APPROXIMATION ALGORITHMS FOR COMBINATORIAL OPTIMIZATION WITH PREDICTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We initiate a systematic study of utilizing predictions to improve over approximation guarantees of classic algorithms, without increasing the running time. We propose a systematic method for a wide class of optimization problems that ask to select a feasible subset of input items of minimal (or maximal) total weight. This gives simple (near-)linear time algorithms for, e.g., Vertex Cover, Steiner Tree, Min-Weight Perfect Matching, Knapsack, and Clique. Our algorithms produce optimal solutions when provided with perfect predictions and their approximation ratios smoothly degrade with increasing prediction error. With small enough prediction error we achieve approximation guarantees that are beyond reach without predictions in the given time bounds, as exemplified by the NP-hardness and APX-hardness of many of the above problems. Although we show our approach to be optimal for this class of problems as a whole, there is a potential for exploiting specific structural properties of individual problems to obtain improved bounds; we demonstrate this on the Steiner Tree problem. We conclude with an empirical evaluation of our approach.

1 INTRODUCTION

Combinatorial optimization studies problems of selecting an optimal solution, with respect to a given cost function, from a discrete set of potential solutions. This framework is ubiquitous, both in theory and in practice, finding application in a vast number of areas, e.g., resource allocation, machine learning, efficient networks, or logistics. However, these problems are typically NP-hard, and hence they cannot be solved to optimality in polynomial time (unless P equals NP). It is thus necessary to develop efficient algorithms even if this comes at the cost of relaxing the optimality requirement. This trade-off between optimality and tractability is the main paradigm of the (classic) theory of approximation algorithms, see, e.g., (Williamson and Shmoys, 2011).

Despite the success of the classic analysis of approximation algorithms, its distinction between tractable and intractable problems is often too crude. Indeed, for modern big data applications, a running time polynomial in the input size can hardly be considered efficient, as we often require algorithms with a running time that is a small polynomial or even linear in the input size. This more fine-grained point of view on algorithm design has received considerable attention in recent years. Many combinatorial optimization problems (e.g., Vertex Cover, Steiner Tree, or Knapsack) admit simple and fast (near-)linear time constant-factor approximation algorithms (Bar-Yehuda and Even, 1981; Mehlhorn, 1988; Dantzig, 1957), but improving upon their approximation guarantees often seems to require significantly more running time (sometimes even exponentially, as is the case for Vertex Cover (Khot and Regev, 2008)). However, it is important to note that such limitations are based on worst-case analysis, and do not necessarily represent the difficulty of a typical instance.

It is natural to expect that instances arising in practice can be modelled as coming from a fixed distribution, making them amenable to machine-learning techniques. Using historic input data and some (possibly computationally expensive) training, we can hope to guide an efficient algorithm with poor (worst-case) approximation guarantee to obtain near-optimal solutions.

1.1 Our contribution

In this paper, we initiate a systematic study of utilizing predictions to improve over the approximation guarantees of classic algorithms without increasing the running time. This approach fits the current line of research on utilizing predictions to improve performance of algorithms, started by the seminal papers of Kraska et al. (2018) and Lykouris and Vassilvitskii (2021). We focus on a broad class of optimization problems, which we call *selection problems*, and which captures many fundamental problems in combinatorial optimization, e.g., Set Cover, Travelling Salesperson Problem (TSP), Steiner Tree, or Knapsack.

Definition 1 (Selection problem)**.** *We are given n items, numbered with integers from $[n] := \{1, 2, \ldots, n\}$, with non-negative weights $w\colon [n] \to \mathbb{R}_{\geqslant 0}$ and an implicit collection $\mathcal{X} \subseteq 2^{[n]}$ of feasible subsets of items. Our task is to find a feasible solution $X \in \mathcal{X}$ minimizing (or maximizing) its total weight $w(X) := \sum_{i \in X} w(i)$.*

Note that the complexity of this task comes from the size of $\mathcal{X}$, which is usually exponential in n.

Our work considers the *offline* setting, where the whole input is available to the algorithm from the start. This is in contrast to much of the to-date research on learning-augmented algorithms, which focuses on *online* problems, where the input is not known to the algorithm in advance and the predictions are used to decrease uncertainty about the future (see the survey of Mitzenmacher and Vassilvitskii (2020)). So far, offline problems were studied mostly in the *warm-start* setting, where predictions in the form of solutions to past instances are used to speed up exact algorithms (see Section 1.4 on related work). The focus of such works is on the dependence between the running time and the quality of the predictions.

Our target is to maintain a superb running time in all situations, and we study the dependence of approximation ratio on the prediction quality. Having access to past instances, one can hope to learn which items are likely to be part of an optimal solution and provide the set of such items to the algorithm as a prediction. Utilizing such prediction comes with two main challenges.

(1) The predicted set of items may be infeasible. Ensuring feasibility of solutions produced by deep-learning models is a challenging problem, and enforcing even simple combinatorial constraints requires very complex neural architectures (Bengio et al., 2021). In order to utilize potentially infeasible predictions effectively, we need algorithms that can transform the prediction into a feasible solution without discarding the correct parts of the prediction. We carefully design a *black-box* mechanism that turns *any* approximation algorithm into a learning-augmented algorithm (utilizing possibly infeasible predictions) whose approximation ratio smoothly deteriorates with increasing prediction error. Our approach gives (near-)linear time algorithms for many problems, e.g., Vertex Cover, Steiner Tree, Min-Weight Perfect Matching, Knapsack, and Clique.

(2) The predicted set of items may contain costly mispredictions. When given a mostly correct prediction, it is certainly possible to detect a single mispredicted item with a huge weight, say, larger than the total weight of the optimal solution. Our goal is to push this idea to its limits by detecting mispredictions which are as small as possible. We study this challenge on the case of the Steiner Tree problem. We develop an algorithm that can detect and eliminate mispredictions of cost larger than the cost of a single connection between a pair of terminals in an optimal solution.

1.2 Our results in more detail

Before stating our results formally, we have to define two crucial quantities: the approximation ratio and the prediction error.

Approximation ratio. We measure the quality of solutions produced by approximation algorithms using the standard notion of *approximation ratio*. For minimization problems, we say that an algorithm is a ρ-approximation algorithm if we are always guaranteed that $w(X) \leqslant \rho \cdot w(X^*)$, where X denotes the solution returned by the algorithm, and X^* denotes an optimal solution for the same instance. Analogously, for maximization problems, a ρ-approximate algorithm always satisfies $w(X) \geqslant \rho \cdot w(X^*)$. We refer to ρ as the approximation ratio. For minimization problems we always have $\rho \geqslant 1$ and smaller ratios are better; for maximization problems $\rho \in [0, 1]$ and larger ratios are better.

Prediction and prediction error. The prediction received by our algorithms is an arbitrary (not necessarily feasible) set of items $\widehat{X} \subseteq [n]$. We say that $\widehat{X}$ *has error* (η^+, η^-) *with respect to a solution* X, if

$$\eta^+ := w(\widehat{X} \setminus X), \quad \text{and} \quad \eta^- := w(X \setminus \widehat{X}),$$

i.e., η^+ is the total weight of false positives, and η^- is the total weight of false negatives. Usually we consider the prediction error with respect to an optimal solution, denoted by X^*. We note that such predictions are PAC-learnable (see Section 1.3). In our performance bounds, the prediction error can be always considered with respect to *the closest* optimal solution.

Let us also remark that an alternative setting, where each item $i \in [n]$ is associated with a fractional prediction $p_i \in [0, 1]$ (supposed to denote the a *confidence* that i is part of the optimal solution), is actually equivalent to the setting we consider. Indeed, one can convert such fractional predictions to a set $\hat{X}$ with simple randomized rounding (adding each item i to $\hat{X}$ independently with probability p_i) and the expected value of the prediction error stays the same.

Minimization problems. Let Π be a problem of selecting a feasible solution $X \in \mathcal{X}$ of minimum total weight. In Section 2, we show a black-box approach to turn any off-the-shelf ρ-approximation algorithm A for Π into a learning-augmented algorithm for Π with approximation ratio

$$1 + \frac{\eta^+ + (\rho - 1) \cdot \eta^-}{\mathrm{OPT}}.$$

Here, (η^+, η^-) denotes the error of the prediction $\widehat{X}$ given to the algorithm with respect to an optimal solution X^* of weight $\mathrm{OPT} := w(X^*)$. The asymptotic running time of the resulting algorithm is the same as that of A.

To gain some intuition about the above approximation ratio guarantee, note that for the trivial prediction $\widehat{X} = \emptyset$ we have $\eta^+ = 0$ and $\eta^- = \mathrm{OPT}$, and in turn $1 + (\eta^+ + (\rho - 1) \cdot \eta^-)/\mathrm{OPT} = \rho$, which squarely corresponds to simply running the (prediction-less) algorithm A. This shows that the $\rho - 1$ factor in front of η^- is necessary.

In Section 2.1 we give several examples of how this black-box approach can be applied in order to obtain (near-)linear time learning-augmented approximation algorithms for some fundamental problems in combinatorial optimization, namely Minimum (and Min-Weight) Vertex Cover, Minimum (and Min-Weight) Steiner Tree, and Min-Weight Perfect Matching (in graphs with edge weights satisfying the triangle inequality).

Maximization problems. In Section 3 we give a similar result for maximization problems. Let Π be a problem of selecting a set *maximizing* the total weight from the collection of sets of items $\mathcal{X}$. Let A be a ρ-approximation algorithm for the corresponding *complementary* problem of selecting a set *minimizing* the total weight over sets of items Y such that $([n] \setminus Y) \in \mathcal{X}$. We construct an algorithm for Π with running time asymptotically the same as that of A and with approximation ratio

$$1 - \frac{(\rho - 1) \cdot \eta^+ + \eta^-}{\mathrm{OPT}}$$

given predictions of error (η^+, η^-) with respect to an optimal solution X^* of weight $\mathrm{OPT} := w(X^*)$.

Even though the notion of the complementary problem may not seem intuitive at first, we note that for many natural problems the complementary problem also happens to be a natural and well studied problem, e.g., Vertex Cover is the complementary problem of Independent Set. In Section 3.1 we elaborate on how to apply our black-box construction to Clique and Knapsack.

Lower bounds. In Section C, we show that our black-box approach from Sections 2 and 3, despite being simple, cannot be improved for the class of selection problems as a whole. More specifically, regarding minimization problems, we show that for the Vertex Cover problem with predictions, any (polynomial-time) learning-augmented algorithm with an approximation ratio with a better dependence on the prediction error would contradict the Unique Games Conjecture (UGC), which is a standard assumption in computational complexity but to the best of our knowledge has not been used before in the context of learning-augmented algorithms. Regarding maximization problems, we give a similar UGC-based lower bound for the Clique and Independent Set problems.

Refined algorithm for Steiner Tree. Although our lower bounds are tight for the considered classes of combinatorial optimization problems as a whole, they do not rule out refined upper bounds, e.g., for specific problems or in terms of other (more fine-grained) measures of prediction errors. In Section 4 we propose such a refined algorithm for the Steiner Tree problem. In this problem, a small number of false positives with a large weight can have a large impact on the performance of our generic black-box algorithm from Section 2. Our refined algorithm is guided by a hyperparameter α in order to detect and avoid false positives with high weight. It is based on a 2-approximation algorithm called the *MST heuristic* (Kou et al., 1981; Mehlhorn, 1988). The contribution of false positives to our algorithm's performance guarantee depends only on their number, and not on their weights, and it is capped by the cost of individual connections made by the MST heuristic (without predictions). The final approximation guarantee follows from a careful analysis of how the output of our algorithm converges to the prediction as its hyperparameter α increases. We also show that our analysis of this algorithm is tight.

