On the Expressivity of GNN for Solving Second Order Cone Programming

Ruizhe Li[1], **Enming Liang**[2,*], **Minghua Chen**[2,3,*]
[1] Department of Mathematics, Southern University of Science and Technology
[2]Department of Data Science, City University of Hong Kong
[3]School of Data Science, The Chinese University of Hong Kong, Shenzhen

Abstract

Graph Neural Networks (GNNs) have demonstrated both empirical efficiency and universal expressivity for solving constrained optimization problems such as linear and quadratic programming. However, extending this paradigm to more general convex problems with universality guarantees, particularly Second-Order Cone Programs (SOCPs), remains largely unexplored. We address this challenge by proposing a novel graph representation that captures the structure of conic constraints. We then establish a key universality theorem: *there exist GNNs that can provably approximate essential SOCP properties, such as instance feasibility and optimal solutions*. This work provides a rigorous foundation linking GNN expressive power to conic optimization structure, opening new avenues for scalable, data-driven SOCP solvers. Our approach extends to p-order cone programming for any $p \geq 1$ with universal expressivity preserved, requiring no GNN structural modifications. Numerical experiments on randomly generated SOCPs demonstrate the expressivity of the proposed GNN, which achieves better prediction accuracy with fewer parameters than fully connected neural networks.

1 Introduction:

Second Order Cone Programming (SOCP) represents a fundamental class of convex optimization problems with numerous real-world applications [1], including optimal power flow [2, 3], signal processing [4], and grasping force optimization [5]. However, traditional algorithms, such as interior point methods, face computational limitations in large-scale applications, particularly in real-time scenarios where rapid response is crucial.

Recent advances in machine learning have enabled solving optimization problems in real-time. Specifically, graph neural networks (GNNs) have been proven effective in learning input-to-solution mappings by leveraging problem structures. For instance, linear programs can be modeled as bipartite graphs with variable and constraint nodes [8], enabling efficient learning with a parameter sharing mechanism over GPUs. Beyond empirical success, theoretical foundations, including universal approximation capabilities, have been established for GNN applications in (mixed-integer) linear/quadratic programming [8, 9, 7] and convex quadratically constrained quadratic programming [7, 6].

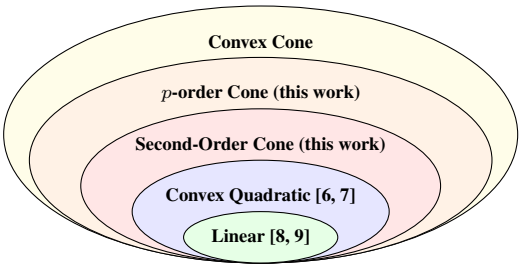


Figure 1: Hierarchy for convex constraints.

*Corresponding authors: Enming Liang (enming.cityu@gmail.com) and Minghua Chen (minghua@cuhk.edu.cn)

39th Conference on Neural Information Processing Systems (NeurIPS 2025). ScaleOPT Workshop.

Despite these advances, extending GNNs to more general convex programs like SOCP remains an open challenge. This paper proposes a novel GNN architecture with universal approximation capabilities for SOCPs, making the following contributions:

$\triangleright$ We propose a novel graph representation for SOCPs, which exploits linear relationships within the non-linear conic constraint and decomposes it into separated nodes for efficient graph representations.

$\triangleright$ Based on proposed graph representations, we design SOCP-GNNs, to predict the key properties of SOCPs, including instance feasibility and optimal solutions, with universal expressivity guarantees.

$\triangleright$ Our GNN design and universality guarantees can be extended directly to p-order conic programming for $p \geq 1$ (covering a class of polynomial programs) without GNN structural modifications.

$\triangleright$ Our experiments demonstrate that the expressivity of designed GNNs, which use fewer parameters to achieve better prediction accuracy compared to fully connected NNs.

To the best of our knowledge, this is the first GNN design for SOCP with universality guarantees, providing key insights for GNNs to solve general conic and polynomial programming.

2 Related Work

We review two primary paradigms for analyzing GNN expressivity for optimization problems: the **Weisfeiler-Lehman** (WL)-based and **Algorithm-Unrolling** (AU)-based frameworks.

The WL-based framework models optimization problems as graphs, where nodes represent variables and constraints, with edges modeling their interactions. It then links the GNN's expressive power with WL tests on graphs. Building on established foundations for (mixed-integer) linear programs (LP) [8, 9], researchers have extended this framework to more complex problems such as quadratic programs (QP) [7] and quadratically constrained QP (QCQP) [6]. A key challenge is representing non-linear constraints, as encoding complex interactions into nodes and edges is non-trivial. Recent work has addressed convex quadratic constraints through dynamic edge updates [7] or augmented quadratic variable nodes [6]. However, extending existing frameworks to represent general conic constraints like second-order cones remains an open question (see Appendix A.2 for details).

The AU-based paradigm maps iterative steps of specialized algorithms (e.g., primal-dual methods) onto GNN layers. By aligning GNN layers with known algorithms for specific problems such as LP [10, 11], QP [12, 13], and Max-CSP problems [14], universality and parameter complexity can be naturally established through existing algorithmic convergence properties. However, representing more complex algorithmic steps involving non-linear operations (e.g., factorization or projection) is non-trivial. Furthermore, the GNN's expressivity is inherently limited by the underlying capability of the algorithm itself (see Appendix A.3 for details).

In this work, we extend the WL-based framework to optimization problems with second-order cone constraints—a general class encompassing convex quadratic constraints with broad real-world applications. Our specialized GNN achieves both universal representation and computational efficiency, providing a foundational step toward applying GNNs to more general conic programming.

3 Problem and Open Issues

We consider a general second-order cone programming (SOCP) [15], defined as:

$$\begin{aligned} \text{minimize} \quad & e^\top x \\ \text{subject to} \quad & Fx \leq g, \quad l \leq x \leq r, \\ & \|A_i x + b_i\|_2 \leq c_i^T x + d_i, \quad i \in [m] \end{aligned} \tag{1}$$

where decision variables are $x \in \mathbb{R}^n$ and the problem parameters are $e \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{k_i \times n}$, $b_i \in \mathbb{R}^{k_i}$, $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $F \in \mathbb{R}^{b \times n}$, $g \in \mathbb{R}^b$, $l \in (\{-\infty\} \cup \mathbb{R})^n$, and $r \in (\{+\infty\} \cup \mathbb{R})^n$. For more details of SOCP-related concepts, please refer to Appendix B.1

Open issue: While GNNs have successfully modeled linear and convex quadratic relationships, handling SOC constraints remains challenging. Although SOC constraints can be reformulated as quadratic constraints, this transformation has critical limitations: the resulting quadratic coefficient

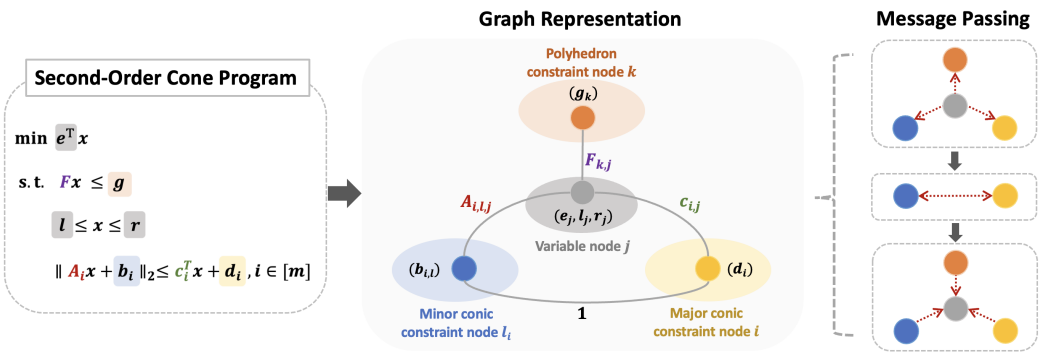


Figure 2: The graph representation of SOCPs and the message passing steps in GNN design. A specific SOCP instance and its corresponding SOCP-graph are included in Fig. 5, Appendix B.4.

matrices may be non-positive semidefinite, making existing GNN architectures for convex quadratic constraints theoretically inapplicable, and the transformation increases graph complexity while failing to exploit the inherent sparse structure of SOC constraints. Developing GNN models that efficiently process conic constraints with theoretical guarantees remains an open problem.

4 Methodology

We address this challenge of GNN for SOCP by introducing the following graph representations:

4.1 Graph Representation of SOCPs

As shown in Fig. 2, the graph representation of a SOCP consists of four types of nodes, to represent decision variables (V_1), polyhedron constraints (V_2), minor conic constraints (V_3), and major conic constraints (V_4):

- $V_1 := \{v_j\}_{j\in[n]}$ denotes decision variables, where each node v_j is associated with a feature tuple (e_j, l_j, r_j), representing the objective coefficient, variable lower, and upper bounds.
- $V_2 := \{s_k\}_{k\in[b]}$ denotes polyhedron constraints equipped with feature (g_k) for each node.
- $V_3 := \{o_{il}\}_{i\in[m]}^{l\in[k_i]}$ denotes the minor conic constraint, where each node o_{il} represents the i-th conic constraint's l-th component, with feature $(b_{i,l})$.
- $V_4 := \{q_i\}_{i\in[m]}$ denotes the i-th conic constraint with feature (d_i).

The SOCP graph includes four types of edges to model the interactions between the decision variables and different constraints:

- $e_{kj} \in V_1 \times V_2$ denote the edges between the variable node v_j and the polyhedron constraint node s_k, with weight F_{kj}.
- $e_{j,il} \in V_1 \times V_3$ denote the edge between variable node v_j and minor conic constraint node o_{il}, with weight $A_{i,lj}$.
- $e_{ij} \in V_1 \times V_4$ denote the edge between the variable node v_j and major conic constraint node q_i, with weight $c_{i,j}$.
- $e_{il,i} \in V_3 \times V_4$ denote the edge between node o_{il} and node q_i, with a constant weight 1.

Remark 1 (**Insights of Graph Design**). For linear objectives and polyhedral constraints, our structure builds upon previous works [8]. However, for nonlinear second-order cone constraints, we exploit the linear relationships within the conic constraint—specifically, between A_i and x, and between c_i and x. By representing the left-hand side and right-hand side as separate constraint nodes with linear interactions, and connecting them via an edge, we decompose the challenging nonlinear conic constraint into components amenable to efficient graph representations.

Remark 2 (**SOCP $\rightarrow$ QCQP**[2]). One may note that SOC constraints, $\|Ax + b\|_2 \le c^\top x + d$, can be transformed into quadratic constraints by squaring both sides, potentially enabling the application of

[2]Please refer to Appendix B.2 for detailed equivalent SOCP formulations.

previous work on quadratic constraints [6, 7]. However, this transformation introduces two significant challenges: (i) the resulting quadratic coefficient matrix $A^\top A - cc^\top$ may not be positive semidefinite, rendering previous work theoretically inapplicable for such a non-convex QC; and (ii) the quadratic coefficient matrix $A^\top A - cc^\top$ may be dense, losing the potential sparse/low-rank structure of A and c in the SOC constraint and making the graph representation and message passing inefficient.

Remark 3 (**Convex QCQP $\rightarrow$ SOCP**). Conversely, we may transform convex quadratic constraints of the form $x^\top Qx + c^\top x + d \leq 0$ into SOC constraints for more effective graph representation. For example, we can apply matrix decomposition $Q = LL^\top$ where $L \in \mathbb{R}^{n\times r}$, and reformulate the constraint as $\left\|[(1 + c^\top x + d)/2;\ L^T x]\right\|_2 \leq (1 - c^\top x - d)/2$. Such a transformation is particularly efficient for low-rank matrices Q where $r \ll n$, as it reduces the complexity of the graph representation for original convex quadratic constraints, from quadratic node [6] to minor conic constraint node via SOC graph representation. The convex quadratic objective in QCQP can also be converted to a linear objective by adding the epigraph constraint [15]. Thus, a convex QCQP with n variables and m quadratic constraints is equivalent to an SOCP with $n + 1$ variables and $m + 1$ conic constraints (potentially low-rank). We further provide a quantitative comparison in the next section (Table 3).

4.2 Message Passing in SOCP-GNNs

Given the established graph representation of SOCPs, we propose message-passing (MP)-GNNs, consisting of an embedding layer, T message-passing layers (each comprised of three sub-layers), and a readout layer, detailed as follows:

- **Embedding Layer**: For all nodes, the input features $h^{0,v}, h^{0,s}, h^{0,o}, h^{0,q}$ are initialized by embedding the node features into a hidden space $\mathbb{R}^{h_0}$, where h_0 is the space dimension. Specifically,

$$h^{0,v} \leftarrow \hat{g}_1^0(h^v), \forall v \in V_1, \quad h^{0,s} \leftarrow \hat{g}_2^0(h^s), \forall s \in V_2$$
$$h^{0,o} \leftarrow \hat{g}_3^0(h^o), \forall o \in V_3, \quad h^{0,q} \leftarrow \hat{g}_4^0(h^q), \forall q \in V_4$$

  where $\hat{g}_l^0$ are learnable embedding functions for $l = 1, 2, 3, 4$, and h^v, h^s, h^o, h^q denotes the node features for $v \in V_1, s \in V_2, o \in V_3, q \in V_4$, respectively.

