# MSP-GNN: LEARNING FIRST-STAGE DECISIONS IN MULTISTAGE STOCHASTIC PROGRAMMING VIA SUBTREE PERMUTATION INVARIANCE

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Scenario-based multistage stochastic programming (MSP) provides a principled framework for sequential decision-making under uncertainty, but the exponential growth of scenario trees with the planning horizon poses a major computational barrier to real-time optimization. Learning-based solution predictors offer a promising alternative. However, fully connected architectures that flatten scenario trees into fixed-length vectors overlook their hierarchical symmetries and require parameter counts that grow with tree size. We focus on first-stage, or here-and-now, decisions, which are operationally critical because they must be committed before future uncertainty unfolds while accounting for all subsequent recourse. Our central observation is that optimal first-stage decisions are invariant to recursive permutations of child subtrees at any node of the scenario tree. We exploit this symmetry to introduce MSP-GNN, an end-to-end graph neural network that enforces subtree permutation invariance by construction and has a parameter count independent of scenario-tree size. To characterize its expressive power, we develop a Tree-Weisfeiler-Lehman (TWL) framework tailored to MSP. We prove that TWL-indistinguishable scenario trees admit identical optimal first-stage decisions and that MSP-GNN can simulate the TWL test. Building on these results, we establish a universal approximation guarantee for the optimal first-stage decision mapping on compact domains. Experiments on multistage asset allocation and weighted stochastic linear-quadratic regulator problems demonstrate that, across balanced and asymmetric scenario trees, MSP-GNN achieves predictive accuracy comparable to or better than fully connected networks and DeepSets while using substantially fewer parameters. These findings establish subtree permutation invariance as an effective inductive bias for learning first-stage decisions in MSP.

## 1 INTRODUCTION

Multistage stochastic programming (MSP) is a foundational framework for sequential decision-making under uncertainty, in which a decision-maker successively observes random outcomes and adapts decisions to the information revealed at each stage. By allowing decisions to adapt as uncertainty unfolds, MSP provides a natural modeling framework for a broad class of real-world planning problems, including power systems operations (Agarwal et al., 2022; Conejo et al., 2010), supply chain management (Santoso et al., 2005; Ahmed et al., 2003), and financial planning (Davari-Ardakani et al., 2016; Carino & Ziemba, 1998). Despite this broad applicability, solving MSP at scale remains computationally intractable. Central to this difficulty is that, at each stage, the uncertainty space branches into multiple possible realizations, and the scenario tree that discretizes this branching process grows exponentially with the planning horizon. Conventional iterative solution methods (Pereira & Pinto, 1991; Rockafellar & Wets, 1991; Birge, 1985), such as stochastic dual dynamic programming and scenario decomposition, must repeatedly traverse this scenario tree, either via forward-backward passes or node-wise subproblem decomposition, incurring a computational cost that grows rapidly with the number of stages. This makes these methods prohibitively slow for time-sensitive applications.

Recent advances in machine learning have introduced end-to-end neural network approaches that bypass iterative solvers by directly learning the mapping from problem parameters to optimal deci-



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sions (Bengio et al., 2021; Kotary et al., 2021; Donti et al., 2017; Liang et al., 2023; Zhou et al., 2022), enabling real-time inference via a single forward pass. However, in MSP, the uncertainty space is discretized into a *scenario tree*, where each node represents the history of uncertainty realizations up to a given stage, and each root-to-leaf path represents a complete future scenario. Applying end-to-end architectures to this structure introduces a fundamental representational bottleneck: existing approaches concatenate all scenario tree nodes into a flat vector fed into a fully connected network (Donti et al., 2017; Pan et al., 2020), whose input dimension grows with the scenario tree size and quickly becomes intractable as the number of stages increases. A natural remedy is to exploit permutation symmetry via permutation-invariant architectures (Zaheer et al., 2017; Qi et al., 2017), which have demonstrated strong performance in two-stage stochastic programming problems (Zhou et al., 2025; Zhang et al., 2023). However, scenario tree nodes are not globally exchangeable; each node is uniquely bound to its parent through the state transition dynamics, and directly applying global permutation invariance fails to preserve these parent-child relationships (Shapiro et al., 2021). Designing an end-to-end architecture that is simultaneously scalable and structure-preserving therefore remains an important open problem. See Table 1 for an overview of existing ML methods for MSP and their limitations.

In this paper, we propose MSP-GNN, an efficient end-to-end learning framework for predicting the optimal first-stage decision of an MSP directly from its scenario-tree representation by exploiting the tree's intrinsic permutation symmetry. Our main contributions are as follows:

▷ In Section 4, we formalize a structural symmetry of MSP: the optimal first-stage decision is invariant under recursive permutations of the child subtrees at each node of the scenario tree, and we propose a Graph Neural Network (GNN) that natively encodes this symmetry by construction. The GNN's permutation equivariance, combined with a readout function anchored at the root node, guarantees that the predicted decision inherits the required invariance without additional regularization, decoupling model complexity from scenario tree size and directly resolving the scalability bottleneck of fully connected baselines. A visual comparison between the vector-flattened input of conventional FCNNs and the graph-structured input of MSP-GNN is provided in Appendix B.

▷ In Section 5, we develop a Tree-Weisfeiler-Lehman (TWL) framework to theoretically certify the expressive power of our architecture. We establish that TWL-indistinguishable scenario trees share identical optimal first-stage decisions (Lemma 1), prove that our GNN simulates the TWL test (Lemma 2), and conclude that our architecture can uniformly approximate the optimal first-stage decision mapping to arbitrary precision over any compact domain of scenario trees via a homeomorphism argument on the TWL quotient space (Theorem 1).

▷ In Section 6, we evaluate the empirical performance of MSP-GNN on asset allocation and weighted stochastic LQR problems. Our computational experiments confirm the expressive capacity of the GNN model, demonstrating that it delivers superior predictive accuracy while requiring a substantially smaller parameter count compared to standard fully connected neural network baselines, and yields millisecond-level inference for first-stage decision prediction.

To the best of our knowledge, MSP-GNN is the first end-to-end learning framework for learning first-stage decisions from general MSP scenario trees that jointly preserves temporal tree topology, admits universal approximation guarantees, and achieves strong empirical parameter efficiency.

## 2 Related work

Machine learning has been applied broadly to constrained optimization (Pan et al., 2020; Donti et al., 2017); a detailed review is provided in Appendix A. Here we focus on methods specifically targeting MSP.

### 2.1 Machine Learning for MSP Problems

Recent research has explored machine learning approaches for accelerating MSP decision-making along three distinct directions, each with a characteristic limitation, as summarized in Table 1.

**Neural scenario reduction.** Data-driven scenario management employs machine learning to handle the uncertainty space, either by generating statistically representative scenarios via generative adversarial networks (Chen et al., 2018) or by selecting a subset of representative scenarios via neural networks (Bengio et al., 2020; Keutchayan et al., 2023). While scenario reduction decreases compu-

Table 1: Existing machine learning methods for MSP. ✓ denotes that the property is satisfied and ✗ denotes otherwise.

| Category | Method | Multistage support | Low inference cost | Scalable to tree size | Theoretical guarantee |
|---|---|---|---|---|---|
| Scenario reduction | CSSC (Keutchayan et al., 2023) | ✓ | ✓ | ✓ | ✗ |
| NN-based iterative methods | NeuralADP (Shah et al., 2020) | ✓ | ✗ | ✗ | ✗ |
| | DRL (Ghadimi et al., 2020) | ✓ | ✗ | ✗ | ✗ |
| | Neur2SP (Patel et al., 2022) | ✗ | ✗ | ✗ | ✗ |
| | VFGL (Bae et al., 2023) | ✓ | ✗ | ✗ | ✗ |
| | TranSDDP (Kim et al., 2023) | ✓ | ✗ | ✗ | ✗ |
| End-to-end methods | SO-EBM (Kong et al., 2022) | ✓ | ✓ | ✗ | ✓ |
| | Task-based learning (Donti et al., 2017) | ✓ | ✓ | ✗ | ✓ |
| | **MSP-GNN (this work)** | ✓ | ✓ | ✓ | ✓ |

[1] Scenario reduction methods achieve low inference cost at the expense of tail-risk coverage, potentially compromising solution quality for rare yet critical uncertainty realizations.

[2] Neur2SP (Patel et al., 2022) is designed for two-stage stochastic programs and does not extend to multistage scenario trees.

[3] End-to-end methods achieve low inference cost but concatenate scenario inputs into flat vectors, causing model size to grow significantly with tree size.

[4] MSP-GNN is the first end-to-end method to jointly achieve all four properties by treating the scenario tree as a graph and exploiting its intrinsic permutation symmetry.

tational complexity, it risks degrading solution quality if critical tail-risk scenarios are inadvertently removed, a limitation that the reduction procedure itself cannot resolve.

**NN-based iterative methods.** NN-based iterative methods integrate machine learning components into iterative MSP solvers to enhance computational efficiency, through strategies such as guiding branch-and-bound variable selection (Lodi & Zarpellon, 2017), replacing interior point sub-routines with input convex neural networks (Bae et al., 2023), and accelerating stage-wise decomposition algorithms via Transformers (Kim et al., 2023). While these approaches improve solver efficiency, they remain computationally expensive at inference time due to their reliance on repeated neural network evaluations and iterative solution procedures (Nair et al., 2018; Ghadimi et al., 2020; Patel et al., 2022; Kim et al., 2023).

**End-to-end methods.** End-to-end methods bypass iterative solvers entirely by directly learning the mapping from uncertainty inputs to optimal decisions, minimizing task-oriented loss functions (Donti et al., 2017; Rychener et al., 2023; Wilder et al., 2019). This paradigm, pioneered by OptNet (Amos & Kolter, 2017) and Smart "Predict, then Optimize" (SPO) (Elmachtoub & Grigas, 2022), consistently outperforms traditional predict-then-optimize strategies with respect to solution quality. However, existing end-to-end architectures rely on fixed, finite-dimensional input structures. When extended to MSP, they process scenario inputs as flat vectors, causing model size to grow exponentially with the number of stages, a scalability failure that becomes severe in the multistage setting where the scenario tree grows combinatorially with the planning horizon.

The three directions above collectively reveal a single unresolved gap: as summarized in Table 1, no existing approach jointly achieves multistage support, low inference cost, scalability to scenario tree size, and theoretical approximation guarantees. To close this gap, we propose MSP-GNN, the first end-to-end learning framework to jointly satisfy all four properties.

## 3 Problem Setting

### 3.1 Multistage Stochastic Programming

We consider a finite-horizon discrete-time sequential decision-making problem over stages $t \in \mathcal{T} := \{1, \ldots, T\}$. The problem is initialized at a given deterministic initial state $s_1$. At each stage $t$, the

system occupies a state $s_t \in \mathcal{S}_t$. Upon observing an exogenous random uncertainty vector $\xi_t \in \Xi_t$, where $\xi_1$ is deterministic without loss of generality, a control decision $x_t \in \mathcal{X}_t$ is executed, following the standard observe-then-decide temporal sequence (Shapiro et al., 2021; Birge & Louveaux, 1997):

$$s_1 \to \xi_1 \to x_1 \to s_2 \to \cdots \to \xi_T \to x_T.$$

In this work, we consider a rolling-horizon setting (Siddig et al., 2021), in which an MSP problem is solved at each decision epoch using the current state and a scenario tree constructed from the latest available information; only the first-stage decision is implemented, after which the horizon advances and the problem is re-solved as new information becomes available.

To make this sequential structure tractable, we operate under the Markov assumption, under which the current state $s_t$ serves as a sufficient statistic for the entire historical trajectory up to stage $t$ (Puterman, 2014; Powell, 2007). Under this assumption, the decision at each stage need only depend on the currently available information $(s_t, \xi_t)$, rather than the full history of past realizations. We therefore model each stage-$t$ decision as a *policy* $\pi_t : \mathcal{S}_t \times \Xi_t \to \mathcal{X}_t$, which maps the currently available information $(s_t, \xi_t)$ to a control decision $x_t = \pi_t(s_t, \xi_t)$. Non-anticipativity requires that the decision at each stage is based solely on the history of uncertainty realizations observed so far, ensuring that no decision inadvertently relies on future information that is not yet available to the decision-maker.

