Applied Mathematics on Science and Engineering, 2(1):1-20, 2025.

Applied Mathematics on Science and Engineering

Journal homepage: https://furthersci.org/journal-details.php?journal-id=49

ISSN (E): 3048-8362

Review Article

A Brief Overview of Applications of Tree-Width and Other Graph Width Parameters

Takaaki Fujita [1*]

[1] *Independent Researcher, Shinjuku, Shinjuku-ku, Tokyo, Japan.*
[*] *Corresponding author: takaaki.fujita060@gmail.com*

Article Info

Keywords: *Treewidth, Pathwidth, Graph, Width parameters*

Received: *02.10.2024*
Accepted: *05.05.2025*
Published: *26.05.2025*

Abstract

Graph theory, a fundamental branch of mathematics, centers on the study of networks composed of vertices (nodes) and edges, examining their paths, structures, and properties. One essential metric in this field is the "graph width parameter," which quantifies the maximum width across all cuts or layers within a hierarchical decomposition of the graph. Tree-width, in particular, has garnered significant attention due to its broad range of applications. In this work, we revisit these parameters and their applications, focusing specifically on tree-width and related graph width parameters in the contexts of Bond Graphs, Factor Graphs, and Graph Entropy.

1. Introduction

1.1. Graph Structures and Graph Classes

Graph theory, a fundamental area of mathematics, focuses on the study of structures consisting of vertices (nodes) and edges that represent relationships or connections [1]. Due to its flexibility and strong theoretical foundation, graph theory has been extensively applied across a wide range of disciplines, including computer science, biology, social sciences, and engineering [2–4].

The study of graph theory often involves the investigation of specific graph structures such as paths [5], trees [6], cycles [7], complete graphs [8], cliques [9], wheels [10], and forests [11] (cf. [12]). For example, a tree is defined as a connected, acyclic graph in which there exists exactly one path between any pair of vertices.

In this context, graph algorithms—systematic procedures for solving computational problems—are of particular interest. Many such algorithms are tailored to specific graph structures in order to enhance efficiency and scalability (cf. [13–15]).

A *graph parameter* is a numerical value assigned to a graph to describe certain structural characteristics, such as its size, connectivity, or topological configuration [16, 17]. These parameters are crucial for analyzing, comparing, and optimizing graph-based models.

To address the needs of diverse applications, various generalized graph models have been developed. These include:

- **Fuzzy Graphs** [18–20], which incorporate uncertainty in edge and vertex membership.
- **Neural Graphs** [21–23], used in machine learning to represent neural architectures and data flow.
- **Code Property Graphs** [24], applied in static code analysis and vulnerability detection.
- **Reeb Graphs** [25], used in topology-based shape analysis.

- **Disjunctive Graphs** [26, 27], important in scheduling and combinatorial optimization.
- **Neutrosophic Graphs** [28, 29], incorporating indeterminacy into graph models.
- **Plithogenic Graphs** [30, 31], capturing contradictory, neutral, and supportive relationships.
- **Hypergraphs** [32, 33], where edges (hyperedges) can connect any number of vertices.
- **Hypersoft Graphs** [34, 35], generalizing soft sets in graph structures.
- **Rough Graphs** [36, 37], used in uncertainty modeling and approximation theory.
- **Superhypergraphs** [38–40], employing higher-order power sets to represent layered relationships.
- **Flow Graphs** [41–43], essential for representing control or data flow in systems.
- **Collaboration Graphs** [44, 45], modeling co-authorship and social networks.
- **Majority-Inverter Graphs** [46], used in logic synthesis and circuit design.
- **And-Inverter Graphs** [47–49], optimized representations in digital logic.
- **Tensor Graphs** [50, 51], integrating tensor structures into graph models for high-dimensional data.

These diverse graph classes demonstrate the adaptability and theoretical richness of graph theory, allowing it to model complex systems across domains.

1.2. Graph Width Parameters and Tree-width

One of the central metrics in graph theory is the *graph width parameter*, which quantifies the structural complexity of a graph by measuring the maximum width across all cuts or layers in a hierarchical decomposition [52–55]. This parameter is essential for analyzing the internal structure of a graph and has significant implications in algorithmic graph theory. In particular, many computationally intractable problems become tractable when restricted to graphs with bounded width.

Among various width parameters, *tree-width* is one of the most widely studied and practically useful [56–59]. Tree-width measures how close a graph is to being a tree by identifying the minimum width among all possible tree decompositions. A smaller tree-width indicates that the graph exhibits more tree-like behavior, which often translates to lower computational complexity in solving problems on such graphs.

Extensive research has led to the development of numerous algorithms for computing tree-width and other width parameters [60–63]. These algorithms are particularly valuable because many real-world applications benefit from exploiting tree-like structures. Moreover, a wide range of NP-hard problems[1] can be efficiently solved on trees, often in polynomial or even linear time. This efficiency also holds for graphs with constant tree-width, where dynamic programming methods can be used to design polynomial-time algorithms.

1.3. Our Contribution in This Paper

We summarize our contributions as follows. Given the significance of tree-width in both theoretical and practical contexts, we aim to deepen the understanding of width parameters and extend their applications. Specifically, we revisit the definitions and properties of tree-width and related parameters, examining their roles in the structure and analysis of *Bond Graphs*, *Factor Graphs*, and *Graph Entropy*.

1.4. Contents of the paper

The remainder of the paper is organized as follows:

- Section 2 provides an overview of various application backgrounds related to graph width.
- Section 3 formally defines the tree-width parameter.
- Section 4 explores the behavior of tree-width in specific classes of graphs.
- Section 5 concludes the paper and outlines directions for future research.

2. Background of Applications

Tree-width and its related width parameters have already found numerous applications, owing to their tractability and the structural advantages of tree-like graphs such as tree graphs and path graphs. A brief overview of these applications is provided below (cf. [64]). This overview also serves to introduce prior research, with the expectation that these foundational results will inspire further applications and new ideas in the future.

2.1. Application aspect of tree-width

In this subsection, we consider about application aspect for tree-width. Tree-width is a graph parameter that measures how closely a graph resembles a tree. Lower tree-width means the graph is more ”tree-like,” making certain algorithms more efficient for such graphs[55]. Due to the versatility of tree structures in handling various types of data, tree-width has numerous applications across different fields (cf. [64]).

Tree-width for Bayesian network

A Bayesian network is a graphical model representing probabilistic relationships among variables using nodes and directed edges (cf.[65]).

Tree decomposition in Bayesian Networks simplifies the structure by breaking complex networks into tree-like components. This helps to efficiently perform exact inference and structure learning, particularly for networks with bounded tree-width, allowing scalable algorithms to handle large datasets while maintaining computational tractability[66, 67].

Tree-width for Neural networks

Neural networks are computational models inspired by the human brain, consisting of interconnected nodes that process data and learn patterns[68]. In Graph Neural Networks (GNNs), tree decomposition is used to simplify complex graphs by breaking them down into tree-like structures. This allows GNNs to more efficiently process local graph structures, improving their expressive power and enabling more accurate predictions, especially for graphs with bounded tree-width [21, 69–73].

Tree-width for Tensor network

Tensor networks are mathematical structures used to efficiently represent and compute high-dimensional data. They are applied in quantum physics, machine learning, and compressing large datasets by capturing complex relationships among variables (cf.[74]).

Applications for Tensor network have also been studied in the context of tree-width [75–78]. In tensor networks, tree decomposition helps optimize tensor contraction by minimizing the size of intermediate tensors generated during the contraction process. By reducing the network's tree-width, it becomes possible to identify efficient contraction orders, lowering computational complexity and memory requirements, thus enabling faster and more efficient simulations of large tensor networks.

Tree-width for RNA and DNA structure

Research on RNA and DNA is also advancing rapidly in the field of medicine. DNA is a stable, double-stranded molecule that stores genetic information in the form of a double helix, while RNA is single-stranded and more flexible, playing a crucial role in protein synthesis.

Tree-width has been applied in the study of RNA and DNA as well [79–82]. In particular, tree decomposition simplifies the analysis of DNA structures by breaking down complex interactions into tree-like components. This allows for more efficient solutions to problems such as pathway prediction, by integrating both sequence and structural information, which improves the accuracy of computational biology methods.

Tree-width for Protein Structure

Protein research is flourishing in the medical field, particularly regarding its structure and function. Protein structure refers to the three-dimensional arrangement of amino acids in a protein, which can be represented as a graph. In this graph, nodes correspond to amino acids, and edges represent bonds or interactions, capturing the intricate folding and connectivity of the protein (cf.[83, 84]).

The concept of tree-width has also been applied in protein structure analysis [85–88]. Tree-width helps decompose complex protein interaction graphs into simpler, tree-like structures. This decomposition is particularly useful in computational protein design, as it reduces the complexity of the search space, making it easier to solve problems such as identifying stable protein conformations.

Tree-width for Markov network

Markov networks, or Markov random fields, are undirected graphical models representing the joint distribution of variables. They are used in applications like image processing and statistical physics, emphasizing local dependencies (cf.[89]).

Applications for Markov network have also been studied in the context of tree-width [90–94]. In Markov networks, tree decomposition simplifies complex probabilistic models by breaking them into tree-like structures, which helps efficiently perform inference and computations. By limiting the tree-width, it becomes feasible to approximate joint probability distributions and perform accurate predictions, especially for real-world systems constrained by small tree-width values.

Tree-width for Social Networks

A social network consists of individuals or organizations connected through social relationships, such as friendships or collaborations. The study of social networks has also explored applications of tree-width [95–97]. By applying tree decomposition techniques, complex networks can be simplified, enabling more efficient analysis of connectivity and interactions within these networks.

2.2. Application aspect for path-width

We consider about application aspect for path-width. Path-decomposition is a method of representing a graph as a sequence of overlapping subgraphs (called "bags"), forming a path-like structure. Path-width is a measure of how close a graph is to a path. Lower path-width indicates simpler, more path-like structures, useful for simplifying complex graphs (cf.[98–100]).

Path-width for VLSI (Very Large Scale Integration)

VLSI refers to the process of integrating thousands to millions of transistors onto a single semiconductor chip (cf.[101–103]). It's crucial for developing modern electronic devices, enabling complex circuits and high-performance computing. Applications for VLSI have also been studied in the context of path-width [104–106].

Path-width for Compiler on computers

A compiler is software that translates high-level programming code into machine-readable code, enabling program execution on computers. Applications for Compiler have also been studied in the context of path-width [107, 108]. Additionally, concepts related to path-width, such as proper path-width, are well-known and have applications in game theory [64, 109, 110].

Path-width for Blockchain

Blockchain is a decentralized, distributed ledger technology that securely records transactions across multiple computers, ensuring transparency and immutability. Applications for blockchain have also been studied in the context of path-width[111, 112].

