On Representing Convex Quadratically Constrained Quadratic Programs via Graph Neural Networks

Chenyang Wu[1], Qian Chen[2,3], Akang Wang[2,4,*], Tian Ding[2], Ruoyu Sun[2,4], Wenguo Yang[1,*], and Qingjiang Shi[2,5]

[1]University of Chinese Academy of Sciences, China
[2]Shenzhen Research Institute of Big Data, China
[3]School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China
[4]School of Data Science, The Chinese University of Hong Kong, Shenzhen, China
[5]School of Computer Science and Technology, Tongji University, Shanghai, China

Reviewed on OpenReview: `https://openreview.net/forum?id=GC2Z06Asoa`

Abstract

Convex quadratically constrained quadratic programs (QCQPs) involve finding a solution within a convex feasible region defined by quadratic constraints while minimizing a convex quadratic objective function. These problems arise in various industrial applications, including power systems and signal processing. Traditional methods for solving convex QCQPs primarily rely on matrix factorization, which quickly becomes computationally prohibitive as the problem size increases. Recently, graph neural networks (GNNs) have gained attention for their potential in representing and solving various optimization problems such as linear programs and linearly constrained quadratic programs. In this work, we investigate the representation power of GNNs in the context of QCQP tasks. Specifically, we propose a new tripartite graph representation for general convex QCQPs and properly associate it with message-passing GNNs. We demonstrate that there exist GNNs capable of reliably representing key properties of convex QCQPs, including *feasibility*, *optimal value*, and *optimal solution*. Our result deepens the understanding of the connection between QCQPs and GNNs, paving the way for future machine learning approaches to efficiently solve QCQPs.

1 Introduction

Quadratic programs (QPs) are a pivotal class of optimization problems where the objective function is quadratic, and the constraints are typically linear or quadratic. Based on the nature of constraints, QPs can be further classified as *linearly constrained quadratic programs* (LCQPs) and *quadratically constrained quadratic programs* (QCQPs). When the objective and constraint matrices are positive semi-definite, the problem becomes a convex QCQP, making it both theoretically interesting and practically important. Convex QCQPs arise in various critical applications such as robust optimization in uncertain environments (Ben-Tal & Nemirovski, 2001; Boyd & Vandenberghe, 2004), power flow (Bienstock et al., 2020), and signal processing (Luo et al., 2010).

Solving QPs, especially those with quadratic constraints, presents significant challenges. Traditional methods often involve computationally intensive procedures that would struggle with scalability and real-time processing requirements. For example, the *interior-point method* (Nocedal & Wright, 1999) for a general n-variable QP involves solving a sequence of linear systems of equations, necessitating matrix decomposition with a runtime complexity of $\mathcal{O}(n^3)$. This leads to substantial computational burden in the large-scale case. Similarly, active-set algorithms (Gill et al., 2019), which work by iteratively adjusting the set of active constraints, can also become computationally demanding as the number of constraints and variables increase.

*Corresponding authors: Wenguo Yang <yangwg@ucas.ac.cn>, Akang Wang <wangakang@sribd.cn>

In recent years, advances in *machine learning* (ML) have opened new avenues for enhancing the solving process of QPs. There are mainly two categories of ML-aided QP methods. The first category aims to learn adaptive configurations of a specific QP algorithm or solver to accelerate the solving process (Bonami et al., 2018; Ichnowski et al., 2021; Jung et al., 2022), while the second focuses on predicting an initial solution of QPs, which is either directly taken as a final solution or further refined by subsequent algorithms or QP solvers (Bertsimas & Stellato, 2022; Gao et al., 2021; Sambharya et al., 2023; Tan et al., 2024; Wang et al., 2020; Xiong et al.). Additionally, Xiong et al. (2024) proposes a hypergraph-based framework for solving QCQPs. Most of these methods utilize *graph neural networks* (GNNs) to leverage the structural properties of graph-structured data, making them particularly well-suited for representing the relationships and dependencies inherent in QPs. By encoding QP instances into graphs, GNNs can capture intricate features and provide adaptive guidance or approximate solutions efficiently.

In addition to these studies, theoretical research on the expressive power of GNNs (Zhang et al., 2024; Li & Leskovec, 2022) and their relation to optimization problems has further strengthened the understanding of their capabilities. For instance, Chen et al. (2023a) and Chen et al. (2023b) established theoretical foundations for applying GNNs to solving *linear programs* (LPs) and *mixed-integer linear programs*, respectively. Further, such foundations are extended to LCQPs and their discrete variant, mixed-integer LCQPs in Chen et al. (2024).

Previous studies have empirically and theoretically demonstrated the utility of GNNs in speeding up existing QP solvers and directly approximating solutions for various QP instances. However, the question of *whether GNNs can accurately predict key properties of QCQPs, such as feasibility, optimal objective value, and optimal solution*, remains open. This paper aims to address the aforementioned gap by exploring both theoretical foundations and practical implementation of using GNNs for solving convex QCQPs. Specifically, we propose a tripartite graph representation for general convex QCQPs, and establish theoretical foundations of applying GNNs to optimize QCQPs. The distinct contributions of this paper can be summarized as follows.

- **Graph Representation**. We propose a novel tripartite graph representation for general QCQPs, which divides a QCQP into three types of nodes: linear-term, quadratic-term, and constraint nodes, with edges added between heterogeneous nodes to indicate problem parameters.
- **Theoretical Foundation**. We conduct analysis on the *separation power* as well as *approximation power* of *message-passing GNNs* (MP-GNNs). We showed that MP-GNNs are capable of capturing some key properties of convex QCQPs.
- **Empirical Evidence**. We conduct initial numerical tests of the tripartite MP-GNNs on small QCQP instances. The results showed that MP-GNNs can be trained to approximate the key properties well.

Building on the theoretical insights of our work, we establish a deeper understanding of the equivalence properties in convex QCQPs. By examining what key properties of QCQPs can be represented by GNNs, we gain a clearer view of QCQPs' intrinsic geometry and feasibility structure.

Although our work is primarily theoretical, it opens up interesting opportunities for future empirical investigation. In particular, our insights can inform the design of practical GNN-based solvers that improve efficiency in tackling convex QCQPs. Moreover, while our main focus is on convex QCQPs, many solvers for more general, non-convex QCQPs rely on relaxation steps involving convex QCQPs at intermediate stages. Consequently, our insights could help accelerate or guide the solving of these relaxation-based subproblems, offering a pathway to enhanced performance even in non-convex scenarios.

Notations

Throughout this paper, scalars or vectors are denoted by lowercase letters (e.g., a), and matrices are denoted by uppercase letters (e.g., A). For a vector a, we denote its i-th entry by a_i. For a matrix A, the entry in the i-th row and the j-th column is denoted by $a_{i,j}$. We use $\mathbf{0}$ and $\mathbf{1}$ to denote vectors or matrices with all-zero and all-one entries, respectively. For any positive integers m, n with $m < n$, we define $[m, n] \coloneqq \{m, m+1, \cdots, n\}$ to be the set of all integers ranging from m to n. For brevity, we define $[n] \coloneqq [1 : n] = \{1, 2, \cdots, n\}$.

2 Graph Representation of QCQPs

2.1 Quadratically Constrained Quadratic Programs

In this work, we study QCQPs defined in the following form:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) := \frac{1}{2}x^\top Q x + p^\top x \\ \text{s.t.} \quad & \frac{1}{2}x^\top Q^i x + (p^i)^\top x + b^i \leq 0 \quad \forall\, i \in [m] \\ & x^{\mathrm{L}} \leq x \leq x^{\mathrm{U}} \end{aligned} \tag{2.1}$$

where $Q, Q^i \in \mathbb{S}^{n\times n}$, $p, p^i \in \mathbb{R}^n$, $b^i \in \mathbb{R}$, $x^{\mathrm{L}} \in (\mathbb{R}\cup\{-\infty\})^n$, and $x^{\mathrm{U}} \in (\mathbb{R}\cup\{+\infty\})^n$. The problem has n optimization variables and m constraints. We refer to the tuple (m, n) as the *problem size* of QCQP. Both the objective function and the constraints are associated with quadratic functions. The QCQP problem is *convex* if Q and Q^i's are all positive semi-definite.

We denote the *feasible set* of Problem (2.1) by $\mathcal{X}$. If $\mathcal{X} \neq \emptyset$, the QCQP is said to be *feasible*; otherwise, it is said to be *infeasible*. A feasible QCQP is said to be *bounded* if the objective is bounded from below on $\mathcal{X}$, i,e., there exists $z \in \mathbb{R}$ such that $f(x) \geq z$ for every $x \in \mathcal{X}$; otherwise, it is said to be *unbounded*. For a feasible and bounded QCQP, $x^* \in \mathcal{X}$ is said to be an optimal solution if $f(x^*) \leq f(x)$ for every $x \in \mathcal{X}$. We remark that a QCQP always admits an optimal solution if it is feasible and bounded, but such an optimal solution might not be unique.

2.2 Tripartite Representation of QCQPs

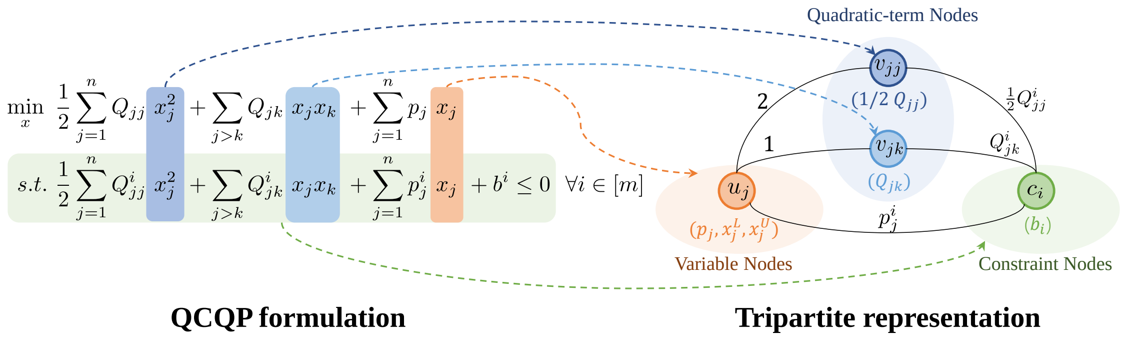
The first theoretical result demonstrating the representation power of GNNs in solving optimization problems was provided by Chen et al. (2023a). Their work employs a bipartite graph representation where variables and constraints are modeled as nodes. In this encoding, the linear constraint coefficients are edge features, the right-hand-side values are constraint node features, and the objective coefficients are variable node features. They showed that GNNs based on this graph representation can universally approximate the optimal solution of LPs, as well as properties of feasibility and boundedness. This bipartite graph modeling was later extended by encoding quadratic objective terms as edge features between variable nodes to analyze the representation power of GNNs for LCQPs (Chen et al., 2024).

Despite these advances, it remains challenging to develop graph representation to encode all information of general QCQPs while maintaining simplicity for GNN processing. Due to the presence of quadratic terms, a QCQP generally involves $\mathcal{O}(n^2 \times m)$ coefficients. Consequently, a graph encoding all QCQP information inherently exhibits a complexity of the same order, $\mathcal{O}(n^2 \times m)$. There are two natural extensions of the traditional bipartite representation of LPs/LCQPs to QCQPs.

– **Hyperedge Representation**. This approach adds hyperedges to the traditional bipartite graph to represent quadratic coefficients, turning the graph into a hypergraph. However, to the best of our knowledge, most of the current GNN architectures struggle to handle hyperedges.

– **Vector Feature Representation**. In this extension, all coefficients are encoded as features associated with the n variable nodes and the m constraint nodes, resulting in a graph with vector features of varying sizes, depending on the problem. However, existing GNNs are generally incapable of processing features of varying dimensions.

To fill this gap, we introduce an undirected *tripartite graph* representation $G_{\mathrm{QCQP}} := (V, E)$ that encodes all elements of a QCQP (2.1). Compared to the traditional bipartite graph modeling for LPs and LCQPs, our tripartite graph representation introduces an additional class of nodes to model the quadratic terms of variables. This modification allows us to represent QCQPs without any loss of information. In this paper, we will show that the tripartite representation enables GNNs to universally approximate solutions for convex QCQPs.



