphical models

Problems *(1)* and *(2)* often arise in the context of PGMs, a formalism to encode dependencies between random variables in a probabilistic model. Two sorts of graphical structures are commonly used by statisticians to describe model dependencies: the Bayesian Network

(Pearl, 1985), which summarizes the conditional independence structure of a Bayesian model using a directed acyclic graph, and undirected graphs, which are often used to describe models specified via the full conditional distribution of each node such as Markov random fields (see below) and many spatial models such as conditional autoregressions (Besag, 1974). Here, we focus on the abstract factor graph formalism, and remind the reader that the two formalisms mentioned above can be easily converted to factor graphs; see, e.g., Bishop (2006) for details.

Two assumptions are required in order to write a model as a factor graph. First, that the state space, X, takes the form of a product space, $\mathsf{X} = \mathsf{X}_n = \widetilde{\mathsf{X}}_1 \times \widetilde{\mathsf{X}}_2 \times \cdots \times \widetilde{\mathsf{X}}_n$. It is convenient to define the set of *variables*, $V = \{1, \ldots, n\}$, corresponding to the elements of this factorization. Second, that the unnormalized density γ can be decomposed as, $\gamma(\mathbf{x}_n = (\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_n)) = \prod_{\phi \in F} \phi(S_\phi(\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_n))$, where F is a set of *factors* and the function S_ϕ returns a sub-vector of $(\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_n)$ containing those elements upon which factor ϕ depends.

Under these assumptions, a factor graph can be defined as a bipartite graph, where the set of vertices is given by $F \cup V$, and where we place an edge between a variable $v \in V$ and a factor $\phi \in F$ whenever the function ϕ depends on $\widetilde{\mathsf{X}}_v$, i.e. when $\widetilde{\mathbf{x}}_v$ is included in the vector returned by $S_\phi(\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_n)$. Throughout the paper, we use the convention that a variable with a tilde denotes a variable taking values in a single subspace ($\widetilde{\mathbf{x}}_n \in \widetilde{\mathsf{X}}_n$), while variables without tilde are elements of a product space ($\mathbf{x}_n = (\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_n) \in \mathsf{X}_n = \widetilde{\mathsf{X}}_1 \times \ldots \widetilde{\mathsf{X}}_n$). Note, however, that each $\widetilde{\mathsf{X}}_j$ may be multi-dimensional.

2.3 Sequential Monte Carlo

Sequential Monte Carlo (SMC) methods constitutes a class of sampling algorithms capable of addressing problems *(1)* and *(2)* defined in Section 2.1. More precisely, SMC can be used to simulate from a *sequence of probability distributions* defined on a sequence of spaces of increasing dimension. Let $\pi_t(\mathbf{x}_t)$, with $\mathbf{x}_t := (\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_t)$, be a probability density function defined on the product space

$$\mathsf{X}_t = \widetilde{\mathsf{X}}_1 \times \widetilde{\mathsf{X}}_2 \times \cdots \times \widetilde{\mathsf{X}}_t. \tag{1}$$

Furthermore, as above, assume that $\pi_t(\mathbf{x}_t) = \gamma_t(\mathbf{x}_t)/Z_t$ where γ_t can be evaluated point-wise, but where the normalizing constant Z_t is computationally intractable. SMC provides a way to sequentially approximate the sequence of distributions $\pi_1, \pi_2, \ldots, \pi_n$. As a byproduct, it also provides *unbiased* estimates of the normalizing constants $Z_1, Z_2, \ldots, Z_n$ (Del Moral, 2004, Prop. 7.4.1).

The SMC approximation of π_t at iteration t ($1 \leq t \leq n$) takes the form of a *particle population.* This population consists in a collection of N pairs of *particles* and *weights*:

$\{\mathbf{x}_t^i, \mathbf{w}_t^i\}_{i=1}^N$, where $\mathbf{x}_t^i \in \mathsf{X}_t$ and $\mathbf{w}_t^i \geq 0$. The particle population provides an approximation of π_t, in the (weak) sense that expectations of a (sufficiently regular) *test* function, f, with respect to the discrete probability distribution obtained after normalizing the weights,

$$\widehat{\pi}_t^N(\cdot) := \frac{1}{\sum_{j=1}^N \mathbf{w}_t^j} \sum_{i=1}^N \mathbf{w}_t^i \delta_{\mathbf{x}_t^i}(\cdot), \tag{2}$$

approximate the expectation of that test function under π_t:

$$\int \pi_t(\mathbf{x}_t) f(\mathbf{x}_t) \mathrm{d}\mathbf{x}_t \approx (\sum_{j=1}^N \mathbf{w}_t^j)^{-1} \sum_{i=1}^N \mathbf{w}_t^i f(\mathbf{x}_t^i).$$

One can consider test functions of direct interest (as well as considering the weak convergence of the approximating distributions which can be established under various conditions) for example, one would use $f(\mathbf{x}) = \mathbf{x}$ to approximate a mean, and $f(\mathbf{x}) = \mathbf{1}_A(\mathbf{x})$ to approximate the probability that $\mathbf{x} \in A$.

We review here the simplest type of SMC algorithm, Sequential Importance Resampling (SIR), and refer the reader to Doucet and Johansen (2011) for a more in-depth exposition. Pseudo-code for the SIR method is given in Algorithm 1. We present the algorithm in a slightly non-standard recursive form because it will be convenient to present the proposed D&C-SMC algorithm recursively, and presenting SIR in this way makes it easier to compare the two algorithms. Furthermore, since the focus of this paper is "static" problems (i.e., we are not interested in online inference, such as filtering), the sequential nature of the procedure need not be emphasized. For ease of notation, we allow the procedure to be called for $t = 0$, which returns an "empty" set of particles and, by convention, $\gamma_0(\emptyset) = 1$. (Hence, we do not need to treat the cases $t = 1$ and $t > 1$ separately in the algorithm.) The main steps of the algorithm, *resampling*, *proposal sampling*, and *weighting*, are detailed below.

Resampling (Line 3), in its simplest form, consists of sampling N times from the previous population approximation $\widehat{\pi}_{t-1}^N$, as defined in (2). This is equivalent to sampling the number of copies to be made of each particle from a multinomial distribution with number of trials N and probability vector $(\mathbf{w}_{t-1}^1, \ldots, \mathbf{w}_{t-1}^N)/(\sum_{i=1}^N \mathbf{w}_{t-1}^i)$. Since resampling is done with replacement, a given particle can be resampled zero, one, or multiple times. Informally, the goal of the resampling step is to prune particles of low weights in order to focus computation on the promising parts of the state space. This is done in a way that preserves the asymptotic guarantees of importance sampling. After resampling, the weights are reset to $1/N$, since the weighting is instead encoded in the random multiplicities of the particles. Note that more sophisticated resampling methods are available, see, e.g., Douc et al. (2005). Proposal sampling (Line 4), is based on user-provided proposal densities $q_t(\widetilde{\mathbf{x}}_t \mid \mathbf{x}_{t-1})$. For each particle

Algorithm 1 $\mathsf{sir}(t)$

```
1. If $t = 0$, return $(\{\emptyset, 1\}_{i=1}^N, 1)$.
2. $(\{\mathbf{x}_{t-1}^i, \mathbf{w}_{t-1}^i\}_{i=1}^N, \widehat{Z}_{t-1}^N) \leftarrow \mathsf{sir}(t-1)$.
3. Resample $\{\mathbf{x}_{t-1}^i, \mathbf{w}_{t-1}^i\}_{i=1}^N$ to obtain the unweighted particle population $\{\check{\mathbf{x}}_{t-1}^i, 1\}_{i=1}^N$.
4. For particle $i = 1, \dots, N$:
    (a) Simulate $\widetilde{\mathbf{x}}_t^i \sim q_t(\cdot \mid \check{\mathbf{x}}_{t-1}^i)$.
    (b) Set $\mathbf{x}_t^i = (\check{\mathbf{x}}_{t-1}^i, \widetilde{\mathbf{x}}_t^i)$.
    (c) Compute $\mathbf{w}_t^i = \frac{\gamma_t(\mathbf{x}_t^i)}{\gamma_{t-1}(\check{\mathbf{x}}_{t-1}^i)} \frac{1}{q_t(\widetilde{\mathbf{x}}_t^i \mid \check{\mathbf{x}}_{t-1}^i)}$.
5. Compute $\widehat{Z}_t^N = \left\{ \frac{1}{N} \sum_{i=1}^N \mathbf{w}_t^i \right\} \widehat{Z}_{t-1}^N$.
6. Return $(\{\mathbf{x}_t^i, \mathbf{w}_t^i\}_{i=1}^N, \widehat{Z}_t^N)$.
```

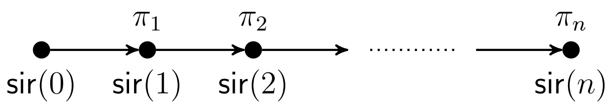


Figure 1: Computational flow of SIR (analogous for any SMC sampler). Each node corresponds to a call to sir, the labels above show the corresponding target distribution, and the arrows illustrate the recursive dependencies of the algorithm. Note that this "computational graph" of SMC is a chain, even if the sequence of target distributions does not correspond to a chain-structured PGM.

$\check{\mathbf{x}}_{t-1}^i \in \mathsf{X}_{t-1}$ output from the resampling stage, we sample a successor state $\widetilde{\mathbf{x}}_t^i \sim q_t(\cdot \mid \check{\mathbf{x}}_{t-1}^i)$. The sampled successor is a single state $\widetilde{\mathbf{x}}_t^i \in \widetilde{\mathsf{X}}_t$ which is appended to $\check{\mathbf{x}}_{t-1}^i$, to form a sample for the t-th product space, $\mathbf{x}_t^i = (\check{\mathbf{x}}_{t-1}^i, \widetilde{\mathbf{x}}_t^i) \in \mathsf{X}_t$. Finally, weighting (Line 4c) is used to correct for the discrepency between $\pi_{t-1}(\check{\mathbf{x}}_{t-1}^i) q_t(\widetilde{\mathbf{x}}_t^i | \check{\mathbf{x}}_{t-1}^i)$ and the new target $\pi_t(\check{\mathbf{x}}_{t-1}^i, \widetilde{\mathbf{x}}_t^i)$. Importantly, weighting can be performed on the unnormalized target densities γ_t and γ_{t-1}. The algorithm returns a particle-based approximation $\widehat{\pi}_t^N$ of π_t, as in (2), as well as an unbiased estimate $\widehat{Z}_t^N$ of Z_t (Line 5). In practice, an important improvement to this basic algorithm is to perform resampling only when particle degeneracy is severe. This can be done by monitoring the effective sample size (ESS): $(\sum_{i=1}^N \mathbf{w}_t^i)^2 / \sum_{i=1}^N (\mathbf{w}_t^i)^2$, and by resampling only when ESS is smaller than some threshold, say $N/2$ (Kong et al., 1994). In Figure 1 we illustrate the execution flow of the algorithm as arising from the recursive function calls.

