Parallel sampling of decomposable graphs using Markov chain on junction trees

Mohamad Elmasri*

January 2, 2024

Abstract

Bayesian inference for undirected graphical models is mostly restricted to the class of decomposable graphs, as they enjoy a rich set of properties making them amenable to high-dimensional problems. While parameter inference is straightforward in this setup, inferring the underlying graph is a challenge driven by the computational difficulty in exploring the space of decomposable graphs. This work makes two contributions to address this problem. First, we provide sufficient and necessary conditions for when multi-edge perturbations maintain decomposability of the graph. Using these, we characterize a simple class of partitions that efficiently classify all edge perturbations by whether they maintain decomposability. Second, we propose a novel parallel non-reversible Markov chain Monte Carlo sampler for distributions over junction tree representations of the graph. At every step, the parallel sampler executes simultaneously all edge perturbations within a partition. Through simulations, we demonstrate the efficiency of our new edge perturbation conditions and class of partitions. We find that our parallel sampler yields improved mixing properties in comparison to the single-move variate, and outperforms current state-of-the-arts methods in terms of accuracy and computational efficiency. The implementation of our work is available in the Python package parallelDG.

Keywords: Conditional independence graph; Bayesian structure learning; model determination; distributed learning.

*Department of Statistical Sciences, University of Toronto, 100 St. George Street, Toronto, ON M5S 3G3, Canada; E-mail: mohamad.elmasri@utoronto.ca.
1 Introduction

A graphical model represents a collection of joint probability distributions for a vector of random variables $Y = (Y_1, \ldots, Y_p)$. In these models, the distributions are subject to conditional independence constraints specified by a graph, comprised of vertices $\{1, \ldots, p\}$. A notable class of graphical models focuses on cases where the underlying graph $G$ is undirected and decomposable (chordal). Bayesian structure learning, or model determination, involves inferring the underlying conditional independence graph and the model parameters concurrently. This process is based on observed data and predefined prior specifications. This work develops Markov chain Monte Carlo methods for computational inference in this settings.

The decomposability assumption is a severe restriction on the space of possible graphs. Less than $8 \times 10^{-5}$ of graphs with 12 vertices are decomposable, and exact enumeration exists only for graphs with up to 15 vertices (Olsson et al., 2022; Wormald, 1985). It is therefore impractical to use Bayesian methods that require quantification of the prior normalization constant. While general Bayesian structure learning methods can take minutes to converge for medium-sized problems (Mohammadi et al., 2023), assuming decomposability shrinks this convergence time to seconds. In sparse high-dimensional problems, this inferential efficiency is vital.

The computational advantage stems from the unique clique-separator factorization property of decomposable graphs. Specifically, when a decomposable graph $G$ represents the conditional independence constraints of a random vector $Y$, the joint distribution of $Y$ can be expressed as:

$$p(Y) = \frac{\prod_{C \in \mathcal{C}(G)} p(Y_C)}{\prod_{S \in \text{sep}(G)} p(Y_S)},$$

where $Y_A$ is a subvector of $Y$ indexed by the set $A$, $\mathcal{C}(G) = \{C_1, \ldots, C_c\}$ is a set of complete subgraphs of $G$, known as maximal cliques, and $\text{sep}(G)$ is a set of intersections between elements in $\mathcal{C}(G)$ (Lauritzen, 1996, Ch. 4.4). This factorization facilitates statistical inference by permitting operations on the marginals of $Y$ that are specified by $\mathcal{C}(G)$.

The line of work by Frydenberg and Lauritzen (1989); Giudici and Green (1999); Thomas and Green (2009); Green and Thomas (2013) has led to the development of an efficient Metropolis–Hastings (MH) algorithm for decomposable graphical models (Hastings, 1970). This method leverages the junction tree representation of decomposable graphs to create efficient proposals that preserve decomposability throughout the states of the chain. The junction tree sampler introduced in Green and Thomas (2013) is particularly fast, even for large-dimensional problems. However, it exhibits a high degree of within-sample correlation, a topic explored in Section 7 and by Olsson et al. (2019). The class of multi-edge perturbations in Green and Thomas (2013) only sufficiently maintains decomposability across the chain, which results in suboptimal proposals. These proposals not only increase the rejection rate of the sampler but also impact its overall efficiency and accuracy.

Our first contribution delineates the sufficient and necessary conditions for multi-edge updates to preserve the decomposability of the underlying graph. We derive this understanding from a simple algorithm that generates decomposable graphs via random walks on trees (Sec. 3), which exhibits junction tree-like properties. Leveraging these properties, we define a partitioning scheme for junction trees (Sec. 4). Specifically, for any given graph vertex, all tree elements are divided into three disjoint sets: those the vertex can connect
to, disconnect from, and all others. These sets enumerate all possible edge perturbations related to the vertex that maintain decomposability.

Our second contribution (Sec. 5) introduces a hierarchical latent sampler for junction trees. This sampler projects junction trees into higher dimensions while retaining their desirable factorization properties (Sec. 5.2). While updates in this latent space do not directly correspond to graph updates, when combined with our multi-edge partitioning sets, they enable simultaneous parallel updates on the graph. To demonstrate the benefits of parallelism, we first implement a single-move reversible MH Markov chain (Mc) (Sec. 5.2) and compare it with a parallel non-reversible MH-Mc (Sec. 6).

![Figure 1: ROC curves from the simulation study (Sec. 7) over 10 replications. Each replication involves a random decomposable graph, with \(p\)-vertices, generated via the Christmas tree algorithm (CTA). The plot compares our parallel and single-move samplers to state-of-the-art MH-Mc (GT13, GG99) (Green and Thomas, 2013; Giudici and Green, 1999) and Gibbs (O19) (Olsson et al., 2019) samplers, for \(n = 100, p = 100\) (left) and \(200\) (right).](image)

It is now well-understood that non-reversible chains offer significant advantages over their reversible counterparts, as documented in various studies Neal (1998); Rey-Bellet and Spiliopoulos (2015); Bierkens (2016); Duncan et al. (2016). To immediately showcase these benefits and the effectiveness of our decomposability-preserving conditions, we present an average Receiver Operating Characteristic (ROC) plot across 10 replications in Figure 1. In each replication, a random decomposable graph \(G\) is generated using the Christmas tree algorithm (Olsson et al., 2022), with \(p = \{50, 200\}\) vertices and a sample size of 100 from a Gaussian graphical model under a \(G\)-Wishart prior (Roverato, 2002). Our proposed parallel and single-move samplers achieve impressive accuracy in the general case (\(p = 50\)) and, more so, in the high-dimensional case (\(p = 200\)), than the state-of-the-art MH-Mc junction tree-based samplers (GT13-\{m,s\}) by Green and Thomas (2013), the MH-Mc graph-based sampler (GG99) by Giudici and Green (1999), and the Gibbs sampler (O19) by Olsson et al. (2019). Remarkably, our samplers converge in less than 3 minutes for all cases. Competing MH-Mc samplers converge in 1–3 for \(p = 50\) and over an hour for \(p = 200\). The Gibbs sampler (O19) demonstrates limited feasibility for larger \(p\) values, requiring as much as 5 hours to converge at \(p = 50\). To our knowledge, this problem has not been addressed by any frequentist methods, nor are there other fully Bayesian approaches.

Our simulation study (Sec. 7) further reveals a substantial reduction in within-sample correlation, accelerated and more stable convergence, and more than five-fold increase in the acceptance rate of our sampler relative to state-of-the-arts methods. For more details, refer to Section 7 and Supplementary Material (SM) Sections G and H.

The multi-edge partitioning sets and parallel sampling approach we’ve characterized have potential applications in various statistical problems, extending beyond the scope of
this work. This includes the sampling of directed acyclical graphs and potential applications in computational graph theory. Similar to the approaches in Giudici and Green (1999); Green and Thomas (2013); Olsson et al. (2019), our latent junction tree sampler is compatible with proper parameter-inference methods across different statistical models. A notable application is in multinomial models for discrete data (Tarantola, 2004).

2 Decomposable graphs and junction trees

We begin by reviewing some theoretical underpinnings and properties of decomposable graphs and junction trees. We frame this work in the classic graphical modelling literature, where Lauritzen (1996) and Cowell et al. (2006) are excellent references on the topic.

An undirected graph $G = (V, E)$ is composed of a set of vertices $V$ connected by a set of undirected edges $E$. For any subset $V \subseteq V$, $G_V$ denotes the induced subgraph with the vertex set $V$. A path $v_0 \sim v_k \subseteq G$ consists of a sequence of vertices $(v_0, v_1, \ldots, v_k)$, such that $(v_i, v_{i+1}) \in E$. A graph is connected when there is a path between every pair of vertices, and it is complete (a clique) if an edge exists between every pair of vertices. A complete subgraph is called a maximal clique if it is not a subgraph of any other clique. A graph is called a tree if there is a unique path between any pair of nodes.

A graph $G$ is decomposable if and only if the set of maximal cliques of $G$ can be ordered as $(C_1, \ldots, C_c)$, such that, for each $i = 1, \ldots, c$, if

$$S_i = C_i \cap \bigcup_{j=1}^{i-1} C_j \quad \text{then} \quad S_i \subseteq C_k, \text{ for some } k < i; \quad (2.1)$$

$S_i$ may be empty. The relation in (2.1) is called the running intersection property, and the sequence $(C_1, \ldots, C_c)$ is called the perfect ordering sequence. The set $\text{sep}(G) = \{S_1, \ldots, S_c\}$, formed by (2.1), is known as the minimal separators of $G$, and define $\text{cl}(G) = \{C_1, \ldots, C_c\}$. Maximal cliques are unique to a decomposable graph, while separators can repeat in (2.1). The edge-relation $(C_i, C_k)$ that $S_i$ forms between $C_i$ and $C_k$ in (2.1) leads to a tree representation of $G$. A (reduced) junction tree of $G$ is a tree with a vertex set as the maximal cliques and an edge set as the minimal separators of $G$ written as $J = (\text{cl}(G), \text{sep}(G))$. It follows that $G$ is decomposable if and only if it admits a junction tree representation. A decomposable graph admits multiple (reduced) junction tree representations. We denote $J^G$ to be the set of (reduced) junction trees of $G$. The maximal cardinality search algorithm of Tarjan and Yannakakis (1984) allows a junction tree representation to be found in time of order $|V| + |E|$. Refer to SM Table S2 for all notations.

As in Green and Thomas (2013), we adopt the convention of allowing separators to be empty, ensuring that every junction tree is connected. We use the terms vertex and edge specifically for the elements of $G$, and reserve cliques and separators for the elements of junction trees. This distinction is important: a vertex represents a single element, while a clique or a separator can represent multiple vertices.

Junction trees are general graphical objects defined independently of decomposable graphs. A tree $T$, with vertices as subsets of $V$, is termed a junction tree if for any pair of vertices $C_1, C_2$ in $T$, and any vertex $C$ on the unique path $C_1 \sim C_2 \subseteq T$,

$$C_1 \cap C_2 \subseteq C. \quad (2.2)$$

The relationship described by (2.2) is known as the junction property. This property can also be expressed by stating that for any subset $V \subseteq V$, the vertices in $T$ containing
V form a connected subtree. Therefore, any tree $T = (C, S)$ comprising subsets of cliques $C$ from $G$ that satisfy (2.2) qualifies as a junction tree. In light of this, we can construct junction trees for a decomposable graph $G$, where $C$ may include but is not limited to the maximal cliques $c_1(G)$.

![Decomposable graph G](image1)

Figure 2: A decomposable graph $G$, its unique junction tree $J = (c_1(G), \text{sep}(G))$, and an expanded tree $T = (C, S)$, where $c_1(G) \subseteq C$.