- **Message-Passing Layer:** Each message-passing layer consists of three sub-layers for updating the features of nodes with learnable functions f_l^t, g_l^t. For notation simplicity, w_{ij} represents the weight of edge e_{ij} and $\tau(n) \in \{1, 2, 3, 4\}$ denotes the index of the node set for a node n.
  - **Updating for all constraint nodes ($V_1 \rightarrow V_2 + V_3 + V_4$)**: $\forall s \in V_2$ and $\forall n \in V_3 \cup V_4$, we update the embedding as:

$$h^{t+1,s} \leftarrow g_1^t\left(h^{t,s}, \sum_{v\in V_1} w_{v,s} f_1^t(h^{t,v})\right),\ \bar{h}^{t,n} \leftarrow g_{\tau(n)-1}^t\left(h^{t,n}, \sum_{v\in V_1} w_{v,n} f_{\tau(n)-1}^t(h^{t,v})\right)$$

  - **Updating between major and minor conic constraint nodes** ($V_3 \rightarrow V_4$ and $V_4 \rightarrow V_3$): $\forall q \in V_4$ and $\forall o \in V_3$, we update the embedding as:

$$h^{t+1,q} \leftarrow g_4^t\left(\bar{h}^{t,q}, \sum_{o\in V_3} w_{o,q} f_4^t(\bar{h}^{t,o})\right),\ h^{t+1,o} \leftarrow g_5^t\left(\bar{h}^{t,o}, \sum_{q\in V_4} w_{q,o} f_5^t(h^{t+1,q})\right)$$

  - **Updating for variable nodes ($V_2 + V_3 + V_4 \rightarrow V_1$)**: $\forall v \in V_1$, we update the embedding as:

$$h^{t+1,v} \leftarrow g_6^t\left(h^{t,v}, \sum_{s\in V_2} w_{s,v} f_6^t(h^{t+1,s}), \sum_{o\in V_3} w_{o,v} f_7^t(h^{t+1,o}), \sum_{q\in V_4} w_{q,v} f_8^t(h^{t+1,q})\right)$$

- **Readout layer**: The readout layer leverages a learnable function f_{out} to map the node embedding $h^{T,v}$ output by the T-th (i.e., last) message-passing layer for $v \in V_1 \cup V_2 \cup V_3 \cup V_4$, to a readout y in a desired output space $\mathbb{R}^a$, where a is the output dimension. For example:
  - Graph-level scalar output (e.g., predicting SOCP feasibility with $a = 1$):

$$y = f_{\text{out}}\left(I_1, I_2, I_3, I_4\right)$$

  - Node-level vector output (e.g., predicting SOCP optimal solutions with $a = n$)

$$y_i = f_{\text{out}}\left(h^{T,v_i}, I_1, I_2, I_3, I_4\right)$$

  where $I_1 = \sum_{v\in V_1} h^{T,v},\ I_2 = \sum_{s\in V_2} h^{T,s},\ I_3 = \sum_{o\in V_3} h^{T,o},\ I_4 = \sum_{q\in V_4} h^{T,q}$.

As mentioned in Remarks 2 and 3, our SOCP-GNN also efficiently handles convex QCQPs by reformulating them into SOCP. Based on the GNN architecture described above, we analyze both the node and message passing complexity compared to previous works on convex QCQP [6, 7]. Our SOCP-GNN achieves the same order of node and message passing complexity as state-of-the-art GNNs designed specifically for QCQP. Moreover, SOCP-GNN becomes more efficient when the quadratic matrices exhibit low-rank structures.

	Number of Nodes	**1-Layer MP Complexity**
[6]	$\mathcal{O}(n^2+m)$	$\mathcal{O}(n^3+mn^2)$
[7]	$\mathcal{O}(mn)$	$\mathcal{O}(mn^2)$
Ours	$\mathcal{O}(n+\sum_{i=0}^{m} r_i)$	$\mathcal{O}(n\cdot\sum_{i=0}^{m} r_i)$

Figure 3: Complexity comparison of GNNs for convex QCQP with n variables, m quadratic constraints, and quadratic coefficient matrix of ranks $r_i \leq n$, $i = 0, \ldots, m$, where $i = 0$ indicate the quadratic matrix from objective.

Therefore, *SOCP-GNN not only extends theoretical applicability to the broader class of SOCP beyond convex QCQPs, but also maintains competitive computational complexity when restricted to the convex QCQP subclass*. See detailed discussion in Appendix B.5.

5 Universality of GNN for SOCPs

With the established graph representation and corresponding GNN, we formally prove the universality of the GNN for predicting key properties of SOCPs, like the instance feasibility and optimal solutions.

5.1 Formal Definitions

Definition 5.1 (Spaces of SOCP-Graphs)**.** Let $\mathcal{G}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}$ denote the set of graph representations for all SOCPs with n variables, m conic constraints with dimension $k_1, ..., k_m$, and b polyhedron constraints.

Definition 5.2 (Spaces of SOCP-GNNs)**.** We define $\mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^a)$ as the set of message passing GNNs proposed in Sec. 4.2 that map the input graph in $\mathcal{G}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}$ to a target output in $\mathbb{R}^a$. Each GNN is parameterized by continuous embedding functions $g_{l_1}^0, l_1 \in [4]$, continuous hidden functions in the message passing layers $g_{l_2}^t, l_2 \in [6]$ and $f_{l_3}^t, l_3 \in [8]$, and the continuous readout function f_{out}.

Definition 5.3 (Target mappings[3])**.** Let $\mathcal{G}_{\text{SOCP}}$ be a graph representation of a SOCP problem. We define the following target mappings.

- **Feasibility mapping:** $\Phi_{\text{feas}}(\mathcal{G}_{\text{SOCP}}) = 1$ if the SOCP is feasible and $\Phi_{\text{feas}}(\mathcal{G}_{\text{SOCP}}) = 0$ otherwise.
- **Optimal solution mapping:** $\Phi_{\text{sol}}(\mathcal{G}_{\text{SOCP}}) = x^*$, where x^* is the optimal solution of the SOCP[4].

5.2 Universal Approximation of SOCP-GNNs

Here, we provide the main theoretical results to validate the SOCP-GNN's universal expressivity for SOCP, i.e., *there always exists an SOCP-GNN that can universally approximate target mappings in Def. 5.3 within given error tolerance*:

Theorem 1. *For any Borel regular probability measure $\mathbb{P}$ on the space of SOCPs $\mathcal{G}_{SOCP}^{n,m,,k_1,\ldots,k_m,b}$ and any $\delta, \epsilon > 0$, there exists $F \in \mathcal{F}_{SOCP}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^a)$ such that for any target mapping $\Phi : \mathcal{G}_{SOCP}^{n,m,,k_1,\ldots,k_m,b} \to \mathbb{R}^a$ defined in Def. 5.3, we have*

$$\mathbb{P}\{||F(\mathcal{G}_{SOCP}) - \Phi(\mathcal{G}_{SOCP})|| > \delta\} < \epsilon. \tag{2}$$

The detailed proof is provided in Appendix C. This Theorem formally establishes the universal expressivity of the proposed SOCP-GNNs. The proof structure follows established foundations for LP in [8]. To deal with the non-linear conic constraints, we leverage the *equivariance*, *convexity*, and *separability*[5] of the ℓ_2 norm, and then establish the expressive power of SOCP-GNNs. We further extend the universal expressivity of the proposed GNN to p-order cone programming in Appendix C.6, since the core Lemmas in our proof are also satisfied for the ℓ_p norm.

[3]For more target mappings, please refer to Def. B.1. Theorem 1 also holds for these target mappings.

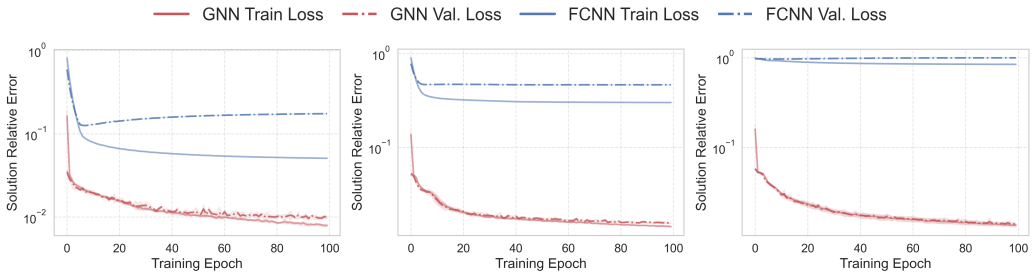
[4]Since an SOCP may admit multiple optimal solutions, we choose the one with minimum l_2 norm [8].

[5]Please refer to those definitions in Definition C.2 and C.3.

6 Numerical Experiments

In this section, we generate random SOCP instances to demonstrate the effectiveness of SOCP-GNN. For dataset generation, we randomly sample coefficient matrices and constraint parameters following the CVXPY example code structure. We denote an SOCP instance by a tuple (n, b, m), where n represents the number of decision variables, b denotes the number of polyhedral constraints, and m indicates the number of second-order cone constraints. The total input parameters for an SOCP (n, b, m) are of dimension $\mathcal{O}(n \cdot (b + m))$. Each instance is solved in CVXPY to obtain ground truth solutions, forming our training dataset. We collected `5,000` samples for three problem scales, respectively. We then train SOCP-GNN using standard supervised learning procedures for optimal solution predictions. We also test the feasibility classification in Appendix D.

Although SOCP instances can be reformulated as QCQPs, existing GNN-based approaches for QCQP [6, 7] do not provide publicly available implementations. Therefore, we employ a fully-connected neural network (FCNN) as our primary baseline for comparison, where the FCNN receives the same problem parameters as input in vectorized form. This comparison allows us to isolate the benefits of the graph structure and message-passing mechanisms inherent in SOCP-GNN against a standard FCNN baseline. Besides the network structure, other training settings, such as learning rate and batch size, remain the same for both networks. All experiments are executed on an NVIDIA H200 GPU.



(a) Small SOCP: `(50,10,10)` (b) Medium SOCP `(100,50,50)` (c) Large SOCP `(500,100,100)`

Figure 4: Comparison between the proposed GNN and FCNN for predicting optimal solutions of SOCPs. The GNN uses approximately `0.6M` parameters across all three problem scales, while the FCNN uses approximately `0.6M`, `5.9M`, and `58M` parameters for the respective scales.

As shown in Fig. 6, we compare the solution relative error [6] of our SOCP-GNN against the FCNN baseline across three different problem scales: Small (a), Medium (b), and Large (c), over 100 training epochs. Our SOCP-GNN demonstrates superior performance across all problem scales, consistently achieving substantially lower error on both training and validation sets. In particular, for the large-scale SOCP with input dimension `452,400`, our GNN achieves better prediction accuracy while using significantly fewer parameters, only `0.6M` parameters compared to `58M` for the FCNN baseline, representing approximately $100\times$ reduction in model complexity. This demonstrates SOCP-GNN's superior parameter efficiency and validates its ability to effectively learn the target mappings in SOCPs by leveraging the sparse graph structure that naturally characterizes these problems.

7 Conclusions, Limitations, and Future Works

This paper introduces a novel graph representation for SOCP, a fundamental class of convex programs, covering previous LP, QP, and convex QCQP. We then design a GNN for predicting key properties of SOCP, like feasibility and optimal solutions, proving its universal expressivity guarantees. Our designs and guarantees directly extend to p-order cone programming, broadening the scope of GNNs in conic and polynomial optimization. These results are validated by numerical experiments.

Beyond the expressivity guarantees, several limitations are worth discussing and motivating future directions: (i) the sample and parameter complexity for GNNs to solve optimization problems, which is also a challenge shared by prior WL-test-based frameworks [8, 9, 7, 6]; (ii) extending this GNN paradigm to more general convex problems, such as Semidefinite Programs (SDPs) with matrix variables, is a significant future step. (iii) unifying existing WL-based and AU-based frameworks for general GNN analysis will also be explored.

[6] Following [7], the solution relative error between prediction $\hat{x}$ and ground truth x^* is as $\frac{\|\hat{x}-x^*\|_2^2}{\max(1,\|x^*\|_2^2)}$

Acknowledments

This work is supported in part by a General Research Fund from Research Grants Council, Hong Kong (Project No. 11214825), a Collaborative Research Fund from Research Grants Council, Hong Kong (Project No. C1049-24G), an InnoHK initiative, The Government of the HKSAR, Laboratory for AI-Powered Financial Technologies, a Shenzhen-Hong Kong-Macau Science & Technology Project (Category C, Project No. SGDX20220530111203026), and a Start-up Research Grant from The Chinese University of Hong Kong, Shenzhen (Project No. UDF01004086). The authors would also like to thank the anonymous reviewers for their helpful comments.

References

[1] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear algebra and its applications*, 284(1-3):193–228, 1998.

[2] Md Mahmud-Ul-Tarik Chowdhury, Sukumar Kamalasadan, and Sumit Paudyal. A second-order cone programming (socp) based optimal power flow (opf) model with cyclic constraints for power transmission systems. *IEEE Transactions on Power Systems*, 39(1):1032–1043, 2023.

[3] Burak Kocuk, Santanu S Dey, and X Andy Sun. Strong socp relaxations for the optimal power flow problem. *Operations Research*, 64(6):1177–1196, 2016.

[4] Qingjiang Shi, Weiqiang Xu, Tsung-Hui Chang, Yongchao Wang, and Enbin Song. Joint beamforming and power splitting for miso interference channel with swipt: An socp relaxation and decentralized algorithm. *IEEE Transactions on Signal Processing*, 62(23):6194–6208, 2014.

[5] Amin Fakhari, Aditya Patankar, Jiayin Xie, and Nilanjan Chakraborty. Computing a task-dependent grasp metric using second-order cone programs. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4009–4016. IEEE, 2021.

[6] Chenyang Wu, Qian Chen, Akang Wang, Tian Ding, Ruoyu Sun, Wenguo Yang, and Qingjiang Shi. On representing convex quadratically constrained quadratic programs via graph neural networks. *arXiv preprint arXiv:2411.13805*, 2024.

[7] Ziang Chen, Xiaohan Chen, Jialin Liu, Xinshang Wang, and Wotao Yin. Expressive power of graph neural networks for (mixed-integer) quadratic programs. *arXiv preprint arXiv:2406.05938*, 2024.

[8] Ziang Chen, Jialin Liu, Xinshang Wang, Jianfeng Lu, and Wotao Yin. On representing linear programs by graph neural networks. *arXiv preprint arXiv:2209.12288*, 2022.

[9] Ziang Chen, Jialin Liu, Xinshang Wang, Jianfeng Lu, and Wotao Yin. On representing mixed-integer linear programs by graph neural networks, 2023.

[10] Chendi Qian, Didier Chételat, and Christopher Morris. Exploring the power of graph neural networks in solving linear optimization problems. In *International conference on artificial intelligence and statistics*, pages 1432–1440. PMLR, 2024.

[11] Bingheng Li, Linxin Yang, Yupeng Chen, Senmiao Wang, Qian Chen, Haitao Mao, Yao Ma, Akang Wang, Tian Ding, Jiliang Tang, et al. Pdhg-unrolled learning-to-optimize method for large-scale linear programming. *arXiv preprint arXiv:2406.01908*, 2024.

[12] Chendi Qian and Christopher Morris. Towards graph neural networks for provably solving convex optimization problems. *arXiv preprint arXiv:2502.02446*, 2025.

[13] Linxin Yang, Bingheng Li, Tian Ding, Jianghua Wu, Akang Wang, Yuyi Wang, Jiliang Tang, Ruoyu Sun, and Xiaodong Luo. An efficient unsupervised framework for convex quadratic programs via deep unrolling, 2024.

