LEARNING BETTER PREDICTIONS FOR WARM-STARTING ALGORITHMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently developed warm-starting learning-augmented algorithms offer improved running times for a number of computational problems when provided with predicted solutions that are reasonably *close* to optimal solutions. This poses a fundamental *learning problem*: Find a prediction that works well, on average, for instances drawn from a given distribution. This learning problem is typically solved under the simplifying assumption that each instance has a *unique* optimal solution, which is a strong assumption, and very far from the truth in many practical scenarios. In this paper we study the more realistic learning setting that allows for the presence of multiple optimal solutions. We focus on the two most popular problems in the literature on warm start with predictions, namely *bipartite min-cost perfect matching* and *maximum flow*. For the matching problem, we show how to learn an integral dual prediction minimizing the ℓ_1 distance to the closest integral dual solution, which can be fed to the warm-starting algorithm of Dinitz et al. (NeurIPS 2021). We also show NP-hardness of the analogous learning problem for the ℓ_0 prediction error metric, which is used in the work of Chen et al. (ICML 2022). For the flow problem, we show that, even though the corresponding ℓ_1 learning problem is NP-hard, nonetheless we can optimize a function that is a more direct and *tighter* upper bound on the running time of the warm-starting algorithm of Davies et al. (ICML 2023). We expect that our findings will inspire more research on applying combinatorial optimization toolbox to learning problems motivated by algorithms with predictions, which so far are primarily studied through the continuous optimization lens.

1 INTRODUCTION

Imagine you are solving multiple similar instances of a computational problem, e.g., the instances are drawn at random from a concentrated distribution, or they are snapshots of a single underlying instance changing over time. Could you learn what a typical optimal solution looks like, and then use it to *warm-start* subsequent computations and get a better running time?

Following the seminal work of Dinitz et al. (2021) on the *bipartite min-cost perfect matching* problem, the above question was answered for a number of problems, including, e.g., *shortest paths* (Chen et al., 2022a), *convex discrete optimization* (Sakaue & Oki, 2022; 2023), *maximum flow* (Davies et al., 2023; Polak & Zub, 2024; Davies et al., 2024), *global minimum cut* (Niaparast et al., 2025), and *sorting* (Bai & Coester, 2023). All these works provide algorithms whose running times depend on some notion of *distance* between the given predicted solution vector x (not necessarily feasible nor optimal) and the optimal solution vector for that input instance y. This distance measure, the so-called *prediction error*, is often set to the ℓ_1 distance $\|x - y\|_1$, with x and y viewed as multidimensional vectors in the solution space.

How to *learn* a good prediction for such an algorithm? So far, a prevalent way (Dinitz et al., 2021; Chen et al., 2022a; Dinitz et al., 2022; Khodak et al., 2022; Polak & Zub, 2024) to model this learning problem is as follows: Assume the input instances come from a distribution $\mathcal{D}$. Our goal is to find a prediction x minimizing the expected prediction error, i.e., $\arg\min_x \mathbb{E}_{I\sim\mathcal{D}}\|x - y(I)\|_1$, where $y(I)$ denotes the optimal solution for instance I. Such stated learning problem becomes relatively straightforward to solve: By standard PAC-learning arguments (Dinitz et al., 2021, Section 3.3), it suffices to draw from $\mathcal{D}$ a polynomial number of samples $I_1, \ldots, I_k$ and perform *empirical risk*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

minimization (ERM), i.e., find x minimizing the empirical error over these samples:

$$\arg\min_x \frac{1}{k} \sum_{j\in[k]} \|x - y(I_j)\|_1. \tag{†}$$

This average ℓ_1 distance is simply minimized by the coordinate-wise median of $y(I_1), \ldots, y(I_k)$.

The above approach, which we call *naïve ERM*, has an important shortcoming: It works under the assumption that there is a *unique* optimal solution $y(I)$, for each problem instance I from $\mathcal{D}$. This is a strong assumption, and very far from the truth in many practical scenarios. What happens when there are multiple optimal solutions? The upper bounds on the running times of the above warm-starting algorithms in terms of $\|x-y\|_1$ happen to hold with respect to *any* optimal solution y, so in particular with respect to the one that is *closest* to the prediction x. As a consequence, the actual learning problem that we should be concerned about is

$$\arg\min_x \frac{1}{k} \sum_{j\in[k]} \min\big\{\|x - y\|_1 : y \in \{\text{optimal solutions to } I_j\}\big\}. \tag{‡}$$

This is a much harder problem than naïve ERM (†). Indeed, it does not suffice to first independently solve each instance I_j, for $j \in [k]$,[1] and then calculate the coordinate-wise median of the solutions. Instead, one has to solve these instances simultaneously, looking for solutions that are close to each other.[2] In other words, our driving question is the following:

How to find a solution of an optimization problem that is simultaneously close to ***some*** *optimal solutions to several given instances of that problem?*

1.1 Brief summary of our contributions

We resolve the above question for the two most prominent problems studied in the context of warm start with predictions, namely *bipartite min-cost perfect matching* (Dinitz et al., 2021; Chen et al., 2022a; Sakaue & Oki, 2022; Dinitz et al., 2022; Khodak et al., 2022; Sakaue & Oki, 2023) and *maximum flow* (Davies et al., 2023; Polak & Zub, 2024; Davies et al., 2024; Sakaue & Oki, 2022; 2023; Chen et al., 2026). Our learning algorithms produce predictions which are ready to be used in the existing learning-augmented algorithms by Dinitz et al. (2021) and Davies et al. (2023). In both cases, learning is efficient, because it amounts to solving a min-cost flow problem, in a network of size proportional to the total size of the sampled training instances, which is a problem that can be solved in almost-linear time in theory (Chen et al., 2022b), and also has highly optimized solvers that are very fast in practice, e.g., (Perron & Furnon, 2025). Concretely, our main results are:

- For predicting integral dual solution for bipartite min-cost perfect matching:
  - Algorithm for learning best predictions wrt. the ℓ_1 error, to be fed to the algorithm of Dinitz et al. (2021).
  - NP-hardness of learning predictions optimal wrt. the ℓ_0 error, the error metric used in (Chen et al., 2022a).
- For predicting integral primal solutions for maximum flow:
  - NP-hardness of learning predictions optimal wrt. the ℓ_1 error considered in (Davies et al., 2023; Polak & Zub, 2024).
  - Algorithm for learning optimal predictions to be used in the algorithm of Davies et al. (2023). Instead of the ℓ_1 error, our algorithm minimizes a more direct and tighter upper bound on the running time of their algorithm.

Main conceptual contribution. Our main intention is to bring the combinatorial optimization toolbox to learning problems motivated by algorithms with predictions. These tools can be used to design very efficient learning algorithms, as we demonstrate on the examples of bipartite min-cost perfect matching and maximum flow. We hope to inspire more research in this area, which so far has been primarily viewed through the continuous optimization lens (Khodak et al., 2022; Sakaue & Oki, 2023).

[1] We use $[n]$ to denote $\{1, \ldots, n\}$.

[2] The sample complexity stays the same when moving from (†) to (‡), e.g., the proof in (Polak & Zub, 2024, Section 4) still goes through.

1.2 Related work

Since the seminal paper by Lykouris & Vassilvitskii (2021), learning-augmented algorithms were proposed for plenitude of problems, especially in the *online* setting, see, e.g., (Mitzenmacher & Vassilvitskii, 2020) and (Lindermayr & Megow, 2022) for an overview. However, a noteworthy line of research in this area is concerned with using predictions to speed up *offline* algorithms, e.g., (Mitzenmacher, 2018; Ergun et al., 2022; Carvalho & Lawrence, 2023; Lattanzi et al., 2023; Silwal et al., 2023; Feijen & Schäfer, 2021; Eberle et al., 2024; Cabello et al., 2026), and all the previously mentioned warm-starting algorithms fit into this group.

As for learning predictions for warm start, multiple works (Dinitz et al., 2021; Chen et al., 2022a; Dinitz et al., 2022; Khodak et al., 2022; Polak & Zub, 2024) propose methods in the spirit of the naïve ERM approach (†). There are two notable exceptions:

First, while not explicitly mentioned in their paper, the methods of Khodak et al. (2022) can be extended to (‡) in some specific situations. An example would be a dataset of linear programs, which admit fractional solutions. There, the set of optimal solutions is always convex, and the ℓ_1 distance to this set is a convex function. Consequently, it can be minimized using the online learning methods proposed in (Khodak et al., 2022).[3] However, in this paper we consider problems with integral solutions, where the set of optimal solutions is discrete and the ℓ_1 distance to it is non-convex.