The sequence of *target distributions* $\{\pi_t : t = 1, \dots, n\}$ can be constructed in many different ways, which largely explains the generality and success of SMC. The most basic construction, which is the classical application of SMC, arises from chain-structured factor graphs (for example, state-space models or hidden Markov models). For a chain-graph, the joint probability density function can be factorized as $\pi(\mathbf{x}) = \frac{1}{Z} \prod_{t=1}^n \phi_t(\widetilde{\mathbf{x}}_{t-1}, \widetilde{\mathbf{x}}_t)$, where $\mathbf{x} = (\widetilde{\mathbf{x}}_1, \dots, \widetilde{\mathbf{x}}_n)$; see Section 2.2. (As above, to simplify the notation we have, without loss of generality, introduced a "dummy variable" $\widetilde{\mathbf{x}}_0 = \emptyset$.) To simulate from the target distribution, the standard SIR algorithm employs a sequence of *intermediate distributions*: $\pi_t(\mathbf{x}_t) \propto \prod_{s=1}^{t} \phi_s(\widetilde{\mathbf{x}}_{s-1}, \widetilde{\mathbf{x}}_s)$, where $\mathbf{x}_t = (\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_t)$, $\widetilde{\mathbf{x}}_s \in \widetilde{\mathsf{X}}_s$. Each π_t can be written as γ_t / Z_t, where again γ_t can be evaluated point-wise, but Z_t is hard to compute. Importantly, we also have that $\pi_n = \pi$ by construction. In fact, it is possible to make use of similar *sequential decompositions* even when the original graph is not a chain (Naesseth et al., 2014), as long as it is possible to find a sequence of auxiliary distributions defined on increasing subsets of the model variables.

2.4 SMC samplers and tempering

Another common approach is to make use of a sequence of auxiliary distributions for which we are interested only in one of the marginals. Suppose that the densities of interest are defined over spaces which are not product spaces, $\widetilde{\pi}_t : \widetilde{\mathsf{X}}_t \to [0, \infty)$. For example, we may have $\widetilde{\pi}_t(\widetilde{\mathbf{x}}) \propto (\widetilde{\pi}(\widetilde{\mathbf{x}}))^{\alpha_t}$ as a tempered target distribution, with $\widetilde{\mathsf{X}}_t = \widetilde{\mathsf{X}}_{t-1} = \cdots = \widetilde{\mathsf{X}}_1$, and $q_t(\widetilde{\mathbf{x}}_t \,|\, \widetilde{\mathbf{x}}_{t-1})$ derived from a local MCMC move. We can transform problems of this type into a form suitable for SMC by an auxiliary construction proposed by Del Moral et al. (2006).

The construction used by Del Moral et al. (2006) is to re-introduce a sequence of distributions defined on product spaces $\mathsf{X}_t = \widetilde{\mathsf{X}}_1 \times \cdots \times \widetilde{\mathsf{X}}_t$ by defining,

$$\pi_t(\mathbf{x}_t) = \widetilde{\pi}_t(\widetilde{\mathbf{x}}_t) \prod_{s=1}^{t-1} L_s(\widetilde{\mathbf{x}}_s \,|\, \widetilde{\mathbf{x}}_{s+1}), \tag{3}$$

where $\mathbf{x}_t = (\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_t) \in \mathsf{X}_t$ as before. In the above, L_s is a transition kernel from $\widetilde{\mathsf{X}}_{s+1}$ to $\widetilde{\mathsf{X}}_s$—for instance an MCMC kernel—chosen by the user. For any choice of these *backward kernels*, π_t admits $\widetilde{\pi}_t$ as a marginal by construction, and it can thus be used as a proxy for the original target distribution $\widetilde{\pi}_t$. Standard SMC algorithms can then be applied to the sequence of auxiliary distributions π_t, $t = 1, \ldots, n$. Using the structure of π_t in (3), the weight computation (Line 4c of Algorithm 1) is given by:

$$\mathbf{w}_t^i = \frac{\widetilde{\gamma}_t(\widetilde{\mathbf{x}}_t^i)}{\widetilde{\gamma}_{t-1}(\widetilde{\mathbf{x}}_{t-1}^i)} \frac{L_{t-1}(\widetilde{\mathbf{x}}_{t-1}^i \,|\, \widetilde{\mathbf{x}}_t^i)}{q_t(\widetilde{\mathbf{x}}_t^i \,|\, \widetilde{\mathbf{x}}_{t-1}^i)}, \tag{4}$$

where $\widetilde{\gamma}_t \propto \widetilde{\pi}_t$. While the backward kernels L_t are formally arbitrary (subject to certain support restrictions), they will critically influence the estimator variance. If q_t is a $\widetilde{\pi}_{t-1}$-reversible MCMC kernel, a typical choice is $L_{t-1} = q_t$ which results in a cancellation in the weight expression (4): $\mathbf{w}_t^i = \widetilde{\gamma}_t(\widetilde{\mathbf{x}}_t) / \widetilde{\gamma}_{t-1}(\widetilde{\mathbf{x}}_t)$. See Del Moral et al. (2006) for further details and guidance on the selection of the backward kernels.

2.5 Related work

Before presenting the new methodology in Section 3 we note that a number of related ideas have appeared in the literature, although all have differed in key respects from the approach described in the next section.

Koller et al. (1999); Briers et al. (2005); Sudderth et al. (2010); Lienart et al. (2015) address belief propagation using (sequential) importance sampling, and these methods feature coalescence of particle systems, although they do not provide samples targeting a distribution of interest in an *exact* sense (Andrieu et al., 2010). In contrast, the method proposed here yields consistent estimates of the marginals and normalization constant, even when approximating a graphical model with loops. Moreover, our method can handle variables with constrained or discrete components, while much of the existing literature relies on Gaussian approximations which may not be practical in these cases.

Coalescence of particle systems in a different sense is employed by Jasra et al. (2008) who also use multiple populations of particles; here the state space of the full parameter vector is partitioned, rather than the parameter vector itself. The *island particle model* of Vergé et al. (2015) employs an ensemble of particle systems which themselves interact according to the usual rules of SMC, with every particle system targeting the same distribution over the full set of variables. The *local particle filtering* approach by Rebeschini and van Handel (2015) attempts to address degeneracy (in a hidden Markov model context) via an (inexact) localisation technique. Numerous authors have proposed custom SMC algorithms for the purpose of inferring the structure of a latent tree, see Teh et al. (2008); Bouchard-Côté et al. (2012); Lakshminarayanan et al. (2013). These methods generally employ a single particle population. In contrast, our method assumes a known tree decomposition, and uses several particle populations.

3 Methodology

The proposed methodology is useful when the inference problem described in Section 2.1 can be decomposed into a “tree of auxiliary distributions”, as defined in Section 3.1 below. We present the basic D&C-SMC method in Section 3.2, followed by fundamental convergence results in Section 3.3. Thereafter, we provide a concrete strategy for constructing the aforementioned tree-structured auxiliary distributions on which the D&C-SMC algorithm operates. This strategy applies to many directed and undirected graphical modelling scenarios of practical interest (including models with cycles). It should be noted that, as with standard SMC algorithms, a range of techniques are available to improve on the basic method presented in this section, and we discuss several possible extensions in Section 4.

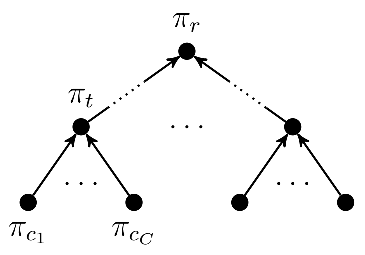


Figure 2: Computational flow of D&C-SMC. Each node corresponds to a target distribution $\{\pi_t : t \in T\}$ and, thus, to a call to D&C-SMC (Algorithm 2). The arrows illustrate the computational flow of the algorithm via its recursive dependencies.

3.1 Tree structured auxiliary distributions

The proposed D&C-SMC methodology generalizes the classical SMC framework from sequences (or chains) to trees. As noted in Section 2, the SMC methodology is a general framework for simulating from essentially any *sequence* of distributions. Any such sequence can be organized on a chain, with subsequent distributions being associated with neighbouring nodes on the chain; see Figure 1. Note that the graph notion here is used to describe the execution flow of the algorithm, and the sequence of distributions organized on the chain does not necessarily correspond to a chain-structured PGM.

In a similar way, D&C-SMC operates on a *tree of distributions*, which need not correspond to a tree-structured PGM. Specifically, as in Section 2.3, assume that we have a collection of (auxiliary) distributions, $\{\pi_t : t \in T\}$. However, instead of taking the index set T to be nodes in a sequence, $T = \{1, 2, \dots, n\}$, we generalize T to be nodes in a tree. For all $t \in T$, let $\mathcal{C}(t) \subset T$ denote the children of node t, with $\mathcal{C}(t) = \emptyset$ if t is a leaf, and let $r \in T$ denote the root of the tree. We assume π_t to have a density, also denoted by π_t, defined on a set X_t. We call such a collection of tree structured auxiliary distributions a *tree decomposition* of the target distribution π (introduced in Section 2.1) if it has two properties: First, the root distribution is required to coincide with the target distribution, $\pi_r = \pi$; second, a consistency condition: we require that the spaces on which the node distributions are defined are constructed recursively as

$$\mathsf{X}_t = \left(\otimes_{c \in \mathcal{C}(t)} \mathsf{X}_c\right) \times \widetilde{\mathsf{X}}_t, \tag{5}$$

where the "incremental" set $\widetilde{\mathsf{X}}_t$ can be chosen arbitrarily (in particular, $\widetilde{\mathsf{X}}_t = \emptyset$ for all t in some proper subset of the nodes in T is a valid choice). Note that the second condition mirrors the product space condition (1). That is, the distributions $\{\pi_t : t \in T\}$ are defined on spaces of increasing dimensions as we move towards the root from the leaves of the tree.

