ML-Guided Primal Heuristics for Mixed Binary Quadratic Programs

Weimin Huang[1], **Natalie M. Isenberg**[2], **Ján Drgoňa**[3], **Draguna L Vrabie**[2], **Bistra Dilkina**[1]
[1]University of Southern California, USA
[2]Pacific Northwest National Laboratory, USA
[3]Johns Hopkins University, USA
weiminhu@usc.edu, natalie.isenberg@pnnl.gov, jdrgona1@jh.edu, draguna.vrabie@pnnl.gov, dilkina@usc.edu

Abstract

Mixed Binary Quadratic Programs (MBQPs) are an important and complex set of problems in combinatorial optimization. As solving large-scale combinatorial optimization problems is challenging, primal heuristics have been developed to quickly identify high-quality solutions within a short amount of time. Recently, a growing body of research has also used machine learning to accelerate solution methods for challenging combinatorial optimization problems. Despite the increasing popularity of these ML-guided methods, a large body of work has focused on Mixed-Integer Linear Programs (MILPs). MBQPs are challenging to solve due to the combinatorial complexity coupled with nonlinearities. This work proposes ML-guided primal heuristics for Mixed Binary Quadratic Programs (MBQPs) by adapting and extending existing work on ML-guided MILP solution prediction to MBQPs. We introduce a new neural network architecture for MBQP solution prediction and a new training data collection procedure. Moreover, we extend existing loss functions in solution prediction and propose to combine contrastive and weighted cross-entropy losses. We evaluate the methods on standard and real-world MBQP benchmarks and show that the developed ML-guided methods significantly outperform existing primal heuristics and state-of-the-art solvers. Furthermore, models trained with our proposed extension with combined losses outperform other ML-based methods adapted from MILPs and improve generalization in cross-regional inference on a real-world wind farm layout optimization problem.

Extended version — https://arxiv.org/abs/2604.23053

Introduction

Mixed Binary Quadratic Programs (MBQPs) are discrete optimization problems with quadratic terms in the objective function subject to a set of linear constraints. MBQPs encode many important problems in Combinatorial Optimization (CO) (Loiola et al. 2007; Rebennack 2024; Kochenberger et al. 2005) and cover a wide range of applications, including finance (Parpas and Rustem 2006), machine learning (Bertsimas and Shioda 2009), as well as chemical (Misener and Floudas 2013) and energy systems (Turner et al. 2014). A significant body of research on CO algorithms has focused on *primal heuristics*, which are algorithms designed to find good feasible solutions quickly and without optimality guarantees (Berthold 2014).

Despite development in solvers and heuristics, solving large-scale COs remains challenging. In recent years, Machine Learning (ML) has been proposed to accelerate solution methods for CO problems. Motivated by the fact that CO problems that share similar structures are solved repeatedly in many applications (Huang et al. 2024b; Scavuzzo et al. 2024), a growing body of research uses ML to guide algorithmic policies or to build new policies customized to instances that appear in specific applications. For example, Han et al. (2023); Nair et al. (2020); Huang et al. (2024a); Ding et al. (2020) propose ML-guided primal heuristics for Mixed Integer Linear Programs (MILPs), wherein they predict the optimal assignment for a subset of the variables. Lee and Kim (2024) have shown that ML-guided methods reach high-quality solutions faster than non-ML primal heuristics on MILPs. While prior work on ML-guided CO methods has shown success across multiple algorithmic components on many challenging CO problems, existing work in this area has mainly focused on MILPs. A small body of research has used ML to advance solution methods for general nonlinear programming problems (Bonami, Lodi, and Zarpellon 2018; Bagga and Delarue 2023; Ghaddar et al. 2023; Ferber et al. 2023; Tang, Khalil, and Drgoňa 2025), but ML-guided methods in this space are not as well developed as in MILPs.

MBQPs are even more challenging to solve than MILPs due to the combinatorial nature (Magnanti 1981) coupled with nonlinearities. In this work, we develop ML-guided primal heuristics for MBQPs by adapting and extending existing work on ML-guided MILP solution methods. We adapt the Weighted Cross-Entropy-based and Contrastive Learning-based methods which are used in MILP solution prediction to MBQPs. To adapt to MBQPs, we propose a novel neural network architecture that extracts input features and a new data collection procedure that generates high-quality solutions as ground truth training data for large-scale MBQPs. Furthermore, we extend existing loss functions used in CO solution prediction and propose to combine Cross-Entropy and Contrastive losses. Computational results show that the adapted and extended ML methods outperform existing primal heuristics and state-of-the-art solvers on standard and real-world MBQP benchmarks. Fur-

thermore, we show that solution prediction models trained with our proposed extended loss function outperform other ML-guided methods adapted from MILPs in in-domain testing and improve generalization in cross-regional inference on a real-world wind farm layout optimization problem.

Background and Related Works

Mixed Binary Quadratic Programs

A Mixed Binary Quadratic Program (MBQP) with n decision variables is defined as

$$\min x^T Hx + c^T x \quad \text{s.t. } Ax \leq b \text{ and } x_j \in \{0,1\}, \forall j \in B \tag{1}$$

where $H \in \mathbb{R}^{n\times n}$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, and $b \in \mathbb{R}^m$. H is a real symmetric matrix that encodes quadratic terms in the objective function and is not necessarily positive semidefinite, allowing for nonconvex objective functions. $B \subseteq \{1, ..., n\}$ is the set of binary decision variables.

Solution Methods MBQPs are NP-hard in general (Del Pia, Dey, and Molinaro 2017). The Branch-and-Bound (BnB) algorithm is an exact tree search algorithm to solve MILPs, MBQPs and more general Mixed-Integer Nonlinear Programming (MINLP) problems. As large-scale MBQPs are challenging to solve with exact methods, a significant body of research has focused on *primal heuristics*, which are algorithms designed to quickly identify high-quality feasible solutions for a given optimization problem without optimality guarantees (Berthold 2014). These heuristics typically involve solving a relaxation of the original problem and then creating a subproblem by fixing a subset of integer variables by rounding the relaxation values to the nearest integer values, such as RENS (Berthold 2014), Undercover (Berthold and Gleixner 2014), and Relax-Search (Huang et al. 2025).

Solution Prediction for MILPs

Previous work on using ML to accelerate solving CO problems has been focused on Mixed Integer Linear Programming (MILP). An MILP can be viewed as the subclass of MBQPs in Eqn. (1) where the quadratic term matrix H is the zero matrix. The goal of an MILP is to find x such that $c^T x$ is minimized, subject to $Ax \leq b$ and integrality constraints $x_j \in \{0,1\}, \forall j \in B$. A large body of ML-guided primal heuristics for MILPs is based on predicting partial solutions (Nair et al. 2020; Han et al. 2023; Huang et al. 2024a; Ding et al. 2020).

