Learning-augmented smooth integer programs with PAC-learnable oracles

Hao-Yuan He[1,2] and Ming Li[1,2]

[1]National Key Laboratory for Novel Software Technology, Nanjing University
[2]School of Artificial Intelligence, Nanjing University
{hehy, lim}@lamda.nju.edu.cn

Preprint

Abstract

This paper investigates learning-augmented algorithms for *smooth integer programs*, covering canonical problems such as MAX-CUT and MAX-k-SAT. We introduce a framework that incorporates a predictive oracle to construct a linear surrogate of the objective, which is then solved via linear programming followed by a rounding procedure. Crucially, our framework ensures that the solution quality is both *consistent* and *smooth* against prediction errors. We demonstrate that this approach effectively extends tractable approximations from the classical *dense* regime to the *near-dense* regime. Furthermore, we go beyond the assumption of oracle existence by establishing its *PAC-learnability*. We prove that the induced algorithm class possesses a bounded pseudo-dimension, thereby ensuring that an oracle with near-optimal expected performance can be learned with polynomial samples.

1 Introduction

Combinatorial optimization stands as a cornerstone of computer science, operations research, and numerous scientific disciplines. A central objective within this domain is the maximization of a function over a discrete feasible set, a formulation that encapsulates canonical challenges such as MAX-CUT and MAX-k-SAT. However, the inherent computational intractability, i.e., NP-hard, of these problems typically precludes the existence of efficient exact algorithms. While *approximation algorithms* offer an alternative by trading optimality for polynomial-time solvability, many fundamental problems are APX-hard [PY91] (e.g., MAX-k-SAT), implying that they do not admit a polynomial-time approximation scheme (PTAS) unless P = NP [Hås01].

Learning-augmented algorithms, or algorithms with predictions, recently have emerged as a promising paradigm [LV21; MV22]. The core observation is that practical instances rarely exhibit the pathological structures found in worst-case scenarios; instead, they typically adhere to underlying distributions or recurring patterns. By utilizing historical data, one can leverage a predictive model—an *oracle*—to guide algorithmic decisions. This data-driven perspective has yielded provable performance gains in domains ranging from caching [LV21] and scheduling [KPS18] to routing [BEX22; Bam+23], enabling algorithms to surpass classical worst-case bounds while retaining robustness against prediction errors.

In this work, we consider the *smooth integer programs*, i.e., maximization of an n-variate degree-d polynomial $p(\boldsymbol{x})$ subject to binary constraints:

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}} & p(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{x} \in \{0,1\}^n. \end{array} \tag{d-IP}$$

This problem was first investigated by Arora, Karger, and Karpinski [AKK99] and is of fundamental importance: it encompasses the entire class of MAX-SNP problems [PY91]. This class constitutes a substantial subset of NP-hard problems, including canonical graph partitioning tasks (e.g., MAX-CUT and MAX-HYPERCUT), as well as general Boolean constraint satisfaction problems (e.g., MAX-k-SAT and MAX-k-CSP), which are known to be APX-hard.

Prior Works Arora, Karger, and Karpinski [AKK99] developed a PTAS for smooth integer programs in the *dense* regime, where the optimal value of (d-IP) is $\Omega(n^d)$. Their approach employs *exhaustive sampling*: for a given precision ϵ, the algorithm selects a random subset of $O(\log n/\epsilon^2)$ variables. It then enumerates all possible assignments for this subset—incurring a time complexity of $2^{O(\log n/\epsilon^2)} = n^{O(1/\epsilon^2)}$—and determines the remaining variables via linear programming followed by randomized rounding. They demonstrated that this approach guarantees an approximation gap of $O(n^d\epsilon)$. Recently, in the context of learning-augmented algorithms, Bampis, Escoffier, and Xefteris [BEX24] introduced a *parsimonious* oracle that predicts the assignments of the $O(\log n/\epsilon^3)$ sampled variables, thereby improving algorithmic efficiency while guaranteeing an approximation gap of $O(n^d(\epsilon+\varepsilon))$, where ε denotes the prediction error of the oracle.

We extend the setting of Bampis, Escoffier, and Xefteris [BEX24] by transitioning from a *parsimonious* oracle to a *full information* oracle. The rationale is intuitive: whereas the parsimonious oracle predicts only a subset of variables to align with the seminal PTAS approach, a full-information oracle predicting all n variables offers richer guidance and more natural in machine learning deployment.

Technically, for smooth integer programs, the objective $p(\boldsymbol{x})$ exhibits β-*smoothness* (see definition 2.1), which is central to our framework and offers two primary advantages:

- *Robust linearization* (see theorem 2.12): We leverage the β-smoothness to construct a linear surrogate that locally approximates the polynomial objective around the oracle prediction $\hat{\boldsymbol{x}}$. Specifically, we decompose the objective into $p(\boldsymbol{x}) \approx c + \sum_{i\in[n]} x_i p_i(\hat{\boldsymbol{x}})$. We establish that the approximation error is strictly bounded by a term dependent on β and the prediction quality. This facilitates a linear programming relaxation of (d-IP) whose optimal value provably tracks that of the original problem.

- *Lossless rounding* (Lemma 3.3 in [AKK99]): The smoothness of the objective further guarantees that an optimal fractional solution of the relaxed linear program can be rounded to an integral solution with negligible degradation in the objective value. Notably, for multilinear objectives, a greedy deterministic rounding strategy can be employed to obtain an integral solution with a non-decreasing objective value (see theorem 2.15).

Based on these properties, our framework comprises three stages: (1) given an instance of problem π, querying the oracle for a solution $f(\pi) = \hat{\boldsymbol{x}}$; (2) solving a linear program relaxation obtained by linearizing the objective around $\hat{\boldsymbol{x}}$; and (3) rounding the fractional solution $\boldsymbol{y}$ to an integral output $\boldsymbol{z}$.

1.1 Our Contributions

Our analysis shows that the algorithm's performance is tightly coupled to the quality of the oracle. Let $\boldsymbol{x}^*$ denote an optimal solution to (d-IP) and define the prediction error as $\varepsilon = \|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_1$. We prove that, with high probability:

$$p(\boldsymbol{z}) \geq p(\boldsymbol{x}^*) - O(n^{d-1/2}\sqrt{\varepsilon}) - \tilde{O}(n^{d-1/2}).$$

The first error term, $O(n^{d-1/2}\sqrt{\varepsilon})$, quantifies the degradation attributable to prediction inaccuracy, while the second term, $\tilde{O}(n^{d-1/2})$, represents the irreducible error arising from the rounding procedure. Significantly, in the case of multilinear objectives, this rounding error can be eliminated via a deterministic greedy rounding strategy (see theorem 2.15).

By utilizing a full-information oracle, our guarantee broadens the applicability of the framework from the *dense* regime of Arora, Karger, and Karpinski [AKK99] and Bampis, Escoffier, and Xefteris [BEX24] to the *near-dense* regime, where the optimum scales as $\Omega(n^{d-1/2+\xi})$ for some $\xi \in (0, \frac{1}{2}]$. In this regime, the approximation ratio of our algorithm is $1 - \tilde{O}(\sqrt{\varepsilon}/n^{\xi})$. Consequently, our algorithm is both *consistent*, converging to optimality as the prediction error vanishes, and *smooth*, degrading gracefully as the error increases; thus, it offers substantial improvements over worst-case bounds even when predictions are imperfect.

Furthermore, by parallelizing with a state-of-the-art approximation algorithm and pick the better one, we readily equip *robustness* against oracle errors.

A critical yet often overlooked aspect of learning-augmented design is the *learnability* of the oracle. Existing works often assume an oracle exists, whereas we establish its learnability. Leveraging the PAC-learning framework for algorithm selection [GR17], we analyze the sample complexity required to train our oracle. We demonstrate that if the hypothesis class of predictors $\mathcal{F}$ has bounded VC-dimension coordinate-wise, the induced algorithm class possesses finite pseudo-dimension. This implies that an empirical risk minimization (ERM) procedure can learn an oracle with *near-optimal expected performance* under polynomial sample complexity.

To summarize, our contributions are as follows:

1. We present a learning-augmented framework for smooth integer programs with *tighter theoretical guarantees* (see theorem 2.16). Our approach linearizes the objective around a predictive oracle and then applies randomized rounding, which enables us to handle a broad class of MAX-SNP problems. Moreover, by leveraging a full-information oracle, we eliminate the sampling phase of Arora, Karger, and Karpinski [AKK99] and Bampis, Escoffier, and Xefteris [BEX24] and obtain an additive error of $\tilde{O}(n^{d-1/2})$, improving upon the $O(n^d)$ error of prior work and extending the valid regime from dense to near-dense instances.

2. We establish the *learnability of the oracle* within the PAC framework (see theorem 3.3). Specifically, we show that the induced algorithm class has finite pseudo-dimension, which implies that the oracle is PAC-learnable with polynomial sample complexity.

Organization We present our learning-augmented framework in section 2. In section 3, we establish the learnability of the oracle within the PAC framework. In section 4, we illustrate applications of our framework to MAX-CUT and MAX-k-SAT as examples. In section 5, we offer concluding remarks. Supplementary materials are provided in the appendix, including an extended discussion of related work, technical details, and an extension of our framework to (d-IP) with degree-d polynomial constraints.

2 Approach

This section presents a learning-augmented approximation framework for Boolean integer programming with smooth polynomial objectives [AKK99]. We begin by introducing the preliminaries, followed by a description of the framework's mechanism and theoretical guarantees, initially for the quadratic case and subsequently generalized to higher orders.

2.1 Preliminaries

Let $[n] = \{1, \ldots, n\}$. We denote the closed interval $[a - b, a + b]$ by $a \pm b$. Throughout this paper, unless otherwise specified, $\boldsymbol{x} \in \{0, 1\}^n$ represents an n-dimensional Boolean vector, and e denotes the base of the natural logarithm.

Definition 2.1 (β-smooth)**.** An n-variate polynomial $p(\boldsymbol{x})$ of degree d is *β-smooth* if there exists a constant $\beta > 0$ such that for every $0 \le l \le d$, the absolute value of the coefficient for any degree-l monomial in the expansion of $p(\boldsymbol{x})$ is bounded by βn^{d-l}.

A wide array of fundamental optimization problems can be formulated as optimizing smooth polynomial objectives. We provide two illustrative examples below.

Example 1. Given an undirected graph $G = (V, E)$, MAX-CUT partitions V to maximize the number of crossing edges. We define the objective function over $\boldsymbol{x} \in \{0,1\}^n$, with $x_i = 1$ denoting that vertex i is in the subset, as $p(\boldsymbol{x}) = \sum_{i\in[n]} x_i \left(\sum_{(i,j)\in E}(1 - x_j)\right)$, which is 2-smooth.

Example 2. Given a formula in conjunctive normal form where each clause has exactly k literals, MAX-k-SAT maximizes the number of satisfied clauses. The objective function is $p(\boldsymbol{x}) = \sum_{C\in\mathcal{C}} \left(1 - \prod_{i\in C^-} x_i \prod_{i\in C^+}(1 - x_i)\right)$, where C^+ and C^- denote indices of positive and negative literals in clause C. This degree-k polynomial is 4^k-smooth.

Indeed, the entire MAX-SNP class can be formulated in the form of (d-IP) and exhibits β-smoothness by employing a formulation analogous to that of MAX-k-SAT; we defer these details to section F in appendix.

Definition 2.2 (Problem Regimes)**.** Consider a discrete optimization problem with a degree-d polynomial objective. The instance is termed *dense* if its optimal objective value is $\Omega(n^d)$ [AKK99], *near-dense* if the optimum scales as $\Omega(n^{d-1/2+\xi})$ for some $\xi \in (0, 1/2]$.

This classification aligns with problem-specific conventions. For instance, a dense graph possesses $\Omega(n^2)$ edges, while a dense 3-SAT formula contains $\Omega(n^3)$ clauses.

An n-variate degree-d β-smooth polynomial $p(\boldsymbol{x})$ can be decomposed while preserving the β-smoothness property.

Lemma 2.3. *Let $p(\boldsymbol{x})$ be an n-variate β-smooth polynomial of degree d. Then, there exist a constant c and degree-$(d-1)$ β-smooth polynomials $\{p_j(\boldsymbol{x})\}_{j\in[n]}$ such that*

$$p(\boldsymbol{x}) = c + \sum_{j=1}^{n} x_j\, p_j(\boldsymbol{x}).$$

Note that this decomposition may not be unique, however it does not affect our analysis. Repeated application of this decomposition yields the following generalized structural property for smooth polynomials.

Lemma 2.4. *Let $p(\boldsymbol{x})$ be an n-variate β-smooth polynomial of degree d. For any integer $l \in [d]$ and any index tuple $I = (i_1, \ldots, i_{d-l}) \in [n]^{d-l}$, the polynomial $p_I(\boldsymbol{x})$ admits the following decomposition:*

$$p_I(\boldsymbol{x}) = c_I + \sum_{j\in[n]} x_j \cdot p_{I,j}(\boldsymbol{x}).$$

Utilizing this decomposition, we can establish quantitative bounds on the magnitude of the polynomial. Based on the definition of β-smoothness, we derive the following recursive bound for the components of the hierarchical decomposition.

Corollary 2.5. *Let $p(\boldsymbol{x})$ be an n-variate β-smooth polynomial of degree d. For any fixed index tuple $I = (i_1, \ldots, i_{d-l})$ and any $\boldsymbol{x} \in \{0,1\}^n$, the component $p_I(\boldsymbol{x})$ satisfies*

$$|p_I(\boldsymbol{x})| \leq \beta(l+1)n^l.$$

Finally, we recall a global bound on the value of smooth polynomials established in prior work.

Lemma 2.6 (Lemma 3.2, [AKK99])**.** *Let $p(\boldsymbol{x})$ be an n-variate β-smooth polynomial of degree d. If $n > d$, then for any $\boldsymbol{x} \in [0,1]^n$, it holds that $|p(\boldsymbol{x})| \leq 2\,\beta e\, n^d$.*

2.2 The Quadratic Case

Given an n-variate degree-2 β-smooth polynomial $p(\boldsymbol{x})$, consider the following problem:

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & p(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \{0,1\}^n. \end{aligned} \tag{2-IP}$$

This problem is known to be NP-hard, as it captures problems such as MAX-CUT (see example 1). Since $p(\boldsymbol{x})$ is a quadratic objective, it can be expanded using lemma 2.4:

$$p(\boldsymbol{x}) = \sum_{i \in [n]} x_i \cdot p_i(\boldsymbol{x}) + c,$$

where $p_i(\boldsymbol{x})$ is a linear function and c is a constant. To derive a tractable approximation, we linearize the objective function. Specifically, we relax (2-IP) to the following formulation:

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & c + \sum_{i \in [n]} x_i \cdot \rho_i \\ \text{s.t.} \quad & \rho_i - \delta \leq p_i(\boldsymbol{x}) \leq \rho_i + \delta, \\ & \boldsymbol{x} \in [0,1]^n. \end{aligned} \tag{2-LP}$$

where ρ_i is an auxiliary variable approximating $p_i(\boldsymbol{x})$ within a tolerance parameter δ. Since $p_i(\boldsymbol{x})$ is linear in $\boldsymbol{x}$, this relaxation is a linear program and is solvable in polynomial time. Moreover, when $\delta = 0$ and $\rho_i = p_i(\boldsymbol{x}^*)$ for all i, the optimal solution of (2-LP) coincides with that of (2-IP). The remaining challenge lies in determining the appropriate approximation ρ_i and tolerance δ.