Figure 2 illustrates the execution flow of the D&C-SMC algorithm (which is detailed in the subsequent section), which performs inference for the distributions $\{\pi_t : t \in T\}$ from

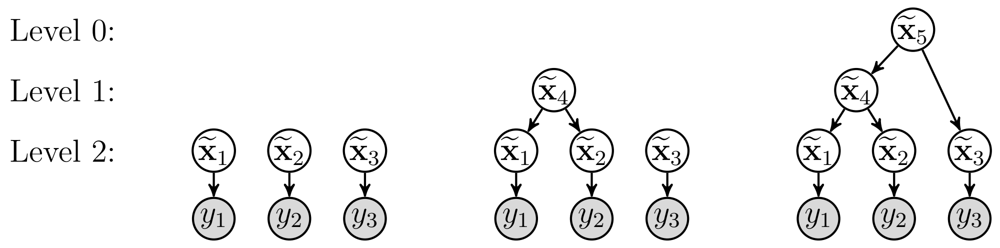

Level 0:
Level 1:
Level 2:
$\widetilde{\mathbf{x}}_5$
$\widetilde{\mathbf{x}}_4$
$\widetilde{\mathbf{x}}_1$
$\widetilde{\mathbf{x}}_2$
$\widetilde{\mathbf{x}}_3$
y_1
y_2
y_3

Figure 3: Decomposition of a hierarchical Bayesian model.

leaves to root in the tree. As pointed out above, the computational tree T does not necessarily correspond to a tree-structured PGM. Nevertheless, when the PGM of interest *is* in fact a tree, the computational flow of the algorithm can be easily related to the structure of the model (just as the computational flow of standard SMC is easily understood when the PGM is a chain, although the SMC framework is in fact more general). Let us therefore consider an example of how the target distributions $\{\pi_t : t \in T\}$ can be constructed in such a case, to provide some intuition for the proposed inference strategy before getting into the details of the algorithm.

Example (Hierarchical models). Consider the simple tree-structured Bayesian network of Figure 3 (rightmost panel), with three observations $y_{1:3}$, and five latent variables $\widetilde{\mathbf{x}}_{1:5}$. The distribution of interest is the posterior $p(\widetilde{\mathbf{x}}_{1:5} \,|\, y_{1:3})$. To put this in the notation introduced above, we define $\mathbf{x}_5 = \widetilde{\mathbf{x}}_{1:5}$ and $\pi(\mathbf{x}_5) = \pi_5(\mathbf{x}_5) = p(\widetilde{\mathbf{x}}_{1:5} \,|\, y_{1:3})$. To obtain a tree decomposition of π_5 we can make use of the hierarchical structure of the PGM. By removing the root node $\widetilde{\mathbf{x}}_5$ we obtain two decoupled components (Figure 3, middle) for which we can define the auxiliary target distributions $\pi_4(\mathbf{x}_4) = p(\mathbf{x}_4 \,|\, y_{1:2})$ and $\pi_3(\mathbf{x}_3) = p(\mathbf{x}_3 \,|\, y_3)$, respectively, where $\mathbf{x}_4 = (\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_4)$ and $\mathbf{x}_3 = \widetilde{\mathbf{x}}_3$. If the marginal priors for the root nodes in the decomposed models (here, $p(\widetilde{\mathbf{x}}_4)$ and $p(\widetilde{\mathbf{x}}_3)$) are intractable to compute, we can instead define the auxiliary distribution $\pi_t(\mathbf{x}_t)$, $t = 3, 4$, using an arbitrary "artificial prior" $u_t(\widetilde{\mathbf{x}}_t)$ for its root (similar to the two-filter smoothing approach of Briers et al. (2010)). This arbitrariness is ultimately corrected for by importance weighting and does not impinge upon the validity of the proposed inference algorithm (see Section 3.3), although the choice of u_t can of course affect the computational efficiency of the algorithm. Finally, by repeating this procedure, we can further decompose $\pi_4(\mathbf{x}_4)$ into two components, $\pi_1(\mathbf{x}_1)$ and $\pi_2(\mathbf{x}_2)$, as illustrated in Figure 3 (left). The target distributions $\{\pi_t(\mathbf{x}_t) : t \in \{1, \ldots, 5\}\}$ can be organised on a tree (with the same graph topology as the PGM under study, excluding the observed variables) which satisfies the conditions for being a tree decomposition of the sought posterior $p(\widetilde{\mathbf{x}}_{1:5} \,|\, y_{1:3})$.

In Section 3.4 we formalise the decomposition strategy illustrated in the example above, and also generalise it to a broader class of, so called, self-similar PGMs.

Algorithm 2 $\mathsf{dc\_smc}(t)$

```
1. For $c \in \mathcal{C}(t)$:
    (a) $(\{\mathbf{x}_c^i, \mathbf{w}_c^i\}_{i=1}^N, \widehat{Z}_c^N) \leftarrow \mathsf{dc\_smc}(c)$.
    (b) Resample $\{\mathbf{x}_c^i, \mathbf{w}_c^i\}_{i=1}^N$ to obtain the equally weighted particle system $\{\check{\mathbf{x}}_c^i, 1\}_{i=1}^N$.
2. For particle $i = 1, \ldots, N$:
    (a) If $\tilde{\mathsf{X}}_t \neq \emptyset$, simulate $\widetilde{\mathbf{x}}_t^i \sim q_t(\cdot \mid \check{\mathbf{x}}_{c_1}^i, \ldots, \check{\mathbf{x}}_{c_C}^i)$, where $(c_1, c_2, \ldots, c_C) = \mathcal{C}(t)$;
        else $\widetilde{\mathbf{x}}_t^i \leftarrow \emptyset$.
    (b) Set $\mathbf{x}_t^i = (\check{\mathbf{x}}_{c_1}^i, \ldots, \check{\mathbf{x}}_{c_C}^i, \widetilde{\mathbf{x}}_t^i)$.
    (c) Compute $\mathbf{w}_t^i = \frac{\gamma_t(\mathbf{x}_t^i)}{\prod_{c \in \mathcal{C}(t)} \gamma_c(\check{\mathbf{x}}_c^i)} \frac{1}{q_t(\widetilde{\mathbf{x}}_t^i \mid \check{\mathbf{x}}_{c_1}^i, \ldots, \check{\mathbf{x}}_{c_C}^i)}$.
3. Compute $\widehat{Z}_t^N = \left\{ \frac{1}{N} \sum_{i=1}^N \mathbf{w}_t^i \right\} \prod_{c \in \mathcal{C}(t)} \widehat{Z}_c^N$.
4. Return $(\{\mathbf{x}_t^i, \mathbf{w}_t^i\}_{i=1}^N, \widehat{Z}_t^N)$.
```

3.2 Divide-and-Conquer Sequential Importance Resampling

We now turn to the description of the D&C-SMC algorithm—a Monte Carlo procedure for approximating the target distribution $\pi = \pi_r$ based on the auxiliary distributions $\{\pi_t : t \in T\}$. For pedagogical purposes, we start by presenting the simplest possible implementation of the algorithm, which can be thought of as the analogue to the SIR implementation of SMC. Several possible extensions are discussed in Section 4.

As in standard SMC, D&C-SMC approximates each π_t by a collection of weighted samples, also referred to as a particle population. Unlike a standard SMC sampler, however, the method maintains multiple *independent* populations of weighted particles, $(\{\mathbf{x}_t^i, \mathbf{w}_t^i\}_{i=1}^N : t \in T_k)$, which are propagated and merged as the algorithm progresses. Here $T_k \subset T$ is the set of indices of "active" target distributions at iteration k, $1 \leq k \leq \text{height}(T)$.

The D&C-SMC algorithm uses a bottom-up approach to simulate from the auxiliary target distributions defined on the tree, by repeated resampling, proposal, and weighting steps, which closely mirror standard SMC. We describe the algorithm by specifying the operations that are carried out at each node of the tree, leading to a recursive definition of the method. For $t \in T$, we define a procedure $\mathsf{dc\_smc}(t)$ which returns, *(1)* a weighed particle population $\{\mathbf{x}_t^i, \mathbf{w}_t^i\}_{i=1}^N$ approximating π_t as $\widehat{\pi}_t^N$ in Equation (2), and *(2)* an estimator $\widehat{Z}_t^N$ of the normalizing constant Z_t (such that $\pi_t(\mathbf{x}_t) = \gamma_t(\mathbf{x}_t)/Z_t$). The procedure is given in Algorithm 2.

The first step of the algorithm is to acquire, for each child node $c \in \mathcal{C}(t)$, a particle approximation of π_c by a recursive call (Line 1a). Jointly, these particle populations provide an approximation of the product measure,

$$\otimes_{c \in \mathcal{C}(t)} \pi_c(\mathrm{d}\mathbf{x}_c) \approx \otimes_{c \in \mathcal{C}(t)} \widehat{\pi}_c^N(\mathrm{d}\mathbf{x}_c). \tag{6}$$

Note that this point-mass approximation has support on N^C, $C = |\mathcal{C}(t)|$, points, although these support points are implicitly given by the NC unique particles (assuming no duplicates among the particles in the individual child populations).

To obtain a computationally manageable approximation of the product measure, we generate N samples from the approximation in (6). This is equivalent to performing standard multinomial resampling for each child particle population (Line 1b), obtaining equally weighted samples $\{\check{\mathbf{x}}_c^i, 1\}_{i=1}^N$ for each c, and for all $i = 1, \ldots, N$, combining all indices i of the c lists to create N equally weighted tuples, $\{(\check{\mathbf{x}}_{c_1}^i, \ldots, \check{\mathbf{x}}_{c_C}^i), 1)_{i=1}^N$. This basic merging strategy can thus be implemented in $O(N)$ computational cost, since there is no need to explicitly form the approximation of the product measure in (6).

The latter, resampling-based, description of how the child populations are merged provide natural extensions to the methodology, e.g. by using low-variance resampling schemes (e.g., Carpenter et al. (1999)) and adaptive methods that monitor effective sample size to perform resampling only when particle degeneracy is severe (Kong et al., 1994).

Remark 1. Note, however, that if we perform resampling amongst the child populations separately and then combine the resulting particles in this way, we require $\mathbb{P}(\check{\mathbf{x}}_c^i = \mathbf{x}_c^j) = (\sum_{l=1}^N \mathbf{w}_c^l)^{-1}\mathbf{w}_c^j$, $j = 1, \ldots, N$, for each $i = 1, \ldots, N$, since the particles are combined based on their indices (i.e., it is not enough that the marginal equality $\sum_{i=1}^N \mathbb{P}(\check{\mathbf{x}}_c^i = \mathbf{x}_c^j) = N(\sum_{l=1}^N \mathbf{w}_c^l)^{-1}\mathbf{w}_c^j$ holds). Consequently, if the resampling mechanism that is employed results in an ordered list of resampled particles, then a random permutation of the particles indices should be carried out before combining particles from different child populations.

Proposal sampling (Line 2), similarly to standard SMC, is based on user-provided proposal densities q_t. However, the proposal has access to more information in D&C-SMC, namely to the state of all the children $c_1, c_2, \ldots, c_C$ of node t: $q_t(\cdot \mid \check{\mathbf{x}}_{c_1}^i, \ldots, \check{\mathbf{x}}_{c_C}^i)$. For each particle tuple $(\check{\mathbf{x}}_{c_1}^i, \ldots, \check{\mathbf{x}}_{c_C}^i)$ generated in the resampling stage, we sample a successor state $\widetilde{\mathbf{x}}_t^i \sim q_t(\cdot \mid \check{\mathbf{x}}_{c_1}^i, \ldots, \check{\mathbf{x}}_{c_C}^i)$. Note that in some cases, parts of the tree structured decomposition do not require this proposal sampling step, namely when $\widetilde{\mathsf{X}}_t = \emptyset$. We simply set $\widetilde{\mathbf{x}}_t^i$ to $\emptyset$ in these cases (the resampling and reweighting are still non-trivial).