Second, shortcomings of the naïve ERM approach (†) were already noticed by Sakaue & Oki (2023), who propose learning algorithms in the spirit of (‡) for a class of convex problems, which includes, among others, bipartite min-cost perfect matching and min-cost flow. Unfortunately, their approach is inherently fractional. Similarly to Khodak et al. (2022), they use the online-to-batch conversion together with online learning techniques that require a convex loss function. Therefore, they minimize the distance to the *convex hull* of integral optimal solutions, which is a valid approach in their setting. Unfortunately, as we show in Appendix A, a fractional prediction with an arbitrarily small ℓ_1 distance from the convex hull of integral optimal solutions may not be useful for the algorithms of Dinitz et al. (2021) and Chen et al. (2022a). Note that the distance to the set of integral optimal solutions, optimized in our paper, is non-convex and differs substantially from the distance to their convex hull.

Brief comparison of the current warm-starting algorithms for matching. While the focus of our paper is on producing predictions, our main motivation comes from the existing literature on learning-augmented algorithms, which is an active area of research and new algorithms are to be expected in future. In the following bounds, we denote by η_p and $\bar{\eta}_p$ the ℓ_p distance of the provided prediction from the closest optimal solution and the convex hull of the optimal solutions respectively. Dinitz et al. (2021) proposed an algorithm requiring integer predictions with running time $O(m\sqrt{n} \cdot \eta_1)$. Sakaue & Oki (2022) proposed an algorithm which can use fractional predictions like those produced by Sakaue & Oki (2023) with running time $O(m\sqrt{n} \cdot \bar{\eta}_\infty)$. Note that $\bar{\eta}_\infty \leq \eta_1$ for any prediction. The algorithm of Chen et al. (2022a) again requires integral predictions and improves the coefficient of the prediction error to $O(m)$, achieving running time $O(m\sqrt{n} + m \cdot \eta_0)$. Since $\eta_0 \leq \eta_1$ (while η_0 and $\bar{\eta}_\infty$ are incomparable), it can use predictions provided by our algorithm achieving running time $O(m\sqrt{n} + m \cdot \eta_1)$. This gives an improvement over (Sakaue & Oki, 2022) as far as $\eta_1 < \sqrt{n}\,\bar{\eta}_\infty$. In a natural scenario of a large perturbation affecting the cost of a relatively small number of edges in the input graph compared to the training dataset, this improvement can be quite large.

1.3 Open problems

Approximate learning for matching with ℓ_0 error. Given that we show NP-hardness of learning dual predictions for bipartite min-cost perfect matching wrt. the ℓ_0 error (Theorem 5), a natural open question is whether this learning problem can be approximated efficiently. Note that in this context any constant (or even logarithmic) factor approximation would be interesting, given that the prediction error appears as a multiplicative factor inside the big-O bound on the running time of the algorithm by Dinitz et al. (2021). Neither our learning algorithm for the ℓ_1 error nor our exact hardness proof seem to generalize to this setting, so an entirely new approach is likely needed.

[3] We stress that the learning algorithm of (Khodak et al., 2022) does produce integral predictions in the spirit of (†), by using a rounding procedure described in their paper, but it does not generalize to (‡) when there are multiple integral optima.

Learning problem (‡) for other warm-starting algorithms. We expect our work to inspire studying the general learning problem (‡), i.e., without the solutions-uniqueness assumption, for many other algorithms with predictions, e.g., for negative-weight shortest paths (Chen et al., 2022a), push-relabel (Davies et al., 2024), or global minimum cut (Niaparast et al., 2025).

2 BIPARTITE MIN-COST PERFECT MATCHING

In the *bipartite min-cost perfect matching* problem (also known as the *assignment* problem), we are given a bipartite graph $G = (V, E)$, where $V = L \sqcup R$ and $E \subseteq L \times R$, with n vertices (where n is even) and m edges, and a vector of edge costs $c \in \mathbb{Z}^m$. The goal is to find a perfect matching[4] minimizing $\sum_{e \in M} c_e$. The standard algorithm used in practice to solve this problem is the so-called Hungarian method (Kuhn, 1955), which runs in time $O(mn + n^2 \log n)$, although there are theoretically faster but significantly more complex and less practical algorithms, e.g., (Duan & Su, 2012; Chen et al., 2022b).

Dinitz et al. (2021) give a learning-augmented algorithm for the problem, based on the Hungarian method. Their algorithm, given a prediction vector $\hat{y} \in \mathbb{Z}^n$, runs in time $O(m\sqrt{n} \cdot \eta(c, \hat{y}))$, where $\eta : \mathbb{Z}^m_{\geq 0} \times \mathbb{Z}^n \to \mathbb{Z}_{\geq 0}$ denotes the prediction error defined as

$$\eta(c, \hat{y}) := \min_{y^* \in \mathcal{Y}^*(c)} ||y^* - \hat{y}||_1,$$

where $\mathcal{Y}^*(c)$ denotes the set of all optimal *dual solutions* for the min-cost perfect matching problem in the input graph with edge costs $c \in \mathbb{Z}^m_{\geq 0}$. Here, an optimal dual solution means an integer vector $y \in \mathbb{Z}^n$ with $y_u + y_v \leq c_{uv}$, for every edge $(u, v) \in E$, that *maximizes* $\sum_{v \in V} y_v$. In order to get such a prediction vector, we shall solve the following learning problem:

Problem 1 (Learning duals for bipartite min-cost perfect matching with ℓ_1 error)**.** *Given a bipartite graph and a collection of k vectors of edge costs $c^1, \ldots, c^k \in \mathbb{Z}^m_{\geq 0}$, find a prediction vector $\hat{y} \in \mathbb{Z}^n$ minimizing the expected prediction error, i.e.,*

$$\frac{1}{k} \sum_{i \in [k]} \min \left\{ ||y^* - \hat{y}||_1 \;\middle|\; y^* \in \mathcal{Y}^*(c^i) \right\}.$$

Again, this problem would be easy if $|\mathcal{Y}^*(c^i)| = 1$. Unfortunately, it is well known that, unless $\mathcal{Y}^*(c)$ is empty, its size is *infinite*.[5] Our first result is a learning algorithm for this general case.

2.1 LEARNING ALGORITHM FOR MATCHING WITH ℓ_1 ERROR

Our algorithm reduces Problem 1 to solving (the dual of) a min-cost *flow* problem, on a network with $O(kn)$ vertices and $O(k(m + n))$ edges and can be solved in almost linear time. Another possible approach would be via L-convex function minimization (see the textbook of (Murota, 2003) for more information about L-convex functions) which would lead to a worse but still polynomial running time. In order to describe the network, we express Problem 1 as a linear program (LP). For $j \in [k]$, the j-th instance of the problem of finding a minimum-cost bipartite perfect matching, with respect to the costs $c^j \in \mathbb{Z}^m_{\geq 0}$, can be modeled using the following LP.

$$\begin{aligned} (P^j) \qquad \text{minimize} \quad & \sum_{uv \in E} c^j_{uv} x^j_{uv} \\ \text{subject to} \quad & \sum_{u \,:\, uv \in E} x^j_{uv} = 1 \quad \forall\, v \in V \\ & x^j_{uv} \geq 0 \quad \forall\, uv \in E \end{aligned}$$

[4] Perfect matching is a subset of edges $M \subseteq E$ such that every vertex from V is incident to exactly one edge in M.

[5] This can be seen in the LP formulation (D^j): given an optimal solution y, decreasing the dual variables in the left partite by δ and increasing those in the right partite by δ gives a feasible integral solution with the same objective value for any $\delta \in \mathbb{Z}$.