The objective of the MSP is to find an optimal sequence of non-anticipative policies $\{\pi_t\}_{t=1}^T$ that minimizes the expected cumulative cost:

$$\min_{\{\pi_t\}_{t=1}^T} \mathbb{E}_{\{\xi_t\}_{t=1}^T} \left[ \sum_{t=1}^{T} f_t(s_t, \pi_t(s_t, \xi_t), \xi_t) \right] \tag{1}$$

$$\text{s.t. } s_{t+1} = g_t(s_t, \pi_t(s_t, \xi_t), \xi_t), \quad \forall t \in \mathcal{T} \setminus \{T\}, \tag{2}$$

$$h_t(s_t, \pi_t(s_t, \xi_t), \xi_t) \le 0, \quad \text{a.s.}, \quad \forall t \in \mathcal{T}, \tag{3}$$

$$\pi_t(s_t, \xi_t) \in \mathcal{X}_t, \quad \forall t \in \mathcal{T}. \tag{4}$$

Here, $f_t$ denotes the stage-wise cost function, $g_t$ defines the state transition dynamics enforcing the Markov property, $h_t \le 0$ represents the feasibility constraints of the system, and $\mathcal{X}_t$ denotes the feasible decision set at stage $t$. The inequality constraints are required to hold almost surely (a.s.), i.e., with probability one over all possible uncertainty realizations $\{\xi_t\}_{t=1}^T$.

### 3.2 Sample Average Approximation with Scenario Tree

We approximate the stochastic process with a finite discrete scenario tree $\mathcal{T}_s = (\mathcal{N}, \mathcal{E})$ (Heitsch & Römisch, 2009; Dupačová et al., 2000), illustrated in Figure 1, where $\mathcal{N} = \bigcup_{t=1}^T \mathcal{N}_t$. The root node $n_0 \in \mathcal{N}_1$ has no parent and is initialized with a fixed initial state $s_{n_0}$ and known initial realization $\xi_{n_0}$. Each non-root node $n \in \mathcal{N}_t$ for $t \ge 2$ has a unique parent $a(n) \in \mathcal{N}_{t-1}$, and each internal node $n \in \mathcal{N}_t$ for $t \le T-1$ has a non-empty set of children $\mathcal{C}(n) \subseteq \mathcal{N}_{t+1}$; leaf nodes $n \in \mathcal{N}_T$ have no children. Each node carries a probability $p_n = 1/|\mathcal{N}_T|$ for leaf nodes and $p_n = \sum_{c \in \mathcal{C}(n)} p_c$ for internal nodes. The state $s_n$ and decision $x_n$ at node $n$ correspond to stage $t(n)$; for each non-root node $n$, the transition into $n$ is driven by the parent decision $x_{a(n)}$, the parent state $s_{a(n)}$, and the newly realized uncertainty at the current node $\xi_n$, consistent with the observe-then-decide sequence. Since each node $n$ encodes a unique root-to-$n$ history, $x_n$ implicitly represents the policy evaluation $\pi_{t(n)}(s_n, \xi_n)$, where $\xi_{n_0}$ denotes the known initial realization at the root, and non-anticipativity is encoded structurally by the tree topology. The MSP problem (1)–(4) thus reduces to the deterministic approximation:

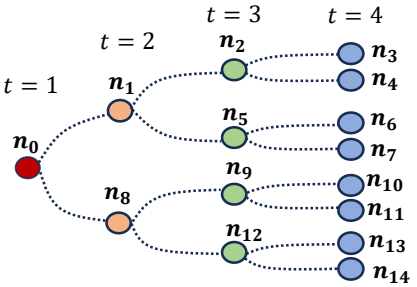


Figure 1: A multistage scenario tree illustrating the branching of uncertainty realizations and the corresponding node hierarchy across stages.

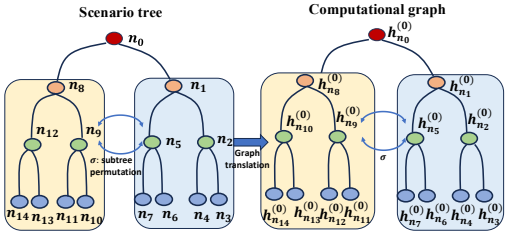


Figure 2: Translation of the scenario tree $\mathcal{T}_s$ into the computational graph $G = (V, E)$. The subtree permutations $\sigma$ correspond to graph automorphisms of $G$.

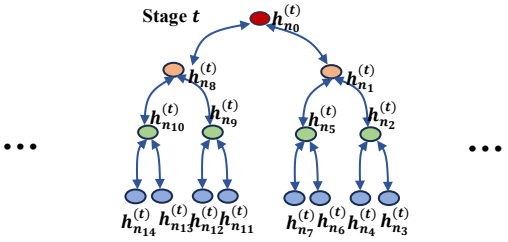


Figure 3: Message passing on the computational graph $G$ over $L = T - 1$ layers. At each layer, every node exchanges messages bidirectionally with its neighbors.

$$\min_{\{x_n\}_{n\in\mathcal{N}}} \quad \sum_{n\in\mathcal{N}} p_n f_{t(n)}(s_n, x_n, \xi_n) \tag{5}$$

$$\text{s.t.} \quad s_n = g_{t(n)-1}\left(s_{a(n)}, x_{a(n)}, \xi_n\right), \quad \forall n \in \mathcal{N} \setminus \mathcal{N}_1, \tag{6}$$

$$h_{t(n)}(s_n, x_n, \xi_n) \leq 0, \quad \forall n \in \mathcal{N}, \tag{7}$$

$$x_n \in \mathcal{X}_{t(n)}, \quad \forall n \in \mathcal{N}, \tag{8}$$

$$s_n = s_1, \quad \forall n \in \mathcal{N}_1. \tag{9}$$

The constraint (7) is enforced for all nodes $n \in \mathcal{N}$, each of which carries strictly positive probability $p_n > 0$ by construction, replacing the almost-sure qualification of the continuous setting with explicit node-wise enforcement over the finite scenario tree.

## 4 Graph Neural Network for MSP

To efficiently learn the optimal first-stage decision $x_1^*$ while circumventing the curse of dimensionality, we require an architecture that explicitly exploits the intrinsic symmetry of MSP. We formalize this symmetry as follows.

**Proposition 1** (Recursive Subtree Permutation Invariance)**.** *Assume the associated MSP problem admits a unique optimal first-stage decision. Let $\mathcal{T}_s = (\mathcal{N}, \mathcal{E})$ be a discrete scenario tree and let $x_1^*(\mathcal{T}_s)$ denote the unique optimal first-stage decision. Let $\sigma : \mathcal{N} \to \mathcal{N}$ be a bijection that (i) fixes the root $\sigma(n_0) = n_0$; (ii) preserves stage indices $t(\sigma(n)) = t(n)$; and (iii) relocates each node $n$ together with its uncertainty realization $\xi_n$ and probability $p_n$ to position $\sigma(n)$, preserving the parent-child structure: $c \in \mathcal{C}(n) \implies \sigma(c) \in \mathcal{C}(\sigma(n))$. Then:*

$$x_1^*(\sigma(\mathcal{T}_s)) = x_1^*(\mathcal{T}_s). \tag{10}$$

*Remark* 1 (Existence of the bijection). The bijection $\sigma$ is induced by the recursive subtree permutation and therefore requires no additional assumption. Let $\mathcal{V}$ and $\mathcal{V}'$ denote the node sets before and after the permutation, respectively. The subtree permutation induces a mapping $\sigma : \mathcal{V} \to \mathcal{V}'$ that associates each node with its corresponding node in the permuted tree. Since the inverse subtree permutation induces the inverse mapping $\sigma^{-1} : \mathcal{V}' \to \mathcal{V}$, the correspondence is one-to-one, and hence $\sigma$ is bijective. This construction is independent of the sizes of the permuted subtrees.

Motivated by this symmetry, we propose MSP-GNN, which represents the scenario tree as a graph and processes it through message passing. As detailed in Sections 4.1 and 4.2, the architecture is constructed to preserve the recursive subtree symmetry established in Proposition 1. In particular, permutation-equivariant message passing ensures that node representations transform consistently under subtree permutations, while the root-anchored readout renders the predicted first-stage decision invariant to these permutations. See Appendix C for the proof of Proposition 1.

### 4.1 Graph Representation and Structural Alignment

We translate the discrete scenario tree $\mathcal{T}_s = (\mathcal{N}, \mathcal{E})$ into a computational graph $G = (V, E)$ through three design choices, each directly motivated by the structure of the MSP.

**Node-vertex correspondence.** We establish a bijection $\mathcal{N} \leftrightarrow V$, so that each scenario tree node corresponds to exactly one graph vertex. Through this bijection, each vertex $v \in V$ inherits the uncertainty realization $\xi_v := \xi_n$ as its node feature, with $V_t := \mathcal{N}_t$ for each stage $t \in \mathcal{T}$. The root node $n_0 \in \mathcal{N}_1$ is mapped to the root vertex $v_{\text{root}} := n_0$.

**Feature initialization.** For equiprobable scenario trees, we initialize each vertex $v \in V$ using its uncertainty realization,

$$h_v^{(0)} = \xi_v. \tag{11}$$

Because the leaf scenarios have equal probability, the corresponding internal-node probabilities are determined by the tree topology and need not be provided explicitly. More generally, non-equiprobable scenario trees can be handled by augmenting the node feature with its probability, $h_v^{(0)} = (\xi_v, p_v)$. Since $(\xi_v, p_v)$ is permuted together with node $v$, this augmentation preserves the subtree-permutation symmetry underlying MSP-GNN.

**Edge construction.** We connect each parent–child pair in the scenario tree with an undirected edge $(u, v) \in E$. Although the edges are undirected, the temporal hierarchy is preserved by the rooted-tree structure: $v_{\text{root}}$ is fixed, and the stage of each vertex is determined by its graph distance from the root. Hence, for any adjacent pair, the parent is uniquely identified as the vertex closer to $v_{\text{root}}$, so parent and child roles are not interchangeable. With $L \geq T - 1$ message-passing layers, information from every node can reach the root before the readout, as formalized in Section 4.2. In contrast, permuting complete sibling subtrees preserves the root, parent–child relations, and node stages, and therefore leaves the represented MSP unchanged.

### 4.2 Message Passing and Decision Readout

Message passing propagates uncertainty information across the scenario tree through successive rounds of neighborhood aggregation, so that after $L \geq T - 1$ layers the root node embedding captures the full structure of the scenario tree and is decoded into the optimal first-stage decision.

**Message passing.** Let $h_v^{(l)} \in \mathbb{R}^{d_l}$ denote the latent representation of node $v$ at layer $l$, initialized as $h_v^{(0)} = \xi_v$ per Equation (11). At each layer $l \in \{1, \ldots, L\}$, every node $v$ first aggregates the latent representations of its immediate neighbors $\mathrm{Nbr}(v)$ into a message vector, and then updates its own representation by combining this message with its current state. Formally:

$$m_v^{(l)} = \bigoplus_{u \in \mathrm{Nbr}(v)} \psi^{(l)}\left(h_v^{(l-1)}, h_u^{(l-1)}\right), \tag{12}$$

$$h_v^{(l)} = \varphi^{(l)}\left(h_v^{(l-1)}, m_v^{(l)}\right), \tag{13}$$

where $\mathrm{Nbr}(v) := \{u \in V : (u, v) \in E\}$ denotes the set of vertices adjacent to $v$ in the computational graph $G$, and $m_v^{(l)}$ summarizes the local neighborhood structure of node $v$ at layer $l$. [1] The message function $\psi^{(l)}$, parameterized by an MLP, extracts pairwise relational features between adjacent nodes. The update function $\varphi^{(l)}$, also parameterized by an MLP, non-linearly fuses the aggregated message $m_v^{(l)}$ with the node's prior representation $h_v^{(l-1)}$ to produce the refined embedding $h_v^{(l)}$.

In our design, the aggregation operator $\bigoplus$ is instantiated as the element-wise sum:

$$\bigoplus_{u \in \mathrm{Nbr}(v)} \psi^{(l)}\left(h_v^{(l-1)}, h_u^{(l-1)}\right) = \sum_{u \in \mathrm{Nbr}(v)} \psi^{(l)}\left(h_v^{(l-1)}, h_u^{(l-1)}\right). \tag{14}$$

**Depth condition.** A key architectural requirement governs the number of message-passing layers $L$. At layer $l$, the representation $h_v^{(l)}$ of any node $v$ aggregates information from all nodes within graph distance $l$ of $v$. Consequently, for the root node $v_{\text{root}}$ to receive information from every leaf node in

[1] We note that bidirectional message passing does not violate non-anticipativity, which requires decisions associated with scenarios sharing the same uncertainty history up to stage $t$ to be identical. In our setting, Eq. (16) produces a single first-stage decision $\hat{x}_1$ from the root representation for the entire scenario tree. Hence, all scenarios sharing the root history receive the same first-stage decision, regardless of the bidirectional information aggregation used within the GNN.

$V_T$, which are at graph distance $T-1$ from the root in the computational graph $G$, it is necessary and sufficient that:

$$L \geq T - 1. \tag{15}$$

Under this condition, the receptive field of $v_{\text{root}}$ spans the entire scenario tree after $L$ layers, and the final root embedding $h_{v_{\text{root}}}^{(L)}$ encodes the complete future uncertainty distribution of $\mathcal{T}_s$.