Solution Prediction Nair et al. (2020) and Han et al. (2023) use Weighted Cross-Entropy (WCE) loss to learn the probability distribution of the solution space of an MILP instance M. The goal is to learn from a set of multiple solutions, weighted by the quality of the solution. Specifically, for a solution x, the energy function $E(x; M)$ is defined as $c^T x$ if x is feasible, or ∞ otherwise, assuming minimization. Given M, the conditional distribution of a solution x is modeled as

$$P(x|M) \equiv \frac{\exp(-E(x;M))}{\sum_{x'} \exp(-E(x';M))} \tag{2}$$

, so that solutions with better objective values have a higher probability. The learning task is to train a model $p(x|\theta; M)$ parameterized by θ that approximates $p(x|M)$. To collect training data, Nair et al. (2020) and Han et al. (2023) obtain the set of solutions by running state-of-the-art MILP solvers for a large amount of time. Instead of using WCE loss, Huang et al. (2024a) learn $p(x|\theta; M)$ using Contrastive Learning (CL). The CL-based method makes discriminative predictions by contrasting the positive samples (i.e., good solutions) and negative samples (i.e., bad solutions). Positive samples are obtained by running MILP solvers, similar to (Nair et al. 2020) and (Han et al. 2023). Negative samples are obtained by solving another MILP that searches for bad variable assignments within some Hamming distance of the good solutions.

Inference Since the full prediction might not be feasible, ML-guided primal heuristics for MILPs involve solving another MILP at inference time. Nair et al. (2020) use Neural Diving (ND), which uses the prediction of a subset of the variables and creates a smaller sub-MILP that is easier to solve after fixing the subset. The size of this sub-MILP is controlled by the ratio of variables that are fixed. Han et al. (2023) and Huang et al. (2024a) use a Predict-and-Search (PaS) framework that searches for feasible solutions within some neighborhood of the full prediction by adding a cut to the original MILP. The degrees of freedom in PaS are controlled by the number of variables that are allowed to be different from the prediction. The ND approach allows for faster runtime at inference time as the subproblem contains a small number of variables, but the solutions returned can be more suboptimal. PaS has more freedom to correct errors from the ML predictions, but it can be harder to optimize because the size of the MILP at inference contains the same number of variables as the original MILP.

ML-Guided Methods for Nonlinear Optimization

There has also been research on ML for general nonlinear optimization. Some of the works in this space focus on BnB and are not directly relevant to primal heuristics. For example, Ghaddar et al. (2023) and Maisonneuve and Lesage-Landry (2024) learn branching policies in BnB; Baltean-Lugojan et al. (2019) learns to select cutting planes; Bonami, Lodi, and Zarpellon (2018) learns whether to linearize quadratic terms in the formulation before solving. Other works are relevant to primal heuristics, but do not apply to general MBQPs. Saravanos et al. (2025) develops an ML-aided distributed optimization technique tailored to decomposable problems. Liang and Chen (2024) and Gao et al. (2024) develop ML-guided methods for continuous problems, while Bagga and Delarue (2023) develops an ML-based method tailored to the quadratic assignment problem. Tang, Khalil, and Drgoňa (2025) develops a gradient-based method for inequality-constrained MINLPs using self-supervised learning and projection. To our knowledge, work on ML-guided methods for solving general MBQPs remains limited. The most closely related is Ferber et al. (2023), which develops SurCo, a gradient-based method for general MINLPs that can also be applied

to MBQPs. Additionally, Chen et al. (2024) studies the theoretical expressiveness of graph neural networks for continuous and mixed-integer quadratic programs, which is complementary to our work.

Methods

We develop an ML-guided primal heuristic for MBQPs based on solution prediction, as shown in Fig. 1. An input MBQP is represented as a tripartite graph (Fig. 1 (B)) and then passed to a Graph Attention Network module (Fig. 1 (C)), which produces solution predictions for binary decision variables in MBQPs. At inference time, the predicted solutions are used to create a sub-MBQP (Fig. 1 (F)). We introduce a new method for collecting training data for MBQPs (Fig. 1 (D)). In training the models, we adapt the WCE and CL losses which have been used in solution prediction in MILPs to MBQPs and propose an extended loss function that combines CL and WCE losses (Fig. 1 (E)).

Neural Network Architecture

Tripartite Graph Representation We propose a tripartite graph representation of MBQP instances (Fig. 1 (B)). The tripartite graph contains three types of nodes: constraint nodes (C), variable nodes (V), and quadratic term nodes (Q). A C–V edge connects a variable node and a constraint node when the variable has a nonzero coefficient in that constraint. A Q–V edge connects a quadratic term node to the two variable nodes that participate in the corresponding quadratic term. The feature sets for constraint and variable nodes are adapted from the MILP solution prediction framework of (Han et al. 2023). For quadratic term nodes, we introduce a custom feature set designed to capture the characteristics of quadratic terms in the MBQP objective function; the full list of features is provided in the appendix in the extended version.

Graph Attention Network We learn a policy $p(x|\theta; M)$ parameterized by θ that processes the featured tripartite graph and outputs predictions for the variables of a given instance M, using a Graph Attention Network (GAT) (Brody, Alon, and Yahav 2021). The GAT performs four rounds of message passing, as shown in Fig. 1 (C). In round one, each quadratic term node in Q_1 attends over its neighbors in V_1 using h attention heads to produce updated quadratic term embeddings Q_2. In round two, each variable node in V_1 attends over its neighbors in Q_2 to produce updated variable embeddings V_2. In round three, each constraint node in C_1 attends over its neighbors in V_2 to produce updated constraint embeddings C_2. In the final round, each variable node in V_2 attends over its neighbors in C_2 to produce the final variable embeddings V_3. The message passing outputs are then passed through Multi-Layer Perceptrons (MLPs) to produce logits z. The final prediction is obtained by taking the sigmoid activation function $p(x|\theta; M) = \sigma(z)$ for the WCE-based method and the tanh activation $p(\theta; M) = \tanh(z)$ for the CL-based method.

Loss Function

The solution prediction policy $p(x|\theta; M)$ can be learned with different approaches (Fig. 1 (E)). In this work, we first adapt the WCE and CL losses that have been used for MILPs to MBQPs. Then, we propose an extension that combines CL and WCE losses to improve the performance.

Weighted Cross-Entropy Following the Weighted Cross-Entropy (WCE) (Han et al. 2023) approach, we create a training dataset that contains N MBQP instances $\left\{\left(M_i,\ \mathcal{S}^+_{M_i}\right)\right\}_{i=1}^N$, where $\mathcal{S}^+_{M_i}$ is a set of unique solutions for the instance M_i. Let $p\left(x^+|\theta; M_i\right)$ denote the probability of solution x^+ given instance M_i as the input. We adapt the energy function $E(x; M)$ in Eqn. (2) to the case of MBQPs to assign higher probability for better solutions. For a solution x, the energy function $E(x; M)$ is defined as $x^T Hx + c^T x$ if x is feasible. During training, for an instance M_i with quadratic term matrix H_i and cost vector c_i, the weight applied to the solution x^+ is $w(x^+|M_i) \equiv \frac{\exp\left(-{x^+}^\top H_i x^+ - {c_i}^\top x^+\right)}{\sum_{x\in\mathcal{S}^+_{M_i}} \exp\left(-x^\top H_i x - {c_i}^\top x\right)}$. Based on the Kullback-Leibler divergence, which measures the distance between the conditional distribution in Eqn. (2) and the learned policy, the loss function to be minimized is:

$$\mathcal{L}^{\text{WCE}}(\theta) \equiv -\sum_{i=1}^{N} \sum_{x^+\in\mathcal{S}^+_{M_i}} w(x^+|M_i) \log p\left(x^+|\theta; M_i\right). \tag{3}$$

To avoid numerical overflow, we follow prior work on WCE-based MILP solution prediction (Han et al. 2023) and divide the objective value by a constant λ (which is known as the weight-norm parameter) before computing the exponentials. Details on the weight-norm parameter are deferred to the appendix in the extended version.

