Parsimonious Learning-Augmented Approximations for Dense Instances of $\mathcal{NP}$-hard Problems

Evripidis Bampis [*1] **Bruno Escoffier** [*1 2] **Michalis Xefteris** [*1]

Abstract

The classical work of (Arora et al., 1999) provides a scheme that gives, for any $\epsilon > 0$, a polynomial time $1 - \epsilon$ approximation algorithm for dense instances of a family of $\mathcal{NP}$-hard problems, such as MAX-CUT and MAX-k-SAT. In this paper we extend and speed up this scheme using a logarithmic number of one-bit predictions. We propose a learning augmented framework which aims at finding fast algorithms which guarantees approximation consistency, smoothness and robustness with respect to the prediction error. We provide such algorithms, which moreover use predictions parsimoniously, for dense instances of various optimization problems.

1. Introduction

In an era marked by the widespread adoption of Machine Learning technology, ML predictors, capable of learning to predict the unknown based on (past) data, are employed to solve numerous problems daily. Due to an effort to exploit this development, there has been a trend in recent years that tries to use ML predictions in order to overcome known worst-case computational limitations. The goal is to provide algorithms that use a possibly erroneous predictor to enhance their performance when the prediction is accurate, while still providing worst case performance guarantees.

The formal framework for these learning-augmented algorithms (or algorithms with predictions) has been presented by Lykouris and Vassilvitskii in their seminal paper (Lykouris & Vassilvitskii, 2021), in which they studied the caching problem. In this framework, no assumption is made about the quality of the predictor and the objective is to design learning-augmented algorithms that are *consistent*, i.e., whose performance is close to the best possible performance that can be achieved when the prediction is perfect, *smooth*, meaning that the quality of the solution produced degrades smoothly with the error made in the prediction, and *robust*. The robustness requires that the performance of the algorithm remains close to the one of the best worst-case algorithm even when the prediction is bad (see Section 2 for formal definitions).

This vein of work has produced various results for online algorithms, i.e., algorithms that are not aware of the whole (future) input of the problem, including scheduling (Mitzenmacher, 2020; Purohit et al., 2018), metrical task systems (Antoniadis et al., 2020), online facility location (Jiang et al., 2022) and online routing problems (Bampis et al., 2022; 2023). More related to our work, in (Lattanzi et al., 2023) they use predictions to speed up the Bellman-Ford algorithm for the shortest path problem. Furthermore, in (Dinitz et al., 2021) they design a faster algorithm for computing matchings utilizing warm-start predicted solutions, and in (Lu et al., 2021; Bai & Coester, 2023) they speed up sorting using predictions. For clustering, Ergund et al. present an algorithm that given a prediction with error rate upper bounded by α achieves an approximation of $1 + O(\alpha)$ in almost optimal running time (Ergun et al., 2022).

All the aforementioned works use the predictor without any limitations. Recently, a new line of work, which uses a small number of predictions to design learning-augmented algorithms, has emerged. Im et al. proposed an algorithm that uses a bounded number of predictions to solve the online caching problem (Im et al., 2022). In (Antoniadis et al., 2023) they solved the paging problem utilizing a minimum amount of predictions. Similar works that present algorithms which take into account the amount of predictions used, penalizing each prediction request by some cost or given a finite budget are the works of (Drygala et al., 2023) and (Benomar & Perchet, 2023), respectively. In this paper, we follow this line of work and use a logarithmic number of one-bit predictions to solve dense instances for a family of problems that includes MAX-CUT and MAX-k-SAT.

Simultaneously and independently of our work, MAXCUT with predictions was studied in two separate papers. Cohen-

[*]Equal contribution [1]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France [2]Institut Universitaire de France, Paris, France. Correspondence to: Michalis Xefteris <michail.xefteris@lip6.fr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Addad et al. studied the approximability of MAXCUT with predictions considering two different models (Cohen-Addad et al., 2024). In their first model, they get a prediction for each vertex that is correct with probability $1/2+\epsilon$ and give a polynomial-time $(0.878 + \tilde{\Omega}(\epsilon^4))$-approximation algorithm. In their second model, they get a correct prediction for each vertex with probability ϵ and design a $(0.858 + \Omega(\epsilon))$-approximation algorithm. Ghoshal et al. (Ghoshal et al., 2024) studied MAXCUT and MAX2-LIN in the two aforementioned models as well (Ghoshal et al., 2024).

Approximation algorithms Approximation algorithms are one standard way of dealing with $\mathcal{NP}$-hard problems as they usually run in polynomial time. An algorithm is an α-approximation for an optimization problem iff for every instance of the problem it can find a solution within a factor of α of the optimum solution. If the problem is a maximization problem, $\alpha \leq 1$ and the approximate solution is at least α times the optimum.

A PTAS (Polynomial Time Approximation Scheme) takes an instance of an optimization problem and a user-defined parameter $\epsilon > 0$ and outputs a solution that is within a factor $1 - \epsilon$ of being optimal (or $1 + \epsilon$ for minimization problems). The running time of a PTAS is required to be polynomial in the problem size for every fixed $\epsilon > 0$, but can be even super-exponential with respect to $1/\epsilon$. However, hardness results have shown that unless $\mathcal{P} = \mathcal{NP}$, problems such as vertex cover, MAX-3-SAT, MAX-CUT and metric TSP do not have a PTAS (Arora et al., 1998; Papadimitriou & Yannakakis, 1988). Moreover, k-DENSEST SUBGRAPH does not admit a PTAS under a complexity assumption (Khot, 2006).

Despite the discouraging results, many approximation algorithms for MAX-$\mathcal{SNP}$ problems[1] like MAX-CUT, MAX-k-SAT have been presented, by exploiting the structure of various classes of instances. One particularly significant line of research is the study of the approximability of dense instances[2] of those problems, which was initiated by Arora, Karger and Karpinski (Arora et al., 1999) and de la Vega (Fernandez de la Vega, 1996). This line of work has produced several results in approximating dense instances of $\mathcal{NP}$-hard problems (Fernandez de la Vega & Karpinski, 2000; Bazgan et al., 2003; Imamura & Iwama, 2005; Cardinal et al., 2012). More specifically, in (Arora et al., 1999) a framework was presented which shows that a family of problems, including MAX-CUT and MAX-k-SAT, admits a PTAS on dense instances. They actually gave additive approximations for the problems, which can be made into a multiplicative $1 - \epsilon$ approximation due to the denseness of each problem. The framework was later extended and generalized to solve almost-sparse instances of the same problems by using subexponential time (Fotakis et al., 2016).

1.1. Our contribution

The first goal of this paper is to utilize the additional power given by a small (logarithmic) number of binary predictions to design a learning-augmented algorithm that significantly improves the running time of the PTAS of (Arora et al., 1999) for dense instances of the following problems[3]:

- **MAX-CUT:** Given an undirected graph $G = (V, E)$, partition the vertices of the graph into two complementary sets so as to maximize the number of edges with exactly one vertex in each set.
- **MAX-DICUT:** The directed version of MAX-CUT. Given a directed graph $G = (V, E)$, find a subset $T \subseteq V$ of vertices to maximize the total number of edges (u, v) with $u \in T$ and $v \in \overline{T}$.
- **MAX-HYPERCUT(d):** A natural generalization of MAX-CUT to hypergraphs of dimension d. In MAX-HYPERCUT an edge is considered cut if it has at least one endpoint on each side.
- **k-DENSEST SUBGRAPH:** Given an undirected graph G, find a subset C of k vertices so that the induced subgraph $G[C]$ has a maximum number of edges.
- **MAX-k-SAT:** Given an instance with n variables that consists of m boolean clauses $f_1, \ldots, f_m$, each clause being a disjunction of at most k literals, we seek a truth assignment to the variables that maximizes the number of satisfied clauses.

We consider in this work a new learning augmented framework (see Section 2 for precise definitions), called Learning Augmented Approximation (LAA), where we want to get approximation ratios close to $1 - \epsilon$, smoothly decreasing with the error made in the predictions, while having a small (polynomial here, with no dependency on ϵ) time complexity. We are particularly focusing on parsimonious predictions, using typically a logarithmic number of prediction bits.

Let us start with MAX-CUT for a smoother exposition (Section 3). For dense MAX-CUT, the PTAS of Arora, Karger and Karpinski (Arora et al., 1999) gives for any user-defined $\epsilon > 0$, an $1 - \epsilon$ randomized approximation that runs in time $n^{O(1/\epsilon^2)}$. Since the work of (Arora et al., 1999), there has been faster PTAS, like the one in (Mathieu & Schudy, 2008),

[1] A formal definition of MAX-$\mathcal{SNP}$ problems is given in (Papadimitriou & Yannakakis, 1988).

[2] For example, a dense instance of MAX-k-SAT is an instance where the number of clauses is $\Omega(n^k)$.

[3] Note that computing an optimal solution for all these problems remains $\mathcal{NP}$-hard even for dense instances (Arora et al., 1999). Moreover, they have no FPTAS unless $\mathcal{P} = \mathcal{NP}$ (deterministic), or $\mathcal{NP} \subseteq \mathcal{BPP}$ (randomized).

but all with an exponential dependence on $1/\epsilon$ in the running time. In this work, we use predictions to improve the running time for MAX-CUT to a low-degree polynomial with no dependency on ϵ while getting an approximation ratio $1-\epsilon-f(\textit{error})$, for a linear function f with respect to the prediction *error* (Theorem 3.3).

More precisely, given $\epsilon > 0$, we sample a set S of $O(\log n/\epsilon^3)$ vertices and get a binary prediction $\hat{a}_i \in \{0,1\}$ for the placement of each vertex i (side of the cut) of the sample at an optimal solution $\mathbf{a} = (a_1, \ldots, a_n)$. The prediction *error* is just the sum of the absolute differences $|\hat{a}_i - a_i|$ of the variables in the sample S. Dealing with the LAA framework, we design an algorithm LAA-CUT which approximates MAX-CUT as follows, where T_{LP} denotes the time to solve an LP with n variables and $O(n)$ constraints.

