Parametric Chordal Sparsity in SDP-Based Neural Network Verification

No Author Given

No Institute Given

Abstract. The recent computational advances in artificial intelligence and machine learning impact many future technologies that rely on the use of neural networks. Neural networks have, however, shown to be fragile and non-robust, which is particularly a concern in safety-critical applications. Various approaches for the safety verification of neural networks have been proposed that all face scalability issues for deeper networks. Addressing these scalability issues relies on conservative approximations of the nonlinear activation functions that result in a trade-off between accuracy and efficiency. In this paper, our starting point is the recently proposed DeepSDP framework that relies on solving large-scale semidefinite programs. Using the concept of chordal sparsity, we show how to decompose DeepSDP to obtain a more scalable formulation of DeepSDP. We call this formulation Chordal-DeepSDP and show how Chordal-DeepSDP can induce different levels of sparsity into DeepSDP that allow us to operate in between the extremes of efficiency and accuracy. We provide numerical evaluations showcasing this trade-off in efficiency and accuracy and illustrate the computational advantages of Chordal-DeepSDP over a vanilla DeepSDP implementation.

Keywords: Neural Network Verification · Semidefinite Programming · Chordal Graphs.

1 Introduction

Neural networks are the most common choice of function approximators used in machine learning and artificial intelligence. Their success is well documented in the literature for various applications, e.g., in Go [28], in handwritten character recognition [20], and in autonomous driving [6]. Neural networks are, however, notoriously opaque. The key challenges in analyzing their behavior are two-fold: first, one must appropriately model the nonlinear activation functions; second, the neural network may be very large. In addition, neural networks are often sensitive to input perturbations [13,30], meaning that test-driven methods may insufficiently cover the input domain. It is hence often unclear what a neural network exactly learns, and how certain desirable properties can be verified. This is of particular concern in safety-critical applications like autonomous driving, where the lack of formal verification guarantees poses a serious barrier to adoption. Given a neural network $f : \mathbb{R}^{n_1} \to \mathbb{R}^m$, an input set $X \subseteq \mathbb{R}^{n_1}$, and a safe output set $Y \subseteq \mathbb{R}^m$, we are interested in the *safety verification* problem of checking whether

$$f(X) \subseteq Y.$$

Our starting point in this paper is the DeepSDP framework presented in [9], but with a focus on scalability and on exploring the aforementioned trade-off between efficiency and accuracy. DeepSDP uses quadratic constraints to abstract activation behavior, yielding a convex relaxation of safety verification as a semidefinite program (SDP). However, when neural networks get larger and more complex, solving DeepSDP becomes intractable. Even for smaller neural networks, one may be interested in improving scalability when one has to iteratively solve the SDP during the training process.

We first observe that DeepSDP is not exploiting any sparsity to improve scalability. Similarly to [25], we present sparse formulations of DeepSDP that can be solved more efficiently by decomposing DeepSDP using the concept of chordal decomposition [33, 39]. While [25] presents the sparsest possible formulation of DeepSDP, we show how to systematically induce different levels of sparsity into DeepSDP resulting in a trade-off between efficiency and accuracy. Importantly, our formulation recovers the sparsity pattern of the formulations of [25] and DeepSDP, respectively, at the extreme ends of the spectrum. Additionally, we show how a chordally sparse formulation of DeepSDP can be retained even when safety verification is cast over input-output properties, e.g., the L_2 gain κ of f as $\|f(x_1)\|^2 \leq \kappa \|x_1\|^2$. The contributions of our work are summarized as follows:

- We obtain chordal decompositions of DeepSDP [9] using the notion of chordal sparsity to significantly improve scalability. Notably, we can induce different levels of sparsity that results in a trade-off between efficiency and accuracy.
- We provide exhaustive numerical evaluations and illustrate the computational advantages over DeepSDP. We further show that different levels of sparsity allow to navigate between efficiency and accuracy.
- We provide open source code of our implementation.

1.1 Related Work

The safety verification of neural networks has found broad interest over the past years and there have been various departures to address the problem [2,21]. Great success was achieved with Reluplex [18], which is a satisfiability modulo theories (SMT) based verification method that uses lazy search techniques in the way the nonlinear activation functions are handled. While Reluplex only applies to feedforward neural networks with Rectified Linear Unit (ReLU) activation functions, the technique was extended to fully connected and convolutional neural networks with arbitrary piecewise-linear activation functions with Marabou [19]. Another departure has been the use mixed-integer programming as for instance presented in [11, 22, 31]. While these tools give accurate results for various types of neural networks, and are in fact complete for ReLU activation functions, they are fundamentally hard-to-solve combinatorial problems.

Another way of approaching the problem is by reformulating it as a reachability problem, see e.g., [16,17,32,34,36,37] which has been of particular interest for the safety verification of closed-loop dynamical systems [8]. While heuristics for decision problems over dynamical systems exist, these methods are typically computationally expensive.

Safety verification methods based on abstract interpretation [12, 24, 29, 32] work by soundly over-approximating the input set into a convenient abstract domain, e.g. polytopes or zonotopes, and then propagating the abstracted representation through the network. Then, properties verified on the abstract output domain imply that the same properties hold on the original system. These approaches generally offer good scalability, but one must carefully choose good abstract domains and transformers to preserve accuracy.

With a particular focus on achieving scalable solutions, there have been recent works reformulating the neural network safety verification problem as convex optimization problems. Techniques founded in convex optimization [7,26,27] commonly give convex over-approximations to the activation functions in order to formulate verification as an convex problem. Linear approximations have been presented in [5,35] where particularly [5] is amendable to distributed computing and parallelization via ADMM. In this way, they are similar to abstract interpretation in that the precision of analysis is dependent on the tightness of the over-approximation.

In this work, we investigate sparse formulations of DeepSDP [9]. In particular, we are inspired by an initial investigation into scalability in [25], where the authors demonstrate that a restricted version of DeepSDP satisfies *chordal sparsity.* These forms of sparsity have proven useful in decomposing large-scale optimization problems, and are extensively studied [33, 39, 40, 42] with application in [4, 15, 23]. We extend the SDP formulations in [9, 25] by showing how to induce different levels of sparsity into DeepSDP while retaining the full expressivity of its quadratic safety constraints. This work is a continuation in the line of SDP-based safety verification of neural networks.

2 Background and Problem Formulation

In this section, we state the problem formulation and provide background on DeepSDP and the concept of chordal sparsity.

2.1 Safety Verification of Neural Networks

In this paper, we consider feedforward neural networks $f : \mathbb{R}^{n_1} \to \mathbb{R}^m$ with $K \geq 2$ layers, i.e., $K-1$ hidden layers and one linear output layer. Let $x_1 \in \mathbb{R}^{n_1}$ denote the input to the neural network. The output of the neural network is recursively computed as

$$\begin{aligned} x_{k+1} &\coloneqq \phi(W_k x_k + b_k), \quad k = 1, \dots, K-1 \\ f(x_1) &\coloneqq W_K x_K + b_K, \end{aligned}$$

where W_k and b_k are the weight matrices and bias vectors of the kth layer that are assumed to be of appropriate size. We denote the dimensions of $x_2, \dots, x_K$ by $n_2, \dots, n_K \in \mathbb{N}$. The function $\phi(u) \coloneqq \begin{bmatrix}\varphi(u_1)\ \varphi(u_2) \dots\end{bmatrix}^\top$ is the stack vector of nonlinear activation functions φ, e.g., ReLU or tanh activation functions, that is applied element-wise to the input $u \coloneqq \begin{bmatrix}u_1\ u_2 \dots\end{bmatrix}^\top$. We assume throughout the paper that the same type of activation function is used across all layers. For convenience, we also introduce the notations

$$\mathbf{x} \coloneqq \begin{bmatrix}x_1^\top \dots x_K^\top\end{bmatrix}^\top, \qquad \mathbf{z} \coloneqq \begin{bmatrix}\mathbf{x}^\top\ 1\end{bmatrix}^\top, \qquad E_k\mathbf{z} = x_k, \qquad E_a\mathbf{z} = 1.$$

2.2 DeepSDP

Towards verifying if $f(X) \subseteq Y$, the authors in [9] have presented DeepSDP which is summarized next. The main idea is to use *quadratic constraints* (QCs) to abstract the input set X, the activation functions ϕ, and the output safety set Y. In essence, the safety verification problem is set up as a large *semidefinite program* (SDP) whose satisfiability implies that the network is safe.