Experimental results. Section D concludes the paper with an experimental evaluation of our refined Steiner Tree algorithm on graphs from the 2018 Parameterized Algorithms and Computational Experiments (PACE) Challenge (Bonnet and Sikora, 2018), using as a benchmark the winning Steiner Tree solver from that challenge (Ruiz et al., 2018). These experiments allow us to better understand the impact of the hyperparameter α on the performance of our algorithm. They also demonstrate that (for a sufficiently concentrated input distribution) we can find near-optimal solutions in time in which conventional algorithms can achieve only rough approximations.

1.3 Further remarks on our setting

Learnability. The predictions that our algorithms require are PAC-learnable via the following simple argument. First, since the space of possible predictions is finite and its size is single exponential in n, it suffices to perform empirical risk minimization (ERM) on a polynomial number of samples (see, e.g., Polak and Zub (2024, Theorem 5)). Second, ERM for the combined prediction error $\eta^+ + \eta^-$ boils down to taking a coordinate-wise majority vote of solutions to the sampled instances. We use this approach in our experiments in Section D.

At the same time, our setting is flexible enough to allow for other methods of generating predictions. For instance, it is not hard to imagine a deep-learning model that assigns to each input element the probability that it belongs to an optimal solution (Joshi et al., 2019; Ahn et al., 2020). Then, a prediction can be obtained by sampling each element with the assigned probability.

We remark that any such learning is likely to be computationally expensive, but this should come at no surprise, because the resulting predictions can then be utilized by our learning-augmented algorithms to "break" known lower bounds. The time saved this way must be spent somewhere else, i.e., during learning. The advantage is that, when we are repeatedly solving similar recurring instances, this costly learning process is performed only once, and the resulting predictions can be (re-)used multiple times.

Infeasible predictions. We stress out that it is an absolutely crucial characteristic of our work that our algorithms accept as predictions sets that are not necessarily feasible solutions.[1] Bengio et al. (2021) argue that ensuring feasibility of predictions significantly increases the complexity of the learning process, with difficulties specific to different types of combinatorial constraints. Accepting infeasible predictions allows for simpler and possibly more versatile learning approaches like the ones outlined in the previous paragraphs.

Even when the input changes very slightly, a previously feasible solution may not be feasible anymore, so one should not expect feasibility of predictions based on past data. One such example scenario is given in our experiments on the Steiner Tree problem in Section D, where the underlying graph is fixed and the set of terminals changes, hence changing the set of feasible solutions. It might be the case that a part of the solution is recurring while the other part is changing constantly, from instance to

[1] It is a common characteristic of learning-augmented algorithms. E.g., in the paper on warm-starting max-flow by Davies et al. (2023) most of the technical insights are in the part of the algorithm that projects any (potentially infeasible) prediction into a feasible solution.

instance. Then, a learning algorithm can easily predict the stable part of the solution, but in advance it is hard to extend such a partial solution into a complete feasible solution with reasonable accuracy.

1.4 Related work

Learning-augmented approximation algorithms. Approximation algorithms with predictions have so far received very little attention and have only been investigated for specific problems and in restricted settings.

Bampis et al. (2024) studied *dense* variants of several hard problems – such as Max Cut, Max k-SAT, or k-Densest Subgraph – which all are examples of maximization selection problems[2]. In the dense setting, these problems admit polynomial-time approximation schemes. In particular, for any fixed $\epsilon > 0$, there is a $(1-\epsilon)$-approximation algorithm for Max Cut by Arora et al. (1999) that runs in time $O(n^{1/\epsilon^2})$, i.e., exponential in the precision parameter $1/\epsilon$. Bampis et al. (2024) propose an algorithm that always runs in time $O(n^{3.5})$ and achieves approximation ratio depending on the quality of the received prediction. An interesting feature of their work is that the algorithms in the dense setting need only a small sample of the predicted solution to achieve their guarantees. In particular, they only need to know $O(\mathrm{poly}(\log n/\epsilon))$ bits of information about items belonging or not belonging to the optimal solution to achieve an approximation ratio of the form $1-\epsilon-f(\eta)$, where η denotes the prediction error.

Ergun et al. (2022) and Gamlath et al. (2022) independently proposed a linear time algorithm for k-Means Clustering that receives labels of the input points as predictions. This problem can be formulated as a minimization selection problem only in the special case where the number of potential centers is bounded (e.g., if the potential centers are the input points themselves). If the prediction has per-cluster label error rate at most α with respect to some $(1+\alpha)$-approximate solution, their algorithm achieves an approximation ratio of $1+O(\alpha)$. The algorithm uses techniques from robust statistics to identify outliers in the predicted labeling. Nguyen et al. (2023) slightly improve upon this guarantee in their follow-up work.

There are also two very recent works, on the Max Cut (Cohen-Addad et al., 2024) and Independent Set (Braverman et al., 2024) problems, in the setting with *ϵ-accurate predictions*. In this setting, each input item comes with a label (indicating whether the item belongs to a fixed optimal solution or not), which is correct with probability $1/2+\epsilon$, *independently* from other labels. The two papers show how to breach the respective approximation ratio barriers of 0.878 and $n^{1-o(1)}$, for any $\epsilon > 0$. Note that our results assume that the incorrect parts of the prediction are selected adversarially rather than randomly.

Other learning-augmented algorithms. Another related line of work is that on exact offline algorithms with *warm start*. In this setting, the algorithm, in contrast to starting its computation from scratch, has access to predictions or information (e.g., a partial solution) that allow it to start computing from a "more advanced" state, leading to an improvement in the running time. Such results include Kraska et al. (2018) on binary search, Dinitz et al. (2021); Chen et al. (2022) on Bipartite Matching, Feijen and Schäfer (2021); Lattanzi et al. (2023) on Shortest Path, Polak and Zub (2024); Davies et al. (2023) on Max Flow, and Bai and Coester (2023) on Sorting. There are also works considering multiple predictions (Dinitz et al., 2022; Sakaue and Oki, 2022) and a work by Tang et al. (2020) on using reinforcement learning to improve the cutting plane heuristic for Integer Programming.

Since the seminal papers by Kraska et al. (2018) and Lykouris and Vassilvitskii (2021), which initiated the study of learning-augmented algorithms in the modern sense, many online computational problems were considered. There are papers on, e.g., ski rental (Purohit et al., 2018), secretary problem (Dütting et al., 2024), energy efficient scheduling (Bamas et al., 2020), online page migration (Indyk et al., 2022), online TSP (Bernardini et al., 2022), and flow-time scheduling (Azar et al., 2021; 2022). Further related works can be found on the website by Lindermayr and Megow (2022).

[2]E.g., for Max Cut, $\mathcal{X} \subseteq 2^E$ contains all sets of edges that correspond to a cut in the input graph.

2 MINIMIZATION PROBLEMS

Our algorithm for minimization selection problems with linear objective receives a prediction $\widehat{X} \subseteq [n]$ which may not be feasible. It changes the weight of each item in $\widehat{X}$ to 0 and runs a conventional ρ-approximation algorithm on the problem with those modified weights. This way, we obtain a solution which is always feasible and its quality is described by the following theorem. In Section C, we show that this simple approach already matches a lower bound based on the UGC.

Theorem 1. *Let Π be a minimization selection problem, and let A be a ρ-approximation algorithm for Π running in time $T(n)$. Then, there exists an $O(T(n))$-time learning-augmented approximation algorithm for Π with the following guarantee: Upon receiving a (not necessarily feasible) predicted solution $\widehat{X} \subseteq [n]$, it outputs a solution X such that, for any feasible solution $X' \in \mathcal{X}$, we have $w(X) \leqslant w(X') + \eta^+ + (\rho - 1) \cdot \eta^-$, where (η^+, η^-) is the error of $\widehat{X}$ with respect to X'.*

We remark that if $X' = X^*$ is an optimal solution with objective value OPT, the preceding bound implies that our algorithm's approximation ratio is at most

$$1 + \frac{\eta^+ + (\rho - 1) \cdot \eta^-}{\mathrm{OPT}}.$$

Proof. The algorithm works as follows: Set

$$\bar{w}(i) = \begin{cases} 0, & \text{if } i \in \widehat{X} \\ w(i), & \text{otherwise} \end{cases} \quad \text{for } i = 1, 2, \ldots, n.$$

Run algorithm A with weight function $\bar{w}$ and return X, the solution returned by the algorithm.

We claim that $w(X) \geqslant w(X') + \eta^+ + (\rho - 1)\eta^-$. Since A is a ρ-approximation algorithm and X' is a feasible solution for $\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_n$, it holds that

$$w(X \setminus \widehat{X}) = \bar{w}(X) \leqslant \rho \cdot \bar{w}(X') = \rho \cdot w(X' \setminus \widehat{X}).$$

Then,

$$\begin{aligned} w(X) &= w(X \cap \widehat{X}) + w(X \setminus \widehat{X}) \\ &\leqslant w(X \cap \widehat{X}) + \rho \cdot w(X' \setminus \widehat{X}) \\ &= w(X \cap \widehat{X}) + w(X' \setminus \widehat{X}) + (\rho - 1) \cdot w(X' \setminus \widehat{X}) \\ &\leqslant w(\widehat{X}) + w(X' \setminus \widehat{X}) + (\rho - 1) \cdot w(X' \setminus \widehat{X}). \end{aligned}$$

Note that

$$w(\widehat{X}) + w(X' \setminus \widehat{X}) = w(\widehat{X} \cup X') = w(X') + w(\widehat{X} \setminus X').$$

Thus we have,

$$w(X) \leqslant w(X') + w(\widehat{X} \setminus X') + (\rho - 1) \cdot w(X' \setminus \widehat{X}) = w(X') + \eta^+ + (\rho - 1)\eta^-,$$

where (η^+, η^-) is the error of $\widehat{X}$ with respect to X'. If $X' = X^*$ is an optimal solution, we have

$$w(X) \leqslant \mathrm{OPT} + \eta^+ + (\rho - 1) \cdot \eta^- = \left(1 + \frac{\eta^+ + (\rho - 1) \cdot \eta^-}{\mathrm{OPT}}\right) \cdot \mathrm{OPT}. \qquad \square$$

In principle, the false-positive prediction error η^+ can be unbounded in terms of OPT, and therefore the approximation ratio of the algorithm of Theorem 3 cannot be bounded by any constant. That is, using the terminology of learning-augmented algorithms, the algorithm is not *robust*. However, as is the case for any offline algorithm, it can be robustified without increasing the asymptotic running time by simply running A in parallel and returning the better of the two solutions.

Corollary 2. *Under the same assumptions as in Theorem 1 there exists a learning augmented algorithm for Π running in time $O(T(n))$ with approximation ratio*

$$\min\left\{1 + \frac{\eta^+ + (\rho - 1) \cdot \eta^-}{\mathrm{OPT}}, \rho\right\}.$$

2.1 EXAMPLE APPLICATIONS

Minimum (and Min-Weight) Vertex Cover. In an undirected graph $G = (V, E)$, a *vertex cover* is a set $X \subseteq V$ such that every edge $e = (u, v) \in E$ has $u \in X$ or $v \in X$. The Min-Weight Vertex Cover problem asks, given a graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}_{\geqslant 0}$, to find a vertex cover $X \subseteq V$ of the minimum total weight $w(X)$, and the Minimum Vertex Cover problem is the special case with unit weights $\forall_{u \in V}\, w(u) = 1$, i.e., it asks for a vertex cover of minimum cardinality.

Already the unweighted variant is a very hard problem: Minimum Vertex Cover is included in Karp's seminal list of 21 NP-hard problems (Karp, 1972). Furthermore, under the UGC it is NP-hard to approximate Minimum Vertex Cover with any better than 2 multiplicative factor (Khot and Regev, 2008). This lower bound matches a folklore 2-approximation algorithm, which runs in linear time. Bar-Yehuda and Even (1981) show that also Min-Weight Vertex Cover admits a linear-time 2-approximation. Therefore, we can directly apply Theorem 1 and get a linear-time learning-augmented algorithm for Min-Weight Vertex Cover with the approximation ratio $1 + (\eta^+ + \eta^-)/\mathrm{OPT}$.

