Statistics and Computing (2022) 32:80
https://doi.org/10.1007/s11222-022-10113-2

# Sequential sampling of junction trees for decomposable graphs

**Jimmy Olsson**[1] · **Tatjana Pavlenko**[2] · **Felix L. Rios**[1]

Received: 19 May 2021 / Accepted: 25 April 2022 / Published online: 19 September 2022


**Abstract**
The *junction-tree* representation provides an attractive structural property for organising a *decomposable graph*. In this study, we present two novel stochastic algorithms, referred to as the *junction-tree expander* and *junction-tree collapser*, for sequential sampling of junction trees for decomposable graphs. We show that recursive application of the junction-tree expander, which expands incrementally the underlying graph with one vertex at a time, has full support on the space of junction trees for any given number of underlying vertices. On the other hand, the junction-tree collapser provides a complementary operation for removing vertices in the underlying decomposable graph of a junction tree, while maintaining the junction tree property. A direct application of the proposed algorithms is demonstrated in the setting of sequential Monte Carlo methods, designed for sampling from distributions on spaces of decomposable graphs. Numerical studies illustrate the utility of the proposed algorithms for combinatorial computations on decomposable graphs and junction trees. All the methods proposed in the paper are implemented in the Python library *trilearn*.

**Keywords** Random graphs · Decomposable graphs · Junction trees · Graphical models · Sequential Monte Carlo

## 1 Introduction

Decomposable graphs, sometimes also called *triangulated* or *chordal* graphs, are characterized by the property that every cycle of length more than three has an edge (or *chord*) joining two nonconsecutive vertices (Lauritzen 1996). Another characteristic property is that these graphs can be recursively decomposed into smaller graphs, called *cliques*, where every pair of vertices are connected by an edge. In this paper we rely on the fact that a graph is decomposable if and only if its cliques can be arranged into a so-called *junction tree*. Figure 1 shows an example of a decomposable graph along with one of its junction tree representations. Decomposable graphs and their junction-tree representations as auxiliary data structure have been used in various contexts; examples include computational geometry, estimation of large-scale random graph models with local dependence, statistical inference (such as sparse covariance- and concentration-matrix computation), contingency-table analysis, probabilistic graphical models, and message passing; see e.g. (Eppstein 2009; Lauritzen 1996; Pearl 1997).

This work is mainly driven by application of decomposability to probabilistic graphical models for representing conditional independence relations. From a statistical point of view, learning the underlying graph structure based on observed data in such models is particularly convenient since the graph likelihood has a closed form. However, the complexity of the graph space makes estimators such as the maximum likelihood graph estimates intractable, which has lead to an increasing interest in Bayesian methods, in particular in Monte Carlo methods for sampling-based approximations of the graph posterior.

The available methods are based on *Markov chain Monte Carlo* (MCMC) schemes (Tierney 1994), especially variations of the *Metropolis–Hastings algorithm* (Hastings 1970; Metropolis et al. 1953), where new graphs are proposed by means of random single-edge perturbations, and the set of possible moves generated by subjecting a given graph to such perturbations defines a neighborhood in the decomposable-graph space; see e.g. (Frydenberg and Lauritzen 1989;

✉ Felix L. Rios
flrios@kth.se

Jimmy Olsson
jimmyol@kth.se

Tatjana Pavlenko
tatjana.pavlenko@statistik.uu.se

[1] Department of Mathematics, KTH Royal Institute of Technology, 100 44 Stockholm, Sweden

[2] Department of Statistics, Uppsala University, Box 513, 751 20 Uppsala, Sweden

Giudici and Green 1999; Green and Thomas 2013; Thomas and Green 2009a). However, since the only vertices that may be connected by an edge in a (connected) decomposable graph while maintaining decomposability are those that already have a neighbour in common and the removable edges are necessarily contained in exactly one clique, operations on the edge set are inherently local. As a consequence, an MCMC sampler based on such moves will most likely suffer from mixing problems (Giudici and Green 1999; Green and Thomas 2013).

Green and Thomas (2013) showed that edge moves on decomposable graph space can sometimes be designed more easily if one operates on the extended junction-tree space. While this approach is mainly computationally motivated, it is feasible also from a statistical point of view; indeed, a given distribution on the space of decomposable graphs can always be embedded into an extended version defined on the space of junction trees in such a way that the push-forward distribution of the extended distribution with respect to the underlying graph equals the given distribution on the decomposable-graph space. Thus, by running an MCMC sampler producing a trajectory of junction trees targeting the extended distribution, an MCMC trajectory targeting the original distribution is obtained as a by-product by simply extracting the underlying graphs of the trees in the former sequence.

Against this background, it is desirable to explore alternative ways of simulating decomposable graphs. In the present paper we take a different approach than the above, which instead of altering the edge set of a graph with a fixed set of vertices, builds new graphs incrementally, starting from the empty graph and adding vertices one by one. More specifically, we present two novel stochastic algorithms operating on junction-tree structures: the *junction-tree expander* (JTE, or the *Christmas-tree algorithm*) and the *junction-tree collapser* (JTC). The JTE (JTC) expands (collapses) a junction tree by randomly adding (removing) one vertex to (from) the underlying decomposable graph. As we shall see, the JTE and JTC have two theoretical properties that are of fundamental importance in Monte Carlo simulation. First, the transition probabilities of the induced Markov kernels are available in a closed form and can be computed efficiently; second, the JTE algorithm is able to generate, with positive probability, when applied sequentially, all junction trees with a given number of vertices in its underlying graph.

In order to illustrate their application potential, we employ jointly the JTE and the JTC to construct a *sequential Monte Carlo* (SMC) *sampler* (Del Moral et al. 2006), sampling from more or less arbitrary distributions defined on spaces of decomposable graphs. In this construction, which relies on the above-mentioned junction-tree embedding proposed by Green and Thomas (2013), the JTC is used to extend the target distribution to a path space of junction trees of increasing dimension, whereas the JTE is used to generating proposals on this new space.

Using the SMC approach, we are able to provide unbiased estimates of the numbers of decomposable graphs and junction trees for any given number of vertices. This importance-sampling approach to the combinatorics of decomposable graphs and junction trees is the first of its kind. In the follow-up paper (Olsson et al. 2019), we cast further such an SMC sampler into the framework of *particle Gibbs samplers* (Andrieu et al. 2010). The resulting MCMC algorithm, which relies heavily on on the JTE and JTC derived in the present paper, allows for *global* MCMC moves across the decomposable-graph space and, consequently, weakly correlated samples and fast mixing.

The JTE is related to other existing approaches of generating junction trees. For instance, the algorithm presented in Markenzon et al. (2008) has similarities to ours in the sense that it expands the underlying graph incrementally in each step of the algorithm. However, unlike our proposed JTE, this algorithm is restricted to connected decomposable graphs and transition probabilities are not directly provided. A completely different strategy for decomposable-graph sampling based on *tree-dependent bipartiet graphs* is presented in Elmasri (2017a, b). A recent MCMC algorithm for joint sampling of general undirected graphs and corresponding concentration matrices in Gaussian graphical models is presented in van den Boom et al. (2022).

The rest of this paper is structured as follows. Sect. 2 introduces notational conventions and a short background on decomposable graphs and junction trees. For a more detailed presentation, the reader is referred to e.g. (Blair and Peyton 1993) or (Lauritzen 1996). Sect. 3 and Sect. 4 present the JTE and the JTC, respectively, along with their corresponding transition probabilities. Sect. 5 provides a novel factorisation of the number of junction trees of a decomposable graph and demonstrates its computational advantage. The application of the JTE and the JTC in the framework of SMC sampling is found in Sect. 6 and Sect. 7 contains our numerical study. Appendix A contains detailed algorithm descriptions along with the proofs of lemmas and theorems stated in the paper, whereas Appendix B provides an algorithm, originally presented in (Thomas and Green 2009b), for randomly connecting a forest into a tree.

Finally, we remark that the code used for generating the examples in the paper is contained in the Python library *trilearn* available at https://github.com/felixleopoldo/trilearn. The junction-tree expander is also available through Benchpress (Rios et al. 2021), a recent software that enables execution and seamless comparison between state-of-the-art structure learning algorithms. The junction-tree expander is implemented as a module in Benchpress to simulate graphs underlying data for benchmarking.

## 2 Preliminaries

### 2.1 Notational convention

For any finite set $a$, we denote its power set by $\wp(a)$. The uniform distribution over the elements in $a$ is denoted by $\mathsf{Unif}\langle a\rangle$. We assume that all random variables are well defined on a common probability space $(\Omega, \mathrm{F}, \mathbb{P})$.Abusing notation, we will always use the same notation for a random variable and a realisation of the same. Further, we will use the same notation for a distribution and its corresponding probability density function. For an arbitrary space $\mathsf{X}$, the *support* of a nonnegative function $h$ defined on $\mathsf{X}$ is denoted by $\mathrm{Supp}(h) := \{x \in \mathsf{X} : h(x) > 0\}$. For all sequences $(a_j)_{j=1}^{\ell}$, we apply the convention $(a_j)_{j=1}^{0} := \emptyset$. Moreover, for all sequences $(a_j)_{j=1}^{\ell}$ of sets and all nonempty sets $b$, we set $b \cap (\cup_{j=1}^{0} a_j) := b$. We denote by $\mathbb{N}$ the set of natural numbers $\{1, 2, \dots\}$ and by $\mathbb{N}_p$ the set $\{1, \dots, p\}$ for some $p \in \mathbb{N}$.

The notation, $\mathsf{pr}(\{w_\ell\}_{\ell=1}^{N})$ is used to denote the categorical distribution induced by a set $\{w_\ell\}_{\ell=1}^{N}$ of positive (possibly unnormalised) numbers. More specifically, writing $x \sim \mathsf{pr}(\{w_\ell\}_{\ell=1}^{N})$ means that the random variable $x$ takes on the value $\ell \in \mathbb{N}_N$ with probability $w_\ell / \sum_{\ell'=1}^{N} w_{\ell'}$.

### 2.2 Graph theory

A pair $(V, E)$ of a *vertex set* $V$ and an *edge set* $E$, where $E$ is a set of unordered pairs $(y, y') \in V \times V$ such that $y \neq y'$, is called an *undirected graph*. Two vertices $y$ and $y'$ in $V$ are *adjacent* if they are directly connected by an edge, *i.e.*, $(y, y') = (y', y)$ belongs to $E$. The *neighbors* $\mathfrak{N}_{(V,E)}(y)$ of a vertex $y$ is the set of vertices in $V$ adjacent to $y$. A sequence $(y_j)_{j=1}^{\ell}$ of distinct vertices is called an $y_1$-$y_\ell$-path, denoted by $y_1 \sim y_\ell$, if for all $j \in \{2, \dots, \ell\}$, $(y_{j-1}, y_j)$ belongs to $E$. Two vertices $y$ and $y'$ are said to be *connected* if there exists an $y$-$y'$-path. Moreover, a *graph* is said to be connected if all pairs of vertices are connected. A graph is called a *tree* if there is a unique path between any pair of vertices in the graph. A *connectivity component* of a graph is a subset of vertices that are pairwise connected. A graph is a *forest* if all connectivity components induce distinct trees. Further, two graphs are said to be *isomorphic* if they have the same number of vertices and equivalent edge sets when disregarding the labels of the vertices.

Now, consider a general graph $(V, E)$ which we call $G$. The *order* and the *size* of $G$ refer to the number of vertices $|V|$ and the number of edges $|E|$, respectively. Let $a$, $b$, and $s$ be subsets of $V$; then the set $s$ *separates* $a$ from $b$ if for all $y \in a$ and $y' \in b$, all paths $y \sim y'$ intersect $s$. We denote this by $a \perp_G b \mid s$. The graph $G$ is *complete* if all vertices are adjacent to each other. A graph $(V', E')$ is a *subgraph* of $G$ if $V' \subseteq V$ and $E' \subseteq E$. A *subtree* is a connected subgraph of a tree. For $V' \subseteq V$, the *induced subgraph* $G[V'] = (V', E')$ is the subgraph of $G$ with vertices $V'$ and edge set $E'$ given by the set of edges in $E$ having both endpoints in $V'$. A subset of $V$ is a *complete set* if it induces a complete subgraph. A complete subgraph is called a *clique* if it is not an induced subgraph of any other complete subgraph.

The primer interest of this paper regards *decomposable graphs* and the *junction-tree* representation.

**Definition 1** A graph $G$ is decomposable if its cliques can be arranged in a so-called *junction tree*, *i.e.* a tree whose nodes are the cliques in $G$, and where for any pair of cliques $c$ and $c'$ in $G$, the intersection $c \cap c'$ is contained in each of the cliques on the unique path $c \sim c'$.

Note that a decomposable graph may have many junction-tree representations (referred to as a junction tree for the specific graph) whereas for any specific junction tree, the underlying graph is uniquely determined. For clarity, from now on we follow Green and Thomas (2013) and reserve the terms vertices and edges for the elements of $G$. Vertices and edges of junctions trees will be referred to as *nodes* and *links*, respectively. Each link $(a, b)$ in a junction tree is associated with the intersection $a \cap b$, which is referred to as a *separator* and denoted by $s_{a,b}$. Note that, the empty set is also a valid separator and could separate any pair of cliques that belong to distinct connected components. The set of distinct separators in a junction tree with graph $G$ is denoted by $S(G)$. Since all junction-tree representations of a specific decomposable graph have the same set of separators, we may talk about the separators of a decomposable graph. In the following we consider a fixed sequence $(y_i)_{i=1}^{p}$, of vertices and denote by $\mathsf{G}_p$ the space of decomposable graphs with vertex set $\{y_i\}_{i=1}^{p}$. The space of junction-tree representations for graphs in $\mathsf{G}_p$ is analogously denoted by $\mathsf{T}_p$. The graph corresponding to a junction tree $T$ is denoted by $g(T)$. We let $T_s$ denote the subtree induced by the nodes of a junction tree $T$ containing the separator $s$ and let $\mathsf{F}_s(T)$ denote the forest obtained by deleting, in $T$, the links associated with $s$.