2.3. Application aspect for Hypertree-width

A hypergraph is a generalization of a graph where edges, called hyperedges, can connect any number of vertices, not limited to just two, allowing more complex relationships [113, 114]. Hypertree-width generalizes tree-width to hypergraphs [33, 115], measuring how "tree-like" a hypergraph is by evaluating the smallest possible width of a hypertree decomposition, improving efficiency in certain computational problems.

Hypertree-width for Artificial intelligence

Artificial intelligence is the simulation of human intelligence in machines, enabling them to perform tasks like reasoning, learning, and problem-solving. Similar to tree-width, the concept of hypertree-width is applied in the techniques and concepts of artificial intelligence that utilize graphs[116, 117].

Hypertree-width for Database

A database is a structured collection of data designed for efficient storage, retrieval, and management of information. The relationship between queries and hypertree-width has been extensively studied in database contexts [115, 118–121]. Additional related graph width parameters include FAQ-width [122, 123], m-width [124], closure tree-width [125], closure hypertree-width [125], TCLUSTER-width [126, 127], hinge-width [128, 129], β-Hyperorder width [130, 131], and # hypertree-width [132, 133].

For readers interested in more detailed information or exploring these applications, we recommend consulting lecture notes, surveys, or books on the subject (e.g., [114, 134–136]).

2.4. Other width parameters

The applications of various width parameters have been widely studied. For example, submodular-width is utilized in databases [137, 138], while bandwidth and carving-width are applied in network analysis [139, 140]. Parameters such as cut-width have been explored in fields like VLSI design [141].

3. Preliminaries and Definitions

In this section, we briefly present the definitions and notations used throughout the paper. Unless otherwise specified, all graphs considered in this paper are assumed to be finite, undirected, and simple.

3.1. Tree-width of Undirected Basic Graphs

In graph theory, the concept of *tree-width* is fundamentally linked to the theory of graph minors. A graph minor is derived from a given graph through a sequence of operations, including the deletion of edges, deletion of vertices, and contraction of edges. Tree-width measures how closely a graph resembles a tree by representing it in a tree-like structure with the minimum possible width [142–146].

The notion of tree-width was first introduced in 1972 by Umberto Bertelè and Francesco Brioschi [147], and it was later independently rediscovered by Rudolf Halin in 1976 [148]. In the 1980s, the concept was significantly advanced by Neil Robertson and Paul Seymour, who established tree-width as a central tool in their seminal work on graph minor theory [142, 143, 149]. This foundational idea has since been generalized to more complex structures, giving rise to parameters such as *hypertree-width* [32, 115] and its further extension, the *superhypertree-width* [150–152], which capture tree-like decomposition characteristics in hypergraphs and superhypergraphs, respectively.

Below, we provide the formal definitions of tree-width.

Definition 3.1. *[142] A tree decomposition of a graph $G = (V,E)$ is a pair $(\mathscr{T},\mathscr{V})$, where:*

- *$\mathscr{T} = (I,F)$ is a tree, with I being the set of nodes (called bags) and F the set of edges of the tree.*
- *$\mathscr{V} = \{B_i \mid i \in I\}$ is a collection of subsets $B_i \subseteq V$, called bags, associated with the nodes of the tree.*

The tree decomposition must satisfy the following properties:

1. *For every vertex $v \in V$, there exists at least one bag $B_i \in \mathscr{V}$ such that $v \in B_i$.*
2. *For every edge $(u,v) \in E$, there exists at least one bag $B_i \in \mathscr{V}$ such that $u \in B_i$ and $v \in B_i$.*
3. *For any three bags $B_i, B_j, B_k \in \mathscr{V}$, if B_k lies on the path between B_i and B_j in the tree $\mathscr{T}$, then $B_i \cap B_j \subseteq B_k$.*

Definition 3.2. *[142] The width of a tree decomposition $(\mathscr{T},\mathscr{V})$ is defined as:*

$$\max_{i\in I} |B_i| - 1$$

The tree-width of a graph G is the minimum width among all possible tree decompositions of G. Formally, the tree-width $tw(G)$ is given by:

$$tw(G) = \min_{(\mathscr{T},\mathscr{V})} \left(\max_{i\in I} |B_i| - 1 \right)$$

where the minimum is taken over all tree decompositions of G.

Example 3.3 (Tree Decomposition of the 4-Cycle C_4)**.** *Let $G=(V,E)$ be the cycle graph on four vertices:*

$$V=\{v_1,v_2,v_3,v_4\},\quad E=\{(v_1,v_2),(v_2,v_3),(v_3,v_4),(v_4,v_1)\}.$$

We describe a tree decomposition $(\mathscr{T},\mathscr{V})$ of G as follows:

- *The tree $\mathscr{T}=(I,F)$ has two nodes $I=\{a,b\}$ and a single edge $F=\{(a,b)\}$.*
- *The bag assignment $\mathscr{V}=\{B_a,B_b\}$ is given by*

$$B_a=\{v_1,v_2,v_4\},\quad B_b=\{v_2,v_3,v_4\}.$$

One checks the three defining properties:

1. *Every vertex appears in at least one bag: $\{v_1,v_2,v_4\}\cup\{v_2,v_3,v_4\}=V$.*
2. *Every edge is contained in some bag: $(v_1,v_2),(v_4,v_1)\subseteq B_a$ and $(v_2,v_3),(v_3,v_4)\subseteq B_b$.*
3. *The bag-intersection property holds: the only nontrivial intersection is $B_a\cap B_b=\{v_2,v_4\}$, and every path in $\mathscr{T}$ whose endpoints are a and b passes through a node whose bag contains that intersection.*

Since $\max\{|B_a|,|B_b|\}=3$, the width of this decomposition is

$$\max_{i\in\{a,b\}}|B_i|-1\;=\;3-1=2.$$

One can show no tree decomposition of C_4 has width less than 2, so $\mathrm{tw}(C_4)=2$.

4. Short Survey and Proposal in this paper

We will describe the results presented in this paper.

4.1. Tree-width for Molecular Graph

A molecular graph, also known as a chemical graph, represents the structural formula of a chemical compound using graph theory[153, 154, 154–156]. In this graph, the vertices represent the atoms, while the edges correspond to the chemical bonds between them. Each vertex and edge is typically labeled to reflect the specific elements and types of bonds present in the compound.

Arthur Cayley was probably the first to publish results that consider molecular graphs as early as in 1874"[157]. The definition is provided as follows.

Definition 4.1 (Molecular Graph)**.** *A molecular graph is a type of graph used to represent the structure of a molecule, where:*

- *Each vertex represents an atom type.*
- *Each edge represents a bond type between two atoms.*

Formally, a molecular graph is a graph $G=(V,E)$, where:

- *V is a set of vertices corresponding to atoms in the molecule.*
- *$E\subseteq V\times V$ is a set of edges representing chemical bonds between atoms.*

Example 4.2 (Molecular Graph of Ethanol C_2H_6O)**.** *Ethanol is a colorless, volatile, flammable liquid commonly used as a biofuel, solvent, and active ingredient in alcoholic beverages (cf.[158, 159]). Consider the ethanol molecule, whose chemical formula is C_2H_6O. We model it by the molecular graph $G=(V,E)$ defined as follows:*

$$V=\{C_1,C_2,O_1,H_1,H_2,H_3,H_4,H_5,H_6\},$$

where each C_i denotes a carbon atom, O_1 denotes the oxygen atom, and each H_j denotes a hydrogen atom. The edge set

$$E\;=\;\{\{C_1,C_2\},\{C_1,H_1\},\{C_1,H_2\},\{C_1,H_3\},\{C_2,H_4\},\{C_2,H_5\},\{C_2,O_1\},\{O_1,H_6\}\}$$

represents all single covalent bonds in the molecule:

- *$\{C_1,C_2\}$ is the carbon–carbon bond.*
- *$\{C_1,H_1\},\{C_1,H_2\},\{C_1,H_3\}$ are the three hydrogen atoms bonded to C_1.*
- *$\{C_2,H_4\},\{C_2,H_5\}$ are the two hydrogens bonded to C_2.*
- *$\{C_2,O_1\}$ is the carbon–oxygen bond.*
- *$\{O_1,H_6\}$ is the hydroxyl (O–H) bond.*

One easily verifies that this graph satisfies the definition of a molecular graph, with vertices for each atom and edges for each bond.

The tree-width of molecular graphs (We call "molecular tree-width") has been explored in various studies [160–162]. In practical applications, further investigations could focus on related parameters such as pathwidth, cyclewidth, and starwidth, as necessary.

4.2. Tree-width for Graph Entropy

Entropy is a mathematical concept that quantifies uncertainty or information content within a system, often used in data analysis, physics, and communication theory[163].

Graph entropy is a metric used to quantify the complexity or amount of structural information within a graph [164–166]. It is calculated based on specific graph properties such as vertex degrees, edge orbits, and other factors that reflect the graph's symmetry and structure. The more intricate or irregular the structure, the higher the entropy. This notion of graph-theoretic entropy traces back to Körner in the 1970s [167], and was later surveyed and extended by Dehmer et al. [166]. Below we give mathematically precise, detailed definitions.

Definition 4.3 (Vertex Orbit Entropy [166, 167])**.** *Let $G = (V,E)$ be a finite undirected graph, and let* $\mathrm{Aut}(G)$ *denote its automorphism group. The action of* $\mathrm{Aut}(G)$ *partitions the vertex set V into k disjoint orbits*

$$V = O_1 \dot\cup O_2 \dot\cup \cdots \dot\cup O_k,$$

where each orbit $O_i \subseteq V$ consists of vertices that are topologically equivalent under graph symmetries. Define

$$p_i \;=\; \frac{|O_i|}{|V|}, \qquad i = 1,\ldots,k.$$

Then the vertex orbit entropy *of G is given by the Shannon formula*

$$H_V(G) \;=\; -\sum_{i=1}^{k} p_i \log_2 p_i.$$

This quantity measures the structural diversity of G in terms of its vertex-equivalence classes.

Definition 4.4 (Edge Orbit Entropy [166])**.** *Similarly,* $\mathrm{Aut}(G)$ *partitions the edge set E into ℓ disjoint orbits*

$$E = Q_1 \dot\cup Q_2 \dot\cup \cdots \dot\cup Q_\ell,$$

where each $Q_j \subseteq E$ consists of edges equivalent under automorphisms. Set

$$q_j \;=\; \frac{|Q_j|}{|E|}, \qquad j = 1,\ldots,\ell.$$

The edge orbit entropy *of G is*

$$H_E(G) \;=\; -\sum_{j=1}^{\ell} q_j \log_2 q_j,$$

which quantifies the information content of G's bond (edge) structure.

Definition 4.5 (Attribute-based Graph Entropy [166])**.** *Let $X : V \to A$ be a discrete vertex attribute (for instance, degree, color, or label), where A is a finite set. For each $a \in A$, let*

$$r_a = \big|\{\, v \in V : X(v) = a \}\big| \quad \textit{and} \quad p(a) = \frac{r_a}{|V|}.$$

Then the attribute-based entropy *of G with respect to X is*

$$H(X) \;=\; -\sum_{a \in A} p(a) \log_2 p(a).$$

This generalized entropy captures the variability of the chosen graph property X.