[14] Morris Yau, Nikolaos Karalias, Eric Lu, Jessica Xu, and Stefanie Jegelka. Are graph neural networks optimal approximation algorithms? *Advances in Neural Information Processing Systems*, 37:73124–73181, 2025.

[15] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical programming*, 95(1):3–51, 2003.

[16] Xiang Pan, Tianyu Zhao, Minghua Chen, and Shengyu Zhang. Deepopf: A deep neural network approach for security-constrained dc optimal power flow. *IEEE Transactions on Power Systems*, 36(3):1725–1735, 2020.

[17] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

[18] Ziang Chen, Jialin Liu, Xiaohan Chen, Xinshang Wang, and Wotao Yin. Rethinking the capacity of graph neural networks for branching strategy. *arXiv preprint arXiv:2402.07099*, 2024.

[19] Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. How much data is sufficient to learn high-performing algorithms? generalization guarantees for data-driven algorithm design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 919–932, 2021.

[20] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch: Generalization guarantees and limits of data-independent discretization. *Journal of the ACM*, 71(2):1–73, 2024.

[21] Qian Chen, Tianjian Zhang, Linxin Yang, Qingyu Han, Akang Wang, Ruoyu Sun, Xiaodong Luo, and Tsung-Hui Chang. Symilo: A symmetry-aware learning framework for integer linear optimization. *Advances in Neural Information Processing Systems*, 37:24411–24434, 2024.

[22] Ian Horrocks Matthew Morris, Bernardo Cuenca Grau. Orbit-equivariant graph neural networks. In *2024 The International Conference on Learning Representations(ICLR)*, pages 7056–7062. ICLR, 2024.

[23] Qian Chen, Lei Li, Qian Li, Jianghua Wu, Akang Wang, Ruoyu Sun, Xiaodong Luo, Tsung-Hui Chang, and Qingjiang Shi. When gnns meet symmetry in ilps: an orbit-based feature augmentation approach. *arXiv preprint arXiv:2501.14211*, 2025.

[24] Chendi Qian and Christopher Morris. Principled data augmentation for learning to solve quadratic programming problems. *arXiv preprint arXiv:2506.01728*, 2025.

[25] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks, 2019.

[26] Qian Li, Tian Ding, Linxin Yang, Minghui Ouyang, Qingjiang Shi, and Ruoyu Sun. On the power of small-size graph neural networks for linear programming. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[27] Qian Li, Minghui Ouyang, Tian Ding, Yuyi Wang, Qingjiang Shi, and Ruoyu Sun. Towards explaining the power of constant-depth graph neural networks for linear programming. In *The Thirteenth International Conference on Learning Representations*, 2025.

[28] Yu He and Ellen Vitercik. Primal-dual neural algorithmic reasoning, 2025.

[29] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Small relu networks are powerful memorizers: a tight analysis of memorization capacity. *Advances in neural information processing systems*, 32, 2019.

[30] Waiss Azizian and Marc Lelarge. Expressive power of invariant and equivariant graph neural networks. *arXiv preprint arXiv:2006.15646*, 2020.

Contents

A Discussions on Related Works

A.1 GNN for Constrained Optimization

In response to the growing demand for solving large-scale optimization problems in real-time, "learning to optimize" paradigms have emerged across multiple domains [16, 17]. Among various neural approaches, Graph Neural Networks (GNNs) have demonstrated particular effectiveness for optimization problems with inherent graph structures, leveraging the natural correspondence between problem formulations and graph representations [8, 6, 9].

To understand the fundamental capabilities of GNNs in optimization contexts, research has examined multiple theoretical and practical aspects, including expressivity [8, 18], generalization properties [19, 20], and symmetry preservation [21, 22, 23, 24]. The analysis of GNN expressive power in optimization is primarily guided by two complementary theoretical paradigms: the **Weisfeiler-Lehman (WL)-test-based framework**, which characterizes what optimization properties GNNs can theoretically distinguish, and the **Algorithm-Unrolling (AU)-based framework**, which establishes connections between classical optimization algorithms and GNN architectures through direct algorithmic simulation.

A.2 WL-based Frameworks

This section reviews optimization problems where GNNs have been proven to achieve universal approximation capabilities through theoretical frameworks based on Weisfeiler-Leman (WL) tests.

Linear Programming (LP) [8]: It establishes a foundational theoretical framework for analyzing GNN expressivity in solving LPs through WL-tests. Building upon the bipartite graph representation introduced by [25], they demonstrate a formal connection between GNN expressivity and WL-tests on graph structures. Their key theoretical contribution proves that GNNs achieve universality over the parameter space of LPs. Specifically, they show the existence of message-passing GNNs capable

of reliably approximating fundamental LP properties, including feasibility, optimal objective value, and optimal solutions.

Mixed-Integer Linear Programming (MILP) [9]: The extension to MILP presents significant theoretical challenges not encountered in the continuous LP setting. The fundamental limitation arises from the discrete nature of integer variables, where GNN expressivity remains constrained by the discriminative power of WL-tests. A critical issue emerges: two MILP instances that are indistinguishable under WL-tests may exhibit fundamentally different properties regarding feasibility and optimal solutions. To address these challenges, the authors identify a restricted class of MILPs satisfying the "unfoldable" property, for which universality guarantees can be established. Additionally, they demonstrate that augmenting the graph representation with random node features enables GNNs to achieve universality over the complete class of MILP problems, effectively circumventing the limitations imposed by deterministic WL-tests.

Linearly Constrained Quadratic Programming (LCQP) [7]: While modeling linear constraints through a bipartite graph is relatively straightforward, extending graph-based approaches to handle quadratic objective functions presents challenges. It addresses this by introducing self-connections within variable nodes to capture quadratic interactions in the objective function. Their framework extends a broader class of mixed-integer LCQP problems satisfying the MP-tractable property, establishing universality results for GNNs on specific computational tasks within this class.

The authors further extend their approach to convex quadratically constrained quadratic programming (QCQP) through dynamic edge update mechanisms, as detailed in their supplementary materials, demonstrating the framework's adaptability to more complex constraint structures.

Convex Quadratically Constrained Quadratic Programming (QCQP) [6]: It provides a comprehensive treatment of convex QCQPs, addressing the significant complexity introduced by multiple convex quadratic constraints. The key innovation lies in their sophisticated design of edge weights and specialized GNN architecture, which together ensure that the resulting message-passing framework achieves universality for the complete class of convex QCQP problems. This represents a significant advancement in handling optimization problems with complex constraint structures through GNNs.

A.3 AU-based Frameworks

Algorithm unrolling represents a fundamental approach in learning-based optimization, enhancing interpretability by directly simulating classical algorithmic procedures through neural network architectures. This section reviews successful applications of GNNs in unrolling established optimization algorithms.

Interior Point Method [10, 12]: The unrolling of Interior Point Methods (IPM) establishes a direct and interpretable correspondence between classical optimization algorithms and GNNs. Qian et al. [10] first provide theoretical foundations demonstrating that standard IPM iterations for LPs can be precisely simulated through sequences of GNN message-passing operations. This framework was extended to the broader class of LCQPs [12], maintaining the fundamental correspondence between algorithmic steps and neural computations.

Primal-Dual Hybrid Gradient [11, 13]: The unrolling of Primal-Dual Hybrid Gradient (PDHG) algorithms provides a scalable framework for accelerating first-order optimization methods through learning-based approaches. Li et al. [11] introduce PDHG-Net for large-scale LPs, demonstrating that optimal LP solutions can be approximated using polynomial-sized neural networks. This foundational work establishes both theoretical guarantees and practical scalability for the unrolled PDHG framework. The extension to QP represents another advancement [13], which introduces an innovative unsupervised training methodology that directly incorporates Karush-Kuhn-Tucker (KKT) optimality conditions into the loss function.

Specialized Algorithms for Structured Problems [26, 27, 14, 28]: For optimization problems with specialized structures, researchers have developed tailored algorithmic approaches that leverage problem-specific properties for effective GNN unrolling.

For covering and packing LPs, Li et al. [26] design variants of the Awerbuch-Khandekar algorithm, successfully unrolling these through careful exploitation of activation function properties. Specifically, they utilize ELU and sigmoid activation functions to simulate exponential operations and Heaviside

step functions, respectively, enabling reproduction of the classical algorithm's behavior within the GNN framework.

In the context of sparse binary LPs, Li et al. [27] propose a constant-round distributed algorithm that applies to almost all sparse binary LP instances. This algorithm naturally aligns with constant-depth, constant-width GNN architectures, providing theoretical justification for the empirical success of shallow networks in this domain.

Yau et al. [14] demonstrate that polynomial-sized GNNs can effectively learn powerful approximation algorithms for Maximum Constraint Satisfaction Problems (Max-CSP). Their approach leverages the equivalence between projected gradient descent on low-rank vector formulations of relaxed semidefinite programs and local message-passing operations inherent in GNN architectures.

Additionally, He et al. [28] align GNN architectures with primal-dual algorithmic reasoning for minimum hitting set problems, achieving empirical success in generalization across problem sizes and out-of-distribution scenarios.

B Preliminary and Basic Concepts

B.1 Basic concepts of SOCPs

For problem 1, we denote all the feasible solution by:

$$\mathcal{X}_{\text{feasible}} := \left\{ x \in \mathbb{R}^n \mid Fx \leq g;\ l \leq x \leq r;\ \|A_i x + b_i\|_2 \leq c_i^T x + d_i,\ \forall i \in [m] \right\}. \tag{3}$$

If $\mathcal{X}_{\text{feasible}}$ is not empty, problem 1 is said to be **feasible**; otherwise, it is said to be **infeasible**. A feasible SOCP is **bounded** if and only if the objective function is bounded from below in $\mathcal{X}_{\text{feasible}}$, i.e., $\exists a \in \mathbb{R}$ such that

$$e^T x \geq a, \forall x \in \mathcal{X}_{\text{feasible}}$$

Otherwise, the SOCP instance is **unbounded**.

For a feasible and bounded SOCP, its optimal value is defined as: $\inf \{e^T x, \forall x \in \mathcal{X}_{\text{feasible}}\}$. Moreover, x^* is said to be an **optimal solution** if it's feasible and

$$e^T x^* \leq e^T x, \forall x \in \mathcal{X}_{\text{feasible}}$$

Unlike convex QCQP, an SOCP instance may not admit an optimal solution even when it's feasible and bounded (corollary 4). Moreover, an SOCP instance can also have multiple solutions.

B.2 Equivalent Formulations of SOCP

Dimension Reduction of SOC Constraints: Consider a second-order cone (SOC) constraint of the form $\|Ax + b\|_2 \leq c^T x + d$, where $A \in \mathbb{R}^{k \times n}$ has rank $r \leq \min(k, n)$. Let the singular value decomposition of A be $A = U\Sigma V^T$, where $U \in \mathbb{R}^{k \times r}$ has orthonormal columns, $\Sigma \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, and $V \in \mathbb{R}^{n \times r}$ has orthonormal columns.

Since U has orthonormal columns, we have $U^T U = I_r$ and UU^T is the orthogonal projection onto the column space of A. We can decompose the vector b as

$$b = b_{\|} + b_{\perp}, \quad \text{where} \quad b_{\|} = UU^T b, \quad b_{\perp} = (I_k - UU^T)b \tag{4}$$

where $b_{\|}$ lies in the column space of A and $b_{\perp}$ is orthogonal to it.

Define $A' = \Sigma V^T \in \mathbb{R}^{r \times n}$ and $b' = U^T b_{\|} \in \mathbb{R}^r$. Then:

$$A = U\Sigma V^T = UA' \tag{5}$$

and

$$Ax + b = UA'x + UU^T b + (I_k - UU^T)b = U(A'x + U^T b_{\|}) + b_{\perp} \tag{6}$$

Since U has orthonormal columns and $b_{\perp}$ is orthogonal to the column space of U, we have:

$$\|Ax + b\|_2 = \|U(A'x + U^T b_{\|}) + b_{\perp}\|_2 = \left\| \begin{pmatrix} A'x + U^T b_{\|} \\ \|b_{\perp}\|_2 \end{pmatrix} \right\|_2 \tag{7}$$

This reformulation reduces the constraint to at most $r+1$ rows, which is beneficial when $k \gg r$.

Reformulation of SOCP to QCQP: A SOC constraint $\|Ax+b\|_2 \leq c^T x + d$ can be equivalently written as the quadratic constraint (may be non-convex) by squaring both sides as:

$$(Ax+b)^T(Ax+b) \leq (c^T x + d)^2$$
$$x^T A^T A x + 2b^T A x + \|b\|_2^2 \leq x^T c c^T x + 2dc^T x + d^2$$

provided that $c^T x + d \geq 0$. Rearranging terms yields:

$$x^T(A^T A - cc^T)x + 2(b^T A - dc^T)x + (\|b\|_2^2 - d^2) \leq 0 \tag{8}$$

This transformation is valid only when the right-hand side of the original SOC constraint is non-negative, which must be enforced as an additional linear constraint $c^T x + d \geq 0$.

Reformulation of Convex QCQP to SOCP: Conversely, we may transform convex quadratic constraints of the form $x^\top Q x + c^\top x + d \leq 0$ into SOC constraints. Since $Q \in \mathbb{S}_+^n$ is positive semidefinite, we can apply matrix decomposition $Q = LL^\top$ where $L \in \mathbb{R}^{n \times r}$ with $r = \text{rank}(Q)$. This decomposition can be obtained through Cholesky factorization when Q is positive definite, or through eigenvalue decomposition in the general case.

The quadratic constraint can then be reformulated as:

$$x^\top Q x + c^\top x + d \leq 0$$
$$x^\top L L^\top x + c^\top x + d \leq 0$$
$$\|L^\top x\|_2^2 + c^\top x + d \leq 0$$

Using the rotated second-order cone representation, we can reformulate the constraint as:

$$\left\| \begin{pmatrix} \frac{1+c^\top x+d}{2} \\ L^\top x \end{pmatrix} \right\|_2 \leq \frac{1 - c^\top x - d}{2} \tag{9}$$

This formulation is valid when $1 - c^\top x - d \geq 0$, which ensures that the right-hand side is non-negative. The constraint $c^\top x + d \leq 0$ from the original quadratic form is automatically satisfied when the SOC constraint holds.