$\min_x \frac{1}{2}\sum_{j=1}^{n} Q_{jj}\, x_j^2 + \sum_{j>k} Q_{jk}\, x_j x_k + \sum_{j=1}^{n} p_j\, x_j$
$s.t.\ \frac{1}{2}\sum_{j=1}^{n} Q^i_{jj}\, x_j^2 + \sum_{j>k} Q^i_{jk}\, x_j x_k + \sum_{j=1}^{n} p^i_j\, x_j + b^i \le 0 \quad \forall i \in [m]$
Quadratic-term Nodes
v_{jj}
$(1/2\ Q_{jj})$
v_{jk}
(Q_{jk})
2
1
$\frac{1}{2}Q^i_{jj}$
Q^i_{jk}
u_j
(p_j, x_j^L, x_j^U)
p^i_j
c_i
(b_i)
Variable Nodes
Constraint Nodes
QCQP formulation
Tripartite representation

Figure 1: A tripartite representation of QCQPs. It consists of three types of nodes: variable nodes, quadratic-term nodes, and constraint nodes. All nodes and the edges connecting them are associated with coefficients from the formulation as features.

Formally, we model a QCQP as a tripartite graph with node sets representing variables, quadratic terms, and constraints.

- **Variable nodes (V_1):** Let $V_1 := \{u_1, \ldots, u_n\}$, where node u_j corresponds to variable x_j and is associated with the feature tuple $(p_j, x_j^{\mathrm{L}}, x_j^{\mathrm{U}})$.
- **Quadratic term nodes (V_2):** Let $\mathcal{L} := \{(j,k) \in [n] \times [n] : j \le k,\ |q_{j,k}| + \sum_{i=1}^{m} |q^i_{j,k}| > 0\}$ be the set of quadratic term indices with non-zero coefficients in either the objective or at least one constraint. Then, $V_2 := \{v_{j,k} : (j,k) \in \mathcal{L}\}$. The feature for node $v_{j,k}$ is $2q_{j,k}$ if $j > k$, and $q_{j,j}$ if $j = k$.
- **Constraint nodes (V_3):** Let $V_3 := \{c_1, \ldots, c_m\}$, where node c_i corresponds to the i-th constraint and is associated with the feature b_i.

The full node set is $V := V_1 \cup V_2 \cup V_3$. [*] The QCQP graph contains three edge sets, each with associated weights:

- **V_1–V_2 edges (E_{12}):** Connect variable node $u_{j'} \in V_1$ to quadratic node $v_{j,k} \in V_2$ if $j' = j$ or $j' = k$. The weight is defined as $w_{u_{j'}, v_{j,k}} := 1$ if $j > k$, and 2 otherwise.
- **V_1–V_3 edges (E_{13}):** Connect $u_j \in V_1$ to constraint node $c_i \in V_3$ if the linear coefficient $p^i_j \neq 0$. The weight is defined as $w_{u_j, c_i} := p^i_j$.
- **V_2–V_3 edges (E_{23}):** Connect $v_{j,k} \in V_2$ to $c_i \in V_3$ if the quadratic coefficient $q^i_{j,k} \neq 0$. The weight is defined as $w_{v_{j,k}, c_i} := 2q^i_{j,k}$ if $j > k$, and $q^i_{j,j}$ otherwise.

The full edge set is given by $E := E_{12} \cup E_{13} \cup E_{23}$. For any edge, we define $w_{x,y} := w_{y,x}$ for connected nodes x and y.

We illustrate this representation in Figure 1. We remark that there is a one-to-one mapping between a QCQP and its tripartite graph representation G_{QCQP}. Our representation has two major advantages:

– Our representation forms a simple graph, making it compatible with most existing GNNs. This facilitates the development of more efficient and task-specific GNN architectures.

– Our representation can leverage the sparsity nature of QCQP problems. That is, the numbers of nodes and edges are also controlled by the total number of nonzero coefficients. Thus, our representation can be efficiently applied to large and sparse instances.

[*]Throughout, we use u for variable nodes, v for quadratic term nodes, and c for constraint nodes. Indices i, j, and k refer to constraints, variables, and quadratic terms, respectively.

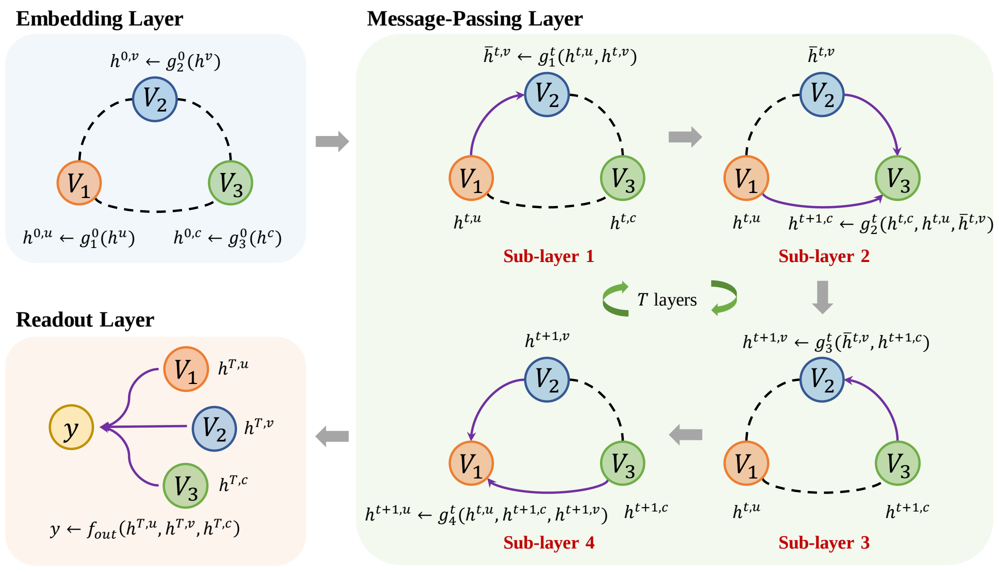

Embedding Layer
$h^{0,v} \leftarrow g_2^0(h^v)$
$h^{0,u} \leftarrow g_1^0(h^u)$
$h^{0,c} \leftarrow g_3^0(h^c)$
Message-Passing Layer
$\bar{h}^{t,v} \leftarrow g_1^t(h^{t,u}, h^{t,v})$
$h^{t+1,c} \leftarrow g_2^t(h^{t,c}, h^{t,u}, \bar{h}^{t,v})$
Sub-layer 1
Sub-layer 2
T layers
$h^{t+1,v} \leftarrow g_3^t(\bar{h}^{t,v}, h^{t+1,c})$
$h^{t+1,u} \leftarrow g_4^t(h^{t,u}, h^{t+1,c}, h^{t+1,v})$
Sub-layer 4
Sub-layer 3
Readout Layer
$y \leftarrow f_{out}(h^{T,u}, h^{T,v}, h^{T,c})$

Figure 2: An overview of the GNN architecture.

Definition 1 (Spaces of Convex QCQP-graphs). We denoted by $\mathcal{G}_{\text{QCQP}}^{m,n}$ the set of tripartite graph representations for all **convex** QCQPs with n variables and m constraints. [†]

3 Theoretical Results

3.1 Tripartite MP-GNNs

To study the capability of GNNs in representing QCQPs, we tailor the general MP-GNNs for the tripartite nature of the introduced QCQP graph representation. An overview is depicted in Figure 2.

Specifically, we consider the family of tripartite MP-GNNs consisting of an embedding layer, T message-passing layers (each comprised of four sub-layers), and a readout layer, detailed as follows:

- **Embedding Layer.** The initial features for all nodes are obtained by projecting their input features into a hidden space $\mathbb{R}^{h_0}$ using learnable embedding functions:

$$\begin{aligned} h^{0,u} &\leftarrow g_1^0(h^u) \quad \forall u \in V_1, \\ h^{0,v} &\leftarrow g_2^0(h^v) \quad \forall v \in V_2, \\ h^{0,c} &\leftarrow g_3^0(h^c) \quad \forall c \in V_3, \end{aligned}$$

  where h^u, h^v, and h^c are the input features of nodes in V_1, V_2, and V_3, respectively, and g_1^0, g_2^0, g_3^0 are the embedding functions for each node type.

- **Message-Passing Layer.** Each message-passing layer employs four distinct sub-layers to update node features, using learnable functions f_l^t and g_l^t. Each sub-layer updates the features of nodes in one set by aggregating messages from a specific subset of neighboring nodes, as detailed below.

[†]For any QCQP graph in $\mathcal{G}_{\text{QCQP}}^{m,n}$, the associated convex QCQP can be characterized by its coefficient tuple $(Q, \{Q^i\}_{i=1}^m, p, \{p^i\}_{i=1}^m, \{b^i\}_{i=1}^m, x^{\text{L}}, x^{\text{U}})$, where $Q, Q^i \in \mathbb{S}_+^n$. We define a topology on $\mathcal{G}_{\text{QCQP}}$: for Q, Q^i and p, p^i we use the topology induced by the norm of the linear mappings defined by the matrices and vectors, and for $x^{\text{L}}, x^{\text{U}}, b^i$ we use euclidean topology on $\mathbb{R}$ and discrete topology on the infinite values. In numerical experiments, we represent the infinite values by introducing an extra infinity indicator.

- **Sub-layer 1: Update Quadratic Nodes from Variables ($V_1 \to V_2$):**

$$\bar{h}^{t,v} \leftarrow g_1^t\left(h^{t,v}, \sum_{u\in V_1} w_{u,v} f_1^t(h^{t,u})\right) \quad \forall v \in V_2.$$

  This step computes an intermediate representation $\bar{h}^{t,v}$ for each quadratic node v based on incoming messages from variable nodes $u \in V_1$.

- **Sub-layer 2: Update Constraint Nodes from Variables and Quadratic Nodes (V_1+$V_2 \to V_3$):**

$$h^{t+1,c} \leftarrow g_2^t\left(h^{t,c}, m_{13}^t, m_{23}^t\right) \quad \forall c \in V_3,$$

  where $m_{13}^t \coloneqq \sum_{u\in V_1} w_{u,c} f_2^t(h^{t,u})$ and $m_{23}^t \coloneqq \sum_{v\in V_2} w_{v,c} f_3^t(\bar{h}^{t,v})$ denote the messages from variable nodes and quadratic nodes, respectively. This sub-layer produces the updated constraint node features for the next step.

- **Sub-layer 3: Update Quadratic Nodes from Constraints ($V_3 \to V_2$):**

$$h^{t+1,v} \leftarrow g_3^t\left(\bar{h}^{t,v}, \sum_{c\in V_3} w_{c,v} f_4^t(h^{t+1,c})\right) \quad \forall v \in V_2.$$

  Here, the intermediate representations $\bar{h}^{t,v}$ are updated using the new constraint node features $h^{t+1,c}$ to produce the final quadratic node features for this layer.

- **Sub-layer 4: Update Variable Nodes from Constraints and Quadratic Nodes (V_3+$V_2 \to V_1$):**

$$h^{t+1,u} \leftarrow g_4^t\left(h^{t,u}, m_{31}^t, m_{21}^t\right) \quad \forall u \in V_1,$$

  where $m_{31}^t \coloneqq \sum_{c\in V_3} w_{c,u} f_5^t(h^{t+1,c})$ and $m_{21}^t \coloneqq \sum_{v\in V_2} w_{v,u} f_6^t(h^{t+1,v})$ are the messages from constraint nodes and the updated quadratic nodes, respectively. This completes the update cycle for the layer.

The four sub-layers implement a structured, alternating update scheme. The first two sub-layers propagate information *from* variable nodes *to* constraint nodes, passing through the quadratic nodes to incorporate information about quadratic terms. This allows the constraint node representations to be informed by the current state of the variables and their quadratic interactions.

Conversely, the last two sub-layers propagate information in the reverse direction, *from* the updated constraint nodes *to* the variable nodes. By again routing messages through the quadratic nodes, the network learns the relationship between constraints and quadratic terms, using this to refine the variable node representations. This bidirectional, structured message-passing is crucial for capturing the complex dependencies when representing QCQPs.