Input QC To abstract the input set X into a QC, let $P(\gamma_{\text{in}}) \in \mathbb{S}^{n_1+1}$ be a set of symmetric indefinite matrices parametrized by a vector $\gamma_{\text{in}} \in \mathit{\Gamma}_{\text{in}}$ where $\mathit{\Gamma}_{\text{in}}$ is the set of permissible vectors (see below for an example). Each such matrix $P(\gamma_{\text{in}})$ now has to satisfy the following QC

$$\mathbf{z}^\top \begin{bmatrix}E_1\\E_a\end{bmatrix}^\top P(\gamma_{\text{in}}) \begin{bmatrix}E_1\\E_a\end{bmatrix} \mathbf{z} \geq 0 \text{ for all } E_1\mathbf{z} \in X \tag{1}$$

where we recall that $E_1\mathbf{z} = x_1$ is the input of the neural network. The vector γ_{in} appears linearly in $P(\gamma_{\text{in}})$ and will be a decision variable in DeepSDP. The dimension of γ_{in} depends on the specific abstraction that is chosen. There are various choices of $P(\gamma_{\text{in}})$. We recall the encoding of polytopes from [9] next.

Example 1. Assume that the input set is $X \coloneqq \{x_1 \in \mathbb{R}^n : Hx \leq h\}$ where H and h are of appropriate size. Then, the set X can be abstracted into (1) with

$$P(\gamma_{\text{in}}) \coloneqq \begin{bmatrix}H^\top \mathit{\Lambda} H & -H^\top \mathit{\Lambda} h\\ -h^\top \mathit{\Lambda} H & h^\top \mathit{\Lambda} h\end{bmatrix}$$

where $\mathit{\Lambda}$ is a symmetric and nonnegative matrix where the off-diagonal elements are equal to be the elements of γ_{in} and where the diagonal elements are zero. As $\mathit{\Lambda}$ is nonnegative, we see that $\mathit{\Gamma}_{\text{in}}$ is the set of nonnegative vectors.

Activation QC To abstract the activation function $\phi : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ at the kth layer, let $Q(\gamma_{\text{ac}}) \in \mathbb{S}^{2n_k+1}$ be a set of symmetric indefinite matrices parametrized by a vector $\gamma_{\mathrm{ac}} \in \mathit{\Gamma}_{\mathrm{ac}}$ where $\mathit{\Gamma}_{\mathrm{ac}}$ is the set of permissible vectors. Each such matrix $Q(\gamma_{\mathrm{ac}})$ now has to satisfy the following QC

$$\begin{bmatrix} u \\ \phi(u) \\ 1 \end{bmatrix}^\top Q(\gamma_{\mathrm{ac}}) \begin{bmatrix} u \\ \phi(u) \\ 1 \end{bmatrix} \geq 0 \text{ for all } u \in U \tag{2}$$

where U is the input set of the corresponding layer. The dimension of γ_{ac} depends on the particular activation function and on the specific abstraction that is chosen. It was shown in [9, Prop. 2] that many activation functions φ are either slope-restricted or sector-bounded[1] for which abstractions as in (2) exist.

Example 2. For slope-restricted activation functions φ with sector $[a, b]$, it was shown in [9, Lemma 2] that the stacked activation function $\phi(u) = \begin{bmatrix}\varphi(u_1)\ \varphi(u_2)\ \ldots\end{bmatrix}^\top$ can be abstracted into (2) with

$$Q(\gamma_{\mathrm{ac}}) := \begin{bmatrix} -2abT(\gamma_{\mathrm{ac}}) & (a+b)T(\gamma_{\mathrm{ac}}) & 0 \\ (a+b)T(\gamma_{\mathrm{ac}}) & -2T(\gamma_{\mathrm{ac}}) & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

for $U := \mathbb{R}^{n_k}$ and with parameter

$$T(\gamma_{\mathrm{ac}}) := \sum_{1 \leq i < j \leq n_k} \lambda_{ij} (e_i - e_j)(e_i - e_j)^\top$$

where $e_i \in \mathbb{R}^{n_k}$ is the ith unit vector and $\lambda_{ij} \geq 0$ are elements of the vector γ_{ac}. Consequently, the set $\mathit{\Gamma}_{\mathrm{ac}}$ is the set of nonnegative vectors.

Importantly, the encoding can be done for a whole multi-layered network and not only for a single layer by a simple rewriting trick of the network. If we define

$$A := \begin{bmatrix} W_1 & 0 & \cdots & 0 & 0 \\ 0 & W_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & W_{K-1} & 0 \end{bmatrix}, \quad b := \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{K-1} \end{bmatrix}, \quad B := \begin{bmatrix} 0 & I_{n_2} & 0 & \cdots & 0 \\ 0 & 0 & I_{n_3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I_{n_K} \end{bmatrix}, \tag{3}$$

where I_{n_k} is the identity matrix of dimension n_k, then we can write

$$B\mathbf{x} = \phi(A\mathbf{x} + b).$$

Similarly to before, let $Q(\gamma_{\mathrm{ac}}) \in \mathbb{S}^{1+2\sum_k n_k}$ be a set of symmetric indefinite matrices. Each such matrix $Q_{\mathrm{ac}}(\gamma_{\mathrm{ac}})$ now has to satisfy (2) where the input set is $U := \{A\mathbf{x} + b : x_1 \in X\}$. We can then write this QC equivalently as

$$\mathbf{z}^\top \begin{bmatrix} A & b \\ B & 0 \\ 0 & 1 \end{bmatrix}^\top Q(\gamma_{\mathrm{ac}}) \begin{bmatrix} A & b \\ B & 0 \\ 0 & 1 \end{bmatrix} \mathbf{z} \geq 0 \text{ for all } E_1\mathbf{z} \in X. \tag{4}$$

[1] In the scalar case, a function φ is slope-restriced or sector-bounded with $[a, b]$ if $a \leq \frac{\varphi(u_1) - \varphi(u_2)}{u_1 - u_2} \leq b$ or $a \leq \frac{\varphi(u_1)}{u_1} \leq b$ for $u_1, u_2 \in \mathbb{R}$, respectively.

Safety QC To abstract the safety set Y into a QC, let $S \in \mathbb{S}^{n_1+m+1}$ be a symmetric matrix. We assume that S encodes safety by the QC

$$\mathbf{z}^\top \begin{bmatrix} E_1 \\ E_K \\ E_a \end{bmatrix}^\top \widehat{S} \begin{bmatrix} E_1 \\ E_K \\ E_a \end{bmatrix} \mathbf{z} \leq 0, \;\; \text{where } \widehat{S} := \begin{bmatrix} I & 0 & 0 \\ 0 & W_K & b_K \\ 0 & 0 & 1 \end{bmatrix}^\top S \begin{bmatrix} I & 0 & 0 \\ 0 & W_K & b_K \\ 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

For instance, when $Y := \{y \in \mathbb{R}^m : c^\top y - d \leq 0\}$, then we can pick S as

$$S := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & c \\ 0 & c^\top & -2d \end{bmatrix}.$$

We can also express input-output properties, e.g., let $S := \operatorname{diag}(-\kappa I_{n_1}, I_m, 0)$ for $\kappa \geq 0$ to find an upper bound of the L_2 gain of f on X as $\|f(x_1)\|^2 \leq \kappa \|x_1\|^2$.

