EnerGNN: Learning Optimization-Compatible Energy Functions for Exact Constrained Combinatorial Inference

Niki Triantafyllou* **Maria M. Papathanasiou**
Sargent Centre for Process Systems Engineering, Department of Chemical Engineering
Imperial College London
{niki.triantafyllou20,maria.papathanasiou11}@imperial.ac.uk

Abstract

Learning-based approaches to combinatorial optimization often predict solutions directly or use predictions to restrict search, while expressive energy-based models typically rely on approximate inference. We introduce EnerGNN, a framework that learns energy functions over complete binary configurations and minimizes them under explicit hard constraints through mixed integer programming (MIP). EnerGNN combines a graph neural network encoder with an optimization-compatible readout and a nonlinear energy head, enabling exact constrained inference over the learned energy. An optimality-aware contrastive loss trains the energy to reflect solution quality and constraint violations. Across Quadratic Knapsack, Combinatorial Auctions, and a personalized cell therapy supply chain problem, EnerGNN achieves near-optimal solutions with 100% feasibility. On the supply chain problem, it achieves a 0.05% mean primal gap and a 17.4× speedup over the monolithic MIP. The learned energy also generalizes to larger instances and supports zero-shot constraint injection, adapting to changing constraints without retraining while preserving feasibility.

1 Introduction

A central challenge in energy-based structured prediction is to combine expressive learned representations with tractable inference over structured outputs. Energy-based models (EBMs) assign scalar energies to complete output configurations, allowing dependencies and interactions between output variables to be captured through learned energy functions (LeCun et al., 2006; Belanger and McCallum, 2016). However, minimizing expressive neural energies over discrete decision spaces is generally challenging, particularly when solutions must satisfy hard operational, physical, or logical constraints. Mixed integer programming (MIP) provides a framework for enforcing such constraints and obtaining globally optimal solutions, but learned energy functions must admit suitable mathematical representations to be optimized directly. This creates an interface problem between neural learning and exact optimization: *How can a neural model learn expressive energy functions that an exact optimizer can minimize under hard constraints?*

Existing approaches address different aspects of this challenge. In learning-based combinatorial optimization, neural models have been used to construct solutions, predict variable assignments, and guide exact solvers (Bengio et al., 2021; Khalil et al., 2017; Gasse et al., 2019). Methods such as Neural Diving (Nair et al., 2020) and ConPaS (Huang et al., 2024) use learned variable-level predictions to identify promising regions of the solution space, which are then explored through restricted optimization. In energy-based structured prediction, inference with expressive neural

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

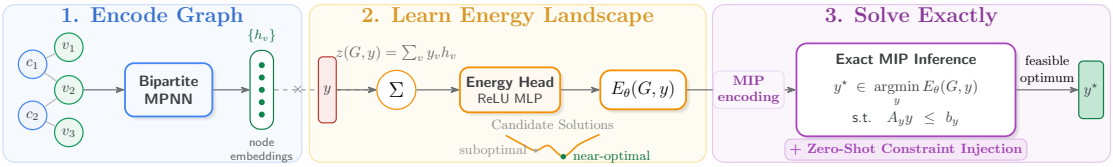


Figure 1: EnerGNN encodes an optimization instance using a bipartite message-passing GNN, learns an optimization-compatible energy function over binary configurations, and performs exact constrained inference by globally minimizing the learned energy over the encoded feasible set.

energies commonly relies on continuous relaxations, gradient-based optimization, or approximate decoding (Belanger and McCallum, 2016; Graber and Schwing, 2019). Meanwhile, ReLU neural networks can be represented exactly using mixed integer formulations (Fischetti and Jo, 2018; Anderson et al., 2020), providing a mechanism for optimizing learned nonlinear functions under explicit constraints. Together, these developments motivate learning energy functions that are designed to be directly optimized under explicit hard constraints, bringing expressive structured prediction and exact combinatorial inference into a unified framework.

We introduce **EnerGNN**, an energy-based framework that learns optimization-compatible energy functions over complete binary configurations and minimizes them subject to explicit hard constraints. As illustrated in Figure 1, EnerGNN combines a bipartite message-passing graph neural network (GNN) encoder with a decision-dependent readout and a piecewise linear ReLU energy head. The GNN encoder produces representations of the problem instance that are computed once and remain fixed during optimization. The readout combines these fixed representations with the binary decision variables to construct a configuration-dependent representation. Together, the readout and nonlinear ReLU energy head allow EnerGNN to model interactions between decision variables while admitting an exact MIP representation. At inference, the learned energy is minimized over the full encoded feasible region using a MIP solver, yielding a feasible solution that is globally optimal with respect to the learned energy when solved to optimality.

A key feature of EnerGNN is the separation of learned preferences from explicit feasibility constraints. The energy-based model learns to distinguish high-quality configurations from suboptimal alternatives through optimality-aware contrastive training, while hard constraints are enforced directly during inference. This separation allows the same learned energy to be optimized under different feasible regions, without retraining the model. In particular, deployment-time constraints can be added, removed, or modified by changing the inference MIP, enabling zero-shot constraint injection. This approach is complementary to neural algorithmic reasoning (Veličković and Blundell, 2021): rather than learning to execute an optimization algorithm, EnerGNN learns an energy function over structured decisions and delegates constrained inference to an exact optimizer.

Our contributions are threefold. First, we formulate constrained combinatorial inference as learning an optimization-compatible energy over complete binary configurations, separating learned preferences from explicit hard constraints. Second, we develop a graph-based energy architecture that captures nonlinear interactions between decision variables while admitting exact MIP representation and globally optimal inference with respect to the learned energy. Third, we develop an optimality-aware contrastive learning objective using feasible and infeasible candidate configurations to learn solution-level energy landscapes. We evaluate EnerGNN on Quadratic Knapsack (QKP), Combinatorial Auctions (CA), and a personalized cell therapy supply chain problem (CART). Our experiments demonstrate the benefits of nonlinear energy modeling for interaction-heavy objectives, the ability to obtain near-optimal solutions while satisfying hard constraints, generalization to larger problem sizes, and zero-shot constraint injection without retraining. The code and data used in this work are available at `https://github.com/nikitrian/EnerGNN`.

2 Related Work

Learning-based methods for combinatorial optimization. Learning-based approaches to combinatorial optimization commonly use neural models to amortize expensive search across related problem instances. Neural combinatorial optimization methods learn policies that construct solutions sequentially using graph-based or attention-based architectures, often trained through reinforcement learning (Khalil et al., 2017; Bello et al., 2016; Kool et al., 2019; Kwon et al., 2020). Another

line of work integrates learning into branch-and-bound algorithms, including branching-variable selection (Khalil et al., 2016; Gasse et al., 2019; Gupta et al., 2020) and search-tree node selection (He et al., 2014). Learning-based predict-and-search methods instead use neural predictions to identify promising variable assignments and solve reduced or restricted optimization problems. Neural Diving predicts distributions over variable assignments and fixes subsets of variables before solving reduced subproblems (Nair et al., 2020). Predict-and-Search (PaS) uses GNN predictions to define search neighborhoods around predicted solutions (Han et al., 2023), while Contrastive Predict-and-Search (ConPaS) extends this approach with contrastive learning to improve solution prediction (Huang et al., 2024).

Differentiable optimization and constraint enforcement. Decision-focused learning trains predictive models using downstream decision quality, while differentiable optimization methods integrate optimization problems into neural networks and propagate learning signals through their solutions (Amos and Kolter, 2017; Agrawal et al., 2019; Ferber et al., 2020; Paulus et al., 2021; Parmentier, 2025). These approaches enable learning of optimization problem parameters, including objective coefficients and, in some formulations, constraints. Related methods enforce feasibility of neural predictions through differentiable correction, projection, or constraint-enforcement layers (Donti et al., 2021; Min and Azizan, 2024; Wang et al., 2023; Tordesillas et al., 2023; Liang et al., 2024). Unlike differentiable optimization approaches, EnerGNN uses the MIP solver for inference without differentiating through it during training.

Energy-based structured prediction. Structured Prediction Energy Networks (SPENs) learn expressive energy functions over structured outputs and perform inference through continuous relaxation and gradient-based optimization (Belanger and McCallum, 2016). Input Convex Neural Networks (ICNNs) impose convexity with respect to the optimized inputs, enabling globally optimal inference in continuous convex settings (Amos et al., 2017). Graph Structured Prediction Energy Networks combine explicit local structure with learned higher-order interactions and perform iterative inference over a constrained relaxation (Graber and Schwing, 2019). Blondel et al. (2022) develop generalized Fenchel–Young losses for training energy networks without differentiating through the inference optimization problem. Energy-based models have also been integrated into decision-focused stochastic optimization, where learned energy functions are used to approximate optimization landscapes (Kong et al., 2022). More recently, Huang et al. (2026) propose an energy-based Langevin sampler for mixed integer optimization, combining neural sampling of integer variables with linear programming for continuous variables. EnerGNN builds on energy-based structured prediction but designs its learned energy function to admit an exact MIP representation, enabling global minimization over binary configurations under explicit hard constraints without requiring convexity.

Mixed integer representations of neural networks. Piecewise linear neural networks, particularly ReLU networks, can be represented exactly using mixed integer linear constraints (Tjeng et al., 2017; Fischetti and Jo, 2018; Anderson et al., 2020; Tsay et al., 2021). These formulations have been used for neural network verification and for embedding trained neural surrogate models inside optimization problems. Bound-tightening techniques can improve the strength and computational performance of these formulations (Grimstad and Andersson, 2019). EnerGNN uses these formulations to optimize a learned energy over binary configurations, with the GNN encoder evaluated before optimization and the readout and ReLU energy head represented exactly within the MIP formulation.

3 Methodology

We present the EnerGNN framework, which combines energy learning over binary configurations with exact constrained inference through mixed integer optimization. We first formulate the learning and inference problems, then describe the graph-based energy architecture, the training objective, and its exact MIP encoding. Finally, we establish the theoretical properties of the resulting framework.

3.1 Problem Setting

We consider mixed integer optimization problems of the form

$$\begin{aligned} \min_{\mathbf{y},\mathbf{x}} \quad & f(\mathbf{y},\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{y} \in \{0,1\}^n. \end{aligned} \tag{1}$$

where y denotes binary decision variables and x denotes any remaining operational variables.

We consider two settings. In direct decision problems, such as Quadratic Knapsack (QKP) and Combinatorial Auctions (CA), $\mathbf{y}$ constitutes the full decision vector. In hierarchical decomposition problems, such as the cell therapy supply chain problem (CART), $\mathbf{y}$ represents high-level planning decisions whose assignments define a reduced downstream scheduling problem. In the latter setting, fixing $\mathbf{y}$ induces the value function

$$F(\mathbf{y}) = \min_{\mathbf{x}} \{ f(\mathbf{y}, \mathbf{x}) \mid \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{x} \leq \mathbf{b} \}. \tag{2}$$

EnerGNN does not attempt to reproduce the original objective analytically. Instead, it learns an energy function $E_\theta(\mathcal{G}, \mathbf{y})$ over assignments of the binary decision variables, referred to as configurations, where $\mathcal{G}$ is a graph representation of the problem instance. Training uses solver-generated candidate configurations and their associated solution quality to learn an ordering in which lower energies correspond to higher-quality configurations. For direct decision problems, this ordering reflects the original objective, while for hierarchical decision problems it approximates the ordering induced by $F(\mathbf{y})$.

At inference, the learned energy is minimized subject to explicit hard constraints:

$$\begin{aligned} \mathbf{y}^\star \in \arg \min_{\mathbf{y} \in \{0,1\}^n} \quad & E_\theta(\mathcal{G}, \mathbf{y}) \\ \text{s.t.} \quad & \mathbf{A}_y \mathbf{y} \leq \mathbf{b}_y. \end{aligned} \tag{3}$$

The constraints define the feasible set, while the learned energy defines the objective optimized over that set. This formulation defines Neural-MIP inference.

3.2 Architecture Design

Graph Neural Network Encoder. Each instance is represented as a bipartite graph $\mathcal{G} = (\mathcal{C} \cup \mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the binary decision variables and $\mathcal{C}$ represents constraints or interactions, depending on the problem structure. Following the variable-constraint graph representation commonly used for learning in MIP solvers (Gasse et al., 2019), a bipartite message-passing GNN produces node embeddings $\{\mathbf{h}_v\}_{v \in \mathcal{V}}$ and $\{\mathbf{h}_c\}_{c \in \mathcal{C}}$, together with a global graph embedding $\mathbf{h}_\mathcal{G}$. The encoder is evaluated once per instance, and its outputs remain fixed during optimization.

Optimization-Compatible Readout. To enable exact optimization, the mapping $\mathbf{y} \mapsto \mathbf{z}^0$ must admit a mixed integer linear representation. We construct a benchmark-specific readout that combines the embeddings of selected binary variables with additional structural features:

$$\mathbf{z}^0(\mathbf{y}) = \Big[\sum_{v \in \mathcal{V}} y_v \mathbf{h}_v,\ \mathbf{z}_{\text{lin}}(\mathbf{y}),\ \mathbf{h}_\mathcal{G} \Big]. \tag{4}$$

The additional features $\mathbf{z}_{\text{lin}}(\mathbf{y})$ capture problem-specific information, such as unselected-item embeddings and capacity utilization in QKP, or occupied-item embeddings in CA. The global graph embedding $\mathbf{h}_\mathcal{G}$ provides an instance-level representation that remains fixed across binary configurations. All decision-dependent readout components are either affine in $\mathbf{y}$ or admit exact linearization using auxiliary variables. Benchmark-specific graph representations and readout definitions are provided in Appendix B.