**Decision readout.** Since $h_{v_{\text{root}}}^{(L)}$ aggregates information from every node in the scenario tree, we decode the first-stage decision exclusively from the root node embedding. To ensure feasibility with respect to the first-stage decision set $\mathcal{X}_1$, the raw output of the readout MLP is projected onto $\mathcal{X}_1$:

$$\hat{x}_1 = \Pi_{\mathcal{X}_1}\left(\rho\left(h_{v_{\text{root}}}^{(L)}\right)\right), \tag{16}$$

where $\rho : \mathbb{R}^{d_L} \to \mathbb{R}^{d_x}$ is a readout MLP producing an unconstrained output in $\mathbb{R}^{d_x}$, and $\Pi_{\mathcal{X}_1} : \mathbb{R}^{d_x} \to \mathcal{X}_1$ denotes the Euclidean projection onto the feasible first-stage decision set $\mathcal{X}_1$, guaranteeing that $\hat{x}_1 \in \mathcal{X}_1$ by construction.

## 5 Theoretical Expressivity and Universal Approximation

While MSP-GNN gains structural efficiency by enforcing subtree-permutation invariance, this architectural constraint may in principle restrict the class of functions it can represent. We therefore establish a universal approximation guarantee, showing that MSP-GNN remains sufficiently expressive to approximate the optimal first-stage decision mapping over the considered scenario-tree domain. We establish the universal approximation guarantee of MSP-GNN in four steps: (i) introduce the TWL test to characterize distinguishability among scenario trees; (ii) show that TWL-indistinguishable trees share identical optimal first-stage decisions (Lemma 1); (iii) prove that our GNN simulates the TWL test (Lemma 2); and (iv) use these results to establish the universal approximation property of MSP-GNN (Theorem 1).

### 5.1 The Tree-Weisfeiler-Lehman Framework

Let $G = (V, E)$ be the computational graph of a scenario tree as defined in Section 4.1. The Tree-Weisfeiler-Lehman (TWL) test iteratively assigns a discrete color to each node to summarize its local structural and feature environment. We initialize the coloring function $c^{(0)} : V \to \Sigma$ by hashing each node's uncertainty realization:

$$c_v^{(0)} = \text{HASH}(\xi_v), \tag{17}$$

which is consistent with the architectural initialization $h_v^{(0)} = \xi_v$ of Equation (11), ensuring that the TWL test operates on precisely the same input information as the GNN. At each iteration $l \geq 1$, the color of node $v$ is updated by aggregating the colors of its immediate neighborhood:

$$c_v^{(l)} = \text{HASH}\left(c_v^{(l-1)}, \left\{\!\left\{ c_u^{(l-1)} \mid u \in \text{Nbr}(v) \right\}\!\right\}\right), \tag{18}$$

where $\{\!\{\cdot\}\!\}$ denotes a multiset and HASH is a strictly injective function that uniquely encodes every distinct combination of a node's color and its neighbors' color multiset. [2]

Because the computational graph is a tree of depth $T$, the coloring stabilizes after $L \geq T-1$ iterations. At this point, the root color $c_{v_{\text{root}}}^{(L)}$ is a complete, injective summary of the entire scenario tree's topology and uncertainty realizations. We define two computational graphs $G_A$ and $G_B$ as *TWL-indistinguishable*, written $G_A \simeq_{\text{TWL}} G_B$, if and only if their root colors coincide after stabilization:

$$c_{v_{\text{root}},A}^{(L)} = c_{v_{\text{root}},B}^{(L)}. \tag{19}$$

TWL-indistinguishability formalizes the notion that two scenario trees present identical aggregated uncertainty environments from the perspective of the root node, and therefore should yield identical optimal first-stage decisions, a claim we formalize in the following section.

[2] No additional assumption is needed. Under the compact feature domain $\Xi \subset \mathbb{R}^d$ and bounded branching factor $B$ in Theorem 1, the Deep Sets representation result (Zaheer et al., 2017) provides a continuous map $\phi$ for which $M \mapsto \sum_{z \in M} \phi(z)$ is injective over multisets of size at most $B$. Hence, $\text{HASH}(a, M) := \left(a, \sum_{z \in M} \phi(z)\right)$ is injective: equality of two outputs implies $a = a'$ and $M = M'$.

### 5.2 Target Equivalence and GNN Simulation

Having defined the TWL benchmark, we now establish two lemmas that together certify the expressive power of our architecture. The first shows that TWL captures the equivalence structure of the MSP optimization problem; the second shows that our GNN attains this level of discrimination exactly.

**Lemma 1** (Target Equivalence)**.** *Let $G_A$ and $G_B$ be two computational graphs representing discrete scenario trees. If $G_A \simeq_{TWL} G_B$, then their corresponding optimal first-stage decisions are identical:*

$$x_1^*(G_A) = x_1^*(G_B). \tag{20}$$

Lemma 1, proved in Appendix D.1, establishes that TWL does not over-discriminate: any two scenario trees that are indistinguishable under TWL are guaranteed to share the same optimal first-stage decision. In other words, TWL-indistinguishability is a *sufficient* condition for target equivalence, though not necessarily a necessary one, as two trees may share the same optimal first-stage decision without being TWL-indistinguishable. The converse does not hold in general; a counterexample is provided in Appendix D.2. Lemma 2 then shows that MSP-GNN attains the discriminative power of the TWL test.

**Lemma 2** (GNN Simulation of the TWL Test)**.** *Let $L \geq T - 1$, where $T$ is the depth of the scenario tree. There exist neural network parameters for the message function $\psi^{(l)}$, the update function $\varphi^{(l)}$, and the readout mapping $\rho$ such that the proposed GNN approximately simulates the $L$-step TWL test. That is, for any two computational graphs $G_A$ and $G_B$:*

$$c^{(L)}_{v_{\text{root}},A} \neq c^{(L)}_{v_{\text{root}},B} \implies \left\| h^{(L)}_{v_{\text{root}},A} - h^{(L)}_{v_{\text{root}},B} \right\| > \delta, \tag{21}$$

*for some separation margin $\delta > 0$ depending on the graphs $G_A$ and $G_B$, and for any $\varepsilon > 0$, the readout $\rho(h^{(L)}_{v_{\text{root}}})$ can be parameterized to approximate any target mapping consistent with this separation to within $\varepsilon$.*

Together, Lemmas 1 and 2 establish that MSP-GNN never assigns different predictions to TWL-indistinguishable trees, achieving the discriminative power of the TWL test. Note that TWL-indistinguishability is sufficient but not necessary for target equivalence, an inherent limitation of WL-based frameworks that does not affect Theorem 1. See Appendix E for the proof of Lemma 2.

### 5.3 Universal Approximation Theorem

**Theorem 1** (Universal Approximation of the MSP-GNN)**.** *Let $\mathcal{G}$ be a compact domain of discrete scenario trees with bounded depth $T$, bounded branching factor $B$, and node features $\xi \in \Xi$ where $\Xi \subset \mathbb{R}^d$ is compact. Assume the multistage stochastic program admits a unique optimal first-stage solution for every $G \in \mathcal{G}$, and the optimal decision mapping $x_1^* : \mathcal{G} \to \mathbb{R}^{d_x}$ is continuous on $\mathcal{G}$. Then for any $\varepsilon > 0$, there exists a GNN $f_\theta$ parameterized by the message-passing framework of Section 4.2 such that:*

$$\sup_{G \in \mathcal{G}} \| f_\theta(G) - x_1^*(G) \| < \varepsilon. \tag{22}$$

Theorem 1, proved in Appendix F, establishes the expressive power of MSP-GNN by showing that the proposed architecture can universally approximate the optimal first-stage decision mapping over the stated compact domain.[3] The results also apply to non-equiprobable scenario trees using the augmented node features $(\xi_v, p_v)$ described in Section 4.1.

## 6 Numerical Experiments

We evaluate MSP-GNN against FCNN, DeepSets (Zaheer et al., 2017), and a bottom-up Tree-GNN baseline (Thost & Chen, 2021) on predictive accuracy and parameter efficiency across five scenario-tree configurations obtained by scaling $T$, $b$, and $A$ (Section 6.1), and further assess the scalability of these models on a substantially larger scenario tree with $4{,}681$ nodes (Appendix G.1.5). We additionally evaluate MSP-GNN on a weighted stochastic LQR problem with an unbalanced

[3] This result characterizes the representational capacity of the architecture and does not address training convergence, finite-sample generalization, decision regret, or optimality gaps.

Table 2: Multistage asset-allocation performance (mean ± standard deviation over three seeds).

| Configuration | Method | Test MSE ($\times 10^{-3}$) | Parameters ($\times 10^{3}$) | Inference (ms) |
|---|---|---|---|---|
| Baseline $T = 3, b = 2, A = 5$ | FCNN | $15.209 \pm 1.132$ | $\mathbf{6.8}$ | $\mathbf{0.66}$ |
| | DeepSets | $116.226 \pm 2.931$ | 7.1 | 1.50 |
| | Tree-GNN | $\mathbf{6.613 \pm 0.769}$ | $\mathbf{6.8}$ | 1.49 |
| | **MSP-GNN** | $6.695 \pm 1.017$ | $\mathbf{6.8}$ | 1.14 |
| Scale $T$ $T = 4, b = 2, A = 5$ | FCNN | $14.026 \pm 1.002$ | 13.5 | $\mathbf{0.68}$ |
| | DeepSets | $108.567 \pm 4.705$ | 13.5 | 1.73 |
| | Tree-GNN | $5.958 \pm 0.769$ | $\mathbf{10.0}$ | 1.90 |
| | **MSP-GNN** | $\mathbf{5.500 \pm 1.282}$ | $\mathbf{10.0}$ | 1.35 |
| Scale $b$ $T = 3, b = 3, A = 5$ | FCNN | $19.642 \pm 1.481$ | 8.7 | $\mathbf{0.66}$ |
| | DeepSets | $126.919 \pm 1.837$ | 7.1 | 1.15 |
| | Tree-GNN | $12.328 \pm 4.694$ | $\mathbf{6.8}$ | 1.89 |
| | **MSP-GNN** | $\mathbf{10.277 \pm 1.212}$ | $\mathbf{6.8}$ | 1.15 |
| Scale $A$ $T = 3, b = 2, A = 10$ | FCNN | $23.962 \pm 1.215$ | 9.4 | $\mathbf{0.66}$ |
| | DeepSets | $83.739 \pm 0.905$ | 7.7 | 1.62 |
| | Tree-GNN | $\mathbf{7.035 \pm 0.152}$ | $\mathbf{7.2}$ | 1.40 |
| | **MSP-GNN** | $7.042 \pm 0.264$ | $\mathbf{7.2}$ | 1.14 |
| Scale all $T = 4, b = 3, A = 10$ | FCNN | $76.386 \pm 2.498$ | 34.6 | $\mathbf{0.68}$ |
| | DeepSets | $98.268 \pm 1.069$ | 14.0 | 1.55 |
| | Tree-GNN | $23.202 \pm 2.228$ | $\mathbf{10.4}$ | 2.08 |
| | **MSP-GNN** | $\mathbf{15.604 \pm 1.489}$ | $\mathbf{10.4}$ | 1.36 |

scenario tree and non-uniform branch probabilities (Appendix G.2). We validate the key architectural design choices via ablation (Appendix G.1.6). Problem formulation and hyperparameter settings are provided in Appendix G.

### 6.1 Comparative Evaluation across Problem Scales

Table 2 reports test MSE, parameter count, and inference time across five configurations. Figure 5 in Appendix G.1.4 presents the corresponding test MSE learning curves. MSP-GNN achieves the lowest test MSE in three out of five configurations, with its advantage becoming more pronounced as the scenario tree grows. In Scale all ($T = 4, b = 3, A = 10$), MSP-GNN reduces the test MSE by $4.9\times$ relative to FCNN ($15.60$ vs. $76.39 \times 10^{-3}$). DeepSets performs substantially worse across all configurations, with test MSE ranging from $6.3\times$ to $19.7\times$ that of MSP-GNN, highlighting the importance of preserving the parent–child structure beyond permutation invariance alone. On parameter efficiency, FCNN's parameter count grows $5.1\times$ from baseline to Scale all ($6.8 \rightarrow 34.6 \times 10^3$), while MSP-GNN's grows only $1.5\times$ ($6.8 \rightarrow 10.4 \times 10^3$), demonstrating substantially better parameter scaling with scenario-tree size. All methods achieve millisecond-level inference; MSP-GNN also exhibits faster and more stable convergence than FCNN, particularly in the Scale $T$ and Scale $b$ settings.

## 7 Conclusion and Limitations

We introduce MSP-GNN, an end-to-end framework for predicting first-stage decisions directly from MSP scenario trees. It exploits recursive subtree symmetry to avoid the poor scaling of conventional fully connected architectures, while preserving subtree-permutation invariance by construction through graph message passing and a root-anchored readout. We establish a universal approximation guarantee for the optimal first-stage decision mapping via a Tree-Weisfeiler-Lehman analysis, and show empirically that MSP-GNN achieves higher accuracy with substantially fewer parameters while retaining millisecond-level inference.