2.2.1 Oracle-guided Relaxation

Given a prediction $\hat{\boldsymbol{x}}$ provided by an oracle, a natural strategy is to set $\rho_i = p_i(\hat{\boldsymbol{x}})$ for all $i \in [n]$. With ρ_i fixed, it suffices to choose $\delta \geq \max_{i\in[n]} |p_i(\hat{\boldsymbol{x}}) - p_i(\boldsymbol{x}^*)|$ to guarantee the feasibility of the optimal solution $\boldsymbol{x}^*$ for the relaxed problem. By exploiting the β-smoothness of $p(\boldsymbol{x})$, we can bound this deviation as follows.

Lemma 2.7. *Let $p(\boldsymbol{x})$ be an n-variate β-smooth polynomial of degree 2 with the decomposition $p(\boldsymbol{x}) = \sum_{i\in[n]} x_i p_i(\boldsymbol{x}) + c$. For any two binary vectors $\boldsymbol{x}^*, \hat{\boldsymbol{x}} \in \{0,1\}^n$, let $\varepsilon = \|\boldsymbol{x}^* - \hat{\boldsymbol{x}}\|_1$. Then, for any $i \in [n]$, the deviation of the linear component is bounded by:*

$$|p_i(\hat{\boldsymbol{x}}) - p_i(\boldsymbol{x}^*)| \leq \beta\sqrt{n}\sqrt{\varepsilon}.$$

Accordingly, we set the tolerance to $\delta := \beta\sqrt{n}\sqrt{\varepsilon}$. Note that the value of ε is initially unknown. However, since $\boldsymbol{x} \in \{0,1\}^n$, ε is an integer bounded by n. Consequently, we can enumerate all possible values of ε and solve (2-LP) to identify the best solution, while preserving polynomial time complexity. With the coefficients ρ_i and tolerance δ determined, we proceed to quantify the relaxation gap.

2.2.2 Analysis of the Relaxation Gap

Let $\boldsymbol{y}$ denote an optimal solution to the relaxed problem formulated in (2-LP). We now demonstrate that (2-LP) serves as an effective approximation for (2-IP) by establishing that $p(\boldsymbol{y})$ closely approximates $p(\boldsymbol{x}^*)$. By leveraging the feasibility and the optimality of $\boldsymbol{y}$ within (2-LP), we derive the following lower bound:

$$\begin{aligned} p(\boldsymbol{y}) &\geq c + \sum_{i\in[n]} y_i(\rho_i - \delta) \\ &\geq c + \sum_{i\in[n]} x_i^*\rho_i - \sum_{i\in[n]} y_i\delta \\ &\geq c + \sum_{i\in[n]} x_i^*(p_i(\boldsymbol{x}^*) - \delta) - n\delta \\ &\geq p(\boldsymbol{x}^*) - 2n\delta. \end{aligned} \tag{1}$$

The first inequality is from the feasibility of $\boldsymbol{y}$. The second inequality is from the optimality of $\boldsymbol{y}$. The third inequality is from the feasibility of $\boldsymbol{x}^*$, alongside the bound $\sum_{i\in[n]} y_i \leq n$. Consequently, substituting $\delta = \beta\sqrt{n}\sqrt{\varepsilon}$ yields:

$$p(\boldsymbol{y}) \geq p(\boldsymbol{x}^*) - 2\beta n^{3/2}\sqrt{\varepsilon}. \tag{2}$$

Remark 2.8. The tolerance parameter δ governs the trade-off between *feasibility* and *relaxation tightness*. An overly small value may render the optimal integer solution $\boldsymbol{x}^*$ infeasible, whereas an excessively large value widens the relaxation gap, which scales linearly with δ. Our choice $\delta = \beta\sqrt{n}\sqrt{\varepsilon}$ guarantees feasibility while bounding the gap by $O(\beta n^{3/2}\sqrt{\varepsilon})$.

2.2.3 Randomized Rounding

The optimal solution $\boldsymbol{y}$ to the relaxation (2-LP) is typically fractional. To recover an integral solution, we round the fractional vector $\boldsymbol{y}$ to a binary one. We employ *randomized rounding*, a standard technique widely used in approximation algorithms [RT87; GW95]. Formally, each variable z_j is independently set to 1 with probability y_j and to 0 otherwise. We further demonstrate that the rounded solution $\boldsymbol{z}$ preserves the objective value with high probability, i.e., $p(\boldsymbol{z}) \approx p(\boldsymbol{y})$.

Theorem 2.9. *Let $p(\boldsymbol{x})$ be an n-variate quadratic β-smooth polynomial. Let $\boldsymbol{y} \in [0,1]^n$ be a fractional vector, and let $\boldsymbol{z} \in \{0,1\}^n$ be the integral vector derived via independent randomized rounding with $\Pr[z_i = 1] = y_i$. For any $k \geq 1$, with probability at least $1 - 4/n^k$, the following bound holds:*

$$|p(\boldsymbol{z}) - p(\boldsymbol{y})| \leq 3n\beta\sqrt{\frac{k+1}{2}}\sqrt{n \ln n}. \tag{3}$$

Remark 2.10. The rounding error in the quadratic case scales as $O(n^{3/2}\sqrt{\ln n})$, i.e., $\tilde{O}(n^{3/2})$, which is almost of the same magnitude as the relaxation gap in (2) for a fixed oracle. Notably, this error is independent of the oracle's quality, implying that (3) captures the intrinsic cost of the randomized rounding strategy, which cannot be reduced by improving the oracle's precision. Furthermore, the randomized rounding procedure can be *derandomized* using the method of conditional expectations [Rag88], leading to a deterministic rounding procedure with the same error bound, at the cost of a polynomial increase in time complexity.

2.2.4 Comprehensive Guarantee

We now establish the performance guarantee by combining the relaxation gap (2) with the rounding error (3).

Theorem 2.11. *Let $\boldsymbol{x}^*$ be the optimal solution to (2-IP), and let $\boldsymbol{z}$ be the integral solution obtained via randomized rounding from the fractional solution to (2-LP). With probability at least $1 - 4n^{-k}$, we have*

$$p(\boldsymbol{z}) \geq p(\boldsymbol{x}^*) - 2\beta n^{3/2}\sqrt{\varepsilon} - 3\beta n^{3/2}\sqrt{\frac{k+1}{2}\ln n}\,. \tag{4}$$

This result holds for several classes of quadratic objectives, such as MAX-CUT, MAX-DICUT, and MAX-2-SAT. For *near-dense* problem where $p(\boldsymbol{x}^*) = \Omega(n^{3/2+\xi})$ with some $\xi \in (0, \frac{1}{2}]$, this yields a multiplicative approximation ratio of $1 - \tilde{O}(\sqrt{\varepsilon}/n^{\xi})$.

2.3 Generalization to Higher Orders

We now extend our analysis to the general case, specifically, for an n-variate β-smooth polynomial $p(\boldsymbol{x})$ of degree d:

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}} & p(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{x} \in \{0,1\}^n. \end{array} \tag{d-IP}$$

Analogous to the quadratic setting (section 2.2), we relax this integer program to a linear programming formulation by leveraging an oracle prediction $\hat{\boldsymbol{x}}$. The primary challenge stems from the *decomposition*: while the identity $p(\boldsymbol{x}) = c + \sum_{i\in[n]} x_i \cdot p_i(\boldsymbol{x})$ remains valid, the coefficient functions $p_i(\boldsymbol{x})$ are generally non-linear; specifically, each $p_i(\boldsymbol{x})$ is a degree-$(d-1)$ β-smooth polynomial. Consequently, a single decomposition step reformulates the primary optimization problem (d-IP) as:

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & c + \sum_{j\in[n]} x_j \cdot \rho_j \\ \text{s.t.} \quad & p_j(\boldsymbol{x}) \in \rho_j \pm \delta_j \quad \forall j \in [n] \\ & \boldsymbol{x} \in \{0,1\}^n \end{aligned}$$

Although the objective function becomes linear in x_j and the auxiliary variables ρ_j, the constraints $p_j(\boldsymbol{x}) \in \rho_j \pm \delta_j$ remain non-linear. To systematically address the non-linearity, we apply the decomposition recursively. For any valid index tuple I encountered during the process, we define the linear approximation $q_I(\boldsymbol{x})$ of the polynomial term $p_I(\boldsymbol{x})$ as: $q_I(\boldsymbol{x}) = c_I + \sum_{j\in[n]} x_j \cdot p_{I,j}(\hat{\boldsymbol{x}})$. The recursive procedure replaces each non-linear constraint on $p_I(\boldsymbol{x})$ with linear constraints derived from $q_I(\boldsymbol{x})$, continuing until the system is fully linearized. Let $\mathcal{I}$ denote the set of all valid index tuples generated by this process. Using these approximations, we formulate the relaxed problem as:

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & c + \sum_{j\in[n]} x_j \cdot p_j(\hat{\boldsymbol{x}}) \\ \text{s.t.} \quad & q_I(\boldsymbol{x}) \in p_I(\hat{\boldsymbol{x}}) \pm \delta_I \quad \forall I \in \mathcal{I} \\ & \boldsymbol{x} \in [0,1]^n. \end{aligned} \tag{d-LP}$$

Since both the objective function and constraints in (d-LP) are linear, the problem can be solved efficiently using standard linear programming solvers. The remaining challenge is to determine the appropriate tolerance δ_I for the constraints in (d-LP).

2.3.1 Oracle-guided Relaxation

Analogous to the quadratic case, we define the tolerance parameter δ_I based on the deviation of the optimal solution $\boldsymbol{x}^*$ from the oracle prediction $\hat{\boldsymbol{x}}$:

$$\delta_I := |q_I(\boldsymbol{x}^*) - q_I(\hat{\boldsymbol{x}})| = |q_I(\boldsymbol{x}^*) - p_I(\hat{\boldsymbol{x}})|\,.$$

In the case where $|I| = d-1$ (i.e., the linear terms), because $|c_{I,j}| \le \beta$, corollary 2.5 implies:

$$\delta_I = \left| \sum_{j\in[n]} (x^*_{I,j} - \hat{x}_{I,j}) \cdot c_{I,j} \right| \le \beta\sqrt{n}\sqrt{\varepsilon}.$$

Next, for the general case where $|I| < d-1$, recall that $p_{I,j}(\hat{\boldsymbol{x}})$ is a degree-$(d-|I|-1)$ β-smooth polynomial. Expanding the expression and applying lemma 2.6, we obtain the

following bound:

$$\delta_I = \left| \sum_{j \in [n]} (x^*_{I,j} - \hat{x}_{I,j}) \cdot p_{I,j}(\hat{\boldsymbol{x}}) \right| \leq 2\beta e n^{d-|I|-1/2} \sqrt{\varepsilon}.$$

These settings ensure that the true optimal solution satisfies the relaxed constraints.

2.3.2 Analysis of the Relaxation Gap

We now bound the relaxation gap, defined as the discrepancy between the optimal value of the relaxed problem and the original integer program.

Theorem 2.12. *Let $\boldsymbol{y}$ be the optimal solution to the relaxed problem (d-LP) and $\boldsymbol{x}^*$ be the optimal solution to the original integer program (d-IP). The relaxation gap is bounded by:*

$$p(\boldsymbol{y}) \geq p(\boldsymbol{x}^*) - 2\left[2e(d-2)+1\right] \beta n^{d-1/2} \sqrt{\varepsilon}.$$

Proofsketch. Analogous to the quadratic case gap (2), the gap between $p(\boldsymbol{y})$ and $p(\boldsymbol{x}^*)$ follows from the feasibility of $\boldsymbol{x}^*$ and the optimality of $\boldsymbol{y}$ within (d-LP). In this general setting, the total error bound consists of the sum of the tolerances δ_I derived in section 2.3.1 over all valid indices I. Summing these δ_I terms yields the final result. □

2.3.3 Rounding

To recover an integral solution $\boldsymbol{z}$ from the fractional solution $\boldsymbol{y}$ obtained via (d-LP), we generally employ independent randomized rounding, analogous to the quadratic case. In this general setting, the rounding error is bounded by the following concentration inequality.

Theorem 2.13. *Let $\boldsymbol{y} \in [0,1]^n$, and let $\boldsymbol{z} \in \{0,1\}^n$ be generated via independent randomized rounding where $\Pr[z_i = 1] = y_i$ for all $i \in [n]$. Consider an n-variate degree-d polynomial $p(\boldsymbol{x})$ that is β-smooth. For any $k > d$, with probability at least $1 - 2d/n^{\,k+1-(d-1)}$,*

$$|p(\boldsymbol{y}) - p(\boldsymbol{z})| \leq \left(1 + 2e\,(d-2)\right) \beta\, n^{d-1} \sqrt{\tfrac{k+1}{2}}\, \sqrt{n \ln n}.$$

However, theorem 2.13 fails to leverage the fine-grained structure of the objective function. In particular, when the objective is *multilinear*, it exhibits structural properties that facilitate a more effective rounding scheme.

Definition 2.14. A polynomial p is said to be *multilinear* if it is affine with respect to any individual variable x_k when all other variables are held fixed:

$$p(x_1, \ldots, x_k, \ldots, x_n) = a \cdot x_k + b,$$

where a and b are independent of x_k.

It is straightforward to verify that the objective functions of both MAX-CUT and MAX-k-SAT are multilinear. Thus we can employ a greedy deterministic rounding strategy, which ensures that the objective value is non-decreasing.

Theorem 2.15. *Let $p(\boldsymbol{x})$ be a multilinear polynomial. For any fractional solution $\boldsymbol{y} \in [0,1]^n$, the greedy deterministic rounding procedure yields an integral vector $\boldsymbol{z} \in \{0,1\}^n$ such that:*

$$p(\boldsymbol{z}) \geq p(\boldsymbol{y}).$$

Proofsketch. With the multilinearity of $p(\boldsymbol{x})$, we observe that the objective function is affine with respect to each variable. Consequently, making a greedy choice at each step guarantees that the objective value is non-decreasing. □

2.3.4 Overall Guarantee

We now synthesize the relaxation gap analysis with the rounding error bounds to derive a comprehensive approximation guarantee. This result extends the quadratic bound presented in (4) to polynomials of arbitrary degree d.