Energy Head. A multilayer perceptron (MLP) with ReLU activations maps $\mathbf{z}^0(\mathbf{y})$ to a scalar energy $E_\theta(\mathcal{G}, \mathbf{y})$. The nonlinear energy head enables the model to capture interactions between binary decisions, while its piecewise linear structure, combined with the optimization-compatible readout, admits an exact mixed integer linear representation.

3.3 Optimality-Aware Energy Learning

We train on candidate configurations $\mathcal{Y} = \mathcal{F} \cup \mathcal{I}$, containing feasible and infeasible configurations. Each feasible candidate $\mathbf{y}_i$ is assigned a relative primal gap g_i, measured against the best available reference solution for that instance, while each infeasible candidate $\mathbf{y}_k$ is associated with a normalized constraint violation magnitude ν_k. The loss combines an optimality-aware contrastive term, feasible calibration, and infeasibility shaping:

$$\mathcal{L} = \mathcal{L}_{\text{con}} + \lambda_{\text{cal}} \mathcal{L}_{\text{cal}} + \lambda_{\text{inf}} \mathcal{L}_{\text{inf}}. \tag{5}$$

Optimality-aware contrastive loss. For each instance, we construct a target distribution that assigns greater probability to feasible configurations with smaller primal gaps and zero probability to infeasible configurations. The learned energy induces a second distribution over the entire candidate pool, and the Kullback–Leibler divergence encourages the two distributions to agree.

$$\begin{aligned} \mathcal{L}_{\mathrm{con}} &= \mathrm{KL}(p^\star \,\|\, p_\theta), \\ p_i^* &= \begin{cases} \dfrac{\exp(-g_i/\tau_{\mathrm{rank}})}{\sum_{j\in\mathcal{F}} \exp(-g_j/\tau_{\mathrm{rank}})}, & i \in \mathcal{F}, \\ 0, & i \notin \mathcal{F}, \end{cases} \\ p_{\theta,i} &= \frac{\exp(-E_\theta(\mathcal{G}, \mathbf{y}_i)/T)}{\sum_{j\in\mathcal{Y}} \exp(-E_\theta(\mathcal{G}, \mathbf{y}_j)/T)}, \qquad i \in \mathcal{Y}. \end{aligned} \tag{6}$$

where T is the model softmax temperature and τ_{rank} controls the target sharpness.

Calibration loss. The contrastive loss primarily supervises relative preferences within the candidate pool. We additionally calibrate the energies of feasible configurations against their relative primal gaps using Huber regression with an instance-adaptive transition scale.

$$\mathcal{L}_{\mathrm{cal}} = \frac{1}{|\mathcal{F}|} \sum_{i\in\mathcal{F}} H_{\delta_{\mathcal{F}}}(E_\theta(\mathcal{G}, \mathbf{y}_i) - g_i)\,. \tag{7}$$

The transition scale $\delta_{\mathcal{F}}$ is computed from the primal-gap span of feasible candidates. Details are given in Appendix E.

Infeasibility loss. For infeasible configurations, we use an auxiliary Huber loss to align their energies with a logarithmically scaled measure of constraint violation:

$$\mathcal{L}_{\mathrm{inf}} = \frac{1}{|\mathcal{I}|} \sum_{k\in\mathcal{I}} H_{\delta_{\mathrm{inf}}}(E_\theta(\mathcal{G}, \mathbf{y}_k) - \log(1+\nu_k))\,, \tag{8}$$

where $\nu_k \geq 0$ is the normalized constraint violation. This term provides a graded training signal within the infeasible region, while $\mathcal{L}_{\mathrm{con}}$ discourages infeasible candidates from receiving probability mass at the expense of high-quality feasible configurations. Feasibility at inference is enforced explicitly by the MIP constraints.

3.4 Inference via Neural-MIP Encoding

At inference, the trained GNN encoder is evaluated once, while the decision-dependent readout and ReLU energy head are represented exactly within the MIP using auxiliary variables and linear constraints. Valid pre-activation bounds are obtained through interval bound propagation using benchmark-specific input bounds (Gowal et al., 2018). Neurons with fixed activation states are eliminated to reduce the formulation size. The exact Neural-MIP formulation is detailed in Appendix C, with the bound-tightening procedure described in Appendix C.2.

3.5 Theoretical Properties

We summarize the formal properties induced by the architectural design. Detailed proofs are provided in Appendix A.

Proposition 3.1 (MIP-representable energy)**.** *For a fixed instance graph $\mathcal{G}$, assume that the encoder is evaluated once, the readout $\mathbf{z}^0(\mathbf{y})$ admits an exact mixed integer linear representation over a bounded domain, and the energy head is a feed-forward ReLU network. Then $\mathbf{y} \mapsto E_\theta(\mathcal{G}, \mathbf{y})$ is exactly MIP-representable on that domain.*

Proof sketch. For fixed $\mathcal{G}$, encoder outputs are constants. The readout admits an exact mixed integer linear representation by construction. A ReLU network is a composition of affine maps and ReLU operators, each of which admits an exact mixed integer linear encoding over bounded input domains. Hence the full energy admits an exact MIP representation. □

Corollary 3.2 (Exact feasible inference)**.** *If the encoded feasible set*

$$\mathcal{C} = \{\mathbf{y} \in \{0,1\}^n \mid \mathbf{A}_y\mathbf{y} \leq \mathbf{b}_y\}$$

is nonempty, then solving

$$\min_{\mathbf{y}\in\mathcal{C}} E_\theta(\mathcal{G}, \mathbf{y})$$

is equivalent to solving a mixed integer linear program and returns a feasible solution that is globally optimal with respect to the learned energy.

Corollary 3.3 (Constraint injection)**.** *The same learned energy can be optimized over any bounded test-time feasible set obtained by adding, removing, or modifying linear constraints over the same decision variables, without retraining the EBM. If the test-time feasible set is not contained in the domain used to compute the original Big-M bounds, the bounds used in the MIP encoding must be recomputed for the new feasible set.*

Proposition 3.4 (Expressivity beyond factorized scoring)**.** *If the readout contains the binary variables* $\mathbf{y}$*, or an affine representation from which they can be recovered, a one-hidden-layer ReLU energy can represent arbitrary quadratic pairwise interactions over binary variables exactly, whereas a factorized linear score cannot represent a nonzero pairwise interaction.*

4 Experiments

We evaluate EnerGNN on three combinatorial optimization problems: Quadratic Knapsack (QKP), Combinatorial Auctions (CA), and a Chimeric Antigen Receptor (CAR) T-cell therapy supply chain (CART). QKP and CA evaluate direct decision inference, while CART considers hierarchical decomposition, where EnerGNN selects high-level capacitated facility location decisions before solving a reduced downstream patient scheduling problem. Our experiments assess solution quality, feasibility, the contribution of nonlinear energy modeling, cross-scale generalization, and zero-shot constraint injection. Each benchmark examines a different aspect of the framework: QKP tests nonlinear interactions between variables, CA evaluates inference under packing constraints, and CART examines learning over complicating variables in hierarchical optimization.

4.1 Benchmarks

Quadratic Knapsack. QKP is an interaction-heavy problem involving the selection of items under a capacity constraint, with both individual profits and pairwise interactions contributing to the objective. We consider instances with 150 binary item-selection variables and an interaction density of 0.30.

Cell Therapy Supply Chain. CART is a time-indexed, capacitated facility location and patient allocation MIP for personalized CAR T-cell therapy supply chains. The network comprises 16 candidate manufacturing facilities. The formulation incorporates 2 transportation modes and daily scheduling over a 130-day horizon, with patient demand varying from 10 to 650 across different demand distributions. EnerGNN optimizes the 16 first-stage facility location variables, with each configuration of these variables inducing a tactical patient scheduling MIP.

Combinatorial Auctions. CA is a packing problem with 175 binary bid-selection variables, extended to 250 in the cross-scale evaluation. Each bid consumes a subset of items, and each item can be allocated at most once. Because the original objective is linear and these instances are comparatively easy for modern MIP solvers, CA primarily evaluates constrained inference and generalization rather than computational speedup.

Detailed benchmark problem formulations are given in Appendix B. Instance generation procedures and dataset statistics are described in Appendix D. Benchmark-specific Neural-MIP input bounds and hard feasibility constraints are detailed in Appendices C.3 and C.4, respectively.

4.2 Baselines and Metrics

We compare against three classes of baselines that differ in how learning is combined with optimization. *Full MIP* solves the original problem using branch-and-cut and provides a reference optimum. *Predict+Search*, *Neural Diving* (Nair et al., 2020), and *ConPaS* (Huang et al., 2024) use learned predictions to restrict or guide the solver's search. Predict+Search fixes variables based on prediction

confidence and solves the resulting reduced problem. Neural Diving samples variable assignments, fixes subsets of variables, and solves multiple reduced subproblems. ConPaS instead imposes a neighborhood constraint around its learned prediction while retaining the original objective.

Linear+MIP and *EnerGNN* learn objectives that are optimized directly under explicit hard constraints. Linear+MIP uses additive per-variable scores, whereas EnerGNN learns a joint energy over complete configurations that can capture interactions between decisions. We evaluate Linear+MIP with binary cross-entropy (BCE) and contrastive supervision, and include *EnerGNN (Linear)* as an ablation that replaces the ReLU energy head with a linear head over the same readout. This isolates the contribution of the nonlinear energy head. All learning-based methods use the same bipartite GNN encoder and input features where applicable, but differ in their prediction heads, training objectives, and inference procedures. Full implementation details and inference settings are provided in Appendix K.

We report mean primal gap, runtime, feasibility, and the fraction of instances with a primal gap of at most 1% relative to the certified global optimum. Primal gaps are computed over all test instances, assigning a 100% penalty to infeasible outputs. EnerGNN achieves 100% feasibility across the reported experiments, with hard constraints enforced directly during inference.

Table 1: Main results across benchmarks (mean over test instances). We report primal gap, runtime, feasibility, and the fraction of instances achieving either exact recovery or a solution within 1% of the global optimum.

	QKP				**CART**				**CA**			
Method	Gap (%)	Time (s)	$\leq$1% (%)	Feas. (%)	Gap (%)	Time (s)	$\leq$1% (%)	Feas. (%)	Gap (%)	Time (s)	$\leq$1% (%)	Feas. (%)
Full MIP (Gurobi)	0.00	525.21	100	100	0.00	3728.82	100	100	0.00	0.14	100	100
Predict+Search	1.13	0.12	97	99	22.02	251.79	21	83	29.81	0.01	0	100
Neural Diving	1.04	1.52	99	99	1.53	398.34	65	100	2.91	1.23	5	100
ConPaS	0.95	1.03	100	100	0.02	3404.19	100	100	0.90	0.26	65	100
Linear+MIP (BCE)	4.92	0.03	20.1	100	15.81	273.60	17	100	83.99	0.02	0	100
Linear+MIP (Contrastive)	58.55	0.03	0.00	100	24.18	410.04	4	100	30.82	0.02	0	100
EnerGNN (Linear)	26.87	0.28	0.00	100	21.80	391.36	5	100	0.20	0.31	95	100
EnerGNN (MLP)	0.31	3.56	99	100	0.05	214.03	100	100	0.53	8.62	77	100

4.3 Main Results

Table 1 summarizes the results across the three benchmarks.

On QKP, EnerGNN (MLP) achieves the lowest mean primal gap among learning-based methods at 0.31%, with 99% of instances within 1% of the global optimum and 100% feasibility. ConPaS, Neural Diving, and Predict+Search achieve mean gaps of 0.95%, 1.04%, and 1.13%, respectively, while the linear scoring baselines perform substantially worse. The difference between EnerGNN (MLP) and its linear head ablation (0.31% versus 26.87%) is particularly informative. Both use the same graph encoder and decision-dependent readout, but the nonlinear head can capture additional interactions between readout features. Predict+Search, Neural Diving, and ConPaS remain faster, illustrating a trade-off between solution quality and inference time.

On CART, the strategic facility location decisions determine a much larger downstream patient allocation MIP. Neural Diving and ConPaS use learned predictions to restrict the strategic search, achieving mean gaps of 1.53% and 0.02% in 398.34 s and 3404.19 s, respectively. While Neural Diving uses parallel dives, allowing flips in the ConPaS neighborhood requires retaining all candidate facilities in our CART formulation, preventing the elimination of facility-dependent patient allocation dimensions. EnerGNN instead learns the downstream value of complete strategic configurations offline and optimizes this learned energy during inference. Once the facility configuration is selected, the operational MIP is constructed using only the selected facilities, reducing the number of variables and constraints by approximately 77% (Appendix F). EnerGNN achieves a 0.05% mean gap in 214.03 s, with all test instances within 1% of the global optimum and 100% feasibility, making it approximately $1.9\times$ faster than Neural Diving, $15.9\times$ faster than ConPaS, and $17.4\times$ faster than the monolithic MIP.