Finally, we form the i-th sample at node t of the tree by concatenating the tuple of resampled child particles $(\check{\mathbf{x}}_{c_1}^i, \ldots, \check{\mathbf{x}}_{c_C}^i)$ and the proposed state $\widetilde{\mathbf{x}}_t^i$ (if it is non-empty). The importance weight is given by the ratio of the (unnormalised) target densities, divided by the proposal density (Line 2c). We use the convention here that $\prod_{c \in \emptyset}(\cdot) = 1$, to take into account the base case of this recursion, at the leaves of the tree.

Example (Hierarchical models, continued). A simple choice for $q_t(\cdot \mid \check{\mathbf{x}}_{c_1}, \ldots, \check{\mathbf{x}}^i_{c_C})$ in this example is to use u_t, the (artificial) prior at the sub-tree rooted at node $\widetilde{\mathbf{x}}_t$. An alternative is to select u_t as a conjugate prior to the distributions of the children, $p(\widetilde{\mathbf{x}}_c \mid \widetilde{\mathbf{x}}_t)$, $c \in \mathcal{C}(t)$, and to propose according to the posterior distribution of the conjugate pair. To illustrate the weight update, we show its simplified form in the simplest situation, where $q_t = u_t$:

$$\mathbf{w}^i_t = \frac{\gamma_t(\mathbf{x}^i_t)}{\prod_{c\in\mathcal{C}(t)} \gamma_c(\check{\mathbf{x}}^i_c)} \frac{1}{q_t(\widetilde{\mathbf{x}}^i_t \mid \check{\mathbf{x}}^i_{c_1}, \ldots, \check{\mathbf{x}}^i_{c_C})} = \frac{u_t(\mathbf{x}^i_t) \prod_{c\in\mathcal{C}(t)} p(\check{\mathbf{x}}^i_c \mid \widetilde{\mathbf{x}}^i_t)}{\prod_{c\in\mathcal{C}(t)} u_c(\check{\mathbf{x}}^i_c)} \frac{1}{u_t(\mathbf{x}^i_t)} = \prod_{c\in\mathcal{C}(t)} \frac{p(\check{\mathbf{x}}^i_c \mid \widetilde{\mathbf{x}}^i_t)}{u_c(\check{\mathbf{x}}^i_c)}.$$

If executed serially, the running time of D&C-SMC is $O(N\cdot|T|)$. However, a running time of $O(N\cdot\text{height}(T))$ can be achieved via parallelized or distributed computing (see Section 5.3). In terms of memory requirements, they grow at the rate of $O(N \cdot \text{height}(T) \cdot \max_t |\mathcal{C}(t)|)$ in a serial implementation. The height(T) factor comes from the maximum size of the recursion stack, and $N \max_t |\mathcal{C}(t)|$ comes from the requirement for each level of the stack to store a particle population for each child. In a parallel implementation where one compute node is used for each leaf of the computational tree the total memory requirement grows at the rate $O(N \cdot |T|)$.

Note that the D&C-SMC algorithm generalizes the usual SMC framework; if $|\mathcal{C}(t)| = 1$ for all internal nodes, then the D&C-SIR procedure described above reduces to a standard SIR method (Algorithm 1).

3.3 Theoretical Properties

As D&C-SMC consists of standard SMC steps combined with merging of particle populations via resampling, it is possible (with care) to extend many of the results from the standard, and by now well-studied, SMC setting (see e.g., Del Moral (2004) for a comprehensive collection of theoretical results). Here, we present two results to justify Algorithm 2. The proofs of the two propositions stated below are given in Section A of the Supplementary material.

First of all, the unbiasedness of the normalizing constant estimate of standard SMC algorithms (Del Moral, 2004, Prop. 7.4.1) is inherited by D&C-SMC.

Proposition 1. *Provided that $\gamma_t \ll \otimes_{c\in\mathcal{C}(t)}\gamma_c \otimes q_t$ for every $t \in T$ and an unbiased, exchangeable resampling scheme is applied to every population at every iteration, we have for any $N \geq 1$:*

$$\mathbb{E}[\widehat{Z}^N_r] = Z_r = \int \gamma_r(d\mathbf{x}_r).$$

An important consequence of Proposition 1 is that the D&C-SMC algorithm can be used to construct efficient block-sampling MCMC kernels in the framework of particle MCMC Andrieu et al. (2010); see Section 4.3. Our second result shows that the particle system generated by the D&C-SMC procedure is consistent as the number of particles tends to infinity.

Proposition 2. *Under regularity conditions detailed in Section A.2 of the Supplementary material, the weighted particle system* $\{\mathbf{x}^i_{r,N}, \mathbf{w}^i_{r,N}\}_{i=1}^N$ *generated by* $\mathsf{dc\_smc}(r)$ *is consistent in that for all functions* $f : \mathsf{X} \to \mathbb{R}$ *satisfying the assumptions listed in Section A.2 of the Supplementary material,*

$$\sum_{i=1}^{N} \frac{\mathbf{w}^i_{r,N}}{\sum_{j=1}^{N} \mathbf{w}^j_{r,N}} f(\mathbf{x}^i_{r,N}) \xrightarrow{\mathbb{P}} \int f(\mathbf{x})\pi(\mathbf{x})d\mathbf{x}, \qquad \text{as } N \to \infty.$$

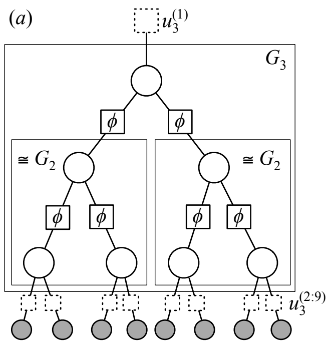
3.4 Tree structured auxiliary distributions from graphical models

We now present one strategy for building tree structured auxiliary distributions from graphical models. There are other ways of constructing these auxiliary distributions, but for concreteness we focus here on a method targeted at posterior inference for PGMs. On the other hand, the method we present here is more general than it may appear at first: in particular, although the flow of the algorithm follows a tree structure, we do not assume that the graphical models are acyclic.

To illustrate the concepts in this section, we will use two running examples: one coming from a directed PGM, and one coming from an undirected one. We use the factor graph notation from Section 2.2 to introduce the features of these examples salient to the present discussion. We give a more detailed description of the two examples in Section 5.

Example (Hierarchical models, continued). Consider a situation where the data is collected according to a known hierarchical structure. For example, test results for an examination are collected by school, which belong to a known school district, which belong to a known county. This situation is similar to the example shown in Figure 3, but where we generalize the number of level to be an arbitrary integer, α. This yields the factor graph shown in Figure 4(a), where we assume for simplicity a binary structure (this is lifted in Section 5). The nodes in the set V correspond to latent variables specific to each level of the hierarchy. For example, a variable, $\widetilde{\mathbf{x}}_v$, at a leaf encodes school-specific parameters from a set, $\widetilde{\mathsf{X}}_v$, those at the second level, district-specific parameters, etc. The set of factors, F, contain one binary factor, $\phi(\widetilde{\mathbf{x}}_v, \widetilde{\mathbf{x}}_{v'})$, between each internal node, v', and its parent, v. There is also one factor, u_r, at the root to encode a top level prior.

Example (Lattice models). Two-dimensional regular lattice models such as the Ising model are frequently used in spatial statistics and computer vision to encourage nearby locations in a spatial latent field to take on similar values; see Figure 4(b). We denote the width of the grid by $\alpha^{(1)}$ and the height by $\alpha^{(2)}$. The cardinality of V is thus $\alpha^{(1)}\alpha^{(2)}$. The bivariate factors connect variables with Manhattan distance of one to each other.



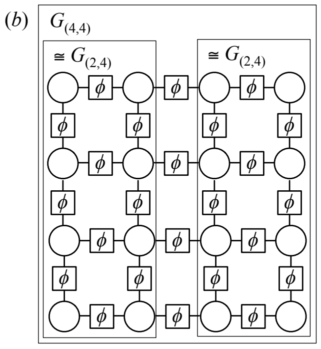


Figure 4: Examples of factor graph families, and self-similarities among them. (a) A hierarchical model for $\alpha = 3$. The unaries u_α consist in a product of 9 individual unary factors: one for the root, and 8 for the leaves (note that the binary factors connected to the 8 observed leaves can be considered as unary factors since one of their arguments is fixed and known for the purpose of posterior inference). Hence, $|V_3| = 7$, $|F_3| = 6$, $k_\alpha = 2$, $\alpha_1 = \alpha_2 = 2$. (b) A rectangular lattice model (e.g., an Ising model) for $\alpha = (4, 4)$. Here, $k_\alpha = 2$, $\alpha_1 = \alpha_2 = (2, 4)$.

Note that the previous two examples actually describe a collection of factor graphs indexed by α: in the hierarchical model example, α in a positive integer encoding the number of hierarchical levels; in the lattice model example, α is a pair of positive integers, $\alpha = (\alpha^{(1)}, \alpha^{(2)})$ encoding the width and height of the grid. To formalize this idea, we define a *model family* as a collection of factor graphs: $\mathcal{M} = \{G_\alpha = (V_\alpha, F_\alpha)\}$, where $V_\alpha \neq \emptyset$. Since we would like the concept of model family to encode the model structure rather than some observation-specific configurations, it will be useful in the following to assume that the sets F_α only contain factors linking at least two nodes. Given G_α and a dataset, it is trivial to add back the unary factors, denoted u_α. We assume that for all α, adding these unary factors to the product of the factors in F_α yields a model of finite normalization, $\int u_\alpha \prod_{\phi \in F_\alpha} \phi \mathrm{d}\mu < \infty$.

To build a tree of auxiliary distributions, we rely on a notion of self-similarity. We start with an illustration of what we mean by self-similarity in the two running examples.

Example (Hierarchical models, continued). Consider the factor graph $G_3 = (V_3, F_3)$ corresponding to a three-level hierarchical model. If we exclude the unary factor at the root, we see that G_3 contains G_2 as a subgraph (see Figure 4(a)). In fact, G_3 contains two distinct copies of the graph G_2.

Example (Lattice models, continued). Consider the factor graph $G_{(4,4)}$ corresponding to a 4-by-4 Ising model (Figure 4(b)). The graph $G_{(4,4)}$ contains the graph $G_{(2,4)}$ as a subgraph. Again, $G_{(4,4)}$ contains in fact two distinct copies of the subgraph.