Contrastive Learning Following the CL-based approach (Huang et al. 2024a), let $\left\{\left(\mathcal{S}^+_{M_i}, \mathcal{S}^-_{M_i}\right)\right\}_{i=1}^N$ be a training dataset of N MBQP instances, where $\mathcal{S}^+_{M_i}$ and $\mathcal{S}^-_{M_i}$ are the sets of positive and negative samples for instance M_i, respectively. We use a form of the NT-Xent Loss (Chen et al. 2020) to learn to distinguish between positive and negative samples. We use the $\cdot$ operator to denote the dot-product similarity. The loss function to be minimized is

$$\mathcal{L}^{\text{CL}}(\theta) = \sum_{i=1}^{N} \sum_{x^+\in\mathcal{S}^+_{M_i}} \mathcal{L}^+\left(\theta \mid x^+, M_i\right), \tag{4}$$

where

$$\mathcal{L}^+\left(\theta \mid x^+, M_i\right) = -\log \frac{\exp\left(x^+ \cdot p(\theta; M_i)/\tau(x^+|M_i)\right)}{\sum_{\tilde{x}\in\mathcal{S}^-_{M_i}\cup\{x^+\}} \exp\left(\tilde{x} \cdot p(\theta; M_i)/\,\tau(x^+|M_i)\right)}. \tag{5}$$

We use a -1/1 encoding for the solution values in order to measure similarity when computing the dot-product in Eqn. (5). Solution values of 0 are encoded as -1, while solution

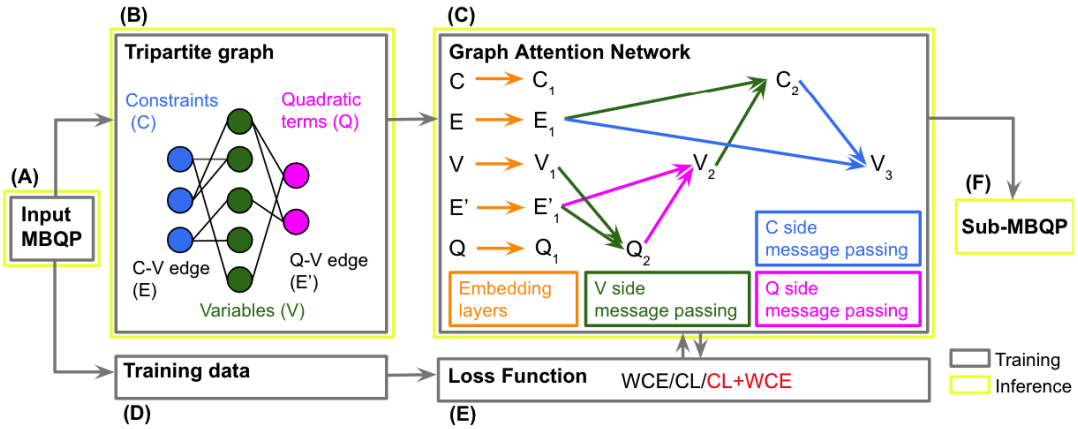


Figure 1: Training/Inference pipeline for ML-guided MBQP solving via solution prediction.

values of 1 remain 1 in x^+ and $\tilde{x}$. To match the -1/1 encoding, we convert the prediction to be in the range of $[-1, 1]$ by taking the tanh activation function for the logits, (i.e., $p(\theta; M_i) = \tanh(z_i)$ where z_i is the logits obtained given instance M_i as the input). Based on the dot-product similarity, the loss value $\mathcal{L}^+(\theta \mid x^+, M_i)$ is low when $p(\theta; M_i)$ is similar to the positive sample x^+ and dissimilar to negative samples $\tilde{x} \in \mathcal{S}^-_{M_i}$. $\tau(x|M_i)$ is a temperature parameter that scales the similarity scores in Eqn. (5). We set $\tau(x|M_i) = \frac{1}{\exp((x^T H_i x + c_i^T x)/s)}$ with $s < 0$ for minimization problems, so that the loss $\mathcal{L}^+(\theta \mid x^+, M_i)$ for better x^+ (i.e., positive sample with a lower objective value) has a lower temperature parameter, which creates a sharper distribution and increases the penalties on the negative samples.

Combining Contrastive Learning and Weighted Cross-Entropy In addition to extending loss functions used in MILP solution prediction, we propose to combine CL and WCE losses. It has been observed that for each positive sample x^+, a subset of variables often have the same assignments across x^+ and the corresponding negative samples in the CL-based approach, and that the CL loss $\mathcal{L}^+(\theta \mid x^+, M_i)$ in Eqn. (5) does not depend on the predicted solution by the ML model for this subset. This observation is formalized in Proposition 1; the proof is deferred to the appendix in the extended version.

Proposition 1 *Given an MBQP instance M_i with sets of positive and negative samples $(\mathcal{S}^+_{M_i}, \mathcal{S}^-_{M_i})$, let $\mathcal{B}_i$ be the index set of all binary decision variables, and let $x^+ \in \mathcal{S}^+_{M_i}$ be any positive sample. Let $\mathcal{U}_i^{x^+} \subseteq \mathcal{B}_i$ be an index set on which the positive sample and all corresponding negative samples agree; that is, $\tilde{x}_d = t_d \; \forall d \in \mathcal{U}_i^{x^+} \; \forall \tilde{x} \in \mathcal{S}^-_{M_i} \cup \{x^+\}$ where $t_d \in \{0, 1\}$ is the common assignment of the variable indexed by d. The CL loss $\mathcal{L}^+(\theta \mid x^+, M_i)$ in Eqn. (5) depends only on the predictions for the subset of variables indexed by $\mathcal{B}_i \setminus \mathcal{U}_i^{x^+}$.*

