Learning-Augmented Maximum Flow

Adam Polak[a,b,*], Maksym Zub[b]

[a] *Max-Planck Institute for Informatics, Saarland Informatics Campus, 66123, Saarbrucken, Germany*
[b] *Jagiellonian University, ul. Lojasiewicza 6, 30-348, Krakow, Poland*

Abstract

We propose a framework for speeding up maximum flow computation by using predictions. A prediction is a flow, i.e., an assignment of non-negative flow values to edges, which satisfies the flow conservation property, but does not necessarily respect the edge capacities of the actual instance (since these were unknown at the time of learning). We present an algorithm that, given an m-edge flow network and a predicted flow, computes a maximum flow in $O(m\eta)$ time, where η is the ℓ_1 error of the prediction, i.e., the sum over the edges of the absolute difference between the predicted and optimal flow values. Moreover, we prove that, given an oracle access to a distribution over flow networks, it is possible to efficiently PAC-learn a prediction minimizing the expected ℓ_1 error over that distribution. Our results fit into the recent line of research on learning-augmented algorithms, which aims to improve over worst-case bounds of classical algorithms by using predictions, e.g., machine-learned from previous similar instances. So far, the main focus in this area was on improving competitive ratios for online problems. Following Dinitz et al. (NeurIPS 2021), our results are among the firsts to improve the running time of an offline problem.

Keywords: combinatorial optimization, maximum flow, algorithms with predictions

1. Introduction

Computing a maximum s-t flow in a flow network (i.e., in a directed graph with nonnegative edge capacities and designated source and sink nodes) is a basic problem in combinatorial optimization. It is a building block of a number of more advanced algorithms, with relevance both in theory (e.g., in graph algorithms and scheduling) and in practice (e.g., in computer vision and transport).

Imagine we are to solve multiple similar instances of the maximum flow problem, e.g., the instances are drawn at random from a distribution, or they are snapshots of a single underlying instance changing over time. Can we learn what a typical optimal solution looks like, and then use it to speed up further computations? Or, to put it differently, assume we have a solution – e.g., obtained from past data or computed by a very fast heuristic – that is not necessarily optimal, maybe not even feasible, but close to an optimal solution. How can we use such an imperfect solution to warm-start a maximum flow algorithm and get a better running time?

Warm-starting maximum flow algorithms has been studied in the past experimentally (e.g., in computer vision, where maximum flow is often used to compute minimum cuts in subsequent frames of a video [19, 18]). In contrast, we propose an approach with theoretical guarantees.[1]

Learning-augmented algorithms (also called *algorithms with predictions*) are the subject of a recent line of research that aims to improve over worst-case bounds of classical algorithms by using possibly imperfect predictions. So far, the main focus in this area was on improving competitive ratios for online problems. Dinitz et al. [6] took a first step to explore improving the running times of offline problems. They gave an algorithm for the weighted bipartite matching problem that uses a learned dual solution to improve over the running time of the classic Hungarian algorithm. Our approach draws inspiration from their work, but it differs significantly in two aspects. First, we learn a primal, not a dual solution. Second, we impose an additional restriction on the learned solution, namely the flow conservation prop-

*Corresponding author
Email addresses: apolak@mpi-inf.mpg.de (Adam Polak), max.zub@student.uj.edu.pl (Maksym Zub)

[1] Interestingly, while the theoretical guarantees of our algorithms are linear in the ℓ_1 error of flow predictions, the empirical performance of the heuristic warm-starting approaches [19, 18] happens to be roughly linear in the ℓ_0 change in edge capacities.

erty. This restriction makes our learning problem harder and the subsequent algorithmic problem easier. In Section 1.2 we discuss these differences in greater depth.

1.1. Our results

We propose a framework for speeding up maximum flow computation by using predicted flow values. Here, by *prediction* we mean a flow, which satisfies the flow conservation property, but does not necessarily respect the edge capacities of the actual instance (since these were unknown at the time of learning). We present an algorithm that, given a flow network $G = (V, E)$ with edge capacities $c \in \mathbb{Z}^E_{\geqslant 0}$, and a predicted flow $f \in \mathbb{Z}^E_{\geqslant 0}$, computes a maximum flow $f^*_c \in \mathbb{Z}^E_{\geqslant 0}$ in $O(|E| \cdot \eta)$ time, where $\eta = \|f - f^*_c\|_1 = \sum_{e \in E} |f(e) - f^*_c(e)|$ is the ℓ_1 error of the prediction. Moreover, we prove that, given an oracle access to a (joint) distribution over edge capacities, it is possible to efficiently PAC-learn a prediction minimizing the expected ℓ_1 error over that distribution.

To formally state our results, let us first define the maximum flow problem and related concepts.

Definition 1. Given a directed graph $G = (V, E)$, *source* and *sink* nodes $s, t \in V$, and nonnegative integral edge *capacities* $c : E \to \mathbb{Z}_{\geqslant 0}$, the *maximum flow problem* asks to find a function $f : E \to \mathbb{Z}_{\geqslant 0}$, assigning nonnegative integral *flow* to the edges, that satisfies

- *capacity constraints*: $\forall_{e \in E}\ f(e) \leqslant c(e)$, and
- *flow conservation*:

$$\forall_{v \in V \setminus \{s,t\}} \sum_{u\,:\,(u,v) \in E} f(u, v) = \sum_{u\,:\,(v,u) \in E} f(v, u)$$

,

and maximizes the *flow value* defined as $\mathrm{val}(f) = \sum_{(s,u) \in E} f(s, u)$. We denote by f^*_c a maximum flow for given capacities c.