The decomposable graph $G$ in Figure 2 admits the junction tree $J = (c_1(G), \text{sep}(G))$, as well as the expanded junction tree $T$ that arises from the junction property. This property allows for the inclusion of additional, though probabilistically superfluous, relations in $T$. In this case, $T$ comprises four cliques: two maximal, $\{1, 2, 3\}$ and $\{2, 3, 4\}$, and two non-maximal, $\{3\}$ and $\{2, 3\}$.

Moving forward, we will consistently refer to junction trees in the context of their junction property, i.e., having $c_1(G) \subseteq C$, and denote them as $T$, unless specified otherwise. In cases where we are discussing reduced junction trees, these will be denoted as $J$. The elements of junction trees will be referred to as cliques and separators, instead of vertices and edges, reflecting their representation as subsets of vertices. A junction tree $T$ can always be compressed into a reduced junction tree $J = \text{reduce}(T) = (c_1(G), \text{sep}(G))$. This compression involves iteratively removing cliques in $T$ that are subsets of an adjacent clique, rewiring their associated edges to the adjacent superset clique, and eliminating self-loops. We denote this compression operation as $\text{reduce}(T)$.

We simplify notations by using $C \in T$ to denote a clique in $T$, and $(C, C') \in T$ to represent a separator in $T$. The notation $C$ and $S$ are reserved for sets of cliques and separators in $T$, respectively, i.e., $T = (C, S)$. For a graph vertex $v \in V$, $T_v$ denotes the induced subtree of $T$, comprising every $C \in T_v$ that includes $v$, indicated as $v \in C$. For a specific junction tree $T$, we define $g(T)$ as the unique decomposable graph represented by $T$. This graph is constructed by connecting vertices within cliques of $T$, namely by adding an edge $(v, u)$ to $G$ for every $v, u \in C$, with $C \in T$. We denote $\text{deg}(C, T) = |\text{nei}(C, T)|$ as the number of cliques adjacent to $C$ in $T$. These notations are applicable to $J$ as well.

### 3 Decomposable graphs from random-walks on trees

In this section, we introduce an algorithm that generates decomposable graphs from random walks on trees, beginning with a basic hierarchical model. Let $\pi$ be a probability measure on the space of the specified random variable. Generate an arbitrary tree skeleton $t$, and conditional on $t$, sample a junction tree $T$ in the hierarchical scheme

$$t \sim \pi(t), \quad T \mid t \sim \pi(T \mid t),$$

while ensuring that the sampling of $T$ adheres to the junction property (2.2). One such sampling method initiates $p$ random walks on a tree $t$ of $p$ vertices. Each walk starts
at an arbitrary vertex and continues exploring \( t \), visiting new vertices with a probability \( \rho \in (0, 1) \). Each walk forms a connected subtree \( t_i \) of \( t \). When combined, these \( p \) subtrees constitute a junction tree of a \( p \)-vertex decomposable graph. We detail this sampling process in Algorithm 1.

**Input:** An arbitrary tree \( t \) with \( \{1, \ldots, p\} \) vertices. \( \rho \in (0, 1) \).

1. for \( i \leftarrow 1 \) to \( p \) do
2.   draw \( j \sim \text{Uniform}\{1, \ldots, p\} \);
3.   \( w = \{j\} \); // visited vertices
4.   \( a = \{} \); // attempted vertices
5.   while \( \text{nei}(t, w) \setminus \{w \cup a\} \neq \emptyset \) do
6.     \( k \sim \text{Uniform}(\text{nei}(t, w) \setminus \{w \cup a\}) \);
7.     if \( \text{Uniform}([0, 1]) \leq \rho \) then add \( k \) to \( w \) else add \( k \) to \( a \);
8.   \( t_i \leftarrow \) a copy of \( t \), where the label of each vertex \( k \in w \) is set to \( i \).
9. return \( \bigcup_{i \leq n} t_i \) // union over labels.

**Algorithm 1:** Random walks on a tree.

Figure 3 showcases an example from Algorithm 1 with four random walks independently initiated on a skeleton \( t \), with \( \rho = 1/2 \). The resulting junction tree \( T \), the reduced junction tree \( J = \text{reduce}(T) \), and the corresponding decomposable graph are illustrated in Fig. 2.

![Diagram](image)

Figure 3: Example of a 4-vertex skeleton tree \( t \) and 4 random walks \( \{t_i\}, i = 1, \ldots, 4 \), generated via Algorithm 1. Visited vertices are labeled by the walk’s count \( i \) with solid-edge subtrees. The resulting \( T, J = \text{reduce}(T) \) and decomposable graph \( G \) are in Fig. 2.

The parameter \( \rho \) in Algorithm 1 (line 7) controls the sparsity of the generated subtrees. Lower values of \( \rho \) result in smaller subtrees, while higher values tend to produce more saturated subtrees. This parameter is utilized as a prior in SM Section E.

From the junction property (2.2), we understand that for a given junction tree \( T \) and a graph vertex \( v \in \mathcal{V} \), \( T_v \) is a connected subtree. In fact, if we remove all vertices in \( T_v \) except for \( v \), the resulting structure is the subtree skeleton \( t_v \), since \( t_v = T_v \setminus \{\mathcal{V} \setminus \{v\}\} \). Therefore, updating \( t_v \) as described in Algorithm 1 ensures that \( T \) maintains its status as a junction tree, as is substantiated in the following theorem.

**Theorem 3.1.** A tree \( T \) generated by Algorithm 1 is a junction tree.

**Proof.** By construction, every \( C, C' \in T \) are connected via a unique path \( C \sim C' \subseteq T \). If \( v \in C \cap C' \), then \( C, C' \in T_v \), and so is every clique on the path \( C \sim C' \). \( \square \)

Algorithm 1 is both simple and insightful in demonstrating how decomposable graphs can be constructed. It highlights three essential properties that underpin our work: (i) a junction tree can be updated solely through vertex-induced subtrees \( \{T_v\} \) for each \( v \in \mathcal{V} \).
\( V \), particularly when conditional independence is not the primary focus; (ii) a random decomposable graph can be constructed from the union of independently generated subtrees \( \{ t_i \} \) as shown in Fig. 3, useful when considering conditional independence; (iii) each subtree \( T_v \) can be expanded and contracted in multiple directions simultaneously by manipulating \( t_v \). Specifically, (iiia) expanding \( T_v \) involves enlarging the \( v \)th random-walk \( t_v \) in parallel to adjacent vertices in \( t \) (lines 5-7 in Alg. 1), and (iiib) contracting \( T_v \) is effectively a reversal of its expansion, achieved by reducing \( t_v \) from its leaf nodes.

Observations (i) – (iii) form the cornerstone of our analysis, facilitating a parallelized approach for manipulating junction trees through their junction property. The forthcoming section establishes a connection between the expansion and contraction methods detailed in (iiia) – (iiib) and updates to decomposable graphs.

4 Decomposable graph updates

In this section, we delineate how the random-walk process, as outlined in Algorithm 1, leads to the effective partitioning of a junction tree. This partitioning streamlines the process of updating a decomposable graph and enables us to establish the precise conditions necessary to preserve decomposability under multi-edge perturbations.

For a given junction tree \( T = (C, S) \) and graph vertex \( v \in V \), we partition \( C \) into three distinct clique sets: cliques that \( v \) can be added (connect) to, referred to as \( \mathcal{N}_v \), cliques that \( v \) can be removed from (disconnect), referred to as \( \mathcal{L}_v \), and everything else. Graphically, \( \mathcal{N}_v \) consists of cliques in \( T \) that are adjacent to \( T_v \), and \( \mathcal{L}_v \) are the leaf cliques of \( T_v \), as

\[
\mathcal{N}_v(T) := \{ C \in T : (C, C') \in T, C' \in T_v, C \notin T_v \},
\]

\[
\mathcal{L}_v(T) := \{ C \in T_v : \deg(C, T_v) = 1 \}. \tag{4.1}
\]

The terms add and remove are used when referring to updates on \( T \), and (dis)connect when referring to updates on the underlying graph \( G \). In \( T \), adding vertex \( v \) to an adjacent clique \( C \in \mathcal{N}_v \) implies updating \( C \) to \( C \cup \{ v \} \). Removing \( v \) from a leaf clique \( C \in \mathcal{L}_v \) implies updating \( C \) to \( C \setminus \{ v \} \). While we primarily work on \( T \), in \( G = g(T) \), these operations correspond to the following: Adding \( v \) to \( C \) entails connecting edges \( (v, u) \) in \( G \), where \( u \in C \setminus C_{adj} \); here, \( C_{adj} \) is the unique clique in \( T_v \) adjacent to \( C \) (i.e., \( (C, C_{adj}) \in T_v \)). Removing \( v \) from a clique \( C \in \mathcal{L}_v \) involves disconnecting all graph edges \( (v, u) \) from \( u \in C \setminus C_{adj} \), where \( (C, C_{adj}) \in T_v \). In both cases, there is a unique clique \( C_{adj} \in T_v \) adjacent to the relevant clique. Example 4.1 illustrates these sets graphically.

![Figure 4: 12-vertex decomposable graph of 8 maximal cliques](image)

**Example 4.1.** Figure 4 illustrates a 12-vertex, 8-maximal-clique decomposable graph \( \{ C_1, \ldots, C_8 \} \), with its junction tree \( T \) shown in Fig. 5 (left). The tree \( T \) includes one clique without any graph vertices, represented as \( \emptyset \), and two non-empty non-maximal cliques labeled as \( (8, 7) \) and 3 to indicate the vertices they contain. The induction of \( T \) to vertex
6 ($T_0$), depicted in Fig. 5 (right), consists of 3 cliques \{C_2, C_3, C_4\} connected with solid edges. The set \(N_6(T)\) is highlighted in blue and \(L_6(T)\) in green.

Clique in \(N_6\) and \(L_6\) are nonadjacent in \(T\), except \(C_4 \in L_6\), which is adjacent to \(\emptyset, C_5, C_8 \in N_6\). Removing 6 from \(C_4\) would hence remove \(\emptyset, C_5\) and \(C_8\) from \(N_6\). Similarly, adding 6 to any of \(\emptyset, C_5\) or \(C_8\), would remove \(C_4\) from \(L_6\). Therefore, we term a clique \(C \in N_6 \cup L_6\) a partition-divisor if it is adjacent to other cliques in those sets. A partition-divisor occurs only when a leaf clique \(C \in L_v\) is adjacent to a clique in \(N_v\).

As implicitly mentioned above, the sets \(N_v\) and \(L_v\) exhibit a congruence relation across updates, where leaf cliques become neighboring cliques, and vice versa, if updated. In Example 4.1, adding vertex 6 to the neighboring clique \(C_5 \in N_6(T)\) to form the clique \(C'_5 = C_5 \cup \{6\}\) in a new junction \(T'\) renders \(C'_5 \in L_6(T')\). Similarly, if 6 is removed from \(C_4\), the new \(C'_4\) becomes a neighboring clique to \(T'_0\). This congruence relation, demonstrated in (4.2), will come in handy in carrying graph updates across a Markov chain, as shown in subsequent sections.

\[
C \in N_v \iff C \cup \{v\} \in L_v. \tag{4.2}
\]

It is evident from Fig. 5 that \(L_v\) and \(N_v\) are both vertex-wise symmetric in relation to graph updates. Specifically, for an unconnected vertex pair \(v, u \in V, u \in C\) for some \(C \in N_v\) if and only if \(v \in C'\) for some \(C' \in N_u\). If \(v\) and \(u\) are in a single clique of \(G\), then \(u \in C\) for some \(C \in L_v\) if and only if \(v \in C'\) for some \(C' \in L_u\). These findings align with the single-edge decomposability-preserving conditions illustrated in Frydenberg and Lauritzen (1989) and Giudici and Green (1999).