Several limitations remain and motivate future work. Our theory does not quantify network size, training convergence, or finite-sample generalization. Moreover, projection onto the feasible region is inexpensive for simple feasible sets, but may become costly for complex or non-convex regions. Combining MSP-GNN with recent low-complexity projection methods that provide theoretical feasibility guarantees is an interesting direction for future work.

## AI USE DISCLOSURE

We used generative AI tools to polish the language of the manuscript and improve its clarity and readability. The authors take full responsibility for the final content of the paper.

## REFERENCES


Aayushya Agarwal, Priya L Donti, J Zico Kolter, and Larry Pileggi. Employing adversarial robustness techniques for large-scale stochastic optimal power flow. *Electric Power Systems Research*, 212:108497, 2022.

Shabbir Ahmed, Alan J King, and Gyana Parija. A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 26(1):3–24, 2003.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pp. 136–145. PMLR, 2017.

Hyunglip Bae, Jinkyu Lee, Woo Chang Kim, and Yongjae Lee. Deep value function networks for large-scale multistage stochastic programs. In *International Conference on Artificial Intelligence and Statistics*, pp. 11267–11287. PMLR, 2023.

Kyri Baker. Learning warm-start points for ac optimal power flow. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE, 2019.

Yoshua Bengio, Emma Frejinger, Andrea Lodi, Rahul Patel, and Sriram Sankaranarayanan. A learning-based algorithm to quickly compute good primal solutions for stochastic integer programs. In *International conference on integration of constraint programming, artificial intelligence, and operations research*, pp. 99–111. Springer, 2020.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

John R Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations research*, 33(5):989–1007, 1985.

John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer, 1997.

David R Carino and William T Ziemba. Formulation of the russell-yasuda kasai financial planning model. *Operations research*, 46(4):433–449, 1998.

Yize Chen, Yishen Wang, Daniel Kirschen, and Baosen Zhang. Model-free renewable scenario generation using generative adversarial networks. *IEEE Transactions on Power Systems*, 33(3):3265–3275, 2018.

Antonio J Conejo, Miguel Carrión, Juan M Morales, et al. *Decision making under uncertainty in electricity markets*, volume 1. Springer, 2010.

Hamed Davari-Ardakani, Majid Aminnayeri, and Abbas Seifi. Multistage portfolio optimization with stocks and options. *International Transactions in Operational Research*, 23(3):593–622, 2016.

Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Priya Donti, Brandon Amos, and J Zico Kolter. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30, 2017.

Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. *arXiv preprint arXiv:2104.12225*, 2021.

Brendan L Douglas. The weisfeiler-lehman method and graph isomorphism testing. *arXiv preprint arXiv:1101.5211*, 2011.

Jitka Dupačová, Giorgio Consigli, and Stein W Wallace. Scenarios for multistage stochastic programs. *Annals of operations research*, 100(1):25–53, 2000.

Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68(1):9–26, 2022.

Ferdinando Fioretto, Terrence WK Mak, and Pascal Van Hentenryck. Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 630–637, 2020.

Saeed Ghadimi, Raymond T Perkins, and Warren B Powell. Reinforcement learning via parametric cost function approximation for multistage stochastic programming. *arXiv preprint arXiv:2001.00831*, 2020.

Holger Heitsch and Werner Römisch. Scenario tree modeling for multistage stochastic programs. *Mathematical Programming*, 118(2):371–406, 2009.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Julien Keutchayan, Janosch Ortmann, and Walter Rei. Problem-driven scenario clustering in stochastic optimization: J. keutchayan et al. *Computational Management Science*, 20(1):13, 2023.

Chanyeong Kim, Jongwoong Park, Hyunglip Bae, and Woo Chang Kim. Transformer-based stagewise decomposition for large-scale multistage stochastic optimization. 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lingkai Kong, Jiaming Cui, Yuchen Zhuang, Rui Feng, B Aditya Prakash, and Chao Zhang. End-to-end stochastic optimization with energy-based model. *Advances in Neural Information Processing Systems*, 35: 11341–11354, 2022.

James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*, 2021.

Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

Meiyi Li, Soheil Kolouri, and Javad Mohammadi. Learning to solve optimization problems with hard linear constraints. *IEEE Access*, 11:59995–60004, 2023.

Enming Liang and Minghua Chen. Efficient bisection projection to ensure neural-network solution feasibility for optimization over general set. In *ICML*, 2025.

Enming Liang, Minghua Chen, and Steven Low. Low complexity homeomorphic projection to ensure neural-network solution feasibility for optimization over (non-) convex set. 2023.

Andrea Lodi and Giulia Zarpellon. On learning and branching: a survey. *Top*, 25(2):207–236, 2017.

Sidhant Misra, Line Roald, and Yeesian Ng. Learning for constrained optimization: Identifying optimal active constraint sets. *INFORMS Journal on Computing*, 34(1):463–480, 2022.

Vinod Nair, Dj Dvijotham, Iain Dunning, and Oriol Vinyals. Learning fast optimizers for contextual stochastic integer programs. In *UAI*, pp. 591–600, 2018.

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.

Xiang Pan, Minghua Chen, Tianyu Zhao, and Steven H Low. Deepopf: A feasibility-optimized deep neural network approach for ac optimal power flow problems. *IEEE Systems Journal*, 17(1):673–683, 2022.

Rahul Mihir Patel, Justin Dumouchelle, Elias Khalil, and Merve Bodur. Neur2sp: Neural two-stage stochastic programming. *Advances in neural information processing systems*, 35:23992–24005, 2022.

Mario VF Pereira and Leontina MVG Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1):359–375, 1991.

Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

R Tyrrell Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2: 21–42, 2000.

Yves Rychener, Daniel Kuhn, and Tobias Sutter. End-to-end learning for stochastic optimization: A bayesian perspective. In *International Conference on Machine Learning*, pp. 29455–29472. PMLR, 2023.

Tjendera Santoso, Shabbir Ahmed, Marc Goetschalckx, and Alexander Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167 (1):96–115, 2005.

Sanket Shah, Meghna Lowalekar, and Pradeep Varakantham. Neural approximate dynamic programming for on-demand ride-pooling. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 507–515, 2020.

Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on stochastic programming: modeling and theory*. SIAM, 2021.

Murwan Siddig, Yongjia Song, and Amin Khademi. Rolling horizon policies in multistage stochastic programming. *arXiv preprint arXiv:2102.04874*, 2021.

Jingrui Sun and Jiongmin Yong. *Stochastic Linear-Quadratic Optimal Control Theory: Open-Loop and Closed-Loop Solutions*. SpringerBriefs in Mathematics. Springer, Cham, 2020. doi: 10.1007/978-3-030-20922-3.

Daniel Tabas and Baosen Zhang. Computationally efficient safe reinforcement learning for power systems. In *2022 American Control Conference (ACC)*, pp. 3303–3310. IEEE, 2022.

Veronika Thost and Jie Chen. Directed acyclic graph neural networks. *arXiv preprint arXiv:2101.07965*, 2021.

Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 1658–1665, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Ling Zhang, Daniel Tabas, and Baosen Zhang. An efficient learning-based solver for two-stage dc optimal power flow with feasibility guarantees. *arXiv preprint arXiv:2304.01409*, 2023.

Tianyu Zhao, Xiang Pan, Minghua Chen, and Steven Low. Ensuring dnn solution feasibility for optimization problems with linear constraints. In *The Eleventh International Conference on Learning Representations*, 2023.

Min Zhou, Minghua Chen, and Steven H Low. Deepopf-ft: One deep neural network for multiple ac-opf problems with flexible topology. *IEEE Transactions on Power Systems*, 38(1):964–967, 2022.

Min Zhou, Enming Liang, Minghua Chen, and Steven H Low. Partially permutation-invariant neural network for solving two-stage stochastic ac-opf problem. *IEEE Transactions on Power Systems*, 2025.

# APPENDICES

## A Machine Learning for Constrained Optimization

Leveraging machine learning to solve constrained optimization problems has become an active research direction (Donti et al., 2021; Pan et al., 2022; Li et al., 2023; Pan et al., 2020; Fioretto et al., 2020), driven by the need for real-time decision-making in applications where conventional iterative solvers are too computationally expensive. Existing methods broadly fall into two categories: hybrid methods that integrate machine learning components into iterative solvers, and end-to-end methods that directly learn the mapping from problem parameters to solutions, thereby avoiding iterative optimization at inference time.

**Hybrid methods.** Hybrid methods use machine learning as a building block for iterative solvers to improve computational efficiency. Representative approaches include predicting active constraint sets to reduce problem dimensionality (Misra et al., 2022), generating warm-start solutions to accelerate convergence (Baker, 2019), and learning update rules to replace hand-crafted iterative steps (Li & Malik, 2016). These approaches can reduce the computational effort required by iterative optimization. However, they still rely on repeated solver iterations at inference time and may remain computationally intensive for large-scale MSP problems, limiting their applicability to time-sensitive settings.

**End-to-end methods.** End-to-end methods directly learn the mapping from problem parameters to optimal decisions, avoiding the repeated iterative optimization required by conventional solvers at inference time. However, a key challenge is ensuring solution feasibility in the presence of neural network approximation errors. For equality constraints, predict-and-reconstruct methods predict a subset of independent variables and recover the remaining variables through the equality relations (Donti et al., 2021; Pan et al., 2022; Li et al., 2023). For inequality constraints, existing approaches either incorporate constraint-violation penalties into the training loss (Pan et al., 2020; Fioretto et al., 2020; Zhao et al., 2023), or apply feasibility-preserving mappings and projections, including gauge mapping (Zhang et al., 2023; Tabas & Zhang, 2022), homeomorphic projection (Liang et al., 2023), and bisection projection (Liang & Chen, 2025). These methods enable low-complexity inference while substantially improving the feasibility of neural network solutions.

However, existing end-to-end methods are largely designed for fixed-dimensional inputs. In MSP, uncertainty is represented by a scenario tree whose number of nodes can grow exponentially with the planning horizon. Existing architectures typically flatten all scenario-tree nodes into a single input vector (Donti et al., 2017), causing the input dimension and consequently the model size to increase rapidly with the tree size. This limits their scalability to large multistage problems and motivates architectures that directly exploit the hierarchical structure of the scenario tree.

## B Comparison of Input Representations

We illustrate the difference between the vectorized input adopted by conventional FCNNs and the graph-structured input used by MSP-GNN. Consider a balanced scenario tree with $T$ stages, branching factor $b$, and node feature dimension $d_\xi$. The total number of nodes is

$$|\mathcal{N}| = 1 + b + \cdots + b^{T-1} = \frac{b^T - 1}{b - 1}.$$

As illustrated in Fig. 4, the flattened FCNN input has dimension

$$d_{\mathrm{FCNN}} = d_\xi |\mathcal{N}| = d_\xi \frac{b^T - 1}{b - 1},$$

which grows exponentially with $T$ for a fixed branching factor $b > 1$. In comparison, MSP-GNN operates directly on the scenario tree, where the feature dimension of each node remains $d_\xi$ regardless of the tree size. The graph structure is retained explicitly, while the same message and update functions are shared across all nodes. This avoids converting the growth of the scenario tree into a corresponding growth of the input dimension of a fully connected network.

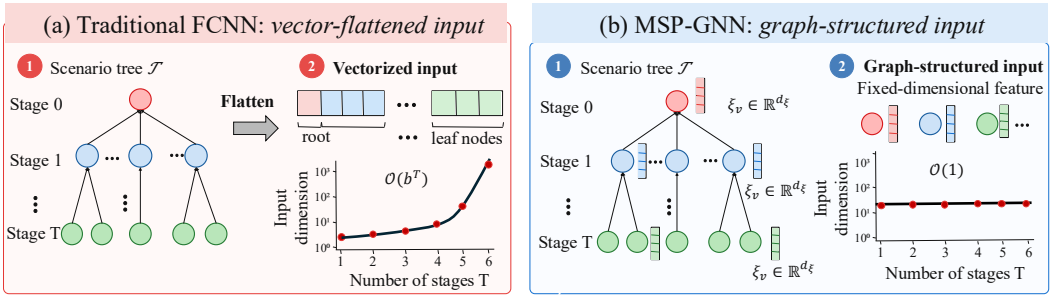


Figure 4: Comparison of the input representations for conventional FCNNs and MSP-GNN. The FCNN concatenates all node features into a single vector of dimension $d_\xi|\mathcal{N}|$, which increases rapidly with the number of stages. In contrast, MSP-GNN preserves the scenario-tree structure and associates each node with a fixed-dimensional feature $\xi_v \in \mathbb{R}^{d_\xi}$.