Theorem 2.16. *Let $p(\boldsymbol{x})$ be an n-variate, degree-d, β-smooth polynomial. Let $\boldsymbol{y}$ denote the optimal solution to the relaxed problem (d-LP), and let $\varepsilon = \|\boldsymbol{x}^* - \hat{\boldsymbol{x}}\|_1$ quantify the L_1 error of the oracle prediction. Define the constant $\eta = 2e(d-2)+1$. With probability at least $1 - 2d/n^{k-d+2}$, the solution $\boldsymbol{z}$ obtained via randomized rounding satisfies:*

$$p(\boldsymbol{z}) \geq p(\boldsymbol{x}^*) - 2\eta\beta n^{d-1/2}\sqrt{\varepsilon} - \eta\beta n^{d-1}\sqrt{\frac{k+1}{2}}\sqrt{n\ln n}. \tag{5}$$

If $p(\boldsymbol{x})$ is multilinear and $\boldsymbol{z}$ is obtained via the deterministic rounding strategy, then:

$$p(\boldsymbol{z}) \geq p(\boldsymbol{x}^*) - 2\eta\beta n^{d-1/2}\sqrt{\varepsilon}. \tag{6}$$

Proof. The proof decomposes the total approximation error into the relaxation gap and the rounding loss. First, theorem 2.12 bounds the relaxation gap as $p(\boldsymbol{y}) \geq p(\boldsymbol{x}^*) - 2\eta\beta n^{d-1/2}\sqrt{\varepsilon}$. The inequality (5) follows by combining this result with the high-probability rounding error bound established in theorem 2.13. For (6), when $p(\boldsymbol{x})$ is multilinear, theorem 2.15 guarantees monotonic improvement, i.e., $p(\boldsymbol{z}) \geq p(\boldsymbol{y})$. This property eliminates the rounding error term, yielding the tighter bound in (6). □

2.4 Algorithmic Framework

A complete optimization protocol is summarized in algorithm 1. The algorithm first invokes the learning oracle to obtain a prediction, constructs and solves the linear relaxation, and finally maps the fractional solution to a valid integer assignment using the specified rounding strategy.

Complexity Analysis. Although the exact prediction error $\varepsilon = \|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_1$ is typically unknown, its discrete nature (ranging over $\{0,\ldots,n\}$) facilitates an exhaustive search. By enumerating all feasible values of ε, we identify the solution maximizing the objective function. The overall computational complexity is dominated by the linear programming steps, totaling $O(n \cdot T_{\texttt{LP}})$, where $T_{\texttt{LP}}$ denotes the time complexity of the LP solver (one of the best result

Algorithm 1: Learning-augmented optimization framework

Input : Problem instance $\pi \in \Pi$, rounding `strategy`
Output: Integer solution $\boldsymbol{z}^*$
Obtain oracle prediction $\hat{\boldsymbol{x}} \leftarrow f(\pi)$
for $\varepsilon \in \{0, 1, \ldots, n\}$ **do**
  (d-LP) $\leftarrow$ RELAX$(p, \hat{\boldsymbol{x}}, \varepsilon)$ `// see algorithm 3`
  Solve (d-LP) to obtain fractional solution $\boldsymbol{y}$
  if *`strategy` is deterministic* **then**
    Obtain $\boldsymbol{z}$ using the deterministic greedy strategy
  else
    Obtain $\boldsymbol{z}$ using randomized rounding
  Yield $\boldsymbol{z}$ and $p(\boldsymbol{z})$
return best performing $\boldsymbol{z}^*$

is approximately $O(n^{2.38})$, [CLS21]). The rounding procedure, operating in linear time, contributes negligibly.

Notably, the proposed framework exhibits three key properties of learning-augmented algorithms [MV22]: (i) *Consistency*: With perfect prediction (i.e., $\hat{\boldsymbol{x}} = \boldsymbol{x}^*$), the algorithm recovers the optimal solution (up to rounding errors); (ii) *Smoothness*: The approximation ratio degrades gracefully with prediction error ε, maintaining reliability; (iii) *Robustness*: By running in parallel with the best worst-case algorithm, performance never falls below the standard baseline.

3 Learnability of Oracles

Theorem 2.16 establishes that a small prediction error ε suffices to guarantee good performance. This naturally motivates a fundamental inquiry:

How can such an oracle be acquired?

In this section, following [GR17], we address this question by establishing a statistical learnability guarantee: under the standard assumption that the complexity of the hypothesis space is controlled by a bounded VC-dimension (e.g., for neural networks of bounded size [Bar+19]), an oracle is *PAC-learnable*.

3.1 Setup

Let Π denote the instance space, where each instance $\pi \in \Pi$ represents a discrete optimization problem as defined in (d-IP). Without loss of generality, we assume all instances have the same fixed size. Let $\mathcal{D}$ be an unknown but fixed distribution over Π. Let $\mathcal{F}$ denote the hypothesis space of functions $f : \Pi \to \{0, 1\}^n$, serving as the set of candidate oracles. Each $f \in \mathcal{F}$ induces an algorithm $\mathcal{A}_f$ that utilizes $f(\pi)$ to generate a candidate solution $\hat{\boldsymbol{x}}(\pi)$,

which is then fed into algorithm 1 to produce the final solution $\boldsymbol{z}_f(\pi)$. Let $\boldsymbol{x}^*(\pi)$ denote the optimal solution to instance π. Let $\mathcal{A} = \{\mathcal{A}_f \mid f \in \mathcal{F}\}$ denote the induced class of algorithms.

We evaluate the performance of an algorithm via a cost function $\texttt{COST} : \mathcal{A} \times \Pi \to [0, H]$, defined by:

$$\texttt{COST}(\mathcal{A}_f, \pi) \triangleq H - p(\boldsymbol{z}_f(\pi)),$$

where $p(\cdot)$ is the objective function and H is a uniform upper bound on the optimal objective value (e.g., the total number of clauses in MAX-SAT). Define the expected objective value attained by an oracle as $\mathcal{P}(f) \triangleq \mathbb{E}_{\pi\sim\mathcal{D}}[p(\boldsymbol{z}_f(\pi))]$, and define the optimal expected objective as $\mathcal{P}^* \triangleq \mathbb{E}_{\pi\sim\mathcal{D}}[p(\boldsymbol{x}^*(\pi))]$.

We adopt the standard PAC framework. The expected risk of an algorithm $\mathcal{A}_f$ is given by $R(f) \triangleq \mathbb{E}_{\pi\sim\mathcal{D}}[\texttt{COST}(\mathcal{A}_f, \pi)]$. Accordingly, let $f^*_{\texttt{cost}} \in \arg\min_{f\in\mathcal{F}} R(f)$ denote the optimal oracle within the hypothesis class. We define the excess risk (error) of any candidate f as

$$\text{error}(f) \triangleq R(f) - R(f^*_{\texttt{cost}}).$$

Given a training set $S = \{\pi_1, \ldots, \pi_m\}$ drawn i.i.d. from $\mathcal{D}$, our goal is to identify a hypothesis $\hat{f} \in \mathcal{F}$ such that $\text{error}(\hat{f})$ is small. We employ the empirical risk minimization (ERM) principle, which selects the hypothesis minimizing the empirical risk on S:

$$\hat{f}_{\text{ERM}} \in \arg\min_{f\in\mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \texttt{COST}(\mathcal{A}_f, \pi_i).$$

3.2 ERM Guarantees

We show that ERM learns, with high probability, a near-optimal oracle within $\mathcal{F}$, and then connect this statistical guarantee to theorem 2.16 to obtain an end-to-end bound on the expected optimization performance.

Recall from theorem 2.16 that the approximation gap of $\mathcal{A}_f$ on an instance π is controlled by the prediction error of the oracle f. Accordingly, we consider the expected prediction error

$$\mathcal{E}(f) \triangleq \mathbb{E}_{\pi\sim\mathcal{D}}\big[\|f(\pi) - \boldsymbol{x}^*(\pi)\|_1\big].$$

Let $f^*_{\text{pred}} \in \arg\min_{f\in\mathcal{F}} \mathcal{E}(f)$ be a minimizer, and define $\varepsilon_{\mathcal{F}} \triangleq \mathcal{E}(f^*_{\text{pred}})$, which quantifies the statistical realizability of the hypothesis space $\mathcal{F}$.

Although a low cost does not, in general, imply a small prediction error—and hence does not immediately translate into a bound on the approximation gap in theorem 2.16—the cost-optimal oracle still admits a guarantee comparable to that of the prediction-error minimizer, as formalized below.

Proposition 3.1. *The expected objective value attained by the cost-optimal oracle $f^*_{\texttt{cost}}$ satisfies:*

$$\mathcal{P}(f^*_{\texttt{cost}}) \geq \mathcal{P}^* - \widetilde{O}\left(n^{d-1/2}\sqrt{\varepsilon_{\mathcal{F}}}\right).$$

In particular, if $\mathcal{F}$ is sufficiently rich to contain a good predictor (i.e., $\varepsilon_{\mathcal{F}}$ is small), then the cost-minimizing oracle is guaranteed to yield a small expected approximation gap.

Next, we establish that an oracle learned by ERM is near-optimal once the sample size is sufficiently large. To state a uniform convergence bound, we characterize the complexity of the induced algorithm class $\mathcal{A}$. Since the cost is real-valued, we use the *pseudo-dimension* [Pol90], a standard extension of VC-dimension. The following lemma shows that the complexity of $\mathcal{A}$ is controlled by the VC-dimensions of the coordinate classes comprising $\mathcal{F}$.

Theorem 3.2. *Let $\mathcal{F}$ be a hypothesis class of predictors $f : \Pi \to \{0,1\}^n$ where each coordinate function class $\mathcal{F}_i$ has VC-dimension $\mathrm{VCdim}(\mathcal{F}_i)$. Let $d_{\mathcal{F}} := \sum_{i=1}^{n} \mathrm{VCdim}(\mathcal{F}_i)$. Let $\mathcal{A} = \{\mathcal{A}_f : f \in \mathcal{F}\}$ be the class of algorithms, where each $\mathcal{A}_f$ predicts $\hat{\boldsymbol{x}} = f(\pi)$ and subsequently executes algorithm 1. Then, the pseudo-dimension of the cost functions induced by $\mathcal{A}$ satisfies*

$$\mathrm{Pdim}(\mathcal{A}) \leq C\, d_{\mathcal{F}} \log(e\, d_{\mathcal{F}})$$

for some absolute constant C.

Leveraging standard uniform convergence results [AB99] of the pseudo-dimension and further combine proposition 3.1, we obtain the following comprehensive learning guarantee.

Theorem 3.3. *Let $\mathcal{F}$ be a hypothesis space as described in theorem 3.2. For any $\epsilon > 0$ and $\delta \in (0,1]$, if the sample size m satisfies*

$$m \geq C \left(\frac{H}{\epsilon}\right)^2 \left(d_{\mathcal{F}} \log(e d_{\mathcal{F}}) + \log\left(\frac{1}{\delta}\right)\right)$$

for some absolute constant C, then with probability at least $1-\delta$, any empirical risk minimizer $\hat{f}$ achieves an excess risk error$(\hat{f}) \leq 2\epsilon$, and its expected objective value satisfies

$$\mathcal{P}(\hat{f}) \geq \mathcal{P}^* - \widetilde{O}\left(n^{d-1/2}\sqrt{\varepsilon_{\mathcal{F}}}\right) - 2\epsilon.$$

We have thus established the feasibility of acquiring an oracle for our learning-augmented optimization framework, demonstrating that the sample complexity admits a standard PAC bound of order $\mathrm{polylog}(1/\epsilon, \log(1/\delta), d_{\mathcal{F}})$.

It is worth noting that this result guarantees that ERM identifies an oracle $\hat{f}$ whose performance is near-optimal *in expectation* (i.e., in the average case under $\mathcal{D}$), rather than ensuring pointwise performance guarantees on every individual instance. Furthermore, while our analysis focuses on statistical learnability, the computational efficiency of ERM remains an open problem for future investigation [GR17; BIW22].

4 Applications

In this section, we demonstrate the efficacy of our framework by applying it to two canonical NP-hard problems: MAX-CUT and MAX-k-SAT. These problems serve as representative benchmarks for quadratic and higher-order smooth polynomial optimization, respectively.

MAX-CUT As formalized in example 1, MAX-CUT involves maximizing a quadratic objective function. Since this objective is 2-smooth, our framework is directly applicable. Consider *near-dense* instances characterized by an average vertex degree of $\Omega(n^{0.5+\xi})$ for some $\xi \in (0, 0.5]$; this is equivalent to assuming $\mathrm{OPT} \geq \kappa \cdot n^{1.5+\xi}$ for a constant $\kappa > 0$. Under these conditions, our main result guarantees an approximation ratio of

$$1 - \frac{4}{\kappa}\sqrt{\varepsilon}/n^{\xi}.$$

Consequently, for near-dense graphs, a sufficiently accurate oracle yields solutions that are provably near-optimal.

MAX-k-SAT We next address problems with higher-order constraints through MAX-k-SAT, which optimizes a degree-k multilinear polynomial (see example 2). The corresponding objective is β-smooth with $\beta \leq 4^k$. Specifically, for instances possessing $\Omega(n^{k-0.5+\xi})$ constraints (clauses)—or equivalently, where $\mathrm{OPT} \geq \kappa \cdot n^{k-0.5+\xi}$ for some constant $\kappa > 0$—the approximation ratio is given by

$$1 - \frac{2(2e(k-2)+1)\beta}{\kappa}\sqrt{\varepsilon}/n^{\xi}.$$

In the special case of $k = 3$, this simplifies to $1 - \frac{128(2e+1)}{\kappa}\sqrt{\varepsilon}/n^{\xi}$. These results illustrate the framework's capability to effectively accommodate complex, high-degree dependencies.

Oracle Instantiation and Learnability For both MAX-CUT and MAX-k-SAT, the predictive oracle f can be parameterized using Graph Neural Networks (GNNs) [Sel+19; Gas+19] or Transformers [Pan+25]. Both architectures are known to possess bounded VC-dimensions [Bar+19]. Therefore, theorem 3.3 guarantees that an oracle with near-optimal expected performance can be learned from a polynomial number of samples, thereby establishing the statistical feasibility.

Finally, we note that our framework can be extended to other problems, particularly MAX-k-CSP, a generalization of MAX-k-SAT that aims to satisfy the maximum number of Boolean constraints. We refer the details of this extension to section F in the appendix.

5 Concluding Remarks

This work uses smooth integer programs as a testbed for learning-augmented discrete optimization. We extend the setting of Bampis, Escoffier, and Xefteris [BEX24] by transitioning from a *parsimonious* oracle to a *full information* oracle. By doing so, we aim to understand when and how machine learning models can be combined with classical approximation algorithms in a principled manner.