Minimum (and Min-Weight) Steiner Tree. Given an undirected graph $G = (V, E)$ and a subset $T \subseteq V$ of the vertices referred to as *terminals*, the Minimum Steiner Tree problem asks for a set of edges $X \subseteq E$ of minimum cardinality such that all terminals in T belong to the same connected component of (V, X). In the Min-Weight Steiner Tree problem the input also contains edge weights $w : E \to \mathbb{R}_{\geqslant 0}$ and the goal is to find such set X minimizing the total weight $w(X)$.

Minimum Steiner Tree is also among Karp's 21 NP-hard problems (Karp, 1972). A folklore algorithm, the so-called *minimum spanning tree heuristic* yields 2-approximation algorithm, also for the Min-Weight Steiner Tree problem, and Mehlhorn (1988) shows how to implement it in (near-)linear time $O(|E| + |V| \log |V|)$. A long line of work contributed many (polynomial-time) better-than-2-approximation algorithms, with the current best approximation factor 1.39 given by Byrka et al. (2013), but none of these algorithms runs in (near-)linear time and for many of them the running time is an unspecified polynomial with a huge exponent. The inapproximability lower bound is $96/95$ (Chlebík and Chlebíková, 2008), leaving a big gap open.

Our Theorem 1 together with Mehlhorn's algorithm gives a linear time learning-augmented Min-Weight Steiner Tree algorithm with approximation factor $1 + (\eta^+ + \eta^-)/\mathrm{OPT}$. For sufficiently accurate predictions, it gives better and faster approximation than the best conventional algorithms. In Section 4 we show how to exploit specific structural properties of the problem in order to obtain an even better algorithm.

Min-Weight Perfect Matching. For an undirected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_{\geqslant 0}$, the Min-Weight Perfect Matching problem asks to find a set $X \subseteq E$ of exactly $|X| = |V|/2$ edges such that each vertex $u \in V$ is an endpoint of exactly one of these edges and their total weight $w(X)$ is minimized. Unlike the previous two examples, this problem belongs to the class P and optimal solutions can be found with a fairly complicated exact algorithm that runs in $O(|V||E|)$ time (Edmonds, 1965; Gabow, 1990). For the special case of edge weights satisfying the triangle inequality[3] Goemans and Williamson (1995) give a linear-time 2-approximation algorithm. It is an open problem whether a better-than-2-approximation faster than $O(|V||E|)$ is possible for this special case. We show how to achieve it assuming sufficiently accurate predictions.

This time we need to work more than in the previous two examples. That is because the problem we tackle is not a selection problem – because of the triangle inequality not every weight function constitutes a correct input. If we tried to apply Theorem 1 directly we would run into the issue that decreasing the weights of predicted edges to zero may violate the triangle inequality.

Let us start by defining the so-called V-join problem: Given an edge-weighted graph find a min-weight set of edges that has odd degree on all the vertices of V. Goemans and Williamson (1995) give a 2-approximation algorithm for the V-join problem. They also show how to short-cut a ρ-approximate solution to the V-join problem to get a ρ-approximate solution to the Min-Weight Perfect Matching, provided that the edge weights satisfy the triangle inequality. Their algorithm runs in near-linear time.

[3]This special case is used as a subproblem in the famous 1.5-approximation TSP algorithm by Christofides (2022).

Our learning-augmented algorithm for Min-Weight Perfect Matching works in two steps. First, it uses Theorem 1 with the algorithm of Goemans and Williamson (1995) to find a solution to the V-join problem with approximation ratio at most $1 + (\eta^+ + \eta^-)/\mathrm{OPT}$. Then, provided that the original graph satisfies the triangle inequality, it transforms the V-join solution into a perfect matching with the same approximation ratio, using the short-cutting procedure of Goemans and Williamson (1995).

3 MAXIMIZATION PROBLEMS

Our algorithm for maximization selection problems with linear objective receives a prediction $\widehat{X} \subseteq [n]$ which may not be feasible. It changes the weight of each item in $[n] \setminus \widehat{X}$ to 0 and runs the ρ-approximation algorithm for the *complementary* problem to find a set Y. This way, it obtains a solution $X = [n] \setminus Y$ that is guaranteed to be feasible. Its quality is given by the following theorem. In Section C, we show that this simple approach already matches a lower bound based on UGC.

Theorem 3. *Let Π be some maximization selection problem. Let A be a $T(n)$-time ρ-approximation algorithm for the following complementary problem: Find a subset of items $Y \subseteq [n]$ minimizing $w(Y)$ such that $([n] \setminus Y) \in \mathcal{X}$. Then, there exists an $O(T(n))$ time learning-augmented approximation algorithm for Π with the following performance. Receiving a (not necessarily feasible) predicted solution $\widehat{X} \subseteq [n]$, it outputs a solution X such that, for any feasible solution $X' \in \mathcal{X}$, we have $w(X) \geqslant w(X') - (\rho - 1)\eta^+ - \eta^-$, where (η^+, η^-) is the error of $\widehat{X}$ with respect to X'.*

If $X' = X^*$ is an optimal solution with objective value OPT, the preceding bound implies that our algorithm's approximation ratio is at most

$$1 - \frac{(\rho - 1) \cdot \eta^+ + \eta^-}{\mathrm{OPT}}.$$

Proof. The algorithm works as follows: Set

$$\bar{w}(i) = \begin{cases} w(i), & \text{if } i \in \widehat{X} \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i \in [n].$$

Run algorithm A with weight function $\bar{w}$ and let Y denote the solution returned by the algorithm. Return $X = [n] \setminus Y$.

By the definition of the complementary problem, we have that $X = [n] \setminus Y \in \mathcal{X}$ is a feasible solution to Π. It remains to lower bound the weight of that solution in terms of $w(X')$. Note that

$$w(\widehat{X}) = w(X') + w(\widehat{X} \setminus X') - w(X' \setminus \widehat{X}).$$

Moreover, $[n] \setminus X'$ is a feasible solution to the complementary problem with weights $\bar{w}$, because $X' \in \mathcal{X}$. Hence, since A is a ρ-approximation algorithm, we get

$$\bar{w}(Y) \leqslant \rho \cdot \bar{w}\big([n] \setminus X'\big) = \rho \cdot w(\widehat{X} \setminus X').$$

Finally, we have

$$\begin{aligned} w(X) = w\big([n] \setminus Y\big) &\geqslant w(\widehat{X}) - \bar{w}(Y) \\ &\geqslant w(X') + w(\widehat{X} \setminus X') - w(X' \setminus \widehat{X}) - \rho w(\widehat{X} \setminus X') \\ &= w(X') - (\rho - 1)\eta^+ - \eta^-, \end{aligned}$$

where (η^+, η^-) is the error of $\widehat{X}$ with respect to X'. If X' is an optimal solution, we have

$$w(X) \geqslant \mathrm{OPT} - (\rho - 1)\eta^+ - \eta^- = \left(1 - \frac{(\rho - 1) \cdot \eta^+ + \eta^-}{\mathrm{OPT}}\right) \cdot \mathrm{OPT}. \qquad \square$$

Similarly to Theorem 1, the algorithm implied by Theorem 3 can be robustified by running it in parallel with a conventional approximation algorithm A' for problem Π.

Corollary 4. *Under the same assumptions as in Theorem 3 there exists a learning augmented algorithm for Π running in time $O(T(n))$ with approximation ratio*

$$\max\left\{1 - \frac{(\rho - 1)\eta^+ + \eta^-}{\mathrm{OPT}}, \rho'\right\},$$

where ρ' is the approximation ratio of the best known algorithm for Π running in time $O(T(n))$.

3.1 EXAMPLE APPLICATIONS

Maximum (and Max-Weight) Clique (and Independent Set). Given an undirected graph $G = (V, E)$, an *independent set* (also called *stable set*) is a subset of vertices $X \subseteq V$, such that no two vertices in X share an edge. The Maximum Independent Set problem asks for an independent set of largest cardinality. In the Max-Weight Independent Set problem the input also includes vertex weights $w : V \to \mathbb{R}_{\geqslant 0}$ and the goal is to find an independent set X maximizing the total weight $w(X)$. In the complement graph $G' = \left(V, \binom{V}{2} \setminus E\right)$, these problems are equivalent to the Maximum (and Max-Weight) Clique problems, which ask for the largest cardinality (weight) complete subgraph $K_\ell \subseteq G$.

Maximum Clique (and Independent Set) is not only NP-hard to solve exactly (Karp, 1972), but it is also NP-hard to approximate within any factor better than $n^{1-\epsilon}$, for any $\epsilon > 0$ (Håstad, 1999). Quite conveniently for us, the complementary problem to Max-Weight Independent Set is Min-Weight Vertex Cover, which can be 2-approximated in linear time (Bar-Yehuda and Even, 1981). Thus, applying Theorem 3, we get a linear-time learning-augmented algorithm for Min-Weight Independent Set (and Clique) with approximation ratio $1 - (\eta^+ + \eta^-)/\mathrm{OPT}$, which is in striking contrast to the aforementioned impossibility of any nontrivial approximation ratio for conventional algorithms.

Knapsack. Given the knapsack capacity c and n items, the i-th of which has size s_i and is worth w_i, the Knapsack problem asks to find a subset of items of total size at most c that maximizes the total worth. Knapsack is another example from Karp's list of NP-hard problems (Karp, 1972), but it admits approximation schemes. In particular, it can be approximated to within factor $(1 - \epsilon)$ in time $\tilde{O}(n + \epsilon^{-2})$ (Chen et al., 2024; Mao, 2024). Unfortunately, these asymptotically optimal algorithms are based on structural results from additive combinatorics that make the constants hidden in the asymptotic notation enormous and render the algorithms themselves impractical. Perhaps a more practical approach is the $O(n \log(\epsilon^{-1}) + \epsilon^{-2.5})$-time algorithm by Chan (2018). Still, if our goal is to solve with high accuracy (i.e., small ϵ) many similar Knapsack instances, it might be a useful strategy to use Chan's algorithm only on a small fraction of those instances, learn from the obtained solutions a predicted solution, and feed it to a much faster learning-augmented algorithm in order to solve the remaining majority of instances.

In the following lemma (proved in Section A) we show that the problem complementary to Knapsack admits an $O(n \log n)$-time 2-approximation algorithm. Theorem 3 then implies the existence of an $O(n \log n)$-time learning-augmented algorithm for Knapsack with approximation ratio $1 - (\eta^+ + \eta^-)/\mathrm{OPT}$.

Lemma 1. *There is an $O(n \log n)$-time 2-approximation algorithm for the following problem: Given n items, the i-th of which has size s_i and is worth w_i, and the target t, find a subset of items of total size at least t that minimizes the total worth.*

4 REFINED BOUNDS FOR STEINER TREE

We describe a refined algorithm for Steiner Tree based on a 2-approximation algorithm by Mehlhorn (1988). Our careful analysis describes how it detects and avoids false positives with high weight. Our algorithm uses a parameter $\alpha \geqslant 1$ to adapt its treatment of the prediction. With $\alpha = 1$ its behavior copies Mehlhorn's algorithm, and α approaching infinity corresponds to the algorithm in Theorem 1. However, a different choice of α may give better results. We illustrate this by an example and show how to achieve performance close to the best choice of α with only a constant factor increase in running time in Section B.1.