As shown in Proposition 1, for each positive sample x^+, the CL loss $\mathcal{L}^+(\theta \mid x^+, M_i)$ penalizes predictions that resemble negative samples only for variables outside the subset $\mathcal{U}_i^{x^+}$. Consequently, it provides no gradient signal for variables in $\mathcal{U}_i^{x^+}$. To improve the accuracy of the ML model in predicting the solution for variables in $\mathcal{U}_i^{x^+}$, we propose to apply classification of whether the variable takes 1 or 0 as the solution value using binary Cross-Entropy (CE) loss, given that variables in this set take the same solution value in $\mathcal{S}^-_{M_i} \cup \{x^+\}$ by definition. For each positive sample x^+, we apply the CL loss in Eqn. (5) (which only applies to $\mathcal{B}_i \setminus \mathcal{U}_i^{x^+}$) and CE loss for $\mathcal{U}_i^{x^+}$. For each instance M_i, the CE loss for each $x^+ \in \mathcal{S}^+_{M_i}$ is weighted by $w(x^+|M_i)$, the weight derived from the adapted energy function in Eqn. (2). Formally, for each positive sample $x^+ \in \mathcal{S}^+_{M_i}$, let $t_{d,i}$ denote the assignment of the d-th variable, which is the same across $\mathcal{S}^-_{M_i} \cup \{x^+\}$ for $d \in \mathcal{U}_i^{x^+}$. Let $\hat{p}_{d,i} \equiv p(x_{d,i} = 1|\theta; M_i)$ be the probability that the d-th variable of M_i takes a solution value of 1 predicted by the ML policy. The CE loss for each x^+ is $\mathcal{L}^{\text{CE}}(\theta \mid x^+, M_i) = -\sum_{d \in \mathcal{U}_i^{x^+}} [t_{d,i} \log(\hat{p}_{d,i}) + (1 - t_{d,i}) \log(1 - \hat{p}_{d,i})]$. Accounting for the weights for each positive sample, the combined loss function to be minimized is

$$\mathcal{L}^{\text{CL+WCE}}(\theta) = \sum_{i=1}^{N} \sum_{x^+ \in \mathcal{S}^+_{M_i}} \Big(\lambda^{\text{CL}} \mathcal{L}^+(\theta \mid x^+, M_i) + w(x^+ \mid M_i) \mathcal{L}^{\text{CE}}(\theta \mid x^+, M_i) \Big), \tag{6}$$

where λ^{CL} is a hyperparameter that controls the weight of the CL loss compared to WCE loss.

Training Data Collection for MBQPs

Training data collection (Fig. 1(D)) aims to compile multiple high-quality solutions, as well as low-quality solutions for the CL-based approach, for use in MBQP solution prediction. To this end, we propose *Randomized Relax-Search*, a novel heuristic that generates a diverse set of high-quality solutions for MBQPs. *Randomized Relax-Search* extends *Relax-Search* (Huang et al. 2025), which constructs a subproblem from a suboptimal relaxation solution of the MBQP.

Algorithm 1: Randomized Relax-Search for training data collection

Input: An MBQP $\mathcal{P}$ with set of binary variables $\mathcal{B}$, relaxation time limit T_r, subproblem time limit T_s, number of random seeds K, candidate fixing ratio p_1, final fixing ratio p_2 ($p_1 > p_2$)
Output: Set of good solutions S^+, Set of bad solutions S^-

Relaxed solution $\bar{x} \leftarrow$ Compute the Nonlinear Programming relaxation of $\mathcal{P}$ given time limit T_r
$S^+, S^- \leftarrow \emptyset$
Candidate set $\mathcal{C} \leftarrow$ select $p_1 * |\mathcal{B}|$ variables that are least fractional in $\bar{x}$
for $k \in 1, 2, ..., K$ **do**
    $\mathcal{C'}_k \leftarrow$ Uniformly at random select $p_2 * |\mathcal{B}|$ variables from $\mathcal{C}$
    $\mathcal{P}_k \leftarrow \mathcal{P} \cup \{x_i = \lfloor \bar{x_i} \rceil\}_{i \in \mathcal{C'}_k}$
    $x_k^+, x_k^- \leftarrow$ Best and worst solutions obtained by solving subproblem $\mathcal{P}_k$ with a complete solver, given time limit T_s
    $S^+ \leftarrow S^+ \cup \{x_k^+\}$
    $S^- \leftarrow S^- \cup \{x_k^-\}$
end for
return S^+, S^-

As shown in Algorithm 1, *Randomized Relax-Search* first forms a candidate set C consisting of variables that are least fractional in the relaxation of $\mathcal{P}$, and then introduces randomization to construct K subproblems of $\mathcal{P}$. To construct the k^{th} subproblem, a subset $C'_k \subset C$ is selected uniformly at random, and constraints are added to $\mathcal{P}$ to fix each variable $i \in C'_k$ to its nearest integer value. In solving the k^{th} sub-MBQP, the best solution x_k^+ and the worst solution x_k^- are stored. The procedure returns sets of best solutions S^+ and worst solutions S^- of $\mathcal{P}$.

For training with WCE loss, the set of good solutions S^+ is used. For CL losses, S^+ is used as the set of positive samples. We denote the worst solution value from S^+ as $v' = \max_{x \in S^+} x^T Hx + c^T x$. For the set of negative samples in CL, we use $\{x | (x^T Hx + c^T x) > v', x \in S^-\}$. In other words, we only include solutions in S^- that have worse objective values than the worst solutions in S^+.

Inference

At inference time, we choose to use the ND-based method discussed in the Background and Related Works section, which reduces the original problem to a smaller sub-MBQP (Fig. 1 (F)), as our goal is to develop fast primal heuristics. The PaS-based method is challenging for MBQPs because it requires solving another MBQP of the same size (number of binary variables). After obtaining the variable predictions, we create a sub-MBQP by fixing the top p percent of variables, for which the ML model is most confident (i.e., least fractional in the predictions). The resulting sub-MBQP is then solved with a CO solver.

Computational Experiments

Setup

Benchmarks We evaluate the methods on synthetic and real-world benchmarks. For synthetic benchmarks, we include the Cardinality-constrained Binary Quadratic Programs (CBQP) (Zheng et al. 2014), Cardinality-constrained Quadratic Knapsack Problem (CQKP) (Létocart, Plateau, and Plateau 2014), and the Quadratic Multidimensional Knapsack Problem (QMKP) (Forrester and Hunt-Isaak 2020). All synthetic benchmark instances contain 1000 binary variables and have a quadratic term density of 25%. Moreover, we test on a real-world *Wind Farm Layout Optimization Problem* (WFLOP). WFLOP seeks to identify the placement of a set of wind turbines within a fixed area to maximize power generation across all turbines and over all wind scenarios while also satisfying minimum separation constraints. We use the MBQP formulation of WFLOP in (Huang et al. 2025) and instantiate the instances using probabilistic wind models (i.e., probability density functions sampled from the NOW-23 dataset (Bodini et al. 2023)) which represents long-term wind patterns over a 10 year time horizon at selected locations in the California offshore region. The WFLOP instances contain 1000 binary variables while the quadratic term densities depend on the wind distribution. On average, the WFLOP-California instances have a quadratic term density of 32.42%. Details on the benchmarks are deferred to the appendix in the extended version.

Evaluation Metrics We use the following metrics to evaluate the effectiveness of different methods: (1) The *Primal Gap* (PG) (Berthold 2013) is the normalized difference between the objective value v found by a method and a best known objective value v^*, defined as PG $= \frac{|v-v^*|}{max(|v|,|v^*|)}$, when $vv^* > 0$. When no feasible solution is found or when $vv^* < 0$, PG is defined to be 1. PG is 0 when $|v| = |v^*| = 0$. (2) The *Primal Integral* (PI) (Berthold 2013) is the integral of the primal gap over time, which captures the speed at which better solutions are found. (3) The *# wins* in terms of PI is the number of test instances for which the method results in the lowest PI across all other methods.