## C Proof of Proposition 1

**Proposition 1.** Assume the associated MSP problem admits a unique optimal first-stage decision. Let $\mathcal{T}_s = (\mathcal{N}, \mathcal{E})$ be a discrete scenario tree and let $x_1^*(\mathcal{T}_s)$ denote the unique optimal first-stage decision. Let $\sigma : \mathcal{N} \to \mathcal{N}$ be a bijection that (i) fixes the root $\sigma(n_0) = n_0$; (ii) preserves stage indices $t(\sigma(n)) = t(n)$; and (iii) relocates each node $n$ together with its uncertainty realization $\xi_n$ and probability $p_n$ to position $\sigma(n)$, preserving the parent-child structure: $c \in \mathcal{C}(n) \implies \sigma(c) \in \mathcal{C}(\sigma(n))$. Then:

$$x_1^*(\sigma(\mathcal{T}_s)) = x_1^*(\mathcal{T}_s). \tag{23}$$

*Proof.* The proof proceeds in three steps: we first characterize the class of admissible permutations $\sigma$, then show that $\sigma$ leaves the deterministic equivalent problem invariant, and finally conclude that the optimal first-stage decision is unchanged.

**Step 1: Characterization of admissible permutations.** Let $\sigma : \mathcal{N} \to \mathcal{N}$ be a bijection satisfying conditions (i)–(iii) of Proposition 1. Condition (i) ensures that the root node $n_0$ is fixed under $\sigma$. Condition (ii) ensures that the stage index is preserved, so that $\sigma$ maps $\mathcal{N}_t$ to $\mathcal{N}_t$ for every $t \in \mathcal{T}$. Condition (iii) ensures that $\sigma$ preserves the parent-child structure of the tree: for every node $n \in \mathcal{N}$ and child $c \in \mathcal{C}(n)$, we have $\sigma(c) \in \mathcal{C}(\sigma(n))$, and the uncertainty realization and probability are preserved at every corresponding node, $\xi_{\sigma(n)} = \xi_n$ and $p_{\sigma(n)} = p_n$. Together, conditions (i)–(iii) imply that $\sigma$ permutes only branches that are statistically exchangeable: it reorders subtrees with identical uncertainty distributions without altering their internal structure or uncertainty realizations. In particular, condition (iii) implies that $\sigma$ preserves parent-child relationships globally:

$$a(\sigma(n)) = \sigma(a(n)), \quad \forall n \in \mathcal{N} \setminus \mathcal{N}_1,$$

which follows directly from the fact that $\sigma(c) \in \mathcal{C}(\sigma(n))$ if and only if $\sigma(n) = a(\sigma(c))$.

**Step 2: Invariance of the deterministic equivalent.** Recall the deterministic equivalent of the MSP formulated over the scenario tree $\mathcal{T}_s$:

$$\begin{aligned} J(\mathcal{T}_s) = \min_{\{x_n\}_{n\in\mathcal{N}}} \quad & \sum_{n\in\mathcal{N}} p_n f_{t(n)}(s_n, x_n, \xi_n) \\ \text{s.t.} \quad & s_n = g_{t(n)-1}(s_{a(n)}, x_{a(n)}, \xi_n), \quad \forall n \in \mathcal{N} \setminus \mathcal{N}_1, && (24) \\ & h_{t(n)}(s_n, x_n, \xi_n) \le 0, \quad \forall n \in \mathcal{N}, && (25) \\ & x_n \in \mathcal{X}_{t(n)}, \quad \forall n \in \mathcal{N}, && (26) \\ & s_n = s_1, \quad \forall n \in \mathcal{N}_1. \end{aligned}$$

We show that $J(\sigma(\mathcal{T}_s)) = J(\mathcal{T}_s)$. Consider any feasible decision sequence $\{x_n\}_{n\in\mathcal{N}}$ on $\mathcal{T}_s$. We construct a corresponding decision sequence on $\sigma(\mathcal{T}_s)$ by setting $x_{\sigma(n)} := x_n$ for all $n \in \mathcal{N}$, and verify that this mapping preserves feasibility and objective value under $\sigma$.

- **Probability preservation.** By condition (iii) of Proposition 1, the admissible permutation $\sigma$ explicitly relocates each node together with its probability. Therefore, node probabilities are strictly preserved under the mapping: $p_{\sigma(n)} = p_n$ for all $n \in \mathcal{N}$. This guarantees probability invariance regardless of whether the scenario tree leaves are equiprobable or carry arbitrary non-uniform distributions.

- **Objective preservation.** By conditions (ii) and (iii), $t(\sigma(n)) = t(n)$ and $\xi_{\sigma(n)} = \xi_n$ for all $n \in \mathcal{N}$. Combined with probability preservation and the bijectivity of $\sigma$:

$$\sum_{n\in\mathcal{N}} p_{\sigma(n)} f_{t(\sigma(n))}(s_{\sigma(n)}, x_{\sigma(n)}, \xi_{\sigma(n)}) = \sum_{n\in\mathcal{N}} p_n f_{t(n)}(s_n, x_n, \xi_n), \tag{27}$$

  where the equality follows from the bijectivity of $\sigma$ (the sum ranges over the same index set $\mathcal{N}$) and the substitutions $x_{\sigma(n)} = x_n$ and $\xi_{\sigma(n)} = \xi_n$.

- **Constraint preservation.** Since $\sigma$ preserves parent-child relationships ($a(\sigma(n)) = \sigma(a(n))$ for all $n \in \mathcal{N} \setminus \mathcal{N}_1$), uncertainty realizations ($\xi_{\sigma(n)} = \xi_n$), stage indices ($t(\sigma(n)) = t(n)$), and feasible decision sets ($x_n \in \mathcal{X}_{t(n)}$ implies $x_{\sigma(n)} \in \mathcal{X}_{t(\sigma(n))}$), any feasible state-decision trajectory $\{(s_n, x_n)\}_{n\in\mathcal{N}}$ on $\mathcal{T}_s$ induces a feasible state-decision trajectory $\{(s_{\sigma(n)}, x_{\sigma(n)})\}_{n\in\mathcal{N}}$ on $\sigma(\mathcal{T}_s)$, satisfying the dynamics (24), the feasibility constraints (25), and the decision set constraints (26) identically.

The above three points establish a bijection between the feasible sets of $\mathcal{T}_s$ and $\sigma(\mathcal{T}_s)$ that preserves the objective value pointwise. Consequently, the two problems are equivalent:

$$J(\sigma(\mathcal{T}_s)) = J(\mathcal{T}_s). \tag{28}$$

**Step 3: Invariance of the first-stage decision.** By Step 2, the optimal cost is identical across $\mathcal{T}_s$ and $\sigma(\mathcal{T}_s)$. It remains to show that the optimal first-stage decision is also preserved. By condition (i), $\sigma$ fixes the root node: $\sigma(n_0) = n_0$. The first-stage decision $x_1^*$ is anchored exclusively to the root node $n_0$, with the initial state $s_{n_0} = s_1$ identical for both $\mathcal{T}_s$ and $\sigma(\mathcal{T}_s)$. Since the bijection constructed in Step 2 maps $x_{n_0}$ to $x_{\sigma(n_0)} = x_{n_0}$, and since $x_{n_0} \in \mathcal{X}_1$ is preserved under this mapping, the unique optimal first-stage decision is unchanged:

$$x_1^*(\sigma(\mathcal{T}_s)) = x_1^*(\mathcal{T}_s). \tag{29}$$

This completes the proof. □

# D TWL Equivalence and Target Equivalence

## D.1 Proof of Lemma 1 (Target Equivalence)

**Lemma 1** Let $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ be two computational graphs representing discrete scenario trees. If $G_A$ and $G_B$ are TWL-indistinguishable ($G_A \simeq_{\mathrm{TWL}} G_B$), then their corresponding optimal first-stage decisions are identical:

$$x_1^*(G_A) = x_1^*(G_B).$$

*Proof.* Throughout this proof, we use the graph-side notation: for each computational graph $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$, the sets $V_{t,A}$ and $V_{t,B}$ denote vertices at stage $t$ in $G_A$ and $G_B$, respectively, $v_{\mathrm{root},A}$ and $v_{\mathrm{root},B}$ denote the respective root vertices, $a(v)$ denotes the parent, $\mathcal{C}(v)$ denotes children, and $p_v$ denotes node probability. We assume that HASH is a perfectly injective function over the continuous feature space $\Xi \subset \mathbb{R}^d$, effectively treating node features $\xi_v$ as exact real-valued labels. Under this idealization, the TWL coloring is a complete isomorphism invariant for rooted feature-labeled trees.

**Step 1: TWL-indistinguishability implies rooted tree isomorphism.** We claim that $G_A \simeq_{\mathrm{TWL}} G_B$ implies the existence of a bijective mapping $\varphi : V_A \to V_B$ satisfying: (i) $(u, v) \in E_A \iff (\varphi(u), \varphi(v)) \in E_B$; (ii) $\xi_{\varphi(v)} = \xi_v$; and (iii) $\varphi(v_{\mathrm{root},A}) = v_{\mathrm{root},B}$. Since the 1-dimensional WL test is a *complete* isomorphism invariant for trees (Douglas, 2011), and our TWL test initializes and

refines colors incorporating both topology and features $\xi_v$, $c^{(L)}_{v_{\text{root}},A} = c^{(L)}_{v_{\text{root}},B}$ guarantees that $G_A$ and $G_B$ are isomorphic as rooted, feature-labeled trees after $L \geq T-1$ iterations.

**Step 2: The isomorphism $\varphi$ is an admissible subtree permutation.** By property (iii), $\varphi$ fixes the root. Since $\varphi$ preserves graph distance from the root, it preserves stage indices: $t(\varphi(v)) = t(v)$, mapping $V_{t,A}$ to $V_{t,B}$. To preserve $E_A$, it must map children bijectively: $\varphi(\mathcal{C}(v)) = \mathcal{C}(\varphi(v))$. Combined with feature preservation ($\xi_{\varphi(v)} = \xi_v$), $\varphi$ permutes subtrees with identical realizations, satisfying all conditions of an admissible permutation $\sigma$ from Proposition 1.

**Step 3: Invariance of the optimal first-stage decision.** Consider the MSP over $G$: $J(G) = \min \sum_{v\in V} p_v f_{t(v)}(s_v, x_v, \xi_v)$ subject to $s_v = g_{t(v)-1}(s_{a(v)}, x_{a(v)}, \xi_v)$ for $v \in V \setminus V_1$, $h_{t(v)} \leq 0$, and $s_v = s_1$ for $v \in V_1$.

- *Probability preservation.* By construction, $p_v = 1/|V_{T,A}|$ for $v \in V_{T,A}$ and $p_v = \sum_{c\in\mathcal{C}(v)} p_c$. Since $\varphi$ maps $V_{T,A}$ to $V_{T,B}$ and preserves branching, $|V_{T,A}| = |V_{T,B}|$. By induction, $p_{\varphi(v)} = p_v$ for all $v \in V_A$.
- *Objective preservation.* Setting $x_{\varphi(v)} := x_v$, we have $p_{\varphi(v)} = p_v$, $t(\varphi(v)) = t(v)$, and $\xi_{\varphi(v)} = \xi_v$. The bijectivity of $\varphi$ ensures the sum over $V_B$ equals the sum over $V_A$ pointwise.
- *Constraint preservation.* Since $\varphi$ preserves parent-child relationships ($a(\varphi(v)) = \varphi(a(v))$) and **current node uncertainties** ($\xi_{\varphi(v)} = \xi_v$), any feasible trajectory $\{(s_v, x_v)\}_{v\in V_A}$ induces a feasible trajectory $\{(s_{\varphi(v)}, x_{\varphi(v)})\}_{v\in V_A}$ on $G_B$, satisfying the dynamics $s_{\varphi(v)} = g(s_{a(\varphi(v))}, x_{a(\varphi(v))}, \xi_{\varphi(v)})$ identically.

The bijection between feasible sets preserves the objective value, thus $J(G_A) = J(G_B)$. Since $\varphi$ fixes the root and initial state, $x^*_{v_{\text{root},A}} = x^*_{v_{\text{root},B}}$, yielding $x^*_1(G_A) = x^*_1(G_B)$. □

### D.2 TWL-Indistinguishability Is Sufficient but Not Necessary

Lemma 1 establishes that TWL-indistinguishability is sufficient for two scenario trees to share the same optimal first-stage decision. The converse, however, does not hold in general, as shown by the following proposition.