For the convex quadratic objective function $\min_x \ x^\top Q x + c^\top x + d$, we can reformulate it using an epigraph variable τ:

$$\begin{aligned} \min_{x,\tau} \quad & \tau \\ \text{s.t.} \quad & x^\top Q x + c^\top x + d \leq \tau \end{aligned}$$

Using the matrix decomposition $Q = LL^\top$, this becomes:

$$\begin{aligned} \min_{x,\tau} \quad & \tau \\ \text{s.t.} \quad & \left\| \begin{pmatrix} \frac{1-\tau+c^\top x+d}{2} \\ L^\top x \end{pmatrix} \right\|_2 \leq \frac{1 + \tau - c^\top x - d}{2} \end{aligned}$$

B.3 Target Mappings for SOCP

Then, we propose all our target mappings.

Definition B.1 (Target mappings)**.** Let $\mathcal{G}_{\text{SOCP}}$ be a graph representation of a SOCP problem. We define the following target mappings.

- **Feasibility mapping:** We define $\Phi_{\text{feas}}(\mathcal{G}_{\text{SOCP}}) = 1$ if the SOCP problem is feasible and $\Phi_{\text{feas}}(\mathcal{G}_{\text{SOCP}}) = 0$ otherwise.
- **Boundedness mapping:** For a feasible SOCP problem, we define $\Phi_{\text{bound}}(\mathcal{G}_{\text{SOCP}}) = 1$ if the SOCP problem is bounded and $\Phi_{\text{bound}}(\mathcal{G}_{\text{SOCP}}) = 0$ otherwise.

- **Optimal value mapping:** For a feasible and bounded SOCP problem, we set $\Phi_{\text{opt}}(\mathcal{G}_{\text{SOCP}})$ to be its optimal objective value.

- **Solution Attainability Mapping :** For a feasible and bounded SOCP problem, its optimal value (infimum) is finite, but this value is not necessarily attained by a feasible point. Therefore, we introduce a mapping $\Phi_{\text{attain}}(\mathcal{G}_{\text{SOCP}})$ which equals 1 if an optimal solution exists, and 0 otherwise.

- **Optimal solution mapping:** For an SOCP problem that admits a solution, its optimal solution might not be unique. Therefore, we define the optimal solution mapping to be $\Phi_{\text{sol}}(\mathcal{G}_{\text{SOCP}}) = x^*$, where x^* is the solution with the smallest l_2 norm of the corresponding SOCP

B.4 An Example for SOCP graphs

Figure 5 is an example of a toy SOCP and its corresponding graph representation:

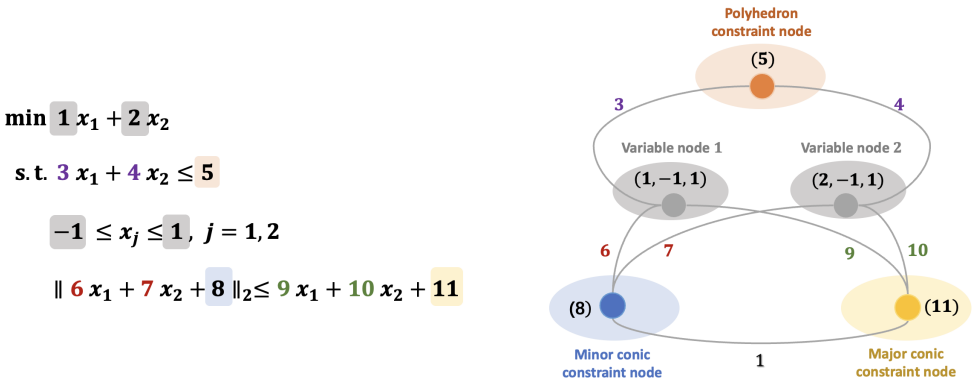


Figure 5: A toy SOCP instance with its graph representation

B.5 Complexity Comparison with SOTA Works:

Complexity for representing convex QCQP: We discuss further about what we mentioned in remark 1, 2, 3. For a convex QCQP instance with m quadratic constraints and n variables, where the i-th constraint matrix has rank $r_i \leq n$, our graph representation requires $n + m + 2 + \sum_{i=0}^{m}(r_i + 1)$ nodes while the architecture in [6] requires $n + m + \frac{1}{2}n(n+1)$ nodes and architecture in [7] requires $m + n + mn$ "nodes" that need to be updated dynamically. It's noteworthy that our graph representation only uses sparse connections between these nodes via using minor conic nodes as a **sparse intermediate information passing layer** between variables and conic constraints. As a result, our SOCP-GNN requires only $\mathcal{O}(n(\sum_{i=0}^{m} r_i))$ messages per iteration. This is in sharp contrast to the architecture by [6], which models each quadratic term explicitly and thus incurs a much higher per-iteration cost of $\mathcal{O}(n^3 + mn^2)$. And result in [7] use $\mathcal{O}(mn^2)$ messages each iteration.

Reducing the Node Complexity of SOCP-GNNs: One may note that for SOC constraints $\|Ax + b\|_2 \leq c^\top x + d$ with $A \in \mathbb{R}^{k \times n}$ of a large $k \gg n$, the GNN need k minor conic constraint nodes to represent it. However, as shown in Appendix B.1, we can reduce the complexity to $\mathcal{O}(n)$ by reformulating it into another equivalent SOC constraint with corresponding $A' \in \mathbb{R}^{k' \times n}$ of $k' \leq n + 1$. This reformulation makes SOCP-GNN more scalable for the large and structured problems in real-world applications.

Algorithm 1 The WL test for SOCP-Graphs (denoted by WL_{SOCP})

Require: A graph instance $\mathcal{G} = (V, E)$ with node sets V_1, V_2, V_3, V_4, initial node features h^v, h^s, h^o, h^q, and an iteration limit $L > 0$.
Initialize initial colors for all nodes:
$C^{0,v} \leftarrow \text{HASH}_{0,V}(h^v), \quad \forall v \in V_1$
$C^{0,s} \leftarrow \text{HASH}_{0,S}(h^s), \quad \forall s \in V_2$
$C^{0,o} \leftarrow \text{HASH}_{0,O}(h^o), \quad \forall o \in V_3$
$C^{0,q} \leftarrow \text{HASH}_{0,Q}(h^q), \quad \forall q \in V_4$
for $l = 1$ **to** L **do**
  Update colors for polyhedron constraint nodes (V_2):
  $C^{l,s} \leftarrow \text{HASH}\Big(C^{l-1,s}, \sum_{v \in V_1} w_{s,v}\text{HASH}(C^{l-1,v})\Big)$
  Update colors for minor conic constraint nodes (V_3):
  $\bar{C}^{l-1,o} \leftarrow \text{HASH}\Big(C^{l-1,o}, \sum_{v \in V_1} w_{o,v}\text{HASH}(C^{l-1,v})\Big)$
  Update colors for major conic constraint nodes (V_4):
  $\bar{C}^{l-1,q} \leftarrow \text{HASH}\Big(C^{l-1,q}, \sum_{v \in V_1} w_{q,v}\text{HASH}(C^{l-1,v})\Big)$
  Update colors for major conic constraint nodes (V_4):
  $C^{l,q} \leftarrow \text{HASH}\Big(\bar{C}^{l-1,q}, \sum_{o \in V_3} w_{q,o}\text{HASH}(\bar{C}^{l-1,o})\Big)$
  Update colors for minor conic constraint nodes (V_3):
  $C^{l,o} \leftarrow \text{HASH}\Big(\bar{C}^{l-1,o}, \sum_{q \in V_4} w_{o,q}\text{HASH}(C^{l,q})\Big)$
  Update colors for variable nodes (V_1):
  $C^{l,v} \leftarrow \text{HASH}\Big(C^{l-1,v}, M_1, M_2, M_3\Big)$,where:

$$M_1 = \sum_{s \in V_2} w_{v,s}\text{HASH}(C^{l,s})$$

$$M_2 = \sum_{o \in V_3} w_{v,o}\text{HASH}(C^{l,o})$$

$$M_3 = \sum_{q \in V_4} w_{v,q}\text{HASH}(C^{l,q})$$

end for
Return The multisets of final colors: $\{\{C^{L,v}\}\}_{v \in V_1}, \{\{C^{L,s}\}\}_{s \in V_2}, \{\{C^{L,o}\}\}_{o \in V_3}, \{\{C^{L,q}\}\}_{q \in V_4}$

C Proof of Main Theorem

C.1 SOCP WL-test

The separation power of traditional GNNs is closely related to the Weisfeiler-Lehman (WL) test, a classical algorithm to identify whether two given graphs are isomorphic. To apply the WL test on SOCP-graphs, we design a modified WL test in Algorithm 1.

We denote Algorithm 1 by $\text{WL}_{\text{SOCP}}(\cdot)$ and we assume that there is no collision of Hash functions and we say that two SOCP-graphs $G, \hat{G}$ can be distinguished by Algorithm 1 if and only if there exist a positive integer L and injective hash functions mentioned above such that the output multisets of $G, \hat{G}$ are different.

C.2 The connection between the WL-indistinguishablity and target property

Here, we analyze the WL test's convergence and corresponding stable properties to lead to the core lemma

Lemma 1. *Assume all hash functions are real-valued and collision-free, and we terminate the SOCP WL-test when the number of distinct colors no longer changes in an iteration. Then the SOCP WL-test terminates in finite iterations.*

Proof. Here, notice that the SOCP WL-test satisfies the following two properties:

- If two nodes v, w have different colors in one (sub)iteration, then they will always have different colors in the following (sub)iterations.
- If after one full iteration, the nodes' color doesn't change under some one-to-one color mapping, then for all iterations after this iteration, the algorithm will always return the same result.

These two facts have shown that, after one iteration, the color collections either get strictly finer or remain unchanged. Since the number of nodes is finite, the algorithm terminates in finite iterations. □

And now, we study the convergence properties of the SOCP-WL test

Lemma 2. *Given the SOCP graph G, assume the SOCP WL-test stabilizes after $T \geq 0$ iterations. The sum of weights from a certain node of one color to all nodes of another color only depends on the color of the given node. Specifically, the sum (taking W_1 for variable nodes and W_2 for polyhederon constraint nodes as an example) is:*

$$S(W_2, W_1; G) := \sum_{C^{T,v}=W_1} w_{s,v}$$

and is well-defined for all s, such that $C^{T,s} = W_2$

Similarly, for any color of variables W_1, color of polyhedron constraints W_2, color of minor conic constraints W_3 and color of major conic constraints W_4, the following sums are well-defined:

$$\begin{aligned}
S(W_3, W_1; G) &:= \sum_{C^{T,v}=W_1} w_{o,v}, \quad C^{T,o} = W_3 \\
S(W_4, W_3; G) &:= \sum_{C^{T,o}=W_3} w_{q,o}, \quad C^{T,q} = W_4 \\
S(W_1, W_2; G) &:= \sum_{C^{T,s}=W_2} w_{v,s}, \quad C^{T,v} = W_1 \\
S(W_1, W_3; G) &:= \sum_{C^{T,o}=W_3} w_{v,o}, \quad C^{T,v} = W_1 \\
S(W_1, W_4; G) &:= \sum_{C^{T,q}=W_4} w_{v,q}, \quad C^{T,v} = W_1 \\
S(W_4, W_1; G) &:= \sum_{C^{T,v}=W_1} w_{q,v}, \quad C^{T,q} = W_4
\end{aligned}$$

Proof. Let v_1, v_2 be two nodes with color $W_1 = C^{T,v_1} = C^{T,v_2}$. Since the SOCP WL-test has stabilized, the node pairs won't be finer, i.e.

$$\sum_s w_{v_1,s}\text{HASH}(C^{T,s}) = \sum_s w_{v_2,s}\text{HASH}(C^{T,s}).$$

Rearranging according to $W_2 = C^{T,s}$, we get:

$$\sum_{W_2} \sum_{C^{T,s}=W_2} w_{v_1,s}\text{HASH}(W_2) = \sum_{W_2} \sum_{C^{T,s}=W_2} w_{v_2,s}\text{HASH}(W_2).$$

Assuming that the hash function is collision-free and maps different colors into different linearly independent vectors, we conclude that:

$$\sum_{C^{T,s}=W_2} w_{v_1,s} = \sum_{C^{T,s}=W_2} w_{v_2,s},$$

i.e., $S(W_1, W_2; G) := \sum_{C^{T,s}=W_2} w_{v,s}, \quad C^{T,v} = W_1$ is well-defined.

Other proofs are similar.

□

An immediate conclusion is listed following.

Corollary 1. *If The SOCP WL-test cannot separate the two instances: $\mathcal{I}, \hat{\mathcal{I}}$ (with given sizes $n, m, , k_1, ..., k_m, b$, encoded by $G, \hat{G} \in \mathcal{G}_{SOCP}^{n,m,,k_1,...,k_m,b}$), then: All the sum in lemma 3 is well defined for G and $\hat{G}$ and equal each other respectively.*

Meanwhile, we define: W_{ij} to be the collection of nodes with node type i and color j. By summing the cross terms and rearranging the sum, we have:

$$S(W_{1j}, W_{2k}; G)|W_{1j}| = S(W_{2k}, W_{1j}; G)|W_{2k}|$$
$$S(W_{1j}, W_{3l}; G)|W_{1j}| = S(W_{3l}, W_{1j}; G)|W_{3l}|$$
$$S(W_{1j}, W_{4m}; G)|W_{1j}| = S(W_{4m}, W_{1j}; G)|W_{4m}|$$

.

Now, we begin to prove the following lemma.

Lemma 3. *Let $\mathcal{I}, \hat{\mathcal{I}}$ (with given sizes $n, m, , k_1, ..., k_m, b$, encoded by $G, \hat{G} \in \mathcal{G}_{SOCP}^{n,m,,k_1,...,k_m,b}$) be two SOCP instances. If the following holds:*

- *The SOCP WL-test cannot separate the two instances;*
- *x is a feasible solution of $\mathcal{I}$.*

Then there exists a feasible solution $\hat{x}$ for $\hat{\mathcal{I}}$ whose objective and ℓ_2-norm are controlled by x, such that:

$$\hat{e} \cdot \hat{x} \leq e \cdot x$$
$$||\hat{x}||_2 \leq ||x||_2$$

Proof. The key to this proof is to take the average among the nodes in the same node pair. Formally, we define $\hat{x}_v = \frac{1}{|W_{1j}|}(\sum_{C^{T,v'}=W_{1j}} x_{v'})$ for all v, such that: $C^{T,v} = W_{1j}$

By the Cauchy-Schwarz inequality, we have:

$$\sum_{C^{T,v'}=W_{1j}} x_{v'}^2 \geq |W_{1j}|[\frac{1}{|W_{1j}|}(\sum_{C^{T,v'}=W_{1j}} x_{v'})]^2$$

Summing over all possible W_{1j}, we get: $||\hat{x}||_2 \leq ||x||_2$

Meanwhile, notice that: for all v', such that: $C^{T,v'} = W_{1j}$, their corresponding $e_{v'}, l_{v'}, r_{v'}$ and $\hat{e}_{v'}, \hat{l}_{v'}, \hat{r}_{v'}$ are the same, respectively.