Baselines We compare the proposed ML-guided MBQP solving methods adapted from MILPs (WCE and CL) and the proposed method with the extended loss function (CL+WCE) with well-established primal heuristics for MBQPs and general MINLPs, including RENS (Berthold 2014), Undercover (Berthold and Gleixner 2014), and Relax-Search (Huang et al. 2025). In addition, we compare with the state-of-the-art MINLP solver SCIP (Bolusani et al. 2024), which uses BnB as its core component and includes primal heuristics as supplementary procedures to improve the primal bound during BnB. We turn on the aggressive mode in SCIP to focus on finding good primal solutions instead of improving the dual bound. We experiment with both the Linear Programming (LP) reformulation (default) and Nonlinear Programming (NLP) formulation in SCIP and report the stronger one. To our knowledge, this is the first work on ML-guided primal heuristics tailored to general MBQPs.

Benchmark	$\lvert\mathcal{S}^+\rvert_{\text{SCIP}}$	$\lvert\mathcal{S}^+\rvert_{\text{Rand-Relax-Search}}$	obj. $\mathcal{S}^+_{\text{SCIP}}$	obj. $\mathcal{S}^+_{\text{Rand-Relax-Search}}$	frac. $\mathcal{U}$
CBQP	2.01	10	-289048.26	-887895.29	59.71%
QMKP	4.31	10	-46736.29	-177384.21	72.48%
CQKP	2.47	10	-58559.04	-187403.44	56.74%
WFLOP	8.59	10	1999.23	1478.76	57.06%

Table 1: Data collection statistics with proposed Randomized Relax-Search vs SCIP. Average number of distinct solutions found by SCIP ($\lvert\mathcal{S}^+\rvert_{\text{SCIP}}$), average number of distinct solutions in the good samples set $\mathcal{S}^+$ returned by Randomized Relax-Search ($\lvert\mathcal{S}^+\rvert_{\text{Rand-Relax-Search}}$), average objective value found by SCIP (obj. $\mathcal{S}^+_{\text{SCIP}}$), average objective value in $\mathcal{S}^+$ by Randomized Relax-Search (obj. $\mathcal{S}^+_{\text{Rand-Relax-Search}}$), and average percentage of variables that take the same value in positive and negative sample pairs (frac. $\mathcal{U}$). The time limit for both strategies is 11000s.

		CBQP			**QMKP**			**CQKP**			**WFLOP**		
	Method	**PG**	**PI**	**# wins**	**PG**	**PI**	**# wins**	**PG**	**PI**	**# wins**	**PG**	**PI**	**# wins**
Adapted	WCE	0.12	33.21	20	0.09	24.98	26	0.16	36.32	14	0.27	34.59	23
	CL	0.3	40.69	6	0.98†	59.21†	0†	1†	60†	0†	0.83†	53.46†	7†
Extended	CL+WCE	**0.02**	**22.4**	**74**	**0.04**	**21.54**	**74**	**0.06**	**27.07**	**82**	**0.14**	**30.96**	**40**
Baselines	SCIP	1	60	0	0.89	57.83	0	1	60	0	0.59	39.52	11
	RENS	1†	60†	0†	1	59.96	0	0.99†	59.69†	0†	0.36	39.45	18
	Undercover	1†	60†	0†	1	60	0	1	60	0	0.99	59.73	0
	Relax-Search	0.57	50.49	0	0.62	52.31	0	0.56	52.32	4	0.54	48.9	1
	SurCo	1	60	0	1	60	0	1	60	0	0.59	49.06	0

Table 2: Primal Gap (PG), Primal Integral (PI), and # wins in terms of PI. Lower is better for PG and PI. Higher is better for # wins. WCE and CL are ML-guided MBQP primal heuristics that we adapted from MILPs. CL+WCE is the extended ML method with the proposed combined loss function. † indicates benchmarks where there are instances for which the method did not produce a feasible solution. For CL, the feasibility rates are 2%, 0%, and 23% for QMKP, CQKP, and WFLOP. For RENS, the feasibility rates are 0% and 14% for CBQP and CQKP. For Undercover, the feasibility rate is 99% for CBQP. For all other methods and benchmarks, the feasibility rate is 100%. Note that PG, PI, and # wins apply to all instances regardless of feasibility.

As an ML-guided baseline, we include SurCo (Ferber et al. 2023), an ML-based primal heuristic for general MINLPs.

Hyperparameters For the proposed combined loss, we experiment with $\lambda^{CL} \in \{1, 2, 5, 7\}$ and choose the λ^{CL} value that results in the lowest weighted Brier Score (BS) on the validation set. Specifically, BS is defined as

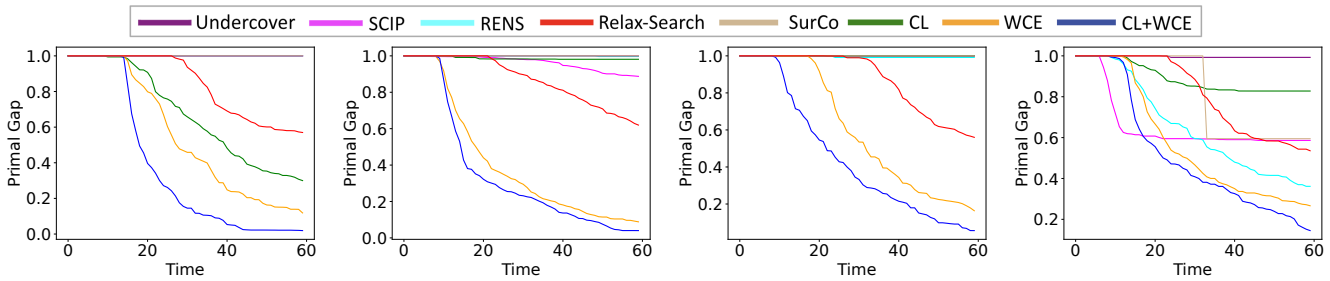
$$BS = \frac{1}{N}\sum_{i=1}^{N}\sum_{x^+\in\mathcal{S}^+_{M_i}}\sum_{d\in\mathcal{B}_i} w(x^+|M_i)(\hat{p}_{d,i} - t^+_{d,i})^2$$

where $t^+_{d,i}$ is the solution assignment of the variable indexed by d in sample x^+ from instance M_i. $\mathcal{B}_i$ is the set of binary variables in M_i. N is the total number of instances in the validation set. For the hyperparameter p, which specifies the fraction of variables fixed when constructing the sub-MBQP at inference time, we experiment with $p \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ on the validation set and choose the best p.

Computational Setup For all methods, we set the time limit to 60s. We report the average PG, PI, and # wins results across 100 test instances for all benchmarks. All models are trained on 800 instances, and training is conducted on an NVIDIA A100 GPU on a node with 128 GB of RAM. We use the AdamW optimizer (Loshchilov and Hutter 2017) with learning rate 10^{-5} with a batch size of 16. Testing (including ML inference and non-ML primal heuristics) is conducted on a cluster with epyc-7542 CPUs with 10 GB RAM. We use SCIP (v9.0) (Bolusani et al. 2024) for solving the sub-MBQPs.