## 3 Expanding and collapsing junction trees

At the highest level, the JTE can be described in a few main steps illustrated in Fig. 3. In the first step, the algorithm starts by drawing, at random, a subtree $T'$ of the given tree $T$ (see Step 1 in Fig. 3). In the second step, a new vertex $y$ is connected to a random subset of each of the cliques in $T'$ to form a new subtree $T''$, which is isomorphic to $T'$. The edges in $T'$ are then removed and each of the nodes in $T''$ are connected to the nodes in $T'$ to which they stem from, while maintaining the junction tree property (see Step 2-4 in Fig. 3). On the other hand, the JTC starts by selecting the unique subtree $T''$ induced by a given vertex $y'$ (see Step 4 in Fig. 3).

The second step amounts to drawing, for each clique $c'$ in $T''$, a neighboring clique $c$ not containing $y'$, for which $c'$ is substituted while maintaining the junction tree property (see Step 3-1 in Fig. 3). The two algorithms are complementary in the sense that the output obtained by subjecting a given tree $T$ to either the JTE followed by the JTC, or, vice versa, the JTC followed by the JTE, coincides with $T$ with positive probability.

### 3.1 Sampling subtrees

Before presenting our main algorithm for expanding junction trees, we present one of its crucial subroutines: an algorithm for random sampling of subtrees of a given, arbitrary tree. It takes two tuning parameters, $(\alpha, \beta) \in (0, 1)^2$, which together control the number of vertices in the subtree. The algorithm either, with probability $1 - \beta$, returns the empty tree or a breadth first tree traversal is performed, where new nodes are visited with probability $\alpha$. Thus, the parameter $\alpha$ controls the number of vertices in the subgraph given that it is nonempty. We call this algorithm the *stochastic breath-first tree traversal* and provide an outline below. Full details are given in Algorithm 3 in Appendix A.

**Stochastic breadth-first tree traversal** *Let $T = (V, E)$ be a tree.*

Step 1. *Perform a Bernoulli trial that with probability $\beta$ determines if the subtree $T'$ will be nonempty.*

*If the empty tree was sampled, return it. Otherwise, proceed according to the following steps.*

Step 2. *Sample a node uniformly at random from $V$ and add it to a list $a$.*
Step 3. *Remove the first item, say $y$, from $a$ and add it to the set $V'$.*
Step 4. *Add independently each of the non-visited neighbors of $y$ to the end of $a$ with probability $\alpha$.*
Step 5. *If $a$ is not empty, go to Step 2.*
Step 6. *Return the induced subtree $T' = T[V']$.*

The probability of extracting the induced subtree $T'$ from $T$ by following the above steps is given by

$$\mathbf{S}(T, T') = \begin{cases} 1 - \beta, & \text{if } T' = (\emptyset, \emptyset) \\ \beta |V'| \alpha^{|V'|-1} (1-\alpha)^w / |V|, & \text{otherwise,} \end{cases}$$

where $w = w(T, T')$ is the number of components in the forest $T[V \setminus V']$. The factor $|V'|$ stems from the fact that any vertex in $V'$ is a valid starting vertex in the breadth-first traversal-like procedure and the probability of extracting a certain subtree is equal for each choice.

### 3.2 Expanding junction trees

In this section we present the main contribution of this paper, namely an algorithm for expanding randomly a given junction tree $T \in \mathsf{T}_m$, $m \in \mathbb{N}$, into a new junction tree $T_+ \in \mathsf{T}_{m+1}$ such that $g(T)$ is the induced subgraph of $g(T_+)$. This operation defines a Markov transition kernel $\mathbf{K}_m : \mathsf{T}_m \times \wp(\mathsf{T}_{m+1}) \to [0, 1]$, whose expression is derived at the end of this section. The full procedure, which in the following will be referred to as the *junction-tree expander*, is given below. Further details of these steps are provided in Algorithm 4 in Appendix A.

**Junction tree expander** *Let $T$ be a junction tree in $\mathsf{T}_m$.*

Step 1. *Sample a random subtree $T' = (V', E')$ of $T$.*

*If $T'$ is empty, proceed as follows:*

Step 2. *Create a new node containing merely the vertex $y_{m+1}$ and connect it to an arbitrary node in $T$.*
Step 3. *Cut the new tree at the empty separator to obtain a forest.*
Step 4. *Randomly reconnect the forest into a tree (see Appendix B).*

*If $T'$ is non-empty, enumerate the nodes in $V'$ as $(c_j)_{j=1}^{|V'|}$, and let, for each $j$, $z_j$ be defined as the union of the separators associated with $c_j$ in $T'$. Proceed as follows:*

Step 2*. *For each node $c_j$, draw uniformly at random a (possibly empty) subset $q_j$ of $c_j \setminus z_j$ to create a new unique node $d_j^+$, consisting of $d_j := z_j \cup q_j$ and the vertex $y_{m+1}$. Note that for $d_j^+$ to be unique, $q_j$ has to be non-empty if any separator associated with $c_j$ in $T'$ equals $z_j$. If $c_j$ was engulfed in $d_j^+$ (i.e. $d_j = c_j$), simply delete $c_j$.*
Step 3*. *To the nodes in $(d_j^+)_{j=1}^{|V'|}$, assign links which replicate the structure of $T'$. Then remove the links in $T'$ and connect by a link each $c_j$ to its corresponding new node $d_j^+$.*
Step 4*. *For each node $c_j$, the neighbors whose links can be moved to $d_j^+$ while maintaining an equivalent junction tree, are distributed uniformly between $c_j$ and $d_j^+$. The set of neighbors of $d_j^+$ is denoted by $n_j$.*

When using the subtree sampler provided in Algorithm 3 at Step 1, the parameters $\alpha$ and $\beta$ have clear impacts on the sparsity of the outcome $T_+$ of the JTE; more specifically, since each node in the selected subtree will give rise to a new node in $T_+$, $\alpha$ controls the number of nodes containing the new vertex $y_{m+1}$. The parameter $\beta$ is simply interpreted as

the probability of $y_{m+1}$ being connected to some vertex in $g(T)$.

***Example 1*** We illustrate two possible scenarios for how the junction tree in Fig. 1 with underlying vertex set $V = \{1, \ldots, 9\}$ could be expanded by the vertex 10. Figure 2 shows the possible scenario where the subtree picked at Step 1 is empty. Figure 3 demonstrates the possible scenario where the subtree sampled at Step 1 contains the nodes $c_1 = \{3, 4\}$, $c_2 = \{1, 4, 5\}$, and $c_3 = \{2, 5, 6\}$, colored in blue. The new nodes, colored in red, are $d_1^+ = \{3, 4, 10\}$, $d_2^+ = \{4, 5, 10\}$, and $d_3^+ = \{5, 6, 10\}$, built from the sets $z_1 = \{4\}$, $z_2 = \{4, 5\}$, $z_3 = \{5\}$ and $q_1 = \{3\}$, $q_2 = \emptyset$, $q_3 = \{6\}$. The sets of moved neighbors are $n_1 = \emptyset$, $n_2 = \emptyset$ and $n_3 = \{\{5, 6, 9\}\}$. The resulting underlying graphs for these two examples are shown in Fig. 4.

Note that in this example, $c_3$ is a leaf node in the resulting tree, making it look like decoration in a Christmas tree.

***Example 2*** Figure 5 should be read in chunks of two rows (except for the first row) and shows the junction trees, the corresponding decomposable graphs and the subgraphs generated by the JTE for $m \in \{1, \ldots, 5\}$. The left column shows the expansion of the junction trees and the right column shows the underlying decomposable graphs. Subtrees are colored in blue and the new nodes are colored in red. Unaffected nodes are black. Vertices in the underlying graphs are colored analogously. For example, the subtree $T_2'$ selected in the generation of $T_3$ on Row 5 is found on Row 4. The underlying nodes in $T_2'$ for creating $T_3$ are also found on Row 4, and so on. Note that the subtree $T_2'$ used in the creation of $T_3$, is the empty tree, thus $T_2'$ is black. The tuning parameters of the junction tree expander are set to $\alpha = 0.3$ and $\beta = 0.9$.

The main reason for operating on junction trees as opposed to decomposable graphs directly is computational tractability. Next we provide explicit expressions of the transition kernel $\mathbf{K}_m$ of the JTE, for any given $m \in \mathbb{N}$.

For $T \in \mathsf{T}_m$ and $T_+$ generated by the JTE, let $\mathsf{T}(T, T_+)$ denote the set of possible subtrees bridging $T$ and $T_+$ through the first step of the JTE. This set contains, depending on $T$ and $T_+$, either one unique or two different trees, whose explicit forms are provided by Proposition 1.

**Proposition 1** *Let $m \in \mathbb{N}$, $T \in \mathsf{T}_m$, and $T_+$ be generated by the JTE. If the subtree of $T_+$ induced by the nodes containing the vertex $y_{m+1}$ has a single node $\{y_{m+1}\} \cup s$ with exactly two neighbors $c_1$ and $c_2$ such that $s_{c_1,c_2} = s$, then $\mathsf{T}(T, T_+) = \{(\{c_1\}, \emptyset), (\{c_2\}, \emptyset)\}$; otherwise, $\mathsf{T}(T, T_+) = \{(V', E')\}$ (a single tree), where $V' = \{c_j \in V : d_j^+ \in V_+\}$ and $E' = \{(c_j, c_k) \in E : (d_j^+, d_k^+) \in E'_+\}$. Here $d_j^+ := d_j \cup \{y_{m+1}\}$ and $d_k^+ := d_k \cup \{y_{m+1}\}$ denote new nodes in $T_+$ and $c_j := d_j \cup r_j$ and $c_k := d_k \cup r_k$ are the corresponding nodes in $T$. The sets $r_j$ and $r_k$ may be empty.*

From a computational point of view, Proposition 1 is crucial since it guarantees a tractable expression of $\mathbf{K}_m$. Before we state this expression we introduce some further notation. We let $\nu_{g(T)}(s)$ denote the number of possible ways that $\mathsf{F}_s(T)$, the tree obtained by cutting $T_s$ at the separator $s$, can be connected to form a tree; this number is described in more detail in Theorem 5. Now, the transition probability of the JTE takes the following form

$$\mathbf{K}_m(T, T_+) = \sum_{T' \in \mathsf{T}(T, T_+)} \mathbb{P}(T_+ \mid T', T)\mathbf{S}(T, T'), \tag{3.1}$$

where $\mathbb{P}(T_+ \mid T', T)$ is understood as the probability that the JTE generates $T_+$ with $T$ as input given that $T'$ was drawn at Step 1. We stress again that the sum in (3.1) has either one or two terms and it is thus easily computed. The conditional probability $\mathbb{P}(T_+ \mid T', T)$ takes two different forms depending on whether $T'$ is empty or not. If $T'$ is empty, since $T_+$ is randomised at $\emptyset$, all the $\nu_{g(T_+)}(\emptyset)$ obtainable equivalent junction trees have equal probability. Otherwise, in case of $T'$ non-empty, the probability of the subsets $q_j$ are calculated according to the uniform subset distributions in Step 2*. Observe that, given $T$ and $T'$, the resulting tree $T_+$ is completely determined by $\{q_j\}_{j=1}^{|V'|}$ and $\{n_j\}_{j=1}^{|V'|}$. Since the pairs $(q_j, n_j)_{j=1}^{|V'|}$ are drawn conditionally independently given $T'$ and $T$ we obtain

$$\mathbb{P}(T_+ \mid T', T) = \begin{cases} 1/\nu_{g(T_+)}(\emptyset) & \text{if } T' = (\emptyset, \emptyset), \\ \prod_{j=1}^{|V'|} \mathbb{P}(q_j \mid T', T)\mathbb{P}(n_j \mid q_j, T', T) & \text{otherwise.} \end{cases} \tag{3.2}$$

We examine the probabilities in (3.2) in the case where $T'$ is nonempty. Since for each $j$, the existence of a node $c \in \mathfrak{N}_{T'}(c_j)$ such that $z_j = s_{c,c_j}$ forces $q_j$ to be nonempty, it holds that

$$\mathbb{P}(q_j \mid T', T) = \begin{cases} 1/(2^{|c_j \setminus z_j|} - 1) & \text{if } z_j = s_{c,c_j} \text{ for some } c \in \mathfrak{N}_{T'}(c_j), \\ 1/2^{|c_j \setminus z_j|} & \text{otherwise.} \end{cases}$$

Conditionally upon $T'$, $T$, and $q_j$, the probability of each neighbor set $n_j$ at Step 4* follows straightforwardly; indeed, the distribution of $n_j$ takes two different forms depending on whether $c_j$ was engulfed into $d_j^+$ (i.e. $d_j = c_j$) or not. If so, all of the neighbors of $c_j$ are moved to $d_j^+$ with probability 1. Otherwise, it has equal probability over all subsets of $U_j := \{c \in \mathfrak{N}_T(c_j) \setminus V' : s_{c,c_j} \subseteq d_j\}$ giving

$$\mathbb{P}(n_j \mid q_j, T', T) = \begin{cases} 1 & \text{if } d_j = c_j, \\ 1/2^{|U_j|} & \text{otherwise.} \end{cases}$$