Hence, we have:

$$\sum_{C^{T,v'}=W_{1j}} e_{v'} x_{v'} = \sum_{C^{T,v'}=W_{1j}} \hat{e}_{v'} \hat{x}_{v'}$$
$$\hat{x}_v \in [\hat{l}_v, \hat{r}_v]$$

for all possible variable node v and color W_{1j}.

Summing over W_{1j} yields:

$$\sum_{v'} e_{v'} x_{v'} = \sum_{v'} \hat{e}_{v'} \hat{x}_{v'}$$

Further, consider the edge properties brought by the above lemma, we get:

For the l-th and t-th polyhedron constraint both with color W_{2k}, $l, t \in \{1, 2, \cdots, b\}$ (Here, we assume in both G and $\hat{G}$,the l-th and t-th polyhedron constraint are both with color W_{2k} respectively) the following inequality holds:

$$\sum_{j=1}^{n} F_{l,j} x_j \leq g_t, \quad \Rightarrow \quad \frac{1}{|W_{2k}|} \sum_{l \in W_{2k}} \sum_{W_{1j}} \sum_{v \in W_{1j}} F_{l,v} x_v \leq g_t,$$

Exchange the order of the sum, we get:

$$\frac{1}{|W_{2k}|} \sum_{W_{1j}} \sum_{v \in W_{1j}} \sum_{l \in W_{2k}} F_{l,v} x_v \leq g_t,$$

Notice that:

$$\begin{aligned}
\frac{1}{|W_{2k}|} \sum_{W_{1j}} \sum_{v \in W_{1j}} \sum_{l \in W_{2k}} F_{l,v} x_v &= \frac{1}{|W_{2k}|} \sum_{W_{1j}} \sum_{v \in W_{1j}} \sum_{l \in W_{2k}} F_{l,v} \hat{x}_v \\
&= \frac{1}{|W_{2k}|} \sum_{W_{1j}} \sum_{v \in W_{1j}} \sum_{l \in W_{2k}} \hat{F}_{l,v} \hat{x}_v \\
&= \sum_{W_{1j}} \sum_{v \in W_{1j}} \hat{F}_{l,v} \hat{x}_v
\end{aligned}$$

Thus, $\sum_{W_{1j}} \sum_{v \in W_{1j}} \hat{F}_{l,v} \hat{x}_v \leq g_t = \hat{g}_t = \hat{g}_l$, which shows that the polyhedron constraint is satisfied.

Similarly, for the u-th and r-th conic constraints both with color W_{4m}, we have:

- After proper rearranging of nodes o_{us_u} and o_{rs_r}, where $s_u = 1, 2, ..., k_u; s_r = 1, 2, ..., k_r$, the color of o_{us_u} and o_{rs_r}, where $s_u = 1, 2, ..., k_u; s_r = 1, 2, ..., k_r$, are the same regarding the order, i.e. $C^{T,o_{ui}} = C^{T,o_{ri}}, \forall i = 1, 2, ..., k_u$ (Notice that :$k_u = k_r$).
- d_u=d_r.
- For any node o_{hi} and $o_{jk} \in V_3$ with the same stable color, either $h = j$ or the color of node q_h and q_j are the same.

Now let's prove the conic part. For the u-th and r-th major conic constraint node both with color W_{4m} in both G and $\hat{G}$ and the minor conic node j_1 corresponds to r-th major conic constraint node in both G and $\hat{G}$, we have:

Right constraint:

$$\begin{aligned}
&\frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} c_u^T x + d_u \\
&= (\frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} \sum_{W_{1j}} \sum_{v \in W_{1j}} c_{uv} x_v) + \hat{d}_r \\
&= (\frac{1}{|W_{4m}|} \sum_{W_{1j}} \sum_{v \in W_{1j}} \sum_{C^{T,u}=W_{4m}} c_{uv} x_v) + \hat{d}_r \\
&= (\frac{1}{|W_{4m}|} \sum_{W_{1j}} \sum_{v \in W_{1j}} S(W_{1j}, W_{4m}; G) x_v) + \hat{d}_r \\
&= (\frac{1}{|W_{4m}|} \sum_{W_{1j}} \sum_{v \in W_{1j}} S(W_{1j}, W_{4m}; G) \hat{x}_v) + \hat{d}_r
\end{aligned}$$

$$= (\frac{1}{|W_{4m}|}\sum_{W_{1j}}|W_{1j}|S(W_{1j},W_{4m};G)\hat{x}_v) + \hat{d}_r$$
$$= (\frac{1}{|W_{4m}|}\sum_{W_{1j}}|W_{4m}|S(W_{4m},W_{1j};G)\hat{x}_v) + \hat{d}_r$$
$$= (\sum_{W_{1j}} S(W_{4m},W_{1j};G)\hat{x}_v) + \hat{d}_r = (\sum_{W_{1j}}\sum_{v\in W_{1j}} \hat{c}_{rv}\hat{x}_v) + \hat{d}_r = \hat{c}_r^T\hat{x} + \hat{d}_r$$

Left Constraint: Recall: two nodes in V_3 has the same stable color W_{3l} if their corresponding major conic constraint node's stable color is the same. So for each stable color W_{3l} in V_3, the corresponding major conic node's stable colors are all the same, denoted by W_{4m}. And each major conic node have $\frac{|W_{3l}|}{|W_{4m}|}$ minor nodes with stable color W_{3l}. And we use $j \in u$ denotes a minor conic node j corresponds to node u

$$\frac{1}{|W_{4m}|}\sum_{C^{T,u}=W_{4m}}\sum_{j\in W_{3l},j\in u}\frac{|W_{4m}|}{|W_{3l}|}(A_ux+b_u)_j$$
$$= (\hat{b}_r)_{j_1} + \frac{1}{|W_{3l}|}\sum_{C^{T,u}=W_{4m}}\sum_{j\in W_{3l},j\in u}(A_ux)_j$$
$$= (\hat{b}_r)_{j_1} + \frac{1}{|W_{3l}|}\sum_{C^{T,u}=W_{4m}}\sum_{j\in W_{3l},j\in u}\sum_{W_{1h}}\sum_{v\in W_{1h}}(A_u)_{jv}x_v$$
$$= (\hat{b}_r)_{j_1} + \frac{1}{|W_{3l}|}\sum_{W_{1h}}\sum_{v\in W_{1h}}\sum_{C^{T,u}=W_{4m}}\sum_{j\in W_{3l},j\in u}(A_u)_{jv}x_v$$
$$= (\hat{b}_r)_{j_1} + \frac{1}{|W_{3l}|}\sum_{W_{1h}}\sum_{v\in W_{1h}}S(W_{1h},W_{3l};G)x_v$$
$$= (\hat{b}_r)_{j_1} + \frac{1}{|W_{3l}|}\sum_{W_{1h}}\sum_{v\in W_{1h}}S(W_{1h},W_{3l};G)\hat{x}_v$$
$$= (\hat{b}_r)_{j_1} + \frac{1}{|W_{3l}|}\sum_{W_{1h}}|W_{1h}|S(W_{1h},W_{3l};G)\hat{x}_v$$
$$= (\hat{b}_r)_{j_1} + \frac{1}{|W_{3l}|}\sum_{W_{1h}}|W_{3l}|S(W_{3l},W_{1h};G)\hat{x}_v$$
$$= (\hat{b}_r)_{j_1} + \sum_{W_{1h}}S(W_{3l},W_{1h};G)\hat{x}_v$$
$$= (\hat{b}_r)_{j_1} + \sum_{W_{1h}}\sum_{v\in W_{1h}}\hat{(A_r)}_{j_1v}\hat{x}_v$$
$$= (\hat{b}_r)_{j_1} + (\hat{A}_r\hat{x})_{j_1} = (\hat{b}_r + \hat{A}_r\hat{x})_{j_1}$$

Conic feasibility for $\hat{I}$

For the u-th conic constraint with stable color $W_{4m}$and its corresponds minor conic node j with color W_{3l}, without loss of generality, we assume (A_ux+b_u) 's first $N = \frac{|W_{3l}|}{|W_{4m}|}$ rows' corresponding to all the minor conic nodes with color W_{3l} for such u and j, Thus, we have:

$$\sum_{j_1\in r,j_1\in W_{3l}}||(\hat{b}_r+\hat{A}_r\hat{x})_{j_1}||_2^2 = \frac{|W_{3l}|}{|W_{4m}|}||\frac{1}{|W_{4m}|}\sum_{C^{T,u}=W_{4m}}\sum_{j=1}^{N}\frac{|W_{4m}|}{|W_{3l}|}(A_ux+b_u)_j||_2^2$$
$$\leq \sum_{j=1}^{N}||\frac{1}{|W_{4m}|}\sum_{C^{T,u}=W_{4m}}(A_ux+b_u)_j||_2^2$$

Hence, summing over all possible W_{3l} for fixed W_{4m}, we have:

$$||\hat{b}_r + \hat{A}_r\hat{x}||_2^2 = \sum_{W_{3l}} \sum_{j_1 \in r, j_1 \in W_{3l}} ||(\hat{b}_r + \hat{A}_r\hat{x})_{j_1}||_2^2 \leq ||\frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} (A_u x + b_u)||_2^2$$

This yields that:

$$\begin{aligned} ||\hat{b}_r + \hat{A}_r\hat{x}||_2 &\leq ||\frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} (A_u x + b_u)||_2 \leq \frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} ||(A_u x + b_u)||_2 \\ &\leq \frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} c_u^T x + d_u = \hat{c}_r^T \hat{x} + \hat{d}_r \end{aligned}$$

Since r is arbitrarily chosen from W_{4m}, the conic feasibility holds obviously. □

Corollary 2. *If two SOCP instances $\mathcal{I}, \hat{\mathcal{I}}$ with their graph representations are indistinguishable by the SOCP-WL test, then the two problems share the same feasibility.*

Proof. Let x be a feasible solution for $\mathcal{I}$, then by lemma 3, there exists a feasible solution $\hat{x}$ for $\hat{\mathcal{I}}$. □

Corollary 3. *If two SOCP instances $\mathcal{I}, \hat{\mathcal{I}}$ with their graph representations are indistinguishable by the SOCP-WL test, then the two problems share the same boundness.*

Proof.

- If one instance is infeasible, by corollary 2, the other instance is infeasible as well, i.e., they are not bounded.
- If one instance is not bounded from below, denoted by $\mathcal{I}$. Since we can always find a feasible solution of $\hat{\mathcal{I}}$ which has a smaller objective value than any fixed feasible solution of $\mathcal{I}$ by Lemma 3, the conclusion is obvious.

□

Corollary 4. *If two SOCP instances $\mathcal{I}, \hat{\mathcal{I}}$ with their graph representations are indistinguishable by the SOCP-WL test, then the two problems share the same optimal objective value.*

Proof. By corollary 3, we only need to consider the case when both instances are feasible and bounded.

Notice that the feasibility with boundness may not lead to the existence of an optimal solution for SOCP problems, for example:

$$\min_{x_1,x_2} \quad x_1 \text{ s.t.} \qquad ||(2, x_1 - x_2)||_2 \leq x_1 + x_2 \, , x_1 \geq 0, x_2 \geq 0$$

So, we prove by "infimum" argument, let p and $\hat{p}$ be the optimal value of $\mathcal{I}$ and $\hat{\mathcal{I}}$ respectively. Then for any $\epsilon > 0$, there exists feasible solution x, s.t. $e^T x \leq p + \epsilon$. By lemma 4, there exists a feasible solution $\hat{x}$ of $\hat{\mathcal{I}}$, such that $\hat{p} \leq \hat{e}^T \hat{x} \leq e^T x \leq p + \epsilon$. Let $\epsilon \to 0$ yields $\hat{p} \leq p$. Similarly, we have: $\hat{p} \geq p$, which finishes the proof. □

Corollary 5. *If two SOCP instances $\mathcal{I}, \hat{\mathcal{I}}$ with their graph representations are indistinguishable by the SOCP-WL test and one of these instances admits an optimal solution, then the other instance has an optimal solution as well.*

Proof. See the proof of corollary 6. □

Corollary 6. *If two SOCP instances $\mathcal{I}, \hat{\mathcal{I}}$ with their graph representations are indistinguishable by the SOCP-WL test, then the two problems share the same optimal solution with the smallest Euclidean norm if one instance admits an optimal solution up to permutation.*

Proof. Without loss of generality, we assume that for each variable j, its corresponding stable color in $\mathcal{I}, \hat{\mathcal{I}}$ after the SOCP-WL test is the same, and $\mathcal{I}$ has an optimal solution x with the smallest Euclidean norm. By using lemma 3 twice, we can construct a feasible solution $\hat{x}$ for $\hat{\mathcal{I}}$ and construct a feasible solution $\hat{\hat{x}}$ for $\mathcal{I}$ again with

$$e^T x \geq \hat{e}^T \hat{x} \geq e^T \hat{\hat{x}} \quad \textit{and} \quad ||x||_2 \geq ||\hat{x}||_2 \geq ||\hat{\hat{x}}||_2$$

Hence, $x = \hat{\hat{x}}$. Recall the proof of the lemma 3, the variables in $\hat{x}$ with the same stable color after SOCP-WL test already have the same value, so averaging them again won't change it anymore, i.e. $\hat{x} = \hat{\hat{x}}$. Hence, $x = \hat{x} = \hat{\hat{x}}$. By corollary 4, $\hat{x}$ is an optimal solution of $\hat{\mathcal{I}}$

Now, if there exists an optimal solution y of $\hat{\mathcal{I}}$ with $||y||_2 < ||\hat{x}||_2 = ||x||_2$, by similar proof above, we can get: y is also an optimal solution of $\mathcal{I}$, which contradicts the fact that: x is the optimal solution of $\mathcal{I}$ with the smallest Euclidean norm. Hence, $\hat{x}$ is an optimal solution of $\hat{\mathcal{I}}$ with the smallest Euclidean norm □

C.3 The measurable property of target mapping

Definition C.1. For an SOCP instance G:

$$\begin{aligned} \text{minimize} \quad & e^T x \\ \text{subject to} \quad & \|A_i x + b_i\|_2 \leq c_i^T x + d_i, \quad i = 1, \ldots, m \\ & Fx \leq g \\ & l_i \leq x_i \leq r_i, i = 1, ..., n \end{aligned}$$

Its parameter is defined as $(e, \{A_i\}_{i=1}^m, \{b_i\}_{i=1}^m, \{c_i\}_{i=1}^m, \{d_i\}_{i=1}^m, F, g, l, r)$, and all the parameter form the parameter space $\mathcal{P}$

Notice that: For an SOCP instance, there exists a bijective mapping $\mathbf{I} : \mathcal{G}_{SOCP}^{n,m,,k_1,\ldots,k_m,b} \to \mathcal{P}$ with $\mathbf{I}(G) = (e, \{A_i\}_{i=1}^m, \{b_i\}_{i=1}^m, \{c_i\}_{i=1}^m, \{d_i\}_{i=1}^m, F, g, l, r)$ for any SOCP instance G parametrized by $(e, \{A_i\}_{i=1}^m, \{b_i\}_{i=1}^m, \{c_i\}_{i=1}^m, \{d_i\}_{i=1}^m, F, g, l, r)$. And we equip both $\mathcal{G}_{SOCP}^{n,m,,k_1,\ldots,k_m,b}$ and $\mathcal{P}$ with the standard Euclidean topology and product topology in its feature space. Then $\mathbf{I}$ is a homeomorphism.