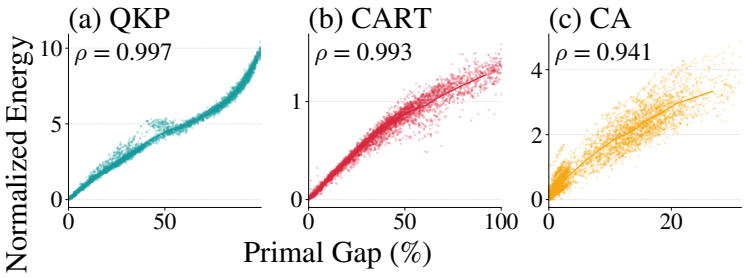


Figure 2: Energy vs. solution quality. Spearman ρ is reported per benchmark.

On CA, the full MIP is solved by Gurobi in 0.14 s, so this benchmark is not intended to demonstrate solver acceleration. Instead, it tests whether a learned energy can be optimized under a large number of explicit packing constraints. The linear EnerGNN head achieves a 0.20% mean gap with 95% of instances within 1% of the optimum and 100% feasibility. It outperforms the nonlinear variant, which achieves a 0.53% mean gap at substantially higher inference cost. In contrast to QKP, CA has a linear original objective, and the linear energy head performs better on these instances. These results indicate that the benefit of nonlinear energy heads depends on the problem structure.

Figure 2 further examines whether the learned energy reflects the quality of complete configurations. The strong rank correlations between energy and primal gap across QKP, CART, and CA ($\rho \geq 0.94$) indicate that lower energy configurations generally correspond to better solutions.

Table 2: Cross-scale generalization without retraining.

Benchmark	**Train $\rightarrow$ test**	**Primal Gap (%)**	**Instances $\leq 1\%$ (%)**	**Feasibility (%)**
CART	$m = 16 \rightarrow 12$	0.98	63	100
CART	$m = 16 \rightarrow 16$	0.05	100	100
CART	$m = 16 \rightarrow 20$	1.23	49	100
CA	$175 \rightarrow 175$	0.20	95	100
CA	$175 \rightarrow 250$	0.98	69	100

4.4 Cross-Scale Generalization

We evaluate whether EnerGNN transfers to different problem sizes without retraining. In the CART setting, the model is trained on instances with $m = 16$ candidate facilities, and is evaluated at $m = 12$ and $m = 20$. In CA, the number of binary bid variables increases from 175 during training to 250 at test time. Table 2 shows that EnerGNN maintains 100% feasibility across all evaluated scales. In CART, the model achieves a mean gap of 0.98% when transferred to $m = 12$ facilities and 1.23% when extrapolated to $m = 20$, compared with 0.05% at the training scale. For CA, we report the linear head variant of EnerGNN, which achieves the best performance on this benchmark. Results for the nonlinear energy head are provided in Appendix I. Increasing the number of bids from 175 to 250 increases the mean primal gap from 0.20% to 0.98%. These results show that the learned energy can be applied to variable spaces of different sizes without retraining, although solution quality degrades under larger extrapolation shifts.

4.5 Zero-Shot Constraint Injection

We evaluate whether a trained EnerGNN can be reused after the feasible region changes without retraining. The learned energy is held fixed, and only the constraints of the inference MIP are modified. In CART ($m = 16$), we deactivate one, two, or three facilities selected in the original optimum. In CA, we remove 5, 10, or 15 bids selected in the original optimal allocation, corresponding to approximately 20%, 40%, and 60% of the selected bids, respectively, and additionally test a setting in which the highest-value bid rejected by the original optimal solution is forced into the allocation. The CART evaluation comprises 1,092 perturbations in total (319, 431, and 342 cases for removing one, two, and three selected facilities, respectively). For CA, each bid-removal setting is evaluated over 1,100 perturbations, while the forced-bid setting contains 100 perturbations. For CA constraint-

Table 3: Zero-shot constraint injection. We report the fraction of originally selected variables removed, the mean Hamming shift from the original optimum, mean primal gap, the fraction of instances within 1% of the modified global optimum, and feasibility.

				EnerGNN			**Neural Diving**			**ConPaS**		
Benchmark	**Constraint shift**	Selected removed (%)	Hamming shift	Gap (%)	$\leq 1\%$ (%)	Feas. (%)	Gap (%)	$\leq 1\%$ (%)	Feas. (%)	Gap (%)	$\leq 1\%$ (%)	Feas. (%)
CART (facilities)	Remove 1	31.35	3.84	0.21	96.24	100	2.99	40.44	89.66	0.30	87.15	100
	Remove 2	50.81	5.27	0.34	93.27	100	4.38	25.06	72.85	0.70	77.96	99.77
	Remove 3	61.99	6.26	0.45	91.23	100	4.32	20.18	49.12	1.08	69.88	98.28
CA (bids)	Remove 5	20.30	34.87	0.21	93.36	100	3.41	9.73	100	1.13	54.09	100
	Remove 10	40.61	38.77	0.17	95.27	100	4.12	5.09	100	1.45	41.09	100
	Remove 15	60.91	41.36	0.16	95.73	100	4.76	3.82	100	1.62	38.27	100
	Force 1	–	31.25	0.07	99.00	100	3.60	17.00	100	1.06	57.00	100

injection experiments, we use the linear EnerGNN variant, which performs best on the in-distribution CA benchmark.

Table 3 compares EnerGNN with Neural Diving and ConPaS under the same constraint shifts. EnerGNN remains feasible in every case and achieves the lowest mean gap across all tested perturbations. On CART, its mean gap increases from 0.21% to 0.45% as one to three originally selected facilities are deactivated, while maintaining at least 91.23% of instances within 1% of the modified global optimum. By contrast, Neural Diving degrades substantially under the same shifts, reaching only 49.12% feasibility when three facilities are deactivated, while ConPaS exhibits larger gaps than EnerGNN across all CART perturbations. On CA, EnerGNN achieves mean gaps between 0.07% and 0.21%, with at least 93.36% of instances within 1% of the modified optimum. Both baselines obtain substantially larger gaps across all CA constraint shifts. The perturbations induce substantial changes in the selected solutions, with a mean Hamming shift of 6.26 when three CART facilities are deactivated and 41.36 when 15 CA bids are removed. Additional constraint-injection experiments are reported in Appendix H.

4.6 Loss Ablations

Table 4: Loss ablation.

	CART		**QKP**		**CA**	
Variant	Gap (%)	$\leq$1% (%)	Gap (%)	$\leq$1% (%)	Gap (%)	$\leq$1% (%)
Full EnerGNN	0.05	100	0.31	99	0.53	77
No calibration loss	0.78	70	0.88	68	7.69	4
No infeasibility loss	0.06	100	0.39	99	0.58	77

We ablate the loss components used to shape the energy landscape. The full loss combines an optimality-aware contrastive loss, feasible calibration, and infeasibility shaping. We additionally evaluate an alternative loss enforcing strict feasible–infeasible ordering in Appendix J.3. Table 4 shows that the calibration loss is important for learning an energy scale that reflects solution quality. Removing calibration increases the mean gap on CART from 0.05% to 0.78% and reduces the fraction of instances within 1% of the global optimum from 100% to 70%. On QKP, the corresponding fraction decreases from 99% to 68%, while the largest degradation occurs on CA, where the mean gap increases from 0.53% to 7.69% and only 4% of instances remain within 1% of the optimum. In contrast, removing the infeasibility term has only a limited effect across all three benchmarks, suggesting that its contribution to solution quality is comparatively small.

4.7 Inference Size and Tractability

Table 5 reports the size of the Neural-MIP, where $|\mathbf{y}|$ denotes the original binary variables optimized by EnerGNN. The graph encoder is evaluated once before optimization and therefore does not

contribute optimization variables. Additional binaries arise from the exact readout linearizations and the compact ReLU energy head. Appendix J.1 shows that increasing the width or depth of the energy head substantially increases inference time without improving solution quality.

Table 5: Neural-MIP formulation dimensions and mean inference time across test instances. For CART, inference time excludes downstream patient scheduling.

Benchmark	$\|y\|$	Readout binaries	ReLU binaries	Total binaries	Continuous variables	Hard feasibility constraints	Total constraints	Inference time (s)
CART	16	4	32	52	290	1	461	1.46
QKP	150	0	32	182	292	1	389	3.56
CA	175	0	32	207	298	131.74	525.74	8.62

4.8 Limitations

EnerGNN requires the decision-dependent energy head to remain MIP-representable, creating an expressiveness–tractability trade-off in the embedded component. In our experiments, compact shallow heads are sufficient, while increasing width or depth can substantially increase inference time without systematically improving solution quality. Additional representational capacity can instead be placed in the GNN encoder, which is evaluated before optimization and does not increase Neural-MIP size. Training also relies on solver-generated candidate configurations and labels, introducing a one-time offline cost that is most appropriate for repeated-deployment settings; a complete cost and amortization analysis is provided in Appendix D.1. Finally, exact inference is with respect to the learned energy rather than the original objective. Constraint injection guarantees feasibility with respect to the encoded constraints, but solution quality may degrade under sufficiently large distribution or feasible-set shifts.

5 Conclusion

We introduced EnerGNN, an energy-based framework that learns optimization-compatible energy functions over binary configurations and minimizes them under explicit hard constraints through mixed integer optimization. The framework combines a graph-based encoder with a MIP-representable readout and nonlinear energy head, enabling exact optimization of the learned energy.

Our experiments demonstrate the framework across direct decision and hierarchical decomposition settings. EnerGNN achieves near-optimal solutions with 100% feasibility across Quadratic Knapsack, Combinatorial Auctions, and a personalized cell therapy supply chain problem. On the latter, EnerGNN achieves substantial computational speedups over the monolithic MIP and learning-based baselines while maintaining near-optimal solution quality. EnerGNN also generalizes to larger problem instances without retraining and supports zero-shot constraint injection, allowing the same learned energy to be optimized under changing operational constraints while preserving feasibility. This adaptability is particularly valuable in deployment settings where operational requirements and resource availability may change over time.

More broadly, EnerGNN suggests a design principle for reliable AI systems. When discrete decisions must satisfy hard constraints, learning can shape objectives that algorithmic modules optimize rather than producing final decisions directly. This shifts the learned object from the decision itself to an optimization-compatible objective over decisions, allowing learned preferences and explicit constraints to remain separate at inference.

References

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992. PMLR, 2016.

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.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid Von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

Taoan Huang, Aaron Ferber, Arman Zharmagambetov, Yuandong Tian, and Bistra Dilkina. Contrastive predict-and-search for mixed integer linear programs. In *Proceedings of the 41st International Conference on Machine Learning*, pages 19757–19771, 2024.

Colin Graber and Alexander G. Schwing. Graph structured prediction energy networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3): 296–309, 2018.

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1):3–39, 2020.

Petar Veličković and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7), 2021.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. *Advances in neural information processing systems*, 33:21188–21198, 2020.

Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Prateek Gupta, Maxime Gasse, Elias Khalil, Pawan Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. *Advances in neural information processing systems*, 33:18087–18097, 2020.

He He, Hal Daumé, and Jason Eisner. Learning to search in branch and bound algorithms. *Advances in neural information processing systems*, 27, 2014.

Qingyu Han, Linxin Yang, Qian Chen, Xiang Zhou, Dong Zhang, Akang Wang, Ruoyu Sun, and Xiaodong Luo. A GNN-guided predict-and-search framework for mixed-integer linear programming. In *The Eleventh International Conference on Learning Representations*, 2023.

Brandon Amos and J Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145, 2017.

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. MIPaaL: Mixed integer program as a layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1504–1511, 2020.

Anselm Paulus, Michal Rolínek, Vít Musil, Brandon Amos, and Georg Martius. CombOptNet: Fit the right np-hard problem by learning integer programming constraints. In *International Conference on Machine Learning*, pages 8443–8453. PMLR, 2021.

Axel Parmentier. Learning structured approximations of combinatorial optimization problems. *Open Journal of Mathematical Optimization*, 6:1–27, 2025.

Priya L. Donti, David Rolnick, and J Zico Kolter. DC3: A learning method for optimization with hard constraints. In *International Conference on Learning Representations*, 2021.

Youngjae Min and Navid Azizan. HardNet: Hard-constrained neural networks with universal approximation guarantees. *arXiv preprint arXiv:2410.10807*, 2024.

Runzhong Wang, Yunhao Zhang, Ziao Guo, Tianyi Chen, Xiaokang Yang, and Junchi Yan. LinSATNet: the positive linear satisfiability neural networks. In *International Conference on Machine Learning*, pages 36605–36625. PMLR, 2023.

Jesus Tordesillas, Jonathan P How, and Marco Hutter. RAYEN: Imposition of hard convex constraints on neural networks. *arXiv preprint arXiv:2307.08336*, 2023.

Enming Liang, Minghua Chen, and Steven H Low. Homeomorphic projection to ensure neural-network solution feasibility for constrained optimization. *Journal of Machine Learning Research*, 25(329):1–55, 2024.

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 146–155. PMLR, 2017.

Mathieu Blondel, Felipe Llinares-López, Robert Dadashi, Léonard Hussenot, and Matthieu Geist. Learning energy networks with generalized Fenchel–Young losses. *Advances in Neural Information Processing Systems*, 35:12516–12528, 2022.

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.

Yixin Huang, Shengyu Feng, and Yiming Yang. Unsupervised neural Langevin sampler for mixed integer linear programming. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 46553–46567. PMLR, 2026.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Calvin Tsay, Jan Kronqvist, Alexander Thebelt, and Ruth Misener. Partition-based formulations for mixed-integer optimization of trained ReLU neural networks. *Advances in neural information processing systems*, 34:3068–3080, 2021.