Next, we will discuss Entropy Tree and Entropy Treewidth. The following includes some conceptual ideas, but the definitions are provided.

Definition 4.6 (Bag Entropy)**.** *Let $G = (V,E)$ be a simple undirected graph and $B \subseteq V$ a nonempty vertex subset. Write $G[B]$ for the subgraph induced by B. For each $v \in B$, let*

$$d_B(v) \;=\; \big|\{\, w \in B : \{v,w\} \in E \}\big| \quad \textit{and} \quad S_B \;=\; \sum_{u \in B} d_B(u).$$

If $S_B > 0$, define a probability distribution

$$p_v \;=\; \frac{d_B(v)}{S_B}, \quad v \in B,$$

and the entropy *of the bag B by the Shannon formula*

$$H(B) \;=\; -\sum_{v \in B} p_v \log_2(p_v).$$

If $S_B = 0$ (i.e. $G[B]$ has no edges), set $H(B) = 0$.

Definition 4.7 (Entropy Tree Decomposition)**.** *An* entropy tree decomposition *of G is a standard tree decomposition $(\mathscr{T} = (I,F), \{B_i\}_{i \in I})$ satisfying:*

1. $\bigcup_{i\in I} B_i = V$.
2. *For every* $\{u,v\} \in E$, *there is some* $i \in I$ *with* $\{u,v\} \subseteq B_i$.
3. *For all* $i, j, k \in I$, *if node* k *lies on the unique* $\mathscr{T}$*-path between* i *and* j, *then* $B_i \cap B_j \subseteq B_k$.

Its entropy width *is*

$$\max_{i\in I} H(B_i).$$

The entropy tree-width *of* G *is*

$$\mathrm{tw}_{\mathrm{entropy}}(G) \;=\; \min_{\substack{(\mathscr{T},\{B_i\}) \\ \textit{tree decompositions}}} \big(\max_{i\in I} H(B_i)\big).$$

Example 4.8 (Entropy Tree Decomposition of a Triangle with a Pendant Vertex)**.** *Let* $G = (V,E)$ *be the graph with*

$$V = \{v_1, v_2, v_3, v_4\}, \quad E = \{\{v_1,v_2\},\{v_2,v_3\},\{v_3,v_1\},\{v_3,v_4\}\}.$$

We construct an entropy tree decomposition $(\mathscr{T}, \{B_i\})$ *as follows:*

- *The tree* $\mathscr{T} = (I,F)$ *has two nodes* $I = \{a,b\}$ *and one edge* $F = \{(a,b)\}$.
- *The bag collection is*

$$B_a = \{v_1, v_2, v_3\}, \quad B_b = \{v_3, v_4\}.$$

One checks that:

1. $\bigcup_{i\in\{a,b\}} B_i = V$.
2. *Each edge of* G *is contained in at least one bag:* $\{v_1,v_2\},\{v_2,v_3\},\{v_3,v_1\} \subseteq B_a$ *and* $\{v_3,v_4\} \subseteq B_b$.
3. *The running intersection property holds since* $B_a \cap B_b = \{v_3\} \subseteq B_b$.

We compute the Shannon entropy $H(B)$ *of each bag* B *using*

$$d_B(v) = |\{w \in B : \{v,w\} \in E\}|, \quad S_B = \sum_{u\in B} d_B(u), \quad H(B) = -\sum_{v\in B} \frac{d_B(v)}{S_B} \log_2\!\big(\tfrac{d_B(v)}{S_B}\big).$$

- *For* $B_a = \{v_1,v_2,v_3\}$, *the induced subgraph is a triangle:* $d_{B_a}(v_i) = 2$ *for* $i = 1,2,3$, *so* $S_{B_a} = 6$ *and* $\frac{d}{S} = 1/3$. *Thus*

$$H(B_a) = -3\big(\tfrac{1}{3}\log_2\tfrac{1}{3}\big) = \log_2 3 \approx 1.585 \textit{ bits}.$$

- *For* $B_b = \{v_3,v_4\}$, *the induced subgraph is a single edge:* $d_{B_b}(v_3) = d_{B_b}(v_4) = 1$, *so* $S_{B_b} = 2$ *and* $\frac{d}{S} = 1/2$. *Hence*

$$H(B_b) = -2\big(\tfrac{1}{2}\log_2\tfrac{1}{2}\big) = 1 \textit{ bit}.$$

The entropy width *of this decomposition is*

$$\max\{H(B_a), H(B_b)\} = \max\{\log_2 3,\, 1\} = \log_2 3.$$

Accordingly, the entropy tree-width of G *satisfies*

$$\mathrm{tw}_{\mathrm{entropy}}(G) \;\le\; \log_2 3.$$

One can show no decomposition achieves a smaller maximum bag entropy, so $\mathrm{tw}_{\mathrm{entropy}}(G) = \log_2 3$.

Example 4.9 (Entropy Tree Decomposition of a 4-Vertex Path)**.** *Let* $G = (V,E)$ *be the simple path on four vertices:*

$$V = \{v_1, v_2, v_3, v_4\}, \quad E = \{\{v_1,v_2\},\{v_2,v_3\},\{v_3,v_4\}\}.$$

We define a tree decomposition $(\mathscr{T}, \{B_i\})$ *and compute its entropy width.*

- *The tree* $\mathscr{T} = (I,F)$ *has three nodes* $I = \{a,b,c\}$ *and edges*

$$F = \{(a,b),(b,c)\}.$$

- *The bags are*

$$B_a = \{v_1,v_2\}, \quad B_b = \{v_2,v_3\}, \quad B_c = \{v_3,v_4\}.$$

One verifies:

1. $\bigcup_{i\in\{a,b,c\}} B_i = V$.
2. *Each edge lies in one bag:* $\{v_1,v_2\} \subseteq B_a$, $\{v_2,v_3\} \subseteq B_b$, $\{v_3,v_4\} \subseteq B_c$.
3. *The running-intersection property holds since* $B_a \cap B_b = \{v_2\} \subseteq B_b$ *and* $B_b \cap B_c = \{v_3\} \subseteq B_b$.

We compute the Shannon entropy $H(B)$ for each bag B. In any bag of two adjacent vertices, the induced subgraph is a single edge, so each vertex has degree 1 and

$$S_B = d_B(u) + d_B(v) = 1 + 1 = 2, \quad p_u = p_v = \tfrac{1}{2}, \quad H(B) = -2\Big(\tfrac{1}{2}\log_2 \tfrac{1}{2}\Big) = 1 \text{ bit}.$$

Hence

$$H(B_a) = H(B_b) = H(B_c) = 1.$$

The entropy width *of this decomposition is*

$$\max\{H(B_a), H(B_b), H(B_c)\} = 1.$$

Since no decomposition of G can have all bag entropies below 1 (each edge forces a bag of two vertices), we conclude

$$\mathrm{tw}_{\mathrm{entropy}}(G) = 1.$$

Theorem 4.10. *For any graph G,*

$$\mathrm{tw}_{\mathrm{entropy}}(G) \;\le\; \log_2\big(\mathrm{tw}(G)+1\big).$$

Proof. Let $(\mathscr{T}, \{B_i\})$ be a decomposition realizing $\mathrm{tw}(G)$, so $|B_i| \le \mathrm{tw}(G)+1$ for all i. In each bag B_i, if $S_{B_i} > 0$ then the Shannon bound $H(B_i) \le \log_2 |B_i|$ applies; if $S_{B_i} = 0$ then $H(B_i) = 0 \le \log_2 |B_i|$. Hence $\max_i H(B_i) \le \log_2(\mathrm{tw}(G)+1)$. Taking the minimum over all decompositions gives the result. □

Corollary 4.11. *For any graph G,*

$$\mathrm{tw}(G) \;\le\; 2^{\mathrm{tw}_{\mathrm{entropy}}(G)} \;-\; 1.$$

Proof. From $H(B_i) \le \log_2 |B_i|$ we get $|B_i| \le 2^{H(B_i)}$. Hence for any entropy-optimal decomposition, $\max_i |B_i| \le 2^{\max_i H(B_i)} = 2^{\mathrm{tw}_{\mathrm{entropy}}(G)}$. Therefore $\mathrm{tw}(G) = \min \max_i (|B_i| - 1) \le 2^{\mathrm{tw}_{\mathrm{entropy}}(G)} - 1$. □

Theorem 4.12 (Subgraph Monotonicity)**.** *If $H = (V', E')$ is an induced subgraph of $G = (V, E)$ (so $V' \subseteq V$ and $E' = E \cap \binom{V'}{2}$), then*

$$\mathrm{tw}_{\mathrm{entropy}}(H) \;\le\; \mathrm{tw}_{\mathrm{entropy}}(G).$$

Proof. Let $(\mathscr{T}, \{B_i\}_{i \in I})$ be an entropy tree decomposition of G achieving width $\mathrm{tw}_{\mathrm{entropy}}(G)$. For each bag B_i, form $B_i' = B_i \cap V'$. Remove any i for which $B_i' = \emptyset$, and keep the same tree $\mathscr{T}$. The resulting $(\mathscr{T}, \{B_i'\})$ is a valid entropy tree decomposition of H, because:

- $\bigcup_i B_i' = V'$.
- Every edge of H lies in some original bag, and hence in the corresponding restricted bag.
- The running-intersection property is preserved under intersection.

Moreover, since $H(B_i') \le H(B_i)$ for each i, the maximum bag entropy cannot increase. Therefore $\mathrm{tw}_{\mathrm{entropy}}(H) \le \max_i H(B_i') \le \max_i H(B_i) = \mathrm{tw}_{\mathrm{entropy}}(G)$. □

Theorem 4.13 (Complete Graph Entropy Tree-width)**.** *For the complete graph K_n on n vertices,*

$$\mathrm{tw}_{\mathrm{entropy}}(K_n) = \log_2 n.$$

Proof. Since K_n is a clique of size n, any tree decomposition must contain a bag B with $|B| \ge n$. The unique optimal decomposition takes a single bag $B = V(K_n)$. In $G[B] = K_n$, each vertex has degree $n-1$, so

$$p_v = \frac{n-1}{n(n-1)} = \frac{1}{n}, \quad H(B) = -n\Big(\tfrac{1}{n}\log_2 \tfrac{1}{n}\Big) = \log_2 n.$$

No other decomposition can lower the maximum bag entropy, and by Theorem **??** we have $\mathrm{tw}_{\mathrm{entropy}}(K_n) \le \log_2(\mathrm{tw}(K_n)+1) = \log_2 n$. Thus equality holds. □

Theorem 4.14 (Zero-Edge Characterization)**.** *For any finite simple graph $G = (V, E)$,*

$$\mathrm{tw}_{\mathrm{entropy}}(G) = 0 \quad \Longleftrightarrow \quad E = \varnothing.$$

Proof. ($\Rightarrow$) If $\mathrm{tw}_{\mathrm{entropy}}(G) = 0$, then there exists an entropy tree decomposition whose maximum bag entropy is 0. By Definition, a bag B has $H(B) = 0$ only if $G[B]$ has no edges. Since every edge of G must lie in some bag, G cannot have any edges, so $E = \varnothing$.
($\Leftarrow$) If $E = \varnothing$, then G is edgeless. Take any tree decomposition—e.g. a single bag $B = V$. Since $G[B]$ has no edges, $H(B) = 0$. Thus $\mathrm{tw}_{\mathrm{entropy}}(G) = 0$. □

Theorem 4.15 (Entropy Tree-width of Forests)**.** *Let F be a forest with at least one edge. Then*

$$\mathrm{tw}_{\mathrm{entropy}}(F) = 1.$$

Proof. Because F is a forest, its classical tree-width $\mathrm{tw}(F) = 1$. By Theorem **??**, $\mathrm{tw}_{\mathrm{entropy}}(F) \le \log_2(\mathrm{tw}(F)+1) = \log_2 2 = 1$.