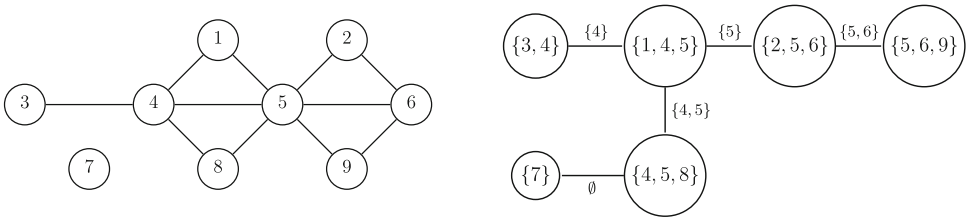
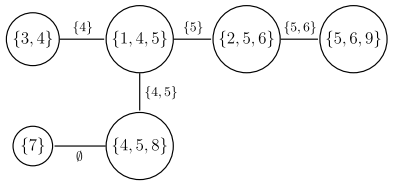
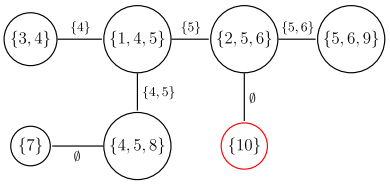


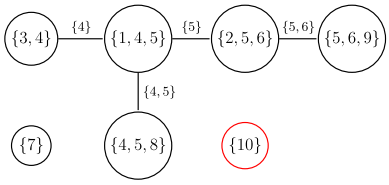

**Fig. 1** A decomposable graph (left panel) and one of its junction tree representations (right panel)



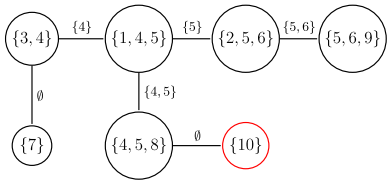

*Step 1*: Simulate subtree (empty in this case).




*Step 2*: Connect the new node, {10} to an arbitrary existing node.

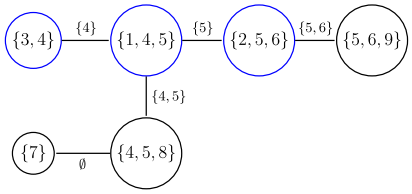



*Step 3*: Create forest.

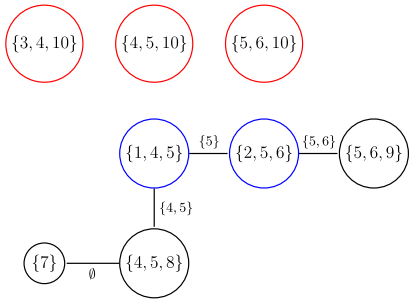



*Step 4*: Reconnect the forest into a tree.

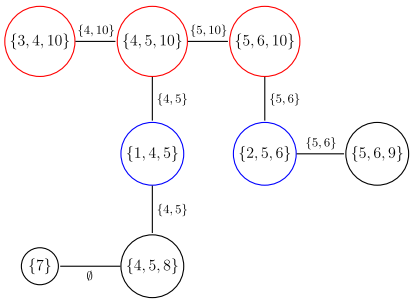
**Fig. 2** A possible expansion of the junction tree in Fig. 1, where the empty subtree is drawn at Step 1



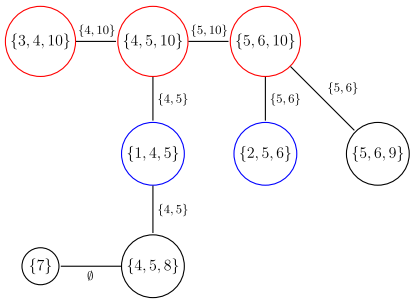

*Step 1*: Simulate subtree.




*Step 2**: Create new nodes.

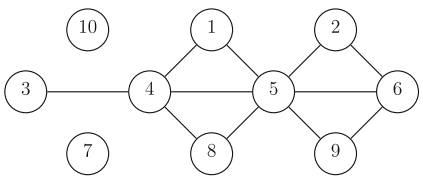



*Step 3**: Replicate structure.




*Step 4**: Relocate neighbors.

**Fig. 3** A possible outcome of the JTE where a non-empty subtree was drawn in the expansion of the junction tree in Fig. 1



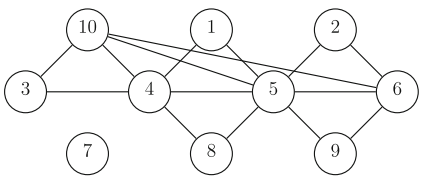




**Fig. 4** Two decomposable graphs resulting from expanding the graph in Fig. 1 by the vertex 10

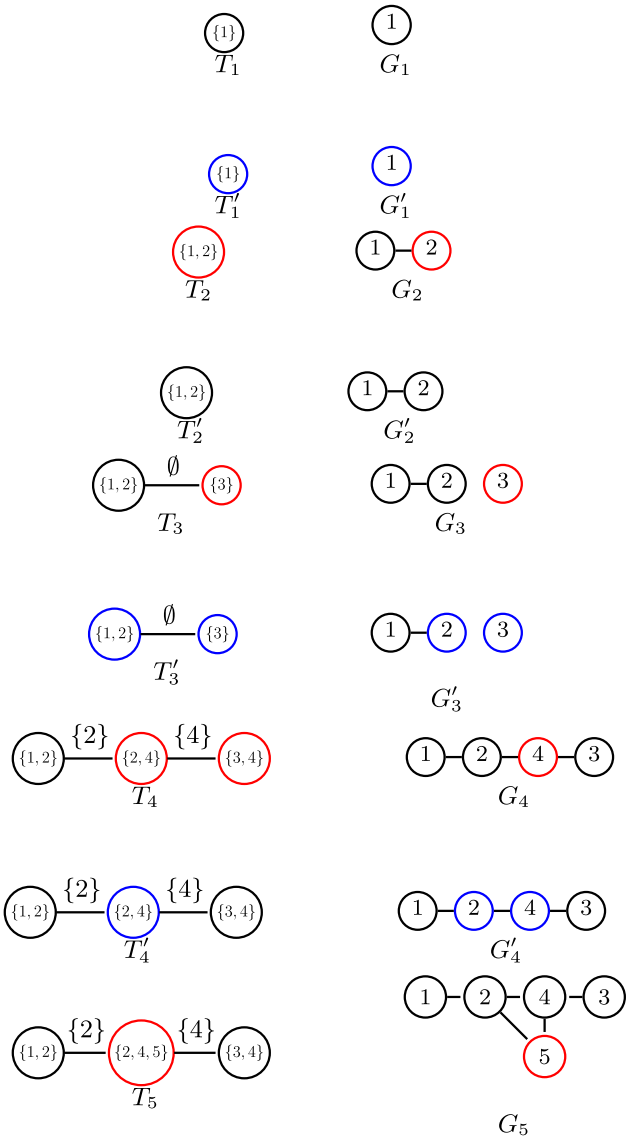



**Fig. 5** Example of a recursive application of the JTE with parameters $\alpha = 0.3$ and $\beta = 0.9$

Observe that the simplicity of (3.1) is appealing from a computational point of view. In particular, as shown in Sect. 7, when $\mathbf{K}_m$ is used as a proposal kernel in an SMC algorithm, fast computation of the transition probability is crucial, especially as the graph space increases.

An important property of the JTE is that for any $m \in \mathbb{N}$ and $T \in \mathsf{T}_m$, a tree $T_+$ generated by the JTE is also a junction tree. In addition, $g(T)$ is an induced subgraph of $g(T_+)$, having one additional vertex.

**Theorem 1** *For any $m \in \mathbb{N}$ and $T \in \mathsf{T}_m$ it holds that*

(i) $\mathrm{Supp}(\mathbf{K}_m(T, \cdot)) \subseteq \mathsf{T}_{m+1}$,
(ii) $g(T_+)[\{y_\ell\}_{\ell=1}^m] = g(T)$ *for all* $T_+ \in \mathrm{Supp}(\mathbf{K}_m(T, \cdot))$.

The following theorem states that for any $m \in \mathbb{N}$, all junction trees in $\mathsf{T}_m$ can be generated with positive probability using recursive application of the JTE. More specifically, we may define the marginal probability $\mathbf{K}^m(T_m) := \sum_{T \in \mathsf{T}_{m-1}} \mathbf{K}^{m-1}(T)\mathbf{K}_{m-1}(T, T_m)$, for any $T_m \in \mathsf{T}_m$, where $\mathbf{K}^1((\{\{y_1\}\}, \emptyset)) = 1$ and state the following theorem.

**Theorem 2** *For any ordering of vertices $(y_\ell)_{\ell=1}^m$, $m \in \mathbb{N}$, it holds that*

$$\mathrm{Supp}(\mathbf{K}^m) = \mathsf{T}_m.$$

For comparison, the algorithm for sequential a sampling of junction trees presented by Markenzon et al. (2008) corresponds to recursive application of a special case of the JTE, where $\alpha = 0$, $\beta = 1$, and where Step 4 is omitted. Note that Theorem 2 does not hold under such assumptions since the algorithm is forced to operate on a restricted space of junction trees for connected decomposable graphs. Markenzon et al. (2008) also proposes a final step that merges neighboring cliques an unspecified number of times in order to increase the number of edges in the underlying graphs. While this step has the intended effect on the graphs, the space is still restricted and calculating the transition probabilities becomes intractable in general.

## 4 Collapsing junction trees

In this section, we present the *junction-tree collapser*, a reversed version of the JTE, introduced in the previous section. The idea is to collapse a junction tree $T_+ \in \mathsf{T}_{m+1}$ into a new tree $T \in \mathsf{T}_m$ by removing $y_{m+1}$ from the underlying graph in such a way that $T \in \mathrm{Supp}(\mathbf{K}_m(\cdot, T_+))$. As will be proved in this section, this procedure defines a Markov kernel $\mathbf{R}_m : \mathsf{T}_{m+1} \times \wp(\mathsf{T}_m) \to [0, 1]$.

Next follows a description of the different suboperations in the sampling procedure for $\mathbf{R}_m$. The details of the steps are given in Algorithm 5 in Appendix A.

**Junction tree collapser** *Let $T_+$ be a junction tree in $\mathsf{T}_{m+1}$. Similarly to the JTE, the JTC takes two different forms depending on whether $\{y_{m+1}\}$ is present as a node in $T_+$ or not.*

*If $\{y_{m+1}\}$ is a node in $T_+$ proceed as follows:*

Step 1. *Remove $\{y_{m+1}\}$ and it incident links to obtain a forest, possibly containing only one tree.*
Step 2. *Randomly connect the forest into a tree.*

*If $\{y_{m+1}\}$ is not a node in $T_+$ proceed as follows:*

Step 1*. *Let $T_+' = (V_+', E_+')$ be the subtree of $T_+$ induced by the nodes containing the vertex $y_{m+1}$ and enumerate the nodes in $V_+'$ by $(d_j^+)_{j=1}^{|V_+'|}$.*

Step 2*. *For all* $j \in \{1, \ldots, |V'_+|\}$, *draw at random* $c_j$ *from* $M_j$, *the set of neighbors of* $d_j^+$ *in* $T_+$ *having the associated separator* $d_j^+ \setminus \{y_{m+1}\}$. *If no such neighbor exists, let* $c_j = d_j^+ \setminus \{y_{m+1}\}$.

Step 3*. *Replace each node* $d_j^+$ *by the corresponding node* $c_j$ *in the sense that* $c_j$ *is assigned all former neighbors of* $d_j^+$.

The next example illustrates a reversed version of Example 1.

***Example 3*** Consider collapsing the junction tree in the bottom right panel of Fig. 3 by the vertex 10. The induced subgraph $T_+$, having the nodes $d_1^+ = \{3, 4, 10\}$, $d_2^+ = \{4, 5, 10\}$, and $d_3^+ = \{5, 6, 10\}$ is colored in red in the same subfigure. Further we see that $M_1 = \emptyset$ implies that $c_1 = \{3, 4\}$ and $M_2 = \{\{1, 4, 5\}\}$ implies $c_2 = \{1, 4, 5\}$. By drawing $c_3 = \{2, 5, 6\}$ from $M_3 = \{\{2, 5, 6\}, \{5, 6, 9\}\}$, the junction tree in the top left panel of Fig. 3 is obtained.

The induced transition probability of collapsing $T_+ \in \mathsf{T}_{m+1}$ into a tree $T \in \text{Supp}(\mathbf{R}_m(T_+, \cdot))$ has the form

$$\mathbf{R}_m(T_+, T) = \begin{cases} 1/\nu_{g(T)}(\emptyset) & \text{if } \{y_{m+1}\} \in V_+, \\ 1/\prod_{j=1}^{|V'_+|} \max(1, |M_j|) & \text{otherwise,} \end{cases}$$

where, as before, $V'_+$ is the set of nodes in $T_+$ containing $y_{m+1}$. The max operation is needed in order to make the expression well defined even when $M_j$ is empty.

The JTC is a reversed version of the JTE in the sense that for all $m \in \mathbb{N}$, a junction tree $T \in \mathsf{T}_m$, generated by the JTC from a junction tree $T_+ \in \mathsf{T}_{m+1}$, can be used as input to the JTE to generate $T_+$. This property is formulated in the next theorem.