Extending this concept to a multi-vertex set \(V \subseteq V\), where \(V\) forms a clique in \(G\), is straightforward by utilizing \(T_V\) in (4.1). For enhanced readability, the partition sets in (4.1) are primarily described for singleton vertices.

Having analytically and visually demonstrated the class of partitions generated by (4.1), we now present theorems that illustrate the necessary and sufficient conditions for a perturbation in the edge set of a decomposable graph to maintain decomposability.

**Theorem 4.2.** Let \(G = (V, E)\) be a decomposable graph, and let \(V, U\) be two disjoint subsets of \(V\), both forming complete graphs in \(G\). Suppose we form a graph \(G' = (V, E')\) by adding edges from every vertex in \(V\) to every vertex in \(U\). Then, \(G'\) is decomposable if and only if \(U\) can be partitioned into mutually exclusive subsets of vertices ordered as \(\{U^{(1)}, \ldots, U^{(K)}\}\), where \(U^{(i)} \cap U^{(j)} = \emptyset\) if \(i \neq j\), such that, there exists a path of maximal
cliques \((C_1, \ldots, C_K)\) in some \(J \in \mathcal{J}^G\) satisfying the recursive relation that \(U^{(1)} \subseteq C_1\) for \(C_1 \in \mathcal{N}_V(J)\), and \(U^{(k)} \subseteq C_k\) for \(C_k \in \mathcal{N}_{U^{(k-1)}}(J)\) where \(k = 2, \ldots, K\).

**Theorem 4.3.** Let \(G = (V, \mathcal{E})\) be a decomposable graph, and let \(V, U\) be two disjoint subsets of \(V\) that are completely connected, i.e. \(V \cup U\) is complete in \(G\). Suppose we form a graph \(G' = (V, \mathcal{E}')\) by disconnecting all edges \((v, u)\), where \(v \in V\) and \(u \in U\). Then, \(G' = (V, \mathcal{E}')\) is decomposable if and only if \(U\) can be partitioned into mutually exclusive subsets of vertices ordered as \((U^{(1)}, \ldots, U^{(K)})\), where \(U^{(i)} \cap U^{(j)} = \emptyset\) if \(i \neq j\), such that, there exists a path of maximal cliques \((C_1, \ldots, C_K)\) in some \(J \in \mathcal{J}^G\) satisfying the recursive relation that \(U^{(1)} \subseteq C_1\) for \(C_1 \in \mathcal{L}_V(J)\), and \(U^{(k)} \subseteq C_k\) for \(C_k \in \mathcal{L}_{U^{(k-1)}}(J)\) where \(k = 2, \ldots, K\).

A special case in Theorems 4.2 and 4.3 occurs when the entire vertex set \(U\) is a subset of \(C_1\), implying \(U = U^{(1)}\) and the sets \(\{U^{(k)}\}\) are empty for \(k = 2, \ldots, K\). Consequently, \(C_1\) is identified as a neighboring clique \((C_1 \in \mathcal{N}_v(J))\) in Theorem 4.2, or as a leaf clique \((C_1 \in \mathcal{L}_v(J))\) in Theorem 4.3. When the sets \(\{U^{(k)}\}\) for \(k = 1, \ldots, K\) are non-empty, it indicates that \(U\) spans multiple adjacent cliques in \(J\), forming a path. Metaphorically, a disconnect move can be likened to peeling a bandage, where \(V\) is sequentially disconnected from \(U^{(1)}\), then \(U^{(2)}\), and so on. In this context, \(U^{(k)}\) is always part of a leaf clique in the junction tree created by disconnecting \(V\) from \(\bigcup_{j<k} U^{(j)}\). Conversely, a connect move is akin to a reverse operation, where \(U^{(k)}\) is always in a neighboring clique to the induced junction tree \(J_v\), formed by connecting \(V\) to \(\bigcup_{j<k} U^{(j)}\). These processes are consistent with the junction property (2.2).

The main consequence of Theorems 4.2 and 4.3 is that they provide a comprehensive enumeration of all decomposability-preserving updates related to a vertex, as defined by the partition sets in (4.1). Although Theorems 4.2 and 4.3 are framed in the context of graph updates, pertinent to graphical models, analogous principles for updates on general junction trees can be directly derived from the junction property (2.2). The next section introduces a single-move sampler and extends it to a parallel sampler over junction trees.

## 5 Single-move sampler

### 5.1 Proposal probability

This section introduces a single-move sampler based on the partition sets in (4.1), highlighting its key properties. For a given state of the chain \(T\), the update of clique \(C\) to \(C'\), resulting in the new state \(T'\), occurs with a probability

\[
q(T, T') = \frac{1}{2} \frac{1}{|V|} \frac{1}{|\mathcal{P}_v(T)|} \mathbb{1}\{C \in \mathcal{P}_v(T)\}.
\]  

(5.1)

The ratio \(1/|\mathcal{P}_v(T)|\) accounts for the probability of selecting clique \(C\) from the set \(\mathcal{P}_v\), where \(\mathcal{P}_v = \mathcal{N}_v\) applies to an addition, and \(\mathcal{P}_v = \mathcal{L}_v\) to a removal move. The factor of \(1/2\) accounts for the choice between update types, and \(1/|V|\) for the selection of a vertex.

For the addition move, the reverse operation involves enumerating \(\mathcal{L}_v(T')\) of the new junction tree \(T'\), which can be computed from \(\mathcal{L}_v(T)\) due to the congruence relation between the sets defined in (4.1) and (4.2). If \(v\) is added to \(C \in \mathcal{N}_v(T)\) to form \(C'\), then \(C'\) becomes a leaf clique in \(\mathcal{L}_v(T')\). The status of other cliques in \(\mathcal{L}_v(T)\) remains unchanged, except when the clique adjacent to \(C\) in \(T_v\) is a partition-divisor. In Example 4.1, adding 6 to \(C_1\) increases the number of leaves in \(T_6\) by one, as \(C_2\) is not a partition-divisor. However,
adding 6 to $C_5$ or any neighbor of the partition-divisor $C_4$ would result in $C_4$ losing its status as a leaf clique in $T'_6$, thereby making $\mathcal{L}_6(T') = \mathcal{L}_6(T)$. Let $C_{\text{adj}}$ be the unique clique in $T_v$ adjacent to $C$, $(C_{\text{adj}}, C) \in T$, then

$$|\mathcal{L}_v(T')| = |\mathcal{L}_v(T)| + 1\{C_{\text{adj}} \notin \mathcal{L}_v(T)\}. \quad (5.2)$$

For the removal move, the reverse operation involves enumerating $\mathcal{N}_v(T')$, which can similarly be computed from $\mathcal{N}_v(T)$. If $v$ is removed from $C$, and $C$ is a partition-divisor, all neighbors of $C$ in $\mathcal{N}_v(T)$ lose their status as neighboring cliques to $T_v$. For instance, if 6 is removed from $C_4$ (Fig. 5), all neighbors (in blue) of $T'_6$ ($\{\emptyset, C_5, C_8\}$) are not in $\mathcal{L}_6(T')$, and $C_4$ would become a leaf node of $T'_6$. Otherwise, $|\mathcal{N}_v(T')| = |\mathcal{N}_v(T)| + 1$. Resulting in

$$|\mathcal{N}_v(T')| = |\mathcal{N}_v(T)| + 2 - \text{deg}(C, T). \quad (5.3)$$

In all cases, quantifying the reverse operation ratio $q(T', T)$ requires only the enumeration of the sets $\mathcal{L}_v(T)$ and $\mathcal{N}_v(T)$. This enumeration can be efficiently carried out with a single pass over $T_v$ and its neighbors, without necessitating an actual modification to

### 5.2 Prior and posterior probabilities

Following (3.1), we propose a hierarchical sampler that iteratively samples a junction tree $T$ and the underlying skeleton $t$ from the posteriors $\pi(T \mid t)$ and $\pi(t \mid T)$, respectively. Assuming a skeleton tree with $p$ vertices (as a decomposable graph with $p$ vertices has at most $p$ maximal cliques), Cayley’s theorem from Cayley (1889) tells us there are $p^{p-2}$ possible trees of size $p$. Thus, a uniform prior on $t$ would be $\pi(t) \propto p^{-(p-2)}$.

Turning to prior specification over the space of junction trees. Byrne et al. (2015); Green and Thomas (2018) have illustrated an important family of graph laws through algebraic characterization, known as the \textit{clique-separator} factorization laws. Laws in this family have densities $\pi(T)$ that factorize as

$$\pi(T) \propto \prod_{C \in \mathcal{C}} \phi(C) \prod_{S \in \mathcal{S}} \psi(S), \quad (5.4)$$

Over the class of decomposable graphs, and consequently over the class of junction trees, this applies for positive functions $\phi, \psi$. As noted by Green and Thomas (2018, Thm. 1), a graph law over the class of decomposable graphs, which encompasses all such graphs, is weakly structural Markov if and only if it adheres to a clique-separator factorization law. Specifying decomposability through junction trees does not modify the clique-separator factorization of the joint distribution in (1.1), as elucidated in the following proposition.

**Proposition 5.1.** For a random vector $Y$ that has a decomposable conditional independence graph $G$, where $T = (\mathcal{C}, \mathcal{S})$ is a junction tree of $G$ having $\mathcal{c}I(G) \subseteq \mathcal{C}$ and $\mathcal{sep}(G) \subseteq \mathcal{S}$, the distribution of $Y$ factorizes as

$$p(Y) = \prod_{C \in \mathcal{C}} p(Y_C) \prod_{S \in \mathcal{S}} p(Y_S) = \left[ \prod_{C \in \mathcal{c}I(G)} p(Y_C) \prod_{S \in \mathcal{sep}(G)} p(Y_S) \right]. \quad (5.5)$$

**Proof.** Let $C \in \mathcal{C}$ be any non-maximal clique in $T$, i.e., there exists a $(C', C) \in \mathcal{S}$ such that $C \subseteq C'$. The terms in (5.5) that are specific to $C$ and $C'$ are

$$\frac{p(Y_C) p(Y_{C'})}{p(Y_{C \cap C'})} = \frac{p(Y_C) p(Y_{C'})}{p(Y_C)} = p(Y_{C'}).$$

Carrying such cancellations over non-maximal cliques leads to the last term in (5.5). \qed
Following a similar argument to Proposition 5.1, the density $\pi(T)$ in (5.4) factorizes in terms of $(c1(G), \text{sep}(G))$, where $G = g(T)$, if $\psi = \phi$, as

$\pi(T) \propto \prod_{C \in C} \phi(C) \prod_{S \in S} \psi(S) = \prod_{C \in c1(G)} \phi(C) \prod_{S \in \text{sep}(G)} \psi(S)$,

(5.6)

otherwise, a multiplicative factor of $\prod_{C \in (C \setminus c1(G))} \phi(C)/\prod_{S \in (S \setminus \text{sep}(G))} \psi(S)$ exists on the right-hand side of (5.6).