On the other hand, any entropy tree decomposition of F must cover each edge $\{u,v\}$ in some bag B. Such a bag satisfies $G[B] \cong K_2$, so its entropy is

$$H(B) = -2\left(\tfrac{1}{2}\log_2 \tfrac{1}{2}\right) = 1.$$

Hence the maximum bag entropy in any decomposition is at least 1, implying $\mathrm{tw}_{\mathrm{entropy}}(F) \ge 1$. Combining the two inequalities yields the claim. □

Theorem 4.16 (Disjoint Union Property)**.** *If $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are graphs with $V_1 \cap V_2 = \varnothing$, then their disjoint union $G = G_1 \cup G_2$ satisfies*

$$\mathrm{tw}_{\mathrm{entropy}}(G) \;=\; \max\{\mathrm{tw}_{\mathrm{entropy}}(G_1), \mathrm{tw}_{\mathrm{entropy}}(G_2)\}.$$

Proof. First, by Theorem 4.12, each G_i is an induced subgraph of G, so $\mathrm{tw}_{\mathrm{entropy}}(G) \ge \mathrm{tw}_{\mathrm{entropy}}(G_i)$ for $i = 1, 2$. Therefore

$$\mathrm{tw}_{\mathrm{entropy}}(G) \;\ge\; \max\{\mathrm{tw}_{\mathrm{entropy}}(G_1), \mathrm{tw}_{\mathrm{entropy}}(G_2)\}.$$

Conversely, let $(\mathscr{T}_i, \{B_j^{(i)}\})$ be entropy-optimal decompositions of G_i of width $\mathrm{tw}_{\mathrm{entropy}}(G_i)$. Construct a decomposition of G by taking the disjoint union of the trees $\mathscr{T}_1$ and $\mathscr{T}_2$, with their bags unchanged. This yields a valid entropy tree decomposition of G, because no bag needs to cover edges between V_1 and V_2. Its maximum bag entropy is the larger of the two individual widths. Hence

$$\mathrm{tw}_{\mathrm{entropy}}(G) \;\le\; \max\{\mathrm{tw}_{\mathrm{entropy}}(G_1), \mathrm{tw}_{\mathrm{entropy}}(G_2)\}.$$

Combining the two inequalities proves the theorem. □

4.3. Tree-width for Factor graph

A *factor graph* is a type of bipartite graph that represents the factorization of a global function into a product of local functions. Factor graphs are used in various fields, including Markov networks and Bayesian networks, and have been the subject of extensive research[168].

The formal definition is provided below.

Definition 4.17 (Hypergraph)**.** *A* hypergraph *is a pair $H = (V, \mathscr{E})$, where:*

- *V is a finite nonempty set of elements called* vertices*,*
- *$\mathscr{E} \subseteq \mathscr{P}(V) \setminus \{\emptyset\}$ is a finite set of nonempty subsets of V, called* hyperedges*.*

Each hyperedge $e \in \mathscr{E}$ may connect any number of vertices. In particular, if all hyperedges have cardinality two, then H reduces to a standard undirected graph.

Definition 4.18 (Factor graph)**.** *Formally, a factor graph is a hypergraph $G = (V, E, att, lab_V, lab_E)$ together with mappings Ω and F, where:*

- *V is the set of variable nodes.*
- *E is the set of factor nodes (edges).*
- *att is the attachment function, which associates each factor $e \in E$ with a set of variable nodes $v_1, \ldots, v_k \in V$.*
- *lab_V and lab_E are labeling functions that assign labels to variable nodes and factor nodes, respectively.*
- *Ω maps node labels to sets of possible values. For a node $v \in V$, $\Omega(v)$ represents the set of values that v can take, and it is defined as $\Omega(v) = \Omega(lab_V(v))$.*
- *F maps edge labels to functions. For an edge $e \in E$, the corresponding function $F(e)$ is defined as:*

  $$F(e) : \Omega(v_1) \times \cdots \times \Omega(v_k) \to \mathbb{R}_{\ge 0},$$

  where $v_1, \ldots, v_k$ are the variables connected to e.

In a factor graph:

- *A node $v \in V$ together with its domain $\Omega(v)$ is called a variable.*
- *An edge $e \in E$ together with its function $F(e)$ is called a factor.*

A variable node $v \in V$ is typically drawn as a circle, while a factor node $e \in E$ is drawn as a small square. Instead of writing the label $lab_E(e)$, the function $F(e)$ is written next to the factor node as an expression in terms of its neighboring variable nodes.

Example 4.19 (Factor Graph for a Simple Binary Markov Chain)**.** *A Binary Markov Chain is a stochastic process with two states, where the next state depends only on the current state (cf.[169, 170]). We model the joint distribution of three binary random variables X, Y, and Z as a factor graph $G = (V, E, \mathrm{att}, \mathrm{lab}_V, \mathrm{lab}_E)$ with mappings Ω and F defined below.*

- *Variable nodes:*

  $$V = \{X, Y, Z\}.$$

  Each variable has label $\mathrm{lab}_V(v) = \texttt{Binary}$ and domain $\Omega(v) = \{0, 1\}$.
- *Factor nodes:*

  $$E = \{f_1, f_2, f_3\},$$

  with labels $\mathrm{lab}_E(f_1) = \varphi_1$, $\mathrm{lab}_E(f_2) = \varphi_2$, and $\mathrm{lab}_E(f_3) = \varphi_3$.

- *Attachment function* att*:*

$$\mathrm{att}(f_1) = \{X\}, \quad \mathrm{att}(f_2) = \{X, Y\}, \quad \mathrm{att}(f_3) = \{Y, Z\}.$$

- *Factor functions* F*:*

$$F(f_1) : \{0,1\} \longrightarrow \mathbb{R}_{\geq 0}, \qquad F(f_1)(x) = \begin{cases} 0.8, & x = 0, \\ 0.2, & x = 1, \end{cases}$$

$$F(f_2) : \{0,1\}^2 \longrightarrow \mathbb{R}_{\geq 0}, \qquad F(f_2)(x,y) = \begin{cases} 0.9, & x = y, \\ 0.1, & x \neq y, \end{cases}$$

$$F(f_3) : \{0,1\}^2 \longrightarrow \mathbb{R}_{\geq 0}, \qquad F(f_3)(y,z) = \begin{cases} 0.7, & y = z, \\ 0.3, & y \neq z. \end{cases}$$

In this factor graph:

- *Each circle* $v \in V$ *denotes a variable with domain* $\Omega(v)$.
- *Each square* $f \in E$ *denotes a factor computing the nonnegative function* $F(f)$ *on the values of its neighboring variable nodes.*

Hence the joint distribution factorizes as

$$P(X,Y,Z) \;=\; F(f_1)(X) \;\times\; F(f_2)(X,Y) \;\times\; F(f_3)(Y,Z).$$

Tree-width is also utilized in factor graphs [171–173]. In practical applications, further investigations could focus on related parameters such as pathwidth, cyclewidth, and starwidth, as necessary.

4.4. Tree-width in Bond Graphs

Bond graphs are a domain-independent formalism for modeling the transfer and storage of energy in multi-domain physical systems [174–176]. They consist of two kinds of vertices—*element nodes* and *junction nodes*—connected by *bonds* carrying conjugate variables effort e and flow f.

Definition 4.20 (Bond Graph)**.** *[174–176] A* bond graph *is an undirected graph*

$$G = (V, E),$$

where

- $V = V_{\text{elem}} \dot{\cup} V_{\text{junc}}$*, a disjoint union of*
  - Element nodes $V_{\text{elem}} = V_{Se} \dot{\cup} V_{Sf} \dot{\cup} V_R \dot{\cup} V_C \dot{\cup} V_I \dot{\cup} V_{TF} \dot{\cup} V_{GY}$,
  - Junction nodes $V_{\text{junc}} = V_0 \dot{\cup} V_1$,
- $E \subseteq \big\{\{u,v\} : u,v \in V,\ u \neq v\big\}$ *is the set of* bonds*, each representing a single power port connection.*

Each bond $\{u,v\} \in E$ *carries two variables:*

$$e \text{ (effort)}, \quad f \text{ (flow)}, \quad \text{with instantaneous power } P = e\,f.$$

Element nodes denote:

- *Se: effort source (e.g. voltage, force),*
- *Sf: flow source (e.g. current, velocity),*
- *R: resistance (energy dissipation),*
- *C: capacitance (potential energy storage),*
- *I: inertia (kinetic energy storage),*
- *TF: transformer (scaling of effort and flow),*
- *GY: gyrator (cross-domain conversion of effort and flow).*

Junction nodes denote:

- 0*-junction: common effort, flows sum to zero,*
- 1*-junction: common flow, efforts sum to zero.*

Example 4.21 (Bond Graph of a Series RLC Circuit)**.** *An RLC circuit is an electrical circuit consisting of a resistor (R), inductor (L), and capacitor (C), exhibiting oscillatory behavior (cf.[177–179]). Consider the electrical series circuit consisting of a DC voltage source* V_0*, a resistor of resistance R, an inductor of inductance L, and a capacitor of capacitance C. Its bond graph* $G = (V,E)$ *is defined as follows.*

$$V \;=\; \{\, Se_{V_0},\, R_R,\, I_L,\, C_C,\, J_1 \,\},$$

where

- Se_{V_0} *is an* effort source *imposing constant voltage* $e_{Se} = V_0$.
- R_R *is a* resistor *element with constitutive relation* $e_R = R\,f_R$.
- I_L *is an* inertia *element (inductor) with* $e_I = L\,\frac{df_I}{dt}$.