**Theorem 3** *For all* $m \in \mathbb{N}$ *and* $T_+ \in \mathsf{T}_{m+1}$,

(i) $\text{Supp}(\mathbf{R}_m(T_+, \cdot)) \subseteq \mathsf{T}_m$,
(ii) $\text{Supp}(\mathbf{R}_m(T_+, \cdot)) \subseteq \text{Supp}(\mathbf{K}_m(\cdot, T_+))$,
(iii) $g(T) = g(T_+)[\{y_\ell\}_{\ell=1}^m]$ *for any* $T \in \text{Supp}(\mathbf{R}_m(T_+, \cdot))$.

Theorem 3 proves to be crucial in the SMC context described in Sect. 6 and in particular in the refreshment step of the particle Gibbs sampler detailed in Olsson et al. (2019).

## 5 Counting the number of junction trees for an expanded decomposable graph

Thomas and Green (2009b) provide an expression for counting the number of equivalent junction trees of a given decomposable graph. In this section we derive a factorisation of the same expression which shows to alleviate the computational burden when calculated for expanded graphs. For sake of completeness, we restate three theorems from (Thomas and Green 2009b). The first counts the number of ways a forest can be reconnected into a tree and was first established in Moon (1967).

**Theorem 4** *(Moon (1967)) The number of distinct ways that a forest of order* $m$ *comprising* $q$ *subtrees of orders* $r_1, \ldots, r_q$ *can be connected into a single tree by adding* $q - 1$ *edges is*

$$m^{q-2} \prod_{i=1}^{q} r_i.$$

For a given junction tree $T$, let $t_s$ denote the order of the subtree $T_s$ induced by the separator $s$. Now, let $m_s$ be the number of links associated with $s$ and let $f_1, \ldots, f_{m_s+1}$ be the orders of the tree components in $\mathsf{F}_s(T)$. Then, by Theorem 4 the following is obtained.

**Theorem 5** *(Thomas and Green (2009b)) The number of ways that the components of* $\mathsf{F}_s(T)$, *where* $s$ *is a separator in a graph* $G$ *with junction tree* $T$, *can be connected into a single tree by adding the appropriate number of links is given by*

$$\nu_G(s) = t_s^{m_s - 1} \prod_{j=1}^{m_s+1} f_j.$$

**Theorem 6** *(Thomas and Green (2009b)) The number of junction trees for a decomposable graph* $G$ *is given by*

$$\mu(G) = \prod_{s \in S(G)} \nu_G(s).$$

In the sequential sampling context considered in this paper it is useful to exploit that any decomposable graph $G_+ \in \mathsf{G}_{m+1}$ can be regarded as an expansion of another decomposable graph $G \in \mathsf{G}_m$, in the sense that $G_+$ is obtained by expanding $G$ with the vertex $y_{m+1}$. This follows for example by induction using (Lauritzen 1996, Corollary 2.8).

The key insight when calculating $\mu(G_+)$ is that when a vertex is added to $G$, not all separators will necessarily be affected. This implies that $\nu_G(s) = \nu_{G_+}(s)$ for some separators.

**Theorem 7** *Let* $G_+ \in \mathsf{G}_{m+1}$ *be an expansion of some graph* $G \in \mathsf{G}_m$ *by the extra vertex* $y_{m+1}$. *Let* $S^\star \subseteq S(G_+)$ *be the set of unique separators created (note that* $S^\star \cap S(G)$ *might be non-empty) by the expansion. Then*

$$\mu(G_+) = \frac{\prod_{s \in A(G_+)} \nu_{G_+}(s)}{\prod_{s \in A(G)} \nu_G(s)} \mu(G), \tag{5.1}$$

*where* $A(G) := \{s \in S(G) : \exists s' \in S^\star$ *such that* $s \subseteq s'\}$ *is the set of separators in* $G$ *contained in some separator in* $S^\star$.

The potential computational gain obtained by using the factorisation in Theorem 7 is illustrated by the following example.

***Example 4*** Let $G_+ \in \mathsf{G}_{m+1}$ be an expansion of a graph $G \in \mathsf{G}_m$ in the sense that $y_{m+1}$ is connected to every vertex in one of the cliques in $G$. Then, since the set of separators is the same in the two graphs, it holds that $\mu(G_+) = \mu(G)$.

## 6 Applications to sequential Monte Carlo sampling

Sequential Monte Carlo (SMC) methods (Chopin and Papaspiliopoulos 2020) are a class of simulation-based algorithms that offers a principled way of sampling online from very general sequences of distributions, known up to normalising constants only, by propagating recursively a population of random draws, so-called particles, with associated importance weights. The particles evolve randomly and iteratively through selection and mutation. In the selection step, the particles are duplicated or eliminated depending on their importance, while the mutation operation disseminates randomly the particles in the state space and assigns new importance weights to the same for further selection at the next iteration. SMC methods have been particularly successful when it comes to online approximation of state posteriors in general state-space hidden Markov models (Arulampalam et al. 2002).

In this section we demonstrate how the JTE and the JTC can be cast into the framework of SMC methods—or, more precisely, the SMC samplers proposed in Del Moral et al. (2006)—in order to sample from a sequence $(\eta_m)_{m\in\mathbb{N}}$ of probability distributions, where each $\eta_m$ is a distribution on $\mathsf{T}_m$. For every $m$ we assume that $\eta_m$ is known only up to a normalising constant, i.e., $\eta_m \propto \gamma_m$, where $\gamma_m$ is a tractable, unnormalised function. Following (Del Moral et al. 2006), we introduce path spaces $\mathsf{T}_{1:m} := \mathsf{T}_1 \times \cdots \times \mathsf{T}_m$ and let

$$\bar{\eta}_m(T_{1:m}) := \prod_{\ell=1}^{m-1} \frac{\eta_{\ell+1}(T_{\ell+1})}{\eta_\ell(T_\ell)} \mathbf{R}_\ell(T_{\ell+1}, T_\ell), \quad T_{1:m} \in \mathsf{T}_{1:m}, \tag{6.1}$$

be extended target distributions. Importantly, each target $\eta_m$ is the marginal of $\bar{\eta}_m$ with respect to the $m$th component. In many applications, the aim is to sample from a given distribution $\pi$ on some junction-tree space $\mathsf{T}_n$ induced by $n$ vertices, and in this case one may let $\eta_n = \pi$ and $(\eta_\ell)_{m=1}^{n-1}$ be the marginals of $\pi$ (if these are known up to normalising constants), serving to guide the distribution flow towards the target $\pi$.

Now, introduce, for all $m$, proposal distributions

$$\bar{\rho}_m(T_{1:m}) := \prod_{\ell=1}^{m-1} \mathbf{K}_\ell(T_\ell, T_{\ell+1}), \quad T_{1:m} \in \mathsf{T}_{1:m}. \tag{6.2}$$

Since Theorem 3 implies that $\mathrm{Supp}(\mathbf{R}_\ell(\cdot, T_\ell)) \subseteq \mathrm{Supp}(\mathbf{K}_\ell(T_\ell, \cdot))$ for all $\ell \in \{1, \ldots, m-1\}$, it is readily checked that $\mathrm{Supp}(\bar{\eta}_m) \subseteq \mathrm{Supp}(\bar{\rho}_m)$. This property, along with Theorems 1 and 2, allows the extended target distributions (6.1) to be sampled by means of an importance-sampling procedure, where independent tree paths $\tau_{1:m}^i = (\tau_1^i, \ldots, \tau_m^i)$ generated sequentially using the JTE, are assigned importance weights

$$\omega_m^i := \prod_{\ell=1}^{m-1} \frac{\gamma_{\ell+1}(\tau_{\ell+1}^i)\mathbf{R}_\ell(\tau_{\ell+1}^i, \tau_\ell^i)}{\gamma_\ell(\tau_\ell^i)\mathbf{K}_\ell(\tau_\ell^i, \tau_{\ell+1}^i)} \propto \frac{\bar{\eta}_m(\tau_{1:m}^i)}{\bar{\rho}_m(\tau_{1:m}^i)}. \tag{6.3}$$

Here $N$ is the Monte Carlo sample size. Thanks to the Markovian structure of the proposal (6.2) and the multiplicative structure of the weights (6.3), this procedure can be implemented sequentially by applying recursively the update described in Algorithm 1. This yields a sequence $(\tau_m^i, \omega_m^i)_{i=1}^N$, $m \in \mathbb{N}$, of weighted samples, where, since $\eta_m$ is the marginal of $\bar{\eta}_m$ with respect to the last component, $\sum_{i=1}^N \omega_m^i h(\tau_m^i)/\Omega_m^N$, with $\Omega_m^N := \sum_{i=1}^N \omega_m^i$, is a strongly consistent self-normalised estimator of the expectation $\eta_m(h) := \sum_{T\in\mathsf{T}_m} h(T)\eta_m(T)$ of any real-valued test function $h$ under $\eta_m$. In the SMC literature, the draws $(\tau_m^i)_{i=1}^N$ are typically referred to as *particles*.

**Input**: $(\tau_m^i, \omega_m^i)_{i=1}^N$
**Output**: $(\tau_{m+1}^i, \omega_{m+1}^i)_{i=1}^N$
1 **for** $i \leftarrow 1, \ldots, N$ **do**
2  draw $\tau_{m+1}^i \sim \mathbf{K}_m(\tau_m^i, \cdot)$;
3  set $\omega_{m+1}^i \leftarrow \dfrac{\gamma_{m+1}(\tau_{m+1}^i)\mathbf{R}_m(\tau_{m+1}^i, \tau_m^i)}{\gamma_m(\tau_m^i)\mathbf{K}_m(\tau_m^i, \tau_{m+1}^i)}\, \omega_m^i$;
4 **return** $(\tau_{m+1}^i, \omega_{m+1}^i)_{i=1}^N$

**Algorithm 1:** Sequential importance sampling.

Even though this *sequential importance sampling* procedure, which is described in Algorithm 1, appears appealing at a first sight, the multiplicative weight updating formula (6.3) (Line 3 in Algorithm 1) is problematic in the sense that it will, inevitably, lead to severe weight skewness and, consequently, high Monte Carlo variance. In fact, it can be shown that updating the weights in this naive manner leads to a Monte Carlo variance that increases geometrically fast with $m$; see e.g. (Cappé et al. 2005, Chapter 7.3) for a discussion. Needless to say, this is impractical for most applications

In order to cope with the weight-degeneracy problem, Gordon et al. (1993) proposed furnishing the previous sequential importance sampling algorithm with a *selection step*, in which the particles are resampled, with replacement, in proportion to their importance weights. Upon selection, all particles are assigned the unit weight, and the particles and importance weights are then updated as in Algorithm 1. Such selection is a key ingredient in SMC methods, and it can be shown mathematically that the resulting *sequential importance sampling with resampling* algorithm, which is given in Algorithm 2, is indeed numerically stable Chopin and Papaspiliopoulos (2020), Del Moral (2004).

**Input**: $(\tau_m^i, \omega_m^i)_{i=1}^N$
**Output**: $(\tau_{m+1}^i, \omega_{m+1}^i)_{i=1}^N$
1 resample $(\tilde{\tau}_m^i)_{i=1}^N$ among $(\tau_m^i)_{i=1}^N$ in proportion to $(\omega_m^i)_{i=1}^N$;
2 **for** $i \leftarrow 1, \ldots, N$ **do**
3 draw $\tau_{m+1}^i \sim \mathbf{K}_m(\tilde{\tau}_m^i, \cdot)$;
4 set $\omega_{m+1}^i \leftarrow \dfrac{\gamma_{m+1}(\tau_{m+1}^i)\mathbf{R}_m(\tau_{m+1}^i, \tilde{\tau}_m^i)}{\gamma_m(\tilde{\tau}_m^i)\mathbf{K}_m(\tilde{\tau}_m^i, \tau_{m+1}^i)}$;
5 **return** $(\tau_{m+1}^i, \omega_{m+1}^i)_{i=1}^N$

**Algorithm 2:** Sequential importance sampling with resampling.

In standard self-normalised importance sampling, the average weight provides an unbiased estimator of the normalising constant of the target. However, when the particles are resampled systematically, as in Algorithm 2, this simple estimator is no longer valid. Instead, it is possible to show that for every $m$, the estimator

$$\gamma_m^N(h) := \frac{1}{N^m}\left(\prod_{\ell=1}^{m-1}\Omega_\ell^N\right)\sum_{i=1}^N \omega_m^i h(\tau_m^i),$$

with $\Omega_\ell^N := \sum_{i=1}^N \omega_\ell^i$, is an unbiased estimator of $\gamma_m(h)$ for any real-valued test function $h$. In particular,

$$\gamma_m^N(\mathbb{1}_{\mathsf{T}_m}) = \frac{1}{N^m}\prod_{\ell=1}^m \Omega_\ell^N \tag{6.4}$$

provides an unbiased estimator of the normalising constant $\gamma_m(\mathbb{1}_{\mathsf{T}_m})$ of $\eta_m$. This estimator will be illustrated in the next section.

## 7 Numerical study

We demonstrate two applications of Algorithm 2 for estimating the cardinalities $|\mathsf{G}_m|$ and $|\mathsf{T}_m|$ of the spaces of decomposable graphs and junction trees, respectively.

### 7.1 Estimating $|\mathsf{G}_m|$

Wormald (1985) provides an exact expression for $|\mathsf{G}_m|$ and evaluates the same for $m \leq 13$. In the same reference, the author also establishes the asymptotic expression $|\mathsf{G}_m| \sim \sum_{\ell=1}^m \binom{m}{\ell} 2^{\ell(m-\ell)}$. Another exact algorithm that calculates $|\mathsf{G}_m|$ for $m \leq 10$ is proposed in Kawahara et al. (2018).