Finally, DeepSDP can be defined as the SDP

$$\begin{aligned} \text{find} \quad & \gamma_{\text{in}}, \gamma_{\text{ac}} \in \Gamma_{\text{in}} \times \Gamma_{\text{ac}} \\ \text{subject to} \quad & Z(\gamma_{\text{in}}, \gamma_{\text{ac}}) \preceq 0, \end{aligned} \tag{6}$$

where $Z(\gamma_{\text{in}}, \gamma_{\text{ac}})$ is built from the QCs in (1), (4), and (5) as

$$Z(\gamma_{\text{in}}, \gamma_{\text{ac}}) := \begin{bmatrix} E_1 \\ E_a \end{bmatrix}^\top P(\gamma_{\text{in}}) \begin{bmatrix} E_1 \\ E_a \end{bmatrix} + \begin{bmatrix} A & b \\ B & 0 \\ 0 & 1 \end{bmatrix}^\top Q(\gamma_{\text{ac}}) \begin{bmatrix} A & b \\ B & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} E_1 \\ E_K \\ E_a \end{bmatrix}^\top \widehat{S} \begin{bmatrix} E_1 \\ E_K \\ E_a \end{bmatrix}. \tag{7}$$

We recall the formal safety gurantees of DeepSDP from [9, Theorem] next.

Lemma 1 ([9], Theorem 2). *Suppose that the SDP in* (6) *is feasible, then*

$$\begin{bmatrix} x_1 \\ f(x_1) \\ 1 \end{bmatrix}^\top S \begin{bmatrix} x_1 \\ f(x_1) \\ 1 \end{bmatrix} \leq 0 \;\; \textit{if } x_1 \in X.$$

Unfortunately, the dimension of the matrix $Z(\gamma_{\text{in}}, \gamma_{\text{ac}})$ grows with the size of the neural network f so that solving the SDP in (6) quickly becomes intractable.

2.3 Chordal Sparsity

The notion of chordal sparsity connects chordal graph theory and sparse matrix decomposition [14,33], and is used to efficiently solve large-scale matrix problems like semidefinite programs. In the context of this paper, if the matrix $Z(\gamma_{\text{in}}, \gamma_{\text{ac}})$ in (7) has a chordal sparsity pattern, which we accomplish later in Section 3, then the matrix constraint $Z(\gamma_{\text{in}}, \gamma_{\text{ac}}) \preceq 0$ within the SDP in (6) can be split into an equivalent set of smaller matrix constraints.

Chordal Graphs and Sparse Matrices Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be an undirected graph with vertices $\mathcal{V} = \{1, \dots, n\}$ and a symmetric set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. A subset of vertices $\mathcal{C} \subseteq \mathcal{V}$ is called a *clique* if the graph $\mathcal{G}(\mathcal{C}, \mathcal{E})$ is complete, i.e., all pairs of vertices in $\mathcal{C}$ share an edge. If there does not exist another clique that contains $\mathcal{C}$, then $\mathcal{C}$ is called a *maximal clique*. A *cycle* of length k is a sequence of vertices $v_1, \dots, v_k \subseteq \mathcal{V}$ with $(v_k, v_1) \in \mathcal{E}$ and where each adjacent pair (v_i, v_{i+1}) satisfies $(v_i, v_{i+1}) \in \mathcal{E}$. A *chord* is an edge that connects two nonadjacent vertices in a cycle. We say that a graph is *chordal* if every cycle of length four has at least one chord [33, Chapter 2].

We next define *sparsity pattern* of block symmetric matrices X with blocks X_{ij}. Let therefore $\alpha := (\alpha_1, \dots, \alpha_n)$ be a set of positive integers and let $N := \sum_{i=1}^{n} \alpha_i$. We say that the block symmetric matrix $X \in \mathbb{S}^N$ has α-partitioning, denoted by $X \in \mathbb{S}^N_\alpha$, if each block X_{ij} is an element of $\mathbb{R}^{\alpha_i \times \alpha_j}$. We can now define the set of block symmetric matrices with sparsity pattern $\mathcal{E}$ as

$$\mathbb{S}^N_\alpha(\mathcal{E}) := \{X \in \mathbb{S}^N_\alpha : X_{ij} = X_{ji}^\top = 0 \text{ if } i \neq j \text{ and } (i,j) \notin \mathcal{E}\}.$$

We also define the cone of block symmetric matrices with sparsity pattern $\mathcal{E}$ as

$$\mathbb{S}^N_{\alpha,+}(\mathcal{E}) := \{X \in \mathbb{S}^N_\alpha(\mathcal{E}) : X \succeq 0\}.$$

The close interplay between matrices and graphs is possible because of the dual role of $\mathcal{E}$, which identifies both the sparsity pattern and the edge set. Our particular interest in this work is in exploiting the relation between matrix decompositions and graph decompositions. In order to effectively discuss this, we first introduce the definition of an $\mathcal{E}$-induced graph.

Definition 1 ($\mathcal{E}$-Induced Graph). *Let $\mathcal{V}$ be the set of vertices selected according to the α-partitioning. Given the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, suppose that an α-partitioned block symmetric matrix X can be written as*

$$X = \sum_{(i,j) \in \mathcal{E}} E_i^\top X_{ij} E_j, \tag{8}$$

then the $\mathcal{E}$-induced graph of X is the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

Remark 1. We may view $Z(\gamma_{\text{in}}, \gamma_{\text{ac}})$ as a block symmetric matrix with partition $\alpha = \{n_1, \dots, n_K, 1\}$. Then each $x_k = E_k \mathbf{z}$ corresponds to the kth vertex in the induced graph, and $1 = E_a \mathbf{z}$ corresponds to the last vertex in this ordering.

This definition means that the same matrix may have different induced graphs, which is important because it allows us to generate different graphs via equivalent algebraic expressions. Specifically, we are interested in finding induced chordal graphs that can give a *chordal decomposition* of a matrix as defined subsequently. We first recall an example from [39] for illustration purposes. For the symmetric matrix X in Fig. 1 (left) and sparsity pattern

$$\mathcal{E} := \{(v_1, v_2), (v_1, v_4), (v_2, v_3), (v_2, v_4), (v_3, v_4)\},$$

$$X := \begin{bmatrix} X_{11} & X_{12} & 0 & X_{14} \\ X_{12} & X_{22} & X_{23} & X_{24} \\ 0 & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix}$$

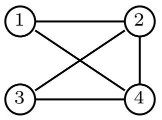

Fig. 1. A symmetric matrix and its induced graph.

Fig. 1 (right) shows the $\mathcal{E}$-*induced graph* of X with vertices $\mathcal{V} := \{v_1, v_2, v_3, v_4\}$. This induced graph has the maximal cliques $\mathcal{C}_1 := \{1, 2, 4\}$ and $\mathcal{C}_2 := \{2, 3, 4\}$.

For a clique $\mathcal{C}_k$ defined by the $\mathcal{E}$-induced graph of X, we can define the block-wise index matrix $E_{\mathcal{C}_k,\alpha} \in \mathbb{R}^{|\mathcal{C}_k|_\alpha \times N}$, where $|\mathcal{C}_k|_\alpha = \sum_{i \in \mathcal{C}_k} \alpha_i$, as

$$(E_{\mathcal{C}_k,\alpha})_{ij} = \begin{cases} I_{\alpha_i}, & \text{if } \mathcal{C}_k(i) = j, \\ 0_{\alpha_i \times \alpha_j}, & \text{otherwise,} \end{cases}$$

where I_{α_i} is the identity matrix of size α_i. When the partitioning is clear, we omit the α subscript to write $\mathcal{C}_{k,\alpha}$ as just $\mathcal{C}_k$. For the example in Fig. 1 and for $X \in \mathbb{S}^4$ and $Y \in \mathbb{S}^3$, we hence have that

$$E_{\mathcal{C}_1} X E_{\mathcal{C}_1}^\top = \begin{bmatrix} X_{11} & X_{12} & X_{14} \\ X_{21} & X_{22} & X_{24} \\ X_{41} & X_{42} & X_{44} \end{bmatrix} \quad \text{and} \quad E_{\mathcal{C}_1}^\top Y E_{\mathcal{C}_1} = \begin{bmatrix} Y_{11} & Y_{12} & 0 & Y_{13} \\ Y_{21} & Y_{22} & 0 & Y_{23} \\ 0 & 0 & 0 & 0 \\ Y_{31} & Y_{32} & 0 & Y_{33} \end{bmatrix}.$$

Definition 2 (Chordal Decomposition of a Matrix). *Let $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and suppose that an α-partitioned block symmetric matrix X can be written as in* (8). *Further suppose that the $\mathcal{E}$-induced graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ of X is chordal with exactly the maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_p$. Then we say that $X_1, \ldots, X_p$ is a chordal decomposition of X with respect to $\mathcal{C}_1, \ldots, \mathcal{C}_p$ if X can be expressed as*

$$X = \sum_{k=1}^{p} E_{\mathcal{C}_k}^\top X_k E_{\mathcal{C}_k}. \tag{9}$$

A matrix may have more than one decomposition, and in fact every matrix is chordally decomposable in a trivial way: take $\mathcal{E} := \mathcal{V} \times \mathcal{V}$, then $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a complete graph, which is chordal and whose only clique is $\mathcal{C}_1 = \mathcal{V}$. The key is to find decompositions that can break a large matrix problem into smaller sub-problems. One important result for this is the following.