- C_C *is a* capacitance *element with* $f_C = C\frac{de_C}{dt}$.
- J_1 *is a* 1-junction *enforcing equal flows and summing efforts.*

The set of bonds is

$$E \;=\; \big\{\{Se_{V_0},J_1\},\,\{R_R,J_1\},\,\{I_L,J_1\},\,\{C_C,J_1\}\big\}.$$

Each bond $\{X,J_1\}$ *carries two variables:*

$$e_X,\; f_X, \qquad \text{with instantaneous power } P_X = e_X\, f_X.$$

At the 1*-junction* J_1*, the following conditions hold:*

$$f_{Se} = f_R = f_I = f_C = i(t), \qquad e_{Se} \;-\; e_R \;-\; e_I \;-\; e_C = 0.$$

Hence the circuit equations are

$$i(t) = f_X \quad (\text{common current}), \quad V_0 \;-\; R\,i(t) \;-\; L\frac{di}{dt} \;-\; \frac{1}{C}\int i(t)\,dt = 0.$$

This bond graph compactly captures the energy flow and storage in the series RLC circuit.

Definition 4.22 (Bond Tree Decomposition)**.** *Let* $G = (V,E)$ *be the underlying undirected graph of a bond graph. A* bond tree decomposition *of* G *is a tree decomposition* $\big(\mathscr{T} = (I,F), \{B_i\}_{i\in I}\big)$ *satisfying:*

1. $\bigcup_{i\in I} B_i = V$.
2. *For every bond* $\{u,v\} \in E$*, there exists* $i \in I$ *with* $\{u,v\} \subseteq B_i$.
3. *For any* $i,j,k \in I$*, if node* k *lies on the unique* $\mathscr{T}$*-path between* i *and* j*, then*

$$B_i \cap B_j \;\subseteq\; B_k.$$

Each bag B_i *represents a subsystem of components whose energy interactions are considered jointly.*

Definition 4.23 (Bond Tree-width)**.** *The* width *of a bond tree decomposition* $(\mathscr{T},\{B_i\})$ *is*

$$\text{width}(\mathscr{T},\{B_i\}) = \max_{i\in I}\big(|B_i|-1\big).$$

The bond tree-width *of* G *is the minimum such width over all bond tree decompositions:*

$$\text{tw}_{\text{bond}}(G) = \min_{(\mathscr{T},\{B_i\})}\max_{i\in I}\big(|B_i|-1\big).$$

Example 4.24 (Bond Tree-width of a Series RLC Bond Graph)**.** *Consider the bond graph* $G = (V,E)$ *of the series RLC circuit introduced earlier, whose underlying undirected graph has*

$$V = \{Se,R,I,C,J_1\}, \quad E = \{\{Se,J_1\},\{R,J_1\},\{I,J_1\},\{C,J_1\}\}.$$

Since G *is a tree, its (standard) tree-width is 1. We now exhibit a* bond tree decomposition *of* G *that achieves this width.*

- *Let the index set of bags be* $I = \{a,b,c,d\}$*, and define the tree* $\mathscr{T} = (I,F)$ *by*

$$F = \{(a,b),(b,c),(c,d)\}.$$

- *Assign bags as follows:*

$$B_a = \{Se,J_1\}, \quad B_b = \{R,J_1\}, \quad B_c = \{I,J_1\}, \quad B_d = \{C,J_1\}.$$

We verify the three decomposition properties:

1. $\bigcup_{i\in\{a,b,c,d\}} B_i = \{Se,R,I,C,J_1\} = V$.
2. *Each bond* $\{X,J_1\}$ *with* $X \in \{Se,R,I,C\}$ *is contained in the corresponding bag:* $\{Se,J_1\} \subseteq B_a$*,* $\{R,J_1\} \subseteq B_b$*, etc.*
3. *For any two bags* B_i,B_j*, their intersection is* $\{J_1\}$*. Along the unique path in* $\mathscr{T}$ *between* i *and* j*, every intermediate bag contains* J_1*. Thus* $B_i \cap B_j \subseteq B_k$ *whenever* k *lies on that path.*

Since each bag has size $|B_i| = 2$*, the width of this decomposition is*

$$\max_{i\in\{a,b,c,d\}}(|B_i|-1) = 2-1 = 1.$$

Therefore, the bond tree-width *of* G *is*

$$\text{tw}_{\text{bond}}(G) = 1.$$

Example 4.25 (Bond Tree-width of a Transformer-Coupled Resistive Network)**.** *A resistive network is an electrical circuit composed solely of resistors and sources, used to analyze voltage, current, and power distribution (cf.[180–182]). Consider the bond graph $G = (V,E)$ modeling two resistive circuits coupled by an ideal transformer:*

$$V = \{Se_1, R_1, J_{1a}, TF, J_{1b}, R_2, Se_2\},$$

where

- *Se_1, Se_2 are effort sources imposing fixed voltages e_{Se_1}, e_{Se_2}.*
- *R_1, R_2 are resistors with constitutive relations $e_{R_i} = R_i f_{R_i}$.*
- *J_{1a}, J_{1b} are 1-junctions enforcing equal flow and summing efforts.*
- *TF is an ideal transformer with turns ratio n, satisfying*

$$e_{sec} = n e_{pri}, \quad f_{pri} = n f_{sec}.$$

The bond set is

$$E = \big\{\{Se_1, J_{1a}\}, \{R_1, J_{1a}\}, \{J_{1a}, TF\}, \{TF, J_{1b}\}, \{J_{1b}, R_2\}, \{J_{1b}, Se_2\}\big\}.$$

Since the underlying graph is a tree, its tree-width is 1. We now exhibit a bond tree decomposition of width 1:

- *Index set $I = \{a,b,c,d,e,f\}$ with tree edges $\mathscr{T}$: a–b–c–d–e–f.*
- *Bags:*

$$B_a = \{Se_1, J_{1a}\}, \quad B_b = \{R_1, J_{1a}\}, \quad B_c = \{J_{1a}, TF\},$$
$$B_d = \{TF, J_{1b}\}, \quad B_e = \{J_{1b}, R_2\}, \quad B_f = \{J_{1b}, Se_2\}.$$

Verification:

1. *$\bigcup_{i\in I} B_i = V$.*
2. *Each bond $\{X,Y\} \in E$ is contained in exactly one bag B_i.*
3. *Any two bags intersect in a single junction (J_{1a} or J_{1b}), and along the unique path their intersection appears in every intermediate bag.*

All bags have size 2, so the width is $\max_i(|B_i| - 1) = 1$. Hence

$$\mathrm{tw}_{\mathrm{bond}}(G) = 1.$$

Theorem 4.26. *For any bond graph G, its bond tree-width coincides with the standard tree-width of the underlying graph:*

$$\mathrm{tw}_{\mathrm{bond}}(G) = \mathrm{tw}(G).$$

Proof. Any tree decomposition of the underlying graph G satisfies the bond tree decomposition properties by definition. Conversely, any bond tree decomposition is a valid tree decomposition of G. Hence the minimum widths agree. □

Theorem 4.27 (Equality with Standard Tree-width)**.** *For any bond graph G,*

$$\mathrm{tw}_{\mathrm{bond}}(G) \;=\; \mathrm{tw}(G),$$

where $\mathrm{tw}(G)$ is the usual tree-width of G.

Proof. By definition, every bond tree decomposition of G is in particular a tree decomposition of the underlying graph, so $\mathrm{tw}_{\mathrm{bond}}(G) \geq \mathrm{tw}(G)$. Conversely, any tree decomposition of G satisfies the bond tree decomposition axioms, hence $\mathrm{tw}(G) \geq \mathrm{tw}_{\mathrm{bond}}(G)$. The two inequalities give the claimed equality. □

Theorem 4.28 (Forest Characterization)**.** *A bond graph G is a forest (i.e. has no cycles) if and only if $\mathrm{tw}_{\mathrm{bond}}(G) = 1$.*

Proof. If G is a forest, it is well-known that its standard tree-width $\mathrm{tw}(G) = 1$. By Theorem 1, $\mathrm{tw}_{\mathrm{bond}}(G) = \mathrm{tw}(G) = 1$.
Conversely, if $\mathrm{tw}_{\mathrm{bond}}(G) = 1$, then $\mathrm{tw}(G) = 1$, which implies G is a forest (since any cycle forces tree-width at least 2). Thus G contains no cycles. □

Theorem 4.29 (Subgraph Monotonicity)**.** *If $H = (V',E')$ is any subgraph of $G = (V,E)$ (i.e. $V' \subseteq V$, $E' \subseteq E \cap \binom{V'}{2}$), then*

$$\mathrm{tw}_{\mathrm{bond}}(H) \;\leq\; \mathrm{tw}_{\mathrm{bond}}(G).$$

Proof. Let $(\mathscr{T}, \{B_i\}_{i\in I})$ be a bond tree decomposition of G achieving width $\mathrm{tw}_{\mathrm{bond}}(G)$. Define $B'_i = B_i \cap V'$. Dropping any bag with $B'_i = \emptyset$ and retaining the same tree structure yields a valid bond tree decomposition of H. Since $|B'_i| \leq |B_i|$, its width is at most $\mathrm{tw}_{\mathrm{bond}}(G)$, establishing the inequality. □

Theorem 4.30 (Articulation Gluing Lemma)**.** *Suppose G can be partitioned into two bond subgraphs G_1 and G_2 that meet exactly at a single vertex v; formally*

$$V(G_1) \cap V(G_2) = \{v\}, \quad E(G_1) \cup E(G_2) = E(G), \quad E(G_1) \cap E(G_2) = \emptyset.$$

Then

$$\mathrm{tw}_{\mathrm{bond}}(G) \;=\; \max\big\{\mathrm{tw}_{\mathrm{bond}}(G_1),\, \mathrm{tw}_{\mathrm{bond}}(G_2)\big\}.$$

Proof. Let $w_i = \mathrm{tw}_{\mathrm{bond}}(G_i)$ for $i = 1,2$. By Subgraph Monotonicity, $\mathrm{tw}_{\mathrm{bond}}(G) \geq w_i$, so $\mathrm{tw}_{\mathrm{bond}}(G) \geq \max\{w_1, w_2\}$.

For the reverse inequality, take bond tree decompositions of G_1 and G_2 of widths w_1 and w_2. Each decomposition contains at least one bag that includes the shared vertex v. Connect these two decomposition trees by adding an edge between a v-bag in the first and a v-bag in the second. The resulting structure respects all decomposition properties for G and has width $\max\{w_1, w_2\}$. Hence $\mathrm{tw}_{\mathrm{bond}}(G) \leq \max\{w_1, w_2\}$, completing the proof. □

5. Conclusion and Future tasks

In this paper, we proposed several practical graph width parameters and conducted a review of the applications of existing graph width parameters. Since there are likely many more potential applications for graph width parameters beyond those discussed, further investigation will be pursued. Additionally, while we have examined several width parameters, we plan to validate their effectiveness through mathematical proofs and computational experiments.