Theorem 1.1. *Let G a δ-dense graph. Then, for any $\epsilon > 0$ with $|S| = \Theta(\ln n/(\epsilon^3\delta^4))$,* LAA-CUT *runs in time $O(n \cdot T_{LP})$ and, with respect to the approximation ratio, is with high probability $(1-\epsilon)$-consistent, $\left(1-\epsilon-8\frac{\textit{error}}{\delta|S|}\right)$-smooth and 0.878-robust, where* error *is the prediction error.*

Note that the density condition is necessary to achieve a consistency of $1-\epsilon$. Indeed, otherwise, using exhaustive search we would get a PTAS for all MaxCut instances while this problem is APX-hard.

In Section 4, we generalize the approach applied to MAX-CUT. Retracing the steps in (Arora et al., 1999) we express each problem as a maximization problem of a low degree polynomial with bounded coefficients and n binary variables. Then, we recursively decompose the polynomial problem into lower-degree polynomials estimating the coefficients by using predictions on a sample of variables. In the end, we get an integer linear program, for which we obtain a fractional solution in polynomial time. Using randomized rounding we obtain an integer solution for the original problem. The running time of our algorithm with predictions is much shorter than the running time of the PTAS (Theorem 4.2). The algorithm we get can be seen as an additive approximation (depending on the prediction *error*), but translates into a multiplicative one when applied to dense instances of the problems studied (Section 5). As for MaxCut, the density condition is necessary for the result to hold.

Using the same approach as for MAX-CUT, we obtain an algorithm corresponding to the LAA framework (Theorem 4.3), which can be applied to all our problems (and possibly many more). Here again, we emphasize the fact that we use the predictions parsimoniously, which is highly desirable as a predictor is typically a machine learned model that can be computationally expensive.

While we acknowledge that there might not be a readily available oracle setting for the specific problems under study, we propose considering a scenario similar to the pricing policy implemented by OpenAI for ChatGPT. OpenAI charges customers based on the number of tokens (words) in both the input and output[4]. Similarly, one could consider a pricing policy for machine learning models in a private company that tackle computational problems. Employing predictions parsimoniously in that case would result in cost savings, as it would require $O(\log n)$ instead of $O(n)$ tokens.

Let us also note that our work can be easily extended to the multiple predictions setting (Anand et al., 2022). Instead of receiving only one prediction for the values of S, we receive k different predictions from different predictors. Running our algorithm k times (with the same S) and outputting the best solution, we get an approximation with respect to the best predictor, i.e., the one with the lowest prediction *error*. The time overhead is just a multiplicative k.

2. Notation and Preliminaries

We start by giving a definition of density for each problem studied in this work.

Definition 2.1. An undirected graph $G(V,E)$ with n vertices is δ-dense when $\delta = \frac{2|E|}{n(n-1)}$. For a directed graph $G(V,E)$, $\delta = \frac{|E|}{n(n-1)}$. A dimension-$d$ hypergraph is δ-dense if it has at least δn^d edges. Similarly, a k-SAT formula is δ-dense when it has at least δn^k clauses.

In the paper, we assume that δ, d, k are constants. As explained in the introduction, we deal with optimizing polynomials. The following definition will be particularly useful in the next sections.

Definition 2.2. A Polynomial Integer Program (PIP) is of the form

$$\begin{array}{lll} & \max p(x_1,\ldots,x_n) & \\ s.t. & l_i \le p_i(\mathbf{x}) \le u_i & i = 1,\ldots,m \\ & x_i \in \{0,1\} & \forall i \le n, \end{array}$$

where $p, p_1, \ldots, p_n$ are polynomials. The PIP could have minimization instead of maximization. If all p, p_i have degree at most d, we call this program a degree-d PIP.

Let us now define a class of PIPs that are easier to approximate. Note that solving PIPs is $\mathcal{NP}$-hard in general.

Definition 2.3. A degree-d polynomial is c-smooth (or it has smoothness c) if the absolute value of each coefficient of each degree i monomial is at most $c \cdot n^{d-i}$.

A c-smooth degree-d PIP is a PIP in which the objective function and the constraints are c-smooth polynomials with degree at most d.

[4] https://openai.com/pricing

We assume that c, d are constants.

Prediction Model Let $\mathcal{I}$ be an instance of an optimization problem and S a subset of the variables of the problem sampled uniformly at random. Then, we are given predictions on the values of the variables in S at an optimal solution for the instance $\mathcal{I}$ (i.e., a prediction value for each distinct variable of S).

In order to measure the quality of the predictions, we define the prediction *error*.

Definition 2.4. (Prediction *error*) Let $S \subseteq \{1, 2, \ldots, n\}$ be a multiset and fix an optimal solution $\mathbf{a} = (a_1, \ldots, a_n) \in \{0, 1\}^n$ for an optimization problem. Given a prediction $\hat{a_j} \in \{0, 1\}$ for every distinct $a_j, \forall j \in S$ $\big($at most $|S|$ predictions in total$\big)$ we define the prediction *error* as follows

$$\textit{error} = \sum_{j \in S} |\hat{a_j} - a_j| = \sum_{j \in S} \textit{error}_j,$$

where $\textit{error}_j = |\hat{a_j} - a_j|, \forall j \in S$.

The *error* is the absolute error, and $\frac{\textit{error}}{|S|}$ is the relative prediction error. In our algorithms, for ease of explanation we sample S uniformly at random with replacement. Throughout the paper we omit the fact that our predictions may use only a subset of S (distinct elements of S).

Learning-augmented approximation framework. Learning-augmented algorithms have three main properties that we adjust in the context of this work. In the Learning Augmented Approximation (LAA) framework, we say that a (randomized) algorithm is:

- α-*consistent*, if it is an α-approximation with high probability when *error* $= 0$,
- β-*robust*, if it is a β-*approximation* with high probability regardless of the value of *error*, and
- γ-*smooth* for a continuous function $\gamma(\textit{error})$, if it is a $\gamma(\textit{error})$-approximation with high probability.

Note that the smoothness of a polynomial has no connection with the smoothness of a learning-augmented algorithm. In the paper, the distinction between the two notions will be made clear due to the context of each sentence.

Notation In the following, we use $a \pm b$ as a shorthand for the interval $[a - b, a + b]$, for $a, b \geq 0$. Moreover, with $[l, u] \pm a$, where $l < u$ and $a \geq 0$, we denote the interval $[l - a, u + a]$. Finally, we often use $|OPT|$ to denote the value of the optimal solution of an optimization problem.

3. MAX-CUT

In this section, we introduce our approach and apply it to MAX-CUT in a graph $G(V, E)$ that is δ-dense. First, we show how to speed up the PTAS of (Arora et al., 1999) using a limited number of predictions (Section 3.1- 3.5), leading to algorithm LA-PTAS-CUT. Then, we use this algorithm to construct the algorithm LAA-CUT (Section 3.6).

3.1. Overview of Algorithm LA-PTAS-CUT

First, let us write MAX-CUT as follows:

$$\begin{aligned} &\max p(\mathbf{x}) = \sum_{i=1}^{n} x_i \cdot \sum_{j \in N(i)} (1 - x_j) \\ &\text{s.t. } x_i \in \{0, 1\} \, \forall i \end{aligned}$$

where $N(i)$ denotes the set of neighbors of vertex i. The vector $\mathbf{x} \in \{0.1\}^n$ characterizes a cut: $x_i = 1$ (resp. $x_i = 0$) indicates that vertex i is placed on the right (resp. left) side of the cut. The objective function $p(\mathbf{x})$ is an n-variate degree-2 2-smooth polynomial.

The above formulation is a quadratic integer program and cannot be approximated efficiently. Our goal is to turn it into a linear program. For that reason, we set $r_i(\mathbf{x}) = \sum_{j \in N(i)} x_j$ and rewrite the MAX-CUT problem in the following way:

$$\begin{aligned} &\max p(\mathbf{x}) = \sum_{i=1}^{n} x_i \cdot \big(|N(i)| - r_i(\mathbf{x})\big) \qquad (1) \\ &\text{s.t. } x_i \in \{0, 1\} \, \forall i. \end{aligned}$$

We first define the algorithm LA-PTAS-CUT which will approximate these $r_i(\mathbf{x})$'s, that are linear functions, by using sampling on the vertices of G and a prediction on the values of the sample at the optimal solution. The main idea (Arora et al., 1999) is that if we have a good estimation of the value of each $\rho_i = r_i(\mathbf{a})$ at the optimal solution $\mathbf{a}$, then we can approximately solve (1). Let us write the Integer Linear Program using our estimates $\hat{e_i}$ for $r_i(\mathbf{x})$ (which will be obtained using the predictions):

$$\begin{aligned} &\max p(\mathbf{x}) = \sum_{i=1}^{n} x_i \cdot \big(|N(i)| - \hat{e_i}\big) \qquad \text{(IP)} \\ &\text{s.t.} \sum_{j \in N(i)} x_j \geq \hat{e_i} - f(\textit{error}, \epsilon, \delta) \cdot n \quad \forall i \in V \\ &\sum_{j \in N(i)} x_j \leq \hat{e_i} + f(\textit{error}, \epsilon, \delta) \cdot n \quad \forall i \in V \\ &x_i \in \{0, 1\} \quad \forall i. \end{aligned}$$

The estimated values $\hat{e_i}$ and the values $f(\textit{error}, \epsilon, \delta)$ are computed such that the optimal solution $\mathbf{a}$ is a feasible solution to the above (IP). Note that we can replace the

right-hand-side of the first n constraints by $\max\{0, \hat{e_i} - f(error, \epsilon, \delta) \cdot n\}$ (as $\sum_{j \in N(i)} x_j \geq 0$) and the one of next n constraints by $\min\{|N(i)|, \hat{e_i} + f(error, \epsilon, \delta) \cdot n\}$ (as $\sum_{j \in N(i)} x_j \leq |N(i)|$). Let (LP) denote the Linear Programming relaxation of (IP), i.e., setting each $x_i \in [0, 1]$.