The algorithm receives as input a graph $G = (V, E)$, set $T \subseteq V$ of k terminals, a weight function $w : E \to \mathbb{R}_{\geqslant 0}$, and a set $\widehat{X} \subseteq E$ of predicted edges. First, it scales down the weight of edges in $\widehat{X}$ by dividing them by the parameter α. Then, it uses the algorithm of Mehlhorn (1988) to compute a minimum spanning tree MST_α of the metric closure of the terminals with respect to the scaled edge weights. In the end, the algorithm outputs the union of edges contained in paths corresponding to the connections in MST_α. The algorithm is summarized in Algorithm 1.

Proposition 1 (Mehlhorn (1988))**.** *The minimum spanning tree of the metric closure $\left(T, \binom{T}{2}\right)$ of the graph G can be computed in time near-linear in the size of G.*

Algorithm 1: Steiner tree with predictions

Parameter: $\alpha \geqslant 1$;
foreach $e \in E$ **do**
  if $e \in \widehat{X}$ **then** $w_\alpha(e) := w(e)/\alpha$;
  else $w_\alpha(e) := w(e)$;
Compute MST_α of the metric closure $\mathbf{G} = (T, \binom{T}{2})$ of G with weights w_α using Proposition 1;
$X := \emptyset$;
foreach edge $e = \{t_1, t_2\}$ **in** MST_α **do**
  Choose $p(e) \subseteq E$ the cheapest path in G from t_1 to t_2 with respect to w_α;
  $X := X \cup p(e)$;
return X;

4.1 Analysis

Recall that $G = (V, E)$ denotes the input graph, and let $\mathbf{G} = (T, \binom{T}{2})$ denote the complete graph on the terminals $T \subseteq V$. For any weight function $w : E \to \mathbb{R}_{\geqslant 0}$, consider the shortest path metric on terminals induced by w, i.e., for an edge e in $\mathbf{G}$ between t_1 and t_2, its cost $\mathbf{c}(e)$ is equal to the length of the shortest path $p(e)$, with respect to w, from t_1 to t_2 in G. There is a natural correspondence between an edge e (the cost $\mathbf{c}(e)$) in $\mathbf{G}$ and the set of edges $p(e) \subseteq E$ in G (weight $w(p(e))$). Let MST denote a minimum spanning tree on $\mathbf{G}$. Our algorithm satisfies the following performance bound.

Theorem 5. *Consider a graph $G = (V, E)$ with edge weights $w\colon E \to \mathbb{R}_{\geqslant 0}$ and a set $T \subseteq V$ of k terminals. Let $X' \subseteq E$ be any Steiner tree on G and S be a set of $\min\{k-1, |\widehat{X} \setminus X'|\}$ edges with the highest cost contained in MST. Our refined algorithm outputs a Steiner tree X of total weight*

$$w(X) \leqslant \left(1 + \frac{1}{\alpha}\right) w(X') + \left(1 - \frac{1}{\alpha}\right) \eta^- + \min\left\{\eta^+, (\alpha - 1) \cdot \sum_{e \in S} \mathbf{c}(e)\right\}, \tag{1}$$

where (η^+, η^-) is the prediction error of $\widehat{X}$ with respect to X'.

With $X' = X^*$ being an optimal solution of cost OPT and $\alpha = 1$, the bound in (1) is equal to $2\mathrm{OPT}$, which corresponds to the conventional algorithm which ignores the predictions. With α approaching infinity, its limit is $\mathrm{OPT} + \eta^+ + \eta^-$, matching the result from Theorem 1. However, it can be much better than both in case of $\widehat{X} \setminus X^*$ containing a small number of edges of very high weight. Compared to Theorem 1, η^+ in (1) is capped by $(\alpha - 1) \sum_{e \in S} \mathbf{c}(e)$ where $\sum_{e \in S} \mathbf{c}(e) \leqslant \mathbf{c}(\mathrm{MST}) \leqslant 2w(X^*)$, since Mehlhorn's algorithm is a 2-approximation algorithm for Minimum Steiner Tree. Since we always have $\eta^- \leqslant w(X^*)$, (1) shows that Algorithm 1 is a 2α-approximation algorithm regardless of the prediction error.

We defer the rest of the analysis to Section B.

5 Discussion

We initiated the study of algorithms with predictions with a focus on improving over the approximation guarantees of classic algorithms without increasing the running time. This paper focused on the wide and important class of selection problems, but it would be interesting to investigate whether similar results can be obtained for central combinatorial optimization problems that do not belong to this class, e.g., clustering and scheduling problems or problems with non-linear (e.g., submodular) objectives.

We demonstrated, with the example of the Steiner Tree problem, that refined algorithms with improved guarantees are possible for specific problems. A second, implicit, advantage of our refined Steiner Tree algorithm is that its actual performance could be bounded in terms of a quantity directly related to the optimal cost, thus avoiding the additional robustification step. An interesting direction for further research would be to identify other problems satisfying these two properties.

REFERENCES

Sungsoo Ahn, Younggyo Seo, and Jinwoo Shin. Learning what to defer for maximum independent sets. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 134–144. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/ahn20a.html.

Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *J. Comput. Syst. Sci.*, 58(1):193–210, 1999. doi: 10.1006/JCSS.1998.1605. Announced at STOC 1995.

Yossi Azar, Stefano Leonardi, and Noam Touitou. Flow time scheduling with uncertain processing time. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1070–1080. ACM, 2021. doi: 10.1145/3406325.3451023.

Yossi Azar, Stefano Leonardi, and Noam Touitou. Distortion-oblivious algorithms for minimizing flow time. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 252–274. SIAM, 2022. doi: 10.1137/1.9781611977073.13.

Xingjian Bai and Christian Coester. Sorting with predictions. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/544696ef4847c903376ed6ec58f3a703-Abstract-Conference.html.

Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. Learning augmented energy minimization via speed scaling. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/af94ed0d6f5acc95f97170e3685f16c0-Abstract.html.

Evripidis Bampis, Bruno Escoffier, and Michalis Xefteris. Parsimonious learning-augmented approximations for dense instances of NP-hard problems. *CoRR*, abs/2402.02062, 2024. doi: 10.48550/ARXIV.2402.02062.

Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algorithms*, 2(2):198–203, 1981. doi: 10.1016/0196-6774(81)90020-1.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2020.07.063.

Giulia Bernardini, Alexander Lindermayr, Alberto Marchetti-Spaccamela, Nicole Megow, Leen Stougie, and Michelle Sweering. A universal error measure for input predictions applied to online graph problems. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/15212bd2265c4a3ab0dbc1b1982c1b69-Abstract-Conference.html.

Édouard Bonnet and Florian Sikora. The PACE 2018 parameterized algorithms and computational experiments challenge: The third iteration. In *13th International Symposium on Parameterized and Exact Computation, IPEC 2018*, volume 115 of *LIPIcs*, pages 26:1–26:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi: 10.4230/LIPICS.IPEC.2018.26. https://pacechallenge.org/2018/steiner-tree/.

Vladimir Braverman, Prathamesh Dharangutte, Vihan Shah, and Chen Wang. Learning-augmented maximum independent set. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2024. URL https://arxiv.org/abs/2407.11364.

Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013. doi: 10.1145/2432622.2432628.

Timothy M. Chan. Approximation schemes for 0-1 knapsack. In *1st Symposium on Simplicity in Algorithms, SOSA 2018*, volume 61 of *OASIcs*, pages 5:1–5:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi: 10.4230/OASICS.SOSA.2018.5.

Justin Y. Chen, Sandeep Silwal, Ali Vakilian, and Fred Zhang. Faster fundamental graph algorithms via learned predictions. In *International Conference on Machine Learning, ICML 2022*, volume 162 of *Proceedings of Machine Learning Research*, pages 3583–3602. PMLR, 2022. URL https://proceedings.mlr.press/v162/chen22v.html.

Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. A nearly quadratic-time FPTAS for knapsack. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024*, pages 283–294. ACM, 2024. doi: 10.1145/3618260.3649730.

Miroslav Chlebík and Janka Chlebíková. The Steiner tree problem on graphs: Inapproximability results. *Theor. Comput. Sci.*, 406(3):207–214, 2008. doi: 10.1016/J.TCS.2008.06.046.

Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Oper. Res. Forum*, 3(1), 2022. doi: 10.1007/S43069-021-00101-Z. Announced as a technical report in 1976.

Vincent Cohen-Addad, Tommaso d'Orsi, Anupam Gupta, Euiwoong Lee, and Debmalya Panigrahi. Learning-augmented approximation algorithms for maximum cut and related problems. In *Proceedings of NeurIPS 2024 (in press)*, 2024. doi: 10.48550/ARXIV.2402.18263.

George B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–288, 1957. doi: 10.1287/opre.5.2.266.

Sami Davies, Benjamin Moseley, Sergei Vassilvitskii, and Yuyan Wang. Predictive flows for faster Ford-Fulkerson. In *International Conference on Machine Learning, ICML 2023*, volume 202 of *Proceedings of Machine Learning Research*, pages 7231–7248. PMLR, 2023. URL https://proceedings.mlr.press/v202/davies23b.html.

Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pages 10393–10406, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/5616060fb8ae85d93f334e7267307664-Abstract.html.

Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Algorithms with prediction portfolios. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/7f9220f90cc85b0da693643add6618e6-Abstract-Conference.html.

Paul Dütting, Silvio Lattanzi, Renato Paes Leme, and Sergei Vassilvitskii. Secretaries with advice. *Math. Oper. Res.*, 49(2):856–879, 2024. doi: 10.1287/MOOR.2023.1384. Announced at EC 2021.

Jack Edmonds. Maximum matching and a polyhedron with $0,1$-vertices. *J. of Res. the Nat. Bureau of Standards*, 69 B:125–130, 1965.

Jon C. Ergun, Zhili Feng, Sandeep Silwal, David P. Woodruff, and Samson Zhou. Learning-augmented k-means clustering. In *The Tenth International Conference on Learning Representations, ICLR 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=X8cLTHexYyY.

Willem Feijen and Guido Schäfer. Using machine learning predictions to speed-up Dijkstra's shortest path algorithm. *CoRR*, abs/2112.11927, 2021. URL https://arxiv.org/abs/2112.11927.

Harold N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443. SIAM, 1990. URL http://dl.acm.org/citation.cfm?id=320176.320229.

Buddhima Gamlath, Silvio Lattanzi, Ashkan Norouzi-Fard, and Ola Svensson. Approximate cluster recovery from noisy labels. In *Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 1463–1509. PMLR, 2022. URL https://proceedings.mlr.press/v178/gamlath22a.html.

Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. doi: 10.1137/S0097539793242618. Announced at SODA 1992.

Johan Håstad. Clique is hard to approximate within $n^{1\text{-epsilon}}$. *Acta Mathematica*, 182(1):105–142, 1999. doi: 10.1007/BF02392825. Announced at FOCS 1996.

Piotr Indyk, Frederik Mallmann-Trenn, Slobodan Mitrovic, and Ronitt Rubinfeld. Online page migration with ML advice. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2022*, volume 151 of *Proceedings of Machine Learning Research*, pages 1655–1670. PMLR, 2022. URL https://proceedings.mlr.press/v151/indyk22a.html.

Chaitanya K. Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem, 2019. URL https://arxiv.org/abs/1906.01227.

Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi: 10.1007/978-1-4684-2001-2_9.

Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. doi: 10.1016/J.JCSS.2007.06.019.

T. Koch, A. Martin, and S. Voß. SteinLib: An updated library on Steiner tree problems in graphs. Technical Report ZIB-Report 00-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, Berlin, 2000. URL https://steinlib.zib.de/.

Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 5 edition, 2012. doi: 10.1007/978-3-642-24488-9.

Lawrence T. Kou, George Markowsky, and Leonard Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15:141–145, 1981. doi: 10.1007/BF00288961.

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, SIGMOD Conference 2018*, pages 489–504. ACM, 2018. doi: 10.1145/3183713.3196909.