Acknowledgments

We extend our sincere gratitude to everyone who provided insights, inspiration, and assistance throughout this research. We particularly thank our readers for their interest and acknowledge the authors of the cited works for laying the foundation that made our study possible. We also appreciate the support from individuals and institutions that provided the resources and infrastructure needed to produce and share this paper. Finally, we are grateful to all those who supported us in various ways during this project.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author Contributions

The paper has been solely authored by the corresponding author at this stage.

Data Availability

This research is purely theoretical, involving no data collection or analysis. We encourage future researchers to pursue empirical investigations to further develop and validate the concepts introduced here.

Ethical Considerations

This work does not involve any experiments or studies involving human participants or animals, and therefore no ethical approvals were required.

Conflicts of Interest

The authors confirm that there are no conflicts of interest related to the research or its publication.

Research Integrity

The authors hereby confirm that, to the best of their knowledge, this manuscript is their original work, has not been published in any other journal, and is not currently under consideration for publication elsewhere at this stage.

Disclaimer (Note on Computational Tools)

No computer-assisted proof, symbolic computation, or automated theorem proving tools (e.g., Mathematica, SageMath, Coq, etc.) were used in the development or verification of the results presented in this paper. All proofs and derivations were carried out manually and analytically by the authors.

Disclaimer (Limitations and Claims)

The theoretical concepts presented in this paper have not yet been subject to practical implementation or empirical validation. Future researchers are invited to explore these ideas in applied or experimental settings. Although every effort has been made to ensure the accuracy of the content and the proper citation of sources, unintentional errors or omissions may persist. Readers should independently verify any referenced materials.

To the best of the authors' knowledge, all mathematical statements and proofs contained herein are correct and have been thoroughly vetted. Should you identify any potential errors or ambiguities, please feel free to contact the authors for clarification.

The results presented are valid only under the specific assumptions and conditions detailed in the manuscript. Extending these findings to broader mathematical structures may require additional research. The opinions and conclusions expressed in this work are those of the authors alone and do not necessarily reflect the official positions of their affiliated institutions.

References

[1] Reinhard Diestel. *Graph theory*. Springer (print edition); Reinhard Diestel (eBooks), 2024.

[2] Alexandru T Balaban. Applications of graph theory in chemistry. *Journal of chemical information and computer sciences*, 25(3): 334–343, 1985.

[3] Vadim Zverovich. *Modern applications of graph theory*. Oxford University Press, 2021.

[4] Basudeb Mondal and Kajal De. An overview applications of graph theory in real field. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 2(5):751–759, 2017.

[5] Haitze J Broersma and Cornelis Hoede. Path graphs. *Journal of graph theory*, 13(4):427–444, 1989.

[6] Ana Figueroa and Eduardo Rivera-Campo. On the tree graph of a connected graph. *Discussiones Mathematicae Graph Theory*, 28(3): 501–510, 2008.

[7] Severino V Gervacio. Cycle graphs. In *Graph Theory Singapore 1983: Proceedings of the First Southeast Asian Graph Theory Colloquium, held in Singapore May 10–28, 1983*, pages 279–293. Springer, 1984.

[8] Lowell W Beineke and Frank Harary. The thickness of the complete graph. *Canadian Journal of Mathematics*, 17:850–859, 1965.

[9] Liliana Alcón, Luerbio Faria, Celina MH de Figueiredo, and Marisa Gutierrez. The complexity of clique graph recognition. *Theoretical Computer Science*, 410(21-23):2072–2083, 2009.

[10] Ronan Le Bras, Carla P Gomes, and Bart Selman. Double-wheel graphs are graceful. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[11] Jenq-Jong Lin, Min-Jen Jou, and Qian-Yu Lin. The k-th largest numbers of maximum independent sets in quasi-forest graphs. *International Journal of Contemporary Mathematical Sciences*, 14(4):237–244, 2019.

[12] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer networks*, 33(1-6):309–320, 2000.

[13] CA Maddra and KA Hawick. Domain modelling and language issues for family history and near-tree graph data applications. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, page 10. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2016.

[14] Yutao Hu, Deqing Zou, Junru Peng, Yueming Wu, Junjie Shan, and Hai Jin. Treecen: Building tree graph for scalable semantic code clone detection. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, 2022.

[15] Alberto Sanfeliu and King-Sun Fu. Tree-graph grammars for pattern recognition. In *International Workshop on Graph Grammars and Their Application to Computer Science*, pages 349–368. Springer, 1982.

[16] Robert Sasak. Comparing 17 graph parameters. Master's thesis, The University of Bergen, 2010.

[17] Joakim Alme Nordstrand. Exploring graph parameters similar to tree-width and path-width. Master's thesis, The University of Bergen, 2017.

[18] M Vijaya and Ms Asha Joyce. Undirected binary fuzzy graphs on composition, tensor and normal products. 2019.

[19] M Vijaya and M Asha Joyce. Operations on interval-valued binary fuzzy graphs. *Journal of Algebraic Statistics*, 13(2):2534–2540, 2022.

[20] John N Mordeson, Sunil Mathew, and Davender S Malik. *Fuzzy graph theory with applications to human trafficking*, volume 365. Springer, 2018.

[21] Rajat Talak, Siyi Hu, Lisa Peng, and Luca Carlone. Neural trees for learning on graphs. *Advances in Neural Information Processing Systems*, 34:26395–26408, 2021.

[22] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.

[23] Wengong Jin. *Neural graph representation learning with application to chemistry*. PhD thesis, Massachusetts Institute of Technology, 2018.

[24] Wang Xiaomeng, Zhang Tao, Wu Runpu, Xin Wei, and Hou Changyu. Cpgva: code property graph based vulnerability analysis by deep learning. In *2018 10th International Conference on Advanced Infocomm Technology (ICAIT)*, pages 184–188. IEEE, 2018.

[25] Yoshihisa Shinagawa and Tosiyasu L Kunii. Constructing a reeb graph automatically from cross sections. *IEEE Computer Graphics and Applications*, 11(06):44–51, 1991.

[26] Robert L Burdett and Erhan Kozan. A disjunctive graph model and framework for constructing new train schedules. *European Journal of Operational Research*, 200(1):85–98, 2010.

[27] Philippe Lacomme, Mohand Larabi, and Nikolay Tchernev. A disjunctive graph for the job-shop with several robots. In *MISTA conference*, volume 20, pages 285–292. Citeseer, 2007.

[28] S Satham Hussain, N Durga, Rahmonlou Hossein, and Ghorai Ganesh. New concepts on quadripartitioned single-valued neutrosophic graph with real-life application. *International Journal of Fuzzy Systems*, 24(3):1515–1529, 2022.

[29] Takaaki Fujita. *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond*. Biblio Publishing, 2025. ISBN 978-1-59973-812-3.

[30] Takaaki Fujita. A short note on the basic graph construction algorithm for plithogenic graphs. *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond*, page 274, 2025.

[31] Takaaki Fujita and Florentin Smarandache. A review of the hierarchy of plithogenic, neutrosophic, and fuzzy graphs: Survey and applications. In *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond (Second Volume)*. Biblio Publishing, 2024. ISBN 978-1-59973-814-7.

[32] Isolde Adler, Tomáš Gavenčiak, and Tereza Klimošová. Hypertree-depth and minors in hypergraphs. *Theoretical Computer Science*, 463:84–95, 2012.

[33] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 21–32, 1999.

[34] Muhammad Saeed, Muhammad Khubab Siddique, Muhammad Ahsan, Muhammad Rayees Ahmad, and Atiqe Ur Rahman. A novel approach to the rudiments of hypersoft graphs. *Theory and Application of Hypersoft Set, Pons Publication House, Brussel*, pages 203–214, 2021.

[35] Muhammad Saeed, Atiqe Ur Rahman, and Muhammad Arshad. A study on some operations and products of neutrosophic hypersoft graphs. *Journal of Applied Mathematics and Computing*, 68(4):2187–2214, 2022.

[36] Takaaki Fujita. General plithogenic soft rough graphs and some related graph classes. *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond*, page 437, .

[37] Muhammad Akram, Hafsa Masood Malik, Sundas Shahzadi, and Florentin Smarandache. Neutrosophic soft rough graphs with application. *Axioms*, 7:14, 2018. URL https://api.semanticscholar.org/CorpusID:4899322.

[38] Takaaki Fujita and Florentin Smarandache. Fundamental computational problems and algorithms for superhypergraphs. In *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond (Second Volume)*. Biblio Publishing, 2024. ISBN 978-1-59973-814-7.

[39] Mohammad Hamidi, Florentin Smarandache, and Elham Davneshvar. Spectrum of superhypergraphs via flows. *Journal of Mathematics*, 2022(1):9158912, 2022.

[40] Florentin Smarandache. *Extension of HyperGraph to n-SuperHyperGraph and to Plithogenic n-SuperHyperGraph, and Extension of HyperAlgebra to n-ary (Classical-/Neutro-/Anti-) HyperAlgebra*. Infinite Study, 2020.

[41] Robert Tarjan. Testing flow graph reducibility. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 96–107, 1973.

[42] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 609–618, 2008.

[43] Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. Flow graph corpus from recipe texts. In *LREC*, pages 2370–2377, 2014.

[44] Vladimir Batagelj and Andrej Mrvar. Some analyses of erdos collaboration graph. *Social networks*, 22(2):173–186, 2000.

[45] Kenan İnce and Ali Karci. Collaboration graph as a new graph definition approach. In *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pages 1–4. IEEE, 2017.

[46] Luca Amaru, Pierre-Emmanuel Gaillardon, and Giovanni De Micheli. Majority-inverter graph: A new paradigm for logic optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(5):806–819, 2015.

[47] Felipe L Machado, Vinicius N Possani, Augusto S Neutzling, Renato P Ribas, and André I Reis. From and-inverter graphs to majority-inverter graphs.

[48] Elmir Dzaka, Dian-Lun Lin, and Tsung-Wei Huang. Parallel and-inverter graph simulation using a task-graph computing system. In *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 923–929. IEEE, 2023.

[49] Cunxi Yu, Maciej Ciesielski, and Alan Mishchenko. Fast algebraic rewriting based on and-inverter graphs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(9):1907–1911, 2017.

[50] Xusheng Zhao, Qiong Dai, Jia Wu, Hao Peng, Mingsheng Liu, Xu Bai, Jianlong Tan, Senzhang Wang, and S Yu Philip. Multi-view tensor graph neural networks through reinforced aggregation. *IEEE Transactions on Knowledge and Data Engineering*, 35(4): 4077–4091, 2022.

[51] Frank Schindler and Adam S Jermyn. Algorithms for tensor network contraction ordering. *Machine Learning: Science and Technology*, 1(3):035001, 2020.

[52] Reinhard Diestel and Sang il Oum. Tangle-tree duality: In graphs, matroids and beyond. *Combinatorica*, 39:879 – 910, 2017. URL https://api.semanticscholar.org/CorpusID:42451064.

[53] Reinhard Diestel, Fabian Hundertmark, and Sahar Lemanczyk. Profiles of separations: in graphs, matroids, and beyond. *Combinatorica*, 39:37–75, 2019.