- **Readout Layer.** The readout layer maps the final node features from the T-th message-passing layer to an output $y \in \mathbb{R}^s$ using a learnable function f_{out}. We consider two output types:
  - **Graph-level output** ($s = 1$): The output is a scalar representing the entire graph. We compute it as:

$$y = f_{\text{out}}\left(a_1, a_2, a_3\right),$$

    where $a_1 = \sum_{u\in V_1} h^{T,u}$, $a_2 = \sum_{v\in V_2} h^{T,v}$, and $a_3 = \sum_{c\in V_3} h^{T,c}$ are the aggregated features from variable, quadratic, and constraint nodes, respectively.
  - **Node-level output** ($s = n$): The output is a vector where each component corresponds to a variable node. For each node $u_j \in V_1$:

$$y_j = f_{\text{out}}\left(h^{T,u_j}, a_{1,u_j}, a_2, a_3\right), \quad \forall j \in [n]$$

    where $a_{1,u_j} = \sum_{u\in V_1\setminus\{u_j\}} h^{T,u}$ aggregates features from all other variable nodes, providing contextual information about the graph excluding node u_j itself.

Definition 2 (Spaces of GNNs). Let $\mathcal{F}_{\text{QCQP}}(\mathbb{R}^s)$ denote the collection of all tripartite MP-GNNs, parameterized by continuous embedding functions $g_l^0, l \in \{1,2,3\}$, continuous hidden functions in the message passing layers g_l^t with $l \in \{1,2,3,4\}$, h_l^t with $l \in \{1,2,3,4,5,6\}$, and the continuous readout function f_{out}. Specifically, for a given problem size (m,n) of QCQP, there exists a subset of GNNs in $\mathcal{F}_{\text{QCQP}}(\mathbb{R}^s)$ that maps the input space $\mathcal{G}_{\text{QCQP}}^{m,n}$ to the output space $\mathbb{R}^s$. This subset of GNNs are denoted by $\mathcal{F}_{\text{QCQP}}^{m,n}(\mathbb{R}^s)$.

We define the following target functions, characterizing some key properties on learning an end-to-end network to predict the optimal solutions of convex QCQPs:

Definition 3 (Target mappings). Let G_{QCQP} be a tripartite graph representation of a QCQP problem. We define the following target mappings.

- *Feasibility mapping*: We define $\Phi_{\text{feas}}(G_{\text{QCQP}}) = 1$ if the QCQP problem is feasible and $\Phi_{\text{feas}}(G_{\text{QCQP}}) = 0$ otherwise.
- *Boundedness mapping*: for a feasible QCQP problem, we define $\Phi_{\text{bound}}(G_{\text{QCQP}}) = 1$ if the QCQP problem is bounded and $\Phi_{\text{bound}}(G_{\text{QCQP}}) = 0$ otherwise.
- *Optimal value mapping*: for a feasible and bounded QCQP problem, we set $\Phi_{\text{opt}}(G_{\text{QCQP}})$ to be its optimal objective value.
- *Optimal solution mapping*: for a feasible, bounded QCQP problem, there must exist at least an optimal solution, but the optimal solution might not be unique. However, if the QCQP is convex, there exists a unique optimal solution x^* with the smallest ℓ_2-norm among all optimal solutions. Therefore, for a *convex* QCQP we define the optimal solution mapping to be $\Phi_{\text{sol}}(G_{\text{QCQP}}) = x^*$. Since the optimal solution with the smallest ℓ_2-norm may not be unique for non-convex QCQPs, we do not define its optimal solution mapping.[‡]

3.2 Universal approximation for convex QCQPs

Our theoretical analysis rests on the following key lemma, which establishes that the tripartite WL-test can transfer solutions between equivalent QCQP instances.

Lemma 1. Let $\mathcal{I}, \bar{\mathcal{I}}$ (with given sizes m, n, encoded by $G, \bar{G} \in \mathcal{G}_{\text{QCQP}}^{m,n}$) be two QCQP instances. If the tripartite WL-test cannot separate G from $\bar{G}$, then for any feasible solution x of $\mathcal{I}$, there exists a feasible solution $\bar{x}$ for $\bar{\mathcal{I}}$ such that:

(i) $\frac{1}{2}\bar{x}^\top \bar{Q}\bar{x} + \bar{p}^\top \bar{x} \le \frac{1}{2}x^\top Q x + p^\top x$;

(ii) $\|\bar{x}\|_2 \le \|x\|_2$.

Lemma 1 resolves a fundamental challenge. In LPs/LCQPs, solution transfer between equivalent instances can be achieved through straightforward averaging of variable values within node color classes. For QCQPs, this direct approach fails because the average of products does not equal the product of averages. Our core contribution is proving that the *stable colorings* generated by the tripartite WL-test provide sufficient structural information to overcome this limitation. The solution $\bar{x}$ is constructed by averaging the values of the original solution x across all variables belonging to the same color class in G. While this averaging does not preserve individual quadratic terms, we prove that the *collective quadratic forms*–the weighted sums of quadratic terms within each constraint–are appropriately controlled due to the color stability of the graph representation. This ensures that feasibility is maintained in $\bar{\mathcal{I}}$ and the objective value is non-increasing. This lemma establishes the fundamental separation power of the tripartite WL-test: it can distinguish between QCQP instances that have different optimal values or solution structures.

Building on this foundation, we demonstrate that for convex QCQPs, any target function in Definition 3 can be universally approximated by MP-GNNs. Formally, we have the following theorem.

[‡]In fact, Section 3.3 shows that there exists a pair of non-convex QCQPs that cannot be distinguished by any MP-GNNs based on tripartite graph representation. Thus, even if an optimal solution mapping for non-convex QCQPs is defined, MP-GNNs cannot universally approximate it.

Theorem 1. For any probability measure $\mathbb{P}$ on the space of convex QCQPs $\mathcal{G}^{m,n}_{\text{QCQP}}$ and any $\delta, \varepsilon > 0$, there exists $F \in \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^s)$ such that for any target mapping $\Phi : \mathcal{G}^{m,n}_{\text{QCQP}} \to \mathbb{R}^s$ defined in Definition 3, we have

$$\mathbb{P}\left\{||F(G_{\text{QCQP}}) - \Phi(G_{\text{QCQP}})|| > \delta\right\} < \varepsilon. \tag{3.1}$$

Theorem 1 highlights that sufficiently expressive GNNs can predict the feasibility, boundedness, optimal value, and optimal solution for convex QCQP problems with an arbitrarily small error. The proof of Theorem 1 is provided in Appendix A. The convexity requirement is essential for our theoretical results. Convexity ensures that the averaged solution $\bar{x}$ constructed in Lemma 1 not only remains feasible but also satisfies the critical objective value bound in condition (i). This property enables the translation of the WL-test's separation power into the GNN's approximation capability for the target functions Φ.

3.3 MP-GNNs can not represent general non-convex QCQPs

In contrast to convex QCQPs, MP-GNNs based on tripartite graph representation do not possess universal representation power for non-convex QCQPs. Formally, we have the following propositions.

Proposition 1. There exists non-convex QCQP instances $\mathcal{I}, \bar{\mathcal{I}}$ encoded by tripartite graph representation $G, \bar{G}$ respectively, such that $\Phi(G)_{\text{feas}} \neq \Phi_{\text{feas}}(\bar{G})$, but any GNN $F \in \mathcal{F}_{\text{QCQP}}(\mathbb{R})$ gives $F(G) = F(\bar{G})$.

Proposition 2. There exists non-convex QCQP instances $\mathcal{I}, \bar{\mathcal{I}}$ encoded by tripartite graph representation $G, \bar{G}$ respectively, such that

(i) $\Phi(G)_{\text{opt}} \neq \Phi_{\text{opt}}(\bar{G})$;

(ii) the optimal solution sets of $\mathcal{I}$ and $\bar{\mathcal{I}}$ do not intersect;

(iii) any GNN $F \in \mathcal{F}_{\text{QCQP}}(\mathbb{R})$ gives $F(G) = F(\bar{G})$.

Proposition 1 implies that GNNs cannot universally predict the feasibility of non-convex QCQPs. Proposition 2 implies that GNNs can neither universally predict the optimal value nor the optimal solution of non-convex QCQPs. We prove both propositions by constructing counter-examples. Below we present the counter-example for Proposition 2. We defer the formal proof of both propositions to Appendix C.

Consider the following pair of non-convex QCQPs:

$$\begin{aligned} \min_{x \in \mathbb{R}^6} \quad & x_1x_2 + x_2x_3 + x_3x_1 + x_4x_5 + x_5x_6 + x_6x_4 \\ \text{s.t.} \quad & \sum_i x_i^2 \leq 1 \end{aligned} \tag{3.2}$$

$$\begin{aligned} \min_{x \in \mathbb{R}^6} \quad & x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_1 \\ \text{s.t.} \quad & \sum_i x_i^2 \leq 1 \end{aligned} \tag{3.3}$$

For the former, the optimal objective value is $\Phi_{\text{obj}} = -\frac{1}{2}$, and all optimal solutions are given by

$$\left\{ x \in \mathbb{R}^6 : x_1 + x_2 + x_3 = 0, x_4 + x_5 + x_6 = 0, \sum_i x_i^2 = 1 \right\}.$$

For the latter, the optimal objective value $\Phi_{\text{obj}} = -1$, and all optimal solutions are given by

$$\left\{ x \in \mathbb{R}^6 : x_1 = x_3 = x_5 = -x_2 = -x_4 = -x_6 = \pm\frac{\sqrt{6}}{6} \right\}.$$

We see that the optimal values of Problem (3.2) and Problem (3.3) are different, and their optimal solution sets do not intersect. The tripartite graph representations of the two instances are illustrated in Figure 3. We

$$\begin{aligned} \min \quad & x_1x_2 + x_2x_3 + x_3x_1 + x_4x_5 + x_5x_6 + x_6x_4 \\ \text{s.t.} \quad & \sum_i x_i^2 \le 1 \end{aligned}$$

$$\begin{aligned} \min \quad & x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_1 \\ \text{s.t.} \quad & \sum_i x_i^2 \le 1 \end{aligned}$$

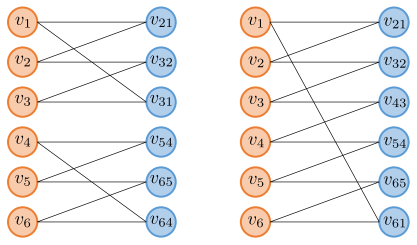


Figure 3: Left: two QCQP instances for proving Prop. 1. Right: Parts of the corresponding tripartite graph representations to show the difference.

will further demonstrate in appendix C that any GNNs on the two tripartite graphs gives the same output. Thus, Problem (3.2) and Problem (3.3) serve as a valid counter-example for proving Proposition 2.

We remark that Propositions 1 and 2 demonstrate limitations specific to our proposed tripartite MP-GNN architecture. They do not rule out the potential for other, more powerful graph representations or network architectures to achieve universal approximation for non-convex QCQPs.

4 Computational Experiments

In this section, we present empirical experiments to validate the proposed theoretical results. The corresponding source code is available at https://github.com/NetSysOpt/L2QP.

4.1 Learning tasks

In Theorem 1, we established that there exists a function $F \in \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^s)$ capable of approximating the target mapping Φ with an arbitrarily small error. To empirically confirm this claim, we design three supervised learning tasks to find such functions F_{feas}, F_{obj} and F_{sol}, which are responsible for predicting feasibility, objective values, and optimal solutions, respectively. For each task, let $\{(G_i, y_i)\}_{i=1}^N$ be a given dataset, where G_i represents a QCQP instance and y_i denotes its corresponding label. The function family $\mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^s)$ is constructed using the tripartite MP-GNNs as defined in Definition 2. With all these ingredients ready, the learned function is obtained by $F = \arg\min_{f \in \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^s)} \frac{1}{N} \sum_{i=1}^N L\left(f(G_i), y_i\right)$, where $L(\cdot,\cdot)$ is the loss function. Specifically, we use mean squared error for predicting objective values and optimal solutions, while binary cross-entropy loss is employed for predicting feasibility. We chose cross-entropy loss for this binary classification task as it is more suitable than a regression-based approach that would be needed if we were to predict numerical degrees of constraint violation instead of a final feasibility state.