While the integrality constraints on the variables are not present explicitly, it is well known that the constraint matrix of this program is *totally unimodular*,[6] the right-hand-side vector is integral, and therefore every extreme point solution to (P^j) is an integral indicator vector $x^j \in \{0,1\}^m$ of some perfect matching of G. We refer to (Schrijver, 2003) for more information. Given that the edge costs $c^j \in \mathbb{Z}^m_{\geq 0}$ are integral, the same is true about its dual linear program:

$$(D^j) \qquad \text{maximize} \qquad \sum_{v \in V} y^j_v$$
$$\text{subject to} \qquad y^j_u + y^j_v \leq c^j_{uv} \quad \forall\, uv \in E$$

The following LP contains a concatenation of the programs $(D^1), \ldots, (D^k)$. In addition, it contains variables for the prediction $\hat{y}$ and variables d and δ capturing the ℓ_1 distance between $\hat{y}$ and the solutions $y^1, \ldots, y^k$ to the programs $(D^1), \ldots, (D^k)$. Namely, for each $j \in [k]$ and each $v \in V$, we have $|\hat{y}_v - y^j_v| \leq d^j_v + \delta^j_v$ by constraints (2) and (3). Choosing the parameter M large enough (e.g., larger than the sum of the edge-costs in the input graphs), the objective function ensures that, in every optimal solution of (D), each y^j is an optimal solution of (D^j) while $\sum_{j=1}^k \|\hat{y} - y^j\|_1$ is minimized.

$$(D) \qquad \text{maximize} \qquad \sum_{j=1}^{k} \sum_{v \in V} (-1)(d^j_v + \delta^j_v) + \sum_{j=1}^{k} \sum_{v \in V} M y^j_v$$

$$\text{subject to} \qquad y^j_u + y^j_v \leq c^j_{uv} \qquad \forall\, j \in [k], \forall\, uv \in E \qquad (1)$$
$$\hat{y}_v - y^j_v - \delta^j_v \leq 0 \qquad \forall\, j \in [k], \forall\, v \in V \qquad (2)$$
$$y^j_v - \hat{y}_v - d^j_v \leq 0 \qquad \forall\, j \in [k], \forall\, v \in V \qquad (3)$$
$$\delta^j_v, d^j_v \geq 0 \qquad \forall\, j \in [k], \forall\, v \in V \qquad (4)$$

Even though Problem 1 requires an integral solution and (D) does not contain explicit integrality constraints, we now argue that (D) still captures Problem 1 exactly. To this end, we consider the dual of (D), whose constraints, up to possible multiplication by -1, happen to coincide with the rows of the constraint matrix of a suitably-defined min-cost flow problem (described below as problem (F)).

We first recall the notion of flow. For a directed graph $G = (V, E)$, with two designated nodes $s \in V$ as source and $t \in V$ as target, an s-t flow is an edge vector $f \in \mathbb{R}^m_{\geq 0}$ satisfying so-called flow-conservation constraints: for each $v \in V \setminus \{s, t\}$, $\sum_{u\,:\,uv \in E} f_{uv} = \sum_{u\,:\,vu \in E} f_{vu}$ (total incoming flow equal to total outgoing flow). An s-t flow is said to satisfy some given edge capacities $u \in \mathbb{Z}^m_{\geq 0}$ if $f_e \leq u_e$ for each $e \in E$. The value of an s-t flow is the total flow leaving the source s, that is $\sum_{v\,:\,sv \in E} f_{sv} - \sum_{v\,:\,vs \in E} f_{vs}$. Given a directed graph G, capacity vector $u \in \mathbb{R}^m$, target flow value λ, and edge costs $c \in \mathbb{R}^m$, the *min-cost flow* problem asks for an s-t flow of value λ satisfying the capacities and of minimum total cost $\sum_{e \in E} c_e f_e$. It is well known that the min-cost flow problem can be formulated as an LP with a constraint matrix that is totally unimodular (Schrijver, 2003).

Min-cost flow problem (F). We consider a network with the vertex set $\{s, t\} \sqcup V \sqcup V^1 \sqcup \cdots \sqcup V^k$, where $V = L \sqcup R$ is the vertex set of G and, for each $j \in [k]$, $V^j = L^j \sqcup R^j = \{v^j \mid v \in V\}$ is one of k additional copies of V, see Figure 1. The edges in the network are as follows.

- For each $uv \in E$ (where $u \in L$ and $v \in R$), and each $j \in [k]$, there is an oriented edge from u^j to v^j with cost c^j_{uv} and unbounded capacity. We denote by x^j_{uv} the corresponding flow variable.
- For each $j \in [k]$, each $v \in L \cup R$ is connected to the corresponding vertex $v^j \in L^j \cup R^j$ by two opposing directed edges of cost 0 and capacity 1. For $v \in L$, variable a^j_v describes the flow from v to v^j and variable b^j_v the flow in the opposite direction. For $v \in R$, variable a^j_v describes the flow from v^j to v and variable b^j_v the flow in the opposite direction.
- For each $j \in [k]$, there is an edge from s to each $u^j \in L^j$ and an edge from each $v^j \in R^j$ to t, both with cost 0 and capacity M.

[6] For the curious reader, we say that an integer matrix M is *totally unimodular* (TU) if every square submatrix of M has determinant in $\{-1, 0, 1\}$. However, the exact definition of total unimodularity is immaterial for this paper. We only need to know what is written above about integrality of extreme point solutions of linear programs with TU constraint matrices, and that a transpose of a TU matrix is also TU and multiplying a row or column by -1 preserves the total unimodularity.

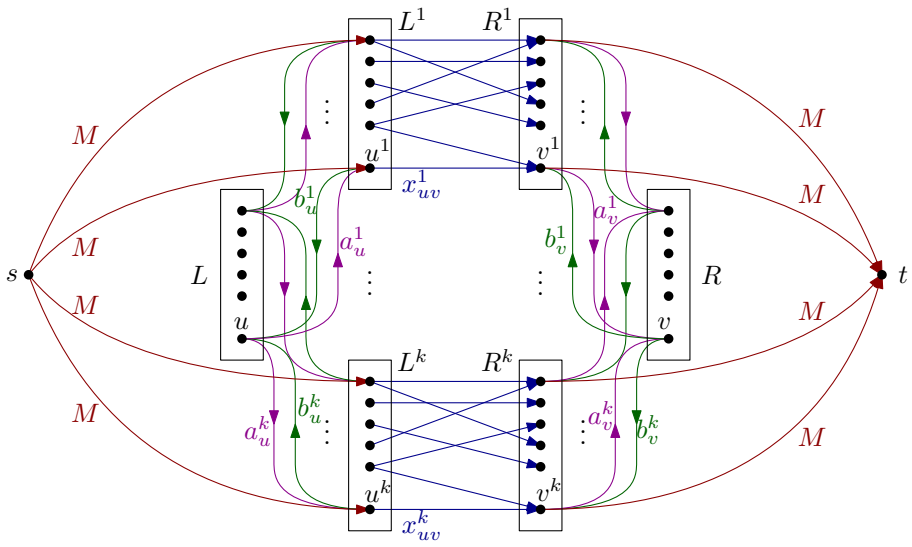


Figure 1: Flow network (F). $L \cup R$ and $L^j \cup R^j$ for $j = 1, \ldots, k$ are copies of the vertex set of G.

This network has $kn + n + 2$ vertices and $km + 2kn + kn$ edges. We seek a min-cost s-t flow of value $\lambda = \frac{1}{2}knM$.

Lemma 2. *Up to multiplication of the columns corresponding to vertices in $L \cup L^1 \cup \cdots \cup L^k$ by -1, the constraint matrix of (D) is a transpose of the constraint matrix of the min-cost flow problem (F).*

The proof of Lemma 2, which is rather mechanical, can be found in Appendix B.

Theorem 3. *An optimal solution to Problem 1 can be computed in almost-linear time. In particular, it can be extracted from a dual optimal solution to (F).*

Proof. Optimal solutions to Problem 1 correspond to optimal integral solutions to the linear program (D). Lemma 2 states that the constraint matrix of (D) is obtained by first transposing a totally unimodular matrix (corresponding to the min-cost flow problem), and then by multiplying some columns by -1. Such operations do not affect the total unimodularity, hence the constraint matrix of (D) is totally unimodular. Since the right-hand side of the constraints of (D) is an integer vector, all the extreme point solutions of (D) have integer coordinates. Hence, to solve Problem 1 it is enough to find an optimal extreme point solution to the linear program (D), which can be done in polynomial time (Grötschel et al., 1988; Renegar, 2001). One can actually speed up the computation, by relying on specialized almost-linear time algorithms: First, one can find an optimal solution to (F) via a specialized flow algorithm (Chen et al., 2022b), and then one can use a classical procedure (Ahuja et al., 1993, Chapter 9.8) to find a corresponding optimal integral dual solution by computing negative-weight single-source shortest paths in the residual network (Bernstein et al., 2025; Bringmann et al., 2023). In this case, since the constraint matrix of (D) has flipped signs in columns corresponding to the vertices in $L \cup L^1 \cup \cdots \cup L^k$ compared to the dual of (F), as the last step one has to multiply the coordinates of the dual variables corresponding to these vertices by -1. □

2.2 Hardness for matching with ℓ_0 error

Chen et al. (2022a) improve the running time of the algorithm of Dinitz et al. (2021) by a factor of $\sqrt{n}$ for reasonably close prediction vectors. Additionally, they replace the ℓ_1 error by the ℓ_0 error, and in the end they reach a running time of $O(m\sqrt{n} + (m + n \log n)\eta_0(c, \hat{y}))$, where η_0 is defined as $\eta_0(c, \hat{y}) := \min_{y^* \in \mathcal{Y}^*(c)} ||y^* - \hat{y}||_0$, where $\|x\|_0 := |\{x_i \colon x_i \neq 0\}|$ denotes the ℓ_0 norm. Note that all the vectors here have integer coordinates, so the ℓ_0 error is never greater than the ℓ_1 error.