In this study we will use Algorithm 2 for estimating $|\mathsf{G}_m|$, $m \in \mathbb{N}$, on the basis of the target probability distributions

$$\eta_m(T_m) \propto \gamma_m(T_m) = \frac{1}{\mu(g(T_m))}\mathbb{1}_{\mathsf{T}_m}(T_m).$$

Note that the normalising constant $\gamma_m(\mathbb{1}_{\mathsf{T}_m})$ of $\eta_m$ equals $|\mathsf{G}_m|$.; indeed,

$$\begin{aligned}\gamma_m(\mathbb{1}_{\mathsf{T}_m}) &= \sum_{G_m\in\mathsf{G}_m}\sum_{T_m:g(T_m)=G_m}\frac{1}{\mu(g(T_m))}\mathbb{1}_{\mathsf{T}_m}(T_m)\\ &= \sum_{G_m\in\mathsf{G}_m}\frac{1}{\mu(G_m)}\sum_{T_m:g(T_m)=G_m}\mathbb{1}_{\mathsf{T}_m}(T_m)\\ &= |\mathsf{G}_m|.\end{aligned}$$

With this formulation, unbiased estimates of $|\mathsf{G}_m|, m \in \mathbb{N}$, can be obtained directly using (6.4). Note that in this setting Line 4 of Algorithm 2 reduces to

$$\frac{\gamma_{m+1}(T_{m+1})}{\gamma_m(T_m)} = \frac{\mu(g(T_m))}{\mu(g(T_{m+1}))}, \tag{7.1}$$

where $T_m \in \mathsf{T}_m$, $T_{m+1} \in \mathsf{T}_{m+1}$, for which, as demonstrated by Example 4, the computational burden can be substantially reduced using the factorisation (5.1) since $T_{m+1} \in \mathrm{Supp}(\mathbf{K}_m(T_m, \cdot))$.

Table 1 shows means and standard errors based on 10 estimates $\widehat{|\mathsf{G}_m|}$ of $|\mathsf{G}_m|$ for $m \in \mathbb{N}_{15}$. The upper panel of the table shows $\widehat{|\mathsf{G}_m|}$ while the lower panel shows $\widehat{|\mathsf{G}_m|}/2^{m(m-1)/2}$, i.e. estimates of the fraction of undirected graphs that are decomposable. For $m \leq 13$ the exact enumerations are given in the second column. We ran the SMC sampler with tuning parameters $\alpha = 0.5$, $\beta = 0.5$ and the number of particles was set to $N = 10000$. Figure 6 displays the asymptotic behavior of $|\mathsf{G}_m|$ and $\widehat{|\mathsf{G}_m|}$ for $m \leq 50$, along with the exact values for $m \leq 13$, justifying a concordance with the exact results. Each of the 10 estimates took about 10 minutes to calculate.

Finally, we also explored other parameterisations for $\alpha$ and $\beta$ and found that, in this case, the estimates seem to be less accurate in terms of standard error when using high values of $\alpha$ about 0.9 and low values of $\beta$ about 0.3. However, for $\alpha$ and $\beta$ about 0.3 and 0.9, respectively, the performance of the estimator was similar to that for the parameterisaion $\alpha = \beta = 0.5$ considered above.

**Table 1** Sequential Monte Carlo estimation of the number of decomposable graphs and the fraction of graphs which are decomposable

| $p$ | $\lvert\mathsf{G}_p\rvert$ | $\widehat{\lvert\mathsf{G}_p\rvert}$ | SE | |
|---|---|---|---|---|
| 4 | 61 | 60.4 | 0.40 | |
| 5 | 822 | 815 | 16.2 | |
| 6 | 1.82 | 1.78 | 0.37 | $\times 10^4$ |
| 7 | 6.18 | 5.92 | 0.15 | $\times 10^5$ |
| 8 | 3.09 | 3.00 | 0.13 | $\times 10^7$ |
| 9 | 2.19 | 2.14 | 0.09 | $\times 10^9$ |
| 10 | 2.15 | 2.11 | 0.10 | $\times 10^{11}$ |
| 11 | 2.88 | 2.80 | 0.18 | $\times 10^{13}$ |
| 12 | 5.17 | 5.15 | 0.51 | $\times 10^{15}$ |
| 13 | 1.23 | 1.21 | 0.17 | $\times 10^{18}$ |
| 14 | – | 3.74 | 0.62 | $\times 10^{20}$ |
| 15 | – | 1.53 | 0.30 | $\times 10^{23}$ |
| $p$ | $\lvert\mathsf{G}_p\rvert/2^{p(p-1)/2}$ | $\widehat{\lvert\mathsf{G}_p\rvert}/2^{p(p-1)/2}$ | SE | |
| 4 | 9.53 | 9.44 | 0.06 | $\times 10^{-1}$ |
| 5 | 8.03 | 7.96 | 0.16 | $\times 10^{-1}$ |
| 6 | 5.54 | 5.43 | 0.11 | $\times 10^{-1}$ |
| 7 | 2.95 | 2.82 | 0.07 | $\times 10^{-1}$ |
| 8 | 1.15 | 1.12 | 0.05 | $\times 10^{-1}$ |
| 9 | 3.19 | 3.11 | 0.13 | $\times 10^{-2}$ |
| 10 | 6.12 | 5.99 | 0.28 | $\times 10^{-3}$ |
| 11 | 7.99 | 7.78 | 0.51 | $\times 10^{-4}$ |
| 12 | 7.00 | 6.98 | 0.70 | $\times 10^{-5}$ |
| 13 | 4.09 | 3.99 | 0.55 | $\times 10^{-6}$ |
| 14 | – | 1.51 | 0.25 | $\times 10^{-7}$ |
| 15 | – | 3.77 | 0.74 | $\times 10^{-9}$ |

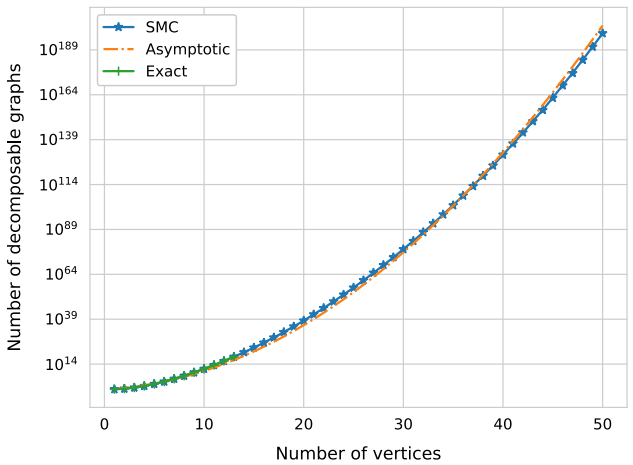



**Fig. 6** The number of decomposable graphs as a function of the number of vertices

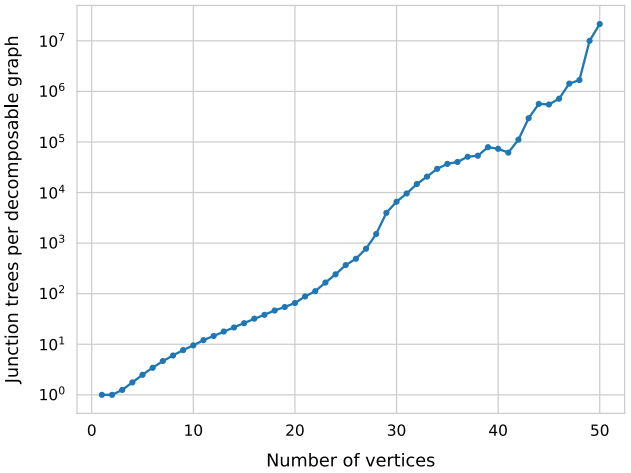



**Fig. 7** Estimates of the expected number of junction trees per decomposable graph

## 7.2 Estimating $|\mathsf{T}_m|$

As far as we know there is no method available in the literature for efficiently calculating $|\mathsf{T}_m| := \sum_{G\in\mathsf{G}_m}\mu(G)$. However, for $m \leq 5$ it is computationally tractable to first find all the 822 graphs $\mathsf{G}_m$ by Monte Carlo sampling and then evaluate $\mu$ for each of them.

As in Sect. 7.1 we find an unbiased estimator of $|\mathsf{T}_m|$ by constructing target distributions

$$\eta_m(T_m) \propto \gamma_m(T_m) = \mathbb{1}_{\mathsf{T}_m}(T_m),$$

so that the normalising constant $\gamma_m(\mathbb{1}_{\mathsf{T}_m})$ equals $|\mathsf{T}_m|$, and then use (6.4). Note that with this setting the first factor in Line 4 in Algorithm 2 simplifies as

$$\frac{\gamma_{m+1}(T_{m+1})}{\gamma_m(T_m)} = 1, \tag{7.2}$$

for all $T_m \in \mathsf{T}_m$, $T_{m+1} \in \mathrm{Supp}(\mathbf{K}_m(T_m,\cdot))$.

The third and fourth columns of the upper panel in Table 2 show estimated means and standard deviations of $\widehat{|\mathsf{T}_m|}$ for $m \leq 15$ based on 10 replicates. The true values for are shown in the first column for $m \leq 5$. The lower panel of Table 2 displays estimates of the number of junction trees per decomposable graph, $|\mathsf{T}_p|/|\mathsf{G}_p|$, for different numbers of vertices. True numbers as are shown in the first column, and estimated means and standard deviations of $\widehat{|\mathsf{T}_p|}/\widehat{|\mathsf{G}_p|}$ are shown in the third and fourth columns. Interestingly, Figure 7 indicates an exponential growth rate of the estimated junction trees per decomposable graph for $p \leq 50$. Each of the 10 estimates took about 6 minutes to compute.

**Table 2** Sequential Monte Carlo estimation of the number junction trees and the expected number of junction trees per decomposable graph

| $p$ | $\lvert\mathsf{T}_p\rvert$ | $\widehat{\lvert\mathsf{T}_p\rvert}$ | SE | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0.00 | |
| 2 | 2 | 2 | 0.00 | |
| 3 | 10 | 10 | 0.03 | |
| 4 | 108 | 106 | 1.41 | |
| 5 | 2.09 | 2.03 | 0.03 | $\times 10^3$ |
| 6 | - | 6.10 | 0.10 | $\times 10^4$ |
| 7 | - | 2.76 | 0.08 | $\times 10^6$ |
| 8 | - | 1.79 | 0.07 | $\times 10^8$ |
| 9 | - | 1.63 | 0.11 | $\times 10^{10}$ |
| 10 | - | 2.00 | 0.18 | $\times 10^{12}$ |
| 11 | - | 3.37 | 0.32 | $\times 10^{14}$ |
| 12 | - | 7.45 | 0.88 | $\times 10^{16}$ |
| 13 | - | 2.12 | 0.30 | $\times 10^{19}$ |
| 14 | - | 7.87 | 1.34 | $\times 10^{21}$ |
| 15 | - | 3.88 | 0.80 | $\times 10^{24}$ |

| $p$ | $\lvert\mathsf{T}_p\rvert/\lvert\mathsf{G}_p\rvert$ | $\widehat{\lvert\mathsf{T}_p\rvert/\lvert\mathsf{G}_p\rvert}$ | SE |
|---|---|---|---|
| 1 | 1 | 1 | 0.00 |
| 2 | 1 | 1 | 0.00 |
| 3 | 1.25 | 1.25 | 0.00 |
| 4 | 1.77 | 1.76 | 0.03 |
| 5 | 2.54 | 2.49 | 0.07 |
| 6 | – | 3.43 | 0.1 |
| 7 | – | 4.66 | 0.23 |
| 8 | – | 5.99 | 0.45 |
| 9 | – | 7.67 | 0.64 |
| 10 | – | 9.51 | 0.90 |
| 11 | – | 12.1 | 1.35 |
| 12 | – | 14.6 | 1.98 |
| 13 | – | 17.9 | 3.11 |
| 14 | – | 21.5 | 4.23 |
| 15 | – | 26.1 | 6.16 |

## 8 Discussion

In this paper we have presented the JTE and the JTC for stochastically generating and collapsing junction trees for decomposable graphs in a vertex-by-vertex fashion. The Markovian nature of these procedures enables the development of sophisticated sampling technology such as SMC and particle MCMC methods; see (Olsson et al. 2019).

Several MCMC methods for approximating distributions on the space of decomposable graphs have been proposed in the literature. Still, in most of these methods, an MCMC chain of graphs (or junction trees) is evolved by means of locally limited random perturbations, leading generally to bad mixing (Giudici and Green 1999; Green and Thomas 2013). The main benefit of casting the JTE and JTC procedures into the particle Gibbs framework is a substantial improvement of the mixing properties of the resulting MCMC chain; this improvement is possible since the JTE procedure allows the produced chain of junction trees to make long-range, global transitions across the state space.