Silvio Lattanzi, Ola Svensson, and Sergei Vassilvitskii. Speeding up Bellman Ford via minimum violation permutations. In *International Conference on Machine Learning, ICML 2023*, volume 202 of *Proceedings of Machine Learning Research*, pages 18584–18598. PMLR, 2023. URL https://proceedings.mlr.press/v202/lattanzi23a.html.

Alexander Lindermayr and Nicole Megow. Algorithms with predictions. https://algorithms-with-predictions.github.io, 2022. Accessed 22 May 2024.

Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4):24:1–24:25, 2021. doi: 10.1145/3447579. Announced at ICML 2018.

Xiao Mao. (1 - ϵ)-approximation of knapsack in nearly quadratic time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024*, pages 295–306. ACM, 2024. doi: 10.1145/3618260.3649677.

Kurt Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Inf. Process. Lett.*, 27(3):125–128, 1988. doi: 10.1016/0020-0190(88)90066-X.

Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 646–662. Cambridge University Press, 2020. doi: 10.1017/9781108637435.037.

Thy Dinh Nguyen, Anamay Chaturvedi, and Huy L. Nguyen. Improved learning-augmented algorithms for k-means and k-medians clustering. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=dCSFiAl_VO3.

Adam Polak and Maksym Zub. Learning-augmented maximum flow. *Information Processing Letters*, 186:106487, 2024. ISSN 0020-0190. doi: 10.1016/j.ipl.2024.106487.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 9684–9693, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html.

Emmanuel Romero Ruiz, Emmanuel Antonio Cuevas, Irwin Enrique Villalobos López, and Carlos Segura González. Source code of the CIMAT solver for the Steiner tree problem. https://github.com/HeathcliffAC/SteinerTreeProblem, 2018. Accessed 22 May 2024.

Shinsaku Sakaue and Taihei Oki. Discrete-convex-analysis-based framework for warm-starting algorithms with predictions. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/844e61124d9e1f58632bf0c8968ad728-Abstract-Conference.html.

Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 9367–9376. PMLR, 2020. URL http://proceedings.mlr.press/v119/tang20a.html.

David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. ISBN 978-0-521-19527-0.

A PROBLEM COMPLEMENTARY TO KNAPSACK

Lemma 1. *There is an $O(n \log n)$-time 2-approximation algorithm for the following problem: Given n items, the i-th of which has size s_i and is worth w_i, and the target t, find a subset of items of total size at least t that minimizes the total worth.*

Algorithm 2: Approximation algorithm for the problem complementary to Knapsack

```
Sort items in the increasing order of w_i/s_i;
X := ∅;
C := ∅;
for i = 1, ..., n do
    if s(X) + s_i < t then
        X := X ∪ {i};
    else
        C := C ∪ {X};
return arg min{w(Y) | Y ∈ C};
```

Proof. The algorithm (see Algorithm 2) maintains an initially empty partial solution X, and keeps adding items to it in a greedy manner in the increasing order of the worth-to-size ratio $\frac{w_i}{s_i}$. Whenever adding the next item i to X would make the total size of X meet or exceed the target size t, the algorithm skips adding that item to X and instead adds $X \cup \{i\}$ to the set of candidate solutions $\mathcal{C}$. At the end, the algorithm returns the cheapest solution out of the candidate solutions $\mathcal{C}$.

To show that this is indeed a 2-approximation algorithm, let $\mathrm{OPT} \subseteq [n]$ denote an optimal solution, and for every $i \in \{0, 1, \ldots, n\}$ let X_i denote the set X after the algorithm considered the first i items. It must be that $\mathrm{OPT} \setminus X_n \neq \emptyset$, because $s(X_n) < t \leqslant s(\mathrm{OPT})$. Consider $i = \min(\mathrm{OPT} \setminus X_n)$, recalling that the items are numbered in an increasing order of the worth-to-size ratios. Since $i \notin X_i \subseteq X_n$, it must hold that $s(X_{i-1}) + s_i \geqslant t$, and thus $X_{i-1} \cup \{i\} \in \mathcal{C}$. By the minimality of i, we know that $\mathrm{OPT} \cap \{1, \ldots, i-1\} \subseteq X_{i-1}$, and thus each item in $X_{i-1} \setminus \mathrm{OPT}$ has smaller worth-to-size ratio than any item in $\mathrm{OPT} \setminus X_{i-1}$. This, together with the fact that $s(X_{i-1}) < t \leqslant s(\mathrm{OPT})$ and hence also $s(X_{i-1} \setminus \mathrm{OPT}) < s(\mathrm{OPT} \setminus X_{i-1})$, implies that $w(X_{i-1} \setminus \mathrm{OPT}) < w(\mathrm{OPT} \setminus X_{i-1})$ and as a consequence $w(X_{i-1}) < w(\mathrm{OPT})$. Clearly also $w_i \leqslant w(\mathrm{OPT})$, so $\min\{w(Y) \mid Y \in \mathcal{C}\} \leqslant w(X_{i-1} \cup \{i\}) = w(X_{i-1}) + w_i < 2 \cdot \mathrm{OPT}$. $\square$

B FULL ANALYSIS OF THE REFINED ALGORITHM FOR STEINER TREE

A crucial property of Algorithm 1 is that it never buys an edge with weight more than α times larger than some connection in MST. The sum in (1) needs to be over connections in MST instead of individual edges in X', since X' may consist of paths containing large number of edges of very small length.

The key part of the proof of Theorem 5 is the analysis of how edges in $\widehat{X} \cap X^*$ and edges in $\widehat{X} \setminus X^*$ respectively influence MST_α found by Algorithm 1, depending on the parameter α. The improvement in the approximation ratio then comes from the short-cutting procedure on the graph with scaled weights which identifies paths over edges which are useful for connecting a higher number of terminals.

Consider a fixed Steiner tree $X' \subseteq E$. For the purpose of analysis, we define another weight function w'_α. We set

$$w'_\alpha(e) := \begin{cases} w(e)/\alpha & \text{if } e \in \widehat{X} \cap X', \\ w(e) & \text{otherwise.} \end{cases}$$

Denote by MST'_α a minimum spanning tree in $\mathbf{G}$ with respect to $\mathbf{c}'_\alpha$, i.e., the metric closure of w'_α. The following observation holds.

Observation 1. *The cost of the connections in* MST'_α *can be bounded as*

$$\mathbf{c}'_\alpha(\mathrm{MST}'_\alpha) \leqslant 2w(X') - 2(1 - 1/\alpha)w(\widehat{X} \cap X'). \tag{2}$$

Proof. In the graph (V, X'), replace each edge $uv \in X'$ by two directed edges (u, v) and (v, u), each of weight $w(uv)$. This way, each vertex has the same in-degree as out-degree and hence, by Euler's theorem (Korte and Vygen, 2012, Theorem 2.24), there is a tour P using each directed edge exactly once. Since X' is a Steiner tree, this tour visits all the terminals and naturally defines a spanning tree on $\mathbf{G}$ of cost at most $w(P)$: Whenever P visits a new terminal t, we add an edge $tt' \in \mathbf{G}$ of cost $\mathbf{c}(tt') = w(p(tt'))$, where t' is the preceding terminal visited by P and $p(tt')$ is the segment of P connecting t' and t.

The weight of the tour P is $2w(X')$ with respect to w and $2w(X') - (1-1/\alpha)w(\widehat{X} \cap X')$ with respect to w'_α, since every edge $e \in \widehat{X} \cap X'$ is used twice and has cost $w'_\alpha(e) = w(e) - (w(e) - w(e)/\alpha)$. Therefore, we have $\mathbf{c}(\mathrm{MST}'_\alpha) \leqslant 2w(X') - (1 - 1/\alpha)w(\widehat{X} \cap X')$. □

The following basic fact about spanning trees follows from the exchange property of matroids which states that for any two spanning trees T_1 and T_2 and any $e \in T_1 \setminus T_2$, there is $e' \in T_2 \setminus T_1$ such that $(T_1 \setminus \{e\}) \cup \{e'\}$ is a spanning tree. For a proof of the exchange property, see for instance Korte and Vygen (2012, Theorem 14.7).

Proposition 2. *Consider a minimum spanning tree T on a graph G with cost function $c\colon E \to \mathbb{R}_{\geqslant 0}$ and an arbitrary spanning tree T' on the same graph. There exists a bijection $\phi\colon T \to T'$ such that $c(e) \leqslant c(\phi(e))$ for each edge $e \in T$.*

Proof of the following lemma contains the key part of our analysis. It uses Proposition 2 to charge each edge in $\widehat{X} \setminus X'$ to a single connection in MST'_α.

Lemma 2. *Let S' be a set of $\min\{k-1, |\widehat{X} \setminus X'|\}$ edges with highest cost $\mathbf{c}'_\alpha$ in MST'_α. We have*

$$w'_\alpha(X) \leqslant \mathbf{c}'_\alpha(\mathrm{MST}'_\alpha) + \min\left\{\eta^+, (\alpha - 1)\sum_{e' \in S'} \mathbf{c}'_\alpha(e')\right\}.$$

Proof. Denote by $e'_1, e'_2, \ldots, e'_{k-1}$ the edges of MST'_α and by $e_1, e_2, \ldots, e_{k-1}$ the edges of MST_α. Since MST_α is a minimum spanning tree with respect to $c_\alpha(\cdot)$ and up to reordering the edges, by Proposition 2, we can assume that

$$c_\alpha(e_i) \leqslant c_\alpha(e'_i), \quad \forall i \in \{1, \ldots, k-1\}.$$

Let $F_{\leqslant 0} := \emptyset$ and set $F_{\leqslant i} := F_{\leqslant i-1} \cup (p(e_i) \setminus F_{\leqslant i-1})$, for $i = \{1, \ldots, k-1\}$, so that $F = F_{k-1}$.

To show the lemma, for each pair e_i, e'_i, we distinguish between two cases:

- $\underline{(p(e_i) \setminus F_{\leqslant i-1}) \cap (\hat{F} \setminus F^*) = \emptyset}$: In this case, no weights of edges on $(p(e_i) \setminus F_{i-1}) \cap (\hat{F} \setminus F^*)$ have been scaled down, hence
$$c'_\alpha(p(e_i) \setminus F_{\leqslant i-1}) = c_\alpha(p(e_i) \setminus F_{i-1}) \leqslant c_\alpha(e'_i) \leqslant c'_\alpha(e'_i).$$
- $\underline{(p(e_i) \setminus F_{\leqslant i-1}) \cap (\hat{F} \setminus F^*) \neq \emptyset}$: In this case, the actual cost of the path corresponding to edge e_i minus the edges in $F_{\leqslant i-1}$ already bought, i.e. $p(e_i) \setminus F_{\leqslant i-1}$, can be at most α times higher:
$$c'_\alpha(p(e_i) \setminus F_{\leqslant i-1}) \leqslant \alpha \cdot c_\alpha(e_i) \leqslant \alpha \cdot c_\alpha(e'_i) \leqslant c'_\alpha(e'_i) + (\alpha - 1) \cdot c'_\alpha(e'_i).$$
At the same time, we have
$$c'_\alpha(p(e_i) \setminus F_{\leqslant i-1}) \leq c_\alpha(e_i) + c(\hat{F} \cap (p(e_i) \setminus F_{\leqslant i-1})) \leqslant c'_\alpha(e'_i) + c(\hat{F} \cap (p(e_i) \setminus F_{\leqslant i-1})).$$

Since the latter case only happens once for each edge in $\hat{F} \setminus F^*$, we see that
$$c'_\alpha(F) \leqslant \underbrace{\sum_{i=1}^{k-1} c'_\alpha(e'_i)}_{c'_\alpha(\mathrm{MST}'_\alpha)} + (\alpha - 1) \max_{S \subseteq [k] : |S| = |\hat{F} \setminus F^*|} \sum_{i \in S} c'_\alpha(e'_i),$$
and at the same time, we have $c'_\alpha(F) \leqslant \mathrm{MST}'_\alpha + c(\hat{F})$. The lemma follows. □

Proof of Theorem 5. Since for all edges $e \in \widehat{X} \cap X'$ we have $w'_\alpha(e) = w(e)/\alpha$, we can write $w(e) = w'_\alpha(e) + (1 - 1/\alpha)w(e)$. Therefore, we have
$$w(X) \leqslant w'_\alpha(X) + (1 - 1/\alpha) \cdot w(\widehat{X} \cap X').$$
Combining this bound with Lemma 2 and (2), we get
$$w(X) \leqslant 2w(X') - (1 - 1/\alpha)w(\widehat{X} \cap X') + \min\left\{\eta^+, (\alpha - 1) \sum_{e' \in S'} \mathbf{c}'_\alpha(e')\right\}.$$
Proposition 2 implies that $\sum_{e' \in S'} c'_\alpha(e') \leq \sum_{e \in S} c(e)$, since MST'_α is a minimum spanning tree on $\mathbf{G}$ with respect to cost c'_α and MST is a spanning tree on the same graph. Now, it is enough to note that
$$2w(X') - (1 - 1/\alpha)w(\widehat{X} \cap X') = (1 + 1/\alpha)w(X') + (1 - 1/\alpha)w(X' \setminus \widehat{X}).$$ □

B.1 Approximating the best alpha

Let α^* be the parameter of Algorithm 1 which minimizes the upper bound in Theorem 5. We show how to find α which achieves a bound at most $(1 + \epsilon)$ times worse than the bound with α^*.