Remark: If we can prove that $\Phi_{target} : \mathcal{P} \to \mathbb{R}$ is measurable, then $\Phi_{target} \circ \mathbf{I} : \mathcal{G}_{SOCP}^{n,m,,k_1,\ldots,k_m,b} \to \mathbb{R}$ is measurable as well.

Theorem 2. *For any Borel regular measure μ defined on $\mathcal{P}$, $\Phi_{feas} : \mathcal{P} \to \{0, 1\}$ is $\mu-$measurable.*

Proof. Below, we use measurable to denote $\mu-$measurable for simplicity.

To prove that Φ_{feas} is measurable, it suffices to show that the preimage of $\{1\}$, denoted $\mathcal{P}_{\text{feas}} = \{P \in \mathcal{P} \mid \Phi_{\text{feas}}(P) = 1\}$, is a measurable set.

First, we define a **feasibility violation function** $V_{\text{feas}} : \mathcal{P} \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$. Let $(y)_+ = \max(0, y)$.

$$V_{\text{feas}}(P, x) = \sum_{i=1}^{m} \left(\|A_i x + b_i\|_2 - (c_i^T x + d_i)\right)_+ + \sum_{j=1}^{p} \left((Fx)_j - g_j\right)_+ + \sum_{k=1}^{n} \left((l_k - x_k)_+ + (x_k - r_k)_+\right)$$

This function $V_{\text{feas}}(P, x)$ is continuous with respect to both P and x, as it is a sum and composition of continuous functions (norms, linear maps, max function). Furthermore, $V_{\text{feas}}(P, x) = 0$ if and only if x is a feasible point for the problem instance P.

A problem P is feasible if and only if there exists an $x \in \mathbb{R}^n$ such that $V_{\text{feas}}(P, x) = 0$. This is equivalent to the condition $\exists R \in \mathbb{N}^+, s.t. \inf_{x \in \mathbb{R}^n \cap B_R} V_{\text{feas}}(P, x) = 0$.

We can now express the set of feasible problems $\mathcal{P}_{\text{feas}}$ by restricting the infimum to a countable dense set. Let B_R be the closed ball of radius R centered at the origin. By continuity of V_{feas} in x, we have:

$$\mathcal{P}_{\text{feas}} = \bigcup_{R \in \mathbb{N}^+} \left\{ P \in \mathcal{P} \mid \inf_{x \in \mathbb{R}^n \cap B_R} V_{\text{feas}}(P, x) = 0 \right\}$$

$$= \bigcup_{R\in\mathbb{N}^+} \bigcap_{k\in\mathbb{N}^+} \left\{ P \in \mathcal{P} \mid \exists x \in \mathbb{R}^n \cap B_R, s.t. V_{\text{feas}}(P,x) < \frac{1}{k} \right\}$$

So, $\mathcal{P}_{\text{feas}}$ can be written as:

$$\mathcal{P}_{\text{feas}} = \bigcup_{R\in\mathbb{N}^+} \bigcap_{k\in\mathbb{N}^+} \bigcup_{x\in B_R\cap\mathbb{Q}^n} \left\{ P \in \mathcal{P} \mid V_{\text{feas}}(P,x) < \frac{1}{k} \right\}$$

For any fixed $x \in \mathbb{Q}^n$, the function $P \mapsto V_{\text{feas}}(P,x)$ is continuous. Thus, for each tuple (R,k,x), the set $\{P \mid V_{\text{feas}}(P,x) < 1/k\}$ is a Borel set. Since $\mathcal{P}_{\text{feas}}$ is formed by countable unions and intersections of measurable sets, it is itself a measurable (Borel) set. Therefore, Φ_{feas} is a measurable function.

□

Theorem 3. *For any Borel regular measure μ defined on $\mathcal{P}$, $\Phi_{bound} : \mathcal{P} \to \{0,1\}$ is $\mu-$measurable.*

Proof. Below, we use measurable to denote $\mu-$measurable for simplicity.

Let $\mathcal{P}_{\text{feas}} = \Phi_{\text{feas}}^{-1}(1)$, which is a measurable set. We only need to show that the set $\mathcal{P}_{\text{bdd}} = \{P \in \mathcal{P}_{\text{feas}} \mid \Phi_{\text{bound}}(P) = 1\}$ is measurable.

A problem $P \in \mathcal{P}_{\text{feas}}$ is bounded if and only if there exists $M \in \mathbb{Z}$ such that for all feasible solutions x of P, $e^T x \geq M$. This can be stated as:

$$\begin{aligned}\mathcal{P}_{\text{bdd}} &= \bigcup_{M\in\mathbb{Z}} \left\{ P \in \mathcal{P}_{\text{feas}} \mid \forall x \in \mathbb{R}^n, s.t. V_{\text{feas}}(P,x) = 0 \Rightarrow e^T x \geq M \right\} \\ &= \bigcup_{M\in\mathbb{Z}} \left\{ P \in \mathcal{P}_{\text{feas}} \mid \inf_{x\in\mathbb{R}^n, s.t. V_{\text{feas}}(P,x)=0} e^T x \geq M \right\}\end{aligned}$$

Let's define the **boundness violation function**:

$$V_{\text{bdd}}(P,x) = \inf_{x\in\mathbb{R}^n, s.t. V_{\text{feas}}(P,x)=0} e^T x$$

Now,we have:

$$\mathcal{P}_{\text{bdd}} = \bigcup_{M\in\mathbb{Z}} \{P \in \mathcal{P}_{\text{feas}} \mid V_{\text{bdd}}(P,x) \geq M\}$$

So it suffices to prove $V_{\text{bdd}}(P,x)$ is measurable and we only need to show that: for any $M \in \mathbb{R}$, the sublevel set $\{P \in \mathcal{P}_{\text{feas}} \mid V_{\text{bdd}}(P) < M\}$ is a measurable set.

The condition $V_{\text{bdd}}(P) < M$ is equivalent to the existence of a feasible point z such that $e^\top z < M$. This can be expressed as:

$$\{P \in \mathcal{P}_{\text{feas}} \mid V_{\text{bdd}}(P) < M\} = \bigcup_{k\in\mathbb{N}_+} \left\{ P \in \mathcal{P}_{\text{feas}} \mid \exists z \in \mathbb{R}^n \text{ s.t. } e^\top z \leq M - \frac{1}{k} \text{ and } z \text{ is feasible} \right\}.$$

Let us define an auxiliary violation function $V_{\text{bdd\_viol}} : \mathcal{P} \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}_{\geq 0}$:

$$V_{\text{bdd\_viol}}(P,z,M) = \max\left((e^\top z - M)_+, V_{\text{feas}}(P,z)\right).$$

This function is continuous in (P,z,M). The condition $V_{\text{bdd\_viol}}(P,z,M') = 0$ holds if and only if z is a feasible point and its objective value satisfies $e^\top z \leq M'$.

Thus, similar to the proof of feasibility, the condition $V_{\text{bdd}}(P) < M$ is equivalent to:

$$\bigcup_{k\in\mathbb{N}_+} \bigcup_{R\in\mathbb{N}^+} \left\{ P \in \mathcal{P}_{\text{feas}} \mid \inf_{z\in\mathbb{R}^n\cap B_R} V_{\text{bdd\_viol}}\left(P,z,M-\frac{1}{k}\right) = 0 \right\}.$$

By continuity of $V_{\text{bdd\_viol}}$ in z, we can restrict the infimum to the countable dense set $\mathbb{Q}^n$:

$$\bigcup_{k\in\mathbb{N}_+} \bigcup_{R\in\mathbb{N}^+} \left\{ P \in \mathcal{P}_{\text{feas}} \mid \inf_{z\in\mathbb{Q}^n\cap B_R} V_{\text{bdd\_viol}}\left(P,z,M-\frac{1}{k}\right) = 0 \right\}.$$

For any fixed $z \in \mathbb{Q}^n$, $R \in \mathbb{N}^+$ and $M' \in \mathbb{R}$, the function $P \mapsto V_{\text{bdd\_viol}}(P, z, M')$ is continuous, hence measurable. The infimum of a countable collection of such measurable functions is also measurable. Therefore, the set $\{P \mid \inf_{z\in\mathbb{Q}^n} V_{\text{bdd\_viol}}(P, z, M') = 0\}$ is measurable for any fixed M'.

Since the sublevel set $\{P \in \mathcal{P}_{\text{feas}} \mid V_{\text{bdd}}(P) < M\}$ is a countable union of such measurable sets, it is measurable. This holds for all $M \in \mathbb{R}$, so V_{bdd} is a measurable function. □

Theorem 4. *For any Borel regular measure μ defined on $\mathcal{P}$, $\Phi_{obj} : \mathcal{P} \to \mathbb{R}$ is $\mu-$measurable.*

Proof. Below, we use measurable to denote $\mu-$measurable for simplicity.

To prove that Φ_{obj} is measurable, we only need to show that for any $\phi \in \mathbb{R}$, the sublevel set $\{P \in \mathcal{P} \mid \Phi_{\text{obj}}(P) < \phi\}$ is measurable.

Let us define an **objective violation function** $V_{\text{obj}} : \mathcal{P} \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}_{\geq 0}$:

$$V_{\text{obj}}(P, x, \phi) = \max\left((e^T x - \phi)_+, V_{\text{feas}}(P, x)\right)$$

This function is continuous in (P, x, ϕ). $V_{\text{obj}}(P, x, \phi) = 0$ if and only if x is a feasible point and its objective value satisfies $e^T x \leq \phi$.

The condition $\Phi_{\text{obj}}(P) < \phi$ is equivalent to the existence of a feasible point x such that $e^T x < \phi$. This can be expressed as:

$$\{P \in \mathcal{P} \mid \Phi_{\text{obj}}(P) < \phi\} = \bigcup_{k\in\mathbb{N}^+} \{P \in \mathcal{P} \mid \exists x \in \mathbb{R}^n \text{ s.t. } e^T x \leq \phi - \frac{1}{k} \text{ and } x \text{ is feasible}\}$$

Similar to the previous proof, this is equivalent to:

$$\begin{aligned}
&\bigcup_{k\in\mathbb{N}^+} \bigcup_{R\in\mathbb{N}^+} \left\{P \in \mathcal{P} \mid \inf_{x\in\mathbb{R}^n\cap B_R} V_{\text{obj}}\left(P, x, \phi - \frac{1}{k}\right) = 0\right\} \\
= &\bigcup_{k\in\mathbb{N}^+} \bigcup_{R\in\mathbb{N}^+} \left\{P \in \mathcal{P} \mid \inf_{x\in\mathbb{Q}^n\cap B_R} V_{\text{obj}}\left(P, x, \phi - \frac{1}{k}\right) = 0\right\}
\end{aligned}$$

For any fixed $x \in \mathbb{Q}^n$, the function $P \mapsto V_{\text{obj}}(P, x, \phi')$ is continuous, hence measurable. The infimum of a countable collection of measurable functions is measurable. Hence, the set $\{P \mid \inf_{x\in\mathbb{Q}^n} V_{\text{obj}}(P, x, \phi') = 0\}$ is measurable for any fixed ϕ'. Since the sublevel set $\{P \mid \Phi_{\text{obj}}(P) < \phi\}$ is a countable union of such measurable sets, it is measurable. This holds for all $\phi \in \mathbb{R}$, so Φ_{obj} is a measurable function. □

Theorem 5. *For any Borel regular measure μ defined on $\mathcal{P}$, $\Phi_{attain} : \mathcal{P} \to \{0, 1\}$ is $\mu-$measurable.*

Proof. Below, we use measurable to denote $\mu-$measurable for simplicity.

Let $\mathcal{P}_{\text{fin}} = \Phi_{\text{obj}}^{-1}(\mathbb{R})$, which is a measurable set. We only need to show that the set $\mathcal{P}_{\text{sol}} = \{P \in \mathcal{P}_{\text{fin}} \mid \Phi_{\text{attain}}(P) = 1\}$ is measurable.

A problem $P \in \mathcal{P}_{\text{fin}}$ attains its optimal solution if and only if there exists a point $x \in \mathbb{R}^n$ such that x is feasible and its objective value is equal to the optimal value, $\Phi_{\text{obj}}(P)$. This can be stated as:

$$\mathcal{P}_{\text{sol}} = \left\{P \in \mathcal{P}_{\text{fin}} \mid \exists x \in \mathbb{R}^n \text{ s.t. } V_{\text{feas}}(P, x) = 0 \text{ and } e^T x = \Phi_{\text{obj}}(P)\right\}$$

Let's define the **optimality violation function**:

$$V_{\text{solu}}(P, x) = \max\left((e^T x - \Phi_{\text{obj}}(P))_+, V_{\text{feas}}(P, x)\right)$$

Notice that:

- For a fixed x, the function $P \mapsto V_{\text{solu}}(P, x)$ is measurable because it is a "composition" of continuous functions and the measurable function Φ_{obj}.
- For a fixed P, the function $x \mapsto V_{\text{solu}}(P, x)$ is continuous.