They state their algorithm only for feasible dual vectors $\hat{y}$. Turning infeasible prediction vectors into feasible ones is delegated to the rounding algorithm by Dinitz et al. (2021), which only guarantees to find a prediction $\hat{y}'$ with $\|\hat{y}' - y^*\|_1 \leq 3\|\hat{y} - y^*\|_1$, for every optimal dual y^*. However, we

note that this rounding algorithm can easily be adapted to give the same guarantee wrt. the ℓ_0 norm: Whenever an edge constraint is violated in $\hat{y}$, any optimal solution must disagree with at least one of the endpoints. In linear time, we can compute a 2-approximate vertex cover for the graph induced by edges whose constraints are violated, e.g., by iteratively removing both endpoints of an edge (Garey & Johnson, 1979). We then sufficiently lower $\hat{y}_v$ for all v in the vertex cover, and hence we produce a feasible solution $\hat{y}'$ such that $\|\hat{y}' - \hat{y}\|_0 \leq 2\|y^* - \hat{y}\|_0$ and therefore $\|\hat{y}' - y^*\|_0 \leq \|\hat{y}' - \hat{y}\|_0 + \|\hat{y} - y^*\|_0 \leq 3\|\hat{y} - y^*\|_0$. With this minor modification, the algorithm of (Chen et al., 2022a) can also work with infeasible predictions, and this motivates the following learning problem.

Problem 4 (Learning duals for bipartite min-cost perfect matching with ℓ_0 error)**.** *Given a bipartite graph $G = (V, E)$, and a collection of k vectors of edge costs $c^1, \ldots, c^k : \mathbb{Z}^m_{\geq 0}$, find a prediction vector $\hat{y} \in \mathbb{Z}^n$ minimizing the expected prediction error, i.e.,*

$$\frac{1}{k} \sum_{i \in [k]} \min \big\{ ||y^* - \hat{y}||_0 \mid y^* \in \mathcal{Y}^*(c^i) \big\}.$$

In Appendix C we give a reduction from the *feedback arc set* problem (Karp, 1972) showing that the above learning problem is intractable.

Theorem 5. *Problem 4 is NP-hard.*

3 Maximum flow

Consider a directed graph $G = (V, E)$ on n vertices and m edges with source s and target t. Given edge capacities $c \in \mathbb{Z}^m_{\geq 0}$, the *maximum flow* problem asks to find an s-t flow satisfying the capacities and of maximum possible value. It is well-known that when the capacity values are integer, we can restrict our attention to integer flows $f \in \mathbb{Z}^m_{\geq 0}$. We denote by $\mathcal{F}^*(c)$ the set of all s-t flows $f \in \mathbb{Z}^m_{\geq 0}$ in G satisfying edge capacities $c \in \mathbb{Z}^m_{\geq 0}$ and of maximum value.

Davies et al. (2023) give an algorithm for maximum flow that, given a prediction vector $\hat{f} \in \mathbb{Z}^m_{\geq 0}$, runs in time $O\big((n+m) \cdot \eta(c, \hat{f})\big)$, where $\eta : \mathbb{Z}^m_{\geq 0} \times \mathbb{Z}^m_{\geq 0} \to \mathbb{Z}_{\geq 0}$ is the prediction error measure defined as $\eta(c, \hat{f}) := \min_{f^* \in \mathcal{F}^*(c)} ||f^* - \hat{f}||_1$. The following learning problem emerges:

Problem 6 (Learning maximum flow with ℓ_1 error)**.** *Given a graph $G = (V, E)$, with designated source and target $s, t \in V$, and a collection of k vectors of edge capacities $c^1, \ldots, c^k : \mathbb{Z}^m_{\geq 0}$, find a prediction vector $\hat{f} \in \mathbb{Z}^m_{\geq 0}$ minimizing the expected prediction error, i.e.,*

$$\frac{1}{k} \sum_{i \in [k]} \min \big\{ ||f^* - \hat{f}||_1 \mid f^* \in \mathcal{F}^*(c^i) \big\}. \tag{5}$$

This problem is solvable in polynomial time under the assumption that all the considered maximum flow instances have unique optimal solutions, i.e., $|\mathcal{F}^*(c^i)| = 1$, for every $i \in [k]$ (Davies et al., 2023). Unfortunately, we prove that without the uniqueness assumption this problem is NP-hard.

Theorem 7. *Problem 6 is NP-hard.*

The proof, by a reduction from 3SAT, can be found in Appendix D. Fortunately, not everything is lost. If we take a closer look at the analysis of the maximum flow algorithm by Davies et al. (2023), we observe that they actually upper bound the running time by

$$O\big((n+m) \cdot \big(\operatorname{ex}(\hat{f}) + \operatorname{def}(\hat{f}) + \operatorname{over}(c, \hat{f}) + \operatorname{OPT}(c) - v(\hat{f})\big)\big), \tag{6}$$

where $\operatorname{over}(c, \hat{f})$ denotes the total violation of the capacity constraints, i.e.,

$$\operatorname{over}(c, \hat{f}) := \sum_{e \in E} \max\{0, \hat{f}_e - c_e\};$$

$\operatorname{ex}(\hat{f})$ and $\operatorname{def}(\hat{f})$ capture the total excess and deficiency in the flow conservation constraints, i.e.,

$$\operatorname{ex}(\hat{f}) := \sum_{u \in V \setminus \{t\}} \max\big\{0, \sum_{v\,:\,vu \in E} \hat{f}_{vu} - \sum_{v\,:\,uv \in E} \hat{f}_{uv}\big\}, \quad \operatorname{def}(\hat{f}) := \sum_{u \in V \setminus \{s\}} \max\big\{0, \sum_{v\,:\,uv \in E} \hat{f}_{uv} - \sum_{v\,:\,vu \in E} \hat{f}_{vu}\big\};$$

and $\mathrm{OPT}(c) - v(\hat{f})$ is the difference between the value of a maximum flow and the value of $\hat{f}$.

They show that each of these four quantities can be bounded from above by $||f^* - \hat{f}||_1$, for every $f^* \in \mathcal{F}^*(c)$, and thus the asymptotic bound $O((n+m) \cdot \eta(c, \hat{f}))$ follows. Expressing the prediction error as the ℓ_1 distance makes their result easy to state and easy to understand intuitively, but it also makes the corresponding learning problem intractable. Instead, we can consider the learning problem of minimizing directly the expression from equation (6).

Problem 8 (Learning maximum flow with optimality-violation error)**.** *Given a graph $G = (V, E)$, with designated source and target $s, t \in V$, and a collection of k vectors of edge capacities $c^1, \ldots, c^k \in \mathbb{Z}^m_{\geq 0}$, find a prediction vector $\hat{f} \in \mathbb{Z}^m_{\geq 0}$ minimizing the expected prediction error defined as*

$$\frac{1}{k} \sum_{i \in [k]} \big(\mathrm{ex}(\hat{f}) + \mathrm{def}(\hat{f}) + \mathrm{over}(c^i, \hat{f}) + \mathrm{OPT}(c^i) - v(\hat{f}) \big). \tag{7}$$

Note that, unlike the ℓ_1 error, none of the four summands depends on a particular choice of an optimal flow $f^* \in \mathcal{F}^*(c^i)$, because the value of all such flows is necessarily the same. This independence is what makes the corresponding learning problem tractable. The theorem is proved in Appendix E.

Theorem 9. *Problem 8 can be solved in almost-linear time. In particular, it reduces to finding a min-cost flow (of any value) in a graph with n nodes and $O(n + km)$ edges.*

4 Experiments

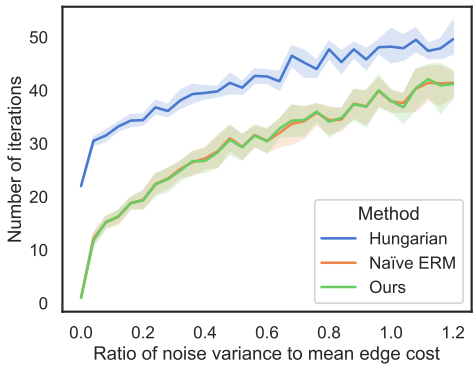
Even though the naïve ERM assumption (that each instance has a unique optimal solution) is not true in many practical scenarios, the simple median-based learning algorithm based on this assumption can still produce useful predictions for warm-starting algorithms. In particular, Dinitz et al. (2021) describe a batch learning experiment in which their matching algorithm equipped with such predictions outruns the popular Hungarian method. We reproduce their experiment, with our learning algorithm from Section 2.1 (coupled with their matching algorithm) added as the third method to the comparison. All our code and data are included in the supplementary material.