The appealing properties of our approach do not come without a certain price. For instance, relying on the junction-tree representation when sampling from a given decomposable-graph distribution imposes an additional computational burden associated with calculating the number of possible junction-tree representations of each of the sampled graphs. In the present paper, we have been able to alleviate this burden by means of the factorisation property derived in Theorem 7, allowing for faster dynamic updates. Another factor that is challenging when using the SMC procedure in Algorithm 2 for sampling distributions over spaces of decomposable graphs with a very large number $p$ of vertices stems from the well-known *particle-path degeneracy phenomenon*; see (Jacob et al. 2015; Koskela et al. 2020). More specifically, since the graphs propagated by Algorithm 2 are resampled systematically, many of them will, eventually, as the number of SMC iterations increases, have parts of their underlying graph in common. This may lead to high variance when $p$ is large compared to the sample size $N$, and the $\mathrm{O}(N)$ bound on the resampling-induced particle-path coalescing time obtained recently in Koskela et al. (2020) suggests that $p$ and $N$ should be of at least the same order in order to keep the Monte Carlo error under control. In the particle Gibbs approach developed in Olsson et al. (2019) the particle-path degeneracy phenomenon is handled by means of an additional JTC-based backward-sampling operation.

As an alternative approach to the JTE, which incrementally constructs a junction tree by adding one vertex at a time to the underlying graph, one may suggest a method that operates directly on the space of decomposable graphs. The main difficulty arising when designing such a scheme is to express the transition probabilities in a tractable form while maintaining the ability to generate any decomposable graph with a given number of vertices, qualities possessed by the methods that we propose.

Finally, we expect that tailored data structures for the junction tree implementation which respect the sequential nature of the algorithms could greatly increase the computational speed. For instance, when propagating the particles in Algorithm 2, the junction trees are not altered but rather copied and expanded (since several trees must be able to stem from the same ancestor); thus, to use persistent data structures—which are widely used in *functional programming* to avoid the copying of data—in the SMC context of the present paper is an interesting line of research.

**Acknowledgements** We are grateful to the editor and the two reviewers for their valuable comments and helpful suggestions, which have improved the paper significantly. Tatjana Pavlenko's work has been supported by the AI4Research Grant, Uppsala University. J. Olsson gratefully acknowledges support by the Swedish Research Council, Grant 2018-05230. We are also thankful to Jim Holmström for sharing his Python knowledge with us.

**Funding** Open access funding provided by Royal Institute of Technology.



# Appendix A

## A.1 Stochastic breath-first tree traversal

Algorithm 3 provides the detailed steps in the stochastic breath-first tree traversal algorithm outlined in Sect. 3.1. The function *push_back* adds a new element to the end of a list and the function *pop_front* returns and removes the first element of a list.

**Input**: A tree $T = (V, E)$, parameters $(\alpha, \beta) \in (0, 1)^2$
**Output**: $T'$ subtree of $T$
draw $u \sim \mathsf{Bernoulli}(\beta)$;
**if** $u = 0$ **then**
  **return** $(\emptyset, \emptyset)$
**else**
  draw $y \sim \mathsf{Unif}\langle V \rangle$;
  $q := []$; // empty list of vertices to visit
  $q.push\_back(y)$;
  $v := []$; // empty list of visited nodes
  **while** $q \neq \emptyset$ **do**
    $y := q.pop\_front()$;
    $v.push\_back(y)$;
    **for** $y' \in \mathfrak{N}_T(y) \setminus v$ **do**
      draw $u \sim \mathsf{Bernoulli}(\alpha)$;
      **if** $u = 1$ **then**
        $q.push\_back(y')$;
  let $T' = T[v]$;
  **return** $T'$;

**Algorithm 3:** Stochastic breadth-first tree traversal.

## A.2 Junction-tree expander (detailed steps)

Below follows a more detailed description of the JTE. The full algorithm is given in Algorithm 4.

**Input**: $T \in \mathsf{T}_m$
**Output**: $T_+ \in \mathsf{T}_{m+1}$
// *Step* 1.
draw $T' = (V', E') \sim \mathbf{S}(T, \cdot)$;
let $T_+$ be a copy of $T$;
**if** $T' = (\emptyset, \emptyset)$ **then**
  // *Step* 2.
  add a link from $\{y_{m+1}\}$ to one of the nodes in $T_+$;
  // *Step* 3.
  cut $T_+$ at the separator $\emptyset$;
  // *Step* 4.
  randomly connect $T_+$ into a forest;
  **return** $T_+$;
**else**
  // *Step* 2*.
  enumerate the nodes in $T'$ by $c_1, \ldots, c_{|V'|}$;
  **for** $j = 1 \to |V'|$ **do**
    set $z_j \leftarrow \bigcup_{c \in \mathfrak{N}_{T'}(c_j)} s_{c_j,c}$;
    **if** *there exists some* $c \in \mathfrak{N}_{T'}(c_j)$ *such that* $z_j = s_{c_j,c}$ **then**
      draw $q_j \sim \mathsf{Unif}\langle \wp(c_j \setminus z_j) \setminus \{\emptyset\} \rangle$;
    **else**
      draw $q_j \sim \mathsf{Unif}\langle \wp(c_j \setminus z_j) \rangle$;
    set $d_j \leftarrow z_j \cup q_j$ and $d_j^+ \leftarrow d_j \cup \{y_{m+1}\}$;
    add $d_j^+$ to $T_+$;
    **if** $d_j = c_j$ **then**
      remove $c_j$ and its incident links from $T_+$;
  // *Step* 3*.
  **foreach** *link* $(c_j, c_k)$ *in* $T'$ **do**
    remove $(c_j, c_k)$ from $T_+$; // if not removed on Line 19
    add $(d_j^+, d_k^+)$ to $T_+$;
  **for** $j = 1 \to |V'|$ **do**
    **if** $d_j = c_j$ **then**
      let $\mathfrak{N}_T(c_j) \setminus V'$ be neighbors of $d_j^+$ in $T_+$;
  **for** $j = 1 \to |V'|$ **do**
    **if** $d_j^+$ *and* $c_j$ *are nodes in* $T_+$ **then**
      add the link $(c_j, d_j^+)$ to $T_+$;
  // *Step* 4*.
  **for** $j = 1 \to |V'|$ **do**
    **if** $d_j \neq c_j$ **then**
      draw $n_j \sim \mathsf{Unif}\langle \wp(\{c \in \mathfrak{N}_{T_+}(c_j) : s_{c,c_j} \subseteq d_j\}) \rangle$;
      move the neighbors $n_j$ of $c_j$ to be neighbors of $d_j^+$ instead;
  **return** $T_+$

**Algorithm 4:** Junction-tree expander.

#### Step 1: Subtree simulation

In this step, a random subtree $T' = (V', E')$ of $T$ is sampled from $\mathbf{S}^{\alpha,\beta}(T, \cdot)$ (Line 1). After this, a new tree $T_+$ is initiated as a copy of $T$ (Line 2), and all the manipulations described below refers to $T_+$. Depending on whether $T'$ is empty or not, the algorithm proceeds in two substantially different ways.

#### Step 2: Node creation

If $T'$ is empty, the new vertex $y_{m+1}$ is added as a node $\{y_{m+1}\}$ in its own and connected to one arbitrary existing node.

#### Step 3 and 4: Randomising the tree

The tree is then cut at each link associated with the empty separator and reconstructed, a process we call *randomisation at the separator* $\emptyset$ (Lines 4–6); see Appendix B or (Thomas and Green 2009b) for details. The randomisation step might seem superfluous at a first glance; however, it turns out to be needed in order to ensure that every junction tree has, as stated in Theorem 2, a positive probability of being produced by iterative application of the algorithm.

#### Step 2*: Node creation

If $T'$ is nonempty, the idea is to replicate its structure so that at the end of the algorithm, a subtree $T'_+$ of $T_+$ has been created where every node contains $y_{m+1}$. More specifically, for each node $c_j$, $j \in \{1, \ldots, |V'|\}$, in $T'$, a new node $d_j^+$ is created by connecting $y_{m+1}$ to a subset of $c_j$ while ensuring that the decomposability of $g(T_+)$ is still maintained. If $T'$ has more than one node, it is, for each $j$, in order to avoid that a 4-cycle is formed in $g(T_+)$, necessary to connect $y_{m+1}$ to all vertices in $z_j := \bigcup_{c \in \mathfrak{N}_{T'}(c_j)} s_{c_j,c}$. For the rest of the vertices in $c_j$, a subset $q_j$ is sampled uniformly at random, and $d_j^+$ is formed as the union of $q_j$, $z_j$, and $\{y_{m+1}\}$ (Lines 11–16). In the case where $z_j$ is identical to one of the separators $s_{c_j,c}$, $c \in \mathfrak{N}_{T'}(c_j)$, it is necessary that $q_j$ is nonempty in order to prevent the new node from being engulfed by some of its neighbors in $T'_+$ (Line 15). In the case where $y_{m+1}$ is connected to every vertex in $c_j$, $c_j$ is replaced by $d_j^+$ (Lines 17–19).

#### Step 3*: Structure replication

Having created the new nodes $\left(d_j^+\right)_{j=1}^{|V'|}$, links will be added between $d_j^+$ and $d_k^+$ whenever there is a link between $c_j$ and $c_k$ in $T'$ (Line 22). In this case, the link between $c_j$ and $c_k$ is removed (Line 21) in order to avoid a 4-cycle to be formed on Line 28. By this measure, $T'_+$ replicates the structure of $T'$. In order to connect $T_+$ into a tree, links are added between each pair of nodes $d_j^+$ and $c_j$ (Line 28).

#### Step 4*: Neighbor relocation

Finally, we observe that for all $j \in \{1, \ldots, |V'|\}$, any potential neighbor $c \in \mathfrak{N}_{T_+}(c_j)$ such that $s_{c,c_j} \subseteq d_j$ can be moved to be a neighbor of $d_j^+$ instead while maintaining the junction tree property (Lines 31–32). In the special case where the node $c_j$ is substituted by $d_j^+$, all the neighbors of $c_j$ will simply be neighbors of $d_j^+$ instead (Line 25).

### A.3 Junction-tree collapser (detailed steps)

Below follows some more detailed description of the JTC. The full algorithm is given in Algorithm 5.

**Input**: $T_+ = (V_+, E_+) \in \mathsf{T}_{m+1}$
**Output**: $T \sim \mathbf{R}_m(T_+, \cdot)$
1 let $T$ be a copy of $T_+$;
2 **if** $\{y_{m+1}\} \in V_+$ **then**
3  remove $\{y_{m+1}\}$ and its incident edges from $T$;
4  connect $T$ into a tree using Algorithm A.4 (Appendix B);
5 **else**
6  let $T'_+$ be the subtree of nodes in $T_+$ containing $y_{m+1}$;
7  enumerate the nodes in $T'_+$ by $d_1^+, \ldots, d_{|V'_+|}^+$;
8  **for** $j \to 1, \ldots, |V'_+|$ **do**
9  let $M_j \leftarrow \{c \in \mathfrak{N}_{T_+}(d_j^+) : s_{c,d_j^+} = d_j^+ \setminus \{y_{m+1}\}\}$;
10  **if** $M_j = \emptyset$ **then**
11  $c_j \leftarrow d_j^+ \setminus \{y_{m+1}\}$;
12  **else**
13  draw $c_j \sim \mathsf{Unif}\langle M_j \rangle$;
14  let $\mathfrak{N}_{T_+}(d_j^+) \setminus c_j$ be neighbors of $c_j$ in $T$;
15  remove $d_j^+$ and its incident links from $T$;
16 **return** $T$;

**Algorithm 5:** Junction-tree collapser.

Similarly to the JTE, the JTC takes two different forms depending on whether $\{y_{m+1}\}$ is present as a node in $T_+$ or not. Specifically, if $\{y_{m+1}\} \in V_+$, then $\{y_{m+1}\}$ is removed from $T_+$ and the resulting forest is reconnected uniformly at random (Lines 2–4 in Algorithm 5).

Otherwise, if $\{y_{m+1}\} \notin V_+$ we denote by $\{d_j^+\}_{j=1}^{|V'_+|}$ the nodes in the subtree $T'_+$ induced by the nodes containing the vertex $y_{m+1}$. The aim is now to identify the nodes that can serve as a subtree in Algorithm 4 to produce $T_+$. Since each node in the subtree sampled initially in Algorithm 4 will give rise to a new node, it is enough to determine, for each $j \in \{1, \ldots, |V'|\}$, the node $c_j$ that can be used for producing $d_j^+$ (reversing Lines 10–19 in Algorithm 4). For each $j$, we define a set of candidate nodes $M_j = \{c \in V_+ : d_j^+ \cap c = d_j\}$. If $M_j = \emptyset$, we let $c_j = d_j^+ \setminus \{y_{m+1}\}$ (Line 11 in Algorithm 5). Otherwise, $c_j$ is drawn at random from the

uniform distribution over $M_j$ (Line 13). In either case, the edges incident to $d_j^+$ are moved to $c_j$ (Line 14).

### A.4 Proofs and lemmas

**Lemma 1** *Let $T$ be a tree where each node is a subset of some finite set. Then $T$ satisfies the junction tree property if and only if for any path $c_1 \sim c_\ell = (c_1, \ldots, c_\ell)$ in $T$ it holds that*

$$c_1 \cap c_\ell \subset \bigcap_{j=2}^{\ell} s_{c_{j-1},c_j}.$$

***Proof*** The statement of the lemma follows by noting that

$$\bigcap_{j=1}^{\ell} c_j = \bigcap_{j=2}^{\ell} (c_{j-1} \cap c_j) = \bigcap_{j=2}^{\ell} s_{c_{j-1},c_j},$$

which implies that

$$c_1 \cap c_\ell \subset \bigcap_{j=1}^{\ell} c_j \iff c_1 \cap c_\ell \subset \bigcap_{j=2}^{\ell} s_{c_{j-1},c_j}.$$

□

***Proof of Theorem 1*** We prove this theorem by taking a generative perspective in the sense that we rely on the sampling procedure of $\mathbf{K}_m(T, \cdot)$ given by Algorithm 4. We also adopt the same notation as in Algorithm 4.