Lemma 2 ([39], Theorem 2.17). *Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a chordal graph with maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_p$. Then $X \in \mathbb{S}^N_{\alpha,+}(\mathcal{E})$ if and only if there exist $X_k \in \mathbb{S}_+^{|\mathcal{C}_k|_\alpha}$ such that $X_1, \ldots, X_p$ is a chordal decomposition of X with respect to $\mathcal{C}_1, \ldots, \mathcal{C}_p$.*

This means that if one can write X in the form (9), then solving for $X \succeq 0$ is equivalent to checking that each $X_k \succeq 0$. This transformation is generally desirable if each X_k is small relative to X. Furthermore, Theorem 2 also extends to the case of negative semidefinite matrices.

3 DeepSDP with Parametric Chordal Sparsity Pattern

The SDP in (6) encodes the search space over $\gamma_{\text{in}}, \gamma_{\text{ac}} \in \Gamma_{\text{in}} \times \Gamma_{\text{ac}}$ as a convex problem. The definition of $Q(\gamma_{\text{ac}})$, however, means that $Z(\gamma_{\text{in}}, \gamma_{\text{ac}})$ is in general dense and lacks obvious sparsity patterns that solvers can automatically exploit for performance. As a consequence, DeepSDP quickly encounters scalability issues despite its convexity, and quickly struggles as the network grows larger. It is therefore desirable to construct variants of $Q(\gamma_{\text{ac}})$ that sparsify the structure of $Z(\gamma_{\text{in}}, \gamma_{\text{ac}})$ — allowing for exploitable (chordal) decomposition patterns. A key contribution of this work is to do precisely this.

Remark 2. The authors in [25] construct a sparsified version of $Z(\gamma_{\text{in}}, \gamma_{\text{ac}})$ by dropping the cross-layer dependency of neurons in $Q(\gamma_{\text{ac}})$ and relaxing the expressivity of S in (5). While scalable, the amount of sparsity introduced will necessarily compromise accuracy. We instead show how to parametrically tune the sparsity, thereby achieving a trade-off between scalability and accuracy. Furthermore, our construction allows for the full expressivity of safety constraints.

In Section 3.1 we construct a family of $Q(\gamma_{\text{ac}})$ subject to a sparsity parameter. This leads to a sparsified version of $Z(\gamma_{\text{in}}, \gamma_{\text{ac}})$, but one must still find a *chordal decomposition* in the sense of Definition 2 in order to apply Lemma 2. Such a decomposition is identified in Section 3.2, allowing us to replace the single big semidefinite constraint from DeepSDP with multiple smaller ones.

3.1 Sparse Abstractions of Activation Function via QCs

Our main idea to abstract activation functions is to define a set of smaller QCs instead of the large QC in (4). Note that DeepSDP's QC in (4) enforces that

$$\begin{bmatrix} W_1 x_1 + b_1 \\ \vdots \\ W_{K-1} x_{K-1} + b_{K-1} \\ x_2 \\ \vdots \\ x_K \\ 1 \end{bmatrix}^\top Q(\gamma_{\text{ac}}) \begin{bmatrix} W_1 x_1 + b_1 \\ \vdots \\ W_{K-1} x_{K-1} + b_{K-1} \\ x_2 \\ \vdots \\ x_K \\ 1 \end{bmatrix} \geq 0 \tag{10}$$

We introduce an integer $\beta \geq 1$ to parametrize a set of symmetric matrices $Q_{k,\beta} \in \mathbb{S}^{1+2(n_{k+1}+\ldots+n_{k+\beta})}$ for $k = 1, \ldots, K - \beta$ with the goal to only abstract the activation function between $x_k, \ldots, x_{k+\beta}$ rather than the entire network as in (10). Formally, let

$$\begin{bmatrix} W_k x_k + b_k \\ \vdots \\ W_{k+\beta-1} x_{k+\beta-1} + b_{k+\beta-1} \\ x_{k+1} \\ \vdots \\ x_{k+\beta} \\ 1 \end{bmatrix}^\top Q_{k,\beta}(\gamma_k) \begin{bmatrix} W_k x_k + b_k \\ \vdots \\ W_{k+\beta-1} x_{k+\beta-1} + b_{k+\beta-1} \\ x_{k+1} \\ \vdots \\ x_{k+\beta} \\ 1 \end{bmatrix} \geq 0, \quad (11)$$

for each $k = 1, \ldots, K - \beta$. Note that each $Q_{k,\beta}(\gamma_k)$ with parameters $\gamma_k \in \mathit{\Gamma}_{\mathrm{ac},k}$ can principally be constructed in the same way as $Q(\gamma_{\mathrm{ac}})$, but in a lower dimension. In fact, each parameter γ_k from the set $\mathit{\Gamma}_{\mathrm{ac},k}$ plays the same role as γ_{ac} from the set $\mathit{\Gamma}_{\mathrm{ac}}$, but for this smaller collection of states. Let us, for convenience, define $\gamma_{\mathrm{ac}}^\beta \coloneqq (\gamma_1, \ldots, \gamma_{K-\beta})$ and $\mathit{\Gamma}_{\mathrm{ac}}^\beta \coloneqq \mathit{\Gamma}_{\mathrm{ac},1} \times \ldots \times \mathit{\Gamma}_{\mathrm{ac},K-\beta}$. The sparsity parameter β allows us to define a family of matrices as follows:

$$\begin{aligned} &Z_\beta(\gamma_{\mathrm{in}}, \gamma_{\mathrm{ac}}^\beta) \quad (12) \\ &\coloneqq \begin{bmatrix} E_1 \\ E_a \end{bmatrix}^\top P(\gamma_{\mathrm{in}}) \begin{bmatrix} E_1 \\ E_a \end{bmatrix} + \begin{bmatrix} E_1 \\ E_K \\ E_a \end{bmatrix}^\top \widehat{S} \begin{bmatrix} E_1 \\ E_K \\ E_a \end{bmatrix} + \sum_{k=1}^{K-\beta} \begin{bmatrix} E_{k,\beta} \\ E_a \end{bmatrix}^\top X_{k,\beta}(\gamma_k) \begin{bmatrix} E_{k,\beta} \\ E_a \end{bmatrix} \end{aligned}$$

where $E_{k,\beta}$ is such that $E_{k,\beta}\mathbf{z} = x_k, \ldots, x_{k+\beta}$ and where

$$X_{k,\beta}(\gamma_k) \coloneqq \begin{bmatrix} A_{k,\beta} & b_{k,\beta} \\ B_{k,\beta} & 0 \\ 0 & 1 \end{bmatrix}^\top Q_{k,\beta}(\gamma_k) \begin{bmatrix} A_{k,\beta} & b_{k,\beta} \\ B_{k,\beta} & 0 \\ 0 & 1 \end{bmatrix},$$

with $A_{k,\beta}$, $b_{k,\beta}$, and $B_{k,\beta}$ constructed following the same idea as in (3)

$$A_{k,\beta} \coloneqq \begin{bmatrix} W_k & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & W_{k+\beta-1} & 0 \end{bmatrix}, \; b_{k,\beta} \coloneqq \begin{bmatrix} b_k \\ \vdots \\ b_{k+\beta-1} \end{bmatrix}, \; B_{k,\beta} \coloneqq \begin{bmatrix} 0 & I_{n_{k+1}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_{n_{k+\beta}} \end{bmatrix}.$$