Training Data Collection We compare the training data collection results with *Randomized Relax-Search* and running the SCIP solver under the same time limit. As shown in Table 1, the number of distinct solutions found by SCIP throughout the time limit is less than 10 across all benchmarks. Moreover, *Randomized Relax-Search* produces solutions with better (i.e., lower for minimization problems) objective values in the positive sample set S^+. Additionally, we report the average fraction of the subset of variables that take the same value in positive and negative sample pairs discussed in Proposition 1, frac. $\mathcal{U} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{S}^+_{M_i}|}\sum_{x^+\in\mathcal{S}^+_{M_i}}\frac{|\mathcal{U}_i^{x^+}|}{|\mathcal{B}_i|}$.

Inference As shown in Table 2, both the adapted WCE method and the extended CL+WCE method outperform the baselines in terms of average PG and PI. The adapted CL-based method fails to produce feasible solutions for some instances in QMKP, CQKP, and WFLOP. The extended CL+WCE method performs best in PG, PI, and the number of PI wins across all four benchmarks, and it signifi-



(a) CBQP (b) QMKP (c) CQKP (d) WFLOP

Figure 2: Primal gap as a function of time (lower is better).

		California			Hawaii			Great Lakes		
	Method	**PG**	**PI**	**# wins**	**PG**	**PI**	**# wins**	**PG**	**PI**	**# wins**
Adapted	WCE	0.29	36.78	12	0.31	42.07	9	**0.19**	40.16	10
	CL	0.71†	48.57†	17†	0.71†	49.97†	24†	0.88†	55.92†	9†
Extended	CL+WCE	**0.24**	**31.71**	**34**	**0.29**	38.57	**25**	**0.19**	**36.3**	**46**
Baselines	SCIP	0.58	39.01	8	0.64	41.83	10	0.61	41.1	3
	RENS	0.32	34.95	20	0.45	**38.47**	24	0.38	36.76	31
	Undercover	0.98	59.28	0	1	60	0	1	60	0
	Relax-Search	0.43	38.64	9	0.61	42.04	9	0.56	42.58	1
	SurCo	0.59	45.91	0	0.37	46.78	0	0.3	44.65	0

Table 3: Cross-regional generalization results on WFLOP. Primal Gap (PG), Primal Integral (PI) results, and # wins in PI. Lower is better for PG and PI. Higher is better for # wins. † indicates benchmarks where there are instances for which the method did not produce a feasible solution. For CL, the feasibility rates are 36%, 34%, and 13% for California, Hawaii, and Great Lakes. Note that PG, PI, and # wins apply to all instances regardless of feasibility.

cantly improves both feasibility and solution quality over the adapted CL-based method. Fig. 2 shows the progress of PG over time. The proposed CL+WCE method finds better solutions at all time steps in most benchmarks (the exception being that SCIP achieves lower PG early on in WFLOP), with the adapted WCE method being the second-best.

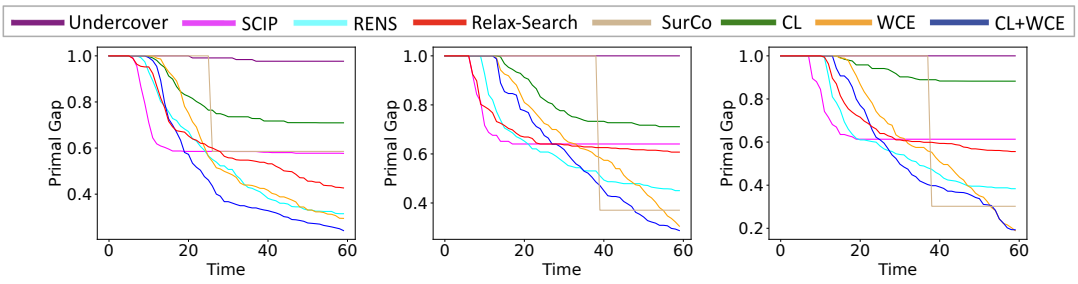
Proposition 1 explains why CL+WCE outperforms CL. In the four benchmarks studied, over 50% of the binary variables take the same assignment across the positive sample and its corresponding negative samples (Table 1). By Proposition 1, the CL loss provides zero gradient signal for these variables, which explains both why pure CL underperforms in PG and PI and why it has low feasibility rates. WCE, in contrast, supervises every variable but is less discriminative on the variables where positive and negative samples differ. CL+WCE combines the two: it applies CL to the variables where negatives differ from positives and WCE to those that share the same assignment across positive and negative samples. Combining the two losses therefore improves solution-prediction performance on this substantial subset of variables, which justifies our proposed extension.

Cross-regional Generalization on WFLOP

The purpose of this study is to understand the impact of regional wind-pattern heterogeneity on the inference performance of the ML-guided primal MBQP heuristics. Wind farm layout optimization in offshore environments is greatly impacted by regional differences in wind resources caused by geographic factors such as proximity to shore, water depth, and local climate. This study attempts to understand the generalization capacity of ML-based primal heuristic methods to unseen regions. The goal is to develop methods that generalize effectively, enabling the efficient design of wind farms using regional wind scenario distributions, even in regions with limited historical wind data for validation. To quantify the cross-regional transferability, we fit the ML-guided primal heuristics using training instances across five U.S. regions and test on multiple unseen regions.

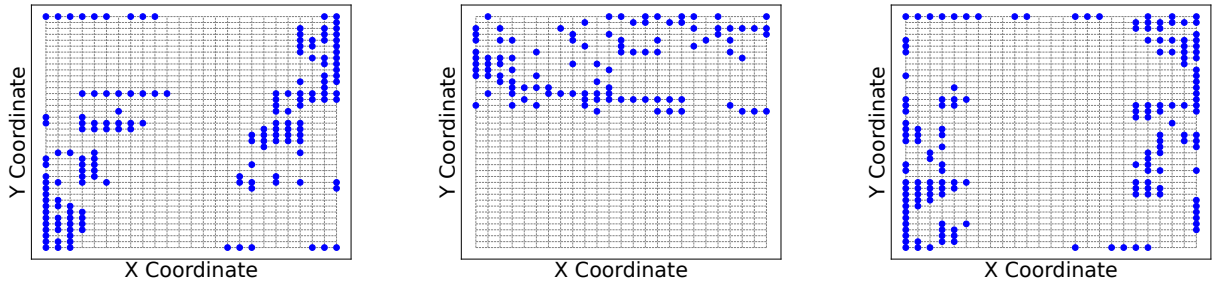
Setup For the ML-guided solution methods, we train the ML models on a mix of 800 instances randomly selected from five regions: Pacific Northwest, Mid Atlantic, Maine, Gulf of Mexico, and South Atlantic. At inference time, we use the trained models to predict solutions for WFLOP instances from three unseen regions: California, Hawaii, and Great Lakes (100 test instances per region).