**Proposition 2** (Non-necessity of TWL-indistinguishability)**.** *There exist two scenario trees $\mathcal{G}_A$ and $\mathcal{G}_B$ such that*

$$x^\star_1(\mathcal{G}_A) = x^\star_1(\mathcal{G}_B),$$

*while*

$$\mathcal{G}_A \not\simeq_{\text{TWL}} \mathcal{G}_B.$$

*Proof.* Consider two two-stage scenario trees $\mathcal{G}_A$ and $\mathcal{G}_B$ with the same topology. Each tree consists of a root node $v_0$ and two leaf nodes $v_1$ and $v_2$. The root realization is fixed as $\xi_{v_0} = 0$, and the two leaf nodes are equiprobable:

$$p_{v_1} = p_{v_2} = \frac{1}{2}.$$

The leaf realizations are defined as

$$\mathcal{G}_A: \quad \xi_{v_1} = -1, \qquad \xi_{v_2} = 1,$$

and

$$\mathcal{G}_B: \quad \xi_{v_1} = -2, \qquad \xi_{v_2} = 2.$$

For either scenario tree $\mathcal{G}$, consider the stochastic optimization problem

$$x^\star_1(\mathcal{G}) = \arg\min_{x_1\in\mathbb{R}} \sum_{v\in\{v_1,v_2\}} p_v \left(x_1 - \xi_v\right)^2. \tag{30}$$

This is a special case of a two-stage stochastic program in which no second-stage recourse variable is required. Since the objective in (30) is strictly convex, the optimal first-stage decision is unique. Its first-order optimality condition gives

$$\sum_{v\in\{v_1,v_2\}} p_v \left(x^\star_1 - \xi_v\right) = 0,$$

and hence

$$x_1^\star(\mathcal{G}) = \sum_{v \in \{v_1, v_2\}} p_v \xi_v. \tag{31}$$

Therefore,

$$x_1^\star(\mathcal{G}_A) = \frac{1}{2}(-1) + \frac{1}{2}(1) = 0,$$

and similarly,

$$x_1^\star(\mathcal{G}_B) = \frac{1}{2}(-2) + \frac{1}{2}(2) = 0.$$

Thus,

$$x_1^\star(\mathcal{G}_A) = x_1^\star(\mathcal{G}_B). \tag{32}$$

It remains to show that the two trees are TWL-distinguishable. Recall that TWL initializes each node by

$$c_v^{(0)} = \mathrm{HASH}(\xi_v),$$

where HASH is injective. The multisets of initial leaf colors are therefore

$$\{\{\mathrm{HASH}(-1), \mathrm{HASH}(1)\}\}$$

for $\mathcal{G}_A$, and

$$\{\{\mathrm{HASH}(-2), \mathrm{HASH}(2)\}\}$$

for $\mathcal{G}_B$. These two multisets are different. Since the root realizations are identical and the TWL update is injective with respect to the root color and its neighbor-color multiset, after one TWL iteration,

$$c_{v_0,A}^{(1)} \neq c_{v_0,B}^{(1)}.$$

Hence,

$$\mathcal{G}_A \not\simeq_{\mathrm{TWL}} \mathcal{G}_B. \tag{33}$$

Combining (32) and (33) proves the result. □

This counterexample shows that two structurally different scenario trees can produce the same optimal first-stage decision. Therefore, target equivalence is generally coarser than TWL equivalence.

## E Proof of Lemma 2 (GNN Simulation of the TWL Test)

**Lemma 2.** *Let $L \geq T - 1$, where $T$ is the depth of the scenario tree. There exist neural network parameters for the message function $\psi^{(l)}$, the update function $\varphi^{(l)}$, and the readout mapping $\rho$ such that the proposed GNN approximately simulates the $L$-step TWL test. That is, for any two scenario trees $G_A$ and $G_B$ and any $\varepsilon > 0$:*

$$c_{v_{\mathrm{root}},A}^{(L)} \neq c_{v_{\mathrm{root}},B}^{(L)} \implies \left\| h_{v_{\mathrm{root}},A}^{(L)} - h_{v_{\mathrm{root}},B}^{(L)} \right\| > \delta, \tag{34}$$

*for some separation margin $\delta > 0$ depending on the trees $G_A$ and $G_B$, and the readout $\rho(h_{v_{\mathrm{root}}}^{(L)})$ can be parameterized to approximate any target mapping consistent with this separation to within $\varepsilon$.*

*Proof.* Throughout this proof, we use the graph-side notation introduced in Section 4.1, and we treat node features $\xi_v \in \Xi \subset \mathbb{R}^d$ as exact real-valued labels under the perfect hash assumption stated in the proof of Lemma 1. The proof proceeds in three steps. We first characterize the TWL update as an injective multiset function over the continuous feature space, then show that our GNN with sum aggregation can separate any two TWL-distinguishable inputs with a positive margin, and finally establish the approximate simulation claim by induction over the $L$ message-passing layers.

**Step 1: The TWL update as an injective multiset function over continuous features.**

At each iteration $l \geq 1$, the TWL test updates the color of node $v$ as follows:

$$c_v^{(l)} = \mathrm{HASH}\Big(c_v^{(l-1)},\ \Big\{\Big\{ c_u^{(l-1)} \mid u \in \mathrm{Nbr}(v) \Big\}\Big\}\Big), \tag{35}$$

where, under the perfect hash assumption, HASH is perfectly injective over the continuous feature space $\Xi \subset \mathbb{R}^d$: it assigns distinct colors to any two distinct input pairs $(c_v^{(l-1)}, \{\{c_u^{(l-1)}\}\})$, treating node features as exact real-valued labels. The TWL update is therefore an injective function defined over the continuous domain $\Xi \times \mathrm{Multisets}(\Xi)$. While this domain is uncountable when $\Xi \subset \mathbb{R}^d$ is continuous, the perfect hash assumption guarantees that any two distinct input pairs produce distinct colors, which is all that is required for the inductive simulation argument in Step 3. Crucially, for any fixed finite collection of scenario trees in $\mathcal{G}$, the set of distinct neighborhood feature multisets encountered is finite, and the injectivity of HASH over this finite collection suffices for the separation margin argument developed in Step 2.

**Step 2: GNNs with sum aggregation can separate TWL-distinguishable inputs with a positive margin.**

Our architecture updates node embeddings via:

$$m_v^{(l)} = \sum_{u \in \mathrm{Nbr}(v)} \psi^{(l)}\left(h_v^{(l-1)}, h_u^{(l-1)}\right), \tag{36}$$

$$h_v^{(l)} = \varphi^{(l)}\left(h_v^{(l-1)},\, m_v^{(l)}\right). \tag{37}$$

We invoke two results to show this architecture can separate any two TWL-distinguishable inputs with a positive margin.

First, by the universal approximation theorem for permutation-invariant functions over multisets (Zaheer et al., 2017), any continuous permutation-invariant function on a multiset can be decomposed into an element-wise transformation, a sum aggregation, and a final non-linear readout. The sum aggregation in Equation (36) realizes precisely this decomposition.

Second, by the structural expressivity theorem for sum-aggregation GNNs (Xu et al., 2018), sum aggregation is strictly more expressive than mean or max aggregation for distinguishing multisets of embeddings. For any two nodes $v$ and $w$ with distinct input pairs:

$$\left(h_v^{(l-1)}, \left\{\!\left\{h_u^{(l-1)} \mid u \in \mathrm{Nbr}(v)\right\}\!\right\}\right) \neq \left(h_w^{(l-1)}, \left\{\!\left\{h_u^{(l-1)} \mid u \in \mathrm{Nbr}(w)\right\}\!\right\}\right), \tag{38}$$

we show there exist parameterizations of $\psi^{(l)}$ and $\varphi^{(l)}$ such that the resulting embeddings are separated by a positive margin $\delta > 0$:

$$\left\| h_v^{(l)} - h_w^{(l)} \right\| > \delta. \tag{39}$$

To see this, note that the two distinct input pairs in (38) correspond to two distinct points in the compact domain $\mathbb{R}^{d_{l-1}} \times \mathrm{Multisets}(\mathbb{R}^{d_{l-1}})$. Since MLPs are universal approximators for continuous functions on compact domains (Hornik et al., 1989), there exists a parameterization of $\psi^{(l)}$ and $\varphi^{(l)}$ that maps these two distinct input pairs to outputs separated by at least $\delta > 0$. Formally, for any two distinct input pairs $(a_1, M_1) \neq (a_2, M_2)$ in a finite collection, the continuous function $\Phi : (a, M) \mapsto \varphi(a, \sum_{u \in M} \psi(a, u))$ satisfies $\|\Phi(a_1, M_1) - \Phi(a_2, M_2)\| > \delta$ for some $\delta > 0$ achievable by an MLP with sufficient capacity. Note that $\delta$ may depend on the specific pair of trees and the layer $l$, but remains strictly positive for any fixed finite collection of TWL-distinguishable inputs.

**Step 3: Inductive approximate simulation of the $L$-step TWL coloring.**

We prove by induction on $l$ that the GNN embeddings separate any two nodes with distinct TWL colors by a positive margin at every layer:

$$c_u^{(l)} \neq c_v^{(l)} \implies \left\| h_u^{(l)} - h_v^{(l)} \right\| > \delta_l, \quad \forall\, u, v \in V, \tag{40}$$

for some margin $\delta_l > 0$ that may depend on $l$ but remains strictly positive.

*Base case ($l = 0$).* The GNN is initialized as $h_v^{(0)} = \xi_v$ per Equation (11), and the TWL coloring is initialized as $c_v^{(0)} = \mathrm{HASH}(\xi_v)$ per Equation (17). Under the perfect hash assumption, $c_u^{(0)} \neq c_v^{(0)} \iff \xi_u \neq \xi_v \iff h_u^{(0)} \neq h_v^{(0)}$. Since $\xi_u$ and $\xi_v$ are distinct points in the compact set $\Xi \subset \mathbb{R}^d$, they are separated by a positive margin $\delta_0 = \|\xi_u - \xi_v\| > 0$. The base case (40) holds at $l = 0$.

*Inductive step.* Assume (40) holds at layer $l-1$ with margin $\delta_{l-1} > 0$, i.e., any two nodes with distinct TWL colors $c_u^{(l-1)} \neq c_v^{(l-1)}$ have GNN embeddings separated by at least $\delta_{l-1}$. We show it holds at layer $l$.

Consider any two nodes $v, w \in V$ with $c_v^{(l)} \neq c_w^{(l)}$. By the injectivity of the TWL hash (35), the full input pairs:

$$\left(c_v^{(l-1)}, \left\{\!\left\{ c_u^{(l-1)} \mid u \in \mathrm{Nbr}(v) \right\}\!\right\}\right) \neq \left(c_w^{(l-1)}, \left\{\!\left\{ c_u^{(l-1)} \mid u \in \mathrm{Nbr}(w) \right\}\!\right\}\right).$$

By the inductive hypothesis, the corresponding GNN embedding pairs satisfy:

$$\left(h_v^{(l-1)}, \left\{\!\left\{ h_u^{(l-1)} \mid u \in \mathrm{Nbr}(v) \right\}\!\right\}\right) \neq \left(h_w^{(l-1)}, \left\{\!\left\{ h_u^{(l-1)} \mid u \in \mathrm{Nbr}(w) \right\}\!\right\}\right),$$

with the separation inherited from the inductive hypothesis at layer $l-1$. Applying the margin separation result (39) from Step 2, the GNN update (36)–(37) produces embeddings separated by a positive margin $\delta_l > 0$:

$$\left\| h_v^{(l)} - h_w^{(l)} \right\| > \delta_l.$$

Hence (40) holds at layer $l$.

By induction, (40) holds for all $l = 0, \ldots, L$. In particular, after $L \geq T-1$ layers, the depth condition (15) guarantees that the receptive field of $v_{\mathrm{root}}$ spans the entire scenario tree. Any two TWL-distinguishable trees $G_A \not\simeq_{\mathrm{TWL}} G_B$ therefore produce root embeddings separated by a positive margin:

$$\left\| h_{v_{\mathrm{root}},A}^{(L)} - h_{v_{\mathrm{root}},B}^{(L)} \right\| > \delta_L > 0. \tag{41}$$

**Readout approximation.** It remains to show that the readout MLP $\rho$ can approximate the optimal first-stage decision mapping from the root embeddings. Since the root embeddings of TWL-distinguishable trees are separated by a positive margin $\delta_L > 0$, they correspond to distinct points in the compact set $\mathbb{R}^{d_L}$. The optimal first-stage decision mapping $x_1^*$ defines a target function on these embeddings: note that $\rho$ need not assign distinct outputs to distinct embeddings. Two TWL-distinguishable trees $G_A \not\simeq_{\mathrm{TWL}} G_B$ may still share the same optimal first-stage decision $x_1^*(G_A) = x_1^*(G_B)$, and $\rho$ is free to assign them the same output. What is required is only that $\rho$ correctly approximates $x_1^*$ as a function of the root embedding. Since $x_1^*$ factors through the TWL root color by Lemma 1 and the root embeddings encode the TWL colors with positive separation margin, $x_1^*$ is a well-defined continuous function on the compact image of the root embeddings in $\mathbb{R}^{d_L}$. By the MLP universal approximation theorem (Hornik et al., 1989), for any $\varepsilon > 0$ there exists a parameterization of $\rho : \mathbb{R}^{d_L} \to \mathcal{X}_1$ such that:

$$\sup_{G \in \mathcal{G}} \left\| \rho\left(h_{v_{\mathrm{root}}}^{(L)}\right) - x_1^*(G) \right\| < \varepsilon, \tag{42}$$

completing the proof. □

## F Proof of Theorem 1 (Universal Approximation)

**Theorem 1.** *Let $\mathcal{G}$ be a compact domain of discrete scenario trees with bounded depth $T$, bounded branching factor $B$, and node features $\xi \in \Xi$ where $\Xi \subset \mathbb{R}^d$ is compact. Assume the multistage stochastic program admits a unique optimal first-stage solution for every $G \in \mathcal{G}$, and the optimal decision mapping $x_1^* : \mathcal{G} \to \mathbb{R}^{d_x}$ is continuous on $\mathcal{G}$. Then for any $\varepsilon > 0$, there exists a GNN $f_\theta$ parameterized by the message-passing framework of Section 4.2 such that:*

$$\sup_{G \in \mathcal{G}} \| f_\theta(G) - x_1^*(G) \| < \varepsilon. \tag{43}$$

*Proof.* The proof proceeds by constructing the quotient space of scenario trees under TWL-indistinguishability, demonstrating that the GNN hypothesis space separates points on this quotient space, and establishing universal approximation via the homeomorphism between the quotient space and the embedding space.