Turning to posterior specification, we have $\pi(t \mid T) \propto 1$, as the likelihood and prior terms cancel out, leaving a symmetric proposal, which follows exactly as the tree randomization step performed in Green and Thomas (2013), detailed in (Thomas and Green, 2009, Sec. 5). For a posterior sample from $\pi(T \mid t)$, we employ a standard Metropolis–Hastings step as outlined by (Hastings, 1970). Given a junction tree $T$, we first uniformly sample a vertex $v \in V$, then compute the partitions $L_v(T)$ and $N_v(T)$. Subsequently, we uniformly select an update-type and a partition $C$ for the update. Let $C_{adj}$ be the unique clique in $T_v$ adjacent to $C$ (i.e., $(C_{adj}, C) \in T$). The acceptance probability of the proposed update of $C$ to $C'$ and the new junction tree $T'$ is

$\alpha(T, T' \mid t) = \min \left\{ 1, \frac{\pi(T' \mid t)q(T, T')}{\pi(T \mid t)q(T', T)} \right\}$

$= \min \left\{ 1, \frac{p(Y_{C_{adj}})p(Y_{C \cap C_{adj}}) \phi(C') \psi(C \cap C_{adj}) | P_v(T')}{p(Y_{C' \cap C_{adj}})p(Y_C) \psi(C' \cap C_{adj}) \phi(C) | P_v(T)} \right\}$,

(5.7)

if $C \in P_v(T)$, and zero otherwise. If the update-type is an addition, $P_v(T) = N_v(T)$ and $P_v(T') = L_v(T')$. Conversely, if the update-type is a removal, $P_v(T) = L_v(T)$ and $P_v(T') = N_v(T')$. Both quantities can be computed using (5.2) and (5.3). In (5.7), it is sufficient to use only the indicator function associated with $q(T, T')$ because the validity of the reverse operation depends on the congruence relation (4.2), which in turn is contingent on the initial move. Another consequence of this congruence relation is that the step in (5.7) is reversible across the chain states. Furthermore, given that $\pi(T \mid t) > 0$, there is always a sequence of junction trees from any state of the chain to the single-clique junction tree. The finiteness of the chain’s state space implies irreducibility, thus ensuring the ergodicity of the chain.

Computing the acceptance probability as specified in (5.7), while not necessitating an actual modification of $T$, does require the enumeration of the two state-dependent sets defined in (4.1). Section 6 introduces a parallel strategy for sampling over junction trees, with localized proposal that bypass the need to consider the entire state $T$. Consequently, the sampler only needs to enumerate a single partition set for each proposal, greatly streamlining the sampling process.

6 Parallel sampler

6.1 Local proposal probability

In the single-move sampler (Sec. 5), a transition from state $T$ to $T'$ involves updating the clique $C$ to $C'$. In this process, only $C$ is modified, while all other cliques in $T'$ remain unchanged from their state in $T$. This aspect is captured in the posterior ratio in (5.7), which depends only on the marginals of $C, C'$, and $C_{adj}$. It is the proposal probability in (5.1) that ties the acceptance ratio (5.7) to the overall state of $T$. This connection is
specifically through the set \( P_v(T) \), which determines (i) the likelihood of selecting \( C \) from \( P_v(T) \), and (ii) the validity of the move, denoted as \( 1 \{ C \in P_v(T) \} \).

To develop a parallel sampler, it is essential that all proposals within a partition be mutually exclusive. Specifically, updating a clique in \( P_v(T) \) should not alter the state or validity of any other clique within the same set. Under this condition, it becomes feasible to update all cliques in \( P_v(T) \) concurrently, this is formalized in the following proposition.

**Proposition 6.1.** Let \( T = (C, S) \) be a junction tree over the vertex set \( V \). For \( v \in V \), define \( N_v(T) \) and \( L_v(T) \) as in (4.1). For any distinct cliques \( C_1, C_2 \in T \):

(I) If \( C_1, C_2 \in N_v(T) \), when \( v \) is added to \( C_1 \) to arrive at state \( T_1 \), then \( C_2 \in N_v(T_1) \).

(II) If \( C_1, C_2 \in L_v(T) \), and \( \{|C : C \in T_v| > 2 \} \), when \( v \) is removed from \( C_1 \) to arrive at state \( T_1 \), then \( C_2 \in L_v(T_1) \).

Proof. The proof follows by construction. By (4.1), in both cases, there exists a clique \( C_{adj} \in T_v \) such that \((C_2, C_{adj}) \in T \) and \( C_{adj} \neq C_1 \). In (I), adding \( v \) to \( C_1 \), neither alters \( C_{adj} \) nor the edge \((C_2, C_{adj}) \implies (C_2, C_{adj}) \in T_1 \implies C_2 \in N_v(T_1) \). In (II), removing \( v \) from \( C_1 \) does not affect the edge \((C_2, C_{adj}) \implies C_2 \in L_v(T_1) \). \( \square \)

Under the conditions of Proposition 6.1, consider a sequence of updates \( C_1, \ldots, C_K \in P_v(T) \), starting at state \( T \) and resulting in states \( T_1, \ldots, T_K \). The validity of the \( K \)th update can be established by induction as

\[
\{C_k \in P_v(T)\} \iff \{C_k \in P_v(T_{k-1})\}.
\]

Therefore, the execution order does not affect the validity of updates in \( P_v(T) \), as long as these updates are valid in the initial state \( T \). To visualize this, all (blue) cliques in \( N_6 \) (Fig. 5) can be updated simultaneously, as their validity relies on cliques in \( T_6 \) (the initial state). In this case, \( T \) can be transformed into one of \( 2^5 \) possible states (since \( |N_6| = 5 \)). The choice among these potential states is made independently based on the acceptance probability of each proposal. Similarly, the (green) cliques in \( L_6 \) can also be updated concurrently, as their validity is contingent upon \( C_2 \), a non-leaf clique of \( T_6 \).

Proposition 6.1 (II) highlights a scenario where proposals in \( P_v(T) \) are not mutually independent. Initially, if \( \{|C : C \in T_v| < 2 \} \), indicating that \( T_v \) is a single-clique tree, it follows that \( L_v(T) = \emptyset \). Secondly, if \( \{|C : C \in T_v| = 2 \} \), meaning \( T_v \) comprises only two cliques, then removing \( v \) from \( C_1 \) results in \( C_2 \not\in L_v(T_1) \), as \( C_2 \) now has degree 0 in \( T_v \). Given the limitation of this scenario to just two possible moves, we revert to using the single-move sampler. This is achieved by adjusting the proposal mechanism in the parallel sampler. Specifically, at state \( T \), the proposal probability for updating \( C_i \in P_v(T) \) to \( C_i' \) is

\[
q(C_i, C_i'; T) = \frac{1}{2} \frac{1}{|V|} \frac{1}{2^{U(C_i, C_{adj(i)})}} \{C_i \in P_v(T)\},
\]

\[
U(C_i, C_{adj(i)}) = 1 \{P_v(T) = \{C_i, C_{adj(i)}\}\}.
\]

Here, \( C_{adj(i)} \in T_v \) is the unique clique adjacent to \( C_i \), such that \((C_{adj(i)}, C_i) \in T \). The condition \( U(C_i, C_{adj(i)}) = 1 \) corresponds to the scenario where \( \{|C : C \in T_v| = 1 \} \), and the move is a removal, namely, \( P_v(T) = L_v(T) = \{C_i, C_{adj(i)}\} \). Consequently, the factor \( 2^{-U(C_i, C_{adj(i)})} \) accounts for the selection choice \( 1/|L_v(T)| \) when \( \{|C : C \in T_v| = 2 \} \), and the single-move proposal is employed. In all other cases, (6.1) does not incorporate the selection probability \( 1/|P_v(T)| \), as specified in (5.1), since all proposals are executed concurrently. By the congruence relation (4.2), the validity of the reverse operation hinges
on the event \( \{ C_i \in P_v(T) \} \). Resulting in a proposal ratio of 
\[ q(C_i, C'_i; T) / q(C'_i, C_i; T') = 2^{U(C'_i, C_{adj(i)}) - U(C_i, C_{adj(i)})} \mathbb{1} \{ C_i \in P_v(T) \}. \]

By eliminating the need to decide which clique to update, the parallel sampler only requires the enumeration of a single partition set, either \( N_v \) or \( L_v \), based on the chosen update-type. In contrast, the single-move sampler requires the enumeration of both sets.

### 6.2 Prior and posterior probabilities

In this section, we construct a parallel sampler over the space of junction trees. We adhere to the iterative sampling method and prior setup used in the single-move sampler (Sec. 5), employing a parallelized Metropolis–Hastings step, as per (Hastings, 1970), to sample from the posterior \( \pi(T \mid t) \). Our sampler is detailed in Algorithm 2.

**Input:** Initiate an arbitrary tree \( t \) with \( p = |V| \) vertices, and set \( T = t \). For \( M \) steps, let \( N < M \) be the frequency at which \( t \) is updated.

1. **for** \( k \leftarrow 1 \) **to** \( M \) **do**
2.   **draw** \( v \sim \text{Uniform} \{1, \ldots, p\} \), choose an update-type uniformly;
3.   **do in parallel:**
4.     **update** every \( C_i \in P_v(T) \) to \( C'_i \), and tree \( T'_i \), with probability
5.     \[ \alpha_i(C_i, C'_i \mid t) = \min \left\{ 1, \frac{\pi(C'_i, C' \mid t) q(C_i, C'_i; T)}{\pi(C_i, C \mid t) q(C'_i, C; T')} \right\} \]
6.     \[ = \min \left\{ 1, \left( \frac{p(C'_i T') p(C_{adj(i)} \cap C'_i) \phi(C'_i) \psi(C_{adj(i)} \cap C'_i) \phi(C'_i) 2^{U(C'_i, C_{adj(i)})}}{p(C_i T) p(C_{adj(i)} \cap C_i) \phi(C_i) \psi(C_{adj(i)} \cap C_i) \phi(C_i) 2^{U(C_i, C_{adj(i)})}} \right) \right\}. \] (6.2)
7.   **let** \( T' \) **be** the resulting junction tree, and set \( T = T' \);
8. **if** \( k \mod N = 0 \) **then** **draw** \( t' \sim \pi(t \mid T) \propto 1 \);

**Algorithm 2:** Parallel sampler over junction trees.

In (6.2), \( \phi \) and \( \psi \) are as defined in (5.4), \( q(C_i, C'_i; T) \) is as specified in (6.1), and \( C_{adj(i)} \in T_v \) is the unique clique adjacent to \( C_i \), such that \( (C_{adj(i)}, C_i) \in T \). Although the first equality in (6.2) is defined in terms of \( T \), the final ratio only involves terms associated with \( C_i \), its update \( C'_i \), and its neighbor \( C_{adj(i)} \), since here \( C'_{adj(i)} = C_{adj(i)} \) by construction. As a consequence, all updates are localized to the clique of interest and its neighbors in \( T \). Moreover, since we do not update the skeleton \( t \) after every update, all updates in \( P_v(T) \) are carried out simultaneously. The proposed parallel sampling scheme is straightforward to implement in practice. Some computational considerations are discussed in Section 6.3.

The chain governed by (6.2) is irreducible and aperiodic over a finite state space, since it is possible to reach the single-clique junction tree with a finite number of steps, from any state, and backward, and return to the single-clique junction tree in any number of steps. Moreover, \( \pi(T_i \mid t) > 0 \). Hence, a unique stationary distribution exists. However, the chain is not reversible, as it only satisfies the partial detail balance equations (Whittle, 1985).

Conditional on the skeleton \( t \), the marginal state space of clique \( C_i \) is reversible by the congruence relation in (4.2) across chain states. This relation entails that leaf cliques become neighboring cliques, and vice versa, upon being updated. Consequently, the congruence relation indicates that all updated cliques in \( P_v(T) \) are now included in \( P_v(T') \). However, the latter set may contain additional cliques. If all cliques in \( P_v(T') \) were updated in a reverse move, the resulting state \( T'' \) might not revert to the original state \( T \). Therefore,
reversibility is partially ensured within the set $P_v(T)$, or when \( \{ C \cup \{ v \} : C \in P_v(T) \} = P_v(T') \) in an addition move and \( \{ C \setminus \{ v \} : C \in P_v(T) \} = P_v(T') \) in a removal move.