4.2 Data generation

To facilitate the supervised learning approach described above, we generate three datasets of convex QCQPs by randomly sampling coefficients from normal distributions. These datasets consist of a general QCQP (GP), a constrained least squares problem (CLS), and a trust-region subproblem (TRS). The formulation and generation strategy for each dataset are outlined below.

GP: A general QCQP formulation is given in (2.1). All coefficients of Q, p, Q^i, p^i, and b^i in the objective and constraints are independently sampled from the normal distribution $\mathcal{N}(0,1)$. The lower and upper bounds x^L and x^U are set to be $\mathbf{0}$ and $\mathbf{1}$, respectively. The number of variables and constraints for instances in this dataset are set to 30 and 5, respectively. We generate a small dataset for general QCQPs because larger-scale instances with more constraints tend to be predominately infeasible.

CLS: Constrained least squares problems with quadratic constraints are common in various applications. We examine the following CLS problem:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|Ax-b\|^2 \\ \text{s.t.} \quad & x^\top Qx \le c \end{aligned} \tag{4.1}$$

The elements in A, b and Q are sampled from the standard normal distribution $\mathcal{N}(0,1)$, while elements in c are sampled from $\mathcal{N}(1,1)$. For this dataset, the number of variables is set to 500, and the sparsity (the proportion of zero elements) of Q is set to 0.95.

TRS: The trust-region subproblem is another form of convex QCQP, defined by the following formulation:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & x^\top Qx + 2c^\top x \\ \text{s.t.} \quad & \|x\|^2 \le \Delta^2 \end{aligned} \tag{4.2}$$

The coefficients in Q and c are independently sampled from the standard normal distribution $\mathcal{N}(0,1)$ and Δ is sampled from $\mathcal{N}(1,1)$. Like the CLS dataset, the number of variables is set to 500, and the sparsity of Q is set to 0.95.

To ensure that the generated instances remain convex, we adjust all sampled matrices Q and Q_is corresponding to the quadratic terms in both the objective function and the constraints. Specifically, each matrix is modified by replacing it with $Q-\alpha I$, where $\alpha<0$ is the minimal eigenvalue of Q. This adjustment ensures that the matrices are positive semi-definite, thus guaranteeing the convexity of the corresponding QCQP instances.

For each dataset, we generate 1,000 instances for training and 300 instances for validation. All instances are solved using the IPOPT solver (Wächter & Biegler, 2006). The resulting feasibility, objective values, and optimal solutions are collected as labels.

4.3 GNN architecture and training settings

For the GNN described in Section 3.1, there are three classes of functions $\{g_1^t,\dots,g_4^t\}_{t=1}^T$, $\{h_1^t,\dots,h_6^t\}_{t=1}^T$ and R remain unspecified. The first class, $\{g_1^t,\dots,g_4^t\}_{t=1}^T$, are two-layer MLPs with layer widths of $[d,d]$, and ReLU as activations, where the inputs of each function are concatenated together. The second class, $\{h_1^t,\dots,h_6^t\}_{t=1}^T$, are linear transformations with output dimension d followed by ReLU activations. The last one, R, is also a two-layer MLP with ReLU activation, with widths of $[d,1]$ for predicting feasibility and objective values, and $[d,n]$ for predicting solutions. The hyper-parameter T is set to 2. For training, we utilized the Adam optimizer with a learning rate of 0.0001 and a batch size of 16.

4.4 Main results

In this section, we present the main results for tasks with different learning targets. For the GP dataset, we perform tasks to predict the feasibility, objective value, and optimal solution. However, the feasibility task is omitted for the CLS and TRS datasets, as their sampled instances are predominantly feasible.

Training loss vs. numbers of parameters. The results from Table 1 show the training loss for models with different numbers of parameters. Generally, the losses are small across all models, validating the claim in Theorem 3.1. Moreover, we observe a consistent trend: as the number of parameters increases, the training loss decreases.

For the GP dataset, the feasibility, objective, and solution losses all decrease as the number of parameters increases, with the loss for the largest model (6.7M parameters) being particularly small, especially for the objective and feasibility targets. For the CLS dataset, similar trends are observed, with the loss for the objective function decreasing substantially from 0.0994 (15K parameters) to 0.0016 (6.7M parameters). The solution loss in CLS remains relatively stable but still decreases as the number of parameters increases. Finally, in the TRS dataset, both the objective and solution losses follow a similar pattern, with the objective loss improving significantly from 2.1098 (15K parameters) to 0.0171 (6.7M parameters).

Table 1: Training loss vs. numbers of parameters.

Dataset	Target	# Parameters					
		15K	21K	42K	126K	1.7M	6.7M
GP	feasibility	0.2114	0.0992	0.0883	0.0779	0.0254	0.0044
	objective	0.0853	0.0293	0.0218	0.0185	0.0110	0.0034
	solution	0.0604	0.0419	0.0413	0.0407	0.0397	0.0350
CLS	objective	0.0994	0.0196	0.0135	0.0096	0.0020	0.0016
	solution	0.2333	0.1454	0.0175	0.0173	0.0171	0.0169
TRS	objective	2.1098	0.1979	0.1089	0.0996	0.0265	0.0171
	solution	1.2004	0.4294	0.4035	0.4022	0.4017	0.4010

Validation loss vs. number of samples. The validation loss results presented in Table 2 indicate that the validation loss remains small across all configurations, highlighting the generalization capability of the models. Additionally, as the number of training samples increases, the validation loss decreases, reflecting the benefit of having more data for model training.

Table 2: Validation loss vs. numbers of training samples.

Dataset	Target	# Samples				
		100	300	500	700	1,000
GP	feasibility	0.5857	0.3167	0.1940	0.1885	0.1004
	objective	0.1067	0.0577	0.0514	0.0421	0.0296
	solution	0.1149	0.0439	0.0438	0.0434	0.0427
CLS	objective	0.0136	0.0121	0.0107	0.0086	0.0049
	solution	0.0181	0.0179	0.0178	0.0177	0.0177
TRS	objective	0.2131	0.0933	0.0834	0.0792	0.0596
	solution	0.4081	0.4055	0.4054	0.4054	0.4054

For the GP dataset, the validation loss for feasibility, objective, and solution all decrease as the number of samples increases. Notably, the solution loss stabilizes after 500 samples, while the objective and feasibility losses continue to improve as the number of samples grows, with the best performance observed at 1,000 samples. The CLS dataset shows a clear reduction in the objective loss as the number of samples increases, with the loss dropping from 0.0136 for 100 samples to 0.0049 for 1,000 samples. The solution loss in CLS remains relatively stable across different sample sizes. Finally, for the TRS dataset, the objective loss improves from 0.2131 at 100 samples to 0.0596 at 1,000 samples, while the solution loss remains nearly constant across all sample sizes.

In conclusion, both the training and validation loss results highlight the effectiveness of the models, showing that increasing the number of parameters and training samples leads to improved performance in terms of both training and generalization.

4.5 Additional comparative results

Runtime comparison. Table 3 reports the average solving time (in seconds) and standard deviation across the validation instances for several QCQP solvers—IPOPT (Wächter & Biegler, 2006), Gurobi (Gurobi Optimization, LLC, 2025), MOSEK (ApS, 2025), SCS (O'Donoghue et al., 2016), and ECOS (Domahidi et al., 2013)—together with the inference time of our GNN. The results demonstrate that our GNN-based approach delivers a substantial speedup, often by orders of magnitude, compared to all baseline solvers. However, it is important to note that the accuracy of these baseline solvers ($\leq 10^{-6}$) significantly surpasses that of the

GNN, as reported in Section 4.4 (with a minimal level of 10^{-3}), indicating a trade-off between speed and precision.

Table 3: Comparison of runtime (in seconds) with solvers having a tolerance of 10^{-6}.

Method	GP	GLS	TRS
IPOPT	0.0374($\pm$ 0.0044)	0.0602($\pm$ 0.0115)	0.0670($\pm$ 0.0158)
GUROBI	0.0054($\pm$ 0.0004)	0.1091($\pm$ 0.0138)	1.8921($\pm$ 0.2751)
MOSEK	0.0300($\pm$ 0.0033)	0.4014($\pm$ 0.0274)	0.2264($\pm$ 0.0079)
SCS	0.0311($\pm$ 0.0058)	0.6799($\pm$ 1.0300)	0.1466($\pm$ 0.8167)
ECOS	0.0314($\pm$ 0.0040)	4.4056($\pm$ 0.4668)	1.6713($\pm$ 0.5785)
GNN	0.0024($\pm$ 0.0032)	0.0022($\pm$ 0.0030)	0.0023($\pm$ 0.0031)

Impact of sparsity. We conducted an ablation study on datasets CLS and TRS by varying sparsity (fraction of zero coefficients) from 0.95 to 0.8. Table 4 reports the training losses. As sparsity decreases,

Table 4: Training loss on different levels of sparsity.

Dataset	Target	Sparsity			
		0.95	0.9	0.85	0.8
CLS	objective	0.0020	0.0036	0.0058	0.0096
	solution	0.0171	0.0217	0.0279	0.0323
TRS	objective	0.0265	0.0497	0.1084	0.1844
	solution	0.4017	0.5338	0.7415	0.9042

losses increase consistently, reflecting the added difficulty of denser problems with more complex interactions. Our focus on highly sparse cases (e.g., 0.95) is motivated by their prevalence in practice—for example, over 35% of QPLib (Furini et al., 2019) instances exhibit sparsity above 0.95.

In addition to comparisons on runtime and sparsity, we also provide evaluations on a real-world dataset, QPLIB, and on a task of predicting boundedness, which can be found in Appendix D.

4.6 Discussions

The computational efficiency of our tripartite GNN stems from two key design choices. First, the graph structure naturally exploits problem sparsity, as edges correspond directly to non-zero parameters, ensuring high efficiency for real-world, sparse instances. Second, the architecture uses simple message-passing layers where the split-layer design organizes computations without increasing the total number of matrix operations, incurring minimal overhead. We remark that our universal approximation theorem establishes expressive power but does not quantify the parameters required for a given approximation error; deriving such bounds remains an important direction for future work.

5 Conclusions

This paper introduces a new tripartite graph representation specifically designed for QCQPs. By leveraging the capabilities of MP-GNNs, this approach shows theoretical promise in predicting key properties of QCQPs with arbitrary desired accuracy, including feasibility, boundedness, optimal values, and solutions. Initial numerical experiments validate the effectiveness of our framework. This work advances learning-to-optimize by extending GNNs to QCQP problems, which have been challenging for traditional graph-based L2O methods. Our findings may inspire future research into more specialized GNN architectures and principled approaches to controlling GNN sizes for practical QCQP applications, going beyond the basic GCN structure used here.

Acknowledgments

This work was supported by the National Key R&D Program of China under grant 2022YFA1003900. Qian Chen and Akang Wang also acknowledge support from National Natural Science Foundation of China (Grant No. 12301416), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2024A1515010306), and Shenzhen Science and Technology Program (Grant No. JCYJ20250604191330040).

References

MOSEK ApS. *The MOSEK Python Fusion API manual. Version 11.0.*, 2025. URL `https://docs.mosek.com/latest/pythonfusion/index.html`.

Waiss Azizian and Marc Lelarge. Expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations*, 2020.

Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications.* SIAM, 2001.

Dimitris Bertsimas and Bartolomeo Stellato. Online mixed-integer optimization in milliseconds. *INFORMS Journal on Computing*, 34(4):2229–2248, 2022.

Dan Bienstock, Mauro Escobar, Claudio Gentile, and Leo Liberti. Mathematical programming formulations for the alternating current optimal power flow problem. *4OR*, 18(3):249–292, 2020.

Pierre Bonami, Andrea Lodi, and Giulia Zarpellon. Learning a classification of mixed-integer quadratic programming problems. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15*, pp. 595–604. Springer, 2018.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

Ziang Chen, Jialin Liu, Xinshang Wang, Jianfeng Lu, and Wotao Yin. On representing linear programs by graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023a.

Ziang Chen, Jialin Liu, Xinshang Wang, Jianfeng Lu, and Wotao Yin. On representing mixed-integer linear programs by graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023b.

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.

Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *2013 European control conference (ECC)*, pp. 3071–3076. IEEE, 2013.