The results suggest that suitably structured predictions can be injected into the optimization pipeline in a controlled way, leading to algorithms whose behavior varies continuously with oracle quality and whose approximation gaps can be rigorously characterized. At the

same time, the oracle itself need not be treated as a black box: under mild complexity assumptions, it is PAC-learnable via empirical risk minimization, so that statistical and algorithmic considerations can be aligned rather than treated in isolation.

Naturally, the present work remains only a first step. Our results currently rely on smooth polynomial objectives within near-dense regimes, focusing primarily on average-case learnability rather than the computational efficiency of the training procedures. Relaxing these assumptions, analyzing the impact of the hypothesis space, delving deeper into the dynamics of the optimization algorithms, and exploring richer forms of interaction between the oracle and the optimization algorithm are all interesting directions for future investigation.

References

[Aam+25] Anders Aamand, Justin Y. Chen, Siddharth Gollapudi, Sandeep Silwal, and Hao Wu. "Improved approximations for hard graph problems using predictions". In: *Proceedings of the 42nd International Conference on Machine Learning*. Vol. 267. 2025, pp. 73–101 (cit. on p. 21).

[AB99] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 1999 (cit. on pp. 14, 30, 32).

[AGR25] Idan Attias, Xing Gao, and Lev Reyzin. *Learning-augmented algorithms for Boolean satisfiability*. 2025. arXiv: 2505.06146 (cit. on p. 21).

[AKK99] Sanjeev Arora, David Karger, and Marek Karpinski. "Polynomial time approximation schemes for dense instances of NP-hard problems". In: *Journal of Computer and System Sciences* 58.1 (1999), pp. 193–210 (cit. on pp. 2–6, 22, 23, 36).

[Ant+23] Antonios Antoniadis, Joan Boyar, Marek Elias, Lene Monrad Favrholdt, Ruben Hoeksma, Kim S. Larsen, Adam Polak, and Bertrand Simon. "Paging with succinct predictions". In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. 2023, pp. 952–968 (cit. on p. 21).

[Aro+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. "Proof verification and the hardness of approximation problems". In: *Journal of the ACM* 45.3 (May 1998), pp. 501–555 (cit. on p. 20).

[Bal+17] Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. "Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems". In: *Proceedings of the 30th Conference on Learning Theory*. Vol. 65. 2017, pp. 213–274 (cit. on p. 22).

[Bal+18] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. "Learning to branch". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. 2018, pp. 344–353 (cit. on pp. 21, 22).

[Bal+21] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. "Sample complexity of tree search configuration: Cutting planes and beyond". In: *Advances in Neural Information Processing Systems, 34*. 2021 (cit. on p. 22).

[Bal+24] Maria-Florina Balcan, Dan Deblasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. "How much data is sufficient to learn high-performing algorithms?" In: *Journal of the ACM* 71.5 (Oct. 2024) (cit. on p. 22).

[Bam+23] Evripidis Bampis, Bruno Escoffier, Themis Gouleakis, Niklas Hahn, Kostas Lakis, Golnoosh Shahkarami, and Michalis Xefteris. "Learning-augmented online TSP on rings, trees, flowers and (almost) everywhere else". In: *31st Annual European Symposium on Algorithms (ESA 2023)*. Vol. 274. 2023, 12:1–12:17 (cit. on p. 2).

[Bar+19] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. "Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks". In: *Journal of Machine Learning Research* 20.63 (2019), pp. 1–17 (cit. on pp. 12, 15).

[BC23] Xingjian Bai and Christian Coester. "Sorting with predictions". In: *Advances in Neural Information Processing Systems, 36*. 2023 (cit. on p. 21).

[BEX22] Evripidis Bampis, Bruno Escoffier, and Michalis Xefteris. "Canadian traveller problem with predictions". In: *Approximation and Online Algorithms: 20th International Workshop, WAOA 2022, Potsdam, Germany, September 8–9, 2022, Proceedings*. 2022, pp. 116–133 (cit. on p. 2).

[BEX24] Evripidis Bampis, Bruno Escoffier, and Michalis Xefteris. "Parsimonious learning-augmented approximations for dense instances of $\mathcal{NP}$-hard problems". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. 2024, pp. 2700–2714 (cit. on pp. 2–4, 15, 22).

[BIW22] Peter Bartlett, Piotr Indyk, and Tal Wagner. "Generalization bounds for data-driven numerical linear algebra". In: *Proceedings of the 35th Conference on Learning Theory*. Vol. 178. 2022, pp. 2013–2040 (cit. on pp. 14, 22).

[CLS21] Michael B Cohen, Yin Tat Lee, and Zhao Song. "Solving linear programs in the current matrix multiplication time". In: *Journal of the ACM* 68.1 (2021), pp. 1–39 (cit. on p. 12).

[Coh+24] Vincent Cohen-Addad, Tommaso d'Orsi, Anupam Gupta, Euiwoong Lee, and Debmalya Panigrahi. "Learning-augmented approximation algorithms for maximum cut and related problems". In: *Advances in Neural Information Processing Systems, 37*. Vol. 37. 2024, pp. 25961–25980 (cit. on p. 21).

[Din+21] Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. "Faster matchings via learned duals". In: *Advances in Neural Information Processing Systems, 34*. 2021. ISBN: 9781713845393 (cit. on p. 21).

[Erg+22] Jon C. Ergun, Zhili Feng, Sandeep Silwal, David Woodruff, and Samson Zhou. "Learning-augmented k-means clustering". In: *Proceedings of the 10th International Conference on Learning Representations*. 2022, pp. 1–30 (cit. on p. 21).

[FL81] W Fernandez de La Vega and George S. Lueker. "Bin packing can be solved within $1+\varepsilon$ in linear time". In: *Combinatorica* 1.4 (1981), pp. 349–355 (cit. on p. 20).

[FLP16] Dimitris Fotakis, Michael Lampis, and Vangelis Th. Paschos. "Sub-exponential approximation schemes for CSPs: From dense to almost sparse". In: *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*. Vol. 47. 2016, 37:1–37:14 (cit. on p. 22).

[Gas+19] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. "Exact combinatorial optimization with graph convolutional neural networks". In: *Advances in Neural Information Processing Systems, 32*. 2019 (cit. on p. 15).

[GMM25] Suprovat Ghoshal, Konstantin Markarychev, and Yury Markarychev. "Constraint satisfaction problems with advice". In: *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2025, pp. 1202–1221 (cit. on p. 21).

[GR17] Rishi Gupta and Tim Roughgarden. "A PAC approach to application-specific algorithm selection". In: *SIAM Journal on Computing* 46.3 (2017), pp. 992–1017 (cit. on pp. 4, 12, 14, 22).

[GW95] Michel X. Goemans and David P. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM* 42.6 (1995), pp. 1115–1145 (cit. on pp. 8, 21).

[Hås01] Johan Håstad. "Some optimal inapproximability results". In: *Journal of the ACM* 48.4 (July 2001), pp. 798–859 (cit. on p. 2).

[Im+22] Sungjin Im, Ravi Kumar, Aditya Petety, and Manish Purohit. "Parsimonious learning-augmented caching". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. 2022, pp. 9588–9601 (cit. on p. 21).

[Kho+22] Mikhail Khodak, Maria-Florina Balcan, Ameet Talwalkar, and Sergei Vassilvitskii. "Learning predictions for algorithms with predictions". In: *Advances in Neural Information Processing Systems, 35*. 2022 (cit. on p. 22).

[KPS18] Ravi Kumar, Manish Purohit, and Zoya Svitkina. "Improving online algorithms via ML predictions". In: *Advances in Neural Information Processing Systems, 31*. 2018, pp. 9684–9693 (cit. on pp. 2, 21).

[Kra+18] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. "The case for learned index structures". In: *Proceedings of the 2018 International Conference on Management of Data*. Houston, TX, USA, 2018, pp. 489–504 (cit. on p. 21).

[LSV23] Silvio Lattanzi, Ola Svensson, and Sergei Vassilvitskii. "Speeding up Bellman Ford via minimum violation permutations". In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. 2023, pp. 18584–18598 (cit. on p. 21).

[LV18] Thodoris Lykouris and Sergei Vassilvtiskii. "Competitive caching with machine learned advice". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. 2018, pp. 3296–3305 (cit. on p. 21).

[LV21] Thodoris Lykouris and Sergei Vassilvitskii. "Competitive caching with machine learned advice". In: *Journal of the ACM* 68.4 (July 2021) (cit. on pp. 2, 21).

[Mit20] Michael Mitzenmacher. "Scheduling with predictions and the price of misprediction". In: *Proceedings of the 11th Innovations in Theoretical Computer Science Conference*. Ed. by Thomas Vidick. Vol. 151. 2020, 14:1–14:18 (cit. on p. 21).

[MV22] Michael Mitzenmacher and Sergei Vassilvitskii. "Algorithms with predictions". In: *Communications of the ACM* 65.7 (2022), pp. 33–35 (cit. on pp. 2, 12, 21).

[Pan+25] Leyan Pan, Vijay Ganesh, Jacob Abernethy, Chris Esposo, and Wenke Lee. "Can Transformers Reason Logically? A Study in SAT Solving". In: *Proceedings of the 42nd International Conference on Machine Learning*. Vol. 267. 2025, pp. 47632–47671 (cit. on p. 15).

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994 (cit. on p. 38).

[Pol90] David Pollard. "Empirical processes: Theory and applications". In: *NSF-CBMS Regional Conference Series in Probability and Statistics*. Vol. 2. Institute of Mathematical Statistics. 1990, pp. 1–86 (cit. on pp. 14, 31).

[PY91] Christos H. Papadimitriou and Mihalis Yannakakis. "Optimization, approximation, and complexity classes". In: *Journal of Computer and System Sciences* 43.3 (1991), pp. 425–440 (cit. on pp. 2, 20, 38).

[Rag88] Prabhakar Raghavan. "Probabilistic construction of deterministic algorithms: Approximating packing integer programs". In: *Journal of Computer and System Sciences* 37.2 (Oct. 1988), pp. 130–143 (cit. on p. 8).

[RT12] Prasad Raghavendra and Ning Tan. "Approximating CSPs with global cardinality constraints using SDP hierarchies". In: *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2012, pp. 373–387 (cit. on p. 21).

[RT87] Prabhakar Raghavan and Clark D. Tompson. "Randomized rounding: A technique for provably good algorithms and algorithmic proofs". In: *Combinatorica* 7.4 (Dec. 1987), pp. 365–374 (cit. on p. 8).

[Sah75] Sartaj Sahni. "Approximate algorithms for the 0/1 knapsack problem". In: *Journal of the ACM* 22.1 (Jan. 1975), pp. 115–124 (cit. on p. 20).

[Sau72] Norbert Sauer. "On the density of families of sets". In: *Journal of Combinatorial Theory, Series A* 13.1 (1972), pp. 145–147 (cit. on p. 32).

[Sel+19] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. "Learning a SAT solver from single-bit supervision". In: *Proceedings of 7th International Conference on Learning Representations*. 2019 (cit. on p. 15).

[SM24] Rana Shahout and Michael Mitzenmacher. "SkipPredict: When to invest in predictions for scheduling". In: *Advances in Neural Information Processing Systems, 37*. 2024 (cit. on p. 21).

Appendix

The organization of this appendix is as follows:

- Section A presents a comprehensive review of related literature.
- Section B elaborates on the specific subroutines utilized in algorithm 1.
- Sections C and D provide technical details and proofs omitted from the main text.
- Section E generalizes (d-IP) to accommodate general polynomial constraints.
- Section F demonstrates the applicability to the MAX-k-CSP problem, focusing on the smoothness condition.

A Related Work

This section provides a comprehensive review of the literature pertinent to this work. We begin by revisiting classical results in approximation algorithms, followed by a detailed survey of recent advancements in learning-augmented algorithms and data-driven algorithm selection. Finally, we discuss the specific problem of smooth integer programming, which forms the theoretical foundation of this work.

A.1 Approximation Algorithms

Approximation algorithms represent a canonical paradigm for addressing computationally intractable optimization problems, typically offering polynomial-time solvability with provable guarantees on solution quality. The efficacy of such an algorithm is quantified by its *approximation ratio*, defined as the worst-case ratio between the objective value it obtains and that of an optimal solution.

A prominent class of such algorithms is the Polynomial-Time Approximation Scheme (PTAS). Given a problem instance π and a precision parameter $\epsilon \in (0, 1]$, a PTAS delivers a solution with an objective value within a factor of $1 - \epsilon$ of the optimum for maximization problems (or $1+\epsilon$ for minimization). Its running time is polynomial in the problem size for any fixed $\epsilon > 0$, although it may be exponential in $1/\epsilon$ (e.g., $O(n^{1/\epsilon})$). A PTAS thus facilitates a trade-off between approximation accuracy and computational complexity. However, seminal results in computational complexity [PY91; Aro+98] indicate that unless P = NP, many fundamental NP-hard problems—such as Vertex Cover, MAX-k-SAT, and MAX-CUT—do not admit a PTAS. Consequently, PTASs are generally attainable only for specific subclasses of NP-hard problems, such as the Knapsack [Sah75] and Bin Packing [FL81] problems.

A.2 Learning-augmented algorithms

The field of learning-augmented algorithms (LAAs)[1], alternatively referred to as "algorithms with predictions" or "data-driven algorithms," has emerged as a vibrant research area at the intersection of algorithm design, optimization, and machine learning.

Pioneered by Lykouris and Vassilvitskii [LV21] in their seminal work on caching [LV18], this framework challenges traditional worst-case analysis by leveraging machine-learned predictions to enhance algorithmic performance in practical scenarios. The primary objective is to design algorithms that effectively integrate such predictions while satisfying three key properties [MV22]:

- *consistency*, ensuring near-optimal performance when predictions are accurate;
- *smoothness*, guaranteeing that performance degrades gracefully with prediction error; and
- *robustness*, maintaining guarantees comparable to traditional baselines, even when predictions are arbitrarily poor.

This research direction has attracted significant attention, yielding diverse results in areas including scheduling [KPS18; Mit20; SM24], matching [Din+21], sorting [BC23], clustering [Erg+22], indexing [Kra+18], branch-and-bound [Bal+18], shortest paths [LSV23], paging [Ant+23], and caching [Im+22].

In the context of MAX-CUT, Cohen-Addad et al. [Coh+24] investigated two prediction models: *label advice*, where each vertex is assigned its true label with probability $1/2 + \eta$, and *subset advice*, where the true labels are revealed for an η-fraction of vertices. Under the label advice model, they improved upon the classic approximation ratio $\alpha_{GW} \approx 0.878$ of Goemans and Williamson [GW95] to $\alpha_{GW} + \tilde{\Omega}(\eta^4)$; under the subset advice model, they enhanced the ratio $\alpha_{RT} \approx 0.858$ of Raghavendra and Tan [RT12] to $\alpha_{RT} + \tilde{\Omega}(\eta)$. Notably, these models rely on the assumption that prediction are *independent* across vertices.