Results and Discussion As shown in Table 3 and Fig. 3, RENS is a strong baseline for WFLOP. While the WCE-based ML method adapted from MILPs outperforms RENS in the in-domain settings of the previous section, it fails to do so in the cross-regional study. Our proposed extension, which combines the CL and WCE losses, improves cross-regional generalization and outperforms RENS—and all other baselines—in PG, PI, and # wins in the California



(a) California (b) Hawaii (c) Great Lakes

Figure 3: Primal gap as a function of time (lower is better).



(a) WCE (b) RENS (c) CL+WCE

Figure 4: Final wind farm layout solutions for a location in the Hawaii region predicted by different algorithms. Filled circles represent a wind turbine at that grid location.

and Great Lakes regions. In the Hawaii region, CL+WCE does not surpass RENS in PI, but it outperforms RENS in PG and # wins. One explanation for the ability of ML-guided methods to predict high-quality wind farm layouts in unseen regions is that the distributional summary of 10 years of simulated wind data can adequately capture the relevant attributes of offshore wind patterns across geographically distinct regions.

Additionally, we show an example of final wind farm layouts predicted by the best performing methods. Fig. 4 shows the solutions returned by the best performing methods in Table 3 in each category: adapted ML (WCE), extended ML (CL+WCE), and non-ML (RENS). The CL+WCE layout solution leads to a 17% reduction in wind speed losses due to wake effects compared to the WCE layout solution (the second best solution). Wind speed losses impact power production and are a function of turbine placement. And although it is also a function of the distribution of the wind scenarios, greater turbine spacing will often lead to lessened wake effects and less wind speed loss. Thus, the improvement in objective value in the design in Fig. 4(c) may be attributed to the slightly more dispersed turbine clustering.

Ablation with Gurobi

To demonstrate that our methods are solver-agnostic, we conduct the same experiments using Gurobi (Gurobi Optimization, LLC 2026), a state-of-the-art commercial solver. For the adapted and extended ML-guided primal heuristics, Gurobi is used both to collect training data and to solve the sub-MBQPs at inference time. As baselines, we compare against standard Gurobi, as well as Relax-Search and SurCo implemented in Gurobi. We do not include RENS or Undercover, as Gurobi does not provide implementations of these methods. As shown in Table 4, the extended CL+WCE method achieves the best PI and the highest number of PI wins in all four benchmarks. In terms of PG, CL+WCE performs best on three of the four benchmarks; the exception is CBQP, where Gurobi performs slightly better. This may be attributed to the strength of Gurobi as a commercial solver. Consistent with our observations in SCIP, the adapted CL-based method fails to produce feasible solutions for some instances in QMKP, CQKP, and WFLOP. These results confirm that the conclusions drawn from the SCIP experiments also carry over when switched to another solver.

Conclusion and Discussion

We develop ML-guided primal heuristics for MBQPs based on solution prediction. We adapt existing WCE- and CL-based methods, originally developed for ML-guided MILP primal heuristics, to the MBQP setting by introducing a tripartite graph representation, a neural network architecture for feature extraction, and a data collection procedure that produces diverse, high-quality solutions for training. In addition, we extend the existing loss functions used in CO so-

	Method	CBQP PG	CBQP PI	CBQP # wins	QMKP PG	QMKP PI	QMKP # wins	CQKP PG	CQKP PI	CQKP # wins	WFLOP PG	WFLOP PI	WFLOP # wins
Adapted	WCE	0.01	27.68	4	**0.01**	23.34	20	**0**	18.61	34	0.17	44.57	35
	CL	0.02	24.05	27	0.68†	45.68†	16†	0.33†	27.82†	21†	0.42†	50.84†	17†
Extended	CL+WCE	0.01	**23.48**	**56**	**0.01**	**20.79**	**52**	**0**	**18.43**	**45**	**0.14**	**43.84**	**43**
Baselines	Gurobi	**0**	28	13	0.02	23.86	12	0.02	29.78	0	0.55	54.83	3
	Relax-Search	0.65	48.66	0	0.32	39.46	0	0.76	56.56	0	0.81	59.18	0
	SurCo	1	60	0	1	60	0	0.99	59.49	0	0.84	55.08	2

Table 4: Ablation with Gurobi as the solver: Primal Gap (PG), Primal Integral (PI), and # wins in terms of PI. Lower is better for PG and PI. Higher is better for # wins. WCE and CL are ML-guided MBQP primal heuristics that we adapted from MILPs. CL+WCE is the extended ML method with the proposed combined loss function. † indicates benchmarks where there are instances for which the method did not produce a feasible solution. For CL, the feasibility rates are 35%, 78%, and 97% for QMKP, CQKP, and WFLOP. For all other methods and benchmarks, the feasibility rate is 100%. Note that PG, PI, and # wins apply to all instances regardless of feasibility.

lution prediction and combine the CL and WCE losses to address the lack of gradient signal in the CL-based method. Experimental results show that the adapted and extended ML-guided methods produce high-quality solutions quickly and converge faster than non-ML primal heuristics. Furthermore, our combined-loss extension significantly improves performance over other ML methods that use existing loss functions, and it improves generalization in cross-regional inference on WFLOP.

A potential limitation of our methods is their extensive memory requirements. The tripartite graph grows quadratically in size: it is $O(n + m + \rho n^2 + mn)$, where n is the number of variables, m is the number of constraints, and ρ is the quadratic term density (since each quadratic term node connects to two variable nodes). In addition, although the proposed ML method significantly outperforms non-ML primal heuristics at inference time, the offline data collection required to train it is computationally expensive.

Acknowledgments

This work was done during Weimin Huang's internship at the Pacific Northwest National Laboratory (PNNL) and at the University of Southern California. PNNL is a multi-program national laboratory operated by Battelle Memorial Institute for the U.S. Department of Energy (DOE) under Contract No. DE-AC05-76RL0-1830. The research is partially supported by the National Science Foundation (NSF) grant #2112533: "NSF Artificial Intelligence Research Institute for Advances in Optimization (AI4OPT)", the NSF grant #2346058: "NRT-AI: Integrating Artificial Intelligence and Operations Research Technologies", the U.S. Department of Energy, Office of Science Energy Earthshot Initiative, as part of the Addressing Challenges in Energy: Floating Wind in a Changing Climate Energy Earthshot Research Center at PNNL, and the Ralph S. O'Connor Sustainable Energy Institute (ROSEI) at Johns Hopkins University.

References

Bagga, P. S.; and Delarue, A. 2023. Solving the Quadratic Assignment Problem using Deep Reinforcement Learning. *arXiv preprint arXiv:2310.01604*.

Baltean-Lugojan, R.; Bonami, P.; Misener, R.; and Tramontani, A. 2019. Scoring positive semidefinite cutting planes for quadratic optimization via trained neural networks. Optimization Online preprint.

Berthold, T. 2013. Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6): 611–614.

Berthold, T. 2014. RENS: the optimal rounding. *Mathematical Programming Computation*, 6: 33–54.

Berthold, T.; and Gleixner, A. M. 2014. Undercover: a primal MINLP heuristic exploring a largest sub-MIP. *Mathematical Programming*, 144: 315–346.