Our learning-augmented algorithm LA-PTAS-CUT approximates the optimal solution of MAX-CUT executing the following steps:

- We sample a set of vertices S and get a prediction on the value of each $x_i, \forall i \in S$ at the optimal solution $\mathbf{a} = (a_1, \ldots, a_n)$. Using these values we then estimate the values of $r_i(\mathbf{a}), \forall i \in \{1, \ldots, n\}$ at the optimal solution $\mathbf{a}$ (Section 3.2).
- For each possible value of the integer variable *error*, we perform the following two steps and output the best solution.
  1. We replace each function r_i by the corresponding estimate $\hat{e_i}$ of $r_i(\mathbf{a})$, formulate (IP) and show that an optimal solution for this (IP) is a good approximation for MAX-CUT (Section 3.3).
  2. Then, we find an optimal fractional solution $\mathbf{y}$ to (LP) and obtain an integral solution $\mathbf{z}$ by applying (naive) randomized rounding to $\mathbf{y}$ (Section 3.4).

3.2. Estimating Coefficients via Sampling and Predictions

We take a random sample $S \subseteq V$ of $O(\log n)$ vertices. Assume for now that we know the values a_j at the optimal cut for all sampled vertices j. Using the Sampling lemma for MAX-CUT from (Fotakis et al., 2016)[5] with these values a_j we can compute an estimate $e_i = \sum_{j \in S \cap N(i)} a_j \cdot n/|S|$ of each $\rho_i = r_i(\mathbf{a}) = \sum_{j \in N(i)} a_j$ for every vertex i such that $e_i \approx \rho_i$ with high probability. Let us now state the Sampling lemma for MAX-CUT of (Fotakis et al., 2016) and show how to get the estimates e_i for ρ_i if we know the values a_j's at the optimal solution.

Lemma 3.1. *Sampling lemma (Fotakis et al., 2016) Let $\mathbf{a}$ be a binary vector and $G(V, E)$ be a δ-dense graph. For $\epsilon > 0$, we let $g = \Theta(1/\epsilon^3)$ and S be a multiset of $|S| = g \ln n/\delta$ vertices chosen uniformly at random with replacement from V. For any vertex i, if $e_i = (n/|S|) \sum_{j \in N(i) \cap S} a_j$ and $\rho_i = \sum_{j \in N(i)} a_j$, with probability at least $1 - 2/n^3$,*

$$(1 - \epsilon)e_i - \epsilon\delta n \leq \rho_i \leq (1 + \epsilon)e_i + \epsilon\delta n$$

Using Lemma 3.1 we get that with probability at least $1 - 2/n^3$,

$$\begin{aligned} & e_i - \epsilon \cdot e_i - \epsilon\delta n \leq \rho_i \leq e_i + \epsilon \cdot e_i + \epsilon\delta n \\ \implies & e_i - (\epsilon + \epsilon\delta)n \leq \rho_i \leq e_i + (\epsilon + \epsilon\delta)n \\ \implies & e_i - 2\epsilon n \leq \rho_i \leq e_i + 2\epsilon n, \end{aligned} \tag{2}$$

since $\delta \leq 1$ and assuming wlog that $|e_i| \leq n$ (since $|\rho_i| \leq n$). Taking the union bound over all vertices, we have that (2) holds for all vertices $i \in V$ simultaneously with probability at least $1 - 2/n^2$.

Of course, the problem is that we do not know the values $a_j, \forall j \in S$. In (Arora et al., 1999) they try all possible ($2^{O(\log n)} = n^{O(1)}$, as $|S| = g \ln n/\delta = O(\ln n)$ for fixed ϵ, δ) placements of the vertices in the sample, so they guess all a_j correctly. Here, we get a prediction $\hat{a_j}$ for each a_j. Using these predicted values we compute an estimate $\hat{e_i} = \sum_{j \in S \cap N(i)} \hat{a_j} \cdot n/|S|$ for each ρ_i. It is easy to see that

$$\begin{aligned} & \sum_{j \in S \cap N(i)} \hat{a_j} \in \sum_{j \in S \cap N(i)} a_j \pm error \\ \implies & \frac{n \sum_{j \in S \cap N(i)} \hat{a_j}}{|S|} \in \frac{n \sum_{j \in S \cap N(i)} a_j}{|S|} \pm \frac{n}{|S|} error \\ \implies & \hat{e_i} \in e_i \pm \frac{n}{|S|} error \\ \implies & e_i - \frac{n}{|S|} error \leq \hat{e_i} \leq e_i + \frac{n}{|S|} error \\ \implies & \hat{e_i} - \frac{n}{|S|} error \leq e_i \leq \hat{e_i} + \frac{n}{|S|} error. \end{aligned} \tag{3}$$

Using (2) and(3) we get that for all vertices $i \in V$ with probability at least $1 - 2/n^2$,

$$\hat{e_i} - \left(2\epsilon + \frac{error}{|S|}\right)n \leq \rho_i \leq \hat{e_i} + \left(2\epsilon + \frac{error}{|S|}\right)n. \tag{4}$$

3.3. Formulating the Integer Linear Program

Now we can use the estimates $\hat{e_i}$ for the coefficients ρ_i of the quadratic integer program of MAX-CUT and write the following Integer Linear Program (IP):

$$\max \sum_i x_i \cdot \left(|N(i)| - \hat{e_i}\right) \qquad \text{(IP)}$$

$$\text{s.t.} \sum_{j \in N(i)} x_j \geq \hat{e_i} - \left(2\epsilon + \frac{error}{|S|}\right)n \quad \forall i \in V$$

$$\sum_{j \in N(i)} x_j \leq \hat{e_i} + \left(2\epsilon + \frac{error}{|S|}\right)n \quad \forall i \in V$$

$$x_i \in \{0, 1\} \quad \forall i \in V.$$

Note again that we can replace the right-hand side of the first n constraints by $\max\{0, \hat{e_i} - \left(2\epsilon + \frac{error}{|S|}\right)n\}$ and the one of the next n constraints by $\min\{|N(i)|, \hat{e_i} + \left(2\epsilon + \frac{error}{|S|}\right)n\}$.

[5] We use the sampling lemma of Fotakis et. al. for a more straightforward analysis.

With probability at least $1 - 2/n^2$, the previous Integer Linear Program (IP) is feasible, since the optimal solution **a** satisfies it. The only problem is that we do not know the value of the *error*. To overcome this issue, we try all possible values and guess it. Note that *error* $\leq |S|$, so the runtime overhead is at most $|S| \leq n$ (actually, even $|S| \leq g \ln n/\delta$). From now on we assume that we know the true value of the *error*.

Let **z** be an optimal solution to this (IP). We show that **z** is a good approximation for the optimal solution **a** of MAX-CUT. We have that

$$\begin{aligned}
\sum_{i\in V} z_i \sum_{j\in N(i)} (1-z_j) &= \sum_{i\in V} z_i\big(|N(i)| - \sum_{j\in N(i)} z_j\big) \\
&\geq \sum_{i\in V} z_i\left(|N(i)| - \big(\hat{e}_i + \big(2\epsilon + \frac{error}{|S|}\big)n\big)\right) \\
&\qquad \text{by the constraints of (IP),} \\
&\geq \sum_{i\in V} z_i(|N(i)| - \hat{e}_i) - \left(2\epsilon + \frac{error}{|S|}\right)n^2 \\
&\geq \sum_{i\in V} a_i(|N(i)| - \hat{e}_i) - \left(2\epsilon + \frac{error}{|S|}\right)n^2 \\
&\qquad \text{since } \mathbf{z} \text{ is an integer optimal solution of (IP),} \\
&\geq \sum_{i\in V} a_i\left(|N(i)| - \rho_i - \left(2\epsilon + \frac{error}{|S|}\right)n\right) \\
&\qquad -\left(2\epsilon + \frac{error}{|S|}\right)n^2, \text{ from (4),} \\
&\geq \sum_{i\in V} a_i\big(|N(i)| - \rho_i\big) - 2\left(2\epsilon + \frac{error}{|S|}\right)n^2 \\
&= \mathbf{a} - 2\left(2\epsilon + \frac{error}{|S|}\right)n^2 \\
&= |OPT| - 2\left(2\epsilon + \frac{error}{|S|}\right)n^2. \qquad (5)
\end{aligned}$$

Thus, with probability at least $1 - 2/n^2$, the integer optimal solution of (IP) is close to the optimum of MAX-CUT.

3.4. Randomized Rounding

Now we relax the integrality constraints, allowing $x_i \in [0, 1]$ and get the Linear Programming relaxation of (IP). We can solve (LP) via linear programming and obtain a fractional optimal solution $\mathbf{y} \in [0, 1]^n$. Then, we use randomized rounding to convert the fractional solution to an integral one with approximately the same cut value. To achieve that we will use the following lemma, which is due to Raghavan and Thomson (Raghavan & Thomson, 1985) and Arora et al. (Arora et al., 1999).