These matrices all have the same partition $\alpha \coloneqq \{n_1, \ldots, n_K, 1\}$, and we refer to the vertex set as $\mathcal{V} \coloneqq \{x_1, \ldots, x_K, x_a\}$ for convenience. To study the sparsity of $Z_\beta(\gamma_{\mathrm{in}}, \gamma_{\mathrm{ac}}^\beta)$, we define $\mathcal{E}_\beta$ by the edge union of the following cliques over $\mathcal{V}$,

$$\begin{aligned} \mathcal{C}_P^\beta \coloneqq \{x_1, x_a\}, \quad \mathcal{C}_S^\beta \coloneqq \{x_1, x_K, x_a\}, \\ \mathcal{C}_{X,k}^\beta \coloneqq \{x_k, \ldots, x_{k+\beta}, x_a\}, \quad k = 1, \ldots, K - \beta. \end{aligned} \quad (13)$$

That is, $(i, j) \in \mathcal{E}_\beta$, if and only if there is a clique $\mathcal{C}$ among (13) such that $i, j \in \mathcal{C}$. Our construction of $\mathcal{E}_\beta$ means that we may write

$$Z_\beta(\gamma_{\mathrm{in}}, \gamma_{\mathrm{ac}}^\beta) = \sum_{(i,j) \in \mathcal{E}_\beta} E_i^\top Z_{ij} E_j,$$

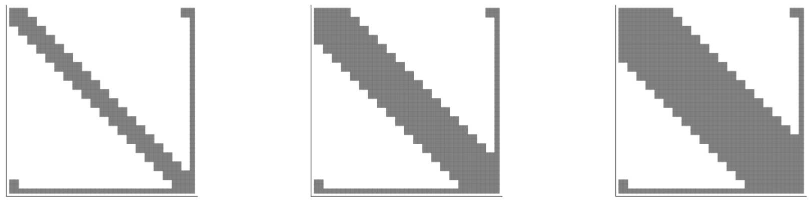

Fig. 2. The sparsity patterns of $\mathcal{E}_\beta$ visualized as an adjacency matrix for $\beta = 1, 3, 5$.

for some Z_{ij} indexed by $\mathcal{E}_\beta$. Consequently, $\mathcal{E}_\beta$ allows us to visualize the sparsity pattern $\mathcal{E}_\beta$ of $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta)$, and we provide an illustration in Fig. 2.

We observe that smaller values of β lead to sparser structures. However, the $\mathcal{E}_\beta$-induced graphs $\mathcal{G}(\mathcal{V}, \mathcal{E}_\beta)$ are not chordal unless β is large. In order to find a chordal decomposition of so that the $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta)$, our approach is to find a *chordal extension* $\mathcal{E}$ of $\mathcal{E}_\beta$, i.e., $\mathcal{E}_\beta \subseteq \mathcal{E}$ and $\mathcal{G}(\mathcal{V}, \mathcal{E})$ chordal — which in turn allows us to find a chordal decomposition of $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta)$.

3.2 A Parametric Chordal Decomposition of $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta)$

We will explicitly construct $\mathcal{E}$ by introducing the following cliques over $\mathcal{V}$,

$$\mathcal{C}_k := \{x_k, \ldots, x_{k+\beta}, x_K, x_a\}, \quad \text{for } k = 1, \ldots, p \text{ where } p = K - \beta - 1, \tag{14}$$

and define $\mathcal{E}$ by their edge union, i.e., $(i, j) \in \mathcal{E}$ if and only if there is a $\mathcal{C}_k$ such that $i, j \in \mathcal{C}_k$. We then show that the following hold:

(1) $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal, and its maximal cliques are precisely $\mathcal{C}_1, \ldots, \mathcal{C}_p$.
(2) Let $\mathcal{E}_\beta$ be defined as in (13), then $\mathcal{E}_\beta \subseteq \mathcal{E}$.

These suffice to show that $\mathcal{E}$ is a chordal extension of $\mathcal{E}_\beta$ so that the $\mathcal{E}$-induced graph of $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta)$ is chordal. Consequently, we may then write a chordal decomposition of $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta)$ using the maximal cliques of $\mathcal{G}(\mathcal{V}, \mathcal{E})$ — which we have constructed to be $\mathcal{C}_1, \ldots, \mathcal{C}_p$. This then sets up an application of Lemma 2, allowing us to decompose the semidefinite constraint $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta) \preceq 0$ into an equivalent set of smaller ones.

Theorem 1. *The graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ defined by the union of $\mathcal{C}_1, \ldots, \mathcal{C}_p$ is chordal, and its maximal cliques are exactly $\mathcal{C}_1, \ldots, \mathcal{C}_p$.*

Proof. Consider a sequence of graphs $\mathcal{G}(\mathcal{V}_1, \mathcal{E}_1), \ldots, \mathcal{G}(\mathcal{V}_p, \mathcal{E}_p)$ constructed by iteratively adding $\mathcal{C}_1, \ldots, \mathcal{C}_p$, such that each $\mathcal{V}_k \subseteq \mathcal{V}_{k+1}$ and $\mathcal{E}_k \subseteq \mathcal{E}_{k+1}$. We show that each graph in this construction is a k-tree [33, Chapter 3.2] for $k = \beta + 2$, which are known to be chordal graphs.

We proceed by induction. In the base case, $\mathcal{V}_1 = \mathcal{C}_1$ and $\mathcal{E}_1 = \mathcal{C}_1 \times \mathcal{C}_1$, and this is a complete graph on $\beta + 3$ vertices, which is a $(\beta + 2)$-tree (the smallest $(\beta + 2)$-tree is the complete graph on $(\beta + 2)$ vertices). Now consider adding $\mathcal{C}_{k+1}$

to $\mathcal{G}(\mathcal{V}_k, \mathcal{E}_k)$. Our ordering means that $x_{k+\beta+1}$ is in $\mathcal{C}_{k+1}$ but not in $\mathcal{V}_k$, and that $(\mathcal{C}_{k+1} \setminus \{x_{k+\beta+1}\}) \subseteq \mathcal{V}_k$. Thus, adding $\mathcal{C}_{k+1}$ to $\mathcal{G}(\mathcal{V}_k, \mathcal{E}_k)$ introduces only $x_{k+\beta+1}$, whose neighborhood is now $\{x_{k+1}, \dots, x_{k+\beta+1}, x_K, x_a\}$, which is a clique of size $\beta + 3$. Thus, $\mathcal{G}(\mathcal{V}_{k+1}, \mathcal{E}_{k+1})$ is also a $(\beta+2)$-tree. $\square$

Next, it remains to show that $\mathcal{E}_\beta \subseteq \mathcal{E}$. An intuition is illustrated in Fig. 3, for which we follow-up with a proof.

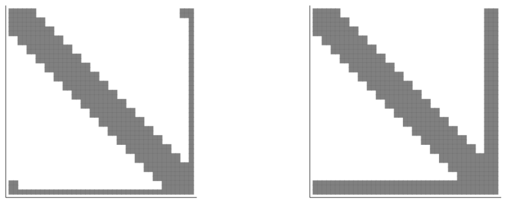

Fig. 3. The adjacency matrices of $\mathcal{E}_\beta$ and $\mathcal{E}$. The $(1, K)$ and $(K, 1)$ blocks in $\mathcal{E}_\beta$ means that $\mathcal{G}(\mathcal{V}, \mathcal{E}_\beta)$ is not chordal; we amend this by filling in the entire Kth row and column (i.e., by adding edges) to get $\mathcal{E}$, such that $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is chordal.