Subsequently, Aamand et al. [Aam+25] extended the framework of Cohen-Addad et al. [Coh+24] to an edge-based variant, developing algorithms applicable to more general graph problems such as Vertex Cover, Set Cover, and Maximum Independent Set. More recently, Ghoshal, Markarychev, and Markarychev [GMM25] achieved an approximation ratio of $1 - O(1/\eta\sqrt{\Delta})$ under the label advice model via semidefinite programming (SDP) techniques, provided the average degree Δ satisfies $\Delta \geq C/\eta^2$.

Regarding MAX-SAT, Attias, Gao, and Reyzin [AGR25] adopted the subset advice model, proposing a black-box framework that fixes the revealed η-fraction of variables and applies an existing α-approximation algorithm to the residual subproblem. This strategy yields an overall approximation ratio of $\alpha + (1 - \alpha)\eta$.

In contrast to previous works [Coh+24; Aam+25; GMM25; AGR25], which assume *independent* prediction errors, our oracle formulation operates without such restrictive assumptions.

[1] A comprehensive repository of LAAs is maintained at `https://algorithms-with-predictions.github.io/`.

A.3 Data-driven algorithm selection

While the LAA framework offers significant theoretical advancements, it typically assumes access to an external oracle, the practical acquisition of which remains a challenge. A complementary perspective is offered by data-driven algorithm selection, which seeks to learn effective algorithmic configurations directly from problem instance distributions.

Gupta and Roughgarden [GR17] established the theoretical foundations of this field by introducing a PAC-learning framework for algorithm selection. They modeled the problem as identifying the optimal parameter configuration for a family of algorithms (e.g., configurations of a SAT solver) with respect to an unknown distribution of inputs. Their analysis demonstrated that if the algorithm class has bounded complexity—measured by pseudo-dimension—the performance of the learned configuration generalizes to unseen instances with high probability. This framework has since been extended to various domains, including branch-and-bound [Bal+18] and general parameter tuning [Bal+17; Bal+21; Kho+22; BIW22; Bal+24].

In this work, we bridge these two perspectives. We treat the oracle f as a learnable function within a family $\mathcal{F}$ and leverage the PAC framework of Gupta and Roughgarden [GR17] to provide learnability guarantees.

A.4 Smooth integer programs

Many approximation algorithms for NP-hard problems rely on formulating the problem as a linear integer program (LIP). As solving LIPs is NP-complete, all problems in NP admit such a formulation. While many problems possess natural LIP formulations that reveal their structural properties and facilitate approximation, this approach can obscure the intrinsic characteristics of others. Specifically, an approximately optimal solution to the LIP formulation may diverge significantly from the optimal solution of the original problem. In such cases, a more natural formulation often involves a *nonlinear* integer program where the objective function is a low-degree polynomial.

In this context, the seminal work of Arora, Karger, and Karpinski [AKK99] introduced a polynomial-time approximation scheme (PTAS) for smooth integer programming problems in the *dense* regime. Their approach is centered on *exhaustive sampling*: given a precision parameter ϵ, the algorithm selects a random subset of $O(\log n/\epsilon^2)$ variables. It then exhaustively enumerates all possible assignments for this subset—incurring a time complexity of $2^{O(\log n/\epsilon^2)} = n^{O(1/\epsilon^2)}$—and determines the values of the remaining variables via linear programming, followed by randomized rounding for each assignment. Arora, Karger, and Karpinski [AKK99] demonstrated that this approach achieves an approximation gap of $O(n^d\epsilon)$. This strategy provably yields a $(1-\epsilon)$-approximation for *smooth* integer programs where the objective function and constraints are *dense* polynomials of constant degree. This framework was subsequently generalized to handle almost-sparse instances of smooth integer programs, albeit at the cost of sub-exponential time complexity [FLP16].

Recently, Bampis, Escoffier, and Xefteris [BEX24] refined the methodology of Arora, Karger, and Karpinski [AKK99] by incorporating a parsimonious oracle. Instead of performing exhaustive enumeration, their algorithm samples a multiset S of $O(\log n/\epsilon^3)$ variables and

queries an oracle for their optimal assignments. It then proceeds in a manner analogous to Arora, Karger, and Karpinski [AKK99], using linear programming and randomized rounding to determine the values of the remaining variables. This method achieves an approximation gap of $O(n^d(\epsilon + \varepsilon))$, where $\varepsilon = \|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_1$ denotes the prediction error. Within the dense regime, they achieve an approximation ratio of $1 - \epsilon - O(\varepsilon/|S|)$.

In contrast, our approach leverages a *full-information* oracle, which avoids the sampling phase and significantly reduces the additive error from $O(n^d(\epsilon + \varepsilon))$ to $\tilde{O}(n^{d-1/2}\sqrt{\varepsilon})$. This enhancement broadens the applicability of the framework from the dense regime to the *near-dense* regime.

B Detailed Algorithmic Subroutines

In this section, we provide the detailed subroutines for the hierarchical decomposition and the oracle-guided relaxation which are used in algorithm 1. These algorithms constitute the core computational engines of our framework, ensuring efficient polynomial decomposition and robust linear relaxation.

B.1 Hierarchical Decomposition

The DECOMPOSE subroutine (algorithm 2) recursively decomposes the polynomial $p(\boldsymbol{x})$ into its hierarchical components, as established in lemma 2.4.

Subroutine 2: DECOMPOSE

Input : n-variate degree-d β-smooth polynomial $p(\boldsymbol{x})$
Output: Decomposition components $\{p_i(\boldsymbol{x})\}_{i=1}^n$ and constant remainder c

```
if p(x) is constant then
    return p(x)
for i ← 1 to n do
    if some monomial in p(x) contains x_i then
        p_i(x) ← coefficient of x_i in p(x)
        p(x) ← p(x) − x_i p_i(x)
        DECOMPOSE(p_i(x))
c ← p(x)
```

B.1.1 Running Example

To elucidate the DECOMPOSE subroutine, consider the polynomial $p(\boldsymbol{x}) = x_1x_2x_3 + x_2x_4 + 3$ over variables $\{x_1, \ldots, x_4\}$. The algorithm iterates through indices $i = 1$ to 4:

1. **Iteration $i = 1$:** Extract terms containing x_1. The coefficient is $p_1(\boldsymbol{x}) = x_2x_3$. The remainder becomes $p(\boldsymbol{x}) \leftarrow x_2x_4+3$. Recursively decomposing $p_1(\boldsymbol{x})$ yields $p_{1,2}(\boldsymbol{x}) = x_3$ and $p_{1,2,3}(\boldsymbol{x}) = 1$.

2. **Iteration** $i = 2$**:** Extract terms containing x_2 from the updated remainder. The coefficient is $p_2(\boldsymbol{x}) = x_4$. The remainder becomes $p(\boldsymbol{x}) \leftarrow 3$. Recursive decomposition gives $p_{2,4}(\boldsymbol{x}) = 1$.

3. **Iterations** $i = 3, 4$**:** No terms contain x_3 or x_4. The remainder persists as $c = 3$.

The resulting non-zero decomposition components are:

$$p_1(\boldsymbol{x}) = x_2x_3, \quad p_{1,2}(\boldsymbol{x}) = x_3, \quad p_{1,2,3}(\boldsymbol{x}) = 1, \quad p_2(\boldsymbol{x}) = x_4, \quad p_{2,4}(\boldsymbol{x}) = 1, \quad c = 3.$$

This corresponds to the expansion $p(\boldsymbol{x}) = x_1(x_2(x_3 \cdot 1)) + x_2(x_4 \cdot 1) + 3$.

B.2 Relaxation

Given an oracle prediction $\hat{\boldsymbol{x}}$ and an error budget ε, the RELAX procedure (algorithm 3) constructs the linear program (d-LP). It computes the necessary tolerance parameters δ_I for each component constraint to ensure the validity of the relaxation.

Subroutine 3: RELAX

Input : n-variate degree-d β-smooth objective $p(\boldsymbol{x})$, oracle prediction $\hat{\boldsymbol{x}} \in \{0,1\}^n$, error budget ε

Output: Linear programming relaxation (d-LP)

Compute decomposition components $\{p_I(\boldsymbol{x})\}_{I \in \mathcal{I}}$ via DECOMPOSE($p(\boldsymbol{x})$)
foreach $I \in \mathcal{I}$ **do**
  if $|I| < d - 1$ **then**
    $\delta_I \leftarrow 2\beta e n^{d-|I|-1/2}\sqrt{\varepsilon}$
  else
    $\delta_I \leftarrow \beta\sqrt{n\varepsilon}$
  Construct constraint: $q_I(\boldsymbol{x}) \in [p_I(\hat{\boldsymbol{x}}) - \delta_I, p_I(\hat{\boldsymbol{x}}) + \delta_I]$
Relax integrality constraints: $\boldsymbol{x} \in \{0,1\}^n \to \boldsymbol{x} \in [0,1]^n$
Construct the linear program (d-LP)
return (d-LP)

C Omitted Material for the Approximation Framework

This section presents the proofs of the intermediate lemmas and the main approximation theorems stated in section 2. We first establish the properties of smooth polynomials and then derive the bounds for the quadratic and general cases.

C.1 Properties of Smooth Polynomials

Proposition C.1 (Bound on constants)**.** *Let $p(\boldsymbol{x})$ be an n-variate degree-d polynomial that is β-smooth. For any fixed index tuple $I = (i_1, \ldots, i_{d-l})$, the $(d-l)$-level decomposition satisfies*

$$p_I(\boldsymbol{x}) = c_I + \sum_{j \in [n]} x_{I,j} \cdot p_{I,j}(\boldsymbol{x}),$$

where the constant term satisfies $|c_I| \leq \beta n^l$.

This result follows directly from definition 2.1.

Corollary 2.5. *Let $p(\boldsymbol{x})$ be an n-variate β-smooth polynomial of degree d. For any fixed index tuple $I = (i_1, \ldots, i_{d-l})$ and any $\boldsymbol{x} \in \{0,1\}^n$, the component $p_I(\boldsymbol{x})$ satisfies*

$$|p_I(\boldsymbol{x})| \leq \beta(l+1)n^l.$$

Proof. The proof proceeds by induction on l. For $l = 0$, $p_I(\boldsymbol{x})$ reduces to the constant c_I. By proposition C.1, we have $|c_I| \leq \beta$, which is consistent with $\beta(0+1)n^0 = \beta$.

Assume the bound holds for level $l-1$; that is, for any index tuple I' of length $d-(l-1)$ and any $\boldsymbol{x} \in \{0,1\}^n$, $|p_{I'}(\boldsymbol{x})| \leq \beta\, l\, n^{l-1}$. Fix $I = (i_1, \ldots, i_{d-l})$. Applying the decomposition:

$$|p_I(\boldsymbol{x})| \leq |c_I| + \sum_{j \in [n]} x_{I,j}\, |p_{I,j}(\boldsymbol{x})|.$$

Using proposition C.1, we have $|c_I| \leq \beta n^l$. By the inductive hypothesis, $|p_{I,j}(\boldsymbol{x})| \leq \beta\, l\, n^{l-1}$ for each j. Since $x_{I,j} \in \{0,1\}$, the summation contains at most n non-zero terms. Therefore,

$$|p_I(\boldsymbol{x})| \leq \beta n^l + n \cdot (\beta\, l\, n^{l-1}) = \beta n^l (1+l) = \beta(l+1)n^l.$$

This completes the proof. □

C.2 Analysis of the Quadratic Case

Lemma 2.7. *Let $p(\boldsymbol{x})$ be an n-variate β-smooth polynomial of degree 2 with the decomposition $p(\boldsymbol{x}) = \sum_{i \in [n]} x_i p_i(\boldsymbol{x}) + c$. For any two binary vectors $\boldsymbol{x}^*, \hat{\boldsymbol{x}} \in \{0,1\}^n$, let $\varepsilon = \|\boldsymbol{x}^* - \hat{\boldsymbol{x}}\|_1$. Then, for any $i \in [n]$, the deviation of the linear component is bounded by:*

$$|p_i(\hat{\boldsymbol{x}}) - p_i(\boldsymbol{x}^*)| \leq \beta\sqrt{n}\sqrt{\varepsilon}.$$

Proof. Recall that for a quadratic polynomial, the component $p_i(\boldsymbol{x})$ is linear, i.e., $p_i(\boldsymbol{x}) = c_i + \sum_{j \in [n]} c_{ij} x_j$. The difference is thus

$$p_i(\hat{\boldsymbol{x}}) - p_i(\boldsymbol{x}^*) = \sum_{j \in [n]} c_{ij}(\hat{x}_j - x_j^*).$$

By the β-smoothness property, the quadratic coefficients satisfy $|c_{ij}| \leq \beta$. Applying the Cauchy-Schwarz inequality yields

$$\left| \sum_{j \in [n]} c_{ij}(\hat{x}_j - x_j^*) \right| \leq \sqrt{\sum_{j \in [n]} c_{ij}^2} \cdot \sqrt{\sum_{j \in [n]} (\hat{x}_j - x_j^*)^2}.$$

The first term is bounded by $\sqrt{n\beta^2} = \beta\sqrt{n}$. For the second term, since the variables are binary, $(\hat{x}_j - x_j^*)^2 = |\hat{x}_j - x_j^*|$, and thus the sum equals $\|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_1 = \varepsilon$. Therefore, the deviation is at most $\beta\sqrt{n}\sqrt{\varepsilon}$. □

In the linear setting, we establish the following lemma regarding the rounding error:

Lemma C.2. *Let $\boldsymbol{y} \in [0,1]^n$, and let $\boldsymbol{z} \in \{0,1\}^n$ be a vector obtained via independent randomized rounding, where $\Pr[z_i = 1] = y_i$ for all $i \in [n]$. Suppose the coefficients $(c_i)_{i=1}^n$ satisfy $|c_i| \leq \beta$. Then, for any $k \geq 1$, with probability at least $1 - 2/n^{k+1}$,*

$$\left| \sum_{i=1}^{n} c_i(y_i - z_i) \right| \leq \beta \sqrt{\tfrac{k+1}{2}} \sqrt{n \ln n}.$$

Proof. Let $X = \sum_{i=1}^n c_i(y_i - z_i)$. Due to the independence of the rounding process, $\mathbb{E}[z_i] = y_i$, which implies $\mathbb{E}[X] = 0$. Consider X as a function $f(z_1, \ldots, z_n) = \sum_{i=1}^n c_i(y_i - z_i)$ of the independent variables $z_1, \ldots, z_n$. Altering a single component z_i changes the value of f by at most $|c_i| \leq \beta$. Consequently, the bounded difference condition holds with $\Delta_i \leq \beta$ for all i. By McDiarmid's inequality, for any $t > 0$, we have:

$$\Pr\left(|X| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n \Delta_i^2}\right) \leq 2\exp\left(-\frac{2t^2}{n\beta^2}\right).$$

Choosing $t = \beta \sqrt{\frac{k+1}{2}} \sqrt{n \ln n}$ yields the claim with probability at least $1 - 2/n^{k+1}$. □

C.3 Proof of theorem 2.9

Theorem 2.9. *Let $p(\boldsymbol{x})$ be an n-variate quadratic β-smooth polynomial. Let $\boldsymbol{y} \in [0,1]^n$ be a fractional vector, and let $\boldsymbol{z} \in \{0,1\}^n$ be the integral vector derived via independent randomized rounding with $\Pr[z_i = 1] = y_i$. For any $k \geq 1$, with probability at least $1 - 4/n^k$, the following bound holds:*

$$|p(\boldsymbol{z}) - p(\boldsymbol{y})| \leq 3n\beta\sqrt{\frac{k+1}{2}}\sqrt{n \ln n}. \tag{3}$$

Proof. Observe that the deviation can be decomposed as follows:

$$\left| \sum_{i \in [n]} \left[z_i p_i(\boldsymbol{z}) - y_i p_i(\boldsymbol{y}) \right] \right| \leq \left| \sum_{i \in [n]} z_i \big(p_i(\boldsymbol{z}) - p_i(\boldsymbol{y}) \big) \right| + \left| \sum_{i \in [n]} (y_i - z_i) p_i(\boldsymbol{y}) \right|.$$

The first term can be bounded by noting that $z_i \in \{0,1\}$. Applying a union bound over $i \in [n]$, lemma C.2 holds simultaneously for all coordinates with probability at least $1-2/n^k$. Specifically,

$$\left| \sum_{i\in[n]} z_i \big(p_i(\boldsymbol{z}) - p_i(\boldsymbol{y})\big) \right| \leq n \cdot \beta \sqrt{\frac{k+1}{2}} \sqrt{n \ln n}, \tag{7}$$

which holds with probability at least $1 - 2/n^k$.