Bjarne Grimstad and Henrik Andersson. Relu networks as surrogate models in mixed-integer linear programs. *Computers & Chemical Engineering*, 131:106580, 2019.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 66–76, 2000.

David Pisinger. The quadratic knapsack problem—a survey. *Discrete applied mathematics*, 155(5):623–648, 2007.

Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2023. URL `https://www.gurobi.com`.

Niki Triantafyllou, Andrea Bernardi, Matthew Lakelin, Nilay Shah, and Maria M Papathanasiou. A digital platform for the design of patient-centric supply chains. *Scientific Reports*, 12(1):17365, 2022.

A Proofs of Theoretical Properties

We provide the proofs for the theoretical properties stated in Section 3.5. Throughout this appendix, the instance graph $\mathcal{G}$ is fixed. The graph encoder is evaluated once before inference, and its outputs are treated as constants. The optimization variables in Neural-MIP inference are therefore the original binary decision variables, the readout auxiliary variables, the ReLU activation variables, and the scalar energy variable.

A.1 Preliminaries

Definition A.1 (MIP-representability)**.** Let $\mathcal{D} \subset \mathbb{R}^d$ be a bounded domain. A function $f : \mathcal{D} \to \mathbb{R}$ is MIP-representable on $\mathcal{D}$ if its graph restricted to $\mathcal{D}$ can be described exactly by a finite collection of linear constraints involving auxiliary continuous and integer variables. Equivalently, there exists a mixed integer linear feasible set whose projection onto the input-output coordinates is exactly

$$\{(\mathbf{x}, t) \mid t = f(\mathbf{x})\}.$$

We use the following standard exact representation of a ReLU unit.

Lemma A.1 (Exact ReLU encoding)**.** *Let $h = \max\{0, a\}$, where the pre-activation a has finite bounds*

$$\underline{a} \le a \le \overline{a}.$$

Assume $\underline{a} \le 0 \le \overline{a}$. Then the graph of the ReLU can be represented exactly using one binary variable $z \in \{0, 1\}$ and the linear constraints

$$h \ge a, \tag{9}$$
$$h \ge 0, \tag{10}$$
$$h \le a - \underline{a}(1 - z), \tag{11}$$
$$h \le \overline{a} z. \tag{12}$$

If the bounds imply that the ReLU is always active or always inactive, the binary variable can be removed and the ReLU can be replaced by a linear equality.

Proof. Consider first the case $z = 0$. Constraint (12) gives $h \le 0$, and constraint (10) gives $h \ge 0$. Hence $h = 0$. Constraint (9) then implies $a \le 0$. Therefore the inactive case represents $h = \max\{0, a\} = 0$.

Now consider the case $z = 1$. Constraint (11) gives $h \le a$, and constraint (9) gives $h \ge a$. Hence $h = a$. Constraint (10) also implies $a \ge 0$. Therefore the active case represents $h = \max\{0, a\} = a$.

The two cases together describe exactly the graph of the ReLU over the bounded interval $[\underline{a}, \overline{a}]$. □

A.2 Proof of Proposition 3.1

Proof. Fix an instance graph $\mathcal{G}$. The graph encoder produces node embeddings and a global graph embedding. Since the encoder is evaluated before inference, these embeddings are constants inside the optimization problem.

The readout maps the binary decision vector $\mathbf{y}$ to an input feature vector $\mathbf{z}^0(\mathbf{y})$ for the energy head. By construction, each component of the readout is either affine in $\mathbf{y}$ or admits an exact mixed integer linear representation using auxiliary variables. The graph of the readout can therefore be described by linear constraints together with integrality restrictions.

The energy head is a feed-forward ReLU network. Each layer applies an affine transformation followed by componentwise ReLU activations. The affine transformations are represented by linear equalities. Each ReLU activation is represented exactly by Lemma A.1, using valid lower and upper bounds on the corresponding pre-activation.

Combining the readout representation, the affine layer equations, the ReLU encodings, and the final linear output equation gives a finite mixed integer linear representation of the graph of

$$\mathbf{y} \mapsto E_\theta(\mathcal{G}, \mathbf{y}).$$

Thus, for fixed $\mathcal{G}$, the learned energy is MIP-representable. □

A.3 Proof of Corollary 3.2

Proof. By Proposition 3.1, the learned energy $E_\theta(\mathcal{G}, \mathbf{y})$ admits an exact mixed integer linear representation. The feasible set

$$\mathcal{C} = \left\{ \mathbf{y} \in \{0,1\}^{|\mathcal{V}|} \mid \mathbf{A}_y \mathbf{y} \leq \mathbf{b}_y \right\}$$

is described by linear constraints and binary restrictions.

The inference problem can therefore be written as the mixed integer linear optimization problem

$$\begin{aligned} \min_{\mathbf{y}, \boldsymbol{\xi}, e} \quad & e \\ \text{s.t.} \quad & e = E_\theta(\mathcal{G}, \mathbf{y}), \\ & \mathbf{A}_y \mathbf{y} \leq \mathbf{b}_y, \\ & \mathbf{y} \in \{0,1\}^{|\mathcal{V}|}, \end{aligned}$$

where $\boldsymbol{\xi}$ denotes all auxiliary readout and ReLU variables.

If $\mathcal{C}$ is nonempty and this optimization problem is solved to global optimality, the returned solution is feasible by construction and globally optimal with respect to the learned energy over $\mathcal{C}$. If the solver is stopped early, any returned incumbent remains feasible, although it need not be globally optimal with respect to the learned energy. □

A.4 Proof of Corollary 3.3

Proof. The learned parameters θ define the energy function $E_\theta(\mathcal{G}, \mathbf{y})$. This function does not depend on the particular deployment-time feasible region. The feasible region enters only through the constraints imposed during inference.

Let

$$\mathcal{C}_{\text{test}} = \left\{ \mathbf{y} \in \{0,1\}^{|\mathcal{V}|} \mid \mathbf{A}_{\text{test}} \mathbf{y} \leq \mathbf{b}_{\text{test}} \right\}$$

be a test-time feasible set obtained by adding, removing, or modifying linear constraints over the same decision variables. The same learned energy can be optimized over the modified feasible set by solving

$$\mathbf{y}^\star \in \arg\min_{\mathbf{y} \in \mathcal{C}_{\text{test}}} E_\theta(\mathcal{G}, \mathbf{y}).$$

Exactness of the mixed integer encoding requires valid finite bounds for all ReLU pre-activations over the feasible set used at inference. If $\mathcal{C}_{\text{test}}$ is contained in the domain used to compute the original Big-M constants, the original bounds remain valid. If the feasible set is expanded or otherwise leaves that domain, the Big-M constants are recomputed by bound propagation over $\mathcal{C}_{\text{test}}$. This changes only the encoding constants and does not require retraining the neural model. □

A.5 Bound Tightening

The exact ReLU encoding requires valid lower and upper bounds on each pre-activation. In practice, these bounds are obtained using interval bound propagation through the assembled readout and energy head. Tighter bounds make the mixed integer linear representation stronger by reducing Big-M constants and removing ReLU binaries whose activation status is fixed. The bound computation used in the experiments is described in Appendix C.2.

A.6 Expressivity Beyond Factorized Linear Scoring

We next formalize why the nonlinear energy head can represent interactions that are impossible for additive per-variable scoring functions.

Definition A.2 (Factorized linear score)**.** A score over binary configurations is *factorized linear* if it decomposes additively over individual variables as

$$L(\mathbf{y}) = b + \sum_i w_i y_i.$$

Such a score assigns a contribution to each selected variable independently of which other variables are selected.

Proposition A.2 (Expressivity beyond linear scoring)**.** *Let*

$$Q(\mathbf{y}) = c + \sum_{i=1}^{n} a_i y_i + \sum_{(i,j)\in\mathcal{E}} q_{ij} y_i y_j$$

be a quadratic real-valued function over binary variables $\mathbf{y} \in \{0,1\}^n$. If the readout contains the binary variables $\mathbf{y}$, or an affine representation from which they can be recovered, then a one-hidden-layer ReLU energy head with at most $n + |\mathcal{E}|$ hidden units can represent $Q(\mathbf{y})$ exactly on the binary domain. If the architecture includes skip connections from the readout input to the output layer, then $|\mathcal{E}|$ hidden units are sufficient. Moreover, no factorized linear score can represent a quadratic function with a nonzero interaction coefficient.

Proof. For binary variables,

$$y_i y_j = \max(0, y_i + y_j - 1).$$

Thus each quadratic interaction can be represented by one ReLU unit with pre-activation $y_i + y_j - 1$. The linear terms can be represented using additional ReLU units since $y_i = \max(0, y_i)$ for $y_i \in \{0,1\}$. The final linear layer combines these units with weights q_{ij} and a_i, and adds the bias c. Hence a one-hidden-layer ReLU network with at most $n + |\mathcal{E}|$ hidden units represents Q exactly on the binary domain.

To show that factorized linear scoring cannot represent nonzero interactions, fix all coordinates of $\mathbf{y}$ except y_i and y_j. For any factorized linear score

$$L(\mathbf{y}) = b + \sum_{\ell=1}^{n} w_\ell y_\ell,$$

the mixed finite difference with respect to coordinates i and j is

$$\Delta_{ij} L = L(\mathbf{y}^{11}) - L(\mathbf{y}^{10}) - L(\mathbf{y}^{01}) + L(\mathbf{y}^{00}) = 0,$$

where $\mathbf{y}^{ab}$ denotes the vector obtained by setting $y_i = a$ and $y_j = b$ while keeping all other coordinates fixed. For the quadratic interaction $q_{ij} y_i y_j$, the same finite difference equals q_{ij}. Therefore, if $q_{ij} \neq 0$, no factorized linear score can represent the function exactly. □

Proposition A.2 shows that the nonlinear energy head is not merely an implementation detail. It strictly enlarges the class of binary objectives representable by exact Neural-MIP inference beyond additive per-variable scoring.

B Benchmark Problem Formulations

We provide the complete MIP formulations for all three benchmarks. Instance generation procedures and dataset statistics are described in Appendix D.

B.1 Combinatorial Auctions (CA)

The combinatorial auction problem (Leyton-Brown et al., 2000) allocates a set of items $\mathcal{C}$ to a set of bids $\mathcal{B}$ so as to maximize total revenue. Each bid $b \in \mathcal{B}$ specifies a bundle $\mathcal{N}(b) \subseteq \mathcal{C}$ together with a value $p_b > 0$. Let $y_b \in \{0,1\}$ indicate whether bid b is accepted. The resulting 0–1 packing formulation is

$$\max_{\mathbf{y}\in\{0,1\}^{|\mathcal{B}|}} \sum_{b\in\mathcal{B}} p_b\, y_b \quad \text{s.t.} \quad \sum_{b:\, c\in\mathcal{N}(b)} y_b \leq 1 \quad \forall c \in \mathcal{C}. \tag{13}$$

The packing constraints enforce that each item is allocated to at most one accepted bid. The problem is NP-hard in general (Leyton-Brown et al., 2000).

We generate CA instances using the arbitrary-relations generator of Leyton-Brown et al. (2000). We consider a custom CA training distribution with $(|\mathcal{C}|, |\mathcal{B}|) = (100, 175)$ and evaluate cross-scale generalization on a harder test distribution with $(|\mathcal{C}|, |\mathcal{B}|) = (100, 250)$. Exact reference solutions are generated with a time limit of 3600 s per instance.

Graph encoding. We represent each instance as a bipartite graph with item nodes $\mathcal{C}$ and bid nodes $\mathcal{B}$. An edge (c, b) is present if and only if item c belongs to bid b, i.e., if $A_{cb} = 1$ in the incidence matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{C}|\times|\mathcal{B}|}$. Our encoder is therefore a bipartite message-passing neural network over constraint-side item nodes and variable-side bid nodes.

Features. For CA, item-node features encode local participation statistics (e.g., item degree and dummy-item indicators), while bid-node features encode bid value, bundle size, value-per-item, and a structural conflict proxy derived from competing bids. Both node types additionally receive a small amount of repeated global context, such as instance density and average bid statistics. Edge features encode normalized incidence-level signals linking item popularity, bid value, and bundle size.

Readout. Let $\mathbf{h}_b$ and $\mathbf{h}_c$ denote the final embeddings of bid b and item c, respectively. For each item, an auxiliary continuous variable u_c indicates whether it is occupied by at least one selected bid:

$$u_c = \mathbb{I}\left[\sum_{b:\, c\in\mathcal{N}(b)} y_b \geq 1\right]. \tag{14}$$

This relationship is represented exactly using

$$u_c \leq \sum_{b:\, c\in\mathcal{N}(b)} y_b, \qquad \forall c \in \mathcal{C}, \tag{15}$$

$$u_c \geq y_b, \qquad \forall c \in \mathcal{C}, \quad \forall b :\, c \in \mathcal{N}(b). \tag{16}$$

The corresponding occupied-item embedding pool is

$$\mathbf{h}_{\text{occ}}(\mathbf{y}) = \sum_{c\in\mathcal{C}} u_c \mathbf{h}_c. \tag{17}$$

The readout additionally includes the compact scalar-summary vector

$$\mathbf{s}_{\text{CA}}(\mathbf{y}) = \left[s_{\text{bid}}, s_{\text{occ}}, s_{\text{dummy}}, s_{\text{conf}}, s_{\text{mass}}, s_{\text{value}}, s_{\text{bundle}}, s_{\text{risk}}\right], \tag{18}$$

whose components are the selected-bid fraction, occupied-item fraction, occupied dummy-item fraction, conflicted-item fraction, normalized conflict mass, normalized selected-value sum, normalized selected-bundle-size sum, and normalized selected conflict-risk sum, respectively.