6.3 Computational considerations

Our proposed single-move and parallel samplers are based on the junction property of trees, as discussed in Section 2. In practice, \( T = (\mathcal{C}, \mathcal{S}) \) can be larger than \textbf{reduce}(\( T \)) = \((c\mathcal{l}(G), \text{sep}(G))\), where \( G = g(T) \). Two factors influence the number of non-empty cliques in \( \mathcal{C} \), the size of the underlying skeleton \( t \), and the prior over \( T \), as specified in (5.4).

The proposed samplers can be initialized with a skeleton \( t \) of an arbitrary number of vertices \( p \). This setup can lead to \( \mathcal{C} \) having up to \( p - 1 \) non-empty cliques, as a complete graph is a single clique. However, a very large \( p \) can result in a slowdown in convergence due to potentially redundant updates in the graph space. Conversely, a conservatively chosen \( p \) might lead to a limited number of accepted updates. We suggest initializing \( t \) with \( p = |V| \) vertices, as the no-edge graph yields the maximum number of cliques over the vertex set.

Prior specification plays a pivotal role in controlling the number of non-empty cliques in \( \mathcal{C} \). Very diffuse priors lead to \( \mathcal{C} \) being saturated with many non-empty and non-maximal cliques. In contrast, tighter priors are more conservative in proposing updates, effectively reducing the size of \( \mathcal{C} \) throughout the chain iterations, even with a large \( t \). This trade-off is formalized in the following example. Consider the \( i \)th acceptance ratio in (6.2).

Proposition 5.1 allows us to factor out likelihood ratios corresponding to non-maximal cliques, effectively computing the likelihood ratio as if over \textbf{reduce}(\( T \)) = \((c\mathcal{l}(G), \text{sep}(G))\). If \( \phi = \psi \), the prior ratio cancels all terms associated with non-maximal cliques. Therefore, for our discussion, we assume that \( \phi \neq \psi \).

If \( C \in \mathcal{C} \) is a non-maximal clique, then \( C \) is contained in one of its neighbors in \( T \), say \( C_{\text{adj}} \). Consequently, \( C \)'s contribution to the prior ratio in the acceptance probability simplifies to the multiplicative factor \( \phi(C)/\psi(C_{\text{adj}} \cap C) = \phi(C)/\psi(C) \). Updating \( C \) to the proposed \( C' \) can alter its maximal status, such as \( C' \) being maximal while \( C \) is non-maximal, and vice versa. This scenario leads to four cases, (a)-(d), as outlined in Table 1. Each case in this table denotes the contribution of the non-maximal clique to the acceptance ratio in the form of a multiplicative factor. For instance, in (a), when both \( C \) and \( C' \) are maximal cliques, the non-maximal clique’s contribution to the acceptance probability ratio is 1. In (b), if \( C \) is maximal while \( C' \) is non-maximal, aside from \( C \)'s contribution, \( C' \) contributes with the prior ratio \( \psi(C')/\phi(C') \). In the special case (d), the acceptance probability is entirely influenced by the prior ratio, as no other clique in \( T \) undergoes a change in status.

| \( C \) maximal | \( C' \) maximal (a) : 1 | \( C' \) non-maximal (b) : \( \phi(C')/\psi(C) \) |
| \( C \) non-maximal | \( \psi(C)/\phi(C) \) (c) : \( \phi(C)/\psi(C) \) |

Adjusting the contribution factors in Table 1 can effectively control the number of non-maximal cliques in \( \mathcal{C} \). For instance, a tight prior that penalizes separators, by setting \( \psi(C) > \phi(C) \), can favor proposals involving maximal cliques. In such a configuration, the likelihood of accepting a non-maximal proposal in case (b) is reduced, while acceptance of the maximal proposal in case (a) is facilitated. The outcome in case (d) is contingent upon the specific prior used. For instance, under the clique exponential family priors (Bornn and
Caron, 2011; Green and Thomas, 2018), where $\phi(C) = \exp(\alpha|C|)$ and $\psi(S) = \exp(\beta|S|)$ for constants $\alpha, \beta > 0$, the ratio $\phi(C)/\psi(C)$ becomes $\exp\{|C|(|\alpha - \beta|)\}$. Thus, in case (d), the factor translates to $\exp\{\xi(\alpha - \beta)\}$, where $\xi = 1$ if $|C'| > |C|$ and $-1$ otherwise. If $\beta > \alpha$, case (d) then tends to favor smaller non-maximal cliques. This phenomenon is explored numerically in SM Section F.

7 Numerical performance of the new samplers

Our samplers, designed to target a posterior over junction trees with $p$ vertices ($\pi(T \mid Y, t)$), map this posterior onto decomposable graphs using the operator $g$. This mapping influences the frequency of graph samples, as more junction tree representations lead to higher sample frequency (Thomas and Green, 2009). Achieving uniform sampling across decomposable graphs necessitates a prior considering these representations, a strategy implemented by Green and Thomas (2013). In SM Section E, we demonstrate uniform sampling over reduced junction trees and approximate this uniformity across broader junction trees using Algorithm 1’s visit probability $\rho$. Despite computational challenges in enumerating these broader spaces, our approach effectively approximates uniform sampling across decomposable graph spaces, as shown for $p = 7$.

In the next subsections, we compare our samplers in the context of a Gaussian decomposable graphical model. The conjugate posterior of a Gaussian graphical model with a Wishart prior for the covariance matrix is derived in SM Section C, following the classic approaches of Dawid and Lauritzen (1993) and Giudici and Green (1999). For alternative setups, such as under a log-linear model, refer to works like Tarantola (2004).

7.1 Gaussian graphical model simulation setup

To illustrate the advantages of our methods, we consider the following graph setups: an autoregressive (AR) graph with lag varying between 1 and 5, and a random decomposable graph sampled following the Christmas tree algorithm (CTA) of Olsson et al. (2022) with parameters set to 0.5.

For each setup, we sample a graph $G$ and generate a dataset of 100 samples following a $N_p(0, \Theta^{-1})$ distribution, where $\Theta \sim$ inverse-Wishart($\delta, I_p|G$) and $I_p$ is the identity matrix. This process is repeated 10 times, setting $p = \{50, 200\}$ and $\delta = 3$, resulting in 10 unique datasets per dimensions $p$, each associated with a distinct conditional independence graph. Figure 6 illustrates an example of two sampled decomposable graphs $G$ alongside their adjacency matrices, for each graph setup and $p = 50$. Other simulated graphs are presented in SM Fig S16 and S17, for $p = 50$.

Figure 6: Simulated 50-vertex autoregressive (AR) and Christmas tree algorithm (CTA)-generated decomposable graphs, alongside their adjacency matrices.
To simulate real-world conditions and avoid bias from prior knowledge of the generative model, in the posterior sampling, we assign a hyper Wishart prior to $\Theta$. For each clique $C$, the degree of freedom is set at $\delta = 1$, with the $|C|$-dimensional identity matrix as the scale matrix. This choice of prior leads to a conjugate posterior, as elaborated in SM Section C. Unless stated otherwise, we employ a uniform prior over $T$, setting $\pi(T) \propto 1$.

The next sections compare our single-move sampler (Sec. 5) and parallel sampler (Sec. 6) with four existing Bayesian samplers for decomposable graphs: the multi-edge (GT13-m) and single-edge (GT13-s) junction tree samplers of Green and Thomas (2013), the single-edge (GG99) sampler of Giudici and Green (1999), and the particle Gibbs sequential Monte Carlo sampler (O19) by Olsson et al. (2019). In contrast to our junction-based sampler, the samplers by Green and Thomas (2013) are graph-based, meaning at every step they reduce $T$ to $J = \text{reduce}(T)$. Their GT13-m sampler updates an arbitrary number of edges in line with their decomposability-preserving proposals. This differs from our all-or-none sampling method, where $v$ is either added to or removed from a clique. Our single-move sampler effectively operates as a multi-edge sampler, except in cases involving only two vertices. This contrasts with GT13-s, which consistently selects two vertices at every step.

We execute the single move sampler for one million steps, and the parallel sampler for 500,000 steps when $p = 50$ and one million when $p = 200$. We initiate the samplers with a random $p$-vertex skeleton $t$ and set it as the junction tree $T$, which corresponds to the junction tree of a no-edge $p$-vertex graph. We sample a posterior of the skeleton $\pi(t|T)$ every 100 Metropolis–Hastings samples of $\pi(T|t)$, following the recommendation in Green and Thomas (2013). The sampling of $\pi(t|T)$ follows the randomization approach in Thomas and Green (2009).

We execute one million steps for all MH-Mc competing samplers when $p = 50$, and 30 million when $p = 200$. In the latter case, this large number of steps was required for convergence. We incorporate junction tree randomization in the samplers of Green and Thomas (2013) every 100 steps. While Green and Thomas (2013) defines this as junction tree randomization, we refer to it as a hierarchical sampling scheme, as outlined in (3.1). The Gibbs sampler of Olsson et al. (2019) is analytically more complex than a Metropolis–Hastings, and more computationally intensive. Therefore, we executed it for 10,000 steps, with each step comprising 50 particles when $p = 50$. This sampler is computationally unfeasible for $p = 200$. All competing samplers are assigned a uniform graph prior. Simulations were performed using the Benchpress framework (Rios et al., 2021) with code outlined in SM Section D.

Updates in both our parallel and single-move samplers are performed on the junction tree $T$, yet not every update on $T$ corresponds to a change in the underlying decomposable graph $G = g(T)$. To evaluate the performance of these samplers in the graph space, we process each chain post-sampling to generate a new chain with the graph $G$ as the state variable. Metrics calculated on this processed chain are termed as graph-updates metrics, distinguishing them from junction-updates metrics.

### 7.2 Simulation results for the general case

Our parallel and single-move chains have achieved convergence within the initial 50,000 steps (Fig. 7, left panel). The within-sample correlation concerning the number of edges in the underlying graph for each chain is almost negligible for these samplers. Specifically, the correlation drops below 0.2 for all parallel chains by lag 2500 and by lag 10000 for the single-move chains. They maintain a level close to zero subsequently (Fig. 8, left panel).

Overall, we observed consistently faster convergence to the chain’s stationary state,
less within-sample correlation and improved computation speed with the parallel sampler versus the single-move sampler. The computational speed improvement is attributable to parallelization; the parallel sampler executed approximately 1,500,000 total updates for 500,000 steps, achieving about 300% greater efficiency. Each update for the parallel sampler requires fewer computations than the single-move sampler, since the former does not require a proposal ratio computation as in (5.7).

Green and Thomas (2013)'s chains required approximately 600,000 steps to converge, with at least two multi-edge (GT13-m) chains failing to do so. The within-sample correlation for these samplers remains high, only dropping below 0.2 at a lag 100,000. Giudici and Green (1999)'s chains (G99) performed similarly to GT13-s. Olsson et al. (2019)'s chains (O19) showed improved within-sample correlation, falling below 0.2 at lag 1000, however,
thier run-time exceeded 4 hours, in contrast to the sub-3-minute for other samplers. The relative computational efficiency for 1000 steps is about 0.2 seconds for all samplers, except O19, which is approximately 1 hour. Simulations are run on an Intel Xeon E5-2698 v4 2.20GHz processor, for detailed analysis of computation time see SM Figure S12. Computational speed of our samplers scales linearly with $p$ (SM Fig. S22).

GT13-m median acceptance rate is $\approx 4\%$, while our parallel sampler exceeds 26% acceptance rate. The GT13-s aligns with our decomposability conditions, showing comparable rates to our single-move sampler and double that of the non-junction tree based GG99 sampler. Further demonstrating the advantage of junction-tree based samplers.