Algorithm 3: Steiner tree: search for the best α

for $i = 0, \ldots, \lceil \log_{1+\epsilon} \epsilon^{-1} \rceil$ **do**
  run Algorithm 1 with $\alpha = (1 + \epsilon)^i$;
output the best solution found during above iterations;

Theorem 6. *For a constant $\epsilon > 0$, Algorithm 3 runs in near-linear time and finds a Steiner tree X with weight*
$$w(X) \leqslant (1 + \epsilon) \min_{\alpha \geqslant 1} \left\{ \left(1 + \frac{1}{\alpha}\right) w(X') + \left(1 - \frac{1}{\alpha}\right) \eta^- + \min\left\{\eta^+, (\alpha - 1) \cdot \sum_{e \in S} \mathbf{c}(e)\right\}\right\}.$$

Proof. Algorithm 3 performs a constant number $\lceil \log_{1+\epsilon} \epsilon^{-1} \rceil + 1$ of iterations of Algorithm 1, which runs in near-linear time.

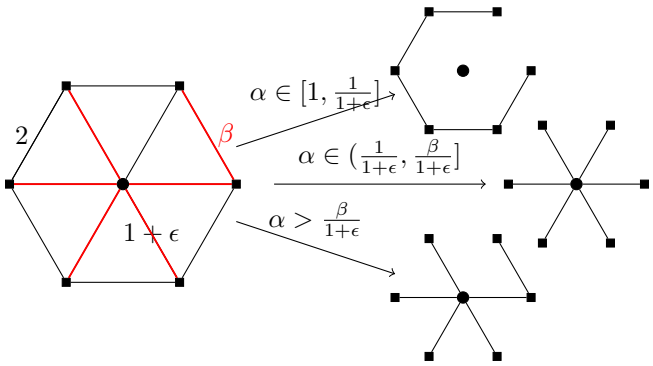


Figure 1: Steiner tree problem with $k = n - 1$ terminals (square) and one non-terminal vertex (circle). Outer edges have weight 2, except one edge of weight β. Inner edges have weight $1 + \epsilon$. Red edges are predicted. On the right are the respective Steiner trees output by Algorithm 1 with different parameters of α.

First, note that we need to consider only $\alpha \leqslant \epsilon^{-1}$, because

$$(1+\epsilon)c(X') + (1-\epsilon)\eta^- + \min\left\{\eta^+, (\epsilon^{-1}-1)\cdot\sum_{e\in S}\mathbf{c}(e)\right\}\Big) \leqslant (1+\epsilon)f(\alpha)$$

for any $\alpha > \epsilon^{-1}$.

It is enough to show that, for any α, we have $f((1+\epsilon)\alpha) \in [(1+5\epsilon)^{-1}f(\alpha), (1+5\epsilon)f(\alpha)]$. We show this for every term of $f(\alpha)$ separately. We have

$$(1+\frac{1}{\alpha})c(X') \geqslant \left(1+\frac{1}{(1+\epsilon)\alpha}\right)c(X') = \frac{(1+\epsilon)\alpha+1}{(1+\epsilon)\alpha}c(X') \geqslant (1+\epsilon)^{-1}(1+\frac{1}{\alpha})c(X').$$

Similarly, we have

$$(1-\frac{1}{\alpha})\eta^- \leqslant \left(1-\frac{1}{(1+\epsilon)\alpha}\right)\eta^- \leqslant \left(\frac{(1+\epsilon)\alpha-1}{(1+\epsilon)\alpha}\right)\eta^- \leqslant (1+\epsilon)\left(\frac{\alpha-1}{(1+\epsilon)\alpha}\right)\eta^-,$$

which is at most $(1+\epsilon)(1-1/\alpha)\eta^-$. For the last term, we have

$$\alpha - 1 \leqslant (1+\epsilon)\alpha - 1 = \alpha - 1 + \epsilon\alpha \leqslant (1+2\epsilon)(\alpha-1)$$

if $\alpha \geqslant 2$. If $\alpha < 2$, we have $\epsilon\alpha\sum_{e\in S}\mathbf{c}(e_i) \leqslant 4\epsilon c(X')$. □

B.2 Tight example for our analysis

Figure 1 describes an instance with $k = n - 1$ terminals and a prediction with a single false-negative edge of weight $1 + \epsilon$ and a single false-positive edge of weight $\beta > 2$. Algorithm 1 with $\alpha \in [1, (1+\epsilon)^{-1})$ achieves approximation ratio approaching 2 as n increases. With $\alpha \in (\frac{1}{1+\epsilon}, \frac{\beta}{1+\epsilon})$, its approximation ratio is equal to 1. With $\alpha > \frac{\beta}{1+\epsilon}$, it approaches $\frac{\mathrm{OPT}+\beta}{\mathrm{OPT}}$, which is equal to 2 if we choose $\beta = \mathrm{OPT}$. This shows that the best choice of α may not be 1 nor approaching ∞.

We can use this construction to show that the coefficient of $\sum_{e\in S}\mathbf{c}(e)$ in (1) is tight for the given algorithm. First consider the input graph in Figure 1 with $\beta = 2n$. Algorithm 1 with $\alpha = \beta$ receiving a prediction $\widehat{X}$ containing all red edges, achieves cost $(n-1)(1+\epsilon) + \beta \geqslant \mathrm{OPT} + \frac{\alpha}{2}2 - (1+\epsilon)$, where 2 is the cost of the most expensive connection in the minimum spanning tree on the metric closure of the terminals. Note that all terms except for $-(1+\epsilon)$ go to infinity as n increases.

Similarly, we can show the tightness of the coefficient of η^-. Algorithm 1 with $\widehat{X} = \emptyset$ achieves cost $2n \geqslant (1+\epsilon)^{-1}\big((1+1/\alpha)\mathrm{OPT} + \eta^-$, since $\eta^- = \mathrm{OPT} = n(1+\epsilon)$.

C LOWER BOUNDS

C.1 PRELIMINARIES

In order to obtain lower bounds for our setting we use the following results of Khot and Regev (2008).

Proposition 3 (Khot and Regev (2008))**.** *Assuming UGC, for every constant $\delta > 0$, it is NP-hard to distinguish, for an input n-vertex graph G, between the following two cases:*

- *(YES) G contains an independent set of size at least $\frac{n}{2} - \delta n$.*
- *(NO) G contains no independent set of size at least δn.*

It directly implies the following corollary for the Minimum Vertex Cover problem:

Corollary 7 (Khot and Regev (2008))**.** *Assuming UGC, for every constant $\delta > 0$ it is NP-hard to distinguish, for an input n-vertex graph G, between the following two cases:*

- *(YES) G has a vertex cover of size at most $\frac{n}{2} + \delta n$.*
- *(NO) G has no vertex cover of size at most $(1 - \delta)n$.*

Note that Corollary 7 implies that, assuming UGC and P≠NP, Minimum Vertex Cover cannot be approximated within a factor of $2 - \epsilon$, for any $\epsilon > 0$.

C.2 LOWER BOUND FOR A MINIMIZATION PROBLEM

We now argue that our Theorem 1 for minimization selection problems cannot be improved. For a specific minimization selection problem, namely the Minimum Vertex Cover problem, which has a folklore 2-approximation algorithm, Theorem 1 and Corollary 2 with $\rho = 2$ imply existence of a learning-augmented algorithm with approximation ratio $\min\{1 + \frac{\eta^+ + \eta}{\mathrm{OPT}}, 2\}$. We show that this is the best possible upper bound.

Theorem 8. *Let A be a polynomial-time learning-augmented approximation algorithm for Minimum Vertex Cover that guarantees an approximation ratio of at most*

$$1 + f\left(\frac{\eta^+}{\mathrm{OPT}}, \frac{\eta^-}{\mathrm{OPT}}\right),$$

for some non-decreasing function f. Then, assuming UGC, we have

$$f(x, y) \geqslant x + y,$$

for any choice of $x, y \geqslant 0$ satisfying $x + y \leqslant 1$.

Proof. By contradiction, consider x, y such that $f(x, y) < x + y$. We denote $\epsilon := (x + y) - f(x, y)$, and let $\delta := \epsilon/20$. We will show how to use the hypothesized learning-augmented algorithm A to construct an algorithm $\bar{A}$ for the (prediction-less) Minimum Vertex Cover problem that can distinguish between instances on n vertices with optimum value more than $(1 - \delta)n$ and instances with optimum value at most $(1 + \delta)\frac{n}{2}$. Such an algorithm would contradict Corollary 7. In order to provide a cleaner exposition, we first assume that A can actually solve *Min-Weight* Vertex Cover with the given approximation ratio. This assumption can be removed easily, as discussed at the end of the proof.

We construct algorithm $\bar{A}$ that, for any input graph G with minimum vertex cover X^* of size $|X^*| \in \left[(1 - \delta)\frac{n}{2}, (1 + \delta)\frac{n}{2}\right]$, produces a solution X such that $|X| < (1 - \delta)n$, certifying that G does not belong to the NO case. Note that for graphs with vertex cover of size $< (1 - \delta)\frac{n}{2}$ such solution can be easily produced by the folklore 2-approximation algorithm.

Given the input graph G, $\bar{A}$ produces a weighted graph $\bar{G}$ and a prediction $\hat{X}$, and runs A on $\bar{G}$ with $\hat{X}$. Graph $\bar{G}$ is a disjoint union of three graphs: G_0, G_+, and G_-. Let $x' := \frac{1-\delta}{1+\delta}x$ and $y' := \frac{1-\delta}{1+\delta}y$.

- G_0 is a complete graph on $\frac{n}{2} + 1$ vertices, we have $\bar{w}(v) = (1 - x' - y')$ for each $v \in V(G_0)$.
- G_+ is a copy of G with scaled weights, we have $\bar{w}(v) = x'$ for each $v \in V(G_+)$.

- G_- is a copy of G with scaled weights, we have $\bar{w}(v) = y'$ for each $v \in V(G_-)$.