The complete CA readout is

$$\mathbf{z}^0_{\text{CA}}(\mathbf{y}) = \left[\sum_{b\in\mathcal{B}} y_b \mathbf{h}_b,\ \sum_{c\in\mathcal{C}} u_c \mathbf{h}_c,\ \mathbf{s}_{\text{CA}}(\mathbf{y}),\ \mathbf{h}_{\mathcal{G}}\right]. \tag{19}$$

The first pool represents the accepted bids, while the second captures their consequences on the item side of the bipartite graph. The scalar summaries provide compact information about selection density, item occupancy, aggregate value, bundle size, and conflict structure.

B.2 Quadratic Knapsack (QKP)

The Quadratic Knapsack Problem (Pisinger, 2007) selects a subset of n items to maximize a quadratic objective subject to a linear capacity constraint:

$$\max_{\mathbf{y}\in\{0,1\}^n} \ \sum_{i=1}^{n} p_i\, y_i \ + \sum_{i=1}^{n}\sum_{j=i+1}^{n} Q_{ij}\, y_i y_j \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i\, y_i \leq W, \tag{20}$$

where p_i is the linear profit of item i, Q_{ij} is the pairwise interaction profit between items i and j, w_i is the weight of item i, and W is the knapsack capacity. The quadratic term captures both positive and negative interactions between co-selected items, which motivates learning an energy function over complete configurations rather than independent per-item scores.

We consider instances with $n = 150$ items and interaction density $\rho = 0.30$. Linear profits are sampled as $p_i \sim \mathcal{U}[0, 10]$, item weights as $w_i \sim \mathcal{U}[1, 10]$, and the knapsack capacity is set to $W = 0.5\sum_i w_i$. Pairwise interactions are obtained from a symmetric random matrix with off-diagonal entries drawn from $\mathcal{U}[-2, 8]$, which is then sparsified to the target density ρ. Reference solutions are computed with Gurobi (Gurobi Optimization, LLC, 2023) using a time limit of 3600 s per instance.

Graph encoding. We represent each instance as a bipartite graph with item nodes I and interaction nodes P. Each interaction node $p = (i, j) \in P$ corresponds to a nonzero upper-triangular pairwise coefficient Q_{ij} with $i < j$, and is connected to both item nodes i and j. This construction allows message passing to propagate pairwise interaction information into the item embeddings used by the downstream readout.

Features. For QKP, item-node features encode linear profit, weight, interaction degree, and average interaction strength, together with repeated global context such as graph density and aggregate interaction scale. Interaction-node features encode the magnitude and sign of the pairwise term, along with simple summaries of the associated item weights and profits. Edge features transmit normalized interaction strength and capacity-relevant pair information.

Readout. Each QKP instance is represented as a bipartite graph between item nodes $\mathcal{I}$ and interaction nodes $\mathcal{P}$. Each interaction node $p = (i, j) \in \mathcal{P}$ corresponds to a nonzero upper-triangular pairwise coefficient Q_{ij} and is connected to item nodes i and j. The resulting item embeddings $\mathbf{h}_i$ therefore encode both item-level attributes and information propagated from the quadratic interactions.

Let $y_i \in \{0, 1\}$ indicate whether item i is selected. In addition to the selected-item embedding pool, QKP uses the unselected-item pool

$$\mathbf{h}_{\text{unsel}}(\mathbf{y}) = \sum_{i \in \mathcal{I}} (1 - y_i)\mathbf{h}_i, \tag{21}$$

the selected-item fraction

$$r_{\text{sel}}(\mathbf{y}) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} y_i, \tag{22}$$

and the normalized capacity usage

$$r_{\text{cap}}(\mathbf{y}) = \frac{1}{W} \sum_{i \in \mathcal{I}} w_i y_i. \tag{23}$$

The additional structured component is

$$\mathbf{z}_{\text{lin}}^{\text{QKP}}(\mathbf{y}) = \left[\mathbf{h}_{\text{unsel}}(\mathbf{y}),\ r_{\text{sel}}(\mathbf{y}),\ r_{\text{cap}}(\mathbf{y})\right]. \tag{24}$$

The complete QKP readout is

$$\mathbf{z}_{\text{QKP}}^{0}(\mathbf{y}) = \left[\sum_{i \in \mathcal{I}} y_i \mathbf{h}_i,\ \sum_{i \in \mathcal{I}} (1 - y_i)\mathbf{h}_i,\ \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} y_i,\ \frac{1}{W} \sum_{i \in \mathcal{I}} w_i y_i,\ \mathbf{h}_{\mathcal{G}}\right]. \tag{25}$$

The selected- and unselected-item pools expose information about both sides of the current partition of the item set. The two scalar summaries describe the size of the selected set and its consumption of the available knapsack capacity. All four decision-dependent components are affine in $\mathbf{y}$.

B.3 Personalized Medicine Supply Chain (CART)

The personalized medicine supply chain problem optimizes the strategic selection of cell therapy manufacturing facilities and the routing of patient therapies through a multi-echelon logistics network under strict capacity and turnaround-time constraints (Triantafyllou et al., 2022).

Problem structure. The problem decomposes naturally into two levels:

- **Strategic master problem**: select which subset of $|\mathcal{M}|$ candidate manufacturing facilities to open. This involves $|\mathcal{M}|$ binary variables $y_m \in \{0, 1\}$.
- **Operational subproblem**: given open facilities, route all patient therapies through the network to minimize total cost subject to capacity and timing constraints. This involves continuous and binary flow variables but becomes tractable once the facility set is fixed.

The monolithic MILP is intractable at scale because the number of patient routing variables grows as $|P| \times |C| \times |M| \times |J| \times |T|$, which reaches millions for realistic instances. EnerGNN learns an energy over facility location configurations $\mathbf{y} = \{y_m\}$ which is then optimized under the strategic feasibility constraints; the operational subproblem induced by the selected configuration is subsequently solved exactly by Gurobi.

Full MILP formulation. We provide the detailed mixed integer linear programming (MILP) formulation for the personalized medicine CAR T-cell therapy supply chain problem. The model jointly optimizes facility location, patient assignment, and time-dependent logistics under capacity and turnaround-time constraints.

Sets and indices Let $\mathcal{P}$, $\mathcal{C}$, $\mathcal{M}$, $\mathcal{H}$, $\mathcal{J}$, and $\mathcal{T}$ denote the sets of patients, clinical leukapheresis sites, candidate manufacturing facilities, hospitals, transportation modes, and discrete time periods, respectively. Indices are denoted by p, c, m, h, j, and t.

Parameters

- C_m^{inv}: capital investment cost for opening manufacturing facility m.
- C_m^{fix}: fixed operating cost of facility m.
- C_p^{var}: patient-specific variable manufacturing cost.
- C^{qc}: quality-control cost per therapy.
- K_m: maximum processing capacity, or number of parallel manufacturing lines, at facility m.
- δ_{pct}: binary parameter equal to 1 if patient p arrives at clinical site c at time t.
- τ^{lp}: duration of leukapheresis.
- τ^{man}: duration of manufacturing.
- τ^{qc}: duration of quality control.
- τ_j^{out}: transportation time from a clinical site to a manufacturing facility using mode j.
- τ_j^{ret}: transportation time from a manufacturing facility to a hospital using mode j.
- C_{cmj}^{out}: transportation cost from clinical site c to facility m using mode j.
- C_{mhj}^{ret}: transportation cost from facility m to hospital h using mode j.
- B: maximum number of manufacturing facilities that can be opened.
- D: maximum allowable vein-to-vein turnaround time.
- N_P: total number of patients in the planning horizon.

Variables with time indices outside $\mathcal{T}$ are fixed to zero.

Decision variables *Facility location variables.*

$$y_m \in \{0,1\}, \qquad \forall m \in \mathcal{M},$$

where $y_m = 1$ if manufacturing facility m is opened.

Network design variables.

$$u_{cm} \in \{0,1\}, \qquad \forall c \in \mathcal{C},\, m \in \mathcal{M},$$

where $u_{cm} = 1$ if clinical site c is connected to manufacturing facility m.

$$v_{mh} \in \{0,1\}, \qquad \forall m \in \mathcal{M},\, h \in \mathcal{H},$$

where $v_{mh} = 1$ if manufacturing facility m is connected to hospital h.

Transportation variables.

$$x_{pcmjt} \in \{0,1\}, \qquad \forall p \in \mathcal{P},\, c \in \mathcal{C},\, m \in \mathcal{M},\, j \in \mathcal{J},\, t \in \mathcal{T},$$

where $x_{pcmjt} = 1$ if patient p's material is transported from clinical site c to facility m using mode j at time t.

$$w_{pmhjt} \in \{0,1\}, \qquad \forall p \in \mathcal{P},\, m \in \mathcal{M},\, h \in \mathcal{H},\, j \in \mathcal{J},\, t \in \mathcal{T},$$

where $w_{pmhjt} = 1$ if therapy p is transported from facility m to hospital h using mode j at time t.

Flow variables.

$$\ell^{\mathrm{dep}}_{pcmjt} \geq 0, \qquad \ell^{\mathrm{arr}}_{pcmjt} \geq 0,$$

denote, respectively, material leaving clinical site c for facility m, and material arriving at facility m, for patient p, mode j, and time t.

$$q^{\mathrm{in}}_{pmt} \geq 0, \qquad q^{\mathrm{out}}_{pmt} \geq 0,$$

denote, respectively, therapy p entering and leaving manufacturing facility m at time t.

$$g^{\mathrm{out}}_{pct} \geq 0, \qquad g^{\mathrm{in}}_{pht} \geq 0,$$

denote, respectively, material leaving clinical site c, and final therapy arriving at hospital h, for patient p at time t.

Timing variables.

$$s_p, e_p, r_p \geq 0, \qquad \forall p \in \mathcal{P},$$

denote the start time, completion time, and turnaround time of patient p.

Objective function The objective minimizes total supply chain cost, including facility investment, fixed operating cost, patient-specific manufacturing cost, quality control, and transportation:

$$\begin{aligned} \min \quad Z = & \sum_{m\in\mathcal{M}} \left(C^{\mathrm{inv}}_m + C^{\mathrm{fix}}_m\right) y_m + \sum_{p\in\mathcal{P}} \left(C^{\mathrm{var}}_p + C^{\mathrm{qc}}\right) \\ & + \sum_{p\in\mathcal{P}} \sum_{c\in\mathcal{C}} \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{T}} C^{\mathrm{out}}_{cmj} x_{pcmjt} \\ & + \sum_{p\in\mathcal{P}} \sum_{m\in\mathcal{M}} \sum_{h\in\mathcal{H}} \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{T}} C^{\mathrm{ret}}_{mhj} w_{pmhjt}. \end{aligned} \tag{26}$$

Equivalently, if facility investment and fixed operating costs are allocated per patient as in the original implementation, the patient-level manufacturing cost may be written as

$$C^{\mathrm{man}}_p = \frac{\sum_{m\in\mathcal{M}} \left(C^{\mathrm{inv}}_m + C^{\mathrm{fix}}_m\right) y_m}{N_P} + C^{\mathrm{var}}_p, \qquad \forall p \in \mathcal{P}. \tag{27}$$

Constraints *Leukapheresis completion.* A patient sample can leave the clinical site only after the leukapheresis duration has elapsed:

$$g^{\mathrm{out}}_{pc,t+\tau^{\mathrm{lp}}} = \delta_{pct}, \qquad \forall p \in \mathcal{P},\, c \in \mathcal{C},\, t \in \mathcal{T}. \tag{28}$$

Outbound shipment selection. Each patient is assigned to exactly one outbound shipment:

$$\sum_{c\in\mathcal{C}} \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{T}} x_{pcmjt} = 1, \qquad \forall p \in \mathcal{P}. \tag{29}$$

The outbound shipment variables define the material leaving the clinical site:

$$\ell^{\mathrm{dep}}_{pcmjt} = x_{pcmjt}, \qquad \forall p, c, m, j, t. \tag{30}$$

All material leaving a clinical site must be assigned to a manufacturing facility and a transportation mode:

$$g^{\mathrm{out}}_{pct} = \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}} \ell^{\mathrm{dep}}_{pcmjt}, \qquad \forall p, c, t. \tag{31}$$

Outbound transportation timing. Material arriving at the manufacturing facility is delayed by the transportation time of the selected mode:

$$\ell^{\mathrm{arr}}_{pcmj,t+\tau^{\mathrm{out}}_j} = \ell^{\mathrm{dep}}_{pcmjt}, \qquad \forall p, c, m, j, t. \tag{32}$$

The total material entering a manufacturing facility equals all arrivals from clinical sites:

$$q^{\text{in}}_{pmt} = \sum_{c\in\mathcal{C}} \sum_{j\in\mathcal{J}} \ell^{\text{arr}}_{pcmjt}, \qquad \forall p, m, t. \tag{33}$$

Manufacturing and quality control timing. A therapy can leave the manufacturing facility only after manufacturing and quality control are completed:

$$q^{\text{out}}_{pm,t+\tau^{\text{man}}+\tau^{\text{qc}}} = q^{\text{in}}_{pmt}, \qquad \forall p, m, t. \tag{34}$$