Alongside improved mixing properties, our samplers demonstrate higher accuracy than competing samplers. This is evidenced by the ROC curves in Figures 9 (right) and 1 (left), computed from the last 300,000 samples of each chain, where the true and false positive rates are computed and then averaged these across all replications.

Figure 9: (Left) Boxplot of acceptance rates over graph-updates comparing our samplers and competitors, across two graph structures and 10 replications. (Right) ROC curves comparing the same samplers under the autoregressive graph structure and $(n, p) = (100, 50)$.

Our samplers aim at a distinct distribution (Sec. 7). Nonetheless, empirically, our chains converged close to competing methods (Fig. S13). Analysis of the autoregressive graph structure show similar results to the Christmas tree algorithm, and thus differed to SM Section G, alongside more diagnostic plots on the simulations of this section.

7.3 Simulation results for the high-dimensional case

In the high-dimensional case ($p = 200$), our samplers demonstrated distinct convergence properties. While similar in acceptance rates and computational time to the general case ($p = 50$, Sec. 7.2), our samplers reached stationarity within the first 600,000 steps (SM Fig. S24), in 2–3 minutes. In contrast, some GT13-s and GG99 chains required around 20 million steps to converge, lasting about 40 minutes. Most GT13-m chains did not converge even after 30 million steps, exceeding an hour in computational time. This discrepancy is evident in the traceplots showing the number of graph edges (SM Fig. S25). The accuracy gains of our samplers far exceeded competing ones, where the gap is much larger than the case of $p = 50$, as shown by the ROC curves in Figure 1 (right) and SM Figure S26 (right). The ROC curves were computed from the last 500,000 samples of each chain. For additional diagnostic plots of the simulations in this section, please refer to SM Section H.
Computational bottlenecks still remain in the quest to design Markov chain samplers for high-dimensional decomposable graphs. Exploiting the junction tree representation led to the largest gain in sampling efficiency, as established in the work of Green and Thomas (2013); Giudici and Green (1999); Olsson et al. (2019, 2022), and herein. A greater part of this efficiency gain is driven mainly by an improvement in graph proposals that the junction property explicitly exposed. Nonetheless, the junction representation is not unique. Counting the number of junction tree representations for a fixed decomposable graph is known to be computationally expensive (Thomas and Green, 2009), which led Green and Thomas (2013) to propose a modified Metropolis–Hastings chain that is inferior to the classical one, to avoid such computation. Our proposed method trades the need to quantify the space of junction trees with a form of latent dimensional expansion coupled with hierarchical sampling of junction trees. Here, the latent expansion represents non-maximal cliques of the underlying decomposable graph.

While some updates in this latent space are probabilistically superfluous, as they involve non-maximal cliques, this work has demonstrated that such latent expansion is promising for two reasons. First, controlling the size of this expansion can be done efficiently and intuitively by restricting the size of the latent junction trees to the number of graph vertices; it is intuitive since a no-edge graph has the maximum number of maximal cliques over the set of vertices. Second, parallel sampling, which improves mixing properties, seamlessly integrates into our framework as it does not necessitate junction tree modification post-update—a requirement in other junction tree-based samplers.

Parallelism is a key benefit of our sampler over competitors, but not the only one. All proposals from our samplers inherently preserve decomposability, with rejections occurring due to likelihood ratio. A consequence of Theorems 4.2 and 4.3. By contrast, the disconnect move in the multi-edge sampler of Green and Thomas (2013, Sec. 3.2 conditions (a)-(d)) can cause decomposability-based rejections, leading to less optimal mixing. These rejections occur at the proposal stage without a full evaluation of the data likelihood (Green and Thomas, 2013, Sec. 3.4). This leads to faster computational time (SM Fig. S22), yet reduced acceptance rates (Fig. 9 left), for the multi-edge sampler when compared to others.

Green and Thomas (2013) implemented a multi-edge update per proposal, choosing edge sets uniformly from a valid superset with probability proportional to $2^m - 1$, where $m$ is the cardinality of the superset. This approach appears less efficient compared to alternatives. The average acceptance rate for their multi-edge proposal is lower than the single-edge proposal, as shown in (Green and Thomas, 2013, Supp. Fig. 3) and Figure 9 (left). This work opted for an all-or-none multi-edge update, since it is more intuitive with respect to the junction property, where it does not require any modification to the tree skeleton. Our proposal outperformed competing ones in efficiency and accuracy.

Recent advances by Uhler et al. (2018) in providing exact formulas for normalizing constants of Wishart priors have addressed a major analytical bottleneck in nondecomposable graphical models, previously a significant challenge in the field as noted by Jones et al. (2005); Wang and Carvalho (2010); Mitsakakis et al. (2011); Roverato (2002); Atay-Kayis and Massam (2005); Dellaportas et al. (2003). However, computational barriers persist. In moderate-dimensional problems, state-of-the-art Bayesian structure learning methods for nondecomposable graphical models executes 1000 steps in 15 minutes, as shown by Mohammadi et al. (2023), compared to less than a second for similar tasks in decomposable models, as shown here and in Green and Thomas (2013); Giudici and Green (1999). In fact, under a Gaussian graphical model of 150 vertices, our samplers execute 1000 steps
in 0.3 seconds. This dramatic reduction in computational time highlights the efficiency advantage of assuming decomposability in the conditional independence graph $G$.

It still remains unclear whether Bayesian inference for graphical models can be achieved for large-dimensional problems, beyond 1000 vertices for example, with no reliance on the decomposability framework.

**Acknowledgements**

The author wishes to acknowledge Felix Rios (Royal Institute of Technology in Stockholm) for providing necessary code through the trilearn Python package, and the structure learning benchmarking workflow Benchpress, and for his comments, which improved the quality of this work. The author was supported by an NSERC postdoctoral fellowship.

**References**

Atay-Kayis, A. and Massam, H. (2005). A Monte Carlo method for computing the marginal likelihood in nondecomposable Gaussian graphical models. *Biometrika* 92 317–335.

Bierkens, J. (2016). Non-reversible Metropolis–Hastings. *Statistics and Computing* 26 1213–1228.

Bornn, L. and Caron, F. (2011). Bayesian clustering in decomposable graphs. *Bayesian Analysis* 6 829–846.

Byrne, S., Dawid, A. P. et al. (2015). Structural Markov graph laws for Bayesian model uncertainty. *The Annals of Statistics* 43 1647–1681.

Cayley, A. (1889). A theorem on trees. *Quart. J. Math.* 23 376–378.

Cowell, R. G., Dawid, P., Lauritzen, S. L. and Spiegelhalter, D. J. (2006). *Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks*. Springer Science & Business Media.

Dawid, A. P. and Lauritzen, S. L. (1993). Hyper Markov laws in the statistical analysis of decomposable graphical models. *The Annals of Statistics* 21 1272–1317.

Dellaportas, P., Giudici, P. and Roberts, G. (2003). Bayesian inference for non-decomposable graphical Gaussian models. *Sankhyā: The Indian Journal of Statistics* 43–55.

Duncan, A. B., Lelievre, T. and Pavliotis, G. A. (2016). Variance reduction using nonreversible Langevin samplers. *Journal of statistical physics* 163 457–491.

Frydenberg, M. and Lauritzen, S. (1989). Decomposition of maximum likelihood in mixed graphical interaction models. *Biometrika* 76 539–555.

Giudici, P. and Green, P. (1999). Decomposable graphical Gaussian model determination. *Biometrika* 86 785–801.
GREEN, P. J. and THOMAS, A. (2013). Sampling decomposable graphs using a Markov chain on junction trees. *Biometrika* 100 91–110.

GREEN, P. J. and THOMAS, A. (2018). A structural Markov property for decomposable graph laws that allows control of clique intersections. *Biometrika* 105 19–29.

GRONE, R., JOHNSON, C. R., SÁ, E. M. and WOLKOWICZ, H. (1984). Positive definite completions of partial Hermitian matrices. *Linear algebra and its applications* 58 109–124.

HASTINGS, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57 97–109.

JONES, B., CARVALHO, C., DOBRA, A., HANS, C., CARTER, C. and WEST, M. (2005). Experiments in stochastic computation for high-dimensional graphical models. *Statistical Science* 20 388–400.

LAURITZEN, S. L. (1996). *Graphical Models*. Oxford University Press.

MITSAKAKIS, N., MASSAM, H. and ESCOBAR, M. D. (2011). A Metropolis–Hastings based method for sampling from the g-Wishart distribution in Gaussian graphical models. *Electronic Journal of Statistics* 5 18–30.

MOHAMMADI, R., MASSAM, H. and LETAC, G. (2023). Accelerating Bayesian structure learning in sparse Gaussian graphical models. *Journal of the American Statistical Association* 118 1345–1358.

NEAL, R. M. (1998). Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. In *Learning in graphical models*. Springer, 205–228.

OLSSON, J., PAVLENKO, T. and RIOS, F. L. (2019). Bayesian learning of weakly structural Markov graph laws using sequential Monte Carlo methods. *Electronic Journal of Statistics* 13 2865–2897.

OLSSON, J., PAVLENKO, T. and RIOS, F. L. (2022). Sequential sampling of junction trees for decomposable graphs. *Statistics and computing* 32 80.

REY-BELLET, L. and SPILOPOULOS, K. (2015). Irreversible Langevin samplers and variance reduction: a large deviations approach. *Nonlinearity* 28 2081.

RIOS, F. L., MOFFA, G. and KUIPERS, J. (2021). Benchpress: A scalable and versatile workflow for benchmarking structure learning algorithms. *arXiv preprint arXiv:2107.03863* .

ROVERATO, A. (2002). Hyper inverse Wishart distribution for non-decomposable graphs and its application to Bayesian inference for Gaussian graphical models. *Scandinavian Journal of Statistics* 29 391–411.

SPEED, T. P. and KIVHERI, H. T. (1986). Gaussian Markov distributions over finite graphs. *The Annals of Statistics* 138–150.

TARANTOLA, C. (2004). MCMC model determination for discrete graphical models. *Statistical Modelling* 4 39–61.
Tarjan, R. E. and Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.* **13** 566–579.

Thomas, A. and Green, P. J. (2009). Enumerating the junction trees of a decomposable graph. *Journal of Computational and Graphical Statistics* **18** 930–940.

Uhler, C., Lenkoski, A. and Richards, D. (2018). Exact formulas for the normalizing constants of Wishart distributions for graphical models. *The Annals of Statistics* **46** 90–118.

Wang, H. and Carvalho, C. M. (2010). Simulation of hyper-inverse Wishart distributions for non-decomposable graphs. *Electronic Journal of Statistics* **4** 1470–1475.

Whittle, P. (1985). Partial balance and insensitivity. *Journal of Applied Probability* **22** 168–176.