In order to prove (i) we assume that $T_+$ is generated by Algorithm 4 with input $T$ and show that $T_+ \in \mathsf{T}_{m+1}$ by going through the algorithm in a step-by-step fashion. At Line 1 a subtree $T'$ is drawn. We treat the cases $T' = (\emptyset, \emptyset)$ and $T' \neq (\emptyset, \emptyset)$ separately.

First, assume that $T' = (\emptyset, \emptyset)$. Since the node $\{y_{m+1}\}$ does not intersect any other node in $T$, it can be connected to an arbitrary node with separator $\emptyset$ without violating the junction tree property (Line 4). In addition, Thomas and Green (2009a) show that randomising a tree at a given separator preserves the junction tree property (Line 6).

For $T' \neq (\emptyset, \emptyset)$, we first show that $T_+$ produced on Lines 9–28 is a tree that satisfies the junction tree property. Indeed, $T_+$ is a tree since the subtrees produced up to Line 25 are all reconnected through the same tree $T_+[\{d_j^+\}_{j=1}^{|V'|}]$ by the operations on Lines 26–28. To ensure the junction tree property of $T_+$, consider a general path

$$(a_1, \ldots, a_{\ell_1}, c_{r_1}, d_{r_1}^+, \ldots, d_{r_{\ell_2}}^+, c_{r_{\ell_2}}, b_1, \ldots, b_{\ell_3})$$

passing through $T'_+$ in $T_+$, where $(a_j)_{j=1}^{\ell_1}$ and $(b_j)_{j=1}^{\ell_2}$ are nonempty sequences of nodes which also belong to $T$.

The fact that $(d_{r_j}^+)_{j=1}^{\ell_2}$ is the $d_{r_1}^+$-$d_{r_{\ell_2}}^+$ path in $T_+$ implies that $(c_{r_j})_{j=1}^{\ell_2}$

is the $c_{r_1}$-$c_{r_{\ell_2}}$ path in $T$, since, by construction (Lines 20–22), $(c_j, c_k) \in E'$ if and only if $(d_j^+, d_k^+) \in E_+$. Thus,

$$(a_1, \ldots, a_{\ell_1}, c_{r_1}, \ldots, c_{r_{\ell_2}}, b_1, \ldots, b_{\ell_3})$$

is the $a$-$b$ path in $T$. The junction tree property of $T$ ensures that $a \cap b \subset I_{a\sim b}$, where

$$I_{a\sim b} := \left(\bigcap_{j=1}^{\ell_1} a_j\right) \cap \left(\bigcap_{j=1}^{\ell_2} c_{r_j}\right) \cap \left(\bigcap_{j=1}^{\ell_3} b_j\right). \tag{A.1}$$

Now, consider the intersection

$$I_{a\sim b}^+ := \left(\bigcap_{j=1}^{\ell_1} a_j\right) \cap c_{r_1} \cap \left(\bigcap_{j=1}^{\ell_2} d_{r_j}^+\right) \cap c_{r_{\ell_2}} \cap \left(\bigcap_{j=1}^{\ell_3} b_j\right)$$

of the nodes in $a \sim b$ in $T_+$. For $\ell_2 = 1$, it holds that $d_{r_1}^+ = c_{r_1} \cup \{y_{m+1}\}$, corresponding to the case where $c_{r_1}$ is engulfed into $d_{r_1}^+$. For $\ell_2 \geq 2$, the junction tree property in $T$ ensures via Lemma 1 that

$$\begin{aligned}\bigcap_{j=1}^{\ell_2} d_{r_j}^+ &= \bigcap_{j=2}^{\ell_2} s_{d_{r_{j-1}}^+, d_{r_j}^+} = \left(\bigcap_{j=2}^{\ell_2} s_{c_{r_{j-1}}, c_{r_j}}\right) \cup \{y_{m+1}\} \\ &= \left(\bigcap_{j=1}^{\ell_2} c_{r_j}\right) \cup \{y_{m+1}\}.\end{aligned}$$

It hence holds that $a \cap b \subset I_{a\sim b} \subset I_{a\sim b}^+$.

Now, consider the final version of $T_+$ obtained after the relocation step on Lines 29–32. Let $c \in n_j$ and let $T_{s_{c_j,c}}$ be the subtree of $T_+$ induced by the nodes containing the separator $s_{c_j,c}$. In addition to $c$ and $c_j$, it is clear that $d_j^+$ is also a node in $T_{s_{c_j,c}}$ since $s_{c_j,c} \subseteq s_{d_j^+,c}$. Now the fact that $s_{c_j,c} \subseteq s_{d_j^+,c}$ also implies that the tree obtained by letting $c$ be a neighbor of $d_j^+$ instead of $c_j$ also satisfies the junction tree property by Thomas and Green (2009b).

Finally, (ii) follows directly since the only new vertex added to $g(T)$ in order to get $g(T_+)$ is $y_{m+1}$ and no edges have been removed between the vertices $\{y_j\}_{j=1}^m$. □

***Proof of Theorem 2*** In this proof we use the property (ii) of $\mathbf{R}_m$ provided by Theorem 3 and proved independently below.

The space containing the trivial junction tree is $\mathsf{T}_1 = \{(\{y_1\}, \emptyset)\}$. We proceed by induction over the number of vertices. For the base case $m = 2$, $\mathsf{T}_2 = \{T_1, T_2\}$, where $T_1 = (\{\{y_1, y_2\}\}, \emptyset)$ is the unique tree constructed from $(\{\{y_1\}\}, \emptyset)$ via the subtree $(\{\{y_1\}\}, \emptyset)$ and $T_2 = (\{\{y_1\}, \{y_2\}\}, \{(\{y_1\}, \{y_2\})\})$ is the unique tree constructed from $(\{\{y_1\}\}, \emptyset)$ via the subtree $(\emptyset, \emptyset)$.

For $m \geq 3$, assume inductively that $\text{Supp}(\mathbf{K}^{m-1}(\cdot)) = \mathsf{T}_{m-1}$ and let $T \in \mathsf{T}_m$ be an arbitrary junction tree. It suffice to show that there exists a junction tree $T_- \in \mathsf{T}_{m-1}$ such that $\mathbf{K}_{m-1}(T_-, T) > 0$ since then $\mathbf{K}^m(T) \geq \mathbf{K}^{m-1}(T_-)\mathbf{K}_{m-1}(T_-, T) > 0$. But this follows directly by drawing any $T_- \sim \mathbf{R}_{m-1}(T, \cdot)$ since $\text{Supp}(\mathbf{R}_{m-1}(T, \cdot)) \subseteq \text{Supp}(\mathbf{K}_{m-1}(\cdot, T))$ (by (ii) in Theorem 3), meaning that $T \in \text{Supp}(\mathbf{K}_{m-1}(T_-, \cdot))$. Thus, every junction tree in $\mathsf{T}_m$ can be constructed, and we conclude the proof by induction. □

***Proof of Proposition 1*** In this proof, we take a generative perspective in the sense that we rely on the sampling procedure given by Algorithm 4 and regard $T_+$ as an expansion of $T$. We further adopt the same notation as in Algorithm 4 when possible.

Let $V'_+ = \{d_j^+\}_{j=1}^{|V'_+|}$ be the set of nodes in $T_+$ containing the vertex $y_{m+1}$. The induced subgraph $T_+[V'_+]$ will necessarily be a subtree of $T_+$ (see e.g. Blair and Peyton (1993)), which we denote by $T'_+ = (V'_+, E'_+)$. For each $j \in \{1, \ldots, |E'_+|\}$, we define a set $M_j = \{c \in V_+ : d_j^+ \cap c = d_j\}$, which we interpret as the candidate nodes in $V$ from which each $d_j^+ = d_j \cup \{y_{m+1}\} \in V_+$ could potentially have emerged. We distinguish between two main situations for $\mathsf{T}(T, T_+)$ depending on $|V'_+|$.

For $|V'_+| = 1$,

- If $|M_1| = 0$, then $c_1 = d_1^+ \setminus \{y_{m+1}\}$ so that $\mathsf{T}(T, T_+) = \{(\{c_1\}, \emptyset)\}$ due to Lines 17–19.
- If $|M_1| = 1$, then clearly $\mathsf{T}(T, T_+) = \{(\{c_1\}, \emptyset)\}$, where $\{c_1\} = M_1$ due to Lines 26–28.
- If $|M_1| \geq 2$, consider the enumeration $M_1 = \{\tilde{c}_j\}_{j=1}^{|M_1|}$. The set $\mathsf{T}(T, T_+)$ is clearly non-empty, thus we can assume that $(\{\tilde{c}_1\}, \emptyset) \in \mathsf{T}(T, T_+)$. Note that, since $M_1$ consists of more than one element, all nodes in $\{\tilde{c}_j\}_{j=2}^{|M_1|}$ are former neighbors of $\tilde{c}_1$ by Lines 29–32. Thus, every node in $\{\tilde{c}_j\}_{j=2}^{|M_1|}$ are neighbors of $\tilde{c}_1$ in $T$. This implies that for $|M_1| = 2$, the subtree could also be $(\{\tilde{c}_2\}, \emptyset)$ since the $\tilde{c}_1$ could be moved at Lines 29–32. Thus $\mathsf{T}(T, T_+) = \{(\{\tilde{c}_1\}, \emptyset), (\{\tilde{c}_2\}, \emptyset)\}$. For $|M_1| \geq 3$, $(\{\tilde{c}_1\}, \emptyset)$ is necessarily the unique subtree in $\mathsf{T}(T, T_+)$ since if there would exist another subtree $(\{\tilde{c}_2\}, \emptyset)$, both $\tilde{c}_1$ and $\tilde{c}_2$ would have $\tilde{c}_3$ as neighbor in $T$, which would form a cycle.

For $|V'_+| \geq 2$, by construction, for each link $(d_j^+, d_k^+) \in E'_+$, where $d_j^+ = d_j \cup \{y_{m+1}\}$ and $d_k^+ = d_k \cup \{y_{m+1}\}$ we can associate a link $(c_j, c_k) \in E$, where $c_j = d_j \cup r_j$ and $c_k = d_k \cup r_k$ are emerging nodes in $T$ and $r_j$ and $r_k$ may be empty sets. Thus we can form the subtree $T' = (V', E')$ which we regard as the subtree in Algorithm 4 (Line 1), where $V' = \{c_j \in V : d_j^+ \in V'_+\}$ and $E' = \{(c_j, c_k) \in E : (d_j^+, d_k^+) \in E'_+\}$. Now, suppose that there exists another subtree $T'' = (V'', E'')$, isomorphic to $T'$, where $V'' = \{c'_j = d_j \cup r'_j : d_j^+ \in V''_+\}$, $c'_j \neq c_j$ for some $j$ and $E'' = \{(c'_j, c'_k) \in E : (d_j^+, d_k^+) \in E'_+\}$. Enumerate the nodes such that $c'_1 \neq c_1$ and let for simplicity $c'_2 = c_2 \in \mathfrak{N}(c'_1)$. Then, since the neighbors of $d_1^+$ except for $c_1$ are neighbors of $c_1$ in $T$, the link $(c_1, c'_1)$ would be present in $T$. Also the link $(c_1, c_2)$ is present in $T$ since $T'$ is a subtree of $T$. Similarly, since $T''$ is a subtree of $T$, the link $(c'_1, c_2)$ would also be present in $T$. Thus we would have a 3-cycle in $T$ which contradicts the assumption of $T$ being a tree. Thus $\mathsf{T}(T, T_+) = \{T'\}$. □

***Proof of Theorem 3*** We prove this theorem by taking a generative perspective in the sense that we rely on the sampling procedures of $\mathbf{K}_m$ and $\mathbf{R}_{m+1}$ given by Algorithm 4 and 5 respectively. We also adopt the same notation as in these algorithms. To show (i) and (ii) we distinguish between the cases $\{y_{m+1}\} \in V_+$ and $\{y_{m+1}\} \notin V_+$. For both cases, we let $T \in \text{Supp}(\mathbf{R}_{m+1}(T_+, \cdot))$. We prove (i) by following the steps in Algorithm 5 with input $T_+$. For (ii), we show that $T_+$ could be obtained by Algorithm 4 with input $T$.

If $\{y_{m+1}\} \in V_+$, then no other node in $T_+$ will contain the vertex $y_{m+1}$ which in turn implies that each neighbor in $\mathfrak{N}_{T_+}(\{y_{m+1}\})$ will have $\emptyset$ as associated separator. Removing one node from a tree will always result in a forest possibly containing only one tree. Thus the removal of $\{y_{m+1}\}$ from $T$ on Line 3 will result in a forest. Since $\{y_{m+1}\}$ is not contained in any of the trees in the forest, these will all trivially satisfy the junction tree property and the connection of $T$ into a tree by Line 4 will give a random junction tree for $g(T_+)[\{y_\ell\}_{\ell=1}^m]$, which proves (i) in this case. For the (ii) part, we simply observe that $T_+$ can be constructed from $T$ by first drawing the empty subtree on Line 1 and then obtaining $T_+$ at the randomization on Line 6 in Algorithm 4.