Lemma 3.2. *Randomized Rounding (Raghavan & Thomson, 1985; Arora et al., 1999) If c and f are positive integers and $0 < \epsilon < 1$, then the following is true for any integers $n > 0$. Let $\boldsymbol{y} = (y_i)$ be a vector of n variables, $0 \leq y_i \leq 1$, that satisfies a certain linear constraint $\boldsymbol{a}^T\boldsymbol{y} = b$, where each $|a_i| \leq c$. Construct a vector $\boldsymbol{z} = (z_i)$ randomly by setting $z_i = 1$ with probability y_i and 0 with probability $1 - y_i$. Then, with probability at least $1 - n^{-f}$, we have that*

$$\boldsymbol{a}^T\boldsymbol{z} \in b \pm c\sqrt{fn \ln n}.$$

Since each $r_i(\mathbf{x})$ is a linear function with $0/1$ coefficients, it follows from the Randomized Rounding lemma and the union bound that with probability at least $1 - n^{-f+1}$ holds (for every vertex simultaneously) that

$$r_i(\mathbf{z}) \in r_i(\mathbf{y}) \pm O(\sqrt{n} \ln n) \qquad \forall i \in V. \qquad (6)$$

Additionally, since each $|N(i)| - r_i(\mathbf{y})$ is at most n, we can use again the Randomized Rounding lemma to get that with probability at least $1 - n^{-f}$

$$\begin{aligned}
\sum_{i\in V} z_i\big(|N(i)| - r_i(\mathbf{y})\big) \in &\sum_{i\in V} y_i\big(|N(i)| - r_i(\mathbf{y})\big) \\
&\pm O(n^{3/2} \ln n). \qquad (7)
\end{aligned}$$

Both inequalities hold simultaneously with probability at least $1 - n^{-f+1} - n^{-f} \approx 1 - n^{-f+1}$.

So, when (5), (6) and (7) hold we get:

$$\begin{aligned}
\sum_{i\in V} z_i\big(|N(i)| - \sum_{j\in N(i)} z_j\big) &= \sum_{i\in V} z_i\big(|N(i)| - r_i(\mathbf{z})\big) \\
&\geq \sum_{i\in V} z_i\big(|N(i)| - r_i(\mathbf{y}) - O(\sqrt{n}\ln n)\big), \text{ from (6),} \\
&\geq \sum_{i\in V} z_i\big(|N(i)| - r_i(\mathbf{y})\big) - O(n^{3/2}\ln n) \\
&\geq \sum_{i\in V} y_i\big(|N(i)| - r_i(\mathbf{y})\big) - O(n^{3/2}\ln n) \text{ from (7),} \\
&\geq |OPT| - 2\left(2\epsilon + \frac{error}{|S|}\right)n^2 - o(1)n^2.
\end{aligned}$$

The last inequality is due to (5) and the fact that the fractional optimal solution **y** cannot be worse than the integer optimal solution **z**.

Finally, all estimations are good with probability at least $1 - 2/n^2$ and the randomized rounding works with probability at least $\approx 1 - 1/n^{-f+1}$. Thus, our learning augmented approximation scheme works with probability $\approx 1 - 2/n^2 - 1/n^{f-1}$.

3.5. Analysis of LA-PTAS-CUT

To conclude, we state and prove the formal theorem for LA-PTAS-CUT.

Theorem 3.3. *Let G a δ-dense graph. Then, for any $\epsilon > 0$ with $|S| = \Theta(\ln n/(\epsilon^3\delta^4))$,* LA-PTAS-CUT *runs in time $O(n \cdot T_{LP})$ and is an $\left(1 - \epsilon - 8\frac{error}{\delta|S|}\right)$-approximation for* MAX-CUT *with probability at least $1 - 3/n^2$, where* error *is the prediction error.*

Proof. For any $\epsilon > 0$, using LA-PTAS-CUT with $f = 3$, $\epsilon' = \epsilon\delta/16, g = 1/\epsilon'^3$ and sample size $|S| = g \ln n/\delta = \Theta\left(\ln n/\epsilon^3\delta^4\right)$, we get a cut $\mathbf{z}$ that with probability at least $\approx 1 - 3/n^2$ satisfies:

$$p(\mathbf{z}) \geq |OPT| - 2\left(2\epsilon' + \frac{error}{|S|}\right)n^2$$
$$= |OPT| - \left(16\epsilon'/\delta + \frac{8error}{\delta|S|}\right)\delta n^2/2$$
$$= |OPT| - \left(\epsilon + \frac{8error}{\delta|S|}\right)\delta n^2/4$$
$$\geq \left(1 - \epsilon - 8\frac{error}{\delta|S|}\right) \cdot |OPT|,$$

as $|OPT|$ is at least $|E|/2 = \delta n(n-1)/2 \geq \delta n^2/4$, $\forall n \geq 2$.

Therefore, the approximation ratio of the algorithm is $\left(1 - \epsilon - 8\frac{error}{\delta|S|}\right)$ with high probability.

Regarding its running time, it is easy to see that it only requires to solve a linear program with n variables and $O(n)$ constraints for each possible value of the *error* (i.e., $|S| \leq n$ rounds). So, it runs in time $O(n \cdot T_{LP})$, where T_{LP} is the time to solve an LP with n variables and $O(n)$ constraints. There are many algorithms that solve linear programs (Vaidya, 1989; Cohen et al., 2019; Lee & Sidford, 2015). The state of the art does it in time $O^*(n^w)$, where w is the matrix multiplication exponent (the current value is $w \approx 2.38$) (Jiang et al., 2020). □

3.6. LAA Framework

We now describe the algorithm LAA-CUT that combines LA-PTAS-CUT (for consistency and smoothness) and a known polytime constant-approximation algorithm (for robustness).

Consistency & Smoothness. For any $\epsilon > 0$, using LA-PTAS-CUT with predictions on a sample of size $|S| = \Theta\left(\ln n/\epsilon^3\delta^4\right)$ we have an algorithm with approximation ratio of $\left(1 - \epsilon - 8\frac{error}{\delta|S|}\right)$ (Theorem 3.3). The algorithm is randomized and gives with probability at least $1 - 3/n^2$ the aforementioned approximation ratio that depends on ϵ, which is user-defined, the density of the graph δ and the $error$ of our prediction (consistency and smoothness of the approximation ratio). The algorithm runs in $O(n \cdot T_{LP})$. Note that the original algorithm in (Arora et al., 1999)[6] runs in time dominated by the exhaustive search which takes time $O(2^{1/(\epsilon\delta)^2 \log n}) = n^{1/(\epsilon\delta)^2}$, which depends on ϵ, δ. For example, for $\epsilon = 1 - 0.878 = 0.122$ (to obtain the approximation ratio of the algorithm by Goemans and Williamson (Goemans & Williamson, 1995)) the running time is $O(n^{67})$!

Additionally, we would like to mention that if the (partial) prediction corresponds to a (global) prediction with approximation ratio α, then our algorithm has approximation ratio at least $\alpha - \epsilon$ with high probability. This is a direct consequence of our proof. The relative (partial) error is an unbiased estimator of the relative (global) error.

Robustness. In case the *error* of our predictions is too large, we would like to be able to ensure an approximation guarantee for the value of the cut (robustness of the approximation ratio). We can do that by running in parallel the celebrated algorithm of Goemans and Williamson (Goemans & Williamson, 1995) which achieves an approximation ratio of ≈ 0.878. The algorithm can run in time $\tilde{O}(n^2)$ using the Arora-Kale algorithm (Arora & Kale, 2007; Trevisan, 2012). Of course, the algorithm is randomized and we should demand that both algorithms (LA-PTAS-CUT and that of Goemans and Williamson) succeed simultaneously. Note that the algorithm can also be derandomized (Mahajan & Hariharan, 1995).

Therefore, the approximation ratio of our learning-augmented scheme is $\max\{1 - \epsilon - 8\frac{error}{\delta|S|}, 0.878\}$ (with probability at least $1 - 3/n^2$) for a δ-dense graph and runs in time $O(n \cdot T_{LP})$. Note that for different values of the parameter $\epsilon > 0$, the prediction *error* is not the same due to the change of the sampling size. Consequently, we restate the theorem for LAA-CUT.

Theorem 1.1. *Let G a δ-dense graph. Then, for any $\epsilon > 0$ with $|S| = \Theta(\ln n/(\epsilon^3\delta^4))$,* LAA-CUT *runs in time $O(n \cdot T_{LP})$ and, with respect to the approximation ratio, is with high probability $(1-\epsilon)$-consistent, $\left(1 - \epsilon - 8\frac{error}{\delta|S|}\right)$-smooth and 0.878-robust, where* error *is the prediction error.*

Proof. The proof becomes now trivial. Here, we just clarify the success probability of the algorithm. The algorithm of Goemans and Williamson succeeds with probability at least τ, for some constant $\tau > 0$. LA-PTAS-CUT as well, so we just have to run both algorithms a constant number of times independently to get a success with probability at least $1 - \eta$, for arbitrarily small η. We can also boost the success probability to $1 - 1/\Omega(n)$ by running the algorithm a logarithmic number of times independently. □

[6] Even the best PTAS for MAX-CUT runs in time that depends exponentially on $1/\epsilon$.

4. Smooth Polynomial Integer Programs

In this section, we extend the approach applied to MAX-CUT to approximately optimize c-smooth polynomials of degree d over all binary vectors $\mathbf{x} \in \{0,1\}^n$, as done in (Arora et al., 1999). We exploit the fact that smooth polynomial integer programs can be recursively decomposed into lower degree PIPs to eventually obtain a linear program. We can assume wlog that instead of solving the optimization problem, we can deal with the feasibility problem of a smooth polynomial integer program, i.e., given a feasible PIP find an integer solution that is approximately feasible. Our general learning-augmented algorithm LA-PTAS follows the same steps as the version for MAX-CUT generalizing our approach[7]. Next, we will use LA-PTAS to get our algorithm for PIPs.