Formally, we say that a model family is *self-similar*, if given any $G_\alpha \in \mathcal{M}$ with $|V_\alpha| > 1$, we can find $\alpha_1, \alpha_2, \ldots, \alpha_{k_\alpha}, k_\alpha > 1$ such that the disjoint union $\sqcup_i G_{\alpha_i}$ can be *embedded* in G_α. By embedding, we mean that there is a one-to-one graph homomorphism from $\sqcup_i G_{\alpha_i}$ into G_α. This graph homomorphism should respect the labels of the nodes and edges (i.e.

differentiates variable, factors, and the various types of factors).

Example (Lattice models, continued). Therefore, if $|V_\alpha| > 1$, then at least one of $\alpha^{(1)}$ or $\alpha^{(2)}$ is greater than one, let us say the first one without loss of generality. As shown in Figure 4(b), we can therefore pick $k_\alpha = 2$ and $\alpha_1 = (\lfloor \alpha^{(1)}/2 \rfloor, \alpha^{(2)})$, $\alpha_2 = (\lceil \alpha^{(1)}/2 \rceil, \alpha^{(2)})$.

Given a member α_0 of a self-similar model family, there is a natural way to construct a tree decomposition of auxiliary distributions. First, we recursively construct T from the self-similar model indices: we set $r = \alpha_0$, and given any $t = \alpha \in T$, we set $\mathcal{C}(t) \subset T$ to $\alpha_1, \alpha_2, \ldots, \alpha_{k_\alpha}$. This recursive process will yield a finite set T: since V_α is assumed to be non-empty, it suffices to show that $|V_{\alpha_i}| < |V_\alpha|$ for all $i \in \{1, \ldots, k_\alpha\}$ whenever $|V_\alpha| > 1$. But since $k_\alpha > 1$, and that the disjoint union $\sqcup_i G_{\alpha_i}$ can be embedded in G_α, it follows that $|V_\alpha| \geq |V_{\alpha_i}| + \sum_{j \neq i} |V_{\alpha_j}|$. Since $|V_{\alpha_j}| > 0$, the conclusion follows. Second, given an index $t = \alpha \in T$, we set π_t to $u_\alpha \prod_{\phi \in F_\alpha} \phi$. Note that by the embedding property, this choice is guaranteed to satisfy Equation (5), where X_{c_i} corresponds to the range of the random vector defined from the indices in V_{α_i}.

4 Extensions

Algorithm 2 is essentially an SIR algorithm where the variables are not rejuvenated after their first sampling. Inevitably, as in particle filtering, this will lead to degeneracy as the repeated resampling steps reduce the number of unique particles. Techniques employed to ameliorate this problem in the particle filtering literature could be used—fixed lag techniques (Kitagawa, 1996) might make sense in some settings, as could incorporating MCMC moves (Gilks and Berzuini, 2001). In this section we present several extensions to address the degeneracy problem more directly, and we also discuss adaptive schemes for improving the computational efficiency of the proposed method.

4.1 Merging subpopulations via mixture sampling

The resampling in Step 1b of the dc_smc procedure, which combines subpopulations to target a new distribution on a larger space, is critical. The independent multinomial resampling of child populations in the basic D&C-SIR procedure corresponds to sampling N times with replacement from the product measure (6). The low computational cost of this approach is appealing, but unfortunately it can lead to high variance when the product $\prod_{c \in \mathcal{C}(t)} \pi_c(\mathbf{x}_c)$ differs substantially from the corresponding marginal of π_t.

An alternative approach, akin to the mixture proposal approach (Carpenter et al., 1999) or the auxiliary particle filter (Pitt and Shephard, 1999), is described below. The idea is to exploit the fact that the product measure has mass upon $N^{|\mathcal{C}(t)|}$ points, in order to capture the dependencies among the variables in the target distribution $\pi_t(\mathbf{x}_t)$. Let $\breve{\pi}_t(\mathbf{x}_{c_1}, \dots, \mathbf{x}_{c_C})$ be some distribution which incorporates this dependency (in the simplest case we might take $\breve{\pi}_t(\mathbf{x}_{c_1}, \dots, \mathbf{x}_{c_C}) \approx \int \pi_t(\mathbf{x}_{c_1}, \dots, \mathbf{x}_{c_C}, \widetilde{\mathbf{x}}_t) \mathrm{d}\widetilde{\mathbf{x}}_t$ or, when $\widetilde{\mathsf{X}}_t = \emptyset$, $\breve{\pi}_t \equiv \pi_t$; see below for an alternative). We can then replace Step 1b of Algorithm 2 with simulating $\{(\breve{\mathbf{x}}_{c_1}^i, \dots, \breve{\mathbf{x}}_{c_C}^i)\}_{i=1}^N$ from

$$Q_t(\mathrm{d}\mathbf{x}_{c_1}, \dots, \mathrm{d}\mathbf{x}_{c_C}) := \sum_{i_1=1}^{N} \cdots \sum_{i_C=1}^{N} \frac{v_t(i_1, \dots, i_C)\delta_{(\mathbf{x}_{c_1}^{i_1}, \dots, \mathbf{x}_{c_C}^{i_C})}(\mathrm{d}\mathbf{x}_{c_1}, \dots, \mathrm{d}\mathbf{x}_{c_C})}{\sum_{j_1=1}^{N} \cdots \sum_{j_C=1}^{N} v_t(j_1, \dots, j_C)}, \tag{7}$$

$$v_t(i_1, \dots, i_C) := \left(\prod_{c \in \mathcal{C}(t)} \mathbf{w}_c^{i_c} \right) \breve{\pi}_t(\mathbf{x}_{c_1}^{i_1}, \dots, \mathbf{x}_{c_C}^{i_C}) \Big/ \prod_{c \in \mathcal{C}(t)} \pi_c(\mathbf{x}_c^{i_c}),$$

with the weights of Step 2c computed using $\mathbf{w}_t^i \propto \pi_t(\mathbf{x}_t^i) / \left[\breve{\pi}_t(\breve{\mathbf{x}}_{c_1}^{i_1}, \dots, \breve{\mathbf{x}}_{c_C}^{i_C}) q_t(\widetilde{\mathbf{x}}_t^i \mid \breve{\mathbf{x}}_{c_1}^{i_1}, \dots, \breve{\mathbf{x}}_{c_C}^{i_C})\right]$. It is necessary to be able to evaluate $\breve{\pi}_t$ only pointwise and up to a normalising constant, as in the standard setting. Naturally, if we take $\breve{\pi}_t(\mathbf{x}_{c_1}, \dots, \mathbf{x}_{c_C}) = \prod_{c \in \mathcal{C}(t)} \pi_c(\mathbf{x}_c)$ we recover the basic approach discussed in Section 3.2. The same unbiasedness and consistency properties detailed in Propositions 1 and 2 holds if this strategy is employed—see Section A of the Supplementary material for details.

The computational cost of simulating from Q_t will be $O(N^{|\mathcal{C}(t)|})$. However, we envisage that both $|\mathcal{C}(t)|$ and the number of coalescence events (i.e. combinations of subpopulations via this step) are sufficiently small that this is not a problem in many cases. Furthermore, if the mixture sampling approach is employed it significantly mitigates the negative impact of resampling, and it is possible to reduce the branching factor by introducing additional (dummy) internal nodes in T. For example, by introducing additional nodes in order to obtain a binary tree (see Section 5.1), the merging of the child populations will be done by coalescing pairs, then pairs of pairs, etc., gradually taking the dependencies between the variables into account.

Should the computational cost of evaluating the full joint distribution Q_t still be prohibitive, a computationally efficient alternative is to make use of a technique similar to multiple matching as in the independent particle filter by Lin et al. (2005, Section 2.2). Instead of sampling from the full product distribution (7), we sample first mN offsprings from each child population independently, according to their importance weights. We then resample N particles from the mN matchings $\{(\mathbf{x}_{c_1}^i, \dots, \mathbf{x}_{c_C}^i)\}_{i=1}^{mN}$ with weights proportional to $\breve{\pi}_t(\mathbf{x}_{c_1}^i, \dots, \mathbf{x}_{c_C}^i) / \prod_{c \in \mathcal{C}(t)} \pi_c(\mathbf{x}_c^i)$, $i = 1, \dots, mN$ to obtain $\{(\breve{\mathbf{x}}_{c_1}^i, \dots, \breve{\mathbf{x}}_{c_C}^i)\}_{i=1}^N$. The resulting importance weights $\mathbf{w}_t^i$ are computed as above. Essentially, this is an intermediate route between the basic D&C-SIR method and using the full product distribution (7), where $1 \leq m \ll N$ is a tuning parameter controlling the trade-off. Since this strategy aims to reap the benefits of mixture sampling, but at a smaller computational cost, we refer to it as lightweight mixture sampling.

Other possibilities, on which we do not elaborate here, include using the strategy of Briers et al. (2005) for efficiently sampling from a product of mixtures and (when dealing with simple local interactions) using techniques borrowed from N-body problems (Gray and Moore, 2001), at the cost of introducing a small but controllable bias.

4.2 SMC samplers and tempering within D&C-SMC

As discussed in Section 2.3, a common strategy when simulating from some complicated distribution using SMC is to construct a synthetic sequence of distributions (3) which evolves from something tractable to the target distribution of interest (Del Moral et al., 2006). The SMC proposals can then, for instance, be chosen as MCMC transition kernels—this is the approach that we detail below for clarity.