Fabio Furini, Emiliano Traversi, Pietro Belotti, Antonio Frangioni, Ambros Gleixner, Nick Gould, Leo Liberti, Andrea Lodi, Ruth Misener, Hans Mittelmann, et al. Qplib: a library of quadratic programming instances. *Mathematical Programming Computation*, 11:237–265, 2019.

Quankai Gao, Fudong Wang, Nan Xue, Jin-Gang Yu, and Gui-Song Xia. Deep graph matching under quadratic constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5069–5078, 2021.

Philip E Gill, Walter Murray, and Margaret H Wright. *Practical optimization.* SIAM, 2019.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2025. URL `https://www.gurobi.com`.

Jeffrey Ichnowski, Paras Jain, Bartolomeo Stellato, Goran Banjac, Michael Luo, Francesco Borrelli, Joseph E Gonzalez, Ion Stoica, and Ken Goldberg. Accelerating quadratic optimization with reinforcement learning. *Advances in Neural Information Processing Systems*, 34:21043–21055, 2021.

Haewon Jung, Junyoung Park, and Jinkyoo Park. Learning context-aware adaptive solvers to accelerate quadratic programming. *arXiv preprint arXiv:2211.12443*, 2022.

Pan Li and Jure Leskovec. The expressive power of graph neural networks. *Graph Neural Networks: Foundations, Frontiers, and Applications*, pp. 63–98, 2022.

Zhi-Quan Luo, Wing-Kin Ma, Anthony Man-Cho So, Yinyu Ye, and Shuzhong Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20–34, 2010.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, June 2016. URL `http://stanford.edu/~boyd/papers/scs.html`.

Rajiv Sambharya, Georgina Hall, Brandon Amos, and Bartolomeo Stellato. End-to-end learning to warm-start for real-time quadratic optimization. In *Learning for Dynamics and Control Conference*, pp. 220–234. PMLR, 2023.

Haoru Tan, Chuang Wang, Sitong Wu, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Ensemble quadratic assignment network for graph matching. *International Journal of Computer Vision*, pp. 1–23, 2024.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.

Tao Wang, He Liu, Yidong Li, Yi Jin, Xiaohui Hou, and Haibin Ling. Learning combinatorial solver for graph matching. in 2020 ieee. In *CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 13–19, 2020.

Jinxin Xiong, Xi Gao, Linxin Yang, Jiang Xue, Xiaodong Luo, and Akang Wang. Solving quadratic programs via deep unrolled douglas-rachford splitting. *Transactions on Machine Learning Research*.

Zhixiao Xiong, Fangyu Zong, Huigen Ye, and Hua Xu. Neuralqp: A general hypergraph-based optimization framework for large-scale qcqps. *arXiv preprint arXiv:2410.03720*, 2024.

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.

Bingxu Zhang, Changjun Fan, Shixuan Liu, Kuihua Huang, Xiang Zhao, Jincai Huang, and Zhong Liu. The expressive power of graph neural networks: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

A Detailed proof of main theorem

A.1 Sketch of the proof

We provide a brief outline of this proof:

(i) **Separation Power of WL-Test:** We first establish that the WL-test has sufficient separation power on the defined target functions.

(ii) **Connection to tripartite MP-GNNs:** We then demonstrate the relationship between the separation power of tripartite MP-GNNs and that of the Tripartite WL-tests, showing that the GNNs can separate our target functions. This result, combined with the generalized Weierstrass theorem, leads to our approximation power conclusions.

(iii) **Universal Approximation:** Assuming the target functions are continuous and have compact support, we prove universal approximation. In this step, we also specify the problem size and apply the Generalized Weierstrass Theorem (Theorem 22 of Azizian & Lelarge (2020)).

(iv) **Addressing Discontinuities:** Since the target functions are neither continuous nor compactly supported, particularly at the boundary of the convex QCQPs universe $\mathcal{G}_{\text{QCQP}}^{m,n}$, we construct a continuous approximation of the target function to apply universal approximation, ensuring convergence in measure.

A.2 WL-test on tripartite graph representation

Here we describe our Tripartite WL-test, which is the WL-test counterpart of the tripartite MP-GNNs:

- **Embedding.** Initial colors $C^{0,u}$, $C^{0,v}$, and $C^{0,c}$ are assigned based on their corresponding features and node types (e.g., from V_1, V_2, or V_3):
  - $C^{0,u} \leftarrow \text{HASH}_1(f(u))$ for $u \in V_1$,
  - $C^{0,v} \leftarrow \text{HASH}_2(f(v))$ for $v \in V_2$,
  - $C^{0,c} \leftarrow \text{HASH}_3(f(c))$ for $c \in V_3$.

  Here, we refer to the color of a node after the t-th message-passing layer as $C^{t,\cdot}$.

- **Update quadratic nodes via variable nodes** ($V_1 \to V_2$):

$$\bar{C}^{t,v} \leftarrow \text{HASH}\left(C^{t,v}, \sum_{u\in V_1} w_{u,v}\text{HASH}(C^{t,u})\right), \forall v \in V_2$$

- **Update constraint nodes via variable and quadratic nodes** ($V_1, V_2 \to V_3$):

$$C^{t+1,c} \leftarrow \text{HASH}\left(C^{t,c}, \sum_{u\in V_1} w_{u,c}\text{HASH}(C^{t,u}), \sum_{v\in V_2} w_{v,c}\text{HASH}(\bar{C}^{t,v})\right), \forall c \in V_3$$

- **Update quadratic nodes again via constraint nodes** ($V_3 \to V_2$):

$$C^{t+1,v} \leftarrow \text{HASH}\left(\bar{C}^{t,v}, \sum_{c\in V_3} w_{c,v}\text{HASH}(C^{t+1,c})\right), \forall v \in V_2$$

- **Update variable nodes via constraint and quadratic nodes** ($V_3, V_2 \to V_1$):

$$C^{t+1,u} \leftarrow \text{HASH}\left(C^{t,u}, \sum_{c\in V_3} w_{c,u}\text{HASH}(C^{t+1,c}), \sum_{v\in V_2} w_{v,u}\text{HASH}(C^{t+1,v})\right), \forall u \in V_1$$

- **Termination and Readout.** Once a termination condition is met, we return the color collection $(C^{T,u})_{u\in V_1}, (C^{T,v})_{v\in V_2}, (C^{T,c})_{c\in V_3}$.[§]

[§]Multiple occurrences of members are counted instead of rejected.

- All hash functions are real-valued and assumed to be collision-free.

In this paper, we terminate the Tripartite WL-test only when the algorithm stabilizes[¶], i.e., when the number of distinct colors no longer changes in an iteration (after all four color updates). Despite not imposing a forced iteration limit, the WL-test is guaranteed to terminate in a finite number of iterations, denoted by T:
Proposition 3 (Tripartite WL-test terminates in finite iterations). The Tripartite WL-test stabilizes in a finite number of iterations.

Proof. It is straightforward to observe from the formulation that if two nodes have different colors, they will continue to have different colors after an (sub-)iteration. Therefore, the number of iterations required for stabilization is capped by the number of distinct nodes, which is finite. □

We say that the Tripartite WL-test *separates* two graphs if the resulting collection of colors differs between the two graphs. We claim that the Tripartite WL-test has the same separation power as its network counterpart, specifically the tripartite MP-GNNs:
Proposition 4 (Tripartite MP-GNNs have equal separation power as the Tripartite WL-Test). Given two instances $\mathcal{I}$ and $\bar{\mathcal{I}}$ (correspondingly encoded by graphs G and $\bar{G}$), the following holds:

(i) For graph-level output cases, the two instances are separated by $\mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R})$, i.e.,

$$F(G) = F(\bar{G}), \forall F \in \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R})$$

if and only if the two instances are also separated by the Tripartite WL-test.

(ii) For node-level output cases, i.e., $\mathbb{R}^s = \mathbb{R}^n$, the two instances are separated by $\mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R})$, i.e.,

$$F(G) = F(\bar{G}), \forall F \in \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^n)$$

if and only if the two instances are separated by the Tripartite WL-test, and additionally, the variables are correspondingly indexed. Specifically, $C^{T,u_j} = C^{T,\bar{u}_j}$ must hold for all $j \in [n]$.

For the detailed proof of this proposition, see Appendix B.3.

A.3 Proof of main theorem

Now we can prove the main theorem. First, we the following proposition.
Proposition 5. Let $\mathcal{I}, \bar{\mathcal{I}}$ (encoded by $G, \bar{G} \in \mathcal{G}^{m,n}_{\text{QCQP}}$) be two QCQP instances. If the tripartite WL-test fails to separate the two instances, then the following holds:

(i) If one is feasible, the other is also feasible, i.e., $\Phi_{\text{feas}}(G) = \Phi_{\text{feas}}(\bar{G})$.

(ii) Assume both instances are feasible. If one is unbounded, the other is also unbounded.

(iii) Assume both instances are bounded. Then they have equal optimal values, i.e., $\Phi_{\text{obj}}(G) = \Phi_{\text{obj}}(\bar{G})$.

(iv) Assume both instances are bounded and that the variables and constraints are indexed such that $C^{T,u_j} = C^{T,\bar{u}_j}$. Then they have the same optimal solution, with the least ℓ_2-norm, i.e., $\Phi_{\text{sol}}(G) = \Phi_{\text{sol}}(\bar{G})$.

Proof. **Passing feasibility**. Assume that $\mathcal{I}$ is feasible, and let x be a feasible solution. By Lemma 1, we obtain another solution $\bar{x}$ for instance $\bar{\mathcal{I}}$, which implies the feasibility of $\bar{\mathcal{I}}$. By switching the roles of $\mathcal{I}$ and $\bar{\mathcal{I}}$, we prove the reverse claim.

[¶]For simplicity, we exclude the final iteration showing that the algorithm has stabilized and return the last iteration in which stabilization occurred.

Passing unboundedness. Assume that $\mathcal{I}$ is unbounded, i.e., for any $M > 0$, there exists a solution x_M such that the objective $f(x) \leq -M$. For each x_M, we can construct a solution $\bar{x}_M$ for $\bar{\mathcal{I}}$ such that the objective $\bar{f}(\bar{x}_M) \leq f(x_M) \leq -M$, implying that $\bar{\mathcal{I}}$ is also unbounded. Again, by switching the roles of $\mathcal{I}$ and $\bar{\mathcal{I}}$, we prove the reverse claim.

Passing optimal values. Assume that $\mathcal{I}$ is feasible and bounded, and let x be its optimal solution. By Lemma 1, we construct a solution $\bar{x}$ for $\bar{\mathcal{I}}$ such that:

$$\bar{f}(\bar{x}) \leq f(x) = \Phi_{\text{obj}}(G)$$

implying that $\Phi_{\text{obj}}(\bar{G}) \leq \Phi_{\text{obj}}(G)$. Similarly, we can show that $\Phi_{\text{obj}}(G) \leq \Phi_{\text{obj}}(\bar{G})$, and thus $\Phi_{\text{obj}}(\bar{G}) = \Phi_{\text{obj}}(G)$.

Passing optimal solutions. To prove the last claim, we need the construction of $\bar{x}$ from the detailed proof of Lemma 1 (see Appendix B.2). Assume that $\mathcal{I}$ is feasible and bounded, and let x be its optimal solution (with the least ℓ_2-norm). By Lemma 1, we construct y for $\bar{\mathcal{I}}$ and z for $\mathcal{I}$ by switching the roles of $\mathcal{I}$ and $\bar{\mathcal{I}}$.

We have $f(z) \leq \bar{f}(y) \leq f(x)$ and $\|z\| \leq \|y\| \leq \|x\|$, which implies that z is not worse than the given optimal solution x, and thus $z = x$. By the construction of the averaged solution (and the assumption $C^{T,u_j} = C^{T,\bar{u}_j}$), we have $y = z$. Combining the two equalities, we conclude that $x = y$.

Let $\bar{x}$ be the optimal solution of $\bar{G}$, and we have $\|\bar{x}\| \leq \|y\| = \|x\|$. By switching the roles of $\mathcal{I}$ and $\bar{\mathcal{I}}$, we obtain $\|x\| \leq \|\bar{x}\|$, and thus $\|\bar{x}\| = \|x\|$. Similarly, we have $\bar{f}(\bar{x}) \leq \bar{f}(y) \leq f(x)$, and by switching the roles, $\bar{f}(\bar{x}) = f(x)$.