We only briefly sketch here the general approach to build LA-PTAS. Details of the construction and proofs are deferred to the appendix. As shown in (Arora et al., 1999), each absolute value of a c-smooth polynomial of degree d is bounded by $2cen^d$ (where $\ln e = 1$). Thus, the optimal value of a PIP is not too large and we can reduce the optimization of a PIP $p(\mathbf{x})$ to the feasibility version of the problem using binary search. Specifically, it is sufficient to find if the problem $p(\mathbf{x}) \geq M$ for $M > 0$ has a feasible solution. The parameter $M > 0$ can be computed by using binary search in $(0, 2cen^d]$ taking at most $O(\log(2cen^d)) = O(\log(cn^d))$ runs of the algorithm. Throughout the section we denote by N the set $N = \{1, \ldots, n\}$.

Furthermore, another key idea to generalize our results is the following decomposition lemma of a degree-d c-smooth polynomial.

Lemma 4.1. *(Arora et al., 1999) A c-smooth polynomial p of degree d on $\boldsymbol{x} = (x_1, \ldots, x_n)$ can be written uniquely as*

$$p(\boldsymbol{x}) = t + \sum_i x_i p_i(x_i, \ldots, x_n)$$

where t is a constant and each p_i is a c-smooth polynomial of degree $d-1$ and depends only on variables with index i or greater.

Assume that we would like to optimize $p(\mathbf{x})$ of degree d. We transform the optimization problem into a feasibility one using binary search as discussed previously. We now have the feasibility problem $p(\mathbf{x}) \geq M$ for a known $M > 0$. Using the decomposition lemma we can write $p(\mathbf{x})$ as $p(\mathbf{x}) = t + \sum x_i p_i(\mathbf{x})$. Computing an estimate $\hat{e_i}$ of the value of $p_i(\mathbf{a})$ at the optimal solution $\mathbf{a}$, we replace the degree d constraint $p(\mathbf{x}) \geq M$ with $t + \sum x_i \hat{e_i} \geq M$ and a family of constraints on the values $p_i(\mathbf{x})$. Then, we recursively expand these degree $d-1$ constraints, continuing until all constraints are linear. We can compute the estimations $\hat{e_i}$ of $p_i(\mathbf{a})$ by writing $p_i(\mathbf{x}) = \sum x_j p_{ij}(\mathbf{x})$. We then recursively estimate the values $p_{ij}(\mathbf{a})$, and use sampling and the predicted values $\hat{a_k}, k \in S$ to estimate p based on the values of p_{ij}. Thus we end up with an Integer Linear Program (see Appendix A.1).

Then, we relax the integral constraints and solve the Linear Programming relaxation of (d-IP). Finally, we use randomized rounding to get an integral solution. The details and proofs of each step can be found in Appendices A.2 (Estimating Polynomials via Sampling and Predictions), A.3 (Transforming degree d constraints into linear constraints) and A.4 (Randomized Rounding for Smooth Polynomials). We finally obtain the following result (see Appendix A.5 for the proof). In the following, T'_{LP} denotes the time to solve an LP with n variables and $poly(n)$ constraints.

Theorem 4.2. *Given a feasible c-smooth degree-d PIP with n variables, its objective function p and $m = poly(n)$ constraints,* LA-PTAS *finds a binary solution $\boldsymbol{z}$ with probability at least $1/2$ such that*

$$p(z_1, \ldots, z_n) \geq |OPT| - \left(\epsilon + 4ced(d-1)\frac{error}{|S|}\right) n^d$$

given predictions on the values of the distinct variables of S at the optimal solution (optimum of PIP) that is chosen uniformly at random (with replacement), where $|S| = \Theta(\frac{c^4 f d^7}{\epsilon^3} \ln n)$, where $f > 0$ is such that $n^f = \Theta(m \cdot n^d)$. The running time of the algorithm is $O\big(n \ln(cn^d) \cdot T'_{LP}\big)$.

LAA Framework Let us now describe our algorithm LAA-GENERAL that relies on LA-PTAS, whose ratio depends on the *error* while guaranteeing a fixed (polynomial) running time.

Consistency & Smoothness. For any $\epsilon > 0$, we use LA-PTAS with $|S| = \Theta(\frac{c^4 f d^7}{\epsilon^3} \ln n)$ and get an algorithm that with high probability outputs a value of at least $|OPT| - \big(\epsilon + 4ced(d-1)\frac{error}{|S|}\big) n^d$ (Theorem 4.2). As we will see in Section 5, this additive approximation leads to a multiplicative one when the instance is dense for the problems we consider. Therefore, we achieve the desired consistency and smoothness in the approximation ratio.The running time of LA-PTAS is $O\big(n \ln(cn^d) \cdot T'_{LP}\big)$, with no dependency on ϵ (compared to exponential dependency in $1/\epsilon$ in the PTAS's). Furthermore, note again that our algorithm is guaranteed to have approximation ratio at least as good as the approximation quality of the (global) prediction (minus ϵ).

Robustness. When the *error* of our predictions is too large, we can ensure an approximation guarantee for the solution value (robustness of the approximation ratio). We can achieve that by running in parallel a constant approximation

[7] In this section, we focus on maximization problems. The minimization version can be handled similarly.

algorithm for the given problem (if it exists in the literature). The running time, here, depends on the approximation algorithm that is used for the problem in question. If the algorithm is randomized, we take the joint probability that both algorithms succeed simultaneously.

Theorem 4.3. *We are given a feasible c-smooth degree-d PIP with n variables, its objective function p and $m = poly(n)$ constraints. Let also ALG be an algorithm for the PIP that runs in time T_{ALG} and produces a solution with cost at least $\alpha|OPT|$. For any $\epsilon > 0$ with $|S| = \Theta\big(\frac{c^4 f d^7}{\epsilon^3} \ln n\big)$, where $f > 0$ is such that $n^f = \Theta(mn^d)$,* LAA-GENERAL *runs in time* $\max\{O\big(n \ln(cn^d) \cdot T'_{LP}\big), T_{ALG}\}$ *and outputs with high probability a solution with cost at least*

$$\max\left\{|OPT| - \left(\epsilon + 4ced(d-1)\frac{error}{|S|}\right)n^d, \alpha|OPT|\right\},$$

where error *is the prediction error.*

5. Applications

In this section, we explain how to apply the algorithm of Section 4. We give the example of MAX-k-SAT, while problems MAX-DICUT, MAX-HYPERCUT(d) and k-DENSEST SUBGRAPH are deferred to Appendix B. Note also that the algorithm can be applied to the more general MAX-k-CSP, as shown in (Arora et al., 1999).

MAX-k-SAT A standard arithmetization technique (see (Arora et al., 1999)) can be used to reduce any instance of MAX-k-SAT with n variables to solving a degree-k polynomial $p(\mathbf{x})$ such that the optimal truth assignment for MAX-k-SAT corresponds to an $\mathbf{a} \in \{0,1\}^n$ that maximizes $p(\mathbf{x})$. Moreover, the value of the optimal MAX-k-SAT solution is equal to $p(\mathbf{a})$.

Let us now assume that the number of clauses is $m \geq \delta n^k$. The number of clauses of size exactly k is $m - O(n^{k-1})$ and a random assignment satisfies each one of them with probability $1 - 2^{-k}$. Thus it follows that the maximum number of clauses that can be made true is

$$|OPT| \geq (1 - 2^{-k})(m - O(n^{k-1})).$$

Therefore, we use LA-PTAS with $\epsilon' = O(\epsilon/2^k)$ and get the desired accuracy for MAX-k-SAT. For robustness, we can use the poly-time randomized 0.797 approximation (Avidor et al., 2006) to robustify, as done by LAA-GENERAL.

Acknowledgement

This work was partially funded by the grant ANR-19-CE48-0016 from the French National Research Agency (ANR).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning by further exploring its interactions with Theoretical Computer Science. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alon, N., de la Vega, W., Kannan, R., and Karpinski, M. Random sampling and approximation of max-csps. *Journal of Computer and System Sciences*, 67(2):212–243, 2003. Special Issue on STOC 2002.

Anand, K., Ge, R., Kumar, A., and Panigrahi, D. Online algorithms with multiple predictions. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 582–598. PMLR, 17–23 Jul 2022.

Andersson, G. and Engebretsen, L. Better approximation algorithms for set splitting and not-all-equal sat. *Information Processing Letters*, 65(6):305–311, 1998.

Antoniadis, A., Coester, C., Elias, M., Polak, A., and Simon, B. Online metric algorithms with untrusted predictions. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 345–355. PMLR, 13–18 Jul 2020.

Antoniadis, A., Boyar, J., Elias, M., Favrholdt, L. M., Hoeksma, R., Larsen, K. S., Polak, A., and Simon, B. Paging with succinct predictions. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 952–968. PMLR, 2023.

Arora, S. and Kale, S. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pp. 227–236, New York, NY, USA, 2007. Association for Computing Machinery.

Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, may 1998.

Arora, S., Karger, D., and Karpinski, M. Polynomial time approximation schemes for dense instances of NP-hard problems. *Journal of Computer and System Sciences*, 58 (1):193–210, 1999.

Asahiro, Y., Iwama, K., Tamaki, H., and Tokuyama, T. Greedily finding a dense subgraph. In Karlsson, R. and Lingas, A. (eds.), *Algorithm Theory - SWAT 1996, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 136–148. Springer Verlag, 1996.

Avidor, A., Berkovitch, I., and Zwick, U. Improved approximation algorithms for max nae-sat and max sat. In Erlebach, T. and Persinao, G. (eds.), *Approximation and Online Algorithms*, pp. 27–40, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Bai, X. and Coester, C. Sorting with predictions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Bampis, E., Escoffier, B., and Xefteris, M. Canadian traveller problem with predictions. In Chalermsook, P. and Laekhanukit, B. (eds.), *Approximation and Online Algorithms*, pp. 116–133, Cham, 2022. Springer International Publishing.