Step 2 of Algorithm 2 corresponds essentially to a (sequential) importance sampling step. Using the notation introduced in the previous section, we obtain after the resampling/mixture sampling step an unweighted sample $\{(\check{\mathbf{x}}^i_{c_1}, \ldots, \check{\mathbf{x}}^i_{c_C})\}_{i=1}^N$ targeting $\check{\pi}_t$, which is extended by sampling from $q_t(\widetilde{\mathbf{x}}_t \,|\, \mathbf{x}_{c_1}, \ldots, \mathbf{x}_{c_C})$, and then re-weighted to target $\pi_t(\mathbf{x}_t)$. We can straightforwardly replace this with several SMC sampler iterations, targeting distributions which bridge from $\pi_{t,0}(\mathbf{x}_t) = \check{\pi}_t(\mathbf{x}_{c_1}, \ldots, \mathbf{x}_{c_C}) q_t(\widetilde{\mathbf{x}}_t \,|\, \mathbf{x}_{c_1}, \ldots, \mathbf{x}_{c_C})$ to $\pi_{t,n_t}(\mathbf{x}_t) = \pi_t(\mathbf{x}_t)$, typically by following a geometric path $\pi_{t,j} \propto \pi_{t,0}^{1-\alpha_j} \pi_{t,n_t}^{\alpha_j}$ with $0 < \alpha_1 < \ldots < \alpha_{n_t} = 1$. Step 2 of Algorithm 2 is then replaced by:

```
2′. (a) For $i = 1$ to $N$, simulate $\widetilde{\mathbf{x}}^i_t \sim q_t(\cdot \,|\, \check{\mathbf{x}}^i_{c_1}, \ldots, \check{\mathbf{x}}^i_{c_C})$.
    (b) For $i = 1$ to $N$, set $\mathbf{x}^i_{t,0} = (\check{\mathbf{x}}^i_{c_1}, \ldots, \check{\mathbf{x}}^i_{c_C}, \widetilde{\mathbf{x}}^i_t)$ and $\mathbf{w}^i_{t,0} = 1$.
    (c) For SMC sampler iteration $j = 1$ to $n_t$:
        i. For $i = 1$ to $N$, compute $\mathbf{w}^i_{t,j} = \mathbf{w}^i_{t,j-1} \gamma_{t,j}(\mathbf{x}^i_{t,j-1}) / \gamma_{t,j-1}(\mathbf{x}^i_{t,j-1})$.
        ii. Optionally, resample $\{\mathbf{x}^i_{t,j-1}, \mathbf{w}^i_{t,j}\}_{i=1}^N$ and override the notation $\{\mathbf{x}^i_{t,j-1}, \mathbf{w}^i_{t,j}\}_{i=1}^N$
            to refer to the resampled particle system.
        iii. For $i = 1$ to $N$, draw $\mathbf{x}^i_{t,j} \sim K_{t,j}(\mathbf{x}^i_{t,j-1}, \cdot)$ using a $\pi_{t,j}$-reversible Markov kernel
             $K_{t,j}$.
    (d) Set $\mathbf{x}^i_t = \mathbf{x}^i_{t,n_t}$ and $\mathbf{w}^i_t = \mathbf{w}^i_{t,n_t}$.
```

The computation of normalizing constant estimates has been omitted for brevity, but follows by standard arguments (the complete algorithm is provided in Section **??** of the Supplementary material); again this leads to unbiased estimates (see Section A of the Supplementary material). Consistency follows directly from proposition 2, noting that annealing steps can be modeled as standard D&C-SMC steps by introducing empty dummy nodes to the tree.

We believe that the mixture sampling (or lightweight mixture sampling) approach introduced in Section 4.1 can be particularly useful when combined with SMC tempering as described above. The reason is that it enables efficient initialization of each (node-specific) SMC sampler by choosing, for $\alpha_\star \in [0,1]$,

$$\check{\pi}_t(\mathbf{x}_{x_1}, \ldots, \mathbf{x}_{c_C}) \propto \left[\prod_{c \in \mathcal{C}(t)} \pi_c(\mathbf{x}_c) \right]^{1-\alpha\star} \left[\int \pi_t(\mathbf{x}_{c_1}, \ldots, \mathbf{x}_{c_C}, \tilde{\mathbf{x}}_t) d\tilde{\mathbf{x}}_t \right]^{\alpha_\star} . \tag{8}$$

That is, we exploit the fact that the mixture sampling (or lightweight mixture sampling) distribution has support on $N^{|\mathcal{C}(t)|}$ (or mN) points to warm-start the annealing procedure at a non-zero value of the annealing parameter $\alpha_\star$. In practice, this has the effect that we can typically use fewer temperatures n_t, since we only need to bridge between $\alpha = \alpha_\star > 0$ and $\alpha = 1$. In particular, if simulating from the MCMC kernels $K_{t,j}$ is computationally costly, requiring fewer samples from these kernels can compensate for the $O(N^{|\mathcal{C}(t)|})$ (or $O(mN)$) computational cost associated with mixture sampling (or lightweight mixture sampling).

4.3 Particle MCMC

The seminal paper by Andrieu et al. (2010) demonstrated that SMC algorithms can be used to produce approximations of idealized block-sampling proposals within MCMC algorithms. By interpreting these particle MCMC algorithms as standard MCMC algorithms on an extended space, incorporating all of the variables sampled during the running of these algorithms, they can be shown to be exact, in the sense that the apparent approximation does not change the invariant distribution of the resulting MCMC kernel. Proposition 1, and in particular the construction used in its proof, demonstrates how the class of D&C-SMC algorithms can be incorporated within the particle MCMC framework. Such techniques are now essentially standard, and we do not dwell on this approach here.

4.4 Adaptation

Adaptive SMC algorithms have been the focus of much attention in recent years. Del Moral et al. (2012) provides the first formal validation of algorithms in which resampling is conducted only sometimes according to the value of some random quantity obtained from the algorithm itself. We advocate the use of low variance resampling algorithms (Douc et al., 2005, e.g.) to be applied adaptively. Other adaptation is possible within SMC algorithms. Two approaches are analyzed formally by Beskos et al. (2014): adapting the parameters of the MCMC kernels (step 2′(c)iii.), and the number and locations of tempering distributions, i.e., n_t and $\alpha_1, \ldots, \alpha_{n_t}$; see e.g., Zhou et al. (2015) for one approach to this.

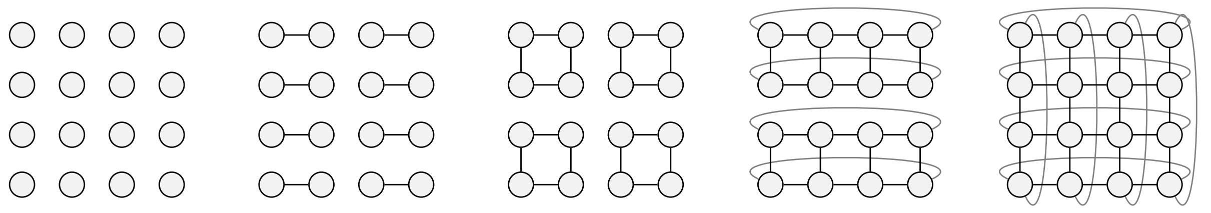


Figure 5: The disconnected components correspond to the groups of variables that are targeted by the different populations of the D&C-SMC algorithm. At the final iteration, corresponding to the rightmost figure, we recover the original, connected model.

Adaptation is especially appealing within D&C-SMC: beyond the usual advantages it allows for the concentration of computational effort on the more challenging parts of the sampling process. Using adaptation will lead to more intermediate distributions for the subproblems (i.e., the steps of the D&C-SMC algorithm) for which the starting and ending distributions are more different. Furthermore, it is also possible to adapt the parameter $\alpha_\star$ in (8)—that is, the starting value for the annealing process—based, e.g., on the effective sample size of the $N^{|\mathcal{C}(t)|}$ particles comprising (7), or the mN subsampled particles if using the lightweight mixture sampling approach. In simulations (see Section 5.1) we have found that the effect of such adaptation can result in $\alpha_\star = 1$ for many of the "simple" subproblems, effectively removing the use of tempering when this is not needed and significantly reducing the total number of MCMC simulations.

As a final remark, we have assumed throughout that all particle populations are of size N, but this is not necessary. Intuitively, fewer particles are required to represent simpler low-dimensional distributions than to represent more complex distributions. Manually or adaptively adjusting the number of particles used within different steps of the algorithm remains a direction for future investigation.

5 Experiments

5.1 Markov Random Field

One model class for which the D&C-SMC algorithm can potentially be useful are Markov random fields (MRFs). To illustrate this, we consider the well-known square-lattice Ising model. Each lattice site is associated with a binary spin variable $x_k \in \{-1, 1\}$ and the configuration probability is given by $p(\mathbf{x}) \propto e^{-\beta E(\mathbf{x})}$, where $\beta \geq 0$ is the inverse temperature and $E(\mathbf{x}) = -\sum_{(k,\ell)\in\mathcal{E}} x_k x_\ell$ is the energy of the system. Here, $\mathcal{E}$ denotes the edge set for the graphical model which we assume correspond to nearest-neighbour interactions with periodic boundary conditions, see Figure 5 (rightmost figure).

Let the lattice size be $M \times M$, with M being a power of 2 for simplicity. To construct the computational tree T we make use of the strategy of Section 3.4. That is, we start by dividing the lattice into two halves, removing all the edges between them. We then continue recursively, splitting each sub-model in two, until we obtain a collection of M^2 disconnected nodes; see Figure 5. This decomposition of the model defines a binary tree T of height $2\log_2 M$, on which the D&C-SMC algorithm operates. At the leaves we initialize M^2 independent particle populations by sampling uniformly on $\{-1, 1\}$. These populations are then resampled, merged, and reweighted as we proceed up the tree, successively reintroducing the "missing" edges of the model (note that $\widetilde{\mathsf{X}}_t = \emptyset$ for all non-leaf nodes t in this example). This defines the basic D&C-SIR procedure for the MRF. We also consider four extensions of this procedure:

D&C-SMC (mix) uses the mixture sampling strategy described in Section 4.1 with $\breve{\pi}_t = \pi_t$, i.e., the edges connecting any two sub-graphs are introduced before the corresponding sub-populations are merged.

D&C-SMC (ann) uses the tempering method discussed in Section 4.2, i.e., when the edges connecting two sub-graphs are reintroduced this is done gradually according to an annealing schedule to avoid severe particle depletion at the later stages of the algorithm.

D&C-SMC (ann+mix) uses tempering and mixture sampling, with $\breve{\pi}$ selected as in (8).

D&C-SMC (ann+lw) uses tempering and mixture sampling, with $\breve{\pi}$ selected as in (8), but employs the lightweight mixture sampling method described in Section 4.1 to reduce the computational cost of the merge step from N^2 to mN. We chose $m = 32$.

The D&C-SMC methods without annealing gave inferior results in this example, and D&C-SMC (ann+mix) performed similarly to D&C-SMC (ann+lw). We therefore focus our attention on D&C-SMC (ann) and D&C-SMC (ann+lw) in this section. The results for all other methods are given in Section B of the Supplementary material.

For the annealed methods, we use single-flip Metropolis-Hastings kernels. The annealing schedules are set adaptively based on the conditional ESS criterion of Zhou et al. (2015), with threshold of 0.995. For D&C-SMC (ann+lw) we warm-start each annealing process by selecting $\alpha_\star$ in (8) based on the ESS of the mN particle system with threshold 0.5; thus, $\alpha_\star$ is increased until the ESS is $0.5mN = 16N$ after which the system is subsampled to obtain N particles, for which the annealing process is then executed for α ranging from $\alpha_\star$ to 1. We also compare these methods with, *(i)* a standard SMC sampler with adaptive annealing Del Moral et al. (2006), and *(ii)* a single-flip Metropolis-Hastings sampler. In the Supplementary material we furthermore compare with a blocked Gibbs sampler—the results are qualitatively the same as the ones presented here. All methods were implemented in Matlab 9.0 on a Dell E7470.