**Step 1: Domain compactness and continuity.** Scenario trees in $\mathcal{G}$ have bounded depth $T$ and branching factor $B$, meaning the number of distinct rooted tree topologies is finite. For each fixed

topology $\tau$, the tree space is homeomorphic to $\Xi^{|V_\tau|}$, which is compact by Tychonoff's theorem. Since $\mathcal{G}$ is a finite union of such compact sets, $\mathcal{G}$ is compact. By assumption, the unique optimal decision mapping $x_1^* : \mathcal{G} \to \mathbb{R}^{d_x}$ is continuous on $\mathcal{G}$.

**Step 2: Quotient space construction.** Let $\simeq_{\mathrm{TWL}}$ denote the TWL-indistinguishability relation. Under the perfect hash assumption, $G_A \simeq_{\mathrm{TWL}} G_B$ if and only if they are isomorphic as rooted feature-labeled trees. Since $\mathcal{G}$ is compact and isomorphism reduces to finite equality conditions on features, $\simeq_{\mathrm{TWL}}$ is a closed equivalence relation. The quotient space $\widetilde{\mathcal{G}} = \mathcal{G}/\simeq_{\mathrm{TWL}}$ is therefore compact and Hausdorff. By **Lemma 1**, $x_1^*$ is invariant to TWL-indistinguishable graphs and thus descends to a well-defined continuous function $\widetilde{x}_1^* : \widetilde{\mathcal{G}} \to \mathbb{R}^{d_x}$ on the quotient space.

**Step 3: Point separation and homeomorphism.** Let $E_\theta : \mathcal{G} \to \mathbb{R}^{d_L}$ be the GNN root embedding. By **Lemma 2**, for $L \geq T-1$, there exists a parameterization such that $G_A \not\simeq_{\mathrm{TWL}} G_B \implies \|E_\theta(G_A) - E_\theta(G_B)\| > \delta > 0$. Combined with the symmetry invariance from Proposition 1, the embedding induces a continuous injection $\widetilde{E}_\theta : \widetilde{\mathcal{G}} \to \mathbb{R}^{d_L}$. Since $\widetilde{\mathcal{G}}$ is compact and $\mathbb{R}^{d_L}$ is Hausdorff, any continuous injection is a homeomorphism onto its image. Thus, $\widetilde{E}_\theta$ is a homeomorphism between $\widetilde{\mathcal{G}}$ and the compact set $\mathcal{Z} = \widetilde{E}_\theta(\widetilde{\mathcal{G}}) \subset \mathbb{R}^{d_L}$.

**Step 4: Uniform approximation.** Define the target function on the embedding space $g : \mathcal{Z} \to \mathbb{R}^{d_x}$ as $g(z) = \widetilde{x}_1^*(\widetilde{E}_\theta^{-1}(z))$. As a composition of continuous maps, $g$ is continuous on the compact set $\mathcal{Z}$. By the universal approximation theorem for MLPs, the readout network $\rho_\theta$ can approximate $g$ to within $\varepsilon$ on $\mathcal{Z}$. Substituting $z = E_\theta(G)$ and noting $g(E_\theta(G)) = x_1^*(G)$ yields:

$$\sup_{G \in \mathcal{G}} \|\rho_\theta(E_\theta(G)) - x_1^*(G)\| < \varepsilon,$$

which completes the proof. □

# G Experimental Setup

## G.1 Asset Allocation

### G.1.1 Problem Formulation

We evaluate MSP-GNN on a multistage asset allocation problem, in which an investor sequentially allocates wealth across $A$ assets over $T$ stages to maximize expected terminal wealth while controlling downside risk (Davari-Ardakani et al., 2016; Carino & Ziemba, 1998). The investor's decisions are *adaptive*: at each scenario tree node $n \in \mathcal{N}_t$, the portfolio weight vector $x_n = (x_{n,1}, \ldots, x_{n,A})^\top \in \mathbb{R}^A$ is selected after observing all asset returns along the unique root-to-$n$ path, i.e., $(\xi_{n_0}, \xi_{n_1}, \ldots, \xi_n)$ where $n_0, n_1, \ldots, n$ are the nodes along this path. Critically, two nodes $n, n' \in \mathcal{N}_t$ at the same stage $t$ may receive *different* portfolio decisions $x_n \neq x_{n'}$ if their realized return histories differ, encoding the full recourse structure of the MSP. The state variable is the portfolio wealth $W_n \in \mathbb{R}_{>0}$ at node $n$, initialized at $W_{n_0} = W_0 = 1$ (normalized initial wealth).

The deterministic equivalent of the asset allocation MSP over the scenario tree is naturally expressed using portfolio weight vectors $x_n$ as decisions, with the wealth transition $W_n = W_{a(n)} x_{a(n)}^\top \xi_n$. However, this transition constraint is bilinear in $W_{a(n)}$ and $x_{a(n)}$ when both are decision variables. Following the standard wealth-allocation reformulation (Carino & Ziemba, 1998), we apply the change of variables $y_n := W_n x_n \in \mathbb{R}_+^A$, where $y_n$ denotes the wealth invested in each asset at node $n$. Under this substitution, the bilinear transition becomes the linear constraint $W_n = \xi_n^\top y_{a(n)}$, and the budget constraint $\mathbf{1}^\top x_n = 1$ becomes $\mathbf{1}^\top y_n = W_n$. The resulting deterministic equivalent, with

decision variables $(y_n,\ W_n,\ \zeta,\ z_n)$, is a genuine LP:[4]

$$\min_{\{y_n\}_{n\in\mathcal{N}},\{W_n\}_{n\in\mathcal{N}},\zeta,\{z_n\}_{n\in\mathcal{N}_T}} \quad -\sum_{n\in\mathcal{N}_T} p_n W_n + \lambda\left(\zeta + \frac{1}{1-\alpha}\sum_{n\in\mathcal{N}_T} p_n z_n\right) \tag{44}$$

$$\text{s.t.} \quad W_n = \xi_n^\top y_{a(n)}, \quad \forall n \in \mathcal{N}\setminus\mathcal{N}_1, \tag{45}$$

$$\mathbf{1}^\top y_n = W_n, \quad y_n \geq 0, \quad \forall n \in \mathcal{N}, \tag{46}$$

$$z_n \geq -W_n - \zeta, \quad z_n \geq 0, \quad \forall n \in \mathcal{N}_T, \tag{47}$$

$$W_{n_0} = W_0, \tag{48}$$

where $\zeta \in \mathbb{R}$ is the Value-at-Risk (VaR) threshold auxiliary variable and the variables $z_n \geq 0$ are the per-leaf tail-loss auxiliaries introduced by the standard CVaR linearization (Rockafellar et al., 2000). Constraint (45) defines the linearized node-wise wealth dynamics: the wealth at node $n$ equals the inner product of the parent's dollar-investment vector $y_{a(n)}$ with the asset returns $\xi_n$ realized at the transition into $n$. Constraint (46) enforces the budget constraint ($\mathbf{1}^\top y_n = W_n$, i.e., total dollars invested equals current wealth) and the no-short-selling constraint ($y_n \geq 0$) at every node. Constraint (47) linearizes the CVaR term: combined with the minimization over $\zeta$, the term $\zeta + \frac{1}{1-\alpha}\sum_{n\in\mathcal{N}_T} p_n z_n$ is the standard finite-scenario reformulation of $\mathrm{CVaR}_\alpha(-W_T)$ (Rockafellar et al., 2000). The objective (44) balances expected terminal wealth against downside risk, with risk-aversion parameter $\lambda \geq 0$. Setting $\lambda = 0$ recovers pure expected wealth maximization.

### G.1.2 Scenario-Tree Data and Labels

All experiments use the multistage asset allocation MSP in Appendix G.1. Each sample is a balanced $b$-ary scenario tree with $T$ stages and $A$ assets. The total number of nodes is $|\mathcal{N}| = 1+b+\cdots+b^{T-1}$, and each node stores an $A$-dimensional gross-return feature vector.

For each instance, the return distribution parameters are sampled independently for each stage and asset as $\mu \sim \mathrm{Unif}(0.02, 0.08)$ and $\sigma \sim \mathrm{Unif}(0.10, 0.30)$. The log-normal return distribution is then discretized using $b$-point Gauss-Hermite quadrature to construct the child returns at every non-leaf node. Each dataset contains $N_{\text{train}} = 30{,}000$ training instances and $N_{\text{test}} = 1{,}000$ test instances.

The full dataset is generated once with data seed 0. We then use random seeds $\{0, 1, 2\}$ for train/test splitting, model initialization, and stochastic training, so all compared architectures are evaluated on the same fixed data distribution. The supervised label for each tree is the optimal first-stage portfolio weight $x_1^*$ obtained by solving the deterministic equivalent LP with CVXPY (Diamond & Boyd, 2016). We use HiGHS as the primary LP solver, with the risk parameters $\lambda = 0.1$ and $\alpha = 0.95$.

### G.1.3 Compared Models

The main experiment compares four architectures. **FCNN** flattens all scenario-tree node features into a vector of dimension $A|\mathcal{N}|$ and therefore has an input layer that grows with tree size. **DeepSets** applies a shared node encoder followed by permutation-invariant sum pooling, using node features but ignoring tree edges. **Tree-GNN** preserves the scenario-tree topology but performs only bottom-up child-to-parent message passing. **MSP-GNN** uses the full bidirectional scenario-tree graph, performs message passing over parent-child edges, and decodes the first-stage portfolio from the root embedding.

At the baseline configuration $(T = 3, b = 2, A = 5)$, the hidden dimensions are chosen to keep parameter counts at the same scale: FCNN uses a hidden dimension of 64, DeepSets uses a hidden dimension of 55, and both Tree-GNN and MSP-GNN use a hidden dimension of 39, yielding approximately 6.8k, 7.1k, 6.8k, and 6.8k trainable parameters, respectively. To isolate the effect of message-passing direction, Tree-GNN and MSP-GNN use the same hidden dimension and number of layers, resulting in identical parameter counts. All models use a softmax output layer so that the predicted first-stage portfolio satisfies $\hat{x}_1 \geq 0$ and $\mathbf{1}^\top \hat{x}_1 = 1$.

[4] A parametric LP may admit multiple optimal solutions. For each generated instance, we numerically verified uniqueness of the first-stage solution by computing the coordinate-wise variation of $x_1$ over the optimal solution set; all coordinate ranges were below $10^{-8}$. If multiple first-stage optima occur, a small strongly convex regularizer, such as $\frac{\mu}{2}\|x_1\|_2^2$ with $\mu > 0$, can be added to select a unique first-stage training target.

### G.1.4 Main Scaling Protocol

The main comparison evaluates scaling from a small baseline instance by increasing one structural dimension at a time: horizon depth $T$, branching factor $b$, or asset dimension $A$. An additional all-scale stress test increases all three dimensions simultaneously. Table 2 summarizes the five configurations. Figure 5 shows the corresponding test MSE learning curves.

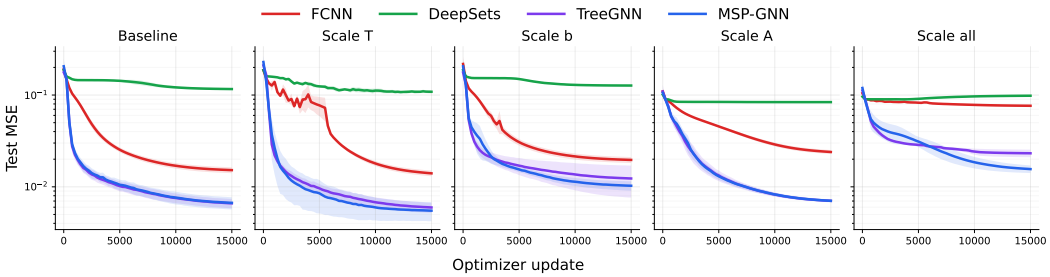


Figure 5: Test MSE learning curves over 15,000 optimizer updates across five scenario tree configurations.