Bampis, E., Escoffier, B., Gouleakis, T., Hahn, N., Lakis, K., Shahkarami, G., and Xefteris, M. Learning-Augmented Online TSP on Rings, Trees, Flowers and (Almost) Everywhere Else. In Gørtz, I. L., Farach-Colton, M., Puglisi, S. J., and Herman, G. (eds.), *31st Annual European Symposium on Algorithms (ESA 2023)*, volume 274 of *LIPIcs*, pp. 12:1–12:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Bazgan, C., Fernandez de La Vega, W., and Karpinski, M. Polynomial time approximation schemes for dense instances of minimum constraint satisfaction. *Random Structures and Algorithms*, 23(1), 2003.

Benomar, Z. and Perchet, V. Advice querying under budget constraint for online algorithms. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Cardinal, J., Karpinski, M., Schmied, R., and Viehmann, C. Approximating vertex cover in dense hypergraphs. *Journal of Discrete Algorithms*, 13:67–77, 2012. Best Papers from the 3rd International Conference on Similarity Search and Applications (SISAP 2010).

Cohen, M. B., Lee, Y. T., and Song, Z. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pp. 938–942, New York, NY, USA, 2019. Association for Computing Machinery.

Cohen-Addad, V., d'Orsi, T., Gupta, A., Lee, E., and Panigrahi, D. Max-cut with ϵ-accurate predictions. *CoRR*, abs/2402.18263, 2024.

Dinitz, M., Im, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Faster matchings via learned duals. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021.

Drygala, M., Nagarajan, S. G., and Svensson, O. Online algorithms with costly predictions. In Ruiz, F., Dy, J., and van de Meent, J.-W. (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 8078–8101. PMLR, 25–27 Apr 2023.

Ergun, J. C., Feng, Z., Silwal, S., Woodruff, D., and Zhou, S. Learning-augmented k-means clustering. In *International Conference on Learning Representations*, 2022.

Feige, U. and Goemans, M. Approximating the value of two power proof systems, with applications to max 2sat and max dicut. In *Proceedings Third Israel Symposium on the Theory of Computing and Systems*, pp. 182–189, 1995.

Feige, U. and Langberg, M. Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41(2):174–211, 2001.

Fernandez de la Vega, W. Max-cut has a randomized approximation scheme in dense graphs. *Random Structures & Algorithms*, 8(3):187–198, 1996.

Fernandez de la Vega, W. and Karpinski, M. Polynomial time approximation of dense weighted instances of max-cut. *Random Structures & Algorithms*, 16(4):314–332, 2000.

Fotakis, D., Lampis, M., and Paschos, V. T. Sub-exponential Approximation Schemes for CSPs: From Dense to Almost Sparse. In Ollinger, N. and Vollmer, H. (eds.), *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, volume 47 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 37:1–37:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Ghoshal, S., Makarychev, K., and Makarychev, Y. Constraint satisfaction problems with advice. *CoRR*, abs/2403.02212, 2024.

Goemans, M. X. and Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42 (6):1115–1145, nov 1995.

Im, S., Kumar, R., Petety, A., and Purohit, M. Parsimonious learning-augmented caching. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9588–9601. PMLR, 17–23 Jul 2022.

Imamura, T. and Iwama, K. Approximating vertex cover on dense graphs. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pp. 582–589, USA, 2005. Society for Industrial and Applied Mathematics.

Jiang, S., Song, Z., Weinstein, O., and Zhang, H. Faster dynamic matrix inverse for faster lps. *CoRR*, abs/2004.07470, 2020.

Jiang, S. H.-C., Liu, E., Lyu, Y., Tang, Z. G., and Zhang, Y. Online facility location with predictions. In *International Conference on Learning Representations*, 2022.

Khot, S. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

Lattanzi, S., Svensson, O., and Vassilvitskii, S. Speeding up Bellman ford via minimum violation permutations. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18584–18598. PMLR, 23–29 Jul 2023.

Lee, Y. T. and Sidford, A. Efficient inverse maintenance and faster algorithms for linear programming. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 230–249, 2015.

Lu, P., Ren, X., Sun, E., and Zhang, Y. Generalized sorting with predictions. In *Symposium on Simplicity in Algorithms (SOSA)*, pp. 111–117, 2021.

Lykouris, T. and Vassilvitskii, S. Competitive caching with machine learned advice. *J. ACM*, 68(4), jul 2021.

Mahajan, S. and Hariharan, R. Derandomizing semidefinite programming based approximation algorithms. *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pp. 162–169, 1995.

Mathieu, C. and Schudy, W. Yet another algorithm for dense max cut: go greedy. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pp. 176–182, USA, 2008. Society for Industrial and Applied Mathematics.

Mitzenmacher, M. Scheduling with Predictions and the Price of Misprediction. In Vidick, T. (ed.), *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 14:1–14:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Papadimitriou, C. and Yannakakis, M. Optimization, approximation, and complexity classes. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pp. 229–234, New York, NY, USA, 1988. Association for Computing Machinery.

Purohit, M., Svitkina, Z., and Kumar, R. Improving online algorithms via ml predictions. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Raghavan, P. and Thomson, C. D. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1985.

Trevisan, L. Max cut and the smallest eigenvalue. *SIAM Journal on Computing*, 41(6):1769–1786, 2012.

Vaidya, P. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science*, pp. 332–337, 1989.

A. Missing material of Section 4

A.1. Integer Linear Program

$$\sum_{j\in N} x_j \hat{e_j} \geq M \qquad (d\text{-IP})$$

$$t_{i_1} + \sum_{j\in N} x_j \hat{e}_{i_1 j} \in \hat{e}_{i_1} \pm f(error,\epsilon,\delta)n^{d-1} \qquad \forall i_1 \in N$$

$$t_{i_1 i_2} + \sum_{j\in N} x_j \hat{e}_{i_1 i_2 j} \in \hat{e}_{i_1 i_2} \pm f(error,\epsilon,\delta)n^{d-2}$$

$$\forall (i_1,i_2) \in N \times N$$

$$\cdots$$

$$t_{i_1\ldots i_{d-\ell}} + \sum_{j\in N} x_j \hat{e}_{i_1\ldots i_{d-\ell} j} \in \hat{e}_{i_1\ldots i_{d-\ell}} \pm f(error,\epsilon,\delta)n^{d-\ell}$$

$$\forall (i_1,\ldots,i_{d-\ell}) \in N^{d-\ell}$$

$$\cdots$$

$$t_{i_1\ldots i_{d-1}} + \sum_{j\in N} x_j \hat{e}_{i_1\ldots i_{d-1} j} \in \hat{e}_{i_1\ldots i_{d-1}} \pm f(error,\epsilon,\delta)n$$

$$\forall (i_1,\ldots,i_{d-1}) \in N^{d-1}$$

$$x_j \in \{0,1\} \qquad \forall j \in N.$$

A.2. Estimating Polynomials via Sampling and Predictions

We show how to estimate the coefficients $p_i(\mathbf{a})$ at an optimal solution that are smooth polynomials of degree at most $d-1$ using sampling and predictions. This step is required to be able to replace the constraint on $p(\mathbf{x})$ by linear constraints. We describe an algorithm EVALUATE, adaptation of the one in (Arora et al., 1999), which can approximate the value of a c-smooth degree-d polynomial $p(x_1,\ldots,x_n)$ on any unknown binary vector $\mathbf{a} = (a_1,\ldots,a_n)$ given a partial prediction about $\mathbf{a}$. So, we take a sample S from $\{1,\ldots,n\}$, uniformly at random and with replacement. The sample size is $O(\log n)$. Next, we get a predicted value $\hat{a_j}$ for each value $a_j, \forall j \in S$ at the optimal solution $\mathbf{a}$ (note that the number of predicted values is at most $|S|$). Showing that ESTIMATE (Algorithm 1) provides us with good estimates for the coefficients $p_i(\mathbf{a})$ becomes easier using the following General Sampling lemma of (Fotakis et al., 2016).

Lemma A.1. *General Sampling lemma (Fotakis et al., 2016) Let $\boldsymbol{a}$ in$\{0,1\}^n$ and let $(\rho_j)_{j\in N}$ be any sequence such that for some integer $d \geq 0$ and some constant $\beta \geq 1$, $|\rho_j| \leq (d+1)\beta n^d$, $\forall j \in N$. For all integers $f \geq 1$ and $\epsilon > 0$, let $g = \Theta(fd\beta/\epsilon^3)$ and S a multiset of $|S| = g\ln n$ indices chosen uniformly at random with replacement from N. If $(n/|S|)\sum_{j\in S}\rho_j a_j$, $\rho = \sum_{j\in N}\rho_j a_j$ and $\bar{\rho} = \sum_{j\in N}|\rho_j|$, with probability at least $1 - 4/n^{f+1}$,*

$$\rho - \epsilon\bar{\rho} - \epsilon n^{d+1} \leq (n/|S|)\sum_{j\in S}\rho_j a_j \leq \rho + \epsilon\bar{\rho} + \epsilon n^{d+1}.$$

Algorithm 1 EVALUATE$(p, S, \{\hat{a_i} : i \in S\})$

Require: polynomial p of degree at most d,
set of variables indices S,
predictions $\hat{a_i}$ for $i \in S$.
Ensure: Estimate for $p(a_1,\ldots,a_n)$.
if deg$(p) = 0$ **then**
return p
else
$p(x_1,\ldots,x_n) = t + \sum x_i p_i(x_1,\ldots,x_n)$
for $i \in S$ **do**
$\hat{e_i} \leftarrow$ EVALUATE$(p_i, S, \{\hat{a_i} : i \in S\})$
end for
return $t + (n/|S|)\sum_{i\in S}\hat{a_i}\hat{e_i}$
end if

Let us now show the following lemma about EVALUATE with set S and the predictions.