Theorem 2. *Let $\mathcal{E}_\beta$ be defined as in* (13)*, then $\mathcal{E}_\beta \subseteq \mathcal{E}$.*

Proof. Recall that $\mathcal{E}_\beta$ is induced by the edge union of (13), and $\mathcal{E}$ is induced by the edge union of (14). It then suffices to show that each clique in (13) is the subset of a clique in (14). First, $\mathcal{C}_P^\beta \subseteq \mathcal{C}_1$ and $\mathcal{C}_S^\beta \subseteq \mathcal{C}_1$. Furthermore, each $\mathcal{C}_{X,k}^\beta \subseteq \mathcal{C}_k$ for $k = 1, \dots, K - \beta - 1$. Finally, $\mathcal{C}_{X,K-\beta} \subseteq \mathcal{C}_p$. $\square$

Because we have established that $\mathcal{E}$ is a chordal extension of $\mathcal{E}_\beta$ with maximal cliques $\mathcal{C}_1, \dots, \mathcal{C}_p$, we may now express $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}})$ as follows,

$$Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta) = \sum_{k=1}^{p} E_{\mathcal{C}_k}^\top Z_k E_{\mathcal{C}_k} \qquad E_{\mathcal{C}_k} = \begin{bmatrix} E_{k,\beta} \\ E_K \\ E_a \end{bmatrix},$$

for some $Z_1, \dots, Z_p$ — which is a chordal decomposition of $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}})$ with respect to $\mathcal{C}_1, \dots, \mathcal{C}_p$. Then, we use Lemma 2 to conclude an equivalence between the following two problems:

$$\begin{aligned} \text{find} \quad & (\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta) \in \mathit{\Gamma}_{\text{in}} \times \mathit{\Gamma}_{\text{ac}}^\beta \\ \text{subject to} \quad & Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta) \preceq 0 \end{aligned} \tag{15}$$

and

$$\begin{aligned} \text{find} \quad & (\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta) \in \mathit{\Gamma}_{\text{in}} \times \mathit{\Gamma}_{\text{ac}}^\beta, \;\; Z_1, \dots, Z_p \preceq 0 \\ \text{subject to} \quad & Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta) = \sum_{k=1}^{p} E_{\mathcal{C}_k}^\top Z_k E_{\mathcal{C}_k}. \end{aligned} \tag{16}$$

Specifically, Lemma 2 allows us to substitute the single semidefinite constraint in (15) with multiple smaller ones plus an equality constraint. Moreover, the necessary and sufficient conditions of Lemma 2 means that these two formulations are in fact equivalent: a solution exists for (15) if and only if a solution also exists for (16).

We refer to (16) as Chordal-DeepSDP, and show that Chordal-DeepSDP provides the same guarantees as in [9, Theorem 2].

Theorem 3. *Suppose that the SDP in* (16) *is feasible, then*

$$\begin{bmatrix} x_1 \\ f(x_1) \\ 1 \end{bmatrix}^\top S \begin{bmatrix} x_1 \\ f(x_1) \\ 1 \end{bmatrix} \leq 0 \;\; \textit{if}\; x_1 \in X.$$

Proof. Suppose that $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta) \preceq 0$ for some nonnegative assignment of its parameters, and consider a trajectory $\mathbf{z} \in \mathbb{R}^{1+\sum_k n_K}$ where $x_1 \in X$. By negative semidefiniteness and expanding (12), we see that

$$\begin{aligned} &\mathbf{z}^\top Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta)\mathbf{z} \qquad (17) \\ &= \begin{bmatrix} x_1 \\ 1 \end{bmatrix}^\top P(\gamma_{\text{in}}) \begin{bmatrix} x_1 \\ 1 \end{bmatrix} + \begin{bmatrix} x_1 \\ x_K \\ 1 \end{bmatrix}^\top \widehat{S} \begin{bmatrix} x_1 \\ x_K \\ 1 \end{bmatrix} + \sum_{k=1}^{K-\beta} \begin{bmatrix} x_k \\ \vdots \\ x_{k+\beta} \\ 1 \end{bmatrix}^\top X_{k,\beta}(\gamma_k) \begin{bmatrix} x_k \\ \vdots \\ x_{k+\beta} \\ 1 \end{bmatrix} \leq 0. \end{aligned}$$

The goal is to show that the scalar induced by the P and $X_{1,\beta}, \ldots, X_{K-\beta,\beta}$ terms are all nonnegative, so therefore the S term must be nonpositive. First for the P term, since $x_1 \in X$ we thus have

$$\begin{bmatrix} x_1 \\ 1 \end{bmatrix}^\top P(\gamma_{\text{in}}) \begin{bmatrix} x_1 \\ 1 \end{bmatrix} \geq 0.$$

Next for the $X_{k,\beta}$ terms, note that $\mathbf{z}$ consists of states of the neural network and that each $Q_{k,\beta}$ as defined in (11) is a valid abstraction of the activation function. Then by the abstracted behavior per (2), the $X_{k,\beta}$ terms will satisfy

$$\begin{bmatrix} x_k \\ \vdots \\ x_{k+\beta} \\ 1 \end{bmatrix}^\top X_{k,\beta}(\gamma_k) \begin{bmatrix} x_k \\ \vdots \\ x_{k+\beta} \\ 1 \end{bmatrix} \geq 0 \;\text{ for all } k = 1, \ldots, p+1.$$

Finally, because the P term and all the $X_{k,\beta}$ terms yield nonnegative scalars yet the summation (17) is nonpositive, it must be the case that

$$\begin{bmatrix} x_1 \\ x_K \\ 1 \end{bmatrix}^\top \widehat{S} \begin{bmatrix} x_1 \\ x_K \\ 1 \end{bmatrix} \leq 0.$$

□

Remark 3. Chordal-DeepSDP resembles DeepSDP in the degenerate case of $p = 0$, i.e. the only summation term in (12) is $X_{1,K-1}$. When $\beta = 1$ ($p = K - 2$), Chordal-DeepSDP resembles the formulation in [25], but with a more expressive S that can constrain both x_1 and $f(x_1)$, rather than only $f(x_1)$.

4 Numerical Experiments

To evaluate the effectiveness of Chordal-DeepSDP we investigate the following.

(RQ1) *How does performance scale with the number of layers?*
(RQ2) *How does the accuracy trade-of with the sparsity parameter β?*
(RQ3) *What is the effect of interval propagation on our technique?*

We are interested in studying the effect of interval propagation because DeepSDP uses these bounds to derive more accurate QCs [9, Section 3.C], and we wish to investigate whether similar benefits carry over to Chordal-DeepSDP.

For solving SDPs, we used MOSEK 9.2 [1], and all instances were run on a quad-core personal computer with 30GB of RAM. We used the ReLU activation for all experiments and performed interval bound propagation as a precomputation step. All code is available open source online at `github.com/user/proj` [2].

Experimental Setup

For evaluation we generated a batch of feedforward networks with ReLU activation. Each network $N_{n,K}$ has intermediate dimension $n \in \{5, 10, 15\}$, depth $K \in \{5, 10, \dots, 45, 50\}$, and input-output dimensions $n_1 = m = 2$. The weights and biases are randomly scaled with respect to a Gaussian distribution $\mathcal{N}(0, \frac{4}{n+K})$, which we found to generate reasonably-sized output ranges when inputs were sampled from $X = \{x : \|x^\star - x\|_\infty \leq \varepsilon\}$, where $x^\star = 1_{n_1} \in \mathbb{R}^{n_1}$ is the vector of ones and $\varepsilon = 10^{-2}$; we assume this X to be the input set for all instances unless otherwise specified. Also, we say Chordal-β to mean an instance of Chordal-DeepSDP with sparsity β, i.e., Chordal-2 means $\beta = 2$.

4.1 RQ1

We first investigated how fast each method could verify safety. To test quadratic constraints over the input-output pair, we used the following S matrix

$$S = \begin{bmatrix} -\kappa I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \kappa = \sqrt{K \cdot n},$$

which constraints that the L_2 gain $\|f(x)\|^2 \leq \kappa \|x\|^2$ holds for all inputs $x \in X$. We hypothesize that for deeper networks, a smaller sparsity parameter β should

[2] Will be made available after de-anonymization

scale better, as the dimension of the semidefinite program for each clique scales quadratically with respect to β. However, we should expect that small values of β may fail to verify otherwise safe instance due to its sparsity.

To test our hypothesis we consider three methods: Chordal-1, Chordal-2, and DeepSDP. We ran each method on our batch of networks with a timeout of 180 seconds and report the results in Figure 4. A checkmark (✓) means that the instance was successfully verified, a question mark (?) means that the solver failed to find a solution (either by infeasibility or by insufficient progress), whereas a blank means a timeout.