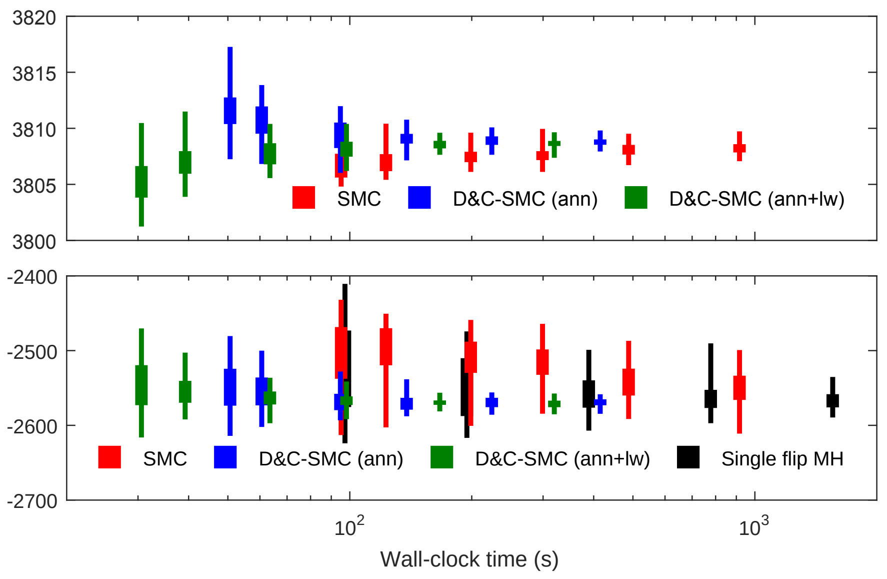


Figure 6: Box-plots (min, max, and inter-quartile) of estimates of $\log Z$ (top) and $\mathbb{E}[E(\mathbf{x})]$ (bottom) over 50 runs of each sampler (excluding single flip MH in the top panel since it does not readily provide an estimate of $\log Z$). The boxes, as plotted from left to right, correspond to increasing number of particles N (or number of MCMC iterations for single flip MH). This figure is best viewed in color.

We consider a grid of size 64×64 with $\beta = 0.4407$ (the critical temperature). The SMC-based methods used a number of particles ranging from $N = 2^6$ to 2^{11}. The single-flip MH sampler was run for 2^{14} MCMC iterations (each iteration being one complete sweep), with the first 2^9 iterations discarded as burn-in. We ran each method 50 times and considered the estimates of *(i)* the normalising constant Z and *(ii)* the expected value of the energy $\mathbb{E}[E(\mathbf{x})]$. The results are given in Figure 6. In this example, D&C-SMC (ann) and D&C-SMC (ann+lw) gave the overall best performance, significantly outperforming both standard SMC and single flip MH sampling for the same computational time. The two D&C-SMC samplers have comparable performances, slightly in favour of D&C-SMC (ann+lw).

It is interesting to note that D&C-SMC (ann+lw) has a lower computational cost than D&C-SMC (ann) for the same number of particles. This might at first seem counter-intuitive, since it employs a more costly merge step, but the reason is that using (lightweight) mixture sampling can result in that fewer annealing steps need to be taken. Indeed, for the simulations presented above, the SMC sampler used on average (over all different settings and runs) 685 MCMC updates for each site. The corresponding numbers for D&C-SMC (ann) and D&C-SMC (ann+lw) were 334 and 197, respectively. That is, for this example lightweight mixture sampling reduces the number MCMC iterations that are taken compared to D&C-SMC (ann) with about 40 % (which in turn uses only half the number of MCMC iterations compared to standard SMC). Hence, for models where simulation from the MCMC kernel is computationally expensive it can be worthwhile to use (lightweight) mixture sampling, even though the computational cost of the merge step itself is higher.

Table 1: Statistics for the merge steps of D&C-SMC (ann+lw) with 2 048 particles. The method runs for 12 iterations (not counting initialisation). #pop is the number of independent particle populations at the beginning of each iteration. The populations are then merged pair-wise before proceeding to the next iteration. #edges is the number of edges of the original MRF that are introduced in the merging of each pair of populations. $\alpha^\star$ (mean and standard deviations computed over all sites and the 50 independent runs of the sampler) is the value at which the annealing parameter is warm-started as explained in Section 4.2.

Iteration	1	2	3	4	5	6	7	8	9	10	11	12
#pop	4096	2048	1024	512	256	128	64	32	16	8	4	2
#edges	1	2	2	4	4	8	8	16	16	32	64	128
$\alpha^\star$ (mean)	1.0	1.0	1.0	0.971	0.965	0.585	0.581	0.371	0.370	0.245	0.169	0.111
$\alpha^\star$ (std $\cdot 10^3$)	0	0	0	3.9	4.0	4.3	3.1	2.5	2.5	1.7	1.2	1.0

The fact that (lightweight) mixture sampling automatically results in more computational effort being spent on the most difficult subproblems can be further illustrated by considering the values obtained for the parameter $\alpha_\star$ in (8) (recall that $\alpha_\star$ is chosen adaptively based on an ESS criterion). As pointed out above, $\alpha_\star \in [0, 1]$ is the value of the annealing parameter at which the annealing process is warm-started. In Table 1 we summarize some statistics recorded during the execution of D&C-SMC (ann+lw) with $N = 2\,048$ particles (similar results were obtained for the other settings as well). Note that, due to the way T is constructed, the number of edges that are "added to the model" at the merge steps increases as we move upward in T. Indeed, for this model the height of T is $2 \log_2(64) = 12$ and the number of edges that are added during the merge-steps of the 12 non-leaf levels are: 1, 2, 2, 4, 4, 8, 8, 16, 16, 32, 64, 128. From Table 1 we can see that at the first few iterations, when few edges are added in the merge steps, we obtained $\alpha_\star = 1$, meaning that no annealing was performed during these steps of the algorithm. For the subsequent levels we obtained values of $\alpha_\star$ less than 1, as can be seen in Table 1, but we were still able to warm-start the annealing at a non-zero value, effectively reducing the number of annealing steps needed.

In Section B of the Supplementary material we report additional numerical results for the Ising model (including different temperatures) as well as for another square-lattice MRF model with continuous latent variables and a multimodal posterior. These additional results are in general agreement with the ones presented here.

5.2 Hierarchical Bayesian Logistic Regression

In this section, we demonstrate the scalability of our method by analysing a dataset containing *New York State Mathematics Test* results for elementary and middle schools.

After preprocessing (data acquisition and preprocessing are described in detail in Section C of the Supplementary material), we organize the data into a tree T. A path from the root to a leaf has the following form: NYC (the root, denoted by $r \in T$), borough, school district, school, year. Each leaf $t \in T$ comes with an observation of m_t exam successes out of M_t trials. There were a total of 278 399 test instances in the dataset, split across five borough (Manhattan, The Bronx, Brooklyn, Queens, Staten Island), 32 distinct districts, and 710 distinct schools.

We use the following model, based on standard techniques from multi-level data analysis (Gelman and Hill, 2006). The number of successes m_t at a leaf t is assumed to be binomially distributed, with success probability parameter $p_t = \text{logistic}(\theta_t)$, where θ_t is a latent parameter. Moreover, we attach latent variables θ_t to internal nodes of the tree as well, and model the difference in values along an edge $e = (t \to t')$ of the tree with the following expression: $\theta_{t'} = \theta_t + \Delta_e$, where, $\Delta_e \sim \text{N}(0, \sigma_e^2)$. We put an improper prior (uniform on $(-\infty, \infty)$) on θ_r. However, when $m_t \notin \{0, M_t\}$ for at least one leaf, this can be easily shown to yield a proper posterior. We also make the variance random, but shared across siblings, $\sigma_e^2 = \sigma_t^2 \sim \text{Exp}(1)$.

We apply the basic D&C-SIR to this problem, using the natural hierarchical structure provided by the model (see Section 3.4). Note that conditionally on values for σ_t^2 and for the θ_t at the leaves, the other random variables are multivariate normal. Therefore, we instantiate values for θ_t only at the leaves, and when proposing at an internal node t', we only need to propose a value for $\sigma_{t'}^2$ as the internal parameters $\theta_{t'}$ can be analytically marginalized conditionally on $\sigma_{t'}^2$ and $\theta_{t'}$ using a simple message passing algorithm.

Each step of D&C-SMC therefore falls in exactly one of two cases: *(i)* At the leaves we propose a value for p_t from a Beta distribution with parameters $1 + m_t$ and $1 + M_t - m_t$, which we map deterministically to $\theta_t = \text{logit}(p_t)$. The corresponding weight update is a constant. *(ii)* At the internal nodes we propose $\sigma_t^2 \sim \text{Exp}(1)$ from its prior. The weight update ratio involves the densities of marginalized multivariate normal distributions which can be computed efficiently using message passing. Our Java implementation is open source and can be adapted to other multilevel Bayesian analysis scenarios (see Section D of the Supplementary material for instructions on how to do so).

The qualitative results obtained from D&C-SMC with 10 000 particles (Figure 14 of the Supplementary material) are in broad agreement with other socio-economic indicators. For example, among the five counties corresponding to each of the five boroughs, Bronx County has the highest fraction (41%) of children (under 18) living below poverty level (New York State Poverty Report, 2013). Queens has the second lowest (19.7%), after Richmond (Staten Island, 16.7%). However the fact that Staten Island contains a single school district means that our posterior distribution is much flatter for this borough.

For comparison, we also applied three additional methods (further details on the baselines and the experimental setup can be found in Section C of the Supplementary material):

Gibbs: A Metropolis-within-Gibbs algorithm, proposing a change on a single variable using a normal proposal of unit variance. As with D&C-SIR, we marginalize the internal θ_t parameters. (Java implementation.)

STD: A standard (single population) bootstrap particle filter with the intermediate distributions being sub-forests incrementally built in post-order. The internal θ_t-parameters are marginalized. (Java implementation.)

Stan: An open-source implementation of the Hamiltonian Monte Carlo algorithm (Carpenter et al., 2016). We did not implement marginalization of the internal θ_t-parameters. Stan includes a Kalman inference engine, however it is limited to chain-shaped PGMs as of version 2.6.0. (C++ implementation.)

We measure efficiency using effective sample size (ESS) per minute, as well as convergence of the posterior distributions on the parameters. For the MCMC methods (Gibbs and Stan), the ESS is estimated using the standard auto-regressive method, as implemented by Plummer et al. (2006). For the SMC methods (D&C-SMC and STD), the non-sequential nature of the samples dictates a different estimator, hence, again following standard practices, we use the estimator described by Kong et al. (1994). For both MCMC and SMC methods, wall-clock time is measured on Intel Xeon E5430 quad-core processors, running at 2.66 GHz. These experiments are replicated ten times. Standard deviations are indicated in parentheses.