[54] Neil Robertson and Paul D. Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[55] Hans L Bodlaender. A tourist guide through treewidth. *Acta cybernetica*, 11(1-2):1–21, 1993.

[56] Ton Kloks. *Treewidth: computations and approximations*. Springer, 1994.

[57] Hans L Bodlaender and Arie MCA Koster. Treewidth computations i. upper bounds. *Information and Computation*, 208(3):259–275, 2010.

[58] Hans L Bodlaender, Alexander Grigoriev, and Arie MCA Koster. Treewidth lower bounds with brambles. *Algorithmica*, 51(1):81–98, 2008.

[59] Hans L Bodlaender, Alexander Grigoriev, and Arie MCA Koster. Treewidth lower bounds with brambles. In *Algorithms–ESA 2005: 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005. Proceedings 13*, pages 391–402. Springer, 2005.

[60] Eyal Amir. Approximation algorithms for treewidth. *Algorithmica*, 56:448–479, 2010.

[61] Hans L Bodlaender, Fedor V Fomin, Arie MCA Koster, Dieter Kratsch, and Dimitrios M Thilikos. On exact algorithms for treewidth. *ACM Transactions on Algorithms (TALG)*, 9(1):1–23, 2012.

[62] Hans L Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.

[63] Fedor V Fomin, Serge Gaspers, Saket Saurabh, and Alexey A Stepanov. On two techniques of combining branching and treewidth. *Algorithmica*, 54:181–207, 2009.

[64] Takaaki Fujita. Various properties of various ultrafilters, various graph width parameters, and various connectivity systems. *arXiv preprint arXiv:2408.02299*, 2024.

[65] Olivier Pourret, Patrick Na, Bruce Marcot, et al. *Bayesian networks: a practical guide to applications*. John Wiley & Sons, 2008.

[66] Gal Elidan and Stephen Gould. Learning bounded treewidth bayesian networks. *Advances in neural information processing systems*, 21, 2008.

[67] Siqi Nie, Denis D Mauá, Cassio P De Campos, and Qiang Ji. Advances in learning bayesian networks of bounded treewidth. *Advances in neural information processing systems*, 27, 2014.

[68] Xin Zheng, Yi Wang, Yixin Liu, Ming Li, Miao Zhang, Di Jin, Philip S Yu, and Shirui Pan. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

[69] Floris Geerts and Juan L Reutter. Expressiveness and approximation properties of graph neural networks. *arXiv preprint arXiv:2204.04661*, 2022.

[70] Boyao Li, Alexandar J Thomson, Matthew M Engelhard, and David Page. On neural networks as infinite tree-structured probabilistic graphical models. *arXiv preprint arXiv:2305.17583*, 2023.

[71] Emily Jin, Michael Bronstein, Ismail Ilkan Ceylan, and Matthias Lanzinger. Homomorphism counts for graph neural networks: All about that basis. *arXiv preprint arXiv:2402.08595*, 2024.

[72] Shouheng Li, Dongwoo Kim, and Qing Wang. Generalization of graph neural networks through the lens of homomorphism. *arXiv preprint arXiv:2403.06079*, 2024.

[73] Pablo Barceló, Floris Geerts, Juan Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. *Advances in Neural Information Processing Systems*, 34:25280–25293, 2021.

[74] Yuwang Ji, Qiang Wang, Xuan Li, and Jie Liu. A survey on tensor techniques and applications in machine learning. *IEEE Access*, 7: 162950–162990, 2019. URL https://api.semanticscholar.org/CorpusID:208127944.

[75] Elina Robeva and Anna Seigal. Duality of graphical models and tensor networks. *Information and Inference: A Journal of the IMA*, 8 (2):273–288, 2019.

[76] John Brennan, Momme Allalen, David Brayford, Kenneth Hanley, Luigi Iapichino, Lee J O'Riordan, Myles Doyle, and Niall Moran. Tensor network circuit simulation at exascale. In *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*, pages 20–26. IEEE, 2021.

[77] Yijia Wang, Yuwen Ebony Zhang, Feng Pan, and Pan Zhang. Tensor network message passing. *Physical Review Letters*, 132(11): 117401, 2024.

[78] Eugene F. Dumitrescu, Allison L. Fisher, Timothy D. Goodrich, T. Humble, Blair D. Sullivan, and Andrew L. Wright. Benchmarking treewidth as a practical component of tensor network simulations. *PLoS ONE*, 13, 2018. URL https://api.semanticscholar.org/CorpusID:49665494.

[79] Frank Gurski. Polynomial algorithms for protein similarity search for restricted mrna structures. *Information processing letters*, 105 (5):170–176, 2008.

[80] Hua-Ting Yao, Bertrand Marchand, Sarah J Berkemer, Yann Ponty, and Sebastian Will. Infrared: a declarative tree decomposition-powered framework for bioinformatics. *Algorithms for Molecular Biology*, 19(1):13, 2024.

[81] Jizhen Zhao, Dongsheng Che, and Liming Cai. Comparative pathway annotation with protein-dna interaction and operon information via graph tree decomposition. In *Biocomputing 2007*, pages 496–507. World Scientific, 2007.

[82] Michael R Fellows, Michael T Hallett, and H Todd Wareham. Dna physical mapping: Three ways difficult. In *Algorithms—ESA'93: First Annual European Symposium Bad Honnef, Germany September 30–October 2, 1993 Proceedings 1*, pages 157–168. Springer, 1993.

[83] Donald J Jacobs, Andrew J Rader, Leslie A Kuhn, and Michael F Thorpe. Protein flexibility predictions using graph theory. *Proteins: Structure, Function, and Bioinformatics*, 44(2):150–165, 2001.

[84] Nataša Pržulj. Protein-protein interactions: making sense of networks via graph-theoretic modeling. *Bioessays*, 33(2):115–123, 2011.

[85] Jinbo Xu, Feng Jiao, and Bonnie Berger. A tree-decomposition approach to protein structure prediction. In *2005 IEEE Computational Systems Bioinformatics Conference (CSB'05)*, pages 247–256. IEEE, 2005.

[86] Jian Peng, Raghavendra Hosur, Bonnie Berger, and Jinbo Xu. itreepack: Protein complex side-chain packing by dual decomposition. *arXiv preprint arXiv:1504.05467*, 2015.

[87] Xiaoqiang Huang, Robin Pearce, and Yang Zhang. Faspr: an open-source tool for fast and accurate protein side-chain packing. *Bioinformatics*, 36(12):3758–3765, 2020.

[88] Jinbo Xu. Rapid protein side-chain packing via tree decomposition. In *Annual International Conference on Research in Computational Molecular Biology*, pages 423–439. Springer, 2005.

[89] Chaohui Wang, Nikos Komodakis, and Nikos Paragios. Markov random field modeling, inference & learning in computer vision & image understanding: A survey. *Comput. Vis. Image Underst.*, 117:1610–1627, 2013. URL https://api.semanticscholar.org/CorpusID:103469.

[90] Nathan Srebro. Maximum likelihood bounded tree-width markov networks. *ArXiv*, abs/1301.2311, 2001. URL https://api.semanticscholar.org/CorpusID:5840526.

[91] Fereshteh R Dastjerdi and Liming Cai. Polynomial-time derivation of optimal k-tree topology from markov networks. *arXiv preprint arXiv:2404.05991*, 2024.

[92] Di Chang, Liang Ding, Russell Malmberg, David Robinson, Matthew Wicker, Hongfei Yan, Aaron Martinez, and Liming Cai. Optimal learning of markov k-tree topology. *Journal of Computational Mathematics and Data Science*, 4:100046, 2022.

[93] Percy Liang and Nathan Srebro. Methods and experiments with bounded tree-width markov networks. 2004. URL https://api.semanticscholar.org/CorpusID:18492354.

[94] David R Karger and Nathan Srebro. Learning markov networks: maximum bounded tree-width graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, 2001. URL https://api.semanticscholar.org/CorpusID:2149813.

[95] Aaron B Adcock, Blair D Sullivan, and Michael W Mahoney. Tree-like structure in large social and information networks. In *2013 IEEE 13th international conference on data mining*, pages 1–10. IEEE, 2013.

[96] Sushmita Gupta, Saket Saurabh, and Meirav Zehavi. On treewidth and stable marriage: parameterized algorithms and hardness results (complete characterization). *SIAM Journal on Discrete Mathematics*, 36(1):596–681, 2022.

[97] Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data (extended version). In *International Conference on Database Theory*, 2019. URL https://api.semanticscholar.org/CorpusID:52085583.

[98] Dariusz Dereniowski. From pathwidth to connected pathwidth. *SIAM Journal on Discrete Mathematics*, 26(4):1709–1732, 2012.

[99] Kenta Kitsunai, Yasuaki Kobayashi, and Hisao Tamaki. On the pathwidth of almost semicomplete digraphs. In *Algorithms-ESA 2015: 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 816–827. Springer, 2015.

[100] Jens Gustedt. On the pathwidth of chordal graphs. *Discret. Appl. Math.*, 45(3):233–248, 1993.

[101] Andrew B Kahng, Jens Lienig, Igor L Markov, and Jin Hu. *VLSI physical design: from graph partitioning to timing closure*, volume 312. Springer, 2011.

[102] Rolf H Möhring and Dorothea Wagner. 24 combinatorial topics in vlsi design. 1996.

[103] Rassul Bairamkulov and Eby G Friedman. *Graphs in VLSI*. Springer, 2023.

[104] Tatsuo Ohtsuki, Hajimu Mori, E Kuh, Toshinobu Kashiwabara, and Toshio Fujisawa. One-dimensional logic gate assignment and interval graphs. *IEEE Transactions on Circuits and Systems*, 26(9):675–684, 1979.

[105] Rolf H Möhring, Dorothea Wagner, and Frank Wagner. Vlsi network design. *Handbooks in Operations Research and Management Science*, 8:625–712, 1995.

[106] Rolf H. Möhring. Graph problems related to gate matrix layout and pla folding. 1990. URL https://api.semanticscholar.org/CorpusID:122828426.

[107] Hans Bodlaender, Jens Gustedt, and Jan Arne Telle. Linear-time register allocation for a fixed number of registers. In *SODA*, volume 98, pages 574–583, 1998.

[108] Giovanna Kobus Conrado, Amir Kafshdar Goharshady, and Chun Kit Lam. The bounded pathwidth of control-flow graphs. *Proceedings of the ACM on Programming Languages*, 7:292 – 317, 2023. URL https://api.semanticscholar.org/CorpusID:264144996.

[109] Atsushi Takahashi, Shuichi Ueno, and Yoji Kajitani. Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Discrete Mathematics*, 127(1-3):293–304, 1994.

[110] Atsushi Takahashi, Shuichi Ueno, and Yoji Kajitani. Minimal forbidden minors for the family of graphs with proper-path-width at most two. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 78(12):1828–1839, 1995.