Bertsimas, D.; and Shioda, R. 2009. Algorithm for cardinality-constrained quadratic optimization. *Computational Optimization and Applications*, 43(1): 1–22.

Bodini, N.; Optis, M.; Redfern, S.; Rosencrans, D.; Rybchuk, A.; Lundquist, J. K.; Pronk, V.; Castagneri, S.; Purkayastha, A.; Draxl, C.; et al. 2023. The 2023 National Offshore Wind data set (NOW-23). *Earth System Science Data Discussions*, 2023: 1–57.

Bolusani, S.; Besançon, M.; Bestuzheva, K.; Chmiela, A.; Dionísio, J.; Donkiewicz, T.; van Doornmalen, J.; Eifler, L.; Ghannam, M.; Gleixner, A.; et al. 2024. The SCIP optimization suite 9.0. *arXiv preprint arXiv:2402.17702*.

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

Brody, S.; Alon, U.; and Yahav, E. 2021. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

Chen, Z.; Chen, X.; Liu, J.; Wang, X.; and Yin, W. 2024. Expressive Power of Graph Neural Networks for (Mixed-Integer) Quadratic Programs. *arXiv preprint arXiv:2406.05938*.

Del Pia, A.; Dey, S. S.; and Molinaro, M. 2017. Mixed-integer quadratic programming is in NP. *Mathematical Programming*, 162: 225–240.

Ding, J.-Y.; Zhang, C.; Shen, L.; Li, S.; Wang, B.; Xu, Y.; and Song, L. 2020. Accelerating primal solution findings for mixed integer programs based on solution prediction. In *AAAI conference on Artificial Intelligence*, volume 34, 1452–1459.

Ferber, A. M.; Huang, T.; Zha, D.; Schubert, M.; Steiner, B.; Dilkina, B.; and Tian, Y. 2023. Surco: Learning linear surrogates for combinatorial nonlinear optimization problems. In *International Conference on Machine Learning*, 10034–10052. PMLR.

Forrester, R. J.; and Hunt-Isaak, N. 2020. Computational Comparison of Exact Solution Methods for 0-1 Quadratic Programs: Recommendations for Practitioners. *Journal of Applied Mathematics*, 2020(1): 5974820.

Gao, X.; Xiong, J.; Wang, A.; Duan, Q.; Xue, J.; and Shi, Q. 2024. IPM-LSTM: A Learning-Based Interior Point Method for Solving Nonlinear Programs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ghaddar, B.; Gómez-Casares, I.; González-Díaz, J.; González-Rodríguez, B.; Pateiro-López, B.; and Rodríguez-Ballesteros, S. 2023. Learning for spatial branching: An algorithm selection approach. *INFORMS Journal on Computing*, 35(5): 1024–1043.

Gurobi Optimization, LLC. 2026. Gurobi Optimizer Reference Manual.

Han, Q.; Yang, L.; Chen, Q.; Zhou, X.; Zhang, D.; Wang, A.; Sun, R.; and Luo, X. 2023. A GNN-Guided Predict-and-Search Framework for Mixed-Integer Linear Programming. In *The Eleventh International Conference on Learning Representations*.

Huang, T.; Ferber, A. M.; Zharmagambetov, A.; Tian, Y.; and Dilkina, B. 2024a. Contrastive Predict-and-Search for Mixed Integer Linear Programs. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 19757–19771. PMLR.

Huang, W.; Huang, T.; Ferber, A. M.; and Dilkina, B. 2024b. Distributional MIPLIB: a Multi-Domain Library for Advancing ML-Guided MILP Methods. *arXiv preprint arXiv:2406.06954*.

Huang, W.; Isenberg, N. M.; Drgoňa, J.; Vrabie, D. L.; and Dilkina, B. 2025. Efficient Primal Heuristics for Mixed Binary Quadratic Programs Using Suboptimal Rounding Guidance. *Proceedings of the International Symposium on Combinatorial Search*, 18(1): 74–82.

Kochenberger, G. A.; Glover, F.; Alidaee, B.; and Rego, C. 2005. An unconstrained quadratic binary programming approach to the vertex coloring problem. *Annals of Operations Research*, 139: 229–241.

Lee, T.-H.; and Kim, M.-S. 2024. RL-SPH: Learning to Achieve Feasible Solutions for Integer Linear Programs. *arXiv preprint arXiv:2411.19517*.

Létocart, L.; Plateau, M.-C.; and Plateau, G. 2014. An efficient hybrid heuristic method for the 0-1 exact k-item quadratic knapsack problem. *Pesquisa Operacional*, 34.

Liang, E.; and Chen, M. 2024. Generative Learning for Solving Non-Convex Problem with Multi-Valued Input-Solution Mapping. In *The Twelfth International Conference on Learning Representations*.

Loiola, E. M.; De Abreu, N. M. M.; Boaventura-Netto, P. O.; Hahn, P.; and Querido, T. 2007. A survey for the quadratic assignment problem. *European journal of operational research*, 176(2): 657–690.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Magnanti, T. L. 1981. Combinatorial optimization and vehicle fleet planning: Perspectives and prospects. *Networks*, 11(2): 179–213.

Maisonneuve, P.; and Lesage-Landry, A. 2024. Learning-accelerated mixed-integer quadratic programming for unit commitment. Les Cahiers du GERAD G–2024–08, GERAD, HEC Montréal, Montréal, Canada.

Misener, R.; and Floudas, C. A. 2013. GloMIQO: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, 57(1): 3–50.

Nair, V.; Bartunov, S.; Gimeno, F.; Von Glehn, I.; Lichocki, P.; Lobov, I.; O'Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; et al. 2020. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*.

Parpas, P.; and Rustem, B. 2006. Global optimization of the scenario generation and portfolio selection problems. In *International Conference on Computational Science and Its Applications*, 908–917. Springer.

Rebennack, S. 2024. Stable set problem: Branch & cut algorithms. In *Encyclopedia of optimization*, 1–14. Springer.

Saravanos, A. D.; Kuperman, H.; Oshin, A.; Abdul, A. T.; Pacelli, V.; and Theodorou, E. 2025. Deep Distributed Optimization for Large-Scale Quadratic Programming. In *The Thirteenth International Conference on Learning Representations*.

Scavuzzo, L.; Aardal, K.; Lodi, A.; and Yorke-Smith, N. 2024. Machine learning augmented branch and bound for mixed integer linear programming. *Mathematical Programming*, 1–44.

Tang, B.; Khalil, E. B.; and Drgoňa, J. 2025. Learning to Optimize for Mixed-Integer Non-linear Programming with Feasibility Guarantees. arXiv:2410.11061.

Turner, S.; Romero, D.; Zhang, P.; Amon, C.; and Chan, T. 2014. A new mathematical programming approach to optimize wind farm layouts. *Renewable Energy*, 63: 674–680.

Zheng, X.; Sun, X.; Li, D.; and Sun, J. 2014. Successive convex approximations to cardinality-constrained convex programs: a piecewise-linear DC approach. *Computational Optimization and Applications*, 59(1): 379–397.