A SOCP instance P attains its solution if and only if there exists $R \in \mathbb{N}^+$, s.t. the infimum of $V_{\text{solu}}(P, x)$ over $x \in B_R$ is zero, i.e. :

$$\mathcal{P}_{\text{sol}} = \bigcup_{R\in\mathbb{N}^+} \left\{ P \in \mathcal{P}_{\text{fin}} \mid \inf_{x\in\mathbb{R}^n\cap B_R} V_{\text{solu}}(P, x) = 0 \right\}$$

Following the same logic used for Φ_{feas}, we can write:

$$\begin{aligned}\mathcal{P}_{\text{sol}} &= \bigcup_{R\in\mathbb{N}^+} \left\{ P \in \mathcal{P}_{\text{fin}} \mid \inf_{x\in\mathbb{R}^n\cap B_R} V_{\text{solu}}(P, x) = 0 \right\} \\ &= \bigcup_{R\in\mathbb{N}^+} \bigcap_{k\in\mathbb{N}^+} \left\{ P \in \mathcal{P}_{\text{fin}} \mid \inf_{x\in B_R\cap\mathbb{Q}^n} V_{\text{solu}}(P, x) < \frac{1}{k} \right\}\end{aligned}$$

For any fixed x, $P \mapsto V_{\text{solu}}(P, x)$ is measurable. The infimum over a countable set of measurable functions is measurable. Therefore, the set

$$\left\{ P \in \mathcal{P}_{\text{fin}} \mid \inf_{x\in B_R\cap\mathbb{Q}^n} V_{\text{solu}}(P, x) < \frac{1}{k} \right\}$$

is a measurable subset of $\mathcal{P}_{\text{fin}}$. Since $\mathcal{P}_{\text{sol}}$ is formed by countable unions and intersections of measurable sets, it is measurable. Thus, Φ_{attain} is a measurable function. □

Theorem 6. *For any Borel regular measure μ defined on $\mathcal{P}$, $\Phi_{solu} : \mathcal{P} \to \mathbb{R}^n$ is $\mu-$measurable.*

Proof. Below, we use measurable to denote $\mu-$measurable for simplicity.

For any $P \in \mathcal{P}_{sol} = \Phi^{-1}_{solu}(\mathbb{R}^n)$, Φ_{solu} is well-defined. And if suffices to prove that:$(\Phi_{\text{solu}})_i$ is measurable for any $i \in [n]$, i.e. for any $\phi \in \mathbb{R}$, the set: $\{P \in \mathcal{P}_{sol} \mid (\Phi_{solu})_i < \phi\}$ is measurable.

Notice that: the followings are equivalent for $P \in \mathcal{P}_{sol}$:

- $P \in \{P \in \mathcal{P}_{sol} \mid (\Phi_{solu})_i < \phi\}$.
- There exists $x \in \mathbb{R}^n$ with $x_i < \phi$, such that $V_{\text{solu}}(P, x) = 0$ and $V_{\text{solu}}(P, x') > 0$, $\forall x' \in B_{\|x\|}$, $x'_i \geq \phi$.
- There exists $R \in \mathbb{Q}_+$, $r \in \mathbb{N}_+$, and $x \in B_R$ with $x_i \leq \phi - 1/r$, such that $V_{\text{solu}}(P, x) = 0$ and $V_{\text{solu}}(P, x') > 0$, $\forall x' \in B_R$, $x'_i \geq \phi$.
- There exists $R \in \mathbb{Q}_+$ and $r \in \mathbb{N}_+$, such that for all $r' \in \mathbb{N}_+$, $\exists x \in B_R \cap \mathbb{Q}^n$, $x_i \leq \phi - 1/r$, s.t. $V_{\text{solu}}(P, x) < 1/r'$ and that $\exists r'' \in \mathbb{N}_+$, s.t., $V_{\text{solu}}(P, x') \geq 1/r''$, $\forall x' \in B_R \cap \mathbb{Q}^n$, $x'_i \geq \phi$.

Hence, we can rewrite $\{P \in \mathcal{P}_{sol} \mid (\Phi_{solu})_i < \phi\}$ as:

$$\bigcup_{R\in\mathbb{Q}_+} \bigcup_{r\in\mathbb{N}_+} \left(\begin{array}{l} \left(\bigcap_{r'\in\mathbb{N}_+} \bigcup_{x\in B_R\cap\mathbb{Q}^n,\, x_i\leq\phi-\frac{1}{r}} \left\{P \in \mathcal{P}_{sol} \mid V_{\text{solu}}(P, x) < \frac{1}{r'}\right\}\right) \\ \cap \left(\bigcup_{r''\in\mathbb{N}_+} \bigcap_{x'\in B_R\cap\mathbb{Q}^n,\, x'_i\geq\phi} \left\{P \in \mathcal{P}_{sol} \mid V_{\text{solu}}(P, x') \geq \frac{1}{r''}\right\}\right) \end{array} \right)$$

,

which is measurable. □

C.4 SOCP GNNs 's separation power's relation with SOCP-WL test's separation power

Remark 4. Thanks to the universality of MLPs, it's noteworthy that we can assume all learnable functions in SOCP-GNN are continuous in the following proof without loss of generality, since they are always parametrized by MLPs.

Theorem 7. *SOCP GNN has the same separation power as the SOCP-WL test.*

Proof. We only need to show: For any SOCP instance I and $\hat{I}$, encoded by $G, \hat{G}$, respectively, the following holds:

- For graph-level output, two instances can't be separated by $\mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R})$, i.e.,

$$F(G) = F(\hat{G}), \quad \forall F \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R})$$

  if and only if the two instances can't be separated by the SOCP-WL test either.

- For node-level output, the two instances can't be separated by $\mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^n)$, i.e.,

$$F(G) = F(\hat{G}), \quad \forall F \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^n)$$

  if and only if the two instances can't be separated by the SOCP-WL test either with $\mathcal{C}^{T,v_j} = \mathcal{C}^{T,\hat{v}_j}$ hold for all $j \in [n]$, i.e. the variables are reindexed according to the SOCP-WL test for both instances.

We first prove that SOCP-GNN can simulate the SOCP WL-test for any fixed SOCP instance. This can be proved by showing that: For any special SOCP-WL test and given graph G, there exists an SOCP-GNN that can simulate arbitrary iterations of this test given the same input for G under the one-hot encoding.

Let $\mathcal{F}$ denote the set of all the initial features for all nodes in G. Then we select $\hat{g}_i^0, i = 1, 2, 3, 4$ to map these features in $\mathcal{F}$ to their one-hot encoding respectively by theorem 3.2 of [29]. So for any initial round in the SOCP-WL test, there exists an SOCP-GNN that can simulate it.

Assume now, we already have: we get an SOCP-GNN which can simulate the first t rounds of a special SOCP-WL test, so that: $h^{t,n}$ is just the one-hot encoding of $C^{t,n}$ for all nodes n. For the first refinement round for the polyhedron constraint node s, we choose f_1^t as an identity mapping, so that: if $\left(C^{t,s}, \sum_{v\in V_1} w_{s,v}\text{HASH}(C^{t,v})\right)$ and $\left(C^{t,s'}, \sum_{v\in V_1} w_{s',v}\text{HASH}(C^{t,v})\right)$ are different, then $\left(h^{t,s}, \sum_{v\in V_1} w_{s,v} f_1^t(h^{t,v})\right)$ and $\left(h^{t,s'}, \sum_{v\in V_1} w_{s',v} f_1^t(h^{t,v})\right)$ are different. Then, by Theorem 3.2 of [29], there exists 4-layered MLP $g_1^t(\cdot)$ with ReLU activation can map these inputs: $\left(h^{t,s}, \sum_{v\in V_1} w_{s,v} f_1^t(h^{t,v})\right)$ to their corresponding output in SOCP-WL test's one-hot encoding.

Similarly, we can prove that: there exists $\{g_i^t(\cdot)\}$ and $\{f_j^t(\cdot)\}$, such that the corresponding SOCP-GNN can simulate the t+1 round of the SOCP-WL test for G. By mathematical induction, for any possible output of G for SOCP-WL test, there exists SOCP-GNNs can output the corresponding one-hot encoding of the stable color, respectively. Consider the two possible outputs:

- Graph-level scalar output. In this case, we set

$$y = f_{\text{out}}\left(I_1, I_2, I_3, I_4\right)$$

- Node-level vector output. In this case, we only consider the output associated with the variable nodes in V_1, given by

$$y_i = f_{\text{out}}\left(h^{T,v_i}, I_1, I_2, I_3, I_4\right), i \in [n]$$

  where, $I_1 = \sum_{v\in V_1} h^{T,v}$, $I_2 = \sum_{s\in V_2} h^{T,s}$, $I_3 = \sum_{o\in V_3} h^{T,o}$, and $I_4 = \sum_{q\in V_4} h^{T,q}$.

If two instances $\mathcal{I}$ and $\hat{\mathcal{I}}$ can't be separated by any SOCP-GNNs but can be separated by some SOCP-WL test $\mathcal{W}$. By applying the results discussed above to the disjoint union of these two instances' corresponding graphs, we get: $h^{T,\cdot}$ is just one-hot encoding of $C^{T,\cdot}$, respectively. Then we can conclude that their output multisets under $\mathcal{W}$ are the same, which causes a contradiction. Hence, if two instances $\mathcal{I}$ and $\hat{\mathcal{I}}$ can't be separated by any SOCP-GNNs, then they can't be separated by any SOCP-WL test $\mathcal{W}$ as well. Similarly, we have:

For any node n', n'' in SOCP instance $\mathcal{I}, \hat{\mathcal{I}}$ respectively, if $h^{t,n'} = \hat{h}^{t,n''}, \forall F \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R})$ holds for any $t \in \mathbb{N}$, then n', n'' have the same stable color for any possible SOCP-WL test.

Now, assume wo instances $\mathcal{I}$ and $\hat{\mathcal{I}}$ can't be separated by any SOCP-WL test. Now, we show that:

$$C^{t,s} = \hat{C}^{t,s} \implies h^{t,s} = \hat{h}^{t,s'}, \quad \forall \text{ polyhedron constraint } s, s' \quad \text{and} \quad F \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}),$$

while a similar result can be derived for other sublayer-iterations using the same method.

When $t = 0$, the conclusion holds obviously.

When $t \geq 1$, assume the conclusion for all nodes holds for $t-1$, then we have: $\left(C^{t-1,s}, \sum_{v \in V_1} w_{s,v}\text{HASH}(C^{t-1,v})\right) = \left(\hat{C}^{t-1,s'}, \sum_{v \in V_1} \hat{w}_{s',v}\text{HASH}(\hat{C}^{t-1,v})\right)$

Hence, we have:

- $C^{t-1,s} = \hat{C}^{t-1,s'} \Rightarrow h^{t-1,s} = \hat{h}^{t-1,s'}$
- For any color W_{1j}, $\sum_{v \in W_{1j}} w_{s,v} = \sum_{v \in W_{1j}} \hat{w}_{s',v}$. This can be shown by assuming Hash function maps different colors to linearly independent vectors.
- For any color W_{1j}, $\sum_{v \in W_{1j}} w_{s,v} f_1^{t-1}(h^{t-1,v}) = \sum_{v \in W_{1j}} \hat{w}_{s',v} \hat{f}_1^{t-1}(\hat{h}^{t-1,v})$ (By inductive assumption for node v at iteration $t-1$)
- $\sum_{W_{1j}} \sum_{v \in W_{1j}} w_{s,v} f_1^{t-1}(h^{t-1,v}) = \sum_{W_{1j}} \sum_{v \in W_{1j}} \hat{w}_{s',v} \hat{f}_1^{t-1}(\hat{h}^{t-1,v})$.

Therefore, $h^{t,s} = \hat{h}^{t,s'}$, which finishes the proof. □

An immediate corollary is:

Corollary 7. *For any node n, n' in SOCP instance $\mathcal{I}, \hat{\mathcal{I}}$ respectively, $C^{t,n} = \hat{C}^{t,n'}$ holds for all possible SOCP-WL test and any $t \in \mathbb{N}$ if and only if $h^{t,n} = \hat{h}^{t,n'}, \forall F \in \mathcal{F}_{SOCP}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R})$ holds for any $t \in \mathbb{N}$.*

By the proof of lemma 3, you can see that:

Corollary 8. *For any node n, n' in SOCP instance $\mathcal{I}, \hat{\mathcal{I}}$ respectively, $C^{t,n} = \hat{C}^{t,n'}$ holds for all possible SOCP-WL test and any $t \in \mathbb{N}$ if and only if $h^{t,n} = \hat{h}^{t,n'}, \forall F \in \mathcal{F}_{SOCP}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R})$ holds for any $t \in \mathbb{N}$. Under such assumption, $(\Phi_{solution}(\mathcal{I}))_n = (\Phi_{solution}(\hat{\mathcal{I}}))_{n'}$*

C.5 Main theorem's proof

Consider the following theorems, which play an important role in real analysis:

Lusin theorem: Let μ be a Borel regular measure on $\mathbb{R}^n$ and let $f : \mathbb{R}^n \to \mathbb{R}^m$ be μ-measurable. Then for any μ-measurable $X \subset \mathbb{R}^n$ with $\mu(X) < \infty$ and any $\epsilon > 0$, there exists a compact set $E \subset X$ with $\mu(X \setminus E) < \epsilon$, such that $f|_E$ is continuous.

By this fundamental but important theorem, we get $\forall \epsilon > 0$, $\exists$ compact $X \subset \mathcal{G}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}$ with $\mu(\mathcal{G}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b} \setminus X) < \epsilon$, such that $\Phi_{target}|_X$ is continuous holds for any Φ_{target} mentioned in Definition B.1.

Moreover, using similar tricks in [8], we can assume that: X remains the same under the action of the permutation group S_n without loss of generality.

Generalized Stone-Weierstrass theorem:[Theorem 22 of [30]] Let X be a compact topology space and let $\mathbf{G}$ be a finite group that acts continuously on X and $\mathbb{R}^n$. Define the collection of all equivariant continuous functions from X to $\mathbb{R}^n$ as follows:

$$C_E(X, \mathbb{R}^n) = \{F \in C(X, \mathbb{R}^n) : F(g * x) = g * F(x),\ \forall x \in X, g \in \mathbf{G}\}.$$

Consider any $\mathcal{F} \subset C_E(X, \mathbb{R}^n)$ and any $\Phi \in C_E(X, \mathbb{R}^n)$. Suppose the following conditions hold:

(i) $\mathcal{F}$ is a subalgebra of $C(X, \mathbb{R}^n)$ and $\mathbf{1} \in \mathcal{F}$.

(ii) For any $x, x' \in X$, if $f(x) = f(x')$ holds for any $f \in C(X, \mathbb{R})$ with $f\mathbf{1} \in \mathcal{F}$, then for any $F \in \mathcal{F}$, there exists $g \in \mathbf{G}$ such that $F(x) = g * F(x')$.