Datasets. We use the two datasets from the experiments of Dinitz et al. (2021), one is synthetic and the other based on real-world data. The synthetic dataset, which they call the *type model*, consists of graphs that have, on each side of the bipartition, $n = 500$ nodes divided into $\ell = 50$ groups.[7] For each pair of groups (one on each side), the cost of all edges from one group to the other is initialized to a value drawn independently at random from the geometric distribution $\mathrm{Geom}(1/250)$, with mean 250. Then, to generate a particular instance of the problem, each edge cost is independently perturbed by adding random noise with mean 0 and variance v, for v varying from 0 to 2^{20}.

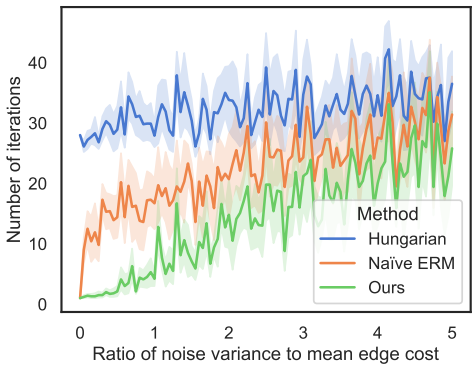
Instances of the second kind, called *geometric*, are generated from five real-world datasets available in the UCI Machine Learning repository (Buza, 2014; Blackard, 1998; Stolfo et al., 1999; Shuttle; Bhatt & Dhall, 2009). Each of these five datasets is treated as a set of n points in $\mathbb{R}^d$. First, the set is split at random into two parts, then k-means clustering with $k = 500$ is computed for each part. Each cluster corresponds to a vertex in the input to the bipartite matching problem. Then, to generate a particular instance of the problem, $2k$ points are sampled, one from each cluster on each side, and the cost of every edge is set to the Euclidean distance between the corresponding points.

We add a third kind of *structured* instances, which better highlight shortcomings of the naïve ERM approach. Our starting point is the synthetic type model described above, with $n = \ell = 50$. We initialize edge costs to $1000 + \mathrm{Geom}(1/1000)$, then plant a matching of size n with lower edge costs, set to $100 + \mathrm{Geom}(1/100)$, and a random sparse set of $n/2$ edges with very low cost $10 + \mathrm{Geom}(1/10)$. Again, to generate a particular instance, each edge cost is perturbed by noise with variance v, for v between 0 and 10 000. Even though in such generated instances the min-cost perfect matching is, usually, the same planted matching, when naïve ERM solves them independently,

[7] Dinitz et al. (2021) also report on experiments with different parameters n and ℓ in their appendix. However, $n = 500$ and $\ell = 50$ is their main setting, reported in the main body of the paper, and their results were consistent across the different settings. In order to keep the experimental compute resources low, we focus only on this main setting.



(a) Type model synthetic data, variance $v \in [0, 300]$ increasing linearly.



(b) Structured synthetic data, variance $v \in [0, 10\,000]$.

Figure 2: Number of iterations (the lower the better) of the classical Hungarian algorithm and its learning-augmented version equipped with predictions learned with naïve ERM (as in (Dinitz et al., 2021)) and with our learning algorithm (Section 2.1). The error bars represent a 95% CI.

it converges to different duals because initial iterations of the Hungarian method are influenced by the cheapest edges, which are more affected by noise yet eventually immaterial to the optimal solutions.

Experimental setup and evaluation metrics. We follow the original experimental procedure of Dinitz et al. (2021). For each considered distribution (i.e., either one of the two synthetic kinds with some variance parameter v, or the geometric kind derived from one of the five real world datasets), we first sample 20 training instances from this distribution and feed them to both learning algorithms – the median-based naïve ERM used in (Dinitz et al., 2021, Section 3.3), and our algorithm from Section 2.1 – in order to obtain two prediction vectors. Then, we sample 10 test instances, and solve each of them with each of the three methods: the classical Hungarian algorithm, and the warm-started Hungarian algorithm of Dinitz et al. (2021) with one prediction vector and with the other. Following the original procedure, we measure and report the number of iterations it took the Hungarian algorithm to reach an optimum. Each data point in Figure 2 and in Figure 5 in Appendix F, which contains further results, represents the mean over the 10 test runs, error bars correspond to 95% confidence intervals. All our experiments completed within 24 hours on a modern consumer-grade laptop with 16 GB of RAM using a single CPU core.

Results. We compare our algorithm with the naïve ERM approach on two types of datasets.

On the original datasets from (Dinitz et al., 2021), our algorithm performs at least as well as the naïve ERM approach. These datasets are highly randomized, and the naïve ERM already performs very well in this regime. As already noted in (Dinitz et al., 2021), warm-starting the Hungarian algorithm with such predictions speeds it up considerably, unless the noise in the distribution exceeds the mean edge cost by an order of magnitude, at which point there is essentially no useful signal to learn. Consequently, the results on these datasets (see Figure 2a and the additional figures in Appendix F) are important for a fair comparison with (Dinitz et al., 2021), but do not yet highlight the advantage of our approach.

The benefit of our approach becomes substantially more pronounced on the structured dataset. In this setting, the Hungarian algorithm may converge to many different optimal dual solutions, effectively jeopardizing the naïve ERM approach of Dinitz et al. (2021). Although the naïve ERM still improves over the vanilla cold-start Hungarian algorithm, our learning algorithm is able to exploit more of the structure in the distribution and yields significant additional improvements; see Figure 2b.

AI USE STATEMENT

In this work, we used generative AI tools to polish writing and reformulate several paragraphs in the paper. We have not used generative AI tools for any other tasks. In particular, we have not used AI for proving mathematical claims. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT.

Complete proofs of our theoretical results are provided in the main text and the appendices. Section 4 describes the datasets, experimental setup, evaluation metric, and computational resources used in our experiments, with additional experimental results reported in Appendix F. All code and data needed to reproduce our experiments are included in the supplementary material.

REFERENCES

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows – theory, algorithms and applications*. Prentice Hall, 1993. ISBN 978-0-13-617549-0.

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

Aaron Bernstein, Danupon Nanongkai, and Christian Wulff-Nilsen. Negative-weight single-source shortest paths in near-linear time. *J. ACM*, 72(4):23:1–23:34, 2025. doi: 10.1145/3742890. Announced at FOCS 2022.

Rajen Bhatt and Abhinav Dhall. Skin Segmentation. UCI Machine Learning Repository, 2009.

Jock Blackard. Covertype. UCI Machine Learning Repository, 1998.

Karl Bringmann, Alejandro Cassis, and Nick Fischer. Negative-weight single-source shortest paths in near-linear time: Now faster! In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pp. 515–538. IEEE, 2023. doi: 10.1109/FOCS57990.2023.00038.

Krisztian Buza. BlogFeedback. UCI Machine Learning Repository, 2014.

Sergio Cabello, Timothy M. Chan, and Panos Giannopoulos. Delaunay triangulations with predictions. In *17th Innovations in Theoretical Computer Science Conference, ITCS 2026*, LIPIcs, pp. 31:1–31:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026. doi: 10.4230/LIPICS.ITCS.2026.31.

Ivan Carvalho and Ramon Lawrence. LearnedSort as a learning-augmented SampleSort: Analysis and parallelization. In *Proceedings of the 35th International Conference on Scientific and Statistical Database Management, SSDBM 2023*, pp. 2:1–2:9. ACM, 2023. doi: 10.1145/3603719.3603731.

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*, pp. 3583–3602. PMLR, 2022a. URL https://proceedings.mlr.press/v162/chen22v.html.

Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pp. 612–623. IEEE, 2022b. doi: 10.1109/FOCS54457.2022.00064.

Zhiyang Chen, Hailong Yao, and Xia Yin. Minimum-cost network flow with dual predictions. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence, AAAI 2026*, pp. 36820–36828. AAAI Press, 2026. doi: 10.1609/AAAI.V40I43.41008.

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*, pp. 7231–7248. PMLR, 2023. URL https://proceedings.mlr.press/v202/davies23b.html.

Sami Davies, Sergei Vassilvitskii, and Yuyan Wang. Warm-starting push-relabel. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/39ec972afab01e0d8ddc6834a9d12ac1-Abstract-Conference.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*, pp. 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.

Ran Duan and Hsin-Hao Su. A scaling algorithm for maximum weight matching in bipartite graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pp. 1413–1424. SIAM, 2012. doi: 10.1137/1.9781611973099.111.

Franziska Eberle, Felix Hommelsheim, Alexander Lindermayr, Zhenwei Liu, Nicole Megow, and Jens Schlöter. Accelerating matroid optimization through fast imprecise oracles. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/aa76039e10ccb71d2922e87a87240f72-Abstract-Conference.html.

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.

M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN 0-7167-1044-7.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988. ISBN 978-3-642-97883-8. doi: 10.1007/978-3-642-97881-4.