Return shipment after manufacturing. Every completed therapy must be assigned to exactly one return shipment:

$$\sum_{h\in\mathcal{H}} \sum_{j\in\mathcal{J}} w_{pmhjt} = q^{\text{out}}_{pmt}, \qquad \forall p, m, t. \tag{35}$$

Hospital arrival timing. The final therapy arrives at the hospital after the return transportation delay:

$$g^{\text{in}}_{ph,t+\tau^{\text{ret}}_j} = \sum_{m\in\mathcal{M}} w_{pmhjt}, \qquad \forall p, h, j, t. \tag{36}$$

Each patient receives exactly one final therapy delivery:

$$\sum_{h\in\mathcal{H}} \sum_{t\in\mathcal{T}} g^{\text{in}}_{pht} = 1, \qquad \forall p \in \mathcal{P}. \tag{37}$$

Manufacturing capacity. The number of therapies simultaneously occupying facility m cannot exceed the installed capacity:

$$\sum_{p\in\mathcal{P}} \left(\sum_{\tau=0}^{t} q^{\text{in}}_{pm\tau} - \sum_{\tau=0}^{t} q^{\text{out}}_{pm\tau} \right) \le K_m y_m, \qquad \forall m \in \mathcal{M},\, t \in \mathcal{T}. \tag{38}$$

Network design and facility activation. A clinical site can only be connected to an opened manufacturing facility:

$$u_{cm} \le y_m, \qquad \forall c \in \mathcal{C},\, m \in \mathcal{M}. \tag{39}$$

A manufacturing facility can only be connected to a hospital if the facility is opened:

$$v_{mh} \le y_m, \qquad \forall m \in \mathcal{M},\, h \in \mathcal{H}. \tag{40}$$

Outbound shipments are allowed only along selected clinical-site-to-facility links:

$$x_{pcmjt} \le u_{cm}, \qquad \forall p, c, m, j, t. \tag{41}$$

Return shipments are allowed only along selected facility-to-hospital links:

$$w_{pmhjt} \le v_{mh}, \qquad \forall p, m, h, j, t. \tag{42}$$

The total number of established manufacturing facilities is bounded:

$$\sum_{m\in\mathcal{M}} y_m \le B. \tag{43}$$

Start, completion, and turnaround times. The start time of each patient is determined by the clinical site arrival time:

$$s_p = \sum_{c\in\mathcal{C}} \sum_{t\in\mathcal{T}} t\, \delta_{pct}, \qquad \forall p \in \mathcal{P}. \tag{44}$$

The completion time is the arrival time of the final therapy at the hospital:

$$e_p = \sum_{h\in\mathcal{H}} \sum_{t\in\mathcal{T}} t\, g^{\text{in}}_{pht}, \qquad \forall p \in \mathcal{P}. \tag{45}$$

Turnaround time is defined as the time between patient arrival and therapy delivery:

$$r_p = e_p - s_p, \qquad \forall p \in \mathcal{P}. \tag{46}$$

The vein-to-vein turnaround time must not exceed the clinical deadline:

$$r_p \le D, \qquad \forall p \in \mathcal{P}. \tag{47}$$

Graph encoding. We represent each instance as a complete bipartite graph between clinical facility-demand nodes $\mathcal{C}$ and candidate manufacturing facility nodes $\mathcal{M}$. Demand-node features summarize the local patient arrival profile at each clinical site using aggregate demand statistics, including total demand, variability, peak demand, empirical quantiles, and seasonal Fourier components. Facility node features encode normalized capacity, operating cost rate, and demand-weighted transport statistics. Each edge (c, m) encodes the transport cost between clinical site c and facility m using a normalized scalar together with an RBF expansion. This compressed graph representation exposes the aggregate operational structure of the full supply chain without explicitly instantiating patient-level routing variables in the learned model.

Features. In the implementation used for our experiments, demand-node features have dimension 14 and include local demand statistics together with a small amount of repeated global context. Facility node features have dimension 9 and encode capacity, normalized fixed cost rate, and transport summaries, again augmented with repeated global context. Edge features have dimension 6, consisting of one normalized transport cost feature and a five-dimensional RBF expansion.

Readout. Each CART instance is represented as a complete bipartite graph

$$\mathcal{G}_{\mathrm{CART}} = (\mathcal{C} \cup \mathcal{M}, \mathcal{E}), \qquad \mathcal{E} = \mathcal{C} \times \mathcal{M},$$

where $\mathcal{C}$ denotes clinical facility-demand nodes and $\mathcal{M}$ denotes candidate manufacturing-facility nodes. The encoder returns demand-node embeddings $\mathbf{h}_c$, facility-node embeddings $\mathbf{h}_m$, and a global graph embedding $\mathbf{h}_{\mathcal{G}}$.

Let $y_m \in \{0, 1\}$ indicate whether manufacturing facility m is selected. For each demand node c, an auxiliary binary variable q_c indicates whether it is covered by at least one selected facility:

$$q_c = \mathbb{I}\left[\sum_{m \in \mathcal{M}(c)} y_m \geq 1\right], \tag{48}$$

where $\mathcal{M}(c)$ is the set of candidate facilities connected to demand node c. This logical relationship is represented exactly through

$$q_c \leq \sum_{m \in \mathcal{M}(c)} y_m, \qquad \forall c \in \mathcal{C}, \tag{49}$$

$$q_c \geq y_m, \qquad \forall c \in \mathcal{C}, \quad \forall m \in \mathcal{M}(c). \tag{50}$$

The additional decision-dependent component is the pooled embedding of the covered demand nodes:

$$\mathbf{z}_{\mathrm{lin}}^{\mathrm{CART}}(\mathbf{y}) = \sum_{c \in \mathcal{C}} q_c \mathbf{h}_c. \tag{51}$$

The complete CART readout is therefore

$$\mathbf{z}_{\mathrm{CART}}^{0}(\mathbf{y}) = \left[\sum_{m \in \mathcal{M}} y_m \mathbf{h}_m,\ \sum_{c \in \mathcal{C}} q_c \mathbf{h}_c,\ \mathbf{h}_{\mathcal{G}}\right]. \tag{52}$$

The first term represents the selected manufacturing facilities, while the second represents the clinical facility-demand nodes covered by the selected facility set. CART therefore uses pooled learned embeddings from both sides of the bipartite graph and does not include additional manually constructed scalar summaries.

C Neural-MIP Encoding

We encode the learned ReLU energy head exactly inside the MIP using a standard Big-M formulation for ReLU activations. The benchmark-specific combinatorial constraints and the readout-level auxiliary variables are then combined with this head encoding to yield the final inference model.

C.1 ReLU Head Encoding

Let

$$a_j^l = \mathbf{w}_j^\top \mathbf{h}^{l-1} + b_j$$

denote the affine pre-activation of neuron j in layer l, and let

$$h_j^l = \max(0, a_j^l)$$

be the corresponding ReLU activation. Suppose valid finite bounds

$$\underline{a}_j^l \le a_j^l \le \overline{a}_j^l$$

are available. For neurons whose bounds satisfy $\underline{a}_j^l < 0 < \overline{a}_j^l$, we introduce a binary activation indicator $z_j^l \in \{0, 1\}$ and impose the exact mixed integer linear formulation

$$h_j^l \ge a_j^l, \tag{53}$$

$$h_j^l \ge 0, \tag{54}$$

$$h_j^l \le a_j^l - \underline{a}_j^l (1 - z_j^l), \tag{55}$$

$$h_j^l \le \overline{a}_j^l z_j^l. \tag{56}$$

If $\underline{a}_j^l \ge 0$, the neuron is always active and we set $h_j^l = a_j^l$. If $\overline{a}_j^l \le 0$, the neuron is always inactive and we set $h_j^l = 0$. The final linear layer is encoded by linear equality constraints, producing the scalar energy minimized by the inference MIP.

C.2 Interval Bound Propagation

The practical strength of the Neural-MIP depends on the tightness of the pre-activation bounds. We obtain these bounds using interval bound propagation through the assembled readout and energy head.

For an affine layer

$$\mathbf{a}^l = W^l \mathbf{h}^{l-1} + \mathbf{b}^l$$

with elementwise input bounds $[\underline{\mathbf{h}}^{l-1}, \overline{\mathbf{h}}^{l-1}]$, the pre-activation bounds are

$$\underline{a}_j^l = b_j^l + \sum_i \min\left(W_{ji}^l \underline{h}_i^{l-1}, W_{ji}^l \overline{h}_i^{l-1} \right), \tag{57}$$

$$\overline{a}_j^l = b_j^l + \sum_i \max\left(W_{ji}^l \underline{h}_i^{l-1}, W_{ji}^l \overline{h}_i^{l-1} \right). \tag{58}$$

The corresponding post-ReLU bounds are

$$\underline{h}_j^l = \max(0, \underline{a}_j^l), \qquad \overline{h}_j^l = \max(0, \overline{a}_j^l).$$

These pre-activation bounds are used directly in the exact ReLU formulation above. When the bounds imply a fixed activation state, the corresponding binary variable is removed. Overall, the benchmark-specific readout linearizations and subsequent interval propagation provide the neuron-wise pre-activation bounds used in exact MIP inference.

C.3 Benchmark-Specific Input Bounds

The input intervals for this propagation step are derived from the exact linearized readout features used by each benchmark.

CART. For CART, the readout combines pooled selected facility embeddings, pooled covered-demand embeddings, and a global graph embedding. Bounds for the selected-facility pooling terms are obtained from the facility cardinality constraints $k_{\min} \le \|\mathbf{y}\|_1 \le k_{\max}$. Coverage terms are linearized using auxiliary binary variables indicating whether each demand node is served by at least one selected facility, and their bounds are computed from sign-split sums over the corresponding coefficients.

CA. For CA, the readout includes pooled selected bid and occupied item embeddings together with scalar summaries such as selected bid fraction, occupied item fraction, and aggregate selected bid statistics. These are encoded with auxiliary continuous occupancy variables and count-based one-hot linearizations, which allow exact bounds to be propagated through the resulting MLP input.

QKP. For QKP, the readout pools selected item embeddings and, in the size-aware variant, also includes unselected item summaries, selected fraction, and used capacity mass. Under the knapsack constraint, selected pool bounds are tightened using a fractional knapsack relaxation rather than naive unconstrained cardinality intervals. This yields substantially tighter input bounds for the downstream ReLU encoding.

Overall, the benchmark-specific readout linearizations and the subsequent interval propagation provide the neuron-wise Big-M constants used in exact MIP inference.

C.4 Problem-Specific Hard Feasibility Constraints

Beyond the energy encoding, the following hard constraints are added to the inference MIP for each benchmark:

CART.

$$\sum_{m} y_m \leq k_{\max}, \tag{59}$$

$$\sum_{m} \kappa_m y_m \geq \kappa_{\mathrm{ref}}, \tag{60}$$

$$y_j \leq y_i \qquad \forall (i,j) \in \mathcal{S}. \tag{61}$$

where (59) opens at most $k_{\max}$ facilities, (60) requires total open capacity to meet the peak reference demand κ_{ref}, and (61) imposes symmetry-breaking cuts over pairs of identical capacity facilities $\mathcal{S}$.

CA.

$$\sum_{b: c \in \mathcal{N}(b)} y_b \leq 1 \qquad \forall c \in \mathcal{C}. \tag{62}$$

where (62) enforces that each item is allocated to at most one bid. Zero-shot injection experiments additionally introduce cardinality bounds $\sum_b y_b \leq K$ and bidder exclusion constraints $y_i + y_j \leq 1$ for specified pairs (i, j).

QKP.

$$\sum_{i} w_i y_i \leq W \tag{63}$$

where (63) enforces that the total selected weight remains within the budget W.

D Dataset Splits and Candidate Pool Construction

Dataset splits. For each benchmark, we use 800 training instances, 100 validation instances, and 100 held-out test instances.

For each training instance, we construct a candidate pool $\mathcal{Y} = \mathcal{F} \cup \mathcal{I}$ containing feasible and infeasible configurations. Pool construction is benchmark-specific.

CART. For CART, each instance has 16 candidate facilities. We first solve the unrestricted patient-level MIP to global optimality and collect incumbent facility selection configurations from branch-and-bound. Additional candidates are generated using capacity- and transport-guided heuristics, local Hamming perturbations, swaps, facility additions and removals, and random configurations with varying cardinality. Each candidate is evaluated by solving the corresponding tactical patient scheduling MIP. Feasible configurations attaining the proven global optimum within numerical tolerance are labeled positive, other feasible configurations are labeled feasible negatives, and

configurations certified infeasible are labeled infeasible negatives. We retain 160 candidates per instance through stratified selection, initially keeping up to three positives and targeting 30% infeasible negatives among the remaining candidates.

CA. For CA, we first solve each instance with Gurobi and collect incumbent solutions from branch-and-bound. Feasible solutions with relative primal gap at most $\epsilon_{\text{pos}} = 0.01$ are labelled positive, up to a maximum of 50 positives per instance. We then construct 64 feasible negatives using a quota-driven mixture of additional incumbent solutions, guided local Hamming perturbations, and farther Hamming perturbations of the best solution. The feasible-negative selection is balanced across Hamming distance buckets and primal gap bands. Finally, we generate 50 infeasible negatives by perturbing feasible masks into different violation bands, corresponding to mild, moderate, and severe item conflicts.