Lemma A.2. *Let p be a c-smooth degree-d polynomial in n variables x_i and $\boldsymbol{a} = (a_1,\ldots,a_n) \in \{0,1\}^n$. Let $f \geq 1$ be an integer, $\epsilon > 0$, $\beta \geq \max\{1, 2ce\}$, and S be a set of $O(g\ln n)$ indices chosen randomly and with replacement with $g = \Theta(fd\beta/\epsilon^3)$. Also, let $\hat{a_j}, \forall j \in S$ be a prediction on the values of $a_j, \forall j \in S$. Then, with probability at least $1 - 4/n^{f+1-d}$, set S is such that* EVALUATE$(p, S, \{\hat{a_i} : i \in S\})$ *returns a value in*

$$p(a_1,\ldots,a_n) \pm \left((2ce+1)d\epsilon + 2ced\frac{error}{|S|}\right)n^d. \qquad (8)$$

Proof. The proof is by induction on the degree d. For the case $d = 0$ we have by definition that $error = 0$ and EVALUATE returns a value that is exactly $p(\mathbf{a}) = t$.

For the inductive step let $\rho_i = p_i(a_1,\ldots,a_n)$. So,

$$p(\mathbf{a}) = t + \sum_{i=1}^{n} a_i\rho_i.$$

Note that each p_i has degree at most $d-1$.

First, we apply the General Sampling lemma (Lemma A.1) for $p(\mathbf{a})$ with $d' = d-1$, $\beta \geq \max\{1, 2ce\}$, $g = \Theta(fd\beta/\epsilon^3)$ and $|S| = g\ln n$. As each p_i is a c-smooth degree-$(d-1)$ polynomial we have that $|\rho_i| \leq 2cen^{d-1} \leq (d'+1)\beta n^{d'}$. Then, we have with probability at least $1 - 4/n^{f+1}$ that

$$(n/|S|)\sum_{i\in S} a_i\rho_i \in \sum_{i\in N} a_i\rho_i \pm (\epsilon\sum_{i\in N}|\rho_i| + \epsilon n^d).$$

Using again $|\rho_i| \leq 2cen^{d-1}$ we get:

$$(n/|S|)\sum_{i\in S} a_i\rho_i \in \sum_{i\in N} a_i\rho_i \pm \epsilon(2ce+1)n^d. \qquad (9)$$

Given the predictions ($\hat{a_j} \in \{0,1\}, \forall j \in S$ we have that $error_j = |\hat{a_j} - a_j|, \forall j \in S$. Note that $error = \sum_{i \in S} error_i$. Using $\hat{a_j} \in a_j \pm error_j, \forall j \in S$, we have that

$$\begin{aligned}\frac{n}{|S|}\sum_{i\in S}\hat{a}_i\rho_i &\in \frac{n}{|S|}\sum_{i\in S}a_i\rho_i \pm \frac{n}{|S|}\sum_{i\in S}error_i\rho_i \\ &\subseteq \frac{n}{|S|}\sum_{i\in S}a_i\rho_i \pm \frac{n}{|S|}2ce n^{d-1}\sum_{i\in S}error_i \\ &= \frac{n}{|S|}\sum_{i\in S}a_i\rho_i \pm \frac{2ce}{|S|}error\cdot n^d. \end{aligned} \tag{10}$$

Let $\hat{e_i} = \textsc{Evaluate}(p_i, S, \{a_i : \hat{i} \in S)$. By the inductive hypothesis, Evaluate outputs estimates $\hat{e_i}$ such that

$$\rho_i \in \hat{e_i} \pm \left((2ce+1)(d-1)\epsilon n^{d-1} + 2ce(d-1)\frac{error}{|S|}n^{d-1}\right)$$

or equivalently

$$\hat{e_i} \in \rho_i \pm \left((2ce+1)(d-1)\epsilon n^{d-1} + 2ce(d-1)\frac{error}{|S|}n^{d-1}\right) \tag{11}$$

with probability at least $1-4/n^{f+1-(d-1)} = 1-4/n^{f+2-d}$. Taking the union bound all n, values ρ_i are (simultaneously) estimated to within this bound with probability at least

$$1 - n\cdot 4/n^{f+2-d} = 1 - 4/n^{f+1-d}.$$

So, together with (9) we get with probability at least $1 - 4/n^{f+1-d} - 4/n^{f+1} \approx 1 - 4/n^{f+1-d}$ the following:

$$\begin{aligned} & t + \frac{n}{|S|}\sum_{i\in S}\hat{a}_i\hat{e}_i \\ \in\ & t + \frac{n}{|S|}\sum_{i\in S}\hat{a}_i\big(\rho_i \pm (2ce+1)(d-1)\epsilon n^{d-1} \\ & +2ce(d-1)\frac{error}{|S|}n^{d-1}\big) \\ & \text{by (11)} \\ \subseteq\ & t + \frac{n}{|S|}\sum_{i\in S}\hat{a}_i\rho_i \pm (2ce+1)(d-1)\epsilon n^d \\ & +2ce(d-1)\frac{error}{|S|}n^d \\ & \text{due to } \sum_{i\in S}\hat{a}_i \le |S| \le n \\ \subseteq\ & t + \frac{n}{|S|}\sum_{i\in S}a_i\rho_i \pm 2ce\frac{error}{|S|}n^d \\ & \pm(2ce+1)(d-1)\epsilon n^d + 2ce(d-1)\frac{error}{|S|}n^d \\ & \text{by (10)} \end{aligned}$$

So, we get that

$$\begin{aligned} & t + \frac{n}{|S|}\sum_{i\in S}\hat{a}_i\hat{e}_i \\ \subseteq\ & t + \frac{n}{|S|}\sum_{i\in S}a_i\rho_i \\ & \pm(2ce+1)(d-1)\epsilon n^d + 2ced\frac{error}{|S|}n^d \\ \subseteq\ & t + \sum_{i\in N}a_i\rho_i \pm (2ce+1)\epsilon n^d \\ & \pm(2ce+1)(d-1)\epsilon n^d + 2ced\frac{error}{|S|}n^d \\ & \text{by (9)} \\ \subseteq\ & t + \sum_{i\in N}a_i\rho_i \pm (2ce+1)d\epsilon\cdot n^d + 2ced\frac{error}{|S|}\cdot n^d. \end{aligned}$$

□

A.3. Transforming degree d constraints into linear constraints

First, let us observe that the previous proof for Evaluate shows implicitly that Evaluate estimates the values of all polynomials arising from the decomposition of a polynomial p with the described accuracy with probability at least $1 - 4/n^{f+1-d}$. Specifically, it estimates every polynomial of degree d' to within $\big((2ce+1)d'\epsilon + 2ced'\frac{error}{|S|}\big)n^{d'}$.

Algorithm 2 Linearize$\big(L \le p(\mathbf{x}) \le U, S, \{\hat{a_i} : i \in S\}, \epsilon\big)$

Require: constraint involving polynomial p of degree d,
set of variables indices S,
predictions $\hat{a_i}$ for $i \in S$
parameter $\epsilon > 0$.
Ensure: A set of linear constraints.
if p is linear **then**
output the input constraint $L \le p(\mathbf{x}) \le U$
else
Out $\leftarrow \emptyset$
$p(x_1, \ldots, x_n) = t + \sum x_i p_i(x_i, \ldots, x_n)$
for $i \in \{1, 2, \ldots, n\}$ **do**
$\hat{e_i} \leftarrow \textsc{Evaluate}(p_i, S, \{\hat{a_i} : i \in S\})$
$l_i \leftarrow \hat{e_i} - \big((2ce+1)(d-1)\epsilon + 2ce(d-1)\frac{error}{|S|}\big)n^{d-1}$
$u_i \leftarrow \hat{e_i} + \big((2ce+1)(d-1)\epsilon + 2ce(d-1)\frac{error}{|S|}\big)n^{d-1}$
Out$\leftarrow$ Out $\cup$ Linearize$\big(l_i \le p_i(x_i, \ldots, x_n) \le u_i, S, \{\hat{a_i} : i \in S\}, \epsilon\big)$
end for
output Out $\cup\{$
$t + \sum x_i\hat{e_i} \ge L - \big((2ce+1)d\epsilon + 2ced\frac{error}{|S|}\big)n^d$,
$t + \sum x_i\hat{e_i} \le U + \big((2ce+1)d\epsilon + 2ced\frac{error}{|S|}\big)n^d\}$
end if

We now use a modified version of algorithm Linearize

(see Algorithm 2) given in (Arora et al., 1999) to transform any polynomial constraint into a family of linear constraints. LINEARIZE is a recursive algorithm that uses EVALUATE to output linear constraints. It is easy to see that LINEARIZE outputs a set of at most $2n^{d-1}$ linear constraints such that the optimal solution $\mathbf{a}$ satisfies all constraints and $\mathbf{a}$ is a feasible solution to (d-IP), as long as EVALUATE estimates all polynomials with the required accuracy. This happens since the decompositions of the polynomials are unique and common between the two algorithms. Thus, by the observation stated previously, we have that with probability at least $1 - 4d/n^{f+1-d}$ the linear constraints output by EVALUATE are jointly feasible. Let us now see why (using induction on the degree d). It is obviously true for $d = 1$. Assume it is for $d - 1 \geq 1$. Then for a polynomial p of degree d, LINEARIZE outputs at most n sets of constraints associated to polynomial of degree $d - 1$, and two "new" constraints associated to p. By union bound (and recursive argument), they are simultaneously satisfied with probability at least $1 - n4(d-1)/n^{f+1-(d-1)} - 4/n^{f+1-d} = 1 - 4d/n^{f+1-d}$.