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

Misha Khodak, Maria-Florina Balcan, Ameet Talwalkar, and Sergei Vassilvitskii. Learning predictions for 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/17061a94c3c7fda5fa24bbdd1832fa99-Abstract-Conference.html.

Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109.

Silvio Lattanzi, Ola Svensson, and Sergei Vassilvitskii. Speeding up Bellman Ford via minimum violation permutations. In *International Conference on Machine Learning, ICML 2023*, Proceedings of Machine Learning Research, pp. 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 30 April 2026.

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.

Michael Mitzenmacher. A model for learned Bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 462–471, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/0f49c89d1e7298bb9930789c8ed59d48-Abstract.html.

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

Kazuo Murota. *Discrete Convex Analysis*. SIAM, 2003.

Helia Niaparast, Benjamin Moseley, and Karan Singh. Faster global minimum cut with predictions. In *Forty-second International Conference on Machine Learning, ICML 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/niaparast25a.html.

Laurent Perron and Vincent Furnon. OR-tools, 2025. URL https://developers.google.com/optimization/. Release v9.12.

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

James Renegar. *A mathematical view of interior-point methods in convex optimization*. MPS-SIAM series on optimization. SIAM, 2001. ISBN 978-0-89871-502-6.

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*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/844e61124d9e1f58632bf0c8968ad728-Abstract-Conference.html.

Shinsaku Sakaue and Taihei Oki. Rethinking warm-starts with predictions: Learning predictions close to sets of optimal solutions for faster L-/$L^\natural$-convex function minimization. In *International Conference on Machine Learning, ICML 2023*, Proceedings of Machine Learning Research, pp. 29760–29776. PMLR, 2023. URL https://proceedings.mlr.press/v202/sakaue23a.html.

Alexander Schrijver. *Combinatorial optimization. Polyhedra and efficiency.* Springer, 2003. URL https://link.springer.com/book/9783540443896.

Shuttle. Statlog (Shuttle). UCI Machine Learning Repository.

Sandeep Silwal, Sara Ahmadian, Andrew Nystrom, Andrew McCallum, Deepak Ramachandran, and Seyed Mehran Kazemi. Kwikbucks: Correlation clustering with cheap-weak and expensive-strong signals. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=p0JSSa1AuV.

Salvatore Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip Chan. KDD Cup 1999 Data. UCI Machine Learning Repository, 1999.

A SMALL ℓ_1 DISTANCE FROM THE CONVEX HULL OF INTEGRAL OPTIMA DOES NOT IMPLY A GOOD PREDICTION.

We show an example of a fractional prediction $\hat{y}$ for bipartite min-cost perfect matching whose ℓ_1 distance to the convex hull of the integral dual optimum solutions is bounded by an arbitrarily small $\epsilon > 0$, while its ℓ_1 distance from the closest optimal integral solution is $\Omega(n)$. While both (Khodak et al., 2022) and (Sakaue & Oki, 2023) propose methods for rounding such a prediction to an integral one, we show that both would produce an integral prediction with ℓ_1 distance $\Omega(n)$ from the closest optimal integral dual solution despite $\hat{y}$ being so close to the convex hull of the optimal solutions. Therefore, such a prediction is not useful for (Dinitz et al., 2021) as they achieve speed-up over the Hungarian method only with predictions with ℓ_1 error up to $o(\sqrt{n})$.

Consider a bipartite graph $G = (L \sqcup R, E)$ on $2n$ vertices, with unit edge costs $c_e = 1$ for each $e \in E$, which contains a unique perfect matching M. Consider the following two optimal dual solutions y', y'': For each $v \in L$ we have $y'_v = 1$ and $y''_v = 0$, and for each $v \in R$, we have $y'_v = 0$ and $y''_v = 1$.

Consider vector $y^\circ := 0.5y' + 0.5y''$, i.e., the vector with all coordinates equal to 0.5, which belongs to the convex hull of y' and y''. Choose $\sigma \in \{-1, 1\}^{L \cup R}$ whose coordinates are chosen uniformly and independently at random. Then $\hat{y} = y^\circ + \frac{\epsilon}{2n}\sigma$ is not a feasible dual solution with probability at least $1 - (\frac{3}{4})^n$ and its ℓ_1 distance to $y^\circ \in \mathrm{conv}(y', y'')$ is ϵ. On the other hand, its distance to the closest integral optimal dual solution is $\Omega(n)$.

(Khodak et al., 2022) propose rounding the coordinates of $\hat{y}$ to the closest integer. This way, we receive a random 0/1 vector whose ℓ_1 distance from any dual feasible solution is $\Omega(n)$: in expectation, a quarter of edges in M will have both endpoints equal to 1, violating the corresponding constraint, i.e., it differs from any feasible dual solution in at least one coordinate per each violated edge in M.[8]

As for (Sakaue & Oki, 2022), they denote by δ the maximum violation over all dual constraints, and they decrease all dual variables in $\hat{y}$ by δ. In our example, $\delta = \epsilon/n$, and after this step we end up with a feasible fractional dual solution $\hat{y}' \in [0, 0.5)^{L \cup R}$. Then, they round the coordinates of $\hat{y}'$ to the closest integer, which gives in our case the all-zeros vector, whose ℓ_1 distance from the closest optimal dual is $\Omega(n)$.

To conclude, we have constructed a fractional prediction $\hat{y}$ very close to the convex hull of the integral optimal dual solutions. Despite this, $\hat{y}$ is very far from the optimal integral solutions and does not contain any useful information about their structure (in this case, they encode the bipartition of the input graph). Therefore, when rounded by (Khodak et al., 2022) and (Sakaue & Oki, 2023), we receive an integral prediction with ℓ_1 distance $\Omega(n)$ from any optimal dual integral solution, which is not a useful prediction for the algorithm by (Dinitz et al., 2021).

B PROOF OF LEMMA 2

Proof. Let $N(v)$ be the set of neighboring vertices of v in the (undirected) input graph $G = (V, E)$, that is $N(v) = \{u \in V : uv \in E\}$. With this notation, we state the following linear program that is the dual of (D).

[8] In fact, this is in accordance with the statement in (Khodak et al., 2022) that the rounding of $\hat{y}$ will have ℓ_1 distance to y' of order $O(\|\hat{y} - y'\|) = O(n/2)$.

$$
\begin{aligned}
(P) \qquad \text{minimize} \quad & \sum_{j=1}^{k} \sum_{uv \in E} c_{uv}^{j} x_{uv}^{j} \\
\text{subject to} \quad & \sum_{j=1}^{k} (a_v^j - b_v^j) = 0 && \forall\, v \in V && (8)\\
& \left(\sum_{u \in N(v)} x_{uv}^{j} \right) - a_v^j + b_v^j = M && \forall\, j \in [k], \forall\, v \in V && (9)\\
& a_v^j \leq 1 && \forall\, j \in [k], \forall\, v \in V && (10)\\
& b_v^j \leq 1 && \forall\, j \in [k], \forall\, v \in V && (11)\\
& x_{uv}^j, a_v^j, b_v^j \geq 0 && && (12)
\end{aligned}
$$

We claim that the constraints of (P) correspond to the flow-conservation constraints in network (F) up to multiplying the constraints (9) and (8) corresponding to the vertices $v \in L$ by -1.

First, we make the following observation. In order to have a flow of value $\lambda = \frac{1}{2}knM$ in the network (F), necessarily the flow on all arcs leaving s, and on all arcs entering t, must be equal to the capacity M. Hence, we can consider these flow values as fixed.

With this observation, note that constraints (9) are the flow conservation constraints for vertices in $L^j \cup R^j$ for $j = 1, \ldots, k$. Consider any $v^j \in R^j$ for some $j \in \{1, \ldots, k\}$. The incoming flow is equal to $b_v^j + \sum_{u \in N(v)} x_{uv}^j$ while the outgoing flow is equal to $M + a_v^j$. Therefore, the flow conservation constraint for v^j is

$$\sum_{u \in N(v)} x_{uv}^j + b_v^j - a_v^j - M = 0,$$

which is equal to constraint (9) corresponding to v^j. Similarly, consider any $u^j \in L^j$ for some $j \in [k]$. The incoming flow is equal to $M + a_u^j$ while the outgoing flow is equal to $\sum_{v \in N(u)} x_{uv}^j + b_u^j$. Therefore, the flow conservation constraint for u^j is

$$-\sum_{v \in N(u)} x_{uv}^j - b_u^j + a_u^j + M = 0,$$

which is equal to constraint (9) corresponding to u^j multiplied by -1.