Since $\|y\| = \|x\| = \|\bar{x}\|$ and $\bar{f}(y) = f(x) = f(\bar{x})$, by uniqueness, we conclude that $y = \bar{x}$, proving the fourth claim. □

The next step is to extend this separation power to approximation power, which leads to our main theorem. We utilize the generalized Weierstrass-Stone theorem (Theorem 22 and Lemma 36 of Azizian & Lelarge (2020)) and Lusin's theorem.

By applying the generalized Weierstrass-Stone theorem, we establish the following proposition, which demonstrates the approximation power on equivariant functions with compact support.

Proposition 6 (Uniform Approximation on Continuous Equivariant Functions with Compact Support). Let $\Phi_c : \mathcal{G}_{\text{c}}^{m,n} \to \mathbb{R}^s$ be a general continuous target function defined on a compact subset $\mathcal{G}_c \subseteq \mathcal{G}_{\text{QCQP}}^{m,n}$, such that:

- If $s = 1$, the output remains unchanged if the input graph is re-indexed.
- If $s = n$, the output re-indexes accordingly if the input graph is re-indexed.

If the following holds:

$$\left(F(G) = F(\bar{G}), \forall F \in \mathcal{F}_{\text{QCQP}}^{m,n}(\mathbb{R}^s) \Rightarrow \Phi(G) = \Phi(\bar{G})\right), \forall G, \bar{G} \in \mathcal{G}_{\text{c}}^{m,n} \tag{A.1}$$

i.e., the family $\mathcal{F}_{\text{QCQP}}^{m,n}(\mathbb{R}^s)$ separates the target function Φ, then for any $\delta > 0$, there exists a function $F_\delta \in \mathcal{F}_{\text{QCQP}}^{m,n}(\mathbb{R}^s)$ such that:

$$\|F_\delta(\mathcal{G}) - \Phi(G)\| < \delta \tag{A.2}$$

For the detailed proof, see Appendix B.4.

However, the requirement for the target function to apply the proposition is too strong. In fact, all target functions defined in 3 are non-continuous and not defined on a compact subset, although equivariance naturally holds. Therefore, we seek a continuous approximation with compact support that can be uniformly approximated. By applying Lusin's theorem, we construct the following continuous approximation:

Proposition 7 (Continuous Approximation with Compact Support). Let $\Phi : \mathcal{G}_{\text{QCQP}}^{m,n} \to \mathbb{R}^s$ be a general target function that is measurable under the probability measure $\mathbb{P}$. For any $\varepsilon > 0$, there exists a compact subset $\mathcal{G}_{\text{c}}^{m,n} \subseteq \mathcal{G}_{\text{QCQP}}^{m,n}$, such that $\mathbb{P}\{G \in \mathcal{G}_{\text{c}}^{m,n}\} > 1 - \varepsilon$, and $\Phi|_{\mathcal{G}_{\text{c}}^{m,n}}$ is continuous.

By combining all the lemmas and propositions, we can now prove the main theorem.

Proof of Theorem 1. Let Φ be any target function defined in Definition 3.

By Proposition 7, Φ is continuous on a compact subset $\mathcal{G}_{\mathrm{c}}^{m,n} \subseteq \mathcal{G}_{\mathrm{QCQP}}^{m,n}$, with $\mathbb{P}(G \in \mathcal{G}_{\mathrm{c}}^{m,n}) \geq 1 - \frac{\varepsilon}{|\Sigma|}$.

We construct $\mathcal{G}_{\mathrm{c,eq}}^{m,n} = \cap_{(\sigma,\tau)\in\Sigma}(\sigma,\tau)(\mathcal{G}_{\mathrm{c}}^{m,n})$. This subset is continuous with compact support, ensuring that $\Phi|_{\mathcal{G}_{\mathrm{c,eq}}^{m,n}}$ remains an equivariant function, with the following measure control:

$$\mathbb{P}(G \in \mathcal{G}_{\mathrm{c,eq}}^{m,n}) > 1 - \varepsilon \tag{A.3}$$

Since by Proposition 5 and the fact that the Tripartite WL-test has equal separation power as the tripartite MP-GNNs, the target functions are equivariant and separated by $\mathcal{F}_{\mathrm{QCQP}}^{m,n}(\mathbb{R}^s)$. Thus, we may apply Proposition 6 and obtain $F \in \mathcal{F}_{\mathrm{QCQP}}^{m,n}(\mathbb{R}^s)$ such that:

$$\|F(G) - \Phi(G)\| < \delta, \forall G \in \mathcal{G}_{\mathrm{c,eq}}^{m,n}$$

This implies that $\mathbb{P}\{\|F(G) - \Phi(G)\| < \delta\} > 1 - \varepsilon$. ☐

B Proof of propositions in Section A

This section provides complete proofs of several propositions in Section A that were not immediately proven.

B.1 Equivariance

We begin by describing equivariance, a key tool used to capture the fact that the indexing of variables and constraints is irrelevant:

Definition 4. Given a function $f : X \to Y$, where X and Y are subsets of Euclidean spaces, and a group Σ that acts continuously on X and Y, the function f is called equivariant (with respect to the group Σ) if the following holds:

$$\sigma \circ f(x) = f \circ \sigma(x), \quad \forall x \in X, \sigma \in \Sigma$$

Since the indexing of variables and constraints does not affect the problem, we take $\Sigma = S_n \times S_m$, which represents all possible re-indexings of variables and constraints. When applied to both the input and output spaces, we re-index the variables, constraints, and possible solutions (in cases where the output is a solution $x \in \mathbb{R}^n$). Specifically, we have:

$$\begin{aligned}
\tilde{q}_{\pi(j),\pi(k)} &= q_{j,k} \\
\tilde{p}_{\pi(j)} &= p_j \\
\tilde{q}^{\tau(i)}_{\pi(j),\pi(k)} &= q^i_{j,k} \\
\tilde{p}^{\tau(i)}_{\pi(j)} &= p^i_j \\
\tilde{b}_{\tau(i)} &= b_i \\
\tilde{x}^{\mathrm{L}}_{\pi(j)} &= x^{\mathrm{L}}_j \\
\tilde{x}^{\mathrm{U}}_{\pi(j)} &= x^{\mathrm{U}}_j
\end{aligned}$$

where the tilde symbols $\tilde{Q}, \tilde{p}, \tilde{b}, \tilde{x}^{\mathrm{L}}, \tilde{x}^{\mathrm{U}}$ denote the re-indexed vectors and matrices.

For $\mathbb{R}^s = \mathbb{R}$, the action on the output space is the identity map: $(\pi,\tau)(\cdot) = \mathrm{id}$. For $\mathbb{R}^s = \mathbb{R}^n$, we correspondingly re-index the output, i.e., $(\pi,\tau)(y)_{\pi(j)} = y_j$.

We can also apply the permutations to:

- A point in $\mathbb{R}^n$ (such as a solution), by $(\pi,\tau)(x)_{\pi(j)} = x_j$.

- A subset of $\mathbb{R}^n$, by applying the permutation to each element in the subset, or to its indicator function by permuting the underlying set.

Equivariance allows us to show that the indices do not matter, while the inputs (in the form of coefficient tuples) necessarily carry these indices.

Remark: Given the group Σ and its action on both the input and output, all message-passing layers are automatically equivariant. Thus, requiring $F \in \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^s)$ to be equivariant is equivalent to requiring the readout layer R to be equivariant. This is why the readout function must take specific forms in the two cases. While the defined forms do not cover all possible equivariant readout functions, they are general enough to capture the separation power.

B.2 Proof of Lemma 1

For simplicity of proof, we extend the definitions of Φ_{obj} and Φ_{sol} to the entire space $\mathcal{G}^{m,n}_{\text{QCQP}}$ by assigning a default value of 0 (or $\mathbf{0}$, depending on the output dimension s) when the target function is not defined at a graph G. This occurs when the corresponding instance is either infeasible or unbounded, and the optimal value or optimal solution does not exist. By doing so, all target functions are defined on the same space $\mathcal{G}^{m,n}_{\text{QCQP}}$. Moreover, since we approximate feasibility and boundedness, we can distinguish whether the output is the default value or genuinely happens to be 0 (or $\mathbf{0}$).

Let $\mathcal{I}$ and $\bar{\mathcal{I}}$ be two instances (with Tripartite graph representations G and $\bar{G} \subseteq \mathcal{G}^{m,n}_{\text{QCQP}}$) that are not separated by the Tripartite WL-test. Without loss of generality, we assume that the variables and constraints are correspondingly indexed, i.e., $C^{T,u_j} = C^{T,\bar{u}_j}$ and $C^{T,c_i} = C^{T,\bar{c}_i}$ hold for all i, j.

We first introduce the following notations. Let I be any color, and we collect all nodes of a graph G with color I, denoting this collection as $G(I)$. Throughout this paper, we use J for the colors of variable nodes, K for quadratic nodes, and I for constraint nodes.

We now present the following lemma:
Lemma 2. Given the graph G, let the Tripartite WL-test stabilize after $T \geq 0$ iterations. The sum of weights from a certain node of one color to all nodes of another color depends only on the color of the given node. Specifically, the sum (taking J for variable nodes and K for quadratic nodes as an example) is:

$$S(J, K; G) := \sum_{C^{T,v}=K} w_{u,v}$$

and is well-defined with $u \in G(J)$ arbitrarily chosen.

Similarly, for any color of constraints I, color of variables J, and color of quadratic terms K, the following sums are well-defined:

$$\begin{aligned}
S(J, I; G) &:= \sum_{C^{T,c}=I} w_{u,c}, \quad C^{T,u} = J \\
S(I, K; G) &:= \sum_{C^{T,v}=K} w_{c,v}, \quad C^{T,c} = I \\
S(K, I; G) &:= \sum_{C^{T,c}=I} w_{v,c}, \quad C^{T,v} = K \\
S(J, K; G) &:= \sum_{C^{T,v}=K} w_{u,v}, \quad C^{T,u} = J \\
S(K, J; G) &:= \sum_{C^{T,u}=J} w_{v,u}, \quad C^{T,v} = K
\end{aligned}$$

Proof. Let v, v' be two nodes with color $K = C^{T,v} = C^{T,v'}$. Since the Tripartite WL-test has stabilized, further iterations do not separate additional node pairs, i.e.,

$$\sum_u w_{u,v} \text{HASH}(C^{T,u}) = \sum_u w_{u,v'} \text{HASH}(C^{T,u}).$$

Rearranging according to $J = C^{T,u}$, we get:

$$\sum_J \sum_{C^{T,u}=J} w_{u,v} \cdot \text{HASH}(J) = \sum_J \sum_{C^{T,u}=J} w_{u,v'} \cdot \text{HASH}(J).$$

Assuming that the hash function is collision-free, we conclude that:

$$\sum_{C^{T,u}=J} w_{u,v} = \sum_{C^{T,u}=J} w_{u,v'},$$

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

The other claims follow similarly. □

By summing all weights between two colors I and J, we derive the following lemma:

Lemma 3. Let J and K be arbitrary node colors. Then, the following holds:

$$|G(J)|S(J, K; G) = |G(K)|S(K, J; G),$$

and similar equalities hold between I and J, and between I and K.

Proof. Summing all edges between all nodes with $C^{T,u} = J$ and $C^{T,v} = K$, and re-arranging the sum according to u and v, by Lemma 2, we have:

$$|G(J)|S(J, K; G) = |G(K)|S(K, J; G).$$

The other two claims are similar. □

We are now ready to proceed. We construct $\bar{x}_j = \frac{1}{|G(J)|} \sum_{j':C^{T,u_{j'}}=C^{T,\bar{u}_{j'}}=J} x_{j'}$, where $J = C^{T,x_j}$. We claim that $\bar{x}$ satisfies all the required conditions.