The prediction $\hat{X}$ contains arbitrary $\frac{n}{2}$ vertices from G_0, all vertices of $V(G_+)$ and no vertices of $V(G_-)$. Receiving a solution $\bar{X}$ found by the algorithm A on $\bar{G}$, we denote $X_+ := \bar{X} \cap V(G_+)$ and $X_- := \bar{X} \cap V(G_-)$. Algorithm $\bar{A}$ returns X which is either X_+ or X_-, choosing the one with the smaller size. We now prove that $|X| < (1-\delta)n$ if G is the YES case.

First, we show that $\eta^+/\mathrm{OPT} \leqslant x$ and $\eta^-/\mathrm{OPT} \leqslant y$. Let OPT be the value of an optimal solution for $\bar{G}$. We have $\mathrm{OPT} = (1-x-y)\frac{n}{2} + x|X^*| + y|X^*|$ which belongs to $\left[(1-\delta)\frac{n}{2}, (1+\delta)\frac{n}{2}\right]$ by the assumption about the size of X^*. For the error, we have

$$\eta^+ = x' \cdot (n - |X^*|) = \frac{1-\delta}{1+\delta} x \cdot (n - |X^*|), \quad \text{and}$$
$$\eta^- = y' \cdot |X^*| = \frac{1-\delta}{1+\delta} y \cdot |X^*|.$$

Since both $(n - |X^*|)$ and $|X^*|$ are at most $(1+\delta)\frac{n}{2}$ and $\mathrm{OPT} \geqslant (1-\delta)\frac{n}{2}$, we have $\eta^+/\mathrm{OPT} \leqslant x$ and $\eta^-/\mathrm{OPT} \leqslant y$. By monotonicity of f, this implies that $\bar{X}$ is a $(1+x+y-\epsilon)$-approximate solution. Therefore, we have

$$\begin{aligned}
\bar{w}(\bar{X}) &\leqslant (1+x+y-\epsilon) \cdot \mathrm{OPT} \\
&\leqslant (1+x+y) \cdot (1+\delta)\frac{n}{2} - \epsilon \cdot (1-\delta)\frac{n}{2} \\
&\leqslant \frac{n}{2} + (x+y)\frac{n}{2} + (1+x+y+\epsilon)\delta\frac{n}{2} - \epsilon\frac{n}{2} \\
&\leqslant \frac{n}{2} + (x+y)\frac{n}{2} + 4\delta\frac{n}{2} - \epsilon\frac{n}{2} \\
&\leqslant \frac{n}{2} + (x+y)\frac{n}{2} - 0.8\epsilon\frac{n}{2},
\end{aligned} \tag{3}$$

by our choice of $\delta = \epsilon/20$. On the other hand, we can write

$$\begin{aligned}
\bar{w}(\bar{X}) &\geqslant (1-x'-y')\frac{n}{2} + x'|X_+| + y'|X_-| \\
&= \frac{n}{2} + x'\big(|X_+| - \frac{n}{2}\big) + y'\big(|X_-| - \frac{n}{2}\big) \\
&\geqslant \frac{n}{2} + x\big(|X_+| - \frac{n}{2}\big) + y\big(|X_-| - \frac{n}{2}\big) - 4\delta\frac{n}{2} \\
&\geqslant \frac{n}{2} + x\big(|X_+| - \frac{n}{2}\big) + y\big(|X_-| - \frac{n}{2}\big) - 0.2\epsilon\frac{n}{2}.
\end{aligned} \tag{4}$$

In the second to last step, we use the definition of x' and y', and the fact that $\frac{1-\delta}{1+\delta} \geqslant 1 - 2\delta$. Inequalities (3) and (4) imply that at least one of $|X_+| - \frac{n}{2}$ and $|X_-| - \frac{n}{2}$ must be less than or equal to $\frac{n}{2} - 0.6\epsilon\frac{n}{2} < \frac{n}{2} - \delta n$. Indeed, otherwise we would have

$$\bar{w}(\bar{X}) > \frac{n}{2} + (x+y)\frac{n}{2} - (x+y)0.6\epsilon\frac{n}{2} - 0.2\epsilon\frac{n}{2} \geqslant \frac{n}{2} + (x+y)\frac{n}{2} - 0.8\epsilon\frac{n}{2},$$

contradicting (3). In other words, we have either $|X_+| < (1-\delta)n$ or $|X_-| < (1-\delta)n$, which concludes the proof.

In order to transform $\bar{G}$ into an unweighted instance, we take (roughly) $1/x'y'$ copies of G_0, $1/(1-x'-y')y'$ copies of G_+ and $1/(1-x'-y')x'$ copies of G_-, each vertex having weight 1. Prediction contains $n/2$ vertices from each copy of G_0, all vertices of all copies of G_+ and no vertices from the copies of G_-. This is to ensure that the minimum vertex cover on the copies of G_0, on the copies of G_+, and on the copies of G_- respectively have approximately the same ratio as in $\bar{G}$. In order to achieve approximation ratio $1+x+y-\epsilon$, the algorithm A would need to find a solution to at least one copy of G_- or G_+ with size smaller than $(1-\delta)n$. □

C.3 Lower bound for a maximization problem

We show a similar result for the class of considered maximization problems.

Theorem 9. *Let A be a polynomial-time learning-augmented approximation algorithm for Maximum Independent Set that guarantees an approximation ratio of at least*

$$1 - f\left(\frac{\eta^+}{\mathrm{OPT}}, \frac{\eta^-}{\mathrm{OPT}}\right),$$

for some non-decreasing function f. Then, assuming UGC, we have

$$f(x, y) \geqslant x + y,$$

for any choice of $x, y \geqslant 0$ satisfying $x + y \leqslant 1$.

Proof. By contradiction, consider x, y such that $f(x, y) < x + y$. We denote $\epsilon := (x + y) - f(x, y)$, and let $\delta := \epsilon/20$. We proceed similarly to the proof of Theorem 8, and we reach contradiction with Proposition 3.

We construct algorithm $\bar{A}$ such that for any input graph G with maximum independent set X^* of size $|X^*| \in [(1-\delta)\frac{n}{2}, (1+\delta)\frac{n}{2}]$ it produces an independent set X of size $|X| > \delta n$, certifying that G does not belong to the NO case. Note that graphs with an independent set larger than $(1+\delta)\frac{n}{2}$ have a minimum vertex cover of size smaller than $(1-\delta)\frac{n}{2}$. Therefore, in such graphs we can find a 2-approximate solution C of the minimum vertex cover problem and then $V(G) \setminus C$ will be an independent set of size $n - |C| > n - (1-\delta)n = \delta n$.

Given the input graph G, $\bar{A}$ constructs a (weighted) graph $\bar{G}$ and a prediction $\hat{X}$, and uses them as an input for algorithm A. Graph $\bar{G}$ is a disjoint union of three graphs: G_0, G_+, and G_-. Let $x' := \frac{1-\delta}{1+\delta}x$ and $y' := \frac{1-\delta}{1+\delta}y$.

- G_0 is a graph with $n/2$ vertices and no edges, we have $\bar{w}(v) = (1 - x' - y')$ for each $v \in V(G_0)$.
- G_+ is a copy of G with scaled weights, we have $\bar{w}(v) = x'$ for each $v \in V(G_+)$.
- G_- is a copy of G with scaled weights, we have $\bar{w}(v) = y'$ for each $v \in V(G_-)$.

The prediction $\hat{X}$ contains all vertices from G_0, all vertices of $V(G_+)$ and no vertices of $V(G_-)$. Receiving a solution $\bar{X}$ found by algorithm A on $\bar{G}$, we denote $X_+ = \bar{X} \cap V(G_+)$ and $X_- = \bar{X} \cap V(G_-)$. $\bar{A}$ returns the larger of the two sets X_+ and X_-, which we denote by X. We now prove that $|X| > \gamma n$ if G is the YES case.

First, we show that $\eta^+/\mathrm{OPT} \leqslant x$ and $\eta^-/\mathrm{OPT} \leqslant y$. Let OPT be the value of an optimal solution on $\bar{G}$. We have $\mathrm{OPT} = (1 - x - y)\frac{n}{2} + x|X^*| + y|X^*|$ which belongs to $\left[(1-\delta)\frac{n}{2}, (1+\delta)\frac{n}{2}\right]$ by the assumption about the size of X^*. For the error, we have

$$\eta^+ = x' \cdot (n - |X^*|) = \frac{1-\delta}{1+\delta}x \cdot (n - |X^*|), \quad \text{and}$$
$$\eta^- = y' \cdot |X^*| = \frac{1-\delta}{1+\delta}y \cdot |X^*|.$$

Since both $(n - |X^*|)$ and $|X^*|$ are at most $(1+\delta)\frac{n}{2}$ and $\mathrm{OPT} \geqslant (1-\delta)\frac{n}{2}$, we have $\eta^+/\mathrm{OPT} \leqslant x$ and $\eta^-/\mathrm{OPT} \leqslant y$. By monotonicity of f, this implies that $\bar{X}$ is at least a $(1-x-y+\epsilon)$-approximate solution. Therefore, we have

$$\begin{aligned}\bar{w}(\bar{X}) &\geqslant (1 - x - y + \epsilon) \cdot \mathrm{OPT} \\ &\geqslant (1 - x - y) \cdot (1-\delta)\frac{n}{2} + \epsilon \cdot (1-\delta)\frac{n}{2} \\ &\geqslant (1 - x - y)\frac{n}{2} - 3\delta\frac{n}{2} + \epsilon\frac{n}{2}.\end{aligned}$$

On the other hand, we can write

$$\begin{aligned}\bar{w}(\bar{X}) &\leqslant (1 - x' - y')\frac{n}{2} + x'|X_+| + y'|X_-| \\ &\leqslant (1 - x - y)\frac{n}{2} + x'|X_+| + y'|X_-| + 4\delta\frac{n}{2}.\end{aligned}$$

In the last step, we use the definition of x' and y', and $\frac{1-\delta}{1+\delta} \geqslant 1 - 2\delta$. The two equations above imply that at least one of $|X_+|$ and $|X_-|$ must be larger than $(\epsilon\frac{n}{2} - 7\delta\frac{n}{2})/2 > \delta n$, by our choice of δ. In other words, we have that $|X_+| > \delta n$ or $|X_-| > \delta n$, which concludes the proof.

In order to transform $\bar{G}$ into an unweighted instance, we take $\frac{1}{x'y'}$ copies of G_0, $\frac{1}{(1-x'-y')y'}$ copies of G_+ and $\frac{1}{(1-x'-y')x'}$ copies of G_-, each vertex having weight 1. $\square$

D EXPERIMENTAL EVALUATION

In this section we present an experimental evaluation of our refined learning-augmented algorithm for the Steiner Tree problem from Section 4. The source code of our experiments is available in the supplementary material.

Dataset. We base our experiments on the heuristic track of the 2018 PACE Challenge (Bonnet and Sikora, 2018), which is to our knowledge the most recent large-scale computational challenge concerning the Steiner Tree problem. In particular, we use their dataset[4], which consists of 199 graphs selected from among the hardest instances in the SteinLib library (Koch et al., 2000).[5]

Algorithms. We also use PACE as the source of the state-of-the-art Steiner Tree solver, with which we compare our algorithm. More specifically, in our experiments we evaluate three algorithms:

- **Mehlhorn**'s 2-approximation algorithm, implemented by us in C++. Its mean empirical approximation factor on PACE instances that we observe is ≈ 1.17. On average it spends 34 milliseconds per instance, and never needs more than 500 milliseconds.
- **CIMAT**, the winning solver from the PACE Challenge, whose C++ implementation is available on GitHub (Ruiz et al., 2018). It is an evolutionary algorithm that can be stopped at any time and outputs the best solution found so far. It was designed to run for 30 minutes per instance, as this was the time limit in PACE. However, we noticed that on 95% of instances already after one minute it produces a solution within 1.01 of the optimum, and therefore we decided to always run it only for one minute, in order to keep the computational costs of the experiments down. We also remark that, on average, it took the CIMAT solver ≈ 10 seconds to output a first feasible solution. In all our experiments, we use the value of the solution returned by CIMAT as an estimate of the value of an optimal solution OPT, which would be difficult and impractical to calculate exactly.
- **ALPS**, our **al**gorithm with **p**redictions from Section 4, with the "confidence" hyperparameter $\alpha \in \{1.1, 1.2, 1.4, 2, 4, \infty\}$. In terms of efficiency it is virtually indistinguishable from Mehlhorn's algorithm, which it runs as a subroutine and which dominates its running time.