For the second term, applying an analogous McDiarmid argument to $Y = \sum_{i\in[n]}(y_i - z_i)p_i(\boldsymbol{y})$, and using the bound $|p_i(\boldsymbol{y})| \leq 2\beta n$, yields

$$\left| \sum_{i\in[n]} (y_i - z_i) p_i(\boldsymbol{y}) \right| \leq n \cdot 2\beta \sqrt{\frac{k+1}{2}} \sqrt{n \ln n}, \tag{8}$$

which holds with probability at least $1 - 2/n^{k+1}$.

Finally, combining (7) and (8) via a union bound yields the desired result. □

C.4 Proof of theorem 2.11

Theorem 2.11. *Let $\boldsymbol{x}^*$ be the optimal solution to (2-IP), and let $\boldsymbol{z}$ be the integral solution obtained via randomized rounding from the fractional solution to (2-LP). With probability at least $1 - 4n^{-k}$, we have*

$$p(\boldsymbol{z}) \geq p(\boldsymbol{x}^*) - 2\beta n^{3/2}\sqrt{\varepsilon} - 3\beta n^{3/2}\sqrt{\frac{k+1}{2}\ln n}\,. \tag{4}$$

Proof. Our approach proceeds by relaxing (2-IP) to (2-LP) using the oracle, solving for a fractional solution $\boldsymbol{y}$, and subsequently applying randomized rounding to recover an integral solution $\boldsymbol{z}$. Combining the rounding error from (3) and the relaxation gap from (2), we derive the following bound with probability at least $1 - 4n^{-k}$:

$$\begin{aligned} p(\boldsymbol{z}) &\geq p(\boldsymbol{y}) - 3\beta\sqrt{\frac{k+1}{2}} \cdot n^{3/2}\sqrt{\ln n} \\ &\geq p(\boldsymbol{x}^*) - 2\beta n^{3/2}\sqrt{\varepsilon} - 3\beta\sqrt{\frac{k+1}{2}} \cdot n^{3/2}\sqrt{\ln n}\,. \end{aligned} \tag{9}$$

In the case of dense instances (e.g., MAX-CUT with average degree $\Omega(n)$), the optimal value scales as $p(\boldsymbol{x}^*) = \mathrm{OPT} = \Theta(n^2)$. Consequently, the additive bound translates to a multiplicative guarantee:

$$p(\boldsymbol{z}) \geq \mathrm{OPT} \cdot \left(1 - \tilde{O}\left(\sqrt{\frac{\varepsilon}{n}}\right)\right). \tag{10}$$

□

C.5 Proof of theorem 2.12

Theorem 2.12. *Let $\boldsymbol{y}$ be the optimal solution to the relaxed problem (d-LP) and $\boldsymbol{x}^*$ be the optimal solution to the original integer program (d-IP). The relaxation gap is bounded by:*

$$p(\boldsymbol{y}) \geq p(\boldsymbol{x}^*) - 2\left[2e(d-2)+1\right]\beta n^{d-1/2}\sqrt{\varepsilon}.$$

Proof. Define the approximation error function $\textsc{gap}(p,q)(\boldsymbol{x}) := |p(\boldsymbol{x}) - q(\boldsymbol{x})|$. By the optimality of $\boldsymbol{y}$ for the maximization problem (d-LP) and the feasibility of $\boldsymbol{x}^*$, we have:

$$\begin{aligned} p(\boldsymbol{y}) &\geq q(\boldsymbol{y}) - \textsc{gap}(p,q)(\boldsymbol{y}) \\ &\geq q(\boldsymbol{x}^*) - \textsc{gap}(p,q)(\boldsymbol{y}) \\ &\geq p(\boldsymbol{x}^*) - \textsc{gap}(p,q)(\boldsymbol{y}) - \textsc{gap}(p,q)(\boldsymbol{x}^*). \end{aligned} \tag{11}$$

Thus, it suffices to bound $\textsc{gap}(p,q)(\boldsymbol{x})$ for any $\boldsymbol{x} \in [0,1]^n$.

Consider the gap for a polynomial term $p_I(\boldsymbol{x})$ and its linear approximation $q_I(\boldsymbol{x})$:

$$\begin{aligned} \textsc{gap}(p_I, q_I)(\boldsymbol{x}) &= \left| \sum_{j\in[n]} x_{I,j}\left(p_{I,j}(\boldsymbol{x}) - p_{I,j}(\hat{\boldsymbol{x}})\right) \right| \\ &\leq \sum_{j\in[n]} |p_{I,j}(\boldsymbol{x}) - p_{I,j}(\hat{\boldsymbol{x}})| \\ &\leq \sum_{j\in[n]} \left(|p_{I,j}(\boldsymbol{x}) - q_{I,j}(\boldsymbol{x})| + |q_{I,j}(\boldsymbol{x}) - p_{I,j}(\hat{\boldsymbol{x}})|\right) \\ &\leq \sum_{j\in[n]} \textsc{gap}(p_{I,j}, q_{I,j})(\boldsymbol{x}) + \sum_{j\in[n]} \delta_{I,j}, \end{aligned} \tag{12}$$

where the first inequality uses $x_{I,j} \in [0,1]$ and the triangle inequality, and the last follows from the constraint $q_{I,j}(\boldsymbol{x}) \in p_{I,j}(\hat{\boldsymbol{x}}) \pm \delta_{I,j}$.

Applying this recurrence relation over the decomposition tree yields:

$$\textsc{gap}(p,q)(\boldsymbol{x}) \leq \sum_{I\in\mathcal{I}} \delta_I.$$

Substituting the bounds for δ_I derived in section 2.3.1:

- For linear terms ($|I| = d-1$), $\delta_I \leq \beta\sqrt{n}\sqrt{\varepsilon}$.
- For higher-order terms ($1 \leq |I| < d-1$), $\delta_I \leq 2\beta e n^{d-|I|-1/2}\sqrt{\varepsilon}$.

Summing over all $I \in \mathcal{I}$, we obtain:

$$\sum_{I\in\mathcal{I}} \delta_I \leq \left[2e(d-2)+1\right]\beta n^{d-1/2}\sqrt{\varepsilon}.$$

Combining the errors for $\boldsymbol{y}$ and $\boldsymbol{x}^*$ completes the proof. □

C.6 Omitted Results for General Case

Theorem 2.13. *Let $\boldsymbol{y} \in [0,1]^n$, and let $\boldsymbol{z} \in \{0,1\}^n$ be generated via independent randomized rounding where $\Pr[z_i = 1] = y_i$ for all $i \in [n]$. Consider an n-variate degree-d polynomial $p(\boldsymbol{x})$ that is β-smooth. For any $k > d$, with probability at least $1 - 2d/n^{\,k+1-(d-1)}$,*

$$|p(\boldsymbol{y}) - p(\boldsymbol{z})| \le \big(1 + 2e\,(d-2)\big)\,\beta\,n^{d-1}\,\sqrt{\tfrac{k+1}{2}}\,\sqrt{n \ln n}.$$

Proof. We proceed by induction over the decomposition depth of p. Fix an index tuple $I = (i_1, \ldots, i_{d-l})$ with $l \in [d-1]$. Using the decomposition,

$$\begin{aligned}\big|p_I(\boldsymbol{z}) - p_I(\boldsymbol{y})\big| &= \Big|\sum_{i\in[n]} z_{I,i}\,p_{I,i}(\boldsymbol{z}) - \sum_{i\in[n]} y_{I,i}\,p_{I,i}(\boldsymbol{y})\Big| \\ &\le \Big|\sum_{i\in[n]} z_{I,i}\,\big(p_{I,i}(\boldsymbol{z}) - p_{I,i}(\boldsymbol{y})\big)\Big| + \Big|\sum_{i\in[n]} (z_{I,i} - y_{I,i})\,p_{I,i}(\boldsymbol{y})\Big|.\end{aligned}$$

For the second term, invoking lemma C.2 together with the bound $|p_{I,i}(\boldsymbol{y})| \le 2\beta e\,n^{d-|I|-1}$ from lemma 2.6, we obtain, with probability at least $1 - 2/n^{k+1}$,

$$\Big|\sum_{i\in[n]} (z_{I,i} - y_{I,i})\,p_{I,i}(\boldsymbol{y})\Big| \le 2\beta e\,n^{d-|I|-1}\,\sqrt{\tfrac{k+1}{2}}\,\sqrt{n \ln n}.$$

For the first term, suppose a uniform per-coordinate bound $|p_{I,i}(\boldsymbol{z}) - p_{I,i}(\boldsymbol{y})| \le \Delta$ holds with failure probability at most δ. A union bound then gives

$$\Big|\sum_{i\in[n]} z_{I,i}\,\big(p_{I,i}(\boldsymbol{z}) - p_{I,i}(\boldsymbol{y})\big)\Big| \le n\,\Delta,$$

with probability at least $1 - n\delta$. At the base level, $|I| = d-1$, lemma C.2 implies

$$|p_I(\boldsymbol{z}) - p_I(\boldsymbol{y})| \le \beta\,\sqrt{\tfrac{k+1}{2}}\,\sqrt{n \ln n}$$

with probability at least $1 - 2/n^{k+1}$. Propagating this bound through the $d-1$ levels of the decomposition and aggregating the contributions yields

$$|p(\boldsymbol{z}) - p(\boldsymbol{y})| \le \big(1 + 2e\,(d-2)\big)\,\beta\,n^{d-1}\,\sqrt{\tfrac{k+1}{2}}\,\sqrt{n \ln n},$$

which holds with probability at least $1 - 2d/n^{\,k+1-(d-1)}$. □

C.7 Proof of theorem 2.15

Below we first demonstrate the procedure of greedy rounding in algorithm 4.

Theorem 2.15. *Let $p(\boldsymbol{x})$ be a multilinear polynomial. For any fractional solution $\boldsymbol{y} \in [0,1]^n$, the greedy deterministic rounding procedure yields an integral vector $\boldsymbol{z} \in \{0,1\}^n$ such that:*

$$p(\boldsymbol{z}) \ge p(\boldsymbol{y}).$$

Algorithm 4: Greedy Deterministic Rounding

Input : $\boldsymbol{y} \in [0,1]^n$, multilinear objective $p(\boldsymbol{x})$
Output: Integral solution $\boldsymbol{z} \in \{0,1\}^n$
for $i \leftarrow 1$ **to** n **do**
  $g_i(t) := p(y_1, \ldots, y_{i-1}, t, y_{i+1}, \ldots, y_n)$;
  $z_i \leftarrow \text{argmax}_{t \in \{0,1\}} g_i(t)$;
  Update $\boldsymbol{y}$ by setting $y_i \leftarrow z_i$;
return $\boldsymbol{z} = \boldsymbol{y}$;

Proof. Let $\boldsymbol{y}^{(i)}$ denote the solution after rounding the first i variables. We show that $p(\boldsymbol{y}^{(i)}) \geq p(\boldsymbol{y}^{(i-1)})$ for all $i \in [n]$. Consider the i-th step where we determine z_i. Since $p(\boldsymbol{x})$ is multilinear, the function $g_i(t)$ defined by fixing all other variables to their values in $\boldsymbol{y}^{(i-1)}$ is linear (affine) in t. A linear function on the interval $[0,1]$ achieves its maximum at one of the endpoints. Therefore:

$$\max_{t \in \{0,1\}} g_i(t) \geq g_i(y_i^{(i-1)}),$$

which implies $p(\boldsymbol{y}^{(i)}) \geq p(\boldsymbol{y}^{(i-1)})$. By induction, $p(\boldsymbol{z}) = p(\boldsymbol{y}^{(n)}) \geq p(\boldsymbol{y}^{(0)}) = p(\boldsymbol{y})$. □

D Omitted Material for Learnability Guarantees

In this section, we provide the complete proofs for the learnability results presented in section 3. We begin by recalling standard definitions from learning theory and then proceed to bound the pseudo-dimension of our algorithm class.

D.1 Standard Definitions and Results

Definition D.1 ((ϵ, δ)-learnable)**.** A learning algorithm L is said to (ϵ, δ)-learn the optimal algorithm in $\mathcal{A}$ using m samples if, for every distribution $\mathcal{D}$ over Π, with probability at least $1 - \delta$ over the draw of $\{\pi_1, \ldots, \pi_m\} \sim \mathcal{D}$, L outputs an algorithm $\hat{A} \in \mathcal{A}$ such that $\text{error}(\hat{A}; \mathcal{D}) \leq \epsilon$.

Definition D.2 (Pseudo-dimension, [AB99])**.** Let $\mathcal{H}$ be a set of real-valued functions defined on Π. A finite subset $S = \{\pi_1, \ldots, \pi_m\} \subseteq \Pi$ is *(pseudo-)shattered* by $\mathcal{H}$ if there exist real-valued witnesses $r_1, \ldots, r_m$ such that for each subset $T \subseteq S$, there exists a function $h \in \mathcal{H}$ satisfying $h(\pi_i) > r_i$ if and only if $i \in T$. The *pseudo-dimension* of $\mathcal{H}$, denoted by $\text{Pdim}(\mathcal{H})$, is the maximum cardinality m for which some subset $S \subseteq \Pi$ of size m is pseudo-shattered by $\mathcal{H}$.