First, we analyze the **linear part** of the constraints and the objective. Let $f^i_{\text{lin}}(x) := p^i \cdot x$ represent the linear part of the i-th constraint. For a certain color I of constraint nodes, we have:

$$\begin{aligned}
\bar{f}^i_{\text{lin}}(\bar{x}) &= \sum_j \bar{p}^i_j \bar{x}_j \\
&= \sum_J \sum_{\bar{v}_j \in G(J)} \bar{p}^i_j \bar{x}_j \\
&= \sum_J S(I, J)\bar{x}_J \\
&= \frac{1}{|G(I)|} \sum_J S(J, I)|G(J)|\bar{x}_J \\
&= \frac{1}{|G(I)|} \sum_J S(J, I) \sum_{u_j \in G(J)} x_j \\
&= \frac{1}{|G(I)|} \sum_{c_i \in G(I)} \sum_J \sum_{j \in G(J)} p^i_j x_j \\
&= \frac{1}{|G(I)|} \sum_{c_i \in G(I)} f^i_{\text{lin}}(x).
\end{aligned} \tag{B.1}$$

Here, $\bar{x}_j$ is the average over the nodes with color J, so it is determined by J, and we denote its value as $\bar{x}_J$.

We define $f_{\text{lin}}(x) = p \cdot x$. For the objective part, we have:

$$\begin{aligned}\sum_j \bar{p}_j \bar{x}_j &= \sum_J p_J |\bar{G}(J)| \bar{x}_J \\ &= \sum_J p_J \sum_{u_j \in G(J)} x_j \\ &= \sum_j p_j x_j,\end{aligned} \tag{B.2}$$

where $p_j, \bar{p}_j$ are the features of the variables, which are determined by the color $J = C^{T,u_j} = C^{T,\bar{u}_j}$. We denote this value by $p_j = \bar{p}_j = p_J$.

Quadratic part. We define $f^i_{\text{quad}}(x) = \frac{1}{2} x^\top Q^i x$ as the **quadratic** part of the i-th constraint.

For a certain color I of constraint nodes, we have the following:

$$\begin{aligned}\bar{f}^i_{\text{quad}}(\bar{x}) &= \frac{1}{2} \sum_{v_{j,k} \in V_2(\bar{G})} \bar{q}^i_{j,k} \bar{x}_j \bar{x}_k \\ &= \frac{1}{2} \sum_K \sum_{\bar{v}_{j,k} \in \bar{G}(K)} \bar{q}^i_{j,k} \bar{x}_j \bar{x}_k \\ &= \frac{1}{2} \sum_K S(I, K) |\bar{G}(K)| \bar{x}_K.\end{aligned}$$

Since all $\bar{v}_{j,k} \in V_2(\bar{G})$ have $\bar{u}_j, \bar{u}_k$ as neighbors in $V_1(\bar{G})$, $\bar{x}_K := \bar{x}_j \bar{x}_k$ is well-defined. This equation shows that the value $\bar{f}^i_{\text{quad}}(\bar{x})$ depends only on the color $I = C^{T,\bar{c}_i}$, and not on the specific selection of $\bar{c}_i \in \bar{G}(I)$. Therefore, $f^i_{\text{quad}}(\bar{x})$ reduces to the sum, and we claim that $f^i_{\text{quad}}(\bar{x}) = \bar{f}^i_{\text{quad}}(\bar{x})$ holds.

Next, we consider the partial derivative. Let $J := C^{T,u_j}$, and we have:

$$\begin{aligned}\partial_j \sum_{c_i \in G(I)} f^i_{\text{quad}}(\bar{x}) &= \sum_{c_i \in G(I)} \sum_k w(u_j, v_{j,k}) w(v_{j,k}, c_i) \bar{x}_k \\ &= \sum_{c_i \in G(I)} \sum_K \sum_{k : v_{j,k} \in G(K)} w(u_j, v_{j,k}) w(v_{j,k}, c_i) \bar{x}_k \\ &= \sum_K S(K, I) \sum_{k : v_{j,k} \in G(K)} w(u_j, v_{j,k}) \bar{x}_k \\ &= \sum_K S(K, I) S(J, K) x_{K;J}.\end{aligned} \tag{B.3}$$

Since u_j is one of the neighbors in $v_{j,k}$, and $v_{j,k} \in G(K)$ has exactly two neighbors in $V_1(G)$, we know that the color of u_k depends only on the colors $K = C^{T,v_{j,k}}$ and $J = C^{T,v_j}$. This makes $\bar{x}_{K;J} := \bar{x}_k$ well-defined, with $u_j \in G(J)$ and $v_{j,k} \in G(K)$.

Thus, the derivative $\partial_j \sum_{c_i \in G(I)}$ depends only on $J = C^{T,u_j}$, i.e.,

$$C^{T,v_{j_1}} = C^{T,v_{j_2}} \Rightarrow \partial_{j_1} \sum_{c_i \in G(I)} f^i_{\text{quad}}(\bar{x}) = \partial_{j_2} \sum_{c_i \in G(I)} f^i_{\text{quad}}(\bar{x}). \tag{B.4}$$

By equation B.4, we know that $\bar{x}$ is a local optimal point within the linear space:

$$\{y \in \mathbb{R}^n : \sum_{u_j \in G, C^{T,u_j} = J} y_j = \sum_{u_j \in G, C^{T,u_j} = J} x_j\}.$$

With the convexity assumption, the local optimal point is a global minimum. Since x is in this linear space, we claim that:

$$\sum_{c_i \in G(I)} f^i_{\text{quad}}(\bar{x}) \leq \sum_{c_i \in G(I)} f^i_{\text{quad}}(x). \tag{B.5}$$

Combining equation B.5 with the fact that $f^i_{\text{quad}}(\bar{x})$ and $\bar{f}^i_{\text{quad}}(\bar{x})$ are equal for all $c_i \in G(I)$, we can control the quadratic parts:

$$\begin{aligned}\bar{f}^i_{\text{quad}}(\bar{x}) &= f^i_{\text{quad}}(\bar{x}) \\ &= \frac{1}{|G(I)|}\sum_{c_i \in G(I)} f^i_{\text{quad}}(\bar{x}) \\ &\le \frac{1}{|G(I)|}\sum_{c_i \in G(I)} f^i_{\text{quad}}(x).\end{aligned} \tag{B.6}$$

For the objective part, we define $f_{\text{quad}}(x) = \frac{1}{2}x^\top Qx$. Similarly, we have:

$$\begin{aligned}\bar{f}_{\text{quad}}(\bar{x}) &= \frac{1}{2}\sum_{\bar{v}_{j,k}} f^0(\bar{v}_{j,k})\bar{x}_j\bar{x}_k \\ &= \frac{1}{2}\sum_K \sum_{\bar{v}_{j,k}\in \bar{G}(K)} f^0(\bar{v}_{j,k})\bar{x}_j\bar{x}_k \\ &= \frac{1}{2}\sum_K \sum_{v_{j,k}\in G(K)} f^0(v_{j,k})\bar{x}_j\bar{x}_k \\ &= f_{\text{quad}}(\bar{x}).\end{aligned}$$

We also have:

$$\begin{aligned}\partial_j f_{\text{quad}}(\bar{x}) &= \sum_k f^0(v_{j,k})w(u_j, v_{j,k})\bar{x}_k \\ &= \sum_K \sum_{k: v_{j,k}\in G(K)} f^0(v_{j,k})w(u_j, v_{j,k})\bar{x}_k \\ &= \sum_K f^0(K)S(J,K)\bar{x}_{K;J},\end{aligned}$$

which depends only on $J = C^{T,u_j}$. Here, $f^0(K) = f^0(v_{j,k})$, and $v_{j,k} \in G(K)$ is well-defined by the stable color assumption.

Combination of the two parts.

The color $C^{T,c_i} = C^{T,\bar{c}_i} = I$ determines the RHS $b_I := b_i$. Defining $f^i_{\text{cons}}(x) = f^i_{\text{quad}}(x) + f^i_{\text{lin}}(x)$, and similarly for $\bar{\mathcal{I}}$, we have:

$$\begin{aligned}\bar{f}^i_{\text{cons}}(\bar{x}) &= \bar{f}^i_{\text{quad}}(\bar{x}) + \bar{f}^i_{\text{lin}}(\bar{x}) \\ &= \frac{1}{|G(I)|}\sum_{c_i\in G(I)} \left(f^i_{\text{quad}}(x) + f^i_{\text{lin}}(x)\right) \\ &= \frac{1}{|G(I)|}\sum_{c_i\in G(I)} f^i_{\text{cons}}(x) \\ &\le b_I.\end{aligned}$$

For the objective, we similarly have:

$$\bar{f}_{\text{quad}}(\bar{x}) + \bar{f}_{\text{lin}}(\bar{x}) \le f_{\text{quad}}(x) + f_{\text{lin}}(x).$$

This completes the proof that $\bar{x}$ is the solution for $\bar{\mathcal{I}}$, satisfying the condition given in Proposition 1.

B.3 Proof of Proposition 4

We prove the separation power by simulating the tripartite WL-test using tripartite MP-GNNs. We define the hidden representation $h^{t,\cdot}$, produced by some network, as a **one-hot representation** of the colors $C^{t,\cdot}$ if all $h^{t,\cdot}$ are one-hot vectors, and they take the same value if and only if they have the same color $C^{t,\cdot}$.

First, we consider the color initialization. We collect all the features paired with the node types (i.e., variable nodes, quadratic nodes, and constraint nodes). Then we select $g^0_{1,2,3}$ to map the features to one-hot vectors, where the enumeration serves as the only index with the value 1.0. For example, if the feature h^{u_j} of a variable node is enumerated by r, then g^0_1 maps h^{u_j} to $h^{0,u_j} = e_r$.

It's easy to see that the embedded hidden feature $h^{0,\cdot}$ is a one-hot representation of the initial color $C^{0,\cdot}$.

Next, we consider the first refinement. Assuming that $h^{t,\cdot}$ is a one-hot representation of $C^{t,\cdot}$ and $g^t_1 = \mathrm{id}$ is a simple and proper hash function, the concatenated vector

$$\left[h^{t,v}, \sum_{u\in V_1} w_{u,v} f^t_1(h^{t,u})\right]$$

is a representation of the colors $\bar{C}^{t,\cdot}$, which is generally not one-hot. The same holds for the other three concatenated vectors from the remaining three sub-layers. By Theorem 3.2 of Yun et al., 2019, a network with four fully connected layers and ReLU activation maps these values back to one-hot. Therefore, we select f^t_1 to concatenate the inputs and then pass them through a 4-layered MLP with ReLU activation, so that the aggregated hidden representation $\bar{h}^{t,\cdot}$ is once again one-hot.

Similarly, we get $h^t_{2,3,4}$ and $g^t_{2,3,4,5,6}$ and simulate an iteration of the Tripartite WL-test with a round of four message-passing sub-layers.

In the case of graph-level output, the readout function takes the following form:

$$R(\cdot) = f_{\text{out}}\left(\sum_j h^{T,u_j}, \sum_{j,k} h^{T,v_{j,k}}, \sum_i h^{T,c_i}\right).$$

Since the hidden representation is a one-hot representation of $C^{T,\cdot}$, if two instances are not separated by the tripartite message-passing GNN, they are not separated by this subset of GNNs (given a fixed initialization and a free readout function). Consequently, all entries must be equal, and the two instances are not separated by the Tripartite WL-test.

Similarly, in the case of node-level output, all equivariant readout functions take the form:

$$R(\cdot)_j = f_{\text{out}}\left(h^{T,u_j}, \sum_j h^{T,u_j}, \sum_{j,k} h^{T,v_{j,k}}, \sum_i h^{T,c_i}\right).$$

Thus, all entries must be equal, and the two instances are not separated. Moreover, the variables are correspondingly indexed.

Conversely, we use induction to prove that for all $t \in \mathbb{N}$, the colors $C^{t,\cdot}$ separate more than the hidden features $h^{t,\cdot}$, i.e.,

$$C^{t,u} = C^{t,u'} \Rightarrow h^{t,u} = h^{t,u'}, \quad \forall u, u' \in V_1 \cup \bar{V}_1, F \in \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^s), \tag{B.7}$$

and similar claims hold for the other three sub-iterations.

For $t = 0$ (i.e., right after embedding), the statement is obviously true. Now, assume that after some sub-iteration (say, before the first sub-iteration of iteration $t \geq 1$, with the other sub-iterations following similarly), the statement holds.