Now, assume that $\{y_{m+1}\} \notin V_+$. We proceed by showing (i), i.e. that $T \in \mathsf{T}_m$. We first show that $T$ is a tree. Since for every $j \in \{1, \ldots, |V'_+|\}$, all elements in $\mathfrak{N}_{T_+}(d_j^+) \setminus c_j$ are set to be neighbors of $c_j$ in $T$, for all $j, k \in \{1, \ldots, |V'_+|\}$ it follows that

$$(d_j^+, d_k^+) \in E_+ \iff (c_j, c_k) \in E.$$

Hence, since $T'_+$ is a subtree of $T_+$, $T' = (V', E')$ is a tree, where $V' = \{c_j\}_{j=1}^{|V'_+|}$ and $E' = \{(c_j, c_k) \in E : (d_j^+, d_k^+) \in E_+\}$. Further, since elements of $\mathfrak{N}_{T_+}(d_j^+) \setminus c_j$ are not mutual neighbors, and parts of distinct subtrees of $T$, $T'$ is a subtree of $T$. As a consequence, we may assume that an arbitrary path (of length at least 2) in $T$ is of form

$$(a_1, \ldots, a_{\ell_1}, c_{r_1}, \ldots, c_{r_{\ell_2}}, b_1, \ldots, b_{\ell_3}),$$

where $\ell_1 \geq 0$, $\ell_2 \geq 0$, $\ell_3 \geq 0$ and $\{c_{r_j}\}_{j=1}^{\ell_2} \subseteq V'$. Let $a$ and $b$ be the first and last element in this path, respectively. Let the intersection $I_{a \sim b}$ be defined by (A.1). We must prove that $a \cap b \subset I_{a \sim b}$. We know that in $T_+$, the node $a_{\ell_1}$ was connected

to either $c_{r_1}$ (in which case $(c_{r_1}, d^+_{r_1}) \in E_+$) or to $d^+_{r_1}$ and $b_1$ was connected to either $c_{r_{\ell_2}}$ (in which case $(c_{r_{\ell_2}}, d^+_{r_{\ell_2}}) \in E_+$) or to $d^+_{\ell_2}$. First, assume that $a_{\ell_1}$ was connected to $d^+_{r_1}$ and $b_1$ was connected to $d^+_{\ell_2}$, then the $a$-$b$ path in $T_+$ is of form

$$(a_1, \ldots, a_{\ell_1}, d^+_{r_1}, \ldots, d^+_{r_{\ell_2}}, b_1, \ldots, b_{\ell_3}).$$

Let

$$I^+_{a\sim b} = \left(\bigcap_{j=1}^{\ell_1} a_j\right) \cap \left(\bigcap_{j=1}^{\ell_2} d^+_{r_j}\right) \cap \left(\bigcap_{j=1}^{\ell_3} b_j.\right)$$

We know that, since $T_+$ is a junction tree, $a \cap b \subset I^+_{a\sim b}$. Moreover, by Lemma 1 it holds that $\bigcap_{j=1}^{\ell_2} d^+_{r_j} = \bigcap_{j=1}^{\ell_2} c_{r_j} \cup \{y_{m+1}\}$. But, $y_{m+1} \notin a$ and $y_{m+1} \notin b$, thus $I_{a\sim b} = I^+_{a\sim b}$ so that $a \cap b \subset I_{a\sim b}$. Now, note that $I^+_{a\sim b} \cap (c_{r_1} \cap c_{r_{\ell_2}}) = I^+_{a\sim b}$, so that adding $c_{r_1}$ and $c_{r_{\ell_2}}$ to the path does not change anything, thus the junction tree property also holds in the case where $a_{\ell_1}$ was connected to $c_{r_1}$ or $b_1$ was connected to $c_{r_{\ell_2}}$.

To show (ii) in this case, observe that $T$ can be expanded to $T_+$ by first drawing the subtree $T'$ on Line 1. Then, by identifying $d_j = d^+_j \cap c_j$ and $z_j = \bigcup_{c \in \mathfrak{N}_{T'_+}(d^+_j)} s_{d^+_j, c}$, there is a positive probability for obtaining $q_j = d_j \setminus z_j$ for $j \in \{1, \ldots, |V'|\}$ at Lines 13–15 (Algorithm 4). The neighbors of $d^+_j$ for the resulting tree can now be set to be identical to that in $T_+$ by letting $n_j = \mathfrak{N}_{T_+}(d^+_j) \setminus c_j \setminus V'_+$ on Line 31.

To show (iii) we simply observe that the only vertex removed from $T_+$ compared to $T$ is $y_{m+1}$ so that $g(T_+)[\{y_\ell\}_{\ell=1}^m] = g(T)$. □

***Proof of Theorem 7*** To reduce some notation we define $A := A(G)$ and $A_+ := A(G_+)$. Consider the partitions of the separator sets $S(G) = A \cup A^{\mathsf{c}}$ and $S(G_+) = A_+ \cup A^{\mathsf{c}}_+$. In order to show that the factorisation holds it is enough to establish that

1. $S(G_+) = A_+ \cup A^{\mathsf{c}}$,
2. $T_s = (T_+)_s$ for $s \in A^{\mathsf{c}}$,

where $T$ and $T_+$ are arbitrary junction tree representations of the graphs $G$ and $G_+$ respectively. Note that, showing (1) is equivalent to showing that $A^{\mathsf{c}}_+ = A^{\mathsf{c}}$.

First let $s \in A^{\mathsf{c}}_+$. It suffice to show that $s \in S(G)$ since then it follows that $s \in A^{\mathsf{c}}$. But since $s$ is in $S(G_+)$ and was not created by the expansion ($s \notin S^\star$), it has to come from $G$, i.e. $s \in S(G)$. It follows that $A^{\mathsf{c}}_+ \subseteq A^{\mathsf{c}}$.

For the other inclusion, let $s \in A^{\mathsf{c}}$. It suffices to show that $s \in S(G_+)$ since then it follows that $s \in A^{\mathsf{c}}_+$. But if $s \in S(G)$ and $s$ is not subset of any separator in $S(G_+)$, it cannot have been removed by the expansion meaning that $s \in S(G_+)$. Thus, $A^{\mathsf{c}} \subseteq A^{\mathsf{c}}_+$. It follows that $A^{\mathsf{c}} = A^{\mathsf{c}}_+$.

To show 2, let $s \in A^{\mathsf{c}}$ and consider the tree $T_s$ spanned by the nodes in $T$ associated with separators which are subsets of $s$. Assume that the tree $(T_+)_s$, spanned by the nodes in $T_+$ associated with separators which are subsets of $s$, is different from $T_s$. This could only occur in two ways: (i) some new separator $s^\star$ that contains $s$ has been created or (ii) some separator containing $s$ has been removed. However, (i) cannot happen since then $s^\star$ would be a new separator in $S^\star$ that would also contain $s$ which was not true by assumption. Thus, (ii) must hold. But the only way a separator $s'$ of $T_s$ can be removed is if a new separator $s' \cup \{y_{m+1}\}$ also is created. But then $s' \cup \{y_{m+1}\}$ would be a new separator in $S^\star$ containing $s$, leading to a contradiction. This implies that $\nu_G(s) = \nu_{G_+}(s)$. □

## Appendix B

**Randomize junction tree at separator** *(Thomas and Green (2009b)) Given any particular junction tree representation $T$, we can choose uniformly at random from the set of equivalent junction trees by applying the following algorithm to the forests $\mathsf{F}_s(T)$ defined by the distinct separators $s$ in $T$. Following the notation in Theorem 4, $r_i$ refers to the size of subtree $i$.*

Step 1. *Label each vertex of the forest $\{i, j\}$ where $1 \leq i \leq q$ and $1 \leq j \leq r_i$, so that the first index indicates the subtree the vertex belongs to and the second reflects some ordering within the subtree. The ordering of the subtrees and of vertices within subtrees are arbitrary.*

Step 2. *Construct a list $v$ containing $q - 2$ vertices each chosen uniformly at random with replacement from the set of all $p$ vertices.*

Step 3. *Construct a set $w$ containing $q$ vertices, one chosen uniformly at random from each subtree.*

Step 4. *Find in $w$ the vertex $x$ with the largest first index that does not appear as a first index of any vertex in $v$. Because the length of $v$ is 2 less than the size of $w$, there must always be at least two such vertices.*

Step 5. *Connect $x$ to $y$, the vertex at the head of the list $v$.*

Step 6. *Remove $x$ from the set $w$, and delete $y$ from the head of the list $v$.*

Step 7. *Repeat from step 4 until $v$ is empty. At this point $w$ contains two vertices. Connect them.*

## References

Andrieu, C., Doucet, A., Holenstein, R.: Particle Markov chain Monte Carlo methods. J. R. Stat. Soc.: Ser. B (Statistical Methodology) **72**(3), 269–342 (2010)

Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Trans. Signal Process. **50**(2), 174–188 (2002)

Blair, J.R., Peyton, B.: An introduction to chordal graphs and clique trees. In: George, A., Gilbert, J.R., Liu, J.W. (Eds.), Graph Theory and Sparse Matrix Computation, volume 56 of The IMA Volumes in Mathematics and its Applications, pages 1–29. Springer New York (1993). ISBN 978-1-4613-8371-0. https://doi.org/10.1007/978-1-4613-8369-7_1

Cappé, O., Moulines, E., Rydén, T.: Inference in hidden Markov models. Springer, New York (2005)

Chopin, N., Papaspiliopoulos, O., et al.: An introduction to sequential Monte Carlo. Springer, Switzerland (2020)

Del Moral, P.: Feynman-Kac formulae: genealogical and interacting particle systems with applications, vol. 88. Springer, Switzerland (2004)

Del Moral, P., Doucet, A., Jasra, A.: Sequential Monte Carlo samplers. J. R. Stat. Soc. Series B (Statistical Methodology) **68**(3), 411–436 (2006). ISSN 13697412, 14679868. URL http://www.jstor.org/stable/3879283

Elmasri, M.: On decomposable random graphs. *ArXiv e-prints*, (2017)

Elmasri, M.: Sub-clustering in decomposable graphs and size-varying junction trees. *ArXiv e-prints*, (2017)

Eppstein, D.: Graph-theoretic solutions to computational geometry problems. In: International Workshop on Graph-Theoretic Concepts in Computer Science, pages 1–16. Springer (2009)

Frydenberg, M., Lauritzen, S.L.: Decomposition of maximum likelihood in mixed graphical interaction models. Biom. **76**(3), 539–555 (1989)

Giudici, P., Green, P.J.: Decomposable graphical Gaussian model determination. Biom. **86**(4), 785–801 (1999)

Gordon, N.J., Salmond, D.J., Smith, A.F.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In: IEE Proceedings F (Radar and Signal Processing), volume 140, pages 107–113. IET (1993)

Green, P.J., Thomas, A.: Sampling decomposable graphs using a Markov chain on junction trees. Biom. **100**(1), 91–110 (2013)

Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biom. **57**(1), 97–109 (1970). ISSN 00063444. URL http://www.jstor.org/stable/2334940

Jacob, P.E., Murray, L.M., Rubenthaler, S.: Path storage in the particle filter. Stat. Comput. **25**(2), 487–496 (2015)

Kawahara, J., Saitoh, T., Suzuki, H., Yoshinaka, R.: Enumerating all subgraphs without forbidden induced subgraphs via multivalued decision diagrams. *arXiv preprint* arXiv:1804.03822, (2018)

Koskela, J., Jenkins, P.A., Johansen, A.M., Spano, D.: Asymptotic genealogies of interacting particle systems with an application to sequential Monte Carlo. Ann. Stat. **48**(1), 560–583 (2020)

Lauritzen, S.L.: Graphical Models. Oxford University Press, United Kingdom (1996). ISBN 0-19-852219-3

Markenzon, L., Vernet, O., Araujo, L.: Two methods for the generation of chordal graphs. Ann. Oper. Res. **157**(1), 47–60 (2008). https://doi.org/10.1007/s10479-007-0190-4. (**ISSN 0254-5330**)

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. J. Chem. hys. **21**(6), 1087–1092 (1953). https://doi.org/10.1063/1.1699114. URL http://scitation.aip.org/content/aip/journal/jcp/21/6/10.1063/1.1699114

Moon, J.: Enumerating labelled trees. Graph Theory and Theoretical Physics, 261271, (1967)

Olsson, J., Pavlenko, T., Rios, F.L.: Bayesian learning of weakly structural Markov graph laws using sequential Monte Carlo methods. Electron. J. Statist. **13**(2), 2865–2897 (2019)

Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Representation and Reasoning Series. Morgan Kaufmann, (1997). ISBN 9781558604797

Rios F.L.,Moffa G., Benchpress J.K.: A scalable and versatile workflow for benchmarking structure learning algorithms for graphical models. arXiv:2107.03863, (2021)

Thomas, A., Green, P.J.: Enumerating the decomposable neighbours of a decomposable graph under a simple perturbation scheme. Comput. stat. & data anal. **53**(4), 1232–1238 (02 2009). https://doi.org/10.1016/j.csda.2008.10.029. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2680312/

Thomas, A., Green, P.J.: Enumerating the junction trees of a decomposable graph. J. Comput. Graph. Stat. **18**(4), 930–940 (2009). https://doi.org/10.1198/jcgs.2009.07129

Tierney, L.: Markov chains for exploring posterior distributions. the Annals of Statistics, 1701–1728 (1994)

van den Boom, W., Jasra, A., De Iorio, M., Beskos, A., Eriksson, J.G.: Unbiased approximation of posteriors via coupled particle Markov chain Monte Carlo. Stat. Comput. **32**(3), 36 (2022)

Wormald, N.C.: Counting labelled chordal graphs. Graphs and Combinatorics **1**(1), 193–200 (1985). (**ISSN 0911-0119**)