QKP. For QKP, we impose a one-hour time limit on Gurobi per instance. Training and validation may include instances for which global optimality is not certified within this limit. In contrast, the test set contains instances with Gurobi certified global optima, allowing all reported primal gaps to be measured against proven optimal solutions. For each training instance, we retain the best available solution together with all incumbent solutions whose relative primal gap is at most $\epsilon_{\text{pos}} = 0.01$; these form the positive subset. We then generate 128 feasible negatives and 64 infeasible negatives using stochastic mask proposals. Each proposal is either a random binary mask with random density or a mutation of the best mask obtained by flipping a random subset of items. Proposed masks that satisfy the knapsack constraint are retained as feasible negatives, while those that violate the capacity constraint are retained as infeasible negatives.

D.1 Offline Computational Cost and Amortization

The main benchmark runtimes report the online inference and optimization phase. EnerGNN also incurs a one-time offline cost for solver-based candidate generation and neural model training. Table 6 reports the complete elapsed wall-clock cost of this offline phase. Candidate generation was distributed across 1000 submitted jobs, with approximately 80 jobs running concurrently, subject to scheduler availability. Reported elapsed wall-clock times include queueing effects and therefore depend on cluster availability.

Table 6: Offline computational cost and approximate amortization break-even points.

Benchmark	Label generation	Training	Total offline	Break-even instances
QKP	14.7 h	3.39 h	18.1 h	≈ 125
CA	0.12 h	5.54 h	5.66 h	Not reached
CART	123.2 h	3.52 h	126.72 h	≈ 130

For a benchmark where EnerGNN is faster online than solving the full problem, the approximate number of deployment instances required to recover the offline investment is calculated as

$$N_{\text{break-even}} = \frac{T_{\text{offline}}}{T_{\text{full}} - T_{\text{EnerGNN}}},$$

where T_{offline} includes label generation and training, T_{full} is the mean time required to solve a new instance using the original optimisation formulation, and T_{EnerGNN} is the mean online EnerGNN time.

The offline investment is recovered after approximately 125 QKP instances and 130 CART instances under the reported experimental conditions. No break-even point is reported for CA because Gurobi is already faster on the original auction formulation. These results make explicit that EnerGNN is intended for repeated decision-making settings in which the one-time solver supervision and training costs can be amortised across many deployment instances.

D.2 Computational Setup

Experiments were conducted using the Imperial College London CX3 high-performance computing cluster and a local workstation. Training, validation, and test datasets were generated on CPU

nodes with up to 250 GB of RAM. Neural models were trained offline on GPU nodes equipped with NVIDIA Quadro RTX 6000 GPUs, using one GPU, 4 CPU cores, and 24 GB of RAM. Test time optimization and Neural-MIP inference were performed on a workstation with 64 GB of RAM. Optimization runs used Gurobi 12.0.2.

Reported runtimes exclude dataset generation, candidate pool construction, and neural network training. For learning-based methods, reported online times include neural prediction or encoding, construction of the corresponding optimization problem, and all solver calls required to produce the final solution.

E Loss Configuration Details

E.1 Loss Configuration Used in Reported Experiments

All reported experiments use the same loss formulation, consisting of an optimality-aware contrastive loss term, a feasible calibration term, and an infeasibility shaping term. The optimality-aware contrastive term is applied over the full candidate pool. Across these runs, we use a bipartite MPNN encoder with hidden dimension 128, two message-passing rounds, and dropout 0.1. The reported runs use a shallow MLP energy head with one hidden layer of width 32.

E.2 Training and Model Selection

All models are trained using Adam with the benchmark-specific hyperparameters reported in Table 7. Model selection is performed on the validation set. For each run, we retain the checkpoint with the lowest validation mean primal gap and report final test results from this checkpoint.

Adaptive Huber scale. For each candidate pool, the Huber transition scale is computed as

$$\delta_{\mathcal{F}} = \min\left\{\delta_{\max},\ \max\left[\delta_{\min},\ \kappa_{\delta}\left(\max_{i\in\mathcal{F}} g_i - \min_{i\in\mathcal{F}} g_i\right)\right]\right\}. \tag{64}$$

Table 7: Training hyperparameters for the full EnerGNN runs reported in the paper. Here λ_{inf} denotes the infeasible shaping weight, T is the prediction softmax temperature, τ_{rank} is the target ranking temperature, δ_{inf} is the infeasible Huber transition scale.

Hyperparameter	CART	CA	QKP
Learning rate	2×10^{-6}	1×10^{-6}	1×10^{-5}
Weight decay	1×10^{-5}	1×10^{-5}	1×10^{-5}
Epochs	2000	2000	2000
λ_{cal}	1.0	0.01	0.05
λ_{inf}	0.1	0.1	0.1
T	0.1	0.2	0.5
τ_{rank}	0.05	0.005	0.05
δ_{inf}	0.1	0.1	0.1
κ_{δ}	0.15	0.15	0.15
$\delta_{\min}$	0.01	0.01	0.01
$\delta_{\max}$	0.5	0.5	0.5

F Scale of the CART Formulation

The number of strategic variables alone understates the computational scale of the CART benchmark. Each strategic facility selection controls a large patient-level assignment and scheduling formulation. Across the 100 test instances with $m = 16$, the dimensions of the complete monolithic formulations are reported in Table 8.

Table 8: Dimensions of the complete monolithic CART formulations in the extended $m = 16$ test set.

Model component	Minimum	Maximum	Mean
Constraints	2,284,802	92,139,833	46,766,602
Binary variables	532,496	21,532,176	10,928,170
Continuous variables	1,177,490	47,445,565	24,082,020
General integer variables	8,336	336,456	170,768
Total variables	1,718,322	69,314,197	35,180,958

EnerGNN selects 3.21 of the 16 candidate manufacturing facilities on average. Fixing the strategic facility set substantially reduces the induced patient-level scheduling formulations, as shown in Table 9.

Table 9: Mean dimensions of the complete and EnerGNN-reduced CART formulations in the extended $m = 16$ setting.

Model component	Full model mean	EnerGNN-reduced mean	Reduction (%)
Constraints	46,766,602	10,819,588	76.9
Binary variables	10,928,170	2,434,726	77.7
Continuous variables	24,082,020	5,499,214	77.2
General integer variables	170,768	170,756	0.01
Total variables	35,180,958	8,104,696	77.0

The numbers of constraints, binary variables, continuous variables, and total variables are reduced by approximately 77% on average. The number of general integer variables changes very little because these variables are primarily associated with patients and scheduling periods rather than with the candidate manufacturing facilities.

G Primal Gap Distributions

Figure 3 shows the primal gap distribution across test instances for all learning-based methods on each benchmark. The x-axis uses a log scale to show both near zero and high gap distributions simultaneously. EnerGNN maintains a tight distribution near zero across all three benchmarks, confirming that the means reported in Table 1 are representative. Primal gaps are computed relative to the global optimum.

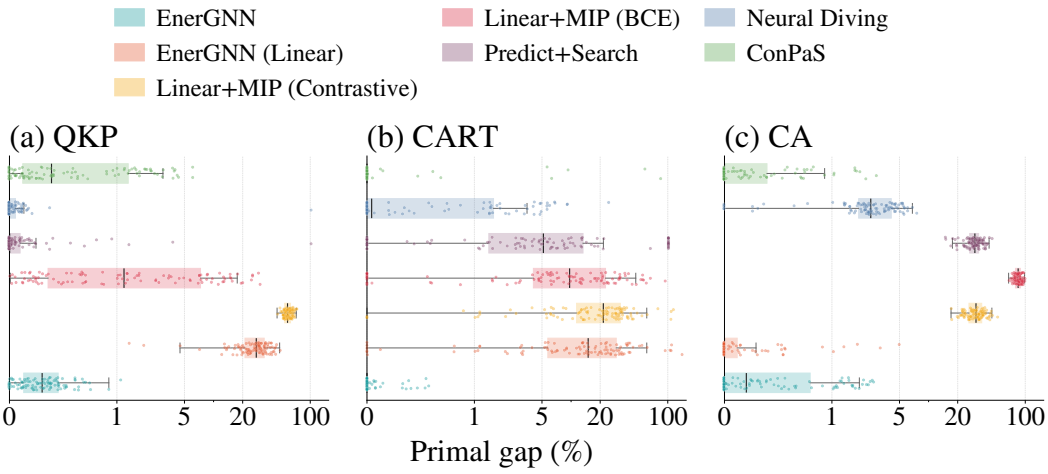


Figure 3: Primal gap distributions for all learning-based methods. Infeasible outputs are assigned a 100% gap penalty.

H Additional Zero-Shot Constraint Injection Results

We provide additional zero-shot constraint injection experiments in which the trained energy is unchanged and only the feasible region of the inference MIP is modified. In CART, cardinality and facility activation constraints restrict the available strategic configurations. We use the linear head EnerGNN model for CA, as it achieved the best performance in the main experiments. Optimality gaps are measured relative to reference optima computed under the same injected constraints. These experiments complement the stronger constraint shifts reported in Section 4, where variables selected in the original optimal solution are explicitly made unavailable.

Table 10: Additional zero-shot constraint injection results on CART.

Injected constraint	Primal Gap (%)	$\leq 1\%$ (%)
$\sum_m y_m \leq 6$	0.06	100
$\sum_m y_m \leq 5$	0.07	100
$\sum_m y_m \leq 4$	0.12	98
$\sum_m y_m \leq 3$	0.30	93
$y_{m_{10}} = 0$	0.05	100
$y_{m_{10}} = 0,\ y_{m_{13}} = 0$	0.03	100
$y_{m_7} = 0,\ y_{m_{10}} = 0,\ y_{m_{13}} = 0$	0.09	99
$y_{m_0} = 0,\ y_{m_3} = 0,\ y_{m_6} = 0$	0.07	99
$y_{m_0} = 1$	0.09	99
$y_{m_{10}} = 1$	1.75	79
$y_{m_0} = 1,\ y_{m_{10}} = 1$	0.00	100
$y_{m_{10}} = 0,\ y_{m_{13}} = 1$	0.99	89
$y_{m_{10}} = 0,\ y_{m_{13}} = 0,\ y_{m_1} = 1,\ y_{m_7} = 1$	4.83	54
$\sum_m y_m \leq 4, y_{m_{10}} = 0,\ y_{m_{13}} = 0,\ y_{m_1} = 1,\ y_{m_7} = 1$	3.55	55
$\sum_m y_m \leq 5, y_{m_0} = 1,\ y_{m_{10}} = 1$	0.00	100

Table 11: Additional zero-shot constraint injection results on CA.

Injected constraint	Primal Gap (%)	$\leq 1\%$ (%)
$\sum_b y_b \leq 30$	0.20	95
$\sum_b y_b \leq 20$	0.22	94
$\sum_b y_b \leq 15$	0.24	92
$\sum_b y_b \leq 10$	0.17	96
$\sum_b y_b \leq 5$	0.14	94

I Cross-scale Generalization with Nonlinear Energy Head on Combinatorial Auctions

For CA, the main paper reports cross-scale generalization using the linear head variant of EnerGNN, which achieves the best performance on this benchmark. For completeness, Table 12 reports the corresponding results using the nonlinear ReLU energy head, evaluated without retraining.

Table 12: CA cross-scale generalization with the nonlinear energy head.

Train $\rightarrow$ test	**Primal Gap (%)**	**$\leq$1% (%)**	**Feasibility (%)**
$175 \rightarrow 175$	0.53	77	100
$175 \rightarrow 250$	1.40	45	100

J Ablation Study Details

J.1 Energy Head Architecture Ablation

We evaluate the effect of energy head width and depth on QKP and CART. In all cases, the resulting Neural-MIP is solved to global optimality with respect to the learned energy. Increasing the width

or depth of the embedded ReLU head adds activation binaries to the inference MIP. Across both benchmarks, larger heads do not systematically improve solution quality, while deeper and wider architectures can substantially increase solution time.

Table 13: Energy head architecture ablation on QKP.

Head	Layers	ReLU binaries	Total binaries	Gap (%)	$\leq 1\%$ (%)	Time (s)
MLP-16	1	16	166	0.47	97	1.07
MLP-16	2	32	182	0.31	100	2.77
MLP-32	1	32	182	0.31	99	3.56
MLP-32	2	64	214	0.35	100	19.78
MLP-64	1	64	214	0.42	99	4.72
MLP-64	2	128	278	0.37	99	186.23

Table 14: Energy head architecture ablation on CART.

Head	Layers	ReLU binaries	Total binaries	Gap (%)	Time (s)
MLP-16	1	16	36	0.09	0.59
MLP-16	2	32	52	0.13	1.33
MLP-32	1	32	52	0.05	1.88
MLP-32	2	64	84	0.02	3.55
MLP-64	1	64	84	0.10	3.11
MLP-64	2	128	148	0.07	24.69

For QKP, increasing head capacity does not consistently improve solution quality. For example, the one-layer width-16 model achieves a 0.47% mean gap in 1.07 s, while the two-layer width-64 model achieves a similar 0.37% gap but requires 186.23 s. The same pattern appears in CART. Increasing from a one-layer width-16 head to a two-layer width-64 head raises inference time from 0.59 s to 24.69 s, while the mean gap changes only from 0.09% to 0.07%.