Next, we show (as in (Arora et al., 1999)) that any feasible solution to the linear system output by EVALUATE is also an approximate solution to the input constraint (degree d polynomial constraint).

Lemma A.3. *Every feasible solution $(y_i) \in [0,1]^n$ to the set of linear constraints output by* LINEARIZE *satisfies (without any assumption on the success of the sampling for set S):*

$$p(\mathbf{y}) \in [L,U] \pm \left((4ce+2)d(d-1)\epsilon + 4ced(d-1)\frac{error}{|S|} \right) n^d.$$

Proof. We show the lemma by induction on degree d. The base case $d = 1$ is trivial. For the inductive step we have that $d \geq 2$ and that $\mathbf{y}$ is feasible for all constraints output by LINEARIZE. By the inductive hypothesis we get for each i:

$$p_i(\mathbf{y}) \in [l_i, u_i]$$
$$\pm \left((4ce+2)(d-1)(d-2)\epsilon + 4ce(d-1)(d-2)\frac{error}{|S|} \right) n^{d-1}$$

So, it follows by substituting l_i, u_i that

$$p_i(\mathbf{y}) \in \hat{e_i}$$
$$\pm \left((2ce+1)(d-1)\epsilon + 2ce(d-1)\frac{error}{|S|} \right) n^{d-1}$$
$$\pm \left((4ce+2)(d-1)(d-2)\epsilon + 4ce(d-1)(d-2)\frac{error}{|S|} \right) n^{d-1}$$
$$\subseteq \hat{e_i} \pm \left((2ce+1)(d-1)(2d-3)\epsilon + 2ce(d-1)(2d-3)\frac{error}{|S|} \right) n^{d-1}. \quad (12)$$

Thus,

$$p(\mathbf{y}) = t + \sum y_i p_i(y_i, \ldots, y_n)$$
$$\subseteq t + \sum y_i \cdot \left(\hat{e_i} \pm \left((2ce+1)(d-1)(2d-3)\epsilon + 2ce(d-1)(2d-3)\frac{error}{|S|} \right) n^{d-1} \right)$$
by (12)
$$\subseteq t + \sum y_i \hat{e_i}$$
$$\pm \left((2ce+1)(d-1)(2d-3)\epsilon + 2ce(d-1)(2d-3)\frac{error}{|S|} \right) n^d$$
$$\subseteq [L,U] \pm \left((2ce+1)d\epsilon + 2ced\frac{error}{|S|} \right) n^d$$
$$\pm \left((2ce+1)(d-1)(2d-3)\epsilon + 2ce(d-1)(2d-3)\frac{error}{|S|} \right) n^d,$$

from the fact that y is feasible for the constraint which was output by LINEARIZE before recursion,

$$\subseteq [L,U] \pm \left((2ce+1)\epsilon(2d^2-4d+3) + 2ce(2d^2-4d+3)\frac{error}{|S|} \right) n^d$$
$$\subseteq [L,U] \pm \left(2(2ce+1)d(d-1)\epsilon + 4ced(d-1)\frac{error}{|S|} \right) n^d.$$

The last inequality holds, since $d \geq 2 > 3/2$. □

A.4. Randomized Rounding for Smooth Polynomials

Finally, we have to round our fractional solution to get an integral one. Using a lemma from (Arora et al., 1999), which shows that the randomized rounding outputs an integer value which is close to the fractional one for every c-smooth degree-d polynomial, we conclude the last step of LA-PTAS. We restate the lemma for completeness.

Lemma A.4. *Randomized rounding for degree-d polynomials (Arora et al., 1999) Let p be a c-smooth degree-d polynomial. Let $\mathbf{y} \in [0,1]^n$ be such that $p(y_1, \ldots, y_n) = b$. Performing randomized rounding on y_i to yield a $0,1$ vector (z_i) we get that with probability at least $1 - n^{d-f}$ we have that*

$$p(z_1, \ldots, z_n) \in [b \pm gdn^{d-1/2}\sqrt{\ln n}], \quad (13)$$

where $g = 2ce\sqrt{f}$.

A.5. Proof of Theorem 4.2

Proof. We have a feasible c-smooth degree-d PIP with $m = \text{poly}(n)$ constraints each one of which has degree at most d. Let $\mathbf{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$ be a feasible solution. Here, we focus on the maximization problem (the minimization one is similar) of a polynomial $p(\mathbf{x})$, where $x \in \{0,1\}^n$.

Assume that we have found the optimal value $|OPT| > 0$ of p using binary search in time $O(\log cn^d)$. Then we write the maximization problem of p as a feasibility one with $p(\mathbf{x}) \geq |OPT|$, which is of course feasible.

Let $f > 0$ be such that $n^f = 2m(n+4d)n^d/n = \Theta(m \cdot n^d)$. We let $\epsilon' = \frac{\epsilon}{(4ce+2)d(d-1)}$, $g = \Theta(cfd/\epsilon'^3)$ and $|S| = g \ln n$. Then, we take a random sample S of variables with replacement and we are given a prediction $\hat{a}_i$ on the values a_i for each $i \in S$. We use LINEARIZE with error parameter ϵ' and replace each degree d' constraint with $O(n^{d'-1})$ linear constraints. Therefore, we construct a linear integer system with $O(m \cdot n^{d-1})$ constraints. This new system is feasible with probability at least $1 - 4md/n^{f+1-d}$, since $\mathbf{a}$ is an optimal solution to the PIP.

Let us now relax the integrality constraint of each variable and solve the linear system with n variables and $O(m \cdot n^{d-1})$ constraints in time T'_{LP}. From Lemma A.3 for the fractional solution $\mathbf{y}$ we get that the following holds:

$$p(\mathbf{y}) \geq |OPT| - \left((4ce+2)d(d-1)\epsilon' + 4ced(d-1)\frac{error}{|S|}\right)n^d.$$

Next, we use randomized rounding to get an integer solution $\mathbf{z}$ that increases the additive loss by at most $O(n^{d-1/2}\sqrt{\ln n}) = o(n^d)$. The rounding from Lemma A.4 works simultaneously for all m constraints with probability at least $1 - m/n^{f-d}$.

Consequently, our randomized learning-augmented approximation scheme works with probability at least $1 - m/n^{f-d} - 4md/n^{f+1-d} > 1/2$ and outputs a solution such that

$$|\text{LA-PTAS}| \geq |OPT| - \left(\epsilon + 4ced(d-1)\frac{error}{|S|}\right)n^d,$$

where $|S| = \Theta(\frac{128c^4e^4fd^7}{\epsilon^3}\ln n) = \Theta(\frac{c^4fd^7}{\epsilon^3}\ln n)$.

For the running time, we have to also guess the value of the *error* which takes at most n. So, in total the general algorithm LA-PTAS runs in time $O\big(n\ln(cn^d) \cdot T'_{LP}\big)$. □

B. Missing material from Section 5

MAX-DICUT Let us write MAX-DICUT of a directed graph $G = (V, E)$ as an 1-smooth degree-2 polynomial integer program as follows:

$$\max \sum_{(i,j)\in E} (1 - x_i)x_j$$
$$\text{s.t. } x_i \in \{0,1\}\, \forall i.$$

In a directed graph with density δ, the value of the maximum cut is at least $\delta n^2/4$. Using LA-PTAS and Theorem 4.2 with $\epsilon' = \delta\epsilon/4$ we find a cut of value at least

$$|OPT|\left(1 - \epsilon - 32e\frac{error}{\delta|S|}\right),$$

where $|S| = O(\ln n/(\epsilon^3\delta^3))$ and the running time is $O(\ln n \cdot T'_{LP})$. For the robustness of LAA-GENERAL, there is a polynomial time randomized approximation algorithm that achieves a ratio of 0.859 (Feige & Goemans, 1995). The rest of the steps to construct our two learning-augmented schemes for MAX-DICUT follow trivially.

MAX-HYPERCUT(d) We can formulate the problem as a smooth degree-d PIP. Given an edge (set of vertices) S', we use the term $1 - \prod_{i\in S'} x_i - \prod_{i\in S'}(1 - x_i)$, which is 1 if S' is cut and 0 otherwise. Moreover, for the robustness of LAA-GENERAL we can use the randomized poly-time algorithm with approximation ratio of 0.72 (Andersson & Engebretsen, 1998).

k-DENSEST SUBGRAPH Let $k \geq \gamma n$. If the graph is δ-dense, using an averaging argument we have that the optimal solution contains at least $\gamma^2\delta n^2/2$ edges. The problem is equivalent to maximizing the following degree-2 1-smooth PIP:

$$\max p(\mathbf{x}) = \sum_{\{i,j\}\in E} x_i x_j$$
$$\text{s.t. } \sum_{i=1}^{n} x_i = k$$
$$x_i \in \{0,1\}.$$

We use the general algorithm LA-PTAS with $\epsilon' = \epsilon\gamma^2\delta/2$ and noticing that our solution z satisfies $\sum_{i=1}^n x_i \in [k \pm g\sqrt{n\ln n}]$ (Lemma A.4). Moving in or out at most $O(\sqrt{n\ln n})$ vertices, as $g = O(1)$, reduces the number of edges included in the subgraph by at most $n\sqrt{n\ln n} = o(n^2)$.

There is no known constant approximation poly-time algorithm for the problem. There is a deterministic greedy algorithm that achieves a ratio of $O(k/n)$ (Asahiro et al., 1996), which is equal to $O(\gamma)$ in our case (we assume in this work that γ is a constant). In (Feige & Langberg, 2001), they give a randomized algorithm with approximation ratio at least $k/n \geq \gamma$. Finally, we can also use the original PTAS of (Arora et al., 1999) or (Alon et al., 2003) for a not too small $\epsilon > 0$.