Wormald, N. C. (1985). Counting labelled chordal graphs. *Graphs and Combinatorics* **1** 193–200.
Supplementary material

A  Table of notations

| Symbol       | Description                                                                 |
|--------------|-----------------------------------------------------------------------------|
| $G = (V, E)$ | An undirected decomposable graph with vertex set $V$ and edge set $E$        |
| $V \subseteq V$ | A subset of vertices $V$                                            |
| $c1(G)$     | The set of unique maximal cliques of $G$                                    |
| $\text{sep}(G)$ | The set of unique minimal separators of $G$                                |
| $J = (c1(G), \text{sep}(G))$ | A junction tree of $G$ with vertex set $c1(G)$ and edge set $\text{sep}(G)$ |
| $J^G$       | The set of junction trees of $G$                                           |
| $C \in \mathcal{C}$ | A superset of cliques of $G$                                             |
| $S \in \mathcal{S}$ | A superset of separators of $G$                                           |
| $T = (C, S)$ | A junction tree with vertex set $C$ and edge set $S$                        |
| $G_V$       | The induced subgraph with the vertex set $V$                               |
| $T_v$       | The induced subtree of $T$, where every $C \in T_v$ includes $v$           |
| $u \sim v \subseteq G$ | A path between $v$ and $u$ in $G$ consisting of a sequence of vertices $|$ |
| $C_1 \sim C_2 \subseteq T$ | A path between cliques $C_1$ and $C_2$ in $T$ consisting of a sequence of cliques in $C$ |
| $|A|$        | The size of the set $A$                                                   |
| $\text{nei}(C, T)$ | The set of cliques in $T$ adjacent to $C$                                 |
| $\text{deg}(C, T)$ | The number of adjacent cliques to $C$ in $T$, that is $|\text{nei}(C, T)|$ |
| reduce$(T)$ | A reduction of $T$ from relations over $(C, S)$ to one over $(c1(G), \text{sep}(G))$ |
| $g(T)$      | The unique decomposable graph represented by $T$                          |

B  Proof of Theorem 4.2 and 4.3

B.1  Theorem 4.2

We consider the simplest case, where $V = \{v\}$ and $U \subset V$ is a subset of vertices that form a complete graph in $G = (V, E)$. There is no edge between $V$ and any node in $U$, and $G$ is connected. Let $J = (c1(G), \text{sep}(G))$, be a junction tree of $G$. Let $G' = (V, E')$ be a graph such that $E \subset E'$, and $G'$ is formed by connecting edges $(v, u)$, for every $u \in U$. Let $C_V \in c1(G)$ be a maximal clique containing $V$. Similarly, let $C_U \in c1(G)$ such that $U \subseteq C_U$. When $U$ is partitioned into mutually exclusive sets $\{U^{(1)}, \ldots, U^{(K)}\}$, then none of those sets are empty, as it contradicts the fact that $U$ is completely connected. It is sufficient to prove Theorem 4.2 for the simplest case when $U = U^{(1)}$, since the same argument can be applied to connect to $U^{(2)}$, once connected to $U^{(1)}$, and so forth.

- If $C_U \in \mathcal{N}_V(J)$, then $G'$ is decomposable. This case follows directly from (Green and Thomas, 2013, Prop. 1). Here we present a different proof. The steps are illustrated graphically in Fig. S10. We know that $(C_V, C_U) \in J$. Let $C'' = V \cup S \cup U$, where $S = C_V \cap C_U$. Create $J'$ as follows, add $C''$ between $C_V$ and $C_U$ by replacing the edge $(C_V, C_U)$ with $(C_V, C'')$ and $(C', C_U)$. $J'$ satisfies the junction property (2.2).
  Since for any clique $C_i, C_j \in J$, if their path intersects the edge $(C_V, C_U)$, then $C_i \cap C_j \subseteq S \subseteq C' \subseteq J'$. Also, it is easily verified that $(C_1, \ldots, C_V, C', C_U, \ldots, C_c)$ is
Figure S10: Updating junction tree $J$ to $J'$ (and backward) by (dis)connecting vertices in $V, U \in V$. The notation $\{A\}_\perp$ indicates a subtree branch that does not contain $A \subset V$. For a disconnect, modify $J$ to include two dummy cliques $S \cup \{V\}$ and $S \cup \{U\}$ in the junction between $S \cup V \cup U$ and both branches $\{U\}_\perp$ and $\{V\}_\perp$. The final tree $J'$ is formed by removing $V \cup U$ from the $S \cup V \cup U$. $J'$ adheres to the running intersection property and forms a decomposable graph that does not include edges between $V$ and $U$. A connect-move is the opposite. Circles are cliques, boxes are separators.

- We will show the necessary part of the condition by showing that $G'$ contains a non-chordal cycle of 4 vertices. We will pick those vertices from the recursive simplicial subsets over a path in $J$. Assume $G'$ is decomposable, and there does not exist a junction tree $J$ of $G$ such that $(C_{V}, C_{U}) \in J$. Assume that there exists a $J = (cl(G'), sep(G'))$ such that the path $C_{V} \sim C_{U}$ contains a single maximal clique $C$, i.e. $C_{V} \sim C_{U} = (C_{V}, C_{U})$. This path, by definition, is part of the perfect ordering sequence of $G$. Define the following: $S_{V} = C_{V} \cap C$ and $S_{U} = C \cap C_{U}$. Now, we know that: (i) $S_{U} \setminus C_{V} \neq \emptyset$, otherwise $S_{U} \subseteq C_{V}$ meaning that there exists a junction tree that has the edge $(C_{V}, C_{U})$, which is a contradiction; and similarly (ii) $S_{V} \setminus C_{U} \neq \emptyset$. Then, let $b \in S_{U} \setminus C_{V}$ and $a \in S_{V} \setminus C_{U}$. Then all the following edges exits in $G'$, $(v, a), (a, b)(b, u), (v, u)$, where $v \in V, u \in U$. However, neither edges $(v, b), (a, u)$ exists in $G'$, hence $G'$ has a non-chordal cycle of the 4 vertices $(v, a, b, u, v)$, leading to a contradiction.

Now if the path $C_{V} \sim C_{U} = (C_{V}, C, C', C_{U})$, one can find a vertex $d$ in $C \cap C' \setminus C_{V}$ such that the non-chordal cycle $(v, a, d, b, u)$ exist in $G'$. Here, $b \in C_{U} \cap C' \setminus C$. It is possible that $d$ equals $a$ or $b$. In all cases, $G'$ is not chordal.

By induction, the results of Theorem 4.2 follows.
B.2 Theorem 4.3

Under similar setup in proof of Theorem 4.2, in Section B.1. Here $U$ is fully connected to $V$ in $G$ and $G'$ is a graph formed from $G$ by removing edges $(v, u)$, where $v \in V, u \in U$. Again, it is sufficient to prove Theorem 4.3 for the simplest case when $U = U^{(1)}$, since the same argument can be applied to disconnect from $U^{(2)}$, once disconnected from $U^{(1)}$, and so forth.

- If $U \in C$ for some $C \in \mathcal{L}_V(T)$, then $V \cup U$ are contained in only one maximal clique $C$, and $G'$ is decomposable. This follows from (Green and Thomas, 2013, Prop. 2). We present a proof based on the running intersection property (2.1). This process is demonstrated visually in Fig. S10. Define $C' = V \cup U \cup U'$, where $S \subset V$. Let $C'_V, C'_U \in \mathcal{c}(G)$ be such that $(C'_1, \ldots, C'_V, C'_U, \ldots, C'_C)$ is the perfect ordering sequence of $T$ formed by (2.1). Here $(C'_V, C'_U) \in T$ and $(C', C'_U) \in T$, by definition of junction trees edges with relation to (2.1). Now remove $C'$ from the ordering and add $S$ instead, alongside the cliques $C_V = S \cup V$ and $C_U = S \cup U$, to form the new ordering $(C_1, \ldots, C'_V, C_V, C_U, C'_U, \ldots, C_C)$ which respects (2.1) and generates $G'$ and $T' = (\mathcal{c}(G'), \text{sep}(G'))$. To see this, let $C_i, C_j \in \mathcal{c}(G)$ if the path $C_i \sim C_j \subseteq T$ intersects $(C'_V, C'_U)$ or $(C', C'_U)$, then $C_i \cap C_j \subseteq S$, where $S$ is contained in $C_V$ and $C_U$ as well. From (2.1), we have $C_V \cap \{C_1, \ldots, C'_V\} \subseteq C'_V, S \cap \{C_1, \ldots, C'_V, C_V\} \subseteq C_V$, and $C_U \cap \{C_1, \ldots, C'_U, S\} \subseteq S$, while everything else remains the same. If $T'$ contains non-maximal cliques, reduce it as reduce($T'$).

- To show the necessary part of the conditions. Assume that $G'$ is decomposable, but there does not exist a $C \in \mathcal{c}(G)$ such that $C \in \mathcal{L}_V(T)$, for any junction tree $T$ of $G$. We will show by contradiction, that $G'$ has a non-chordal cycle of 4 vertices, by picking vertices of recursively simplical subsets of a path in $T$ that contains $V \cup U$. For simplicity, let $V \cup U$ be contained in only two maximal cliques $C_1, C_2 \in \mathcal{c}(G)$, then by the run intersection property (2.1), there exists a junction tree $T$, of $G$, such that the edge $(C_1, C_2) \in T$. Let $C_1 = A \cup V \cup U \cup S$ and $C_2 = S \cup V \cup U \cup B$, such that $C_1 \cap C_2 = V \cup U \cup S$, for $A, B, S \subset V$. Here $A$ and $B$ are no connected, and not empty, i.e. $A \neq \emptyset$ and $B \neq \emptyset$. Otherwise, either $C_1$ or $C_2$ would be non-maximal. Disconnecting edges $(v, u), v \in V, u \in U$, in $G$ would result in the cycle $(a, v, b, u, a)$ for $a \in A$ and $b \in B$. Since the edges in this cycle associated with $a$ and $b$ still exist. This contradicts the fact that $G'$ is decomposable.

This completes the proof of Theorem 4.3.

C Inference setup for Gaussian graphical model

Consider a $p$-dimensional zero-mean (for simplicity) Gaussian random vector $Y \in \mathbb{R}^n$ that is globally Markov with respect to a decomposable graph $G$. Its precision matrix (inverse covariance) $\Theta$ belongs to the set

$$A_G = \{\Theta \in \mathcal{M}_p^+: \Theta_{ij} = 0 \text{ for all } \{i, j\} \not\in G\},$$

where $\mathcal{M}_p^+$ is the space $p \times p$ of positive definite matrices. Conditional independence between $i$th and $j$th variable in Gaussian models is equivalent to having $\Theta_{ij} = 0$ (Speed and Kiiveri, 1986). Let $y = \{y_i\}$, for $i = 1, \ldots, n$, be $n$ observations from this model. Following the
factorization law in (1.1), the full likelihood of $y$ can be specified by the clique marginals $c_1(G)$ of $G$, and their separators $\text{sep}(G)$. For clique $C \in C$, the likelihood marginal is

$$f(y_C \mid \Theta_C) = \frac{1}{(2\pi)^{|C|/2} |\Theta_C|^{n/2}} \exp \left\{ -\text{tr} \left( \frac{\Theta_C D_C}{2} \right) \right\},$$

where $D$ is the sample covariance matrix, as $D = n^{-1} \sum_{i=1}^{n} y_i y_i^\top$, $|\Theta|$ is the determinant of $\Theta$, $|C|$ is the cardinality of $C$, and $y_C$ is the subvector $y$ indexes by $C$, similarly for all other elements. Similarly, for $f(y_S \mid \Theta_S)$, where $S \in S$. Following Dawid and Lauritzen (1993), a conjugate prior for $\Theta$ is a hyper-Wishart distribution that is also hyper Markov with respect to $G$, having the form

$$
\eta(\Theta \mid G, \varphi) = \prod_{C \in C} \eta(\Theta_C \mid \varphi_C) \prod_{S \in S} \eta(\Theta_S \mid \varphi_S).
$$