(iii) For any $x, x' \in X$, if $F(x) = F(x')$ holds for any $F \in \mathcal{F}$, then $\Phi(x) = \Phi(x')$.

(iv) For any $x \in X$, it holds that $\Phi(x)_j = \Phi(x)_{j'}, \forall (j, j') \in J(x)$, where

$$J(x) = \{\{1, 2, \ldots, n\}^n : F(x)_j = F(x)_{j'}, \forall F \in \mathcal{F}\}.$$

Then for any $\epsilon > 0$, there exists $F \in \mathcal{F}$ such that

$$\sup_{x \in X} \|\Phi(x) - F(x)\| < \epsilon.$$

Now we leverage the theorems listed above to give a proof of the main theorem. And we let the group $\mathbf{G}$ to be permutation group S_n. Since our SOCP-GNNs are permutation-equivariant, they are obviously $\mathbf{G} - equivariant$ continuous functions. (The following a refers to 1 or n)

Property (i): $\mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^a)$ is a subalgebra of $C_E(X, \mathbb{R}^a)$

Proof. If suffices to prove this by using similar channel expansion techniques mentioned in [8]. □

Property (ii): For any $x, x' \in X$, if $f(x) = f(x')$ holds for any $f \in C(X, \mathbb{R})$ with $f\mathbf{1} \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^a)$, then for any $F \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^a)$, there exists $g \in \mathbf{G}$ such that $F(x) = g * F(x')$.

Proof. First notice that: $\mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}) \in C(X, \mathbb{R})$ with $f\mathbf{1} \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^a), \forall f \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R})$. Then applying theorem 7 and corollary 7 is enough. □

Property (iii) and (iv):

- For any $x, x' \in X$, if $F(x) = F(x')$ holds for any $F \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^a)$, then $\Phi(x) = \Phi(x')$.
- For any $x \in X$, it holds that $\Phi(x)_j = \Phi(x)_{j'}, \forall (j, j') \in J(x)$, where

$$J(x) = \{\{1, 2, \ldots, a\}^2 : F(x)_j = F(x)_{j'}, \forall F \in \mathcal{F}_{\text{SOCP}}^{n,m,,k_1,\ldots,k_m,b}(\mathbb{R}^a)\}$$

.

Proof. Applying theorems in Appendix C.2, theorem 7, and corollary 8 is enough. □

Applying the generalized Stone-Weierstrass theorem gives us Theorem 1 immediately.

C.6 Extension to p-order cone programming

A general p-order cone programming can be stated as:

$$\begin{aligned} \text{minimize} \quad & e^\top x \\ \text{subject to} \quad & Fx \leq g, \quad l \leq x \leq r, \\ & \|A_i x + b_i\|_p \leq c_i^T x + d_i, \quad i \in [m] \end{aligned} \tag{10}$$

where decision variables are $x \in \mathbb{R}^n$ and the problem parameters are $e \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{k_i \times n}$, $b_i \in \mathbb{R}^{k_i}$, $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $F \in \mathbb{R}^{b \times n}$, $g \in \mathbb{R}^b$, $l_j \in (\{-\infty\} \cup \mathbb{R})^n$, and $r \in (\{+\infty\} \cup \mathbb{R})^n$. Here, we only consider the case: $p \in [1, +\infty]$.

Here, we formally define some concepts that are helpful to the extension of p-order cone programming.

Definition C.2. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be **separable** if $f(x)$ can be expressed as a sum $f(x) = \sum_{j=1}^n f_j(x_j)$, where each function f_j only depends on the scalar x_j. (This definition is stricter than traditional "block separable".)

Definition C.3. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be equivalent (w.r.t. permutation group S_n) if for any rearranging $\{\sigma(1), \sigma(2), \cdots, \sigma(n)\}$ of $\{1, 2, \cdots, n\}$ and any $x \in \mathbb{R}^n$, $f(x_1, x_2, \cdots, x_n) = f(x_{\sigma(1)}, x_{\sigma(2)}, \cdots, x_{\sigma(n)})$

For $p \in [1, +\infty)$, we have: $\|x\|_p^p = \sum_{i=1}^n |x_i|^p$, which is separable and equivalent according to Definition C.2 and C.3.

Situation 1:Use p as a fixed parameter: We don't need to make any modifications to our architectures. As for the proof of the universality, we just need to change $||.||_2$ to $||.||_p$ for $p \geq 1$ in our proof of lemma 3 and other theorems in Appendix C, since our proof only uses the convexity, permutation-invariant property, continuous property, and separability of the l_2 norm, which holds for the l_p norm as well when $p \in [1, +\infty)$. As for $p = +\infty$, lemma 3 can be directly validated by noticing that:

$$\begin{aligned}
|(\hat{b}_r + \hat{A}_r\hat{x})_{j_1}| &= |\frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} \sum_{j\in W_{3l}, j\in u} \frac{|W_{4m}|}{|W_{3l}|}(A_u x + b_u)_j| \\
&\leq \frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} \sum_{j\in W_{3l}, j\in u} \frac{|W_{4m}|}{|W_{3l}|}|(A_u x + b_u)_j| \\
&\leq \frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} \sum_{j\in W_{3l}, j\in u} \frac{|W_{4m}|}{|W_{3l}|}(c_u x + d_u) = \frac{1}{|W_{4m}|} \sum_{C^{T,u}=W_{4m}} (c_u x + d_u) \\
&\leq \hat{c}_r^T \hat{x} + \hat{d}_r
\end{aligned}$$

, where the notions follow the settings in lemma 3. Since the above equation holds for all j_1, we can see that: lemma 3 still holds. Since $\|\cdot\|_\infty$ is continuous, the measurability holds as well.

Situation 2:Use p as a continuous parameter: Here, we need a little modification on our architectures and proofs, while we only consider $p \in [1, +\infty)$ since $\|x\|_p$ is continuous in p when $p \in [1, +\infty)$.

For the graph representation, we only need to augment our variable features from (e_i, l_i, r_i) to (e_i, l_i, r_i, p). [7] And the GNN and related WL test don't need any modification. As for the proof, it suffices to notice that:

- To prove lemma 3, we just need to observe that: If two instances $\mathcal{I}$ and $\hat{\mathcal{I}}$ can't be distinguished by the WL test, then their corresponding p must be the same. Then what remains is just the situation one's proof mentioned above. Other related results hold as well, like the equivalence of the WL test and GNN in separation power.
- As for the measurability, we just need to repeat what we do in Appendix C.3 while taking p as a parameter in the new parameter space.

Situation 3:Mix order conic programming: Here, similar to situation 2, we need to augment features for minor constraint nodes. For the constraint $\|A_i x + b_i\|_p \leq c_i^T x + d_i$, we reset the minor conic node j's feature to be $((b_i)_j, p)$. Then we can prove Lemma 3 by noticing that: two major conic constraints have the same color if and only if their corresponding p are the same. The measurability holds as well, similar to situation 1.

D Experiment Settings and Supplementary Results

D.1 Data generation

D.1.1 Generation of feasible SOCP instances

Following the SOCP generating scheme in CVXPY, we use the following steps to generate feasible and random SOCP instances, which admit at least an optimal solution.

(I) : Generate a secret point $x_s \in \mathbb{R}^n$ by sampling from a standard normal distribution, i.e., $x_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then generate the objective coefficient $e \sim \mathcal{N}(0, 0.25\mathbf{I})$.

(II) : Impose lower bounds and upper bounds on variables $l \leq x_s \leq r$ for the problem. Here $l = x_s - |\Delta_1| - 0.1, r = x_s + |\Delta_2| + 0.1$, where Δ_i are sampled i.i.d. from $\mathcal{N}(0, 0.25\mathbf{I})$ and $|\cdot|$ denotes component-wise absolute value.

[7] Here, we use $p = -1$ to encode $+\infty$ into feature.

(III) : Generate $F \in \mathbb{R}^{b\times n}$, whose nonzero entries are sampled i.i.d. from $\mathcal{N}(0, 0.01)$ and components are nonzero with probability 0.5. The vector g is subsequently sampled by $g = Fx_s + |\Delta_3| + 0.1$, where $\Delta_3 \sim \mathcal{N}(0, 0.25\mathbf{I})$.

(IV) : For each conic constraint, randomly sample the cone dimension (the number of rows of A_i, b_i) in $[1, 7]$ with equal probability. Then, generate A_i, c_i, b_i, whose nonzero entries are sampled i.i.d. from $\mathcal{N}(0, 0.0025)$. Each component of the coefficient matrix A_i, c_i is nonzero with probability 0.5. Then, generate $d_i = \|A_i x_s + b_i\|_2 - c_i^\top x_s + \epsilon$, where $\epsilon \sim \mathcal{U}(0.5, 1)$.

Step (II) ensures that the generated SOCP instances always have an optimal solution. Furthermore, the coefficients are intentionally sampled from distributions with different variances, introducing varying numerical scales to create more challenging test instances.

D.1.2 Generation of (possible) infeasible SOCP instances

We use the following steps to generate a (possible) infeasible SOCP instance with pre-determined probability $h \in [0, 1]$.

(I) : Sample a feasible SOCP instance by methods in Appendix D.1.1.

(II) : Execute step III-IV with probability h and execute step V-VI with probability $1 - h$.

(III) : Sample a random integer p in $[3, 20]$ and a scale coefficient $a \sim \mathcal{U}(0, 1)$. Then repeat step IV for p times

(IV) : Randomly choose a type of constraint to break with equal probability. If the polyhedral constraint is chosen, we randomly choose one component of g with equal probability, denoted by g_i, and then replace g_i by $(Fx_s)_i - \delta - 3$. If the conic constraint is chosen, we randomly choose one with equal probability and then replace its corresponding d_i by $\|A_i x_s + b_i\|_2 - c_i^\top x_s - \delta - 3$. Here $\delta \sim \mathcal{U}(0, a)$.

(V) : Sample a random integer p in $[3, 20]$ and a scale coefficient $a \sim \mathcal{U}(0, 1)$. Then repeat step VI for p times

(VI) : Randomly choose a type of constraint to enhance with equal probability. If the polyhedral constraint is chosen, we randomly choose one component of g with equal probability, denoted by g_i, and then replace g_i by $g_i + \delta$. If the conic constraint is chosen, we randomly choose one with equal probability and then replace its corresponding d_i by $d_i + \delta$. Here $\delta \sim \mathcal{U}(0, a)$.

D.1.3 Data generation for predicting optimal solutions:

We randomly generate 5000 feasible SOCP instances by methods in Appendix D.1.1 of size (50,10,10), (100,50,50), and (500,100,100) respectively. Each instance is solved in CVXPY to obtain a ground truth solution as the label. [8] Then, we divide these instances into training, validation, and test data classes by the ratio $8 : 1 : 1$.

D.1.4 Data generation for predicting the probability:

We randomly generate 5000 infeasible SOCP instances with probability $h = 0.5$ by methods in Appendix D.1.2 of size (50,10,10), (100,50,50) and (500,100,100) respectively. We use CVXPY to detect the feasibility of these instances as well. Then, we divide these instances into training class and validation class by the same ratio.

D.2 Implementations and training settings

For predicting the optimal solution, our SOCP-GNN is implemented with two message-passing layers. The learnable functions, denoted by $g_{l_1}^0$, $g_{l_2}^t$, $f_{l_3}^t$, and f_{out} (where $l_1 \in \{1, \ldots, 4\}$, $l_2 \in \{1, \ldots, 6\}$, and $l_3 \in \{1, \ldots, 8\}$), are all parameterized by neural networks. Specifically, $g_{l_1}^0$ and $g_{l_2}^t$ are simple linear layers, while $f_{l_3}^t$ and f_{out} are constructed with a single hidden layer containing 128 neurons.

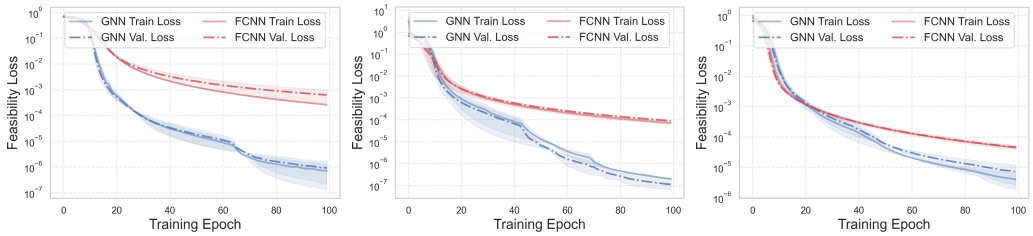
[8] We denote an SOCP instance by a tuple (n, b, m), where n represents the number of decision variables, b denotes the number of polyhedral constraints, and m indicates the number of second-order cone constraints.

For comparison, our baseline FCNN is implemented with three hidden layers, each containing 128 neurons. We use normalized MSE loss (Section 6) as the loss function.

For predicting the feasibility, our SOCP-GNN follows a similar structure to the one in solution prediction. Since the binary classification is simpler than the solution regression, we set the hidden layer with 16 neurons. For comparison, our baseline FCNN is implemented with three hidden layers, each containing 16 neurons. We use binary cross-entropy loss as the loss function.

All MLPs mentioned above use ReLU as the activation function. We use AdamW to optimize our learnable parameters with a maximum learning rate of 5×10^{-4} and a batch size of 40. All experiments are executed on an NVIDIA H200 GPU.

D.3 Results



(a) Small SOCP: $(50, 10, 10)$ (b) Medium SOCP $(100, 50, 50)$ (c) Large SOCP $(500, 100, 100)$

Figure 6: Comparison between the proposed GNN and FCNN for feasibility classification for random SOCP instances. The GNN uses approximately 0.01M parameters across three scales, while the FCNN uses approximately 0.07M, 0.7M, and 7M parameters for all three problem scales, respectively.

As shown in Figures 4 and 6, the proposed SOCP-GNN surpasses the baseline FCNNs in both optimal solution prediction and feasibility classification tasks. Across all problem scales—small, medium, and large—the GNN achieves substantially lower relative MSE and binary cross-entropy loss compared to the FCNN baseline. This superior performance is particularly evident in its parameter efficiency: on large-scale problems, the GNN, with only approximately 0.6M parameters, outperforms the FCNN, which requires 58M parameters for the solution prediction task, representing a nearly 100-fold reduction in model complexity. We also observe similar trends in the feasibility classification tasks.

This dramatic improvement in both performance and efficiency validates the effectiveness of exploiting the inherent sparse geometric structure of optimization problems through graph representations and message passing. These results confirm the potential of our approach as a scalable, data-driven framework for solving complex optimization problems.