	Chordal-1			**Chordal-2**			**DeepSDP**		
K	$n = 5$	$n = 10$	$n = 15$	$n = 5$	$n = 10$	$n = 15$	$n = 5$	$n = 10$	$n = 15$
5	0.1 ✓	2.3 ?	1.8 ✓	0.1 ✓	5.7 ?	4.1 ✓	0.1 ✓	1.3 ?	1.6 ✓
10	0.2 ✓	1.8 ✓	8.2 ✓	0.5 ✓	5.2 ✓	22.8 ✓	0.5 ✓	10.2 ✓	91.2 ✓
15	0.4 ✓	17.8 ?	48.8 ?	0.9 ✓	10.9 ✓	160.4 ?	2.8 ✓	100.1 ✓	
20	0.5 ✓	25.2 ?	64.6 ?	1.4 ✓	88.1 ?		11.9 ✓		
25	0.7 ✓	5.7 ✓	120.6 ?	1.5 ✓	18.8 ✓		41.5 ✓		
30	0.9 ✓	7.5 ✓	158.8 ?	1.6 ✓	20.6 ✓		109.5 ✓		
35	0.9 ✓	9.8 ✓		2.0 ✓	32.6 ✓				
40	1.3 ✓	13.4 ✓		2.4 ✓	30.9 ✓				
45	1.4 ✓	15.1 ✓		2.6 ✓	37.8 ✓				
50	1.6 ✓	14.5 ✓		3.6 ✓	42.8 ✓				

Fig. 4. Safety verification for instances of $N_{n,K}$ with time in seconds. A checkmark (✓) means successful verification; a question mark (?) means failed to find a solution; a blank space means timeout (180 seconds).

These experiments show that, in nearly all cases, exploiting chordal sparsity allows us to outperform DeepSDP by a notable margin. Comparing Chordal-1 and Chordal-2, we see that Chordal-1 is able to verify faster — provided it is able to find a solution. Indeed, there are cases such as $N_{10,15}$ on which Chordal-1 failed but Chordal-2 succeeded. In addition, we observe that if Chordal-1 is unable to verify an instance, the solver will accordingly stop earlier than for Chordal-2; this is seen for many networks of width $n = 15$. Furthermore, it appears that Chordal-1 and Chordal-2 both prefer deep and thin networks, whereas DeepSDP struggles with depth. Lastly, it is interesting to note that the experiments run in [9] show that DeepSDP performs well on wide ($n = 100$) but shallow ($K \leq 5$) networks, with verification times under 180 seconds.

4.2 (RQ2) Accuracy-Sparsity Trade-off

We next investigated the effect of sparsity on solution accuracy. Because a sparser parametrization would induce more zero entries in $Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^{\beta})$, it should become harder for the resulting SDP to enforce cross-layer constraints. Consequently, one should expect a sparser parametrization to yield a lower accuracy. To test our hypothesis, we solve for bounding hyperplanes by considering S matrices of form

$$S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & c_i \\ 0 & c_i^\top & -2d_i \end{bmatrix}$$

where $c_1, c_2, \ldots \in \mathbb{R}^2$ are given normal vectors and $d_i \in \mathbb{R}$ are problem variables. We then let Z and Z_β include this newly parametrized S, and solve the following problems for each $d_1, d_2, \ldots$ in order to compute bounding polytopes for DeepSDP and Chordal-DeepSDP, respectively,

$$\begin{aligned} &\underset{\gamma_{\text{in}}, \gamma_{\text{ac}}, d_i}{\text{minimize}} \;\; d_i, \;\; \text{subject to} \;\; Z(\gamma_{\text{in}}, \gamma_{\text{ac}}, d_i) \preceq 0 \\ &\underset{\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta, d_i}{\text{minimize}} \;\; d_i, \;\; \text{subject to} \;\; Z_\beta(\gamma_{\text{in}}, \gamma_{\text{ac}}^\beta, d_i) \preceq 0, \end{aligned} \tag{18}$$

This is illustrated in Fig. 5, where we compare the bounding polytopes generated by Chordal-1, Chordal-2, and DeepSDP. In addition to the three bounding polytopes, we also sample a batch of random trajectories from X — note that the chosen marker size for the sampled output points may sometimes give the false impression that they intersects or lies outside of the bounding polytope.

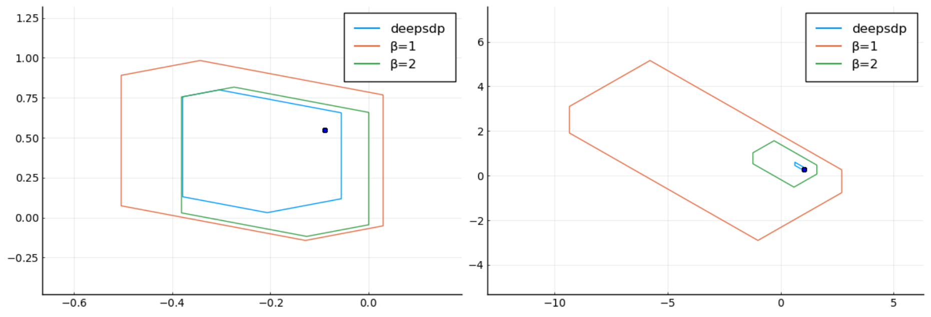

Fig. 5. Bounding polytopes for $N_{10,10}$ and $N_{10,15}$, respectively, as computed by Chordal-1, Chordal-2, and DeepSDP. One hyperplane is derived per problem instance.

We observe that the sparsity parameter does have an impact on the solution accuracy. Furthermore, tuning the sparsity from $\beta = 1$ to $\beta = 2$ already gives an increase in the solution accuracy. This suggests that one may achieve decently accurate solutions while retaining a relatively sparse construction.

4.3 (RQ3) Effect of Interval Propagation

Our experience with DeepSDP suggests that the quality of interval propagation has profound impact on the solution accuracy. With more accurate bounds, one can derive tighter QCs [9, Section 3.C]. We hypothesize that this effect is also present in Chordal-DeepSDP, in that better interval propagation leads to better solution accuracy.

The goal of this experiment is to compare the solution accuracy of Chordal-DeepSDP using our current interval propagation method in contrast to an idealized case. Our current implementation simply propagates a pair of maximum and minimum vectors through each layer, and appropriately update each based on worst-case analysis. This method is simple to implement, but quickly becomes conservative as the number of layers grows. We compare this to the hypothetically perfect interval propagation scheme, which we (unsoundly) approximate by sampling a large number of trajectories and find the extreme values at each layer. The results of our experiment are shown in Fig. 6, which uses the same networks from earlier in Fig. 5.

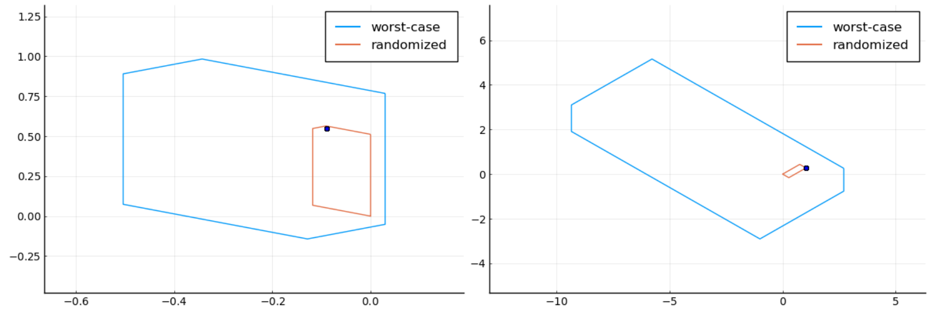

Fig. 6. Bounding boxes for $N_{10,10}$ and $N_{10,15}$ computed by Chordal-1, using worst-case (blue) and sampling-based (red) interval propagation.

This shows that an accurate interval propagation scheme has a significant impact on solution accuracy of Chordal-1, the sparsest formulation. This strongly motivates combining SDP-based techniques with methods that specialize in interval propagation such as [37, 38].

4.4 Discussion

Our experiments show that Chordal-SDP is able to perform safety verification faster than DeepSDP, in particular scaling well to deeper networks. In addition, we observe the predicted sparsity-accuracy trade-off in Fig. 5 — which is also manifested as a sparsity-vs-performance trade-off in Fig. 4. Notably, even values like $\beta = 2$ already give far more accurate solutions than $\beta = 1$. However, our chordal decomposition scheme primarily benefits deep networks, as we do not break down the semidefinite program with respect to network width.