Each prior marginal $\eta(\Theta_C \mid \varphi_C)$ is of the form

$$
\eta(\Theta_C \mid \varphi_C) \propto |\Theta_C|^{\beta_C} \exp \left\{ -\frac{1}{2} \text{tr} \left( \frac{\Theta_C (D_C + Q_C)}{2} \right) \right\},
$$

with normalization constant $\int \eta(\Theta_C \mid \varphi_C) d\Theta_C = 2^{\delta |C|/2} \Gamma(|C|/2)^{|\beta_C|}$, where $\Gamma_k$ denotes the multivariate gamma function. Here $\beta_C = (\delta + |C| - 1)$, $\varphi_C = (Q_C, Q_C) \in M_+^p$ a scale positive definite matrix and $\delta > \max_{C \in C} \{|C| - 1\}$ the number of degrees of freedom. By Dawid and Lauritzen (1993, Thm. 3.9), the collection of priors $\{\eta(\Theta_C \mid \varphi_C)\}$ are pairwise hyper consistent and there exists a unique hyper Wishart joint distribution of the form

$$f(y_C \mid \Theta_C)\eta(\Theta_C \mid \varphi_C) \propto \frac{1}{(2\pi)^{|C|/2} |\Theta_C|^{\alpha_C}} \exp \left\{ -\frac{1}{2} \text{tr} \left( \frac{\Theta_C (D_C + Q_C)}{2} \right) \right\},$$

where $\alpha_C = (\delta + n + |C| - 1)/2$. By conjugacy, it is possible to integrate out $\Theta$ of the term above. With a junction prior of the form (5.4), the junction posterior is

$$\pi(T \mid t) \propto \prod_{C \in C} \phi(C) \rho(C) \prod_{S \in S} \psi(S) \rho(S),$$  \hspace{1cm} (C.1)

with

$$\rho(C) = \frac{|Q_C|^\alpha}{|Q_C + D_C|^{\beta_C}} \Gamma(|C|/2)^{\alpha_C} \Gamma(|C|/2)^{\beta_C},$$

and $\rho(S), S \in S$ is defined similarly. As indicated by Grone et al. (1984), given that $G$ is decomposable and $\Theta$ is positive definite for each clique, $\Theta$ is both existent and unique for each graph.
D Simulation code

To run the simulation code, please follow these steps:

1. **Install Benchpress**: Download and install Benchpress from the official documentation. For detailed instructions, visit the Benchpress documentation website at https://benchpressdocs.readthedocs.io/en/latest/.

2. **Clone the GitHub Repository**: Use the following commands to clone the specific branch of the Benchpress repository:

   ```bash
   git clone https://github.com/felixleopoldo/benchpress.git
   cd benchpress
   git checkout paralleldg
   git pull origin paralleldg
   ``

3. **Run the Simulation**: Execute the simulation using Snakemake as:

   ```bash
   # A single replication of (n,p)=(100,50) (~2hrs)
   snakemake --cores all --use-singularity
   --configfile config/elmasri_parallel_single_run.json
   --configfile config/elmasri_parallel.json

   # A single replications of (n,p)=(100,200) (~4hrs)
   snakemake --cores all --use-singularity
   --configfile config/elmasri_parallel_high-dim_single_rung.json

   # 10 replications of (n,p)=(100,200) (~48hrs)
   snakemake --cores all --use-singularity
   --configfile config/elmasri_parallel_high-dim.json
   ```

   (compute times are based on an Intel i7 with 4 cores)

4. **Results**: will be available at benchpress/results/outputs/.
E Decomposable graphs of seven vertices

We analyzed all 2,097,152 undirected graphs on seven labeled vertices, identifying 61,675 decomposable ones. We indexed each graph’s cliques into a counter table and used the algorithm from Thomas and Green (2009) to calculate the number of possible reduced junction tree representations for each graph, denoted as $|\mathcal{J}^G|$. The no-edge graph is represented by 16,807 reduced junction trees, while there are 18,447 graphs, each represented by a single reduced junction tree.

First, we restricted the sampler to the reduced junction tree space, ensuring that every update led to a change in the underlying graph. We set the junction tree prior as $\pi(T \mid t) \propto 1$, indicating uniform sampling over the space of reduced junction trees. In a subsequent run, we defined the prior as $\pi(T \mid t) \propto 1/|\mathcal{J}^g(T)|$, aiming for uniform sampling over the space of decomposable graphs.

We initiated the single-move sampler with a no-edge graph and ran it for 1,000,000 steps. This produced a trajectory of junction trees, which we then converted into a trajectory of decomposable graphs. For each decomposable graph, we computed empirical frequencies, cumulatively forming an estimated distribution function. As depicted in SM Figure S27, there is a strong match between the expected and estimated cumulative distribution functions for the two priors. This alignment suggests that our single-move sampler effectively and uniformly traverses both the reduced junction tree space and the decomposable graph space.

When not limited to the space of reduced junction trees, our single-move sampler uniformly samples from the broader space of junction trees with $p$-vertices. Enumerating this space is a significant challenge, requiring a brute force approach that’s only viable for very small values of $p$ since no analytical expression exists. Drawing inspiration from Algorithm 1 and the visit probability $\rho \in (0, 1)$, we devised a proposal over the update-type. This approach aims to approximate uniform sampling across the space of decomposable graphs.

Let $\tilde{q}(\text{add})$ be the modified proposal that integrate the probability that the random walk on $T_v$ visits a neighboring clique $C \in \mathcal{N}$, that is $v$ is added to $C$. We express this as

$$\tilde{q}(\text{add}) = \frac{e^{-\rho}}{1 + e^{-\rho}}, \quad \rho \in (-\infty, \infty), \quad (E.1)$$

As $\rho$ deviates from 0, it skews the proposal probability of an addition move. Positive values of $\rho$ discourage such additions, while negative values encourage them. This framework allows us to modulate our sampling behavior based on the desired balance between addition and removal of nodes.

Following the simulation setup above, we equip the single-move and parallel samplers with the proposal (E.1), where the acceptance ratios in (6.2) and (6.2) are multiplied by the factor $\tilde{q}(\text{add})/(1 - \tilde{q}(\text{add}))$, or its reciprocal, accordingly. Figure S11 shows that, when $\rho = 0.4$ (or $\pi(\text{add}) = 0.4$), our single-move sampler is able to traverse the space of decomposable graphs over seven vertices approximately uniformly. Varying $\rho$ can lead to sampling decomposable graphs more or less frequently, according to the number of reduced junction tree representations. The parallel sampler behaves similarly (SM Fig. S28). Performing 1,000,000 updates took around 60 seconds for the single-move sampler, and half that time for the parallel sampler.
Figure S11: Cumulative distribution functions for decomposable graphs over seven vertices, estimated from samples drawn from the single-move sampler with added proposal specified in (E.1). The x-axis enumerates the graphs in decreasing order based on the number of reduced junction tree representations.

F Effect of graph prior on acceptance rate

We observed a notable influence of the graph prior on the acceptance rate in our single-move sampler. With a uniform prior and an autoregressive graph structure, the average acceptance rate for junction-updates is about 48%. However, when using a modified exponential family prior, defined as $\phi(C) = \exp(\alpha(\|C\| - 1))$ and $\psi(S) = \exp(\beta|S|)$ with $\alpha = 2$ and $\beta = 4$, this rate dropped significantly to 22%, as shown in the following table. This substantial decrease in acceptance rate illustrates the impact of more conservative priors in reducing the number of non-maximal cliques in the junction tree $T$, as detailed in Section 6.3.

| Graph-updates | Junction-updates |
|---------------|------------------|
| Uniform prior | 28.17            | 48.05           |
| Exponential prior | 12.38       | 22.52           |

G Extra simulation results for the general case

A step in our parallel sampler can encompass multiple proposals executed concurrently. Specifically, updating the chain from $T$ to $T'$ may entail $k > 0$ parallel updates, which are order-invariant, resulting in a sequence such as $T_i, T_{i1}, \ldots, T_{ik} = T'$. To elucidate the impact of parallelism, we post-process each chain after sampling to derive a corresponding serial chain, where each proposal is individually indexed. Consequently, a step consisting of $k$ parallel updates in the parallel sampler is represented as $k$ individual steps in the serial chain.
Figure S12: Boxplot of the average computation time in seconds for 1000 steps across two graph structures, detailed in Section 7.1, over 10 replications, on an Intel Xeon E5-2698 v4 2.20GHz processor, when \((n, p) = (100, 50)\).

Table S4: Mean and standard deviation of compute time (in seconds) for 1000 steps of the Gaussian decomposable graphical models discussed in Section 7, when \((n, p) = (100, 50)\)

|                | Parallel | Single-move | GT13-m | GT13-s | O19   | GG99 |
|----------------|----------|-------------|--------|--------|-------|------|
| mean           | 0.20     | 0.26        | 0.04   | 0.19   | 3810.16 | 0.06 |
| std            | 0.04     | 0.03        | 0.01   | 0.04   | 122.58 | 0.01 |
Figure S13: Likelihood traceplots for 10 replications (color-matched) for samplers discussed in Section 7, under the autoregressive graph structure and \((n, p) = (100, 50)\).
Figure S14: Autocorrelation plots for 10 replications (color-matched) for samplers discussed in Section 7, under the autoregressive graph structure and \((n, p) = (100, 50)\).
Figure S15: Traceplots showing the number of graph edges for 10 replications (color-matched), as outlined in Section 7.1 and \((n, p) = (100, 50)\).
Figure S16: Graph and adjacency matrix plots discussed in Section 7.1, generated using the autoregressive structure and $p = 50$. 
Figure S17: Graph and adjacency matrix plots discussed in Section 7.1, generated using the Christmas tree algorithm (Olsson et al., 2022) and $p = 50$. 
Figure S18: Heatmap derived from the final 300,000 steps of samplers in Section 7.1, covering replications 1 to 5 under the Christmas tree algorithm and \((n, p) = (100, 50)\).
Figure S19: Heatmap derived from the final 300,000 steps of samplers in Section 7.1, covering replications 5 to 10 under the Christmas tree algorithm and \((n, p) = (100, 50)\).
Figure S20: Heatmap derived from the final 300,000 steps of samplers in Section 7.1, covering replications 1 to 5 under the autoregressive structure and \((n, p) = (100, 50)\).
Figure S21: Heatmap derived from the final 300,000 steps of samplers in Section 7.1, covering replications 6 to 10 under the autoregressive structure for $(n, p) = (100, 50)$. 
Figure S22: Average time (in seconds) for 1000 steps of the Gaussian decomposable graphical model with an autoregressive structure, comparing the parallel sampler, its serial variant, and the single-move sampler. The number of data samples equal the dimension.
Figure S23: Likelihood traceplots for 10 replications (color-matched) for samplers discussed in Section 7, under the autoregressive graph structure for \((n, p) = (100, 200)\).
Figure S24: Autocorrelation plots for 10 replications (color-matched) for samplers discussed in Section 7, under the autoregressive graph structure for \((n, p) = (100, 200)\).
Figure S25: Traceplots showing the number of graph edges for 10 replications (color-matched), as outlined in Section 7.1 for \((n, p) = (100, 200)\).

Figure S26: (Left) Boxplot comparing acceptance rates over graph-updates between our parallel and single-move samplers and competitors, across two graph structures and 10 replications. (Right) ROC curves comparing the same samplers under the autoregressive graph structure only. Here \((n, p) = (100, 200)\).
I  Decomposable graphs over seven vertices

Figure S27: Cumulative distribution functions for decomposable graphs over seven vertices, estimated from samples drawn from the single-move sampler (a) with probability proportional to the number of junction tree representations, and (b) uniformly. The solid lines represent expected frequencies, and the dashed lines represent observed frequencies. The x-axis enumerates the graphs in decreasing order based on the number of reduced junction tree representations.
Figure S28: Cumulative distribution functions for decomposable graphs over seven vertices, estimated from samples drawn from the parallel sampler with proposal specified in (E.1). The x-axis enumerates the graphs in decreasing order based on the number of reduced junction tree representations.