[111] Jun Kawahara. Graph optimization problems and algorithms for dag-type blockchains. In *Advanced Mathematical Science for Mobility Society*, pages 109–124. Springer Nature Singapore Singapore, 2024.

[112] Shoji Kasahara, Jun Kawahara, Shin-ichi Minato, and Jumpei Mori. Dag-pathwidth: graph algorithmic analyses of dag-type blockchain networks. *IEICE TRANSACTIONS on Information and Systems*, 106(3):272–283, 2023.

[113] Claude Berge. *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier, 1984.

[114] Alain Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 1, 2013.

[115] Georg Gottlob, Gianluigi Greco, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: Questions and answers. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 57–74, 2016.

[116] Thomas Eiter and Georg Gottlob. Hypergraph transversal computation and related problems in logic and ai. In *European Workshop on Logics in Artificial Intelligence*, pages 549–564. Springer, 2002.

[117] Luis Ortiz and Mohammad Irfan. Tractable algorithms for approximate nash equilibria in generalized graphical games with tree structure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[118] Lucantonio Ghionna, Luigi Granata, Gianluigi Greco, and Francesco Scarcello. Hypertree decompositions for query optimization. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 36–45. IEEE, 2006.

[119] Susan Tu and Christopher Ré. Duncecap: Query plans using generalized hypertree decompositions. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 2077–2078, 2015.

[120] Gianluigi Greco, Nicola Leone, Francesco Scarcello, and Giorgio Terracina. Structural decomposition methods: Key notions and database applications. *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, pages 253–267, 2018.

[121] Manas Joglekar, Rohan Puttagunta, and Christopher Ré. Aggregations over generalized hypertree decompositions. *arXiv preprint arXiv:1508.07532*, 2015.

[122] Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. Faq: questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 13–28, 2016.

[123] Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. Faq: questions asked frequently. *arXiv preprint arXiv:1504.04044*, 2015.

[124] Manas Joglekar and Christopher Ré. It's all a matter of degree: Using degree information to optimize multiway joins. *arXiv preprint arXiv:1508.01239*, 2015.

[125] Isolde Adler. Tree-width and functional dependencies in databases. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 311–320, 2008.

[126] Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366, 1989.

[127] Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural csp decomposition methods. *Artificial Intelligence*, 124(2):243–282, 2000.

[128] Marc Gyssens, Peter G Jeavons, and David A Cohen. Decomposing constraint satisfaction problems using database techniques. *Artificial intelligence*, 66(1):57–89, 1994.

[129] Marc Gyssens and Jan Paredaens. *A decomposition methodology for cyclic databases*. Springer, 1984.

[130] Georg Gottlob and Reinhard Pichler. Hypergraphs in model checking: Acyclicity and hypertree-width versus clique-width. *SIAM Journal on Computing*, 33(2):351–378, 2004.

[131] Florent Capelli and Oliver Irwin. Direct access for conjunctive queries with negation. *arXiv preprint arXiv:2310.15800*, 2023.

[132] Hubie Chen, Gianluigi Greco, Stefan Mengel, and Francesco Scarcello. Counting solutions to conjunctive queries: Structural and hybrid tractability. *arXiv preprint arXiv:2311.14579*, 2023.

[133] Gianluigi Greco and Francesco Scarcello. Counting solutions to conjunctive queries: structural and hybrid tractability. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 132–143, 2014.

[134] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.

[135] Peter Keevash. Hypergraph turan problems. *Surveys in combinatorics*, 392:83–140, 2011.

[136] Alessia Antelmi, Gennaro Cordasco, Mirko Polato, Vittorio Scarano, Carmine Spagnuolo, and Dingqi Yang. A survey on hypergraph representation learning. *ACM Computing Surveys*, 56(1):1–38, 2023.

[137] Christoph Berkholz and Nicole Schweikardt. Constant delay enumeration with fpt-preprocessing for conjunctive queries of bounded submodular width. *arXiv preprint arXiv:2003.01075*, 2020.

[138] Francesco Scarcello. From hypertree width to submodular width and data-dependent structural decompositions. In *SEBD*, 2018.

[139] Phyllis Z Chinn, Jarmila Chvátalová, Alexander K Dewdney, and Norman E Gibbs. The bandwidth problem for graphs and matrices—a survey. *Journal of Graph Theory*, 6(3):223–254, 1982.

[140] Julia Böttcher, Klaas P Pruessmann, Anusch Taraz, and Andreas Würfl. Bandwidth, treewidth, separators, expansion, and universality. *Electronic Notes in Discrete Mathematics*, 31:91–96, 2008.

[141] Dimitrios M Thilikos, Maria Serna, and Hans L Bodlaender. Cutwidth ii: Algorithms for partial w-trees of bounded degree. *Journal of Algorithms*, 56(1):25–49, 2005.

[142] Neil Robertson and Paul D. Seymour. Graph minors. x. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

[143] Neil Robertson and Paul D Seymour. Graph minors. iv. tree-width and well-quasi-ordering. *Journal of Combinatorial Theory, Series B*, 48(2):227–254, 1990.

[144] Takaaki Fujita. Improved version of short note: Exploring ideals in graph theory. *International Journal of Advanced Multidisciplinary Research and Studies*, 5(2):2043–2050, 2025.

[145] Takaaki Fujita. Ultrafilters and their dual relationship to tree-width in graph theory. *Asian Research Journal of Mathematics*, 21(1): 98–114, 2025.

[146] Neil Robertson and Paul D Seymour. Graph minors. viii. a kuratowski theorem for general surfaces. *Journal of Combinatorial Theory, Series B*, 48(2):255–288, 1990.

[147] Umberto Bertele and Francesco Brioschi. *Nonserial Dynamic Programming*. 1972.

[148] Halin Rudolf. S-functions for graphs. *J. Geom*, 8(1-2):171–186, 1976.

[149] Neil Robertson and Paul D Seymour. Graph minors. v. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1): 92–114, 1986.

[150] Takaaki Fujita and Smarandache Florentin. Some graph parameters for superhypertree-width and neutrosophictree-width. In *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond (Third Volume)*. Biblio Publishing, 2024. ISBN 978-1-59973-815-4.

[151] Takaaki Fujita. Superhyperbranch-width and superhypertree-width. *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond*, page 367, 2025.

[152] Takaaki Fujita. Superhypertree-depth: A structural analysis within superhypergraphs. *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond*, page 11, .

[153] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30:595–608, 2016.

[154] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

[155] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. Pre-training molecular graph representation with 3d geometry. *arXiv preprint arXiv:2110.07728*, 2021.

[156] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

[157] Cayley. Lvii. on the mathematical theory of isomers. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 47(314):444–447, 1874.

[158] Allister Vale. Ethanol. *Medicine*, 35(11):615–616, 2007.

[159] José Goldemberg. Ethanol for a sustainable energy future. *science*, 315(5813):808–810, 2007.

[160] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[161] Xian-bin Ye, Quanlong Guan, Weiqi Luo, Liangda Fang, Zhao-Rong Lai, and Jun Wang. Molecular substructure graph attention network for molecular property identification in drug discovery. *Pattern Recognition*, 128:108659, 2022.

[162] Masatsugu Yamada and Mahito Sugiyama. Molecular graph generation by decomposition and reassembling. *ACS omega*, 8(22): 19575–19586, 2023.

[163] Robert M Gray. *Entropy and information theory*. Springer Science & Business Media, 2011.

[164] Matthias Dehmer. Information processing in complex networks: Graph entropy and information functionals. *Applied Mathematics and Computation*, 201(1-2):82–94, 2008.

[165] Gongxu Luo, Jianxin Li, Jianlin Su, Hao Peng, Carl Yang, Lichao Sun, Philip S Yu, and Lifang He. Graph entropy guided node embedding dimension selection for graph neural networks. *arXiv preprint arXiv:2105.03178*, 2021.

[166] Matthias Dehmer and Abbe Mowshowitz. A history of graph entropy measures. *Information Sciences*, 181(1):57–78, 2011.

[167] J. Korner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *Proceedings of the 6th Prague Conference on Information Theory*, pages 411–425, Prague, Czech Republic, 1973.

[168] WU Xiwei, XIAO Bing, WU Cihang, GUO Yiming, and LI Lingwei. Factor graph based navigation and positioning for control system design: A review. *Chinese Journal of Aeronautics*, 35(5):25–39, 2022.

[169] Yuriy S Kharin and Egor V Vecherko. Detection of embeddings in binary markov chains. *Discrete Mathematics and Applications*, 26 (1):13–29, 2016.

[170] Thomas Harter. Finite-size scaling analysis of percolation in three-dimensional correlated binary markov chain random fields. *Physical Review E-Statistical, Nonlinear, and Soft Matter Physics*, 72(2):026120, 2005.

[171] Per Austrin, Jonah Brown-Cohen, and Johan Håstad. Optimal inapproximability with universal factor graphs. *ACM Transactions on Algorithms*, 2019.

[172] Fritz Obermeyer, Eli Bingham, Martin Jankowiak, Neeraj Pradhan, Justin Chiu, Alexander Rush, and Noah Goodman. Tensor variable elimination for plated factor graphs. In *International Conference on Machine Learning*, pages 4871–4880. PMLR, 2019.

[173] David Chiang and Darcey Riley. Factor graph grammars. *Advances in Neural Information Processing Systems*, 33:6648–6658, 2020.

[174] Henry M Paynter. Analysis and design of engineering systems. *MIT press*, 1961.

[175] Peter J Gawthrop and Geraint P Bevan. Bond-graph modeling. *IEEE Control Systems Magazine*, 27(2):24–45, 2007.

[176] PJ Gawthrop. Bicausal bond graphs. *Simulation series*, 27:83–83, 1994.

[177] Romeo Ortega, Dimitri Jeltsema, and Jacquelien MA Scherpen. Power shaping: A new paradigm for stabilization of nonlinear rlc circuits. *IEEE Transactions on Automatic Control*, 48(10):1762–1767, 2003.

[178] Atsushi Sakurai, Bo Zhao, and Zhuomin M Zhang. Resonant frequency and bandwidth of metamaterial emitters and absorbers predicted by an rlc circuit model. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 149:33–40, 2014.

[179] Huakun Liu, Xiaorong Xie, Chuanyu Zhang, Yu Li, Hui Liu, and Yinghong Hu. Quantitative ssr analysis of series-compensated dfig-based wind farms using aggregated rlc circuit model. *IEEE Transactions on Power Systems*, 32(1):474–483, 2016.

[180] Li Li, Wai Man Au, Kam Man Wan, Sai Ho Wan, Wai Yee Chung, and Kwok Shing Wong. A resistive network model for conductive knitting stitches. *Textile research journal*, 80(10):935–947, 2010.

[181] David I Feinstein. The hexagonal resistive network and the circular approximation. *California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CaltechCSTR*, 1988.

[182] Chung-Kuan Cheng and Ernest S Kuh. Module placement based on resistive network optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 3(3):218–225, 1984.