Exploiting chordal sparsity is one way for SDP-based methods of neural network verification to scale, and indeed there are several ideas that we would like to consider in future works. For instance, DeepSDP uses $O(n^2)$ problem variables, where n is the dimension of $Z(\gamma_{\text{in}}, \gamma_{\text{ac}})$; however not all problem variables need to be used — which is an idea used in the related framework LipSDP [10]. By using less variables, one could allow the solver to converge faster. In addition, one could attempt to deploy Chordal-DeepSDP on specialized sparse SDP solvers such as CDCS [41]. The challenge then is to convert Chordal-DeepSDP into the solver input format. Furthermore, we also look to implement large-scale optimization algorithms such as ADMM [3, 39] while taking advantage of our problem structure. Appropriately setting up the ADMM algorithm would additionally allow us to run safety verification in a distributed setting.

5 Conclusions

We present Chordal-DeepSDP, a SDP-based method for the safety verification of feedforward neural networks. Using the concept of chordal sparsity, Chordal-DeepSDP presents a scalable chordal decomposition of DeepSDP. We show how to parametrically tune the sprasity of Chordal-SDP, allowing us to operate between the extremes of efficiency and accuracy. We provide numerical evaluations showcasing this trade-off in efficiency and accuracy and illustrate the computational advantages of Chordal-DeepSDP over a vanilla DeepSDP implementation.

We are aware of other tools and methods to which a comparison is desirable, but this is ambitious since SDP-based methods for neural network verification are not yet mature. The primary focus of this work is to shrink this gap by investigating how one might take advantage of specialized problem structures. Although the capabilities of Chordal-SDP are not yet competitive with state-of-the-art tooling, we nevertheless believe SDP-based methods to have potential for scalability and parallelizability due to their nature as convex problems.

References

1. Andersen, E.D., Andersen, K.D.: The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In: High performance optimization, pp. 197–232. Springer (2000)
2. Bak, S., Liu, C., Johnson, T.: The second international verification of neural networks competition (vnn-comp 2021): Summary and results. arXiv preprint arXiv:2109.00498 (2021)
3. Boyd, S., Parikh, N., Chu, E.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc (2011)
4. Chen, H., Liu, H.T.D., Jacobson, A., Levin, D.I.: Chordal decomposition for spectral coarsening. arXiv preprint arXiv:2009.02294 (2020)
5. Chen, S., Wong, E., Kolter, J.Z., Fazlyab, M.: Deepsplit: Scalable verification of deep neural networks via operator splitting. arXiv preprint arXiv:2106.09117 (2021)
6. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning. pp. 1–16. PMLR (2017)
7. Dvijotham, K., Stanforth, R., Gowal, S., Mann, T.A., Kohli, P.: A dual approach to scalable verification of deep networks. In: UAI. vol. 1, p. 3 (2018)
8. Everett, M.: Neural network verification in control. arXiv preprint arXiv:2110.01388 (2021)

9. Fazlyab, M., Morari, M., Pappas, G.J.: Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. IEEE Transactions on Automatic Control (2020)
10. Fazlyab, M., Robey, A., Hassani, H., Morari, M., Pappas, G.: Efficient and accurate estimation of lipschitz constants for deep neural networks. Advances in Neural Information Processing Systems **32**, 11427–11438 (2019)
11. Fischetti, M., Jo, J.: Deep neural networks and mixed integer linear optimization. Constraints **23**(3), 296–309 (2018)
12. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: Ai2: Safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 3–18. IEEE (2018)
13. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
14. Griewank, A., Toint, P.L.: On the existence of convex decompositions of partially separable functions. Mathematical Programming **28**(1), 25–49 (1984)
15. Ihlenfeld, L.P., Oliveira, G.H.: A faster passivity enforcement method via chordal sparsity. Electric Power Systems Research **204**, 107706 (2022)
16. Ivanov, R., Carpenter, T., Weimer, J., Alur, R., Pappas, G., Lee, I.: Verisig 2.0: Verification of neural network controllers using taylor model preconditioning. In: International Conference on Computer Aided Verification. pp. 249–262. Springer (2021)
17. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 169–178 (2019)
18. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient smt solver for verifying deep neural networks. In: International Conference on Computer Aided Verification. pp. 97–117. Springer (2017)
19. Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., et al.: The marabou framework for verification and analysis of deep neural networks. In: International Conference on Computer Aided Verification. pp. 443–452. Springer (2019)
20. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
21. Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., Kochenderfer, M.J.: Algorithms for verifying deep neural networks. arXiv preprint arXiv:1903.06758 (2019)
22. Lomuscio, A., Maganti, L.: An approach to reachability analysis for feed-forward relu neural networks. arXiv preprint arXiv:1706.07351 (2017)
23. Mason, R.P., Papachristodoulou, A.: Chordal sparsity, decomposing sdps and the lyapunov equation. In: 2014 American Control Conference. pp. 531–537. IEEE (2014)
24. Mirman, M., Gehr, T., Vechev, M.: Differentiable abstract interpretation for provably robust neural networks. In: International Conference on Machine Learning. pp. 3578–3586. PMLR (2018)
25. Newton, M., Papachristodoulou, A.: Exploiting sparsity for neural network verification. In: Learning for Dynamics and Control. pp. 715–727. PMLR (2021)
26. Newton, M., Papachristodoulou, A.: Neural network verification using polynomial optimisation. In: Proceedings of the IEEE Conference on Decision and Control (2021)

27. Raghunathan, A., Steinhardt, J., Liang, P.: Semidefinite relaxations for certifying robustness to adversarial examples. arXiv preprint arXiv:1811.01057 (2018)
28. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
29. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages **3**(POPL), 1–30 (2019)
30. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation **23**(5), 828–841 (2019)
31. Tjeng, V., Xiao, K., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. arXiv preprint arXiv:1711.07356 (2017)
32. Tran, H.D., Yang, X., Lopez, D.M., Musau, P., Nguyen, L.V., Xiang, W., Bak, S., Johnson, T.T.: Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In: International Conference on Computer Aided Verification. pp. 3–17. Springer (2020)
33. Vandenberghe, L., Andersen, M.S.: Chordal graphs and semidefinite optimization. Foundations and Trends in Optimization **1**(4), 241–433 (2015)
34. Vincent, J.A., Schwager, M.: Reachable polyhedral marching (rpm): A safety verification algorithm for robotic systems with deep neural network components. arXiv preprint arXiv:2011.11609 (2020)
35. Wong, E., Kolter, Z.: Provable defenses against adversarial examples via the convex outer adversarial polytope. In: International Conference on Machine Learning. pp. 5286–5295. PMLR (2018)
36. Xiang, W., Tran, H.D., Johnson, T.T.: Reachable set computation and safety verification for neural networks with relu activations. arXiv preprint arXiv:1712.08163 (2017)
37. Xiang, W., Tran, H.D., Johnson, T.T.: Output reachable set estimation and verification for multilayer neural networks. IEEE transactions on neural networks and learning systems **29**(11), 5777–5783 (2018)
38. Zhang, H., Weng, T.W., Chen, P.Y., Hsieh, C.J., Daniel, L.: Efficient neural network robustness certification with general activation functions. arXiv preprint arXiv:1811.00866 (2018)
39. Zheng, Y.: Chordal sparsity in control and optimization of large-scale systems. Ph.D. thesis, University of Oxford (2019)
40. Zheng, Y., Fantuzzi, G.: Sum-of-squares chordal decomposition of polynomial matrix inequalities. Mathematical Programming pp. 1–38 (2021)
41. Zheng, Y., Fantuzzi, G., Papachristodoulou, A., Goulart, P., Wynn, A.: Chordal decomposition in operator-splitting methods for sparse semidefinite programs
42. Zheng, Y., Fantuzzi, G., Papachristodoulou, A., Goulart, P., Wynn, A.: Fast admm for semidefinite programs with chordal sparsity. In: 2017 American Control Conference (ACC). pp. 3335–3340. IEEE (2017)