Second, constraints (8) are the flow conservation constraints from vertices in $L \cup R$. For each $v \in R$, the incoming flow to v is equal to $\sum_{j=1}^{k} a_v^j$ while the outgoing flow is equal to $\sum_{j=1}^{k} b_v^j$. Therefore the flow conservation constraint for v is as stated in (8). For each $u \in L$, the outgoing flow to u is equal to $\sum_{j=1}^{k} a_u^j$ while the incoming flow is equal to $\sum_{j=1}^{k} b_u^j$. Therefore the flow conservation constraint for u is (8) multiplied by -1.

Finally, (10) and (11) are the capacity constraints on edges incident to the vertices in $L \cup R$. Therefore, all the constraints in (P) are, up to multiplication of the constraints corresponding to vertices $v \in L \cup L^1 \cup \cdots \cup L^k$ by -1, constraints of the min-cost flow problem (F). □

C Proof of Theorem 5

Proof. We prove the NP-hardness by a reduction from the well-known *feedback arc set* (FAS) problem (Karp, 1972). In FAS, we are given a directed graph $G = (V, E)$ and an integer k, and we need to decide if we can remove at most k edges from E to make G acyclic. We let n and m be the number of nodes and arcs of G, respectively. Furthermore, without loss of generality, we can assume $2k < m$, since if $2k \geq m$ the answer is always yes by picking any ordering and removing either forward or backward edges.

We create a bipartite graph with n vertices on each side, that is, with parts $\bar{V} = \{\bar{v} \mid v \in V\}$ and $V' = \{v' \mid v \in V\}$. The graph has every possible edge between $\bar{V}$ and V'. For each oriented edge e

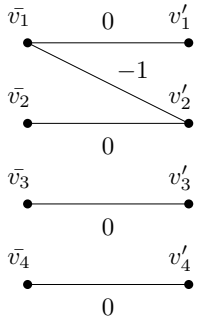


Figure 3: The created instance for the edge e from v_1 to v_2 in a 4 vertex graph G. All other edges have cost $n = 4$.

in the input graph G, we create a cost vector c^e as follows:

$$c^e_{\bar{u}v'} = \begin{cases} 0 & \text{if } u = v, \\ -1 & \text{if } e \text{ goes from } u \text{ to } v, \\ n & \text{otherwise.} \end{cases}$$

Note that all instances have optimal dual value at most 0 since $y_{\bar{w}} + y_{w'} \leq 0$ for all $w \in V$. Also, value 0 can easily be reached: the solution y with all coordinates equal to 0 except for $y_{\bar{v}} = -y_{v'} = 1$ is feasible with respect to the cost vector c^e, see Figure 3.

We remark that the cost vectors can be equivalently turned nonnegative by adding 1 to every cost and every dual value $y_{v'}$ for $v' \in V'$.

The reduction is clearly polynomial. All that remains to prove is that the solution to our problem on the bipartite graph with the m cost vectors decides the FAS instance. We claim that there is a prediction vector $\hat{y}$ with expected error at most $2k/m$ if and only if there is a FAS of size at most k.

($\Leftarrow$) Suppose there is a FAS S with $|S| \leq k$. Since $G - S$ is acyclic, there is a topological ordering $v_1, \ldots, v_n$ of $G - S$. Consider the prediction vector $\hat{y}$ with $\hat{y}_{\bar{v}_i} = -\hat{y}_{v'_i} = i$. Note that for all $e = v_i v_j \in E \setminus S$ we have $i < j$, which implies $i + (-j) \leq -1$. Hence, $\hat{y}$ satisfies the dual constraints for c^e for all $e \in E \setminus S$ and reaches optimal value 0, so it is an optimal solution.

For $e = v_i v_j \in S$ and the cost vector c^e, consider the optimal solution

$$y_{\bar{v}_\ell} = -y_{v'_\ell} = \begin{cases} 0 & \text{if } \ell = i, \\ \ell & \text{otherwise.} \end{cases}$$

Clearly $||\hat{y} - y||_0 = 2$. Thus, in total, we have

$$\frac{1}{m} \sum_{e \in E} \min\{\|y^* - \hat{y}\|_0 \mid y^* \in \mathcal{Y}^*(c^e)\} \leq \frac{2|S|}{m} \leq \frac{2k}{m}.$$

($\Rightarrow$) Let $\hat{y}$ be a prediction vector with expected error at most $2k/m$. First, we claim that in $\hat{y}$, for every $v \in V$, we have $\hat{y}_{\bar{v}} = -\hat{y}_{v'}$. To see that, first observe that $y_{\bar{v}} = -y_{v'}$ for any optimal dual solution y to reach value 0. So if $\hat{y}$ does not satisfy this property for a vertex v, then either $\hat{y}_{\bar{v}} \neq y_{\bar{v}}$ or $\hat{y}_{v'} \neq y_{v'}$. Hence, the expected error will be at least $m/m = 1$, a contradiction to our assumption $2k < m$.

This property also implies that every error $\|\hat{y} - y\|_0$ to an optimal y must be even. Therefore, $\hat{y}$ must be an optimal solution in all but at most k instances. Denoting S the set of all edges $e \in E$ where $\hat{y}$ is not optimal with respect to the costs c^e, we have $|S| \leq k$. We claim that S is a FAS for G. For contradiction, suppose $G - S$ still contains a cycle $w_1, w_2, \ldots, w_\ell$. Since $\hat{y}$ is optimal for all c^e with $e \in E \setminus S$, we get the inequality

$$\hat{y}_{\bar{w}_1} < -\hat{y}_{w'_2} = \hat{y}_{\bar{w}_2} < \ldots = \hat{y}_{\bar{w}_\ell} < -\hat{y}_{w'_1} = \hat{y}_{\bar{w}_1}.$$

This is a contradiction, and therefore the chosen S of size at most k is a FAS. □

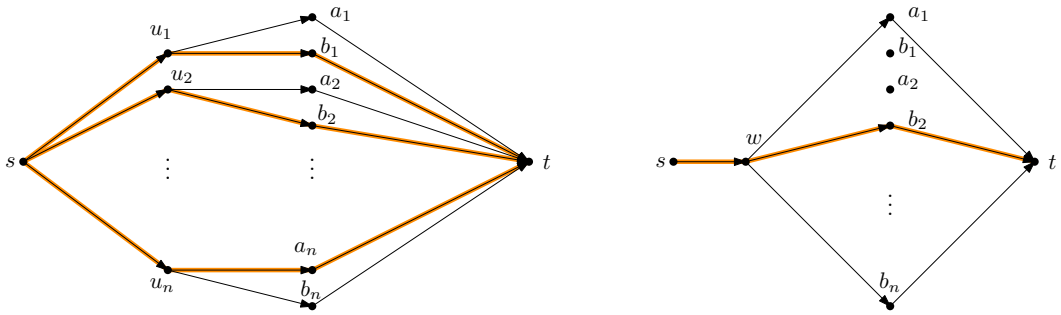


Figure 4: Flow f optimal with respect to capacities c^0 (left) which induces a satisfying assignment of the clause j, ($x_1 \vee \neg x_2 \vee \neg x_n$ in this example). The optimal flow with respect to the capacities c^j which is closest to f is on the right and overlaps with f in a single edge, making the two flows differ in $3n + 1$ edges. If f induced an assignment which does not satisfy the clause j, all optimal flows with respect to c^j would differ from f in $3n + 3$ edges.

D Proof of Theorem 7

Proof. We prove the statement by a reduction from 3SAT. Given a 3SAT formula with n variables and m clauses, the reduction creates a graph $G = (V, E)$ with $|V| = 3n + 3$ vertices and all possible $|V| \cdot (|V| - 1)$ directed edges. The vertex set contains the source s, the target t, a so-called *clause vertex* w, and three sets of n vertices each: $u_1, \ldots, u_n$, $a_1, \ldots, a_n$, and $b_1, \ldots, b_n$.

We construct a sequence of capacity vectors of two types: The capacity vectors of Type A encode all possible assignments to the variables in the given 3SAT formula. The capacity vectors of Type B encode individual clauses of the formula.

The Type A capacity vectors are all identical copies of the vector $c^0 \in \{0, 1\}^{|E|}$ defined as follows. For each edge $e \in E$, we have $c^0_e = 0$ unless e belongs to one of the following three categories, in which case $c^0_e = 1$:

- e goes from s to u_i, for some $i \in [n]$,
- e goes from u_i to a_i or from a_i to t, for some $i \in [n]$, or
- e goes from u_i to b_i or from b_i to t, for some $i \in [n]$.

See Figure 4 left. The maximum flow value with respect to the capacity vector c^0 is n, and there are exactly 2^n different integral flows with this value, each using $3n$ edges. Indeed, for each i the path from s to t can go through either a_i or b_i. Since $e = (s, u_i)$ has capacity 1 and we only consider integer flows, only one of these options is possible. Intuitively, each such solution corresponds to a 3SAT assignment, where the i-th variable is true if the flow chooses a_i or false if it chooses b_i.