Predictions. We generate both synthetic predictions and learned predictions. For both types of predictions we vary their quality, which is captured by the parameter $p \in \{0.0, 0.1, \ldots, 1.0\}$; the higher the parameter p the higher the prediction error. The predictions are then fed to our algorithm (ALPS), and we compare the quality of its outputs with those of the other two algorithms (CIMAT and Mehlhorn's).

- **Synthetic predictions.** In our first experiment, we simulate a predictor by introducing artificial noise to groundtruth labels. For each instance from the dataset, we compute a (near-)optimal solution $S \subseteq E$ using CIMAT. Then, for each value of the parameter p, we swap out a randomly selected p-fraction of the edges in S for a randomly selected subset of edges from $E \setminus S$ of the same cardinality. This yields a prediction with error roughly $\eta^+ \approx p \cdot \mathrm{OPT}$ and $\eta^- \approx p \cdot \mathrm{OPT}$.
- **Learned predictions.** In our second, more realistic experiment we test our approach end-to-end, i.e., we first learn predictions, and then use them to solve instances not seen during learning. For this experiment, we need to work with *distributions over instances* instead of individual problem instances. To this end, for each instance $I = (V, E, T, w)$ from our dataset, and for each value of the parameter p, we consider a distribution that returns instances of the form $I' = (V, E, T', w)$, with a p-fraction of the terminals resampled (but with the same underlying graph and edge weights). Actually, we consider two types of distributions, depending on how they resample terminals:
  - **"Fixed core" distribution.** We fix a $(1-p)$-fraction of the terminals $T_{\text{old}} \subseteq T$ and for each sampled instance I' we swap out the remaining p-fraction by independently sampling $\lceil p \cdot |T| \rceil$ terminals $T_{\text{new}} \subseteq V \setminus T$, returning $I' = (V, E, T_{\text{old}} \cup T_{\text{new}}, w)$.
  - **"No core" distribution.** Each time we sample a fresh $(1-p)$-fraction of the terminals in T to obtain T_{old} (i.e., $T_{\text{old}} \subseteq T$ is not fixed over all the samples).

[4] Available at `github.com/PACE-challenge/SteinerTree-PACE-2018-instances`.

[5] As of 2018, a majority of these instances could not be solved to optimality within one hour with state-of-the-art solvers. The average number of vertices is $\approx$ 27k, the average number of edges is $\approx$ 48k and the average number of terminals is $\approx$ 1k.

For each such distribution, we sample $k = 10$ instances, and we compute a (near-)optimal solution to each of them using CIMAT. Then, we evaluate ALPS on each such sampled instance using the prediction learned from (the solutions to) the remaining $k - 1 = 9$ instances sampled from the same distribution (i.e., we preform leave-one-out cross-validation). To learn the prediction from a set of left-out instances, we predict edge $e \in E$ iff it appeared in more than half of the (near-)optimal solutions to the left-out instances, which coincides with empirical risk minimization of the combined prediction error $\eta_+ + \eta_-$.

Evaluation metrics. To evaluate the performance of our algorithms, we use two different measures. For the synthetic predictions we look at the (empirical) approximation ratios observed for each of the algorithms and averaged over all the instances (see Figure 2a). For the learned predictions such average would not be a useful metric – this is because swapping out a set of terminals often completely changes both the value of an optimal solution and the relative performance of Mehlhorn's algorithm (see Figures 2b–2d). For this reason, before averaging over the instances, we normalize the solutions costs so that 0 corresponds to the optimum and 1 corresponds to the performance of Mehlhorn's algorithm. Specifically, denoting by $c_{\mathrm{ALPS}(\alpha)}, c_{\mathrm{OPT}}$, and c_{MST} the cost of the solution output by ALPS (with parameter α), the optimum value and the cost output by Mehlhorn's algorithm, respectively, we define the normalized cost of ALPS as $\frac{c_{\mathrm{ALPS}(\alpha)} - c_{\mathrm{OPT}}}{c_{\mathrm{ALPS}(\alpha)} - c_{\mathrm{MST}}}$. For predictions of varying quality p and for different values of α, this measure then accurately reflects how ALPS interpolates between the optimum (0) and Mehlhorn's algorithm (1).

Results. The results of our empirical evaluation are depicted in Figure 2. Figure 2a displays the performance of ALPS with synthetic prediction, averaged over all instances. Figures 2b, 2c, and 2d depict several typical behaviors on distributions of the Steiner Tree instances with resampled terminals, which we used in our experiment with learned predictions. Those figures illustrate the need to consider normalized costs. Finally, Figures 2e and 2f display the normalized costs of algorithms averaged over all distributions, with fixed core and with no core, respectively.

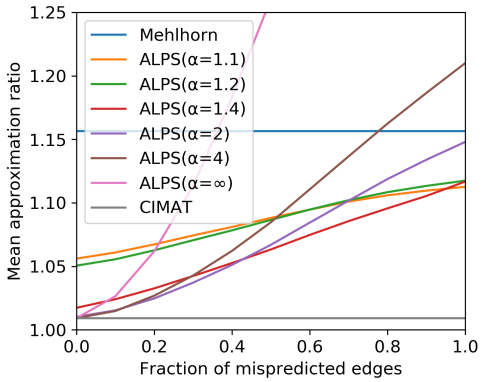
Out of the datapoints that we report, 95% of them had standard deviations below 0.1 of their values, and the maximum standard deviation among them was 0.25 of the corresponding value. It would be interesting to run the experiments with a higher number of iterations in order to bring the standard deviations down, but our computational resources do not permit that (the described experiments already required ≈ 700 CPU hours).

Conclusions. In all our experiments, ALPS (especially with the confidence parameter $\alpha \geqslant 1.4$) equipped with good predictions (low values of the parameter p, left side of the plots) outputs solutions almost as good as the (near-)optimal solutions output by CIMAT, which is orders of magnitude slower. On the other hand, the classic approximation algorithm of Mehlhorn, which has the same running time as ALPS, outputs solutions with a noticeably higher total cost.

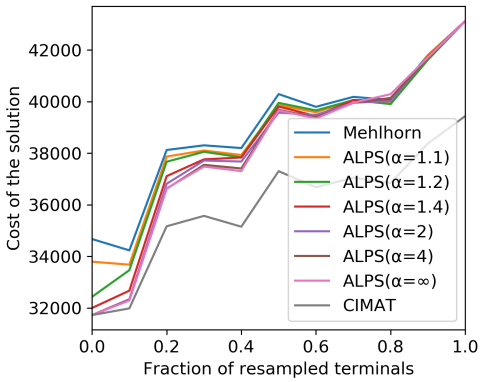
Unsurprisingly, with the increasing prediction error (high values of the parameter p, right side of the plots) performance of ALPS slowly degrades. What is notable though is that already for moderate values of the confidence parameter ($\alpha \leqslant 2$) ALPS seems robust, i.e., it is never significantly worse than Mehlhorn, even with bad predictions. This is despite the fact that our ALPS implementation does not involve a separate robustification step (as in Corollary 2).

As the theory predicts, in the case of accurate predictions it is better to choose a large α while for bad predictions a smaller α is better, and, as we explain in Section B.1, one may want to run ALPS for a geometric progression of α's and pick the best solution. However, in our experiments, it seems that choosing just a single $\alpha = 1.4$ or $\alpha = 2$ is a good choice almost universally.

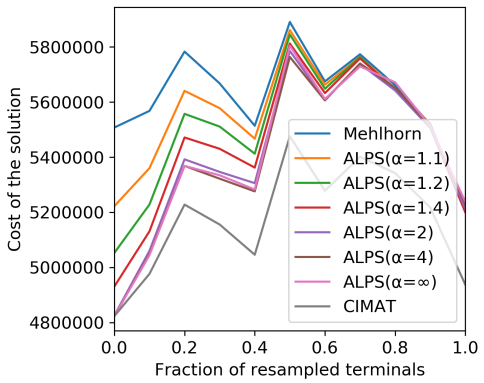
Finally, it is worth to note that, somewhat surprisingly, synthetic predictions with $p = 1.0$, which are almost completely random, still allow (for small enough α) to improve over Mehlhorn's algorithm. This surprising behavior can be explained by the fact that instances that are relatively hardest for the Mehlhorn's algorithm tend to be very symmetric and to have multiple (close-to-)optimal solutions, and it seems that randomly decreasing a small fraction of edge weights by a small factor α breaks the symmetry and guides the algorithm towards a good solution.



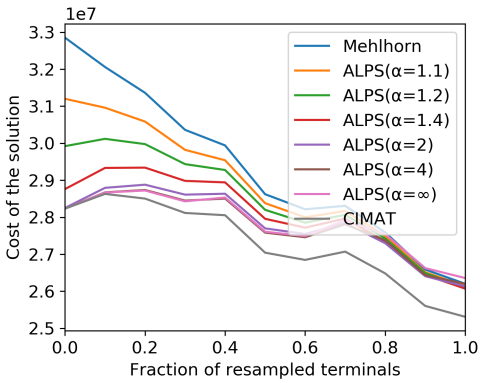
(a) Synthetic predictions. Average approximation ratio over all instances.



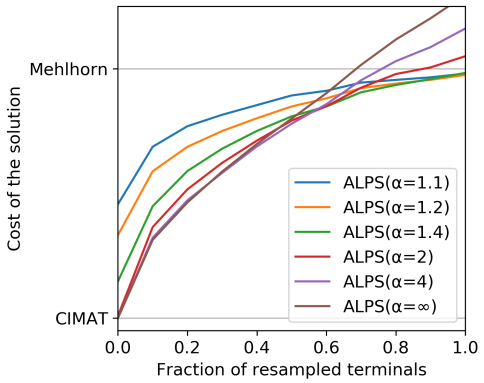
(b) Learned predictions. "Fixed core" distribution based on instance 011.



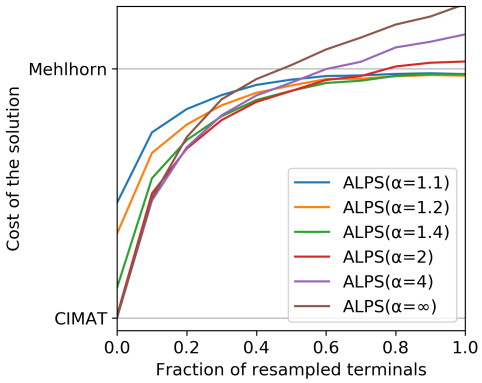
(c) Learned predictions. "No core" distribution based on instance 082.



(d) Learned predictions. "Fixed core" distribution based on instance 178.



(e) Learned predictions. Normalized cost averaged over all distributions with fixed core.



(f) Learned predictions. Normalized cost averaged over all distributions with no core.

Figure 2: Experimental evaluation of our refined Steiner Tree algorithm from Section 4 (ALPS), with different values of the confidence parameter α, compared to an equally fast classic approximation algorithm (Mehlhorn) and a much slower near-optimal solver (CIMAT). The x-axis represents the parameter p controlling the prediction error: accurate predictions are to the left and erroneous predictions are to the right. The y-axis represents the value of the returned solution (the lower the better).