Lemma D.3 (Uniform convergence, [AB99])**.** *Let $\mathcal{H}$ be a class of functions mapping Π to $[0, H]$, with pseudo-dimension $d_{\mathcal{H}}$. For any distribution $\mathcal{D}$ over Π, any precision $\epsilon > 0$, and*

any confidence parameter $\delta \in (0, 1]$, if the sample size satisfies

$$m \geq C \left(\frac{H}{\epsilon}\right)^2 \left(d_{\mathcal{H}} + \log\left(\frac{1}{\delta}\right)\right), \tag{13}$$

where C is an absolute constant, then with probability at least $1-\delta$ over the draw of $\{\pi_1, \ldots, \pi_m\} \sim \mathcal{D}$, the following uniform bound holds:

$$\sup_{h \in \mathcal{H}} \left| \mathbb{E}_{\pi \sim \mathcal{D}}[h(\pi)] - \frac{1}{m} \sum_{i=1}^{m} h(\pi_i) \right| \leq \epsilon.$$

The following lemma formally bridges uniform convergence and the sample complexity of ERM, establishing that a bounded pseudo-dimension is sufficient for learnability.

Lemma D.4. *Fix $\epsilon > 0$, $\delta \in (0, 1]$, instance space Π, and cost function* `COST`*. Suppose the algorithm class $\mathcal{A}$ induces a cost function class with pseudo-dimension $d_{\mathcal{A}}$. Then, any ERM algorithm $(2\epsilon, \delta)$-learns the optimal algorithm in $\mathcal{A}$ with a sample size m satisfying:*

$$m \geq C \left(\frac{H}{\epsilon}\right)^2 \left(d_{\mathcal{A}} + \log\left(\frac{1}{\delta}\right)\right), \tag{14}$$

where C is an absolute constant.

Proof. Let $\mathcal{H} = \{h_A : \Pi \to [0, H] \mid A \in \mathcal{A}\}$ where $h_A(\pi) := \texttt{COST}(A, \pi)$. By assumption, $\text{Pdim}(\mathcal{H}) = d_{\mathcal{A}}$. If m satisfies (13), lemma D.3 guarantees that with probability at least $1 - \delta$,

$$\sup_{A \in \mathcal{A}} \left| \mathbb{E}_{\pi \sim \mathcal{D}}[h_A(\pi)] - \tfrac{1}{m} \sum_{i=1}^{m} h_A(\pi_i) \right| \leq \epsilon.$$

Let $A_{\mathcal{D}} \in \arg\min_{A \in \mathcal{A}} \mathbb{E}_{\pi \sim \mathcal{D}}[h_A(\pi)]$ be the optimal algorithm and let $\hat{A}$ be the ERM solution, which minimizes $\frac{1}{m} \sum_{i=1}^{m} h_A(\pi_i)$. We have:

$$\begin{aligned}
\mathbb{E}_{\pi \sim \mathcal{D}}[h_{\hat{A}}(\pi)] &\leq \tfrac{1}{m} \sum_{i=1}^{m} h_{\hat{A}}(\pi_i) + \epsilon \\
&\leq \tfrac{1}{m} \sum_{i=1}^{m} h_{A_{\mathcal{D}}}(\pi_i) + \epsilon \\
&\leq \mathbb{E}_{\pi \sim \mathcal{D}}[h_{A_{\mathcal{D}}}(\pi)] + 2\epsilon,
\end{aligned}$$

where the first and third inequalities follow from uniform convergence, and the second from the definition of ERM. Rearranging terms yields $\mathbb{E}_{\pi \sim \mathcal{D}}[h_{\hat{A}}(\pi)] - \mathbb{E}_{\pi \sim \mathcal{D}}[h_{A_{\mathcal{D}}}(\pi)] \leq 2\epsilon$. Thus, $\hat{A}$ has an error of at most 2ϵ with probability at least $1 - \delta$. □

The theoretical guarantee of ERM relies on the uniform convergence of empirical means to their true expectations. We characterize the complexity of the function class using the notion of *pseudo-dimension* [Pol90], which generalizes the VC-dimension to real-valued functions.

The subsequent theorem establishes that if $\mathcal{F}$ possesses a bounded VC-dimension, the induced algorithm class $\mathcal{A}$ also exhibits bounded complexity.

Theorem 3.2. *Let $\mathcal{F}$ be a hypothesis class of predictors $f : \Pi \to \{0,1\}^n$ where each coordinate function class $\mathcal{F}_i$ has VC-dimension $\mathrm{VCdim}(\mathcal{F}_i)$. Let $d_{\mathcal{F}} := \sum_{i=1}^{n} \mathrm{VCdim}(\mathcal{F}_i)$. Let $\mathcal{A} = \{\mathcal{A}_f : f \in \mathcal{F}\}$ be the class of algorithms, where each $\mathcal{A}_f$ predicts $\hat{\boldsymbol{x}} = f(\pi)$ and subsequently executes algorithm 1. Then, the pseudo-dimension of the cost functions induced by $\mathcal{A}$ satisfies*

$$\mathrm{Pdim}(\mathcal{A}) \leq C\, d_{\mathcal{F}} \log(e\, d_{\mathcal{F}})$$

for some absolute constant C.

Proof. Since the algorithms in $\mathcal{A}$ involve internal randomness (e.g., in the randomized rounding step), we formalize the cost function by explicitly treating the random seed as part of the input. Let Ξ denote the space of internal random seeds. We define the cost function $h_f : \Pi \times \Xi \to [0, H]$ as $h_f(\pi, \xi) := \texttt{COST}(\mathcal{A}_f(\pi; \xi))$, where $\mathcal{A}_f(\pi; \xi)$ denotes the execution of the algorithm with predictor f and random seed ξ. We now bound the pseudo-dimension of the function class $\mathcal{H} = \{h_f : f \in \mathcal{F}\}$ on the augmented domain $\Pi \times \Xi$.

Fix a finite set of augmented instances $S = \{(\pi_1, \xi_1), \ldots, (\pi_m, \xi_m)\} \subseteq \Pi \times \Xi$ and witnesses $r_1, \ldots, r_m \in \mathbb{R}$. For each $f \in \mathcal{F}$, define the labeling vector $\ell_f \in \{0,1\}^m$ such that $\ell_f(j) = \mathbb{I}\{h_f(\pi_j, \xi_j) > r_j\}$.

Recall that the algorithm proceeds in three stages: 1. Prediction: $\hat{\boldsymbol{x}} = f(\pi)$. 2. Relaxation: Compute fractional solution $\boldsymbol{y} = \mathrm{Pipeline}(\pi, \hat{\boldsymbol{x}})$. 3. Rounding: Compute integral solution $\boldsymbol{z} = \mathrm{Round}(\boldsymbol{y}, \xi)$.

Crucially, for a fixed instance π_j and fixed random seed ξ_j, the final cost $\texttt{COST}(\boldsymbol{z})$ is uniquely determined by the prediction $\hat{\boldsymbol{x}}_j = f(\pi_j)$. Let $P_f \in \{0,1\}^{n \times m}$ be the prediction matrix on the underlying instances $\{\pi_1, \ldots, \pi_m\}$, where $[P_f]_{i,j} = f_i(\pi_j)$. Since the mapping from P_f to the costs on S is deterministic (given the fixed S), the number of distinct labelings $\{\ell_f\}$ is upper-bounded by the number of distinct prediction matrices $\{P_f\}$.

For each coordinate i, let $\mathcal{F}_i = \{f_i : f \in \mathcal{F}\}$ with $\mathrm{VCdim}(\mathcal{F}_i) = d_i$. By Sauer's Lemma [Sau72], the number of distinct binary vectors $(f_i(\pi_1), \ldots, f_i(\pi_m))$ on S is at most $\sum_{k=0}^{d_i} \binom{m}{k} \leq (em/d_i)^{d_i}$. Even if the coordinates are coupled, the set of valid prediction matrices is a subset of the Cartesian product of the coordinate-wise projections. Therefore, the total number of distinct matrices P_f on S is at most the product of the counts for each coordinate:

$$\prod_{i=1}^{n} \left(\frac{em}{d_i}\right)^{d_i} \leq (em)^{\sum d_i} = (em)^{d_{\mathcal{F}}}.$$

Consequently, if S is pseudo-shattered by $\mathcal{H}$, then all 2^m possible labelings must be realizable. Therefore, we must have $2^m \leq (em)^{d_{\mathcal{F}}}$. Taking logarithms implies $m \log 2 \leq d_{\mathcal{F}}(\log m + 1)$. Standard algebraic manipulation (see, e.g., [AB99]) shows that this inequality cannot hold if $m > C d_{\mathcal{F}} \log(e d_{\mathcal{F}})$ for a sufficiently large constant C. Thus, $\mathrm{Pdim}(\mathcal{H}) \leq C d_{\mathcal{F}} \log(e d_{\mathcal{F}})$. □

D.2 Proof of proposition 3.1

Proposition 3.1. *The expected objective value attained by the cost-optimal oracle $f^*_{\texttt{cost}}$ satisfies:*

$$\mathcal{P}(f^*_{\texttt{cost}}) \geq \mathcal{P}^* - \widetilde{O}\left(n^{d-1/2}\sqrt{\varepsilon_{\mathcal{F}}}\right).$$

Proof. The proof proceeds in two steps.

Step 1: Optimality of cost minimization. Recall that the cost function is defined as $\texttt{COST}(\mathcal{A}_f, \pi) = H - p(\boldsymbol{z}_f(\pi))$, where H is a global upper bound on the objective value. By definition, f^*_{cost} minimizes the expected cost, which implies:

$$\mathbb{E}[\texttt{COST}(\mathcal{A}_{f^*_{\text{cost}}}, \pi)] \leq \mathbb{E}[\texttt{COST}(\mathcal{A}_{f^*_{\text{pred}}}, \pi)].$$

Substituting the definition of cost, we obtain:

$$\mathbb{E}[H - p(\boldsymbol{z}_{f^*_{\text{cost}}}(\pi))] \leq \mathbb{E}[H - p(\boldsymbol{z}_{f^*_{\text{pred}}}(\pi))] \implies \mathcal{P}(f^*_{\text{cost}}) \geq \mathcal{P}(f^*_{\text{pred}}).$$

Step 2: Approximation guarantee. We leverage the approximation guarantee established in theorem 2.16. For any instance π and prediction $\hat{\boldsymbol{x}}$, the solution $\boldsymbol{z}_f(\pi)$ satisfies:

$$p(\boldsymbol{z}_f(\pi)) \geq p(\boldsymbol{x}^*(\pi)) - C_1 n^{d-1/2}\sqrt{\|\hat{\boldsymbol{x}} - \boldsymbol{x}^*(\pi)\|_1} - C_2 n^{d-1/2}\sqrt{\log n},$$

where C_1, C_2 are problem-dependent constants. Taking the expectation over $\pi \sim \mathcal{D}$ with $f = f^*_{\text{pred}}$ yields:

$$\mathcal{P}(f^*_{\text{pred}}) \geq \mathcal{P}^* - C_1 n^{d-1/2}\mathbb{E}\left[\sqrt{\|f^*_{\text{pred}}(\pi) - \boldsymbol{x}^*(\pi)\|_1}\right] - \widetilde{O}(n^{d-1/2}).$$

Since the square root function is concave, Jensen's inequality implies $\mathbb{E}[\sqrt{X}] \leq \sqrt{\mathbb{E}[X]}$. Thus,

$$\mathcal{P}(f^*_{\text{pred}}) \geq \mathcal{P}^* - O\left(n^{d-1/2}\sqrt{\varepsilon_{\mathcal{F}}}\right) - \widetilde{O}(n^{d-1/2}).$$

The term involving the prediction error is the dominant factor that the learning algorithm seeks to optimize. Simplifying the expression using $\widetilde{O}$ notation to capture the leading order dependencies completes the proof. □

D.3 Proof of theorem 3.3

Theorem 3.3. *Let $\mathcal{F}$ be a hypothesis space as described in theorem 3.2. For any $\epsilon > 0$ and $\delta \in (0, 1]$, if the sample size m satisfies*

$$m \geq C\left(\frac{H}{\epsilon}\right)^2\left(d_{\mathcal{F}}\log(ed_{\mathcal{F}}) + \log\left(\frac{1}{\delta}\right)\right)$$

for some absolute constant C, then with probability at least $1-\delta$, any empirical risk minimizer $\hat{f}$ achieves an excess risk error$(\hat{f}) \leq 2\epsilon$, and its expected objective value satisfies

$$\mathcal{P}(\hat{f}) \geq \mathcal{P}^* - \widetilde{O}\left(n^{d-1/2}\sqrt{\varepsilon_{\mathcal{F}}}\right) - 2\epsilon.$$

Proof. Let f^*_{cost} be the optimal oracle in $\mathcal{F}$ that minimizes the expected cost. By lemma D.4, the ERM algorithm $(2\epsilon, \delta)$-learns f^*_{cost}. This implies that with probability at least $1 - \delta$, the excess risk satisfies $\text{error}(\mathcal{A}_{\hat{f}}; \mathcal{D}) \leq 2\epsilon$. Substituting the definition of error:

$$\mathbb{E}[\texttt{COST}(\mathcal{A}_{\hat{f}})] - \mathbb{E}[\texttt{COST}(\mathcal{A}_{f^*_{\text{cost}}})] \leq 2\epsilon.$$

Recalling that $\texttt{COST}(\mathcal{A}_f, \pi) = H - p(\boldsymbol{z}_f(\pi))$, this inequality translates to the objective value:

$$\mathcal{P}(f^*_{\text{cost}}) - \mathcal{P}(\hat{f}) \leq 2\epsilon \implies \mathcal{P}(\hat{f}) \geq \mathcal{P}(f^*_{\text{cost}}) - 2\epsilon.$$

We now invoke proposition 3.1 to lower bound the performance of the cost-minimizing oracle:

$$\mathcal{P}(f^*_{\text{cost}}) \geq \mathcal{P}^* - \widetilde{O}\left(n^{d-1/2}\sqrt{\varepsilon_{\mathcal{F}}}\right).$$

Combining these two inequalities yields the stated result. □

E Generalization to Constrained Optimization

In this section, we extend the scope of our investigation from unconstrained optimization to a more general class of problems involving polynomial constraints. This generalization allows us to model complex scenarios where decision variables are subject to structural or resource limitations. Formally, we address the following constrained integer programming problem:

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & p(\boldsymbol{x}) \\ \text{s.t.} \quad & p_c(\boldsymbol{x}) \in [L_c, U_c] \quad \forall c \in \mathcal{C} \\ & \boldsymbol{x} \in \{0,1\}^n, \end{aligned} \tag{d-IP'}$$

where the objective function $p(\boldsymbol{x})$ and each constraint function $p_c(\boldsymbol{x})$ (indexed by $c \in \mathcal{C}$) are n-variate polynomials of degree at most d that satisfy the β-smoothness property.