Each capacity vector of Type B corresponds to a clause of the input 3SAT formula. For $j \in [m]$, we have $c^j_e = 0$ for every $e \in E$ unless e belongs to the following categories:

- e goes from s to w,
- e goes from w to a_i or from a_i to t, and the j-th clause contains the positive literal x_i,
- e goes from w to b_i or from b_i to t, and the j-th clause contains the negative literal $\neg x_i$.

See Figure 4 right. For each $j \in [m]$, capacity vector c^j admits maximum flow of value 1, and there are exactly three different optimal solutions, since there is one path from s to t of capacity 1 for each of the three literals in the clause. If an integral flow f that is optimal with respect to c^0 induces a satisfying assignment of the clause j, there is an optimal flow with respect to c^j which uses one edge which is also used by f. In this case, the ℓ_1 distance between them equals $3n + 1$. Otherwise, none of the flows optimal with respect to c^j uses any edge which is also used by f. In such a case, their ℓ_1 distance is equal to $3n + 3$.

Let $a := m(3n+1)$. We construct a sequence of $k = a + 1 + m$ capacity vectors as follows: The first $a+1$ vectors are of Type A, i.e., identical copies of c^0. The remaining m vectors are of Type B, each corresponding to one of the m clauses in the input 3SAT formula. We now argue that the existence of an integral prediction $\hat{f}$ with an average ℓ_1 error at most $a/k = 1 - O(1/n)$ is equivalent to the satisfiability of the input 3SAT formula.

(⇒) Assume that the learning problem instance admits a solution $\hat{f}$ with error at most a/k. We construct a satisfying 3SAT assignment $x \in \{0,1\}^n$. First, note that $\hat{f}$ must be optimal with respect to the capacity vector c^0. Otherwise, $\hat{f}$ would have error at least 1 in all $a+1$ (identical) input instances of Type A, implying the average ℓ_1 error of at least $(a+1)/k$. Hence, $\hat{f}$ encodes a truth assignment of the variables in the 3SAT formula: for each $i \in [n]$, $x_i = 1$ if $\hat{f}$ passes through a_i and $x_i = 0$ if it passes through b_i.

Now, we show that x is a satisfying assignment. Recall that $\hat{f}$ has average ℓ_1 error of at most $a/k = m(3n+1)/k$. Therefore, for every $j \in [m]$, it must overlap with an optimal flow with respect to c^j, having ℓ_1 error $3n+1$ instead of $3n+3$. This means that x induced by $\hat{f}$ is a satisfying assignment for each clause.

(⇐) Assume the 3SAT formula is satisfiable with assignment $x_1, \ldots, x_n$. Define $\hat{f}$ by assigning flow value 1 to the edges su_i for all $i \in [n]$ and $u_i a_i$, $a_i t$ if $x_i = 1$ or $u_i b_i$, $b_i t$ otherwise. This way, $\hat{f}$ is a maximum flow for each of the first $a+1$ identical capacity vectors and they contribute 0 to the error. Since each clause is satisfied by some literal corresponding to a variable x_i, each of the last m instances contributes only $(3n+1)/k$ to the error and the total error is a/k. □

E Proof of Theorem 9

Proof. The reduction creates a new flow network G' by modifying the input graph $G = (V, E)$. Each original edge $uv \in E$ is replaced by $k+1$ parallel edges. First we reorder the capacities of uv in each instance so that $c^{j_1}_{uv} \le c^{j_2}_{uv} \le \cdots \le c^{j_k}_{uv}$. The costs and capacities of the new edges will be as follows: The first edge has capacity $c^{j_1}_{uv}$ and cost 0, since flow up to this size does not violate capacity constraints in any input instance. For each $i \in [k-1]$, we add an edge from u to v with capacity $c^{j_{i+1}}_{uv} - c^{j_i}_{uv}$ and cost i, and the last edge has capacity $+\infty$ and cost k. At the end, we decrease the cost of every outgoing edge from s by k, so that the cost of a flow of value x passing from u to v is

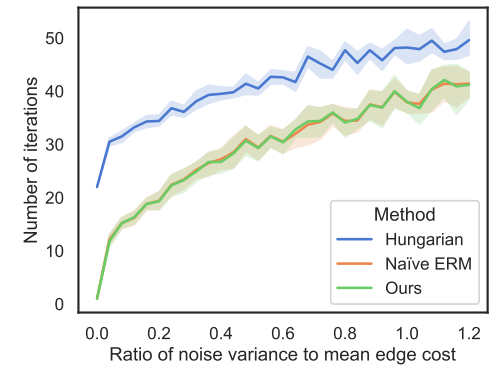
$$\mathrm{cost}_{uv}(x) = -\mathbf{1}_{[u=s]} \cdot kx + \sum_{i \in [k]} \max\{0, x - c^i_{uv}\}.$$

Let E' denote the set of those edges over all $uv \in E$. Moreover, for each node $u \in V \setminus \{s,t\}$, we add two edges su and ut with capacity $+\infty$ and cost k. Let E'' denote the set of these edges.

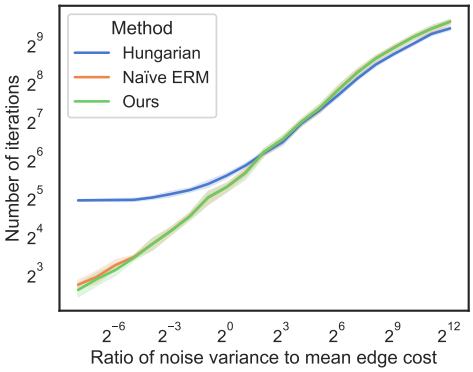
Given a min-cost s-t flow f^* of arbitrary value in $(V, E' \cup E'')$, we construct the prediction vector $\hat{f}$ in $G = (V, E)$ as follows: For each edge $uv \in E$, $\hat{f}_{uv}$ is chosen as the sum of the flow through the $k+1$ edges from u to v in E' according to f^*. This way, $\sum_{i \in [k]} (\mathrm{over}(c^i, \hat{f}) - v(\hat{f}))$ is equal to the cost of f^* incurred on the edges in E' and $k(\mathrm{ex}(\hat{f}) + \mathrm{def}(\hat{f}))$ is the cost of f^* incurred on the edges in E''. Therefore, since $\sum_{i \in [k]} \mathrm{OPT}(c^i)$ is constant, $\hat{f}$ minimizes the sum in (7). □

F Further empirical results

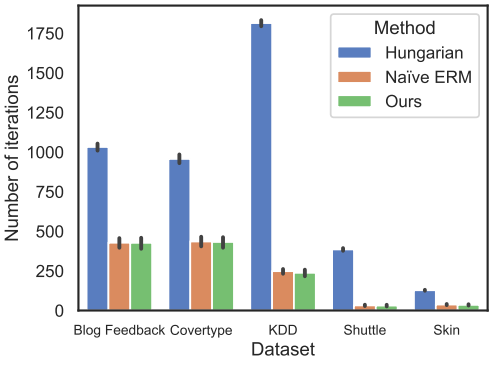
Here, we show further empirical results on the original datasets used in (Dinitz et al., 2021), see Figures 5b and 5c. To allow more direct comparison, we also reproduce Figures 5a and 5d, which are already included in the main body of the paper (2a and 2b). Similar to Figure 2a (or 5a), Figures 5b and 5c show that our algorithm offers comparable performance to the naïve ERM approach in (Dinitz et al., 2021) on the original datasets used by Dinitz et al. (2021).



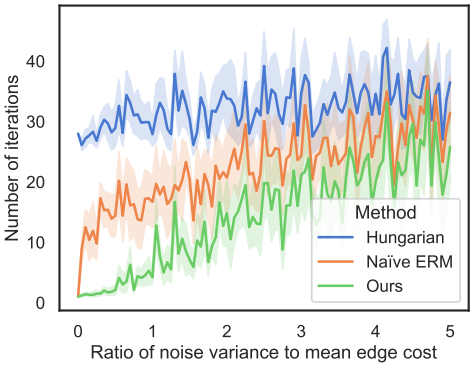
(a) Type model synthetic data, variance $v \in [0, 300]$ increasing linearly, linear scale.



(b) Type model synthetic data, variance $v \in [2^0, 2^{20}]$, increasing geometrically, logarithmic scale.



(c) Geometric real world data.



(d) Structured synthetic data, variance $v \in [0, 10\,000]$.

Figure 5: Number of iterations (the lower the better) of the classical Hungarian algorithm and its learning-augmented version equipped with predictions learned with naïve ERM (as in (Dinitz et al., 2021)) and with our learning algorithm (Section 2.1). The error bars represent a 95% CI.