These results illustrate the expressiveness–tractability trade-off of the Neural-MIP formulation. Increasing the capacity of the embedded energy head adds ReLU binaries and can substantially increase solution time without a corresponding improvement in solution quality. Importantly, the GNN encoder is evaluated before optimization and is therefore not part of the inference MIP. Representational capacity can consequently be placed primarily in the graph encoder while keeping the optimization-compatible energy head compact.

J.2 Loss Variant Ablation

We evaluate three loss configurations on each benchmark:

- **Full loss**: all three components active ($\mathcal{L}_{\text{con}} + \mathcal{L}_{\text{cal}} + \mathcal{L}_{\text{inf}}$).
- **No primal gap term**: $\lambda_{\text{cal}} = 0$, contrastive and infeasibility only.
- **No infeasibility term**: $\lambda_{\text{inf}} = 0$, contrastive and calibration only.

J.3 Strict Feasible–Infeasible Dominance

The standard training objective does not enforce that every feasible candidate must have lower energy than every infeasible candidate. To test whether such a complete ordering would be beneficial, we evaluate a stricter alternative. Let

$$E^F_{\max} = \max_{i \in \mathcal{F}} E_\theta(G, \mathbf{y}_i)$$

denote the highest feasible energy in the current candidate pool. During backpropagation, $E^F_{\max}$ is treated as constant. We replace the original infeasibility shaping term with

$$\mathcal{L}_{\text{inf-margin}} = \frac{1}{|\mathcal{I}|} \sum_{k \in \mathcal{I}} \left[E^F_{\max} + \log(1 + \nu_k) - E_\theta(G, \mathbf{y}_k) \right]_+ ,$$

where $[a]_+ = \max(0, a)$ and ν_k is the normalized constraint violation. This term penalizes an infeasible candidate unless its energy lies at least $\log(1 + \nu_k)$ above the highest energy feasible candidate in the sampled candidate pool, thereby explicitly enforcing feasible–infeasible ordering within that pool.

We evaluate this alternative under the same Combinatorial Auctions protocol using EnerGNN with the linear head, which is the best performing EnerGNN variant for CA.

Table 15: Effect of enforcing strict feasible–infeasible dominance on CA.

Infeasibility loss	**Mean gap (%)**	**Exact optimum (%)**	$\leq 1\%$ **(%)**	**Feasible (%)**	**Mean time (s)**
Current calibration	0.20	67	95	100	0.28
Strict dominance margin	3.96	4	18	100	0.25

The stricter loss enforces the desired ordering but substantially reduces solution quality: the mean gap increases from 0.20% to 3.96%, exact recovery decreases from 67% to 4%, and the fraction of solutions within 1% of optimality decreases from 95% to 18%. Both variants return feasible solutions on every test instance.

This result clarifies the role of infeasibility supervision. The ranking and calibration terms are not intended to learn a global feasibility certificate, feasibility with respect to the encoded constraints is enforced by the inference MIP. Complete feasible–infeasible separation is therefore not required when those constraints are explicitly encoded, and in this experiment enforcing it substantially degrades optimization quality.

J.4 Readout Design Ablation

We evaluate how the choice of MIP-compatible readout affects solution quality and inference time. The proposed readouts combine pooled learned embeddings with compact structural summaries. We compare them with simpler embedding-only readouts and with variants that concatenate the complete binary decision vector $\mathbf{y}$ directly to the learned embedding pools. The latter reduces the need for structured summaries but produces an order-dependent, dimension-specific input.

Table 16: Readout design ablation on CA and QKP.

Problem	**Readout**	**Mean gap (%)**	**Exact (%)**	$\leq 1\%$ **(%)**	**Time (s)**
CA	Embedding pools only	0.20	68	95	0.25
CA	Full proposed readout	0.20	67	95	0.28
CA	Embedding pools + full $\mathbf{y}$	0.21	67	95	0.25
QKP	Selected item pool only	0.81	0	72	0.50
QKP	Full proposed readout	0.31	4	99	3.56
QKP	Selected item pool + full $\mathbf{y}$	0.94	0	61	1.17

For CA, removing the compact scalar summaries leaves performance essentially unchanged, while directly exposing the complete decision vector provides no improvement. For QKP, the structured readout is substantially more effective: the mean gap decreases from 0.81% to 0.31% and the fraction within 1% increases from 72% to 99%. Directly concatenating the full decision vector performs worse.

These results indicate that EnerGNN does not require a large collection of hand-crafted features. A practical readout should remain compact, permutation-invariant where possible, compatible with varying problem sizes, and exactly MIP-representable. Pooled learned embeddings can provide most of the representation, while compact structural summaries are useful when they capture meaningful consequences of the selected configuration.

K Baseline Implementation Details

All neural methods use the same bipartite GNN encoder architecture, with method-specific prediction or energy heads. We describe these heads, training objectives, and inference procedures below.

Predict+Search. Predict+Search applies a single linear layer to each variable embedding, followed by a sigmoid, to predict its probability p_j of taking value one. The model is trained using binary cross entropy supervision on the variable assignments of positive candidate solutions. At inference, we rank variables by confidence $|p_j - 0.5|$ and fix the most confident 85% for QKP and 80% for CA to their predicted values $\mathbf{1}[p_j \geq 0.5]$. For CART, we instead use a two-sided confidence band, where facility variables with $p_j \geq 0.6$ are fixed to one, those with $p_j \leq 0.4$ are fixed to zero, and the remaining facility variables are left free. We then solve the reduced MIP.

Linear+MIP. Linear+MIP applies a single linear layer to each variable embedding to produce a scalar score s_j. These scores define an additive objective $s(y) = \sum_j s_j y_j$, which is maximized under the original hard constraints. We evaluate two training variants with the same architecture: (i) binary cross entropy supervision on variable assignments, using $\sigma(s_j)$ as the predicted probability, and (ii) contrastive supervision over candidate solutions, using the energy $E(y) = -s(y)$. At inference, both variants optimize the learned linear objective over all decision variables using a MIP solver.

Neural Diving. Our Neural Diving (Nair et al., 2020) baseline uses a two-layer variable-level MLP head, with one hidden layer of width d_h, a ReLU activation, and dropout, followed by a sigmoid, to predict an independent Bernoulli probability p_j for each binary decision variable. The model is trained with binary cross entropy supervision on the variable assignments of positive candidate solutions. At inference, we perform K stochastic dives by sampling candidate assignments from these Bernoulli probabilities. For each dive, a confidence-selected subset of variables is fixed to the sampled assignment, and the reduced MIP is optimized over the remaining free variables. The final solution is the best feasible solution found across all dives. We use $K = 20$ dives for QKP, $K = 8$ for CART, and $K = 100$ for CA. The smaller dive budget for CART reflects the higher computational cost of solving its reduced MIPs. For QKP and CA, we fix the most confident 85% and 50% of variables, respectively. For CART, because the number of facility variables is small, we instead use the same two-sided confidence band as Predict+Search: variables with $p_j \geq 0.6$ or $p_j \leq 0.4$ are fixed to their sampled values, and the remaining variables are left free.

ConPaS. Our controlled ConPaS-style (Huang et al., 2024) baseline applies a 3-layer variable-level MLP head, with 2 hidden layers of width 64 and ReLU activations, followed by a sigmoid, to predict independent Bernoulli probabilities p_j for the binary decision variables. It scores a candidate assignment by its factorized log probability,

$$s(y) = \sum_j y_j \log p_j + (1 - y_j) \log(1 - p_j).$$

Training uses an InfoNCE contrastive loss over the candidate pool, encouraging positive solutions to receive higher scores than suboptimal feasible and infeasible candidates. At inference, the neural model is not embedded in the MIP. Instead, variables are ranked by p_j. Let X_0 denote the k_0 lowest probability variables and X_1 the k_1 highest probability variables. We solve the original MIP with the additional rank-neighbourhood constraint

$$\sum_{j \in X_0} y_j + \sum_{j \in X_1} (1 - y_j) \leq \Delta,$$

which allows at most Δ disagreements with the predicted low/high probability sets while leaving all other variables unrestricted. Increasing Δ relaxes this rank-neighbourhood restriction by allowing more disagreements with the neural ranking. This can improve solution quality under constraint shift, where variables predicted to be unlikely may become necessary because of exogenous constraints, but it also enlarges the MIP search space. We therefore tune Δ per benchmark to balance solution quality and solve time. The rank-neighbourhood parameters are $(k_0, k_1, \Delta) = (55, 20, 5)$ for QKP, $(8, 2, 1)$ for CART, and $(88, 0, 3)$ for CA. For CA, we keep the paper-scaled choice $k_0 = \lfloor n/2 \rfloor$ used by ConPaS. The original in-distribution ConPaS setting uses $\Delta = 0$. However, under our constraint shift setting, exogenous outage and force-on constraints can make a small number of low-ranked bids

necessary. We therefore introduce a small slack of $\Delta = 3$. We also evaluated the stricter paper-style setting $(88, 0, 0)$, but it produced poor solution quality, with a mean gap of 6.23%, despite being fast, with an average solve time of 0.05s compared with Gurobi's 0.14s.

Broader Impact

This work develops learning-augmented optimization methods for constrained combinatorial decision-making. A potential positive impact is to make learned decision systems more reliable in settings where feasibility is essential, such as logistics, supply chain planning, manufacturing, and healthcare operations. Rather than producing decisions directly, EnerGNN learns an objective landscape that is optimized under explicit hard constraints. This may help decision-support tools adapt to changing demand, capacity, or operational requirements while preserving feasibility through the optimization layer. In personalized medicine supply chains, such methods could support the evaluation of strategic planning decisions under different patient demand and capacity scenarios. More broadly, the approach may be useful in domains where learned models must interact with exact algorithms to produce valid decisions. The method is not intended to replace domain expertise, clinical judgement, or human oversight in high-stakes settings. Exact constrained inference preserves feasibility with respect to the specified constraints, but it does not guarantee that the learned energy ranks all decisions correctly or that the specified objective captures all clinical, ethical, or operational priorities. If deployed without careful validation, a learned energy could favor suboptimal allocations or reinforce undesirable trade-offs.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state that EnerGNN learns MIP-representable energy functions over binary decisions, performs constrained inference over the learned energy, and supports feasibility, generalization, and zero-shot constraint injection. These claims are supported by the methodology, theoretical results, and experiments in Sections 3 and 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The experimental section frames the benchmarks as testing different capabilities rather than providing a uniform speed comparison. It states that CA is not a solver-acceleration benchmark and that CART is a strategic-decomposition setting. Section 4.8 further discusses the expressiveness–tractability trade-off of the embedded energy head, the offline cost of solver-generated supervision, the distinction between learned-energy exactness and optimality for the original objective, and degradation under large distribution or feasible-set shifts.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: The assumptions for MIP-representability are stated in Section 3.5, including a fixed graph encoder, a mixed integer representable readout, and a ReLU energy head. Complete proofs are provided in Appendix A, including the ReLU encoding, feasible learned-energy inference, constraint injection, and expressivity beyond factorized linear scoring.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The paper describes the benchmark formulations, graph encodings, candidate-pool construction, loss terms, training hyperparameters, Neural-MIP encoding, and baseline implementations in the main text and appendices. These details provide the information needed to reproduce the experimental protocol to the extent required for the main claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The implementation, experimental configurations, and instructions for reproducing the main results are publicly available at `https://github.com/nikitrian/EnerGNN`. Additional details on benchmark formulations, dataset generation, training procedures, and baseline implementations are provided in the appendices.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The main text describes the benchmarks, baselines, metrics, and evaluation protocol. The appendices provide candidate-pool construction details, benchmark-specific training hyperparameters, Neural-MIP encoding details, and baseline implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer:[Yes].

   Justification: Primal gap distributions across test instances for all methods are reported in Appendix G, Figure 3, using box plots that show the interquartile range, whiskers at 1.5×IQR, and individual test instances as points. This characterizes the variability of primal gap across test instances. The within-1% recovery and exact optimality rates reported in Table 1 provide additional instance-level distributional evidence for all methods.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: Computational setup and runtime conventions are reported in Appendix D.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

Justification: The work develops and evaluates algorithms for constrained combinatorial optimization. The experiments use optimization benchmarks and supply chain planning instances, and we do not release sensitive personal data or deploy the method in a real decision-making system.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes].

Justification: The paper includes a broader impact discussion describing potential positive impacts for reliable constrained optimization in logistics, supply chain planning, and healthcare operations, as well as potential risks from deploying learned decision models in high-stakes allocation settings without domain validation, monitoring, and human oversight.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A].

Justification: The paper does not introduce or release high-risk generative models, scraped datasets, or pretrained models with obvious misuse potential. The method is an optimization framework evaluated on combinatorial optimization benchmarks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The paper cites the original sources for the benchmark problem classes, solver tools, and prior supply chain model used in the experiments, including Gurobi, combinatorial auctions, quadratic knapsack, and the personalized medicine supply chain formulation. No third-party datasets or models are redistributed as part of the current submission.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper does not release a new dataset, benchmark package, pretrained model, or other standalone asset as part of the current submission. The generated instances and experimental procedures are documented in the paper and appendices.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: The paper does not involve crowdsourcing experiments or research with human subjects. The experiments are computational optimization studies.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A].

Justification: The paper does not involve crowdsourcing or human-subject experiments. No participant recruitment, intervention, or collection of identifiable human-subject data was conducted for this study.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: The core method, experiments, and scientific contributions do not use LLMs as a methodological component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.