Table 3: Performance on the 4,681-node asset-allocation stress test, reported as the mean ± standard deviation over three random seeds. Best results are shown in **bold**.

| **Method** | **Test MSE** $(\times 10^{-3})$ | **Parameters** $(\times 10^{3})$ | **Inference (ms)** |
|---|---|---|---|
| FCNN | $316.89 \pm 3.68$ | 1510.8 | $\mathbf{0.779 \pm 0.002}$ |
| DeepSets | $159.64 \pm 0.18$ | 19.9 | $1.156 \pm 0.007$ |
| Tree-GNN | $22.00 \pm 1.25$ | **13.2** | $1.614 \pm 0.007$ |
| **MSP-GNN** | $\mathbf{17.80 \pm 1.96}$ | **13.2** | $1.688 \pm 0.001$ |

### G.1.5 Large-Scale Evaluation

We further evaluate the scalability of MSP-GNN on a larger balanced scenario tree using the same models. We consider $T = 5$, $b = 8$, and $A = 5$, corresponding to a scenario tree with 4,681 nodes, and use four message-passing layers. All other training settings follow Appendix G.1.7, and the results are reported as the mean and standard deviation over three random seeds.

On the 4,681-node tree, MSP-GNN achieves the lowest test MSE, reducing the error of the bottom-up Tree-GNN by approximately 19.1% while using the same number of parameters. Compared with FCNN, MSP-GNN achieves higher predictive accuracy with approximately $114.5\times$ fewer parameters. Its consistent performance across three random seeds further indicates stable performance at this scale.

### G.1.6 Ablation Study

We conduct the ablation study on the Scale $T$ configuration $(T = 4, b = 2, A = 5)$ in Table 2, using the training settings in Table 4. The full MSP-GNN uses sum aggregation, $L = T - 1$ message-passing layers, and a root readout. To evaluate the contribution of each design choice, we modify one component at a time while keeping all other settings unchanged: the mean-aggregation variant replaces sum aggregation by mean aggregation; the shallow variant uses $L = T - 2$ layers; and the non-root readout variant replaces the root embedding by global mean pooling over all node embeddings.

Figure 6 reports the test MSE learning curves and final test MSE of MSP-GNN against three ablation variants on the Scale $T$ configuration $(T = 4, b = 2, A = 5)$. Replacing sum aggregation with mean aggregation yields comparable performance, indicating limited sensitivity to the choice of aggregation operator in this setting. By contrast, reducing the number of message-passing layers to $L = T - 2$ (Shallow) causes a more significant increase in test MSE, supporting the depth requirement $L \geq T - 1$ for information from all scenario-tree nodes to reach the root before readout.

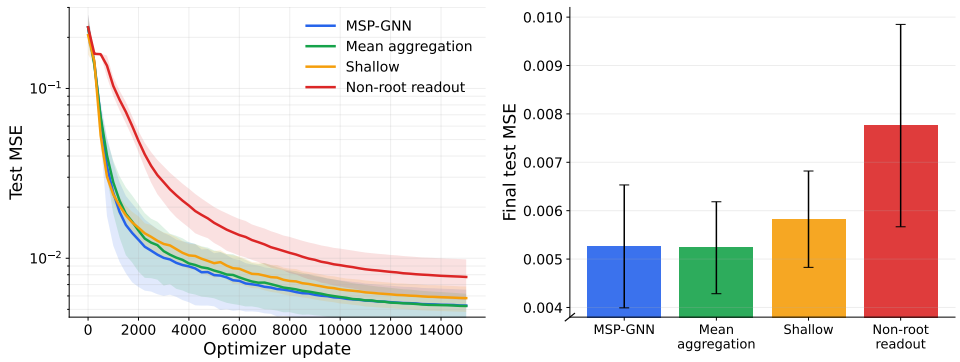


Figure 6: Ablation study on the Scale $T$ configuration ($T = 4, b = 2, A = 5$). *Left:* Test MSE learning curves over 15,000 optimizer updates; shaded regions denote $\pm$ one standard deviation over 3 random seeds. *Right:* Final test MSE with error bars.

The non-root readout variant performs worst among all ablations and exhibits the largest variance, demonstrating that anchoring the readout at the root node is a critical architectural choice, whereas aggregating representations over all nodes weakens the structural correspondence between the root and the first-stage decision. Together, these results provide empirical support for the depth and readout design choices of MSP-GNN, consistent with the theoretical analysis in Section 5.

### G.1.7 Training Hyperparameters

All models are trained with Adam (Kingma & Ba, 2014) for 15,000 optimizer updates using a learning rate of $2 \times 10^{-3}$, cosine learning-rate decay to $10^{-4}$, zero weight decay, and an exponential moving average with decay 0.99. For the main experiments, both the logical and physical batch sizes are 1,536. For the 4,681-node large-scale experiment, the batch size is reduced to 128 for computational efficiency. All models use `float32` training, with each model trained on a single NVIDIA H200 GPU. Table 4 summarizes the full experimental settings.

Table 4: Hyperparameter and implementation settings for the asset-allocation experiments.

| **Category** | **Hyperparameter** | **Value** |
|---|---|---|
| Architecture | Hidden dimension (FCNN) | 64 |
| | Hidden dimension (DeepSets) | 55 |
| | Hidden dimension (Tree-GNN/MSP-GNN) | 39 |
| | Message-passing layers | $L = T - 1$ |
| | Aggregation operator | Sum |
| Training | Optimizer | Adam (Kingma & Ba, 2014) |
| | Weight decay | 0 |
| | Initial learning rate | $2 \times 10^{-3}$ |
| | Learning-rate schedule | Cosine decay |
| | Minimum learning rate | $10^{-4}$ |
| | Optimizer updates | 15,000 |
| | Batch size (main / large-scale) | 1,536 / 128 |
| | EMA decay | 0.99 |
| | Random seeds | $\{0, 1, 2\}$ |
| | Data type | `float32` |
| | Device | NVIDIA H200 GPU |
| Dataset | Training instances | 30,000 |
| | Test instances | 1,000 |
| | Data-generation seed | 0 |
| Optimization labels | LP solver | HiGHS |
| | Risk-aversion coefficient $\lambda$ | 0.1 |
| | CVaR level $\alpha$ | 0.95 |

The training objective is the mean squared error between the predicted first-stage portfolio and the LP-optimal first-stage portfolio:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left\| \hat{x}_1^{(i)} - x_1^{*(i)} \right\|_2^2. \tag{49}$$

Results are reported as the mean $\pm$ standard deviation over the three random seeds. Inference time is measured per instance using all $1{,}000$ test instances and averaged over 10 repetitions.

#### G.1.8 Evaluation Metrics

We evaluate all methods using test MSE, trainable parameter count, and single-instance inference time.

$$\text{Test MSE} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left\| \hat{x}_1^{(i)} - x_1^{*(i)} \right\|_2^2. \tag{50}$$

Parameter count measures the number of trainable parameters. FCNN scales with the flattened input dimension $A|\mathcal{N}|$, whereas MSP-GNN and DeepSets share node-level parameters across tree nodes. Inference time is reported in milliseconds per instance.

### G.2 Weighted Stochastic LQR

#### G.2.1 Problem Formulation

We further evaluate MSP-GNN on a multistage stochastic linear–quadratic regulator (LQR) problem (Sun & Yong, 2020). This setting complements the asset-allocation experiment by considering an unbalanced scenario tree with non-uniform branch probabilities. At each non-leaf node $n \in \mathcal{N} \setminus \mathcal{N}_T$, the system has state $s_n \in \mathbb{R}^{d_s}$ and applies control $u_n \in \mathbb{R}^{d_u}$. For each child node $c \in \mathcal{C}(n)$, the state evolves according to

$$s_c = As_n + Bu_n + Dw_c, \tag{51}$$

where $w_c \in \mathbb{R}^{d_s}$ is the disturbance associated with node $c$. Let $p_n$ denote the probability of reaching node $n$ from the root, with $p_{n_0} = 1$. The resulting deterministic equivalent is

$$\min_{\{s_n\},\{u_n\}} \quad \sum_{n \in \mathcal{N} \setminus \mathcal{N}_T} p_n \left( s_n^\top Q s_n + u_n^\top R u_n \right) + \sum_{n \in \mathcal{N}_T} p_n s_n^\top Q_f s_n \tag{52}$$

$$\text{s.t.} \quad s_c = As_n + Bu_n + Dw_c, \quad \forall n \in \mathcal{N} \setminus \mathcal{N}_T, \quad \forall c \in \mathcal{C}(n), \tag{53}$$

$$s_{n_0} = s_0. \tag{54}$$

#### G.2.2 Data Preparation

We consider a four-stage non-regular scenario tree with 1, 2, 3, and 6 nodes at successive stages, for a total of 12 nodes. The six non-leaf nodes have branching degrees $(2, 1, 2, 3, 1, 2)$. Conditional branch probabilities are sampled independently for each tree. Binary branches use a $\mathrm{Beta}(2, 2)$ distribution, while ternary branches use a symmetric Dirichlet distribution. The probability $p_n$ of node $n$ is obtained by multiplying the conditional probabilities along the path from the root to $n$.

We set $d_s = 10$, $d_u = 5$, $Q = I$, $R = 0.1I$, $Q_f = 5I$, and $D = 0.2I$. The system matrices are fixed across all instances. Using a fixed random seed, $A$ is sampled and rescaled to have spectral radius $0.85$, while the entries of $B$ are sampled once from a zero-mean Gaussian distribution with standard deviation $0.35$. The initial state is fixed as $s_0 = (-0.5, -0.389, \ldots, 0.389, 0.5)$. Disturbances at non-root nodes are sampled as $w_n \sim \mathcal{N}(0, 0.5^2 I)$, and the root disturbance is set to zero. Each node is represented by the feature vector $(w_n, p_n)$.

Using data-generation seed 0, we generate a fixed pool of $11{,}000$ scenario trees. For each experimental seed, the pool is randomly partitioned into $10{,}000$ training trees and $1{,}000$ test trees. For every tree, the optimal root control $u_{n_0}^*$ and the corresponding objective value are computed exactly by backward dynamic programming.

### G.2.3 Baseline Methods

Similar to the asset-allocation experiment, we compare MSP-GNN against FCNN and DeepSets. We omit Tree-GNN from this experiment because its implementation assumes a balanced tree with a fixed branching factor, whereas the LQR scenario tree has a branching factor that varies across nodes.

### G.2.4 Training Hyperparameters

All models are trained for $3{,}000$ epochs using Adam with a learning rate of $10^{-3}$, a batch size of $512$, and an exponential-moving-average decay of $0.999$. Results are reported over seeds $\{0, 1, 2\}$.

Table 5: Performance on the four-stage weighted stochastic LQR problem with an asymmetric scenario tree. Test MSE and objective gap are reported as the mean $\pm$ population standard deviation over three random seeds.

| **Method** | **Test MSE** ($\times 10^{-3}$) | **Objective gap** (%) | **Parameters** ($\times 10^{3}$) | **Inference** (ms) |
|---|---|---|---|---|
| FCNN | $0.49 \pm 0.05$ | $0.12 \pm 0.01$ | 17.2 | **0.23** |
| DeepSets | $16.26 \pm 1.11$ | $2.99 \pm 0.15$ | 12.4 | 0.25 |
| **MSP-GNN** | $\mathbf{0.10 \pm 0.01}$ | $\mathbf{0.04 \pm 0.01}$ | **10.0** | 0.33 |

### G.2.5 Evaluation Metrics and Results

We evaluate the learned models using two metrics: (i) *prediction accuracy*, measured by the test MSE between the predicted and optimal root controls; and (ii) *decision quality*, measured by the objective gap after fixing the predicted root control and optimally solving the remaining stages through backward recursion. Specifically, for a predicted root control $\hat{u}_{n_0}$, we define

$$\mathrm{Gap}(\hat{u}_{n_0}) = \frac{J(\hat{u}_{n_0}) - J^*}{|J^*|} \times 100 \tag{55}$$

Since the LQR dynamics contain only equality constraints and the control variables are unconstrained, the downstream optimization remains feasible for any predicted root control.

MSP-GNN achieves the lowest test MSE and objective gap while using the fewest parameters. Relative to FCNN, it reduces test MSE by approximately $80\%$ and parameter count by approximately $42\%$. The poor performance of DeepSets further indicates that incorporating explicit probability features alone is insufficient without preserving the parent–child structure of the scenario tree. Overall, this experiment demonstrates the applicability of MSP-GNN to non-regular scenario trees with non-uniform branch probabilities.