We begin with the ESS per minute results for the SMC methods ran with 10 000 particles. For D&C-SMC, we obtained a mean ESS/min of 636.8 (19.3), and for STD, of 537.8 (53.2). The diagnostics suggest that both methods perform reasonably well, with a slight advantage to D&C-SMC. In contrast, the MCMC diagnostics raised inefficiency concerns. For Gibbs (300 000 iterations), we obtained a mean ESS/min of 0.215 (0.010). The performance of Stan (20 000 iterations) was inferior, and more volatile, with a mean of 0.000848 (0.0016). We attribute the poor performance of the Stan baseline to the fact that it does not marginalize the parameters θ (the reason for this is explained in the previous section).

Since the different types of samples impose the use of two different ESS estimators, direct comparisons of ESS/min between an SMC and an MCMC method should be taken with a pinch of salt. However, these results show that the sampling problem we are investigating in this section is indeed a challenging one. This is not a surprise, given the high-dimensionality of the latent variables (3 555 remaining parameters after marginalization of the multivariate normal). Moreover, our results on the convergence of the posterior distributions on the parameters (Section C of the Supplementary material) recapitulate that *(i)* the SMC methods strongly outperform the MCMC baselines in this problem, and *(ii)* D&C-SMC and STD perform similarly, with a slight advantage for D&C-SMC.

Next, to better differentiate the two SMC methods, we investigate estimation of the log-normalizing constant $\log(Z)$. The results are shown in Figure 7. By combining Proposition 1 with Jensen's inequality, we have the bound $\mathbb{E}[\log \widehat{Z}^N] \leq \log Z$. Since both D&C-SMC and STD have this property, we can conclude that higher mean values for the estimates are indicative of better performance (see also Section C of the Supplementary material, where

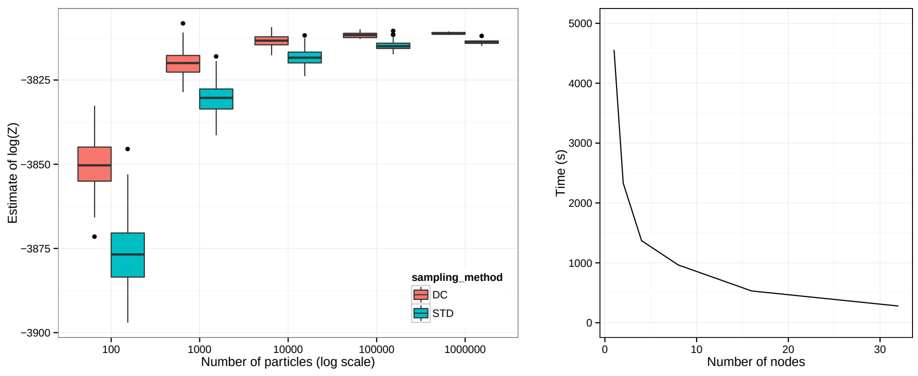


Figure 7: Left: Estimates of $\log(Z)$ obtained using D&C-SMC and STD with different numbers of particles. Each experiment was replicated 110 times (varying the Monte Carlo seed), except for the experiments with 1M particles, which were replicated only 10 times. Right: Wall-clock times for the distributed D&C-SMC algorithm. See Section C of the Supplementary material for speed-up results.

we use an example such that the true value of Z is known, to empirically validate the correctness of our implementation of the log normalization computation). Therefore we can conclude that D&C-SMC outperforms STD on all computational budgets.

5.3 Distributed Divide-and-Conquer

To demonstrate the suitability of D&C-SMC to distributed computing environments, we have implemented a proof-of-concept distributed D&C-SMC algorithm. The main idea in this implementation is to split the work at the granularity of populations, instead of the more standard particle granularity. The description and benchmarking of this implementation can be found in Section C of the Supplementary material. Using this distributed implementation, we see for example (Figure 7, right) that the running time for 100 000 particles can be reduced from 4 557 seconds for one machine (about $1\frac{1}{4}$ hours), to 279 seconds (less than five minutes) using 32 compute nodes (each using a single thread).

6 Discussion

We have shown that trees of auxiliary distributions can be leveraged by D&C-SMC samplers to provide computationally efficient approximations of the posterior distribution of high-dimensional and possibly loopy probabilistic graphical models. Our method, which generalizes the SMC framework, is easy to distribute across several compute nodes.

As with standard SMC (and other advanced computational inference methods) D&C-SMC allows for a large degree of flexibility, and the method should be viewed as a toolbox rather than as a single algorithm. Indeed, we have discussed several possible extensions of the basic method, and their utility is problem-specific. Furthermore, the interplay between these extensions needs to be taken into account. In particular, based on the numerical results in Section 5.1 we argue that (lightweight) mixture sampling can be useful when used in conjunction with MCMC-based tempering, especially when simulating from the MCMC kernel is computationally costly. In such scenarios, the warm-starting of the tempering process enabled by mixture sampling can compensate for the increased computational cost of mixture sampling.

We have assumed in this work that the topology of the tree of auxiliary distributions is known and fixed. In practice, several different decompositions are possible. We have presented one systematic way of obtaining a tree decomposition for self-similar graphical models. However, a natural question to ask is how to choose an optimal decomposition. We are exploring several approaches to address this question, including strategies that mix several decompositions. How the components of these mixtures should interact is a question we leave for future work.

Supplementary material

Appendices: The appendices contain proofs of Propositions 1 and 2 as well as additional details and results on the numerical examples. (supplementary-material.pdf)

Code: The code used to generate the results in Sections 5.1 and 5.2 is publicly available on GitHub, at `https://github.com/freli005/divide-and-conquer-smc` and `https://github.com/alexandrebouchard/divide-and-conquer-smc`, respectively.

Acknowledgments

This research is financially supported by: the Swedish research Council (VR) via the projects *Learning of complex dynamical systems* (Ref no. 637-2014-466) and *Probabilistic modeling of dynamical systems* (Ref no. 621-2013-5524) and the Linnaeus Center CADICS; the Engineering and Physical Sciences Research Council (UK, Ref no. EP/K021672/2), and; the Natural Sciences and Engineering Research Council (Canada) via a Discovery Grant.

References

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):269–342.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice structures (with discussion). *Journal of the Royal Statistical Society: Series B*, 36:192–236.

Beskos, A., Jasra, A., Kantas, N., and Thiéry, A. H. (2014). On the convergence of adaptive sequential Monte Carlo algorithms. arxiv, arxiv:1306.6462v3.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bouchard-Côté, A., Sankararaman, S., and Jordan, M. I. (2012). Phylogenetic inference via sequential Monte Carlo. *Systematic Biology*, 61(4):579–593.

Briers, M., Doucet, A., and Maskell, S. (2010). Smoothing algorithms for state-space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61–89.

Briers, M., Doucet, A., and Singh, S. S. (2005). Sequential auxiliary particle belief propagation. In *Proceedings of the 8th International Conference on Information Fusion*, PA, USA.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., and Riddell, A. (2016). Stan: A probabilistic programming language. *Journal of Statistical Software*. In Press.

Carpenter, J., Clifford, P., and Fearnhead, P. (1999). Improved particle filter for nonlinear problems. *IEE Proceedings Radar, Sonar and Navigation*, 146(1):2–7.

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika*, 89(3):539–551.

Del Moral, P. (2004). *Feynman-Kac Formulae - Genealogical and Interacting Particle Systems with Applications*. Probability and its Applications. Springer.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 68(3):411–436.

Del Moral, P., Doucet, A., and Jasra, A. (2012). On adaptive resampling procedures for sequential Monte Carlo methods. *Bernoulli*, 18(1):252–278.

Douc, R., Cappé, O., and Moulines, E. (2005). Comparison of resampling schemes for particle filters. In *Proceedings of the 4th IEEE International Symposium on Image and Signal Processing and Analysis*, pages 64–69, Zagreb, Croatia.

Doucet, A. and Johansen, A. M. (2011). A tutorial on particle filtering and smoothing. In Crisan, D. and Rozovskii, B., editors, *The Oxford Handbook of Nonlinear Filtering*, pages 656–704. OUP.

Gelman, A. and Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. CUP.

Gilks, W. R. and Berzuini, C. (2001). Following a moving target – Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society. Series B*, 63(1):127–146.

Gray, A. G. and Moore, A. W. (2001). 'N-body' problems in statistical learning. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems (NIPS) 13*. MIT Press.

Jasra, A., Doucet, A., Stephens, D. A., and Holmes, C. C. (2008). Interacting sequential Monte Carlo samplers for trans-dimensional simulation. *Computational Statistics and Data Analysis*, 52(4):1765–1791.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25.

Koller, D., Lerner, U., and Angelov, D. (1999). A general algorithm for approximate inference and its application to hybrid Bayes nets. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 15, pages 324–333.

Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288.

Lakshminarayanan, B., Roy, D. M., and Teh, Y. W. (2013). Top-down particle filtering for Bayesian decision trees. In *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, GA, USA.

Lienart, T., Teh, Y. W., and Doucet, A. (2015). Expectation particle belief propagation. In *Advances in Neural Information Processing Systems (NIPS) 28*.

Lin, M. T., Zhang, J. L., Cheng, Q., and Chen, R. (2005). Independent particle filters. *Journal of the American Statistical Association*, 100(472):1412–1421.

Naesseth, C. A., Lindsten, F., and Schön, T. B. (2014). Sequential Monte Carlo for graphical models. In *Advances in Neural Information Processing Systems (NIPS) 27*, pages 1862–1870.

New York State Community Action Association (2013). New york state poverty report. `http://ams.nyscommunityaction.org/`.

Pearl, J. (1985). Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society*, pages 329–334, University of California, Irvine, CA, USA. Cognitive Science Society.

Pitt, M. K. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599.

Plummer, M., Best, N., Cowles, K., and Vines, K. (2006). Coda: Convergence diagnosis and output analysis for mcmc. *R News*, 6(1):7–11.

Rebeschini, P. and van Handel, R. (2015). Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866.

Sudderth, E. B., Ihler, A. T., Isard, M., Freeman, W. T., and Willsky, A. S. (2010). Nonparametric belief propagation. *Communications of the ACM*, 53(10):95–103.

Teh, Y. W., Daumé III, H., and Roy, D. (2008). Bayesian agglomerative clustering with coalescents. In *Advances in Neural Information Processing Systems (NIPS) 20*, pages 1473–1480. MIT Press.

Vergé, C., Dubarry, C., Del Moral, P., and Moulines, E. (2015). On parallel implementation of sequential monte carlo methods: the island particle model. *Statistics and Computing*, 25(2):243–260.

Zhou, Y., Johansen, A. M., and Aston, J. A. D. (2015). Towards automatic model comparison: An adaptive sequential Monte Carlo approach. *Journal of Computational and Graphical Statistics*. In press.