To tackle this problem, we generalize the oracle-guided learning-augmented framework developed in section 2.3. The central tenet of our approach is to linearize both the objective function and the polynomial constraints by leveraging the oracle's prediction $\hat{\boldsymbol{x}}$. For each constraint polynomial $p_c(\boldsymbol{x})$, we construct a hierarchical linear approximation analogous to that of the objective function. Let $\mathcal{I}_c$ denote the set of all valid index tuples derived from the recursive decomposition of p_c, and let $q_{c,I}(\boldsymbol{x})$ represent the linear approximation of the component $p_{c,I}(\boldsymbol{x})$ centered at $\hat{\boldsymbol{x}}$.

We formulate the linear programming relaxation, denoted as (d-LP'), by enforcing linearization guarantees for both the objective and the constraints. Crucially, we incorporate tolerance bounds δ_I and $\delta_{c,I}$ to account for the deviation between the oracle prediction and the optimal solution.

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & q(\boldsymbol{x}) \\ \text{s.t.} \quad & q_c(\boldsymbol{x}) \in [L_c - \delta_c, U_c + \delta_c] \quad \forall c \in \mathcal{C} \\ & q_I(\boldsymbol{x}) \in p_I(\hat{\boldsymbol{x}}) \pm \delta_I \quad \forall I \in \mathcal{I} \\ & q_{c,I}(\boldsymbol{x}) \in p_{c,I}(\hat{\boldsymbol{x}}) \pm \delta_{c,I} \quad \forall c \in \mathcal{C}, \forall I \in \mathcal{I}_c \\ & \boldsymbol{x} \in [0,1]^n. \end{aligned} \tag{d-LP'}$$

Here, $q(\boldsymbol{x})$ and $q_c(\boldsymbol{x})$ are the top-level linear approximations of $p(\boldsymbol{x})$ and $p_c(\boldsymbol{x})$, respectively. The tolerances δ_I and $\delta_{c,I}$ are determined by the oracle's accuracy $\varepsilon = \|\boldsymbol{x}^* - \hat{\boldsymbol{x}}\|_1$ and the polynomial structure, as defined in section 2.3.1. Specifically, for the top-level constraint approximation, we set $\delta_c = \sum_{I \in \mathcal{I}_c} \delta_{c,I}$.

A fundamental requirement for the validity of this relaxation is that it must admit the optimal integer solution $\boldsymbol{x}^*$. We establish this feasibility in the following lemma.

Lemma E.1. *Let $\boldsymbol{x}^*$ be the optimal solution to (d-IP'). Then, $\boldsymbol{x}^*$ is a feasible solution to the relaxation (d-LP').*

Proof. For clarity of exposition, we elucidate the proof using the quadratic case ($d = 2$) for a single constraint $L \le p_c(\boldsymbol{x}) \le U$. The argument generalizes straightforwardly to higher-degree polynomials. Assume, without loss of generality, that the constant term of $p_c(\boldsymbol{x})$ is

zero. The polynomial admits the decomposition $p_c(\boldsymbol{x}) = \sum_{i\in[n]} x_i \cdot p_{c,i}(\boldsymbol{x})$. The corresponding constraints in the relaxation are:

$$\begin{aligned} L - \delta_c \le q_c(\boldsymbol{x}) \le U + \delta_c, \\ p_{c,i}(\boldsymbol{x}) \in p_{c,i}(\hat{\boldsymbol{x}}) \pm \delta_{c,i}, \end{aligned} \tag{15}$$

where $\delta_{c,i} = \beta\sqrt{n}\sqrt{\varepsilon}$ and $\delta_c = \sum_{i\in[n]} \delta_{c,i}$. Note that for $d = 2$, the components $p_{c,i}(\cdot)$ are linear, so $q_{c,i}(\cdot) \equiv p_{c,i}(\cdot)$.

Substituting $\boldsymbol{x}^*$ into the component constraints, we observe that $p_{c,i}(\boldsymbol{x}^*) \in p_{c,i}(\hat{\boldsymbol{x}}) \pm \delta_{c,i}$ holds by the definition of $\delta_{c,i}$ (cf. lemma 2.7). For the top-level constraint $q_c(\boldsymbol{x}^*)$, we have:

$$\begin{aligned} q_c(\boldsymbol{x}^*) &= \sum_{i\in[n]} x_i^* \cdot q_{c,i}(\hat{\boldsymbol{x}}) \\ &\in \sum_{i\in[n]} x_i^* \cdot [p_{c,i}(\boldsymbol{x}^*) \ \pm \ \delta_{c,i}] \\ &\in p_c(\boldsymbol{x}^*) \ \pm \ \delta_c \subseteq [L - \delta_c, U + \delta_c]. \end{aligned} \tag{16}$$

The final inclusion follows from the original feasibility $p_c(\boldsymbol{x}^*) \in [L, U]$. Thus, $\boldsymbol{x}^*$ satisfies all relaxed constraints. □

We solve (d-LP') to obtain a fractional solution $\boldsymbol{y}$, and subsequently employ independent randomized rounding to recover an integral solution $\boldsymbol{z}$. We formalize the notion of approximate constraint satisfaction as follows.

Definition E.2 ([AKK99])**.** A solution $\boldsymbol{x}$ satisfies a constraint $L \le p_c(\boldsymbol{x}) \le U$ with an additive error δ if $L - \delta \le p_c(\boldsymbol{x}) \le U + \delta$.

The following theorem characterizes the quality of the rounded solution $\boldsymbol{z}$.

Theorem E.3. *The proposed algorithm yields a rounded solution $\boldsymbol{z} \in \{0,1\}^n$ for (d-IP') in polynomial time. With probability at least $1 - 2(|\mathcal{C}|+1)d/n^{k-d+2}$, the objective value satisfies:*

$$p(\boldsymbol{z}) \ge p(\boldsymbol{x}^*) - 2\eta\beta n^{d-1/2}\sqrt{\varepsilon} - \eta\beta n^{d-1}\sqrt{\frac{k+1}{2}}\sqrt{n\ln n},$$

where $\boldsymbol{x}^*$ is the optimal solution of (d-IP'), and $\eta = 2e(d-2)+1$ is a constant respect to d. Furthermore, for each constraint $c \in \mathcal{C}$ of degree $d' \ge 2$, the solution $\boldsymbol{z}$ satisfies the constraint within an additive error of:

$$\Delta = \eta'\beta n^{d'-1/2}\sqrt{\varepsilon} + \eta'\beta n^{d'-1}\sqrt{\frac{k+1}{2}}\sqrt{n\ln n},$$

where $\eta' = 2e(d'-2)+1$ is a constant respect to d'.

Proof. We analyze the objective value and constraint violations by synthesizing the relaxation gap analysis with rounding error bounds. Without loss of generality, we assume throughout this proof that all polynomials are of degree d.

Feasibility and Relaxation Gap. According to lemma E.1, the optimal integer solution $\boldsymbol{x}^*$ is feasible for (d-LP'). Let $\boldsymbol{y}$ denote the optimal fractional solution to (d-LP'). Following the derivation presented in theorem 2.12, the objective value of the relaxation satisfies:

$$p(\boldsymbol{y}) \geq p(\boldsymbol{x}^*) - 2\eta\beta n^{d-1/2}\sqrt{\varepsilon}.$$

Similarly, for each constraint $c \in \mathcal{C}$, since $\boldsymbol{y}$ satisfies $q_c(\boldsymbol{y}) \in [L_c - \delta_c, U_c + \delta_c]$, we can bound the deviation of $p_c(\boldsymbol{y})$ from $q_c(\boldsymbol{y})$. Applying the relaxation gap bound to $p_c(\boldsymbol{y})$ yields:

$$|p_c(\boldsymbol{y}) - q_c(\boldsymbol{y})| \leq \sum_{I \in \mathcal{I}_c} \delta_{c,I} \leq \eta\beta n^{d-1/2}\sqrt{\varepsilon}.$$

Consequently, $p_c(\boldsymbol{y})$ satisfies the constraint within an additive error of $\eta\beta n^{d-1/2}\sqrt{\varepsilon}$.

Rounding Error. Invoking theorem 2.13 for the objective polynomial $p(\boldsymbol{x})$, the following bound holds with probability at least $1 - 2d/n^{k-d+2}$:

$$|p(\boldsymbol{z}) - p(\boldsymbol{y})| \leq \eta\beta n^{d-1}\sqrt{\frac{k+1}{2}}\sqrt{n \ln n}.$$

An analogous probabilistic bound applies to each constraint polynomial $p_c(\boldsymbol{x})$. By applying a union bound over the objective function and all $|\mathcal{C}|$ constraints, we establish that these bounds hold simultaneously with probability at least $1 - 2(|\mathcal{C}| + 1)d/n^{k-d+2}$.

Conclusion. Combining the relaxation gap and the rounding error yields the final performance guarantees. For the objective function, we have:

$$p(\boldsymbol{z}) \geq p(\boldsymbol{y}) - |p(\boldsymbol{z}) - p(\boldsymbol{y})| \geq p(\boldsymbol{x}^*) - 2\eta\beta n^{d-1/2}\sqrt{\varepsilon} - \eta\beta n^{d-1}\sqrt{\frac{k+1}{2}}\sqrt{n \ln n}.$$

Regarding the constraints, the total additive error is given by the sum of the relaxation deviation and the rounding deviation:

$$\Delta = \eta\beta n^{d-1/2}\sqrt{\varepsilon} + \eta\beta n^{d-1}\sqrt{\frac{k+1}{2}}\sqrt{n \ln n}.$$

□

F Application to MAX-k-CSP

In this section, we demonstrate the applicability of our framework to the MAX-k-CSP problem. We show that MAX-k-CSP can be naturally formulated as a smooth integer program, thereby allowing our learning-augmented algorithms to be effectively applied.

It is a well-established result that problems in MAX-SNP can be reduced, via L-reductions, to MAX-k-CSP for some constant k [PY91; Pap94], while preserving the approximation ratio. Although specific graph problems such as MAX-CUT admit natural low-degree polynomial formulations (as illustrated in example 1), the reduction to MAX-k-CSP offers a unified perspective. Therefore, we focus on modeling MAX-k-CSP within our framework.

A standard arithmetization technique allows us to represent the MAX-k-CSP problem as a smooth integer program of degree k. Consider an instance with n decision variables $x_1, \ldots, x_n \in \{0,1\}$ and m constraints. Each constraint C_j (for $j \in [m]$) is defined over a subset of k variables and is specified by a Boolean function $c_j : \{0,1\}^k \to \{0,1\}$.

To formulate the objective function, we encode each constraint as a polynomial. For a specific constraint c on variables $\boldsymbol{x}_S = (x_{i_1}, \ldots, x_{i_k})$, we define its polynomial representation $t_c(\boldsymbol{x}_S)$ as the sum of indicator polynomials for all its satisfying assignments. Specifically, for each assignment $a \in \{0,1\}^k$ such that $c(a) = 1$, the indicator polynomial is:

$$\delta_a(\boldsymbol{x}_S) = \prod_{r=1}^{k} x_{i_r}^{a_r}(1 - x_{i_r})^{1-a_r}.$$

Then, $t_c(\boldsymbol{x}_S) = \sum_{a:c(a)=1} \delta_a(\boldsymbol{x}_S)$. The global objective function $p(\boldsymbol{x})$ is the sum over all m constraints:

$$p(\boldsymbol{x}) = \sum_{j=1}^{m} t_{c_j}(\boldsymbol{x}).$$

This function is a polynomial of degree at most k.

To analyze the smoothness of $p(\boldsymbol{x})$, we introduce a parameter M that captures the "local density" of the constraints. We define M as the maximum number of constraints defined on any single set of k variables. Intuitively, M measures the multiplicity of constraints sharing the same variable scope.

- For MAX-CUT on simple graphs, any pair of vertices has at most one edge, so $M = 1$.
- For MAX-k-SAT you might have multiple clauses involving the same set of variables (e.g., $x_1 \vee x_2 \vee x_3$ and $\neg x_1 \vee x_2 \vee x_3$). Since there are 2^k possible distinct clauses over k variables, if we assume no duplicate identical clauses, $M \leq 2^k$.

We now establish that the objective function $p(\boldsymbol{x})$ is smooth, with the smoothness parameter β depending on M and k.

Theorem F.1. *The polynomial objective $p(\boldsymbol{x})$ derived from a MAX-k-CSP instance is β-smooth for any $\beta \geq M2^k$.*

Proof. Let $p(\boldsymbol{x}) = \sum_{j=1}^{m} t_{c_j}(\boldsymbol{x})$. We analyze the coefficients of the monomials in the expansion of $p(\boldsymbol{x})$ by bounding the contribution from each constraint.

First, observe that for any single constraint c on a set of k variables, the polynomial $t_c(\boldsymbol{x})$ is a sum of at most 2^k terms of the form $\delta_a(\boldsymbol{x})$. The expansion of each $\delta_a(\boldsymbol{x})$ produces monomials with coefficients ± 1. Consequently, the magnitude of the coefficient of any monomial in the expansion of a single constraint $t_c(\boldsymbol{x})$ is bounded by 2^k.

We now bound the coefficients of the global objective $p(\boldsymbol{x})$ by considering two cases based on the degree l of the monomials.

- *Case 1: Monomials of degree $l = k$.* Consider a monomial involving a specific set of k variables, say S. This monomial can only appear in the polynomials $t_{c_j}(\boldsymbol{x})$ corresponding to constraints defined exactly on the variable set S. By the definition of M, there are at most M such constraints. Since the coefficient of the monomial in each such $t_{c_j}(\boldsymbol{x})$ is bounded by 2^k, the magnitude of the coefficient for this monomial in $p(\boldsymbol{x})$ is bounded by $M2^k$. This satisfies the β-smoothness condition $|c_S| \leq \beta n^{k-k} = \beta$ for $\beta \geq M2^k$.

- *Case 2: Monomials of degree $l < k$.* Consider a monomial defined by a set S of l variables. This monomial contributes to the expansion of a constraint defined on a set T of k variables only if $S \subset T$. To form such a superset T, we must select $k-l$ additional variables from the remaining $n - l$ variables. Thus, the number of such supersets T is $\binom{n-l}{k-l}$. For each set T, there are at most M constraints. Using the bound of 2^k for the coefficient from each constraint, the coefficient of the monomial corresponding to S in $p(\boldsymbol{x})$ is bounded by:

$$M2^k \cdot \binom{n-l}{k-l} \leq M2^k \cdot \frac{n^{k-l}}{(k-l)!} \leq M2^k n^{k-l}.$$

  This satisfies the β-smoothness requirement $|c_S| \leq \beta n^{k-l}$ provided that $\beta \geq M2^k$.

Combining both cases, we conclude that $p(\boldsymbol{x})$ is β-smooth for any $\beta \geq M2^k$. □