Let v, v' satisfy:

$$\sum_u w_{u,v} \text{HASH}(C^{t,u}) = \sum_u w_{u,v'} \text{HASH}(C^{t,u}).$$

Organizing the sum by $C^{t,u} = J$, and assuming the hash function is collision-free, we have:

$$\sum_{u:C^{t,u}=J} w_{u,v} = \sum_{u:C^{t,u}=J} w_{u,v'}, \quad \forall J. \tag{B.8}$$

Next, we organize the sum $\sum_u w_{u,v} f_1^t(h^{t,u})$ by the value of $h^{t,u}$. By the induction assumption, the set $\{u : h^{t,u} = h\}$ is the union of $\{u : C^{t,u} = J_l\}$ for some colors J_l. Summing the equality in equation B.8 over the colors, we have:

$$\sum_{h^{t,u}=h} w_{u,v} = \sum_{h^{t,u}=h} w_{u,v'}, \quad \forall h.$$

Thus, we conclude:

$$\sum_u w_{u,v} f_1^t(h^{t,u}) = \sum_h \sum_{u:h^{t,u}=h} w_{u,v} f_1^t(h) = \sum_h \sum_{u:h^{t,u}=h} w_{u,v'} f_1^t(h) = \sum_u w_{u,v'} f_1^t(h^{t,u}),$$

which completes the induction.

For the case of graph-level output, this means that all entries of the input to the readout function are equal for the two graphs, i.e.,

$$\left(\sum_j h^{T,u_j}, \sum_{j,k} h^{T,v_{j,k}}, \sum_i h^{T,c_i}\right) = \left(\sum_j h^{T,\bar{u}_j}, \sum_{j,k} h^{T,\bar{v}_{j,k}}, \sum_i h^{T,\bar{c}_i}\right),$$

and the GNNs give the same output for all possible readout functions.

For the case of node-level output, we again have:

$$\left(h^{T,u_j}, \sum_j h^{T,u_j}, \sum_{j,k} h^{T,v_{j,k}}, \sum_i h^{T,c_i}\right) = \left(h^{T,\bar{u}_j}, \sum_j h^{T,\bar{u}_j}, \sum_{j,k} h^{T,\bar{v}_{j,k}}, \sum_i h^{T,\bar{c}_i}\right).$$

Here, we use the assumption that the variables are correspondingly indexed to guarantee $h^{T,u_j} = h^{T,\bar{v}_j}$.

B.4 Proof of Proposition 6

The requirement for the general target function Φ_c is simply equivariance under re-indexing. Thus, we need to verify the conditions required by the generalized Weierstrass theorem (Theorem 22 of Azizian & Lelarge (2020)) to apply.

First, we verify that $\mathcal{F} = \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^s)$ is a sub-algebra. By multiplying the readout function by λ, we construct $\lambda F \in \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^s)$. Now, we construct the sum and product of two functions $F_1, F_2 \in \mathcal{F}^{m,n}_{\text{QCQP}}(\mathbb{R}^s)$.

Given F_1 and F_2, we proceed as follows:

- We construct
$$g^0_{1,F}(h^{0,u}) := \left[g^0_{1,F_1}(h^{0,u}), g^0_{1,F_2}(h^{0,u})\right].$$
We give similar constructions for $g^0_{2,F}$ and g^0_3.
- After initialization, all hidden features take the form $h^{t,u} = [h^{t,u}_{F_1}, h^{t,u}_{F_2}]$ (considering variable nodes as an example, and similarly for quadratic nodes). We construct
$$g^t_{1,F}(h^{t,u}) := \left[g^t_{1,F_1}(h^{t,u}_{F_1}), g^t_{1,F_2}(h^{t,u}_{F_2})\right],$$
and
$$f^t_{1,F}(h^{t,v}, \sum_u w_{uv} h^{t,u}) := \left[f^t_{1,F_1}(h^{t,v}_{F_1}, \sum_u w_{uv} h^{t,u}_{F_1}), f^t_{1,F_2}(h^{t,v}_{F_2}, \sum_u w_{uv} h^{t,u}_{F_2})\right].$$
We give similar constructions for other $g^t_{\cdot,F}, f^t_{\cdot,F}$. Using this construction, we compute both hidden representations in one concatenated network.

- Finally, we obtain $F = F_1 + F_2$ by constructing $R(\cdot) = R_1(\cdot_{F_1}) + R_2(\cdot_{F_2})$, and similarly for $F = F_1 \times F_2$.

Thus, we conclude that $F_1 + F_2, F_1 \times F_2 \in \mathcal{F}^{m,n}_{\mathrm{QCQP}}(\mathbb{R}^s)$.

Next, we verify the inclusion $\rho(\mathcal{F}_{\mathrm{scal}}) \subseteq \rho(\pi_\Sigma \circ \mathcal{F})$:

Graph-level output case. In this case, we have $\mathcal{F}_{\mathrm{scal}} = \mathcal{F}$ and $\pi_\Sigma = \mathrm{id}$, so the two sides are exactly the same.

Node-level output case. Given any R_1 that maps the final hidden representation to a graph-level output, $R_1 \cdot \mathbf{1}_n = (R_1, R_1, \ldots, R_1)$ is a valid equivariant readout function in the node-level case. Thus, given any $F \in \mathcal{F}^{m,n}_{\mathrm{QCQP}}(\mathbb{R})$, we can construct $F' \in \mathcal{F}^{m,n}_{\mathrm{QCQP}}(\mathbb{R}^n)$ using $R_1 \cdot \mathbf{1}_n$, along with all the f and g functions, and conclude that any pair $(G_1, G_2) \in \rho(\mathcal{F}_{\mathrm{scal}})$ is not separated by the Tripartite WL-test.

For any pair of graphs $(G, \bar{G})$ that is not separated by the Tripartite WL-test, after re-indexing variables and constraints, all $F \in \mathcal{F}$ map them to the same output. This means that, without re-indexing, all $F \in \mathcal{F}$ map the two graphs to outputs that differ at most by a re-indexing. Thus, $(G, \bar{G})$ is contained in $\rho(\pi_\Sigma \circ \mathcal{F})$. This completes our verification.

Applying the Generalized Weierstrass-Stone theorem to the sub-algebra $\mathcal{F} = \mathcal{F}^{m,n}_{\mathrm{QCQP}}(\mathbb{R}^s)$ completes the proof.

C Proof of propositions in Section 3.3

The two instances are QCQP instances. Both graphs G and $\bar{G}$ consist of the following:

- 6 variable nodes, i.e., u_j or $\bar{u}_j$, where $j \in [6]$. All nodes carry the feature $h^{u_j} = (0, -1, 1)$. here we assume that $-1 \leq x_i \leq 1$ by the unit ball constraint.
- 12 effective quadratic nodes. The squared nodes carry $h^{v_{j,j}} = (0)$, while others carry the feature $h^{v_{j,k}} = (1)$.
- 1 constraint node c representing the unit ball constraint. The node carries feature (-1) for both graphs.

We now verify that the Tripartite WL-test does not separate the two graphs:

- After initialization, we have $h^0_1 := h^{0,u} = h^{0,\bar{u}} = \mathrm{HASH}_1((0,-1,1))$, $h^0_2 := h^{0,v_{j,j}} = h^{0,\bar{v}} = \mathrm{HASH}_2((0))$, $h^0_3 := h^{0,v_{j,k}} = \mathrm{HASH}_2((1))$ and $h^0_4 := h^{0,c} = h^{0,\bar{c}} = \mathrm{HASH}_3((-1))$.
- After the first sub-iteration, we have
$$\bar{h}^0_2 := \bar{h}^{0,v_{j,k}} = \mathrm{HASH}(h^0_2, 2h^0_1),$$
and
$$\bar{h}^0_3 := \bar{h}^{0,v_{j,k}} = \mathrm{HASH}(h^0_3, 2h^0_1),$$
which remains equal for all $v \in V_2(G)$ and $\bar{v} \in V_2(\bar{G})$.
- After the second sub-iteration, we have
$$h^1_4 := h^{1,c} = h^{1,\bar{c}} = \mathrm{HASH}(h^0_4, 0, 1 \cdot \bar{h}^0_2),$$
which remains equal for both graphs.
- After the third sub-iteration, we have
$$h^1_2 := h^{1,v_{j,j}} = \mathrm{HASH}(\bar{h}^0_2, 1 \cdot h^1_3),$$
and
$$h^1_3 := h^{1,v_{j,k}} = \mathrm{HASH}(\bar{h}^0_3, 0),$$
which remains equal for both graphs.

- After the final sub-iteration, we have
$$h_1^1 := h^{1,u} = h^{1,\bar{u}} = \mathrm{HASH}(h_1^0, 0, 2 \cdot h_2^1 + 1 \cdot h_3^1),$$
which remains equal for both graphs.
- The Tripartite WL-test terminates after one iteration since no further node pairs are separated.

The Tripartite WL-test returns $C^{0,\cdot}$, which is the same for both instances. Thus, we conclude that the two graphs are not separated, with variables and constraints correspondingly indexed. By Proposition 4, we conclude that, in both the node-level and graph-level cases, tripartite MP-GNNs cannot separate the two instances.

Therefore, we conclude that tripartite MP-GNNs cannot approximate the optimal solution or optimal value for non-convex QCQP instances (even QP instances). To demonstrate that tripartite MP-GNNs cannot accurately predict feasibility, we slightly modify the two instances:

Proof of Proposition 1. We reconstruct the objective as a constraint. Specifically, consider the following two instances:

$$\begin{aligned} \min_{x \in \mathbb{R}^6} \quad & 0 \\ \text{s.t.} \quad & x_1x_2 + x_2x_3 + x_3x_1 + x_4x_5 + x_5x_6 + x_6x_4 \le -\frac{3}{4} \\ & \sum_{i=1}^{6} x_i^2 \le 1 \end{aligned} \tag{C.1}$$

and

$$\begin{aligned} \min_{x \in \mathbb{R}^6} \quad & 0 \\ \text{s.t.} \quad & x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_1 \le -\frac{3}{4} \\ & \sum_{i=1}^{6} x_i^2 \le 1 \end{aligned} \tag{C.2}$$

Clearly, instance C.1 is not feasible, while instance C.2 is feasible.

In the graph generated by the Tripartite graph representation, we change the objective to another special constraint and add a new dummy objective. Similarly, we see that tripartite MP-GNNs fail to separate $\mathcal{I}$ and $\bar{\mathcal{I}}$. □

D Additional experiments

Evaluation on the QPLIB dataset. We incorporated a dataset derived from real-world instances in QPLib. Due to computational constraints, we used all instances in QPLIB that are convex and have no more than 5,000 nonzeros as training sets, and augmented these instances by perturbations. Below, we report the training and validation losses in Table 5 and Table 6, respectively.

Table 5: Training loss vs. numbers of parameters on QPLIB.

Target	# Parameters				
	15K	42K	126K	450K	1.7M
objective	4.3306	0.9986	0.8679	0.7999	0.6614
solution	19.2789	18.9083	18.8303	18.7472	18.6495

As shown, both training and validation losses improve consistently with increased model capacity and training samples. Such a trend supports the method's effectiveness on real-world QCQP instances.

Table 6: Validation loss vs. numbers of training samples on QPLIB.

Target	# Samples				
	100	300	500	700	1000
objective	1.5414	0.8922	0.8564	0.6176	0.5598
solution	19.9106	19.7552	19.6981	19.6788	19.6165

Evaluation on boundedness. We further evaluated performance on the task of predicting the boundedness of convex QCQPs. Specifically, unbounded cases arise when there exists a direction d such that $Qd = 0$ and $p^\top d < 0$ for both the objective and constraints. To examine performance under these challenging edge cases, we generated a dedicated dataset by randomly enforcing such conditions. Using the same training setup as in Section 4.4, the corresponding training and validation losses are reported in Tables 7 and 8.

Table 7: Training loss vs. number of parameters on predicting boundedness.

#Params	21k	42k	126k	1.7M	6.7M
Train loss	0.6933	0.2504	0.1241	0.0470	0.0427

Table 8: Validation loss vs. number of training samples on predicting boundedness.

#Samples	100	300	500	700	1000
Valid loss	0.6876	0.3495	0.2786	0.2360	0.1285

The results show a clear trend: training loss decreases with larger model sizes, and validation loss improves with more training data. These results suggest that our approach is capable of learning effectively on unbounded problem instances.