Let us note that we have made the decision to focus on the integral version of the problem for two reasons. First, in many applications edge capacities are integral anyway, and hence there always exists an integral solution as well, see, e.g., [1]. Second, the error measure we work with, namely the ℓ_1 distance, is meaningless if one allows arbitrary scaling without changing the problem, as it would be the case for rational edge capacities. Finally, we remark that all the fastest maximum flow algorithms, including the almost-linear time one [4], are only *weakly polynomial*, i.e., their running times depend polylogarithmically on the maximum edge capacity value and, in particular, they work only with integral edge capacities.

In Section 2, we prove the following theorem giving an algorithm that can be seen as the Ford-Fulkerson method with a warm start.

Theorem 2. *Given a directed graph $G = (V, E)$, source and sink $s, t \in V$, edge capacities $c : E \to \mathbb{Z}_{\geqslant 0}$, and a predicted flow function $f : E \to \mathbb{Z}_{\geqslant 0}$ satisfying flow conservation, one can compute a maximum (s, t)-flow in G, in time*

$$O(|E| \cdot \|f - f^*_c\|_1).$$

Note that the above bound holds simultaneously for every maximum flow f^*_c, which might not be unique. In other words, the prediction is good if it is close to at least one optimal solution.

One of the sought-after properties of learning-augmented algorithms is *robustness*, i.e., retaining worst-case guarantees of classic algorithms even for arbitrarily bad predictions. However, in the case of running time bounds, robustness comes essentially for free (up to a multiplicative factor of 2, vanishing in the asymptotic notation). Indeed, one can always run step-by-step an algorithm with predictions alongside the fastest known classic algorithm, stopping when either of them stops. Therefore, Theorem 2 paired with the recent $O(|E|^{1+o(1)})$ time algorithm for the maximum flow problem [4] actually leads to a robust learning-augmented algorithm with running time

$$O(|E| \cdot \min\{\|f - f^*_c\|_1, |E|^{o(1)}\}).$$

Naturally, a similar bound but with the ℓ_∞ error (instead of the ℓ_1 error) would be more desirable, but let us argue that such a bound is unlikely. Indeed, the maximum flow problem with unit edge capacities is not known to be significantly easier than the general problem, yet for that special case the ℓ_∞ prediction error can trivially be bounded by a constant, and hence a meaningful bound of the desired form would mean that it is an easier problem. Let us also remark that our choice of the ℓ_1 distance as the prediction error metric is consistent with related works [6, 3].

The proof of Theorem 2 that we give at the beginning of Section 2 uses a simple algorithm that calls the Ford-Fulkerson method (see Section 1.2) as a black-box but also spends $O(|E| \cdot \|f - f^*_c\|_1)$ time on top of that. Later, in Section 2.1, we prove the following stronger result that implies Theorem 2.

Theorem 3. *Given a directed graph $G = (V, E)$, source and sink $s, t \in V$, edge capacities $c : E \to \mathbb{Z}_{\geqslant 0}$, and a predicted flow function $f : E \to \mathbb{Z}_{\geqslant 0}$ satisfying flow conservation, one can reduce computing a maximum (s, t)-flow in G to computing maximum flows in two*

graphs, each with $O(|E|)$ edges and the maximum flow value bounded by $O(\|f - f_c^*\|_1)$. The reduction works in $O(|E|)$ time.

Composing Theorem 3 with the Ford-Fulkerson method gives the same worst-case running time bound as Theorem 2. However, the alternative approach has the advantage that all the computations that take more than linear time can be delegated to one of many available highly optimized implementations of maximum flow algorithms, and therefore this approach might be more efficient in practice

Next, we want to argue that predictions required by the above algorithm can be efficiently learned, in a PAC-learning sense. We assume that the underlying graph, as well as the choice of the source and sink nodes, are fixed. (This assumption is almost without loss of generality, because one can take the underlying graph to be a clique, with capacities zero for nonexistent edges; that, however, may cause a running time overhead, because of the increased number of edges.) Our goal is to prove that, given a joint distribution over edge capacities, we can efficiently learn a flow approximately minimizing the expected ℓ_1 error over that distribution. We do it in two steps. First, in Section 3, we prove Theorem 4, giving an algorithm that finds a prediction that minimizes the error on a given set of samples. It works under the assumption[2] that $f_{c_i}^*$'s are unique (or, arbitrarily fixed) solutions to the maximum flow problem instances given by capacities c_i's. Then, in Section 4, we prove Theorem 5, which states that if the number of samples is large enough, then such a prediction for these samples also approximately minimizes the error for the whole distribution.

Theorem 4. *Given a directed graph $G = (V, E)$, with source and sink $s, t \in V$, and a collection of k lists of edge capacities $c_1, c_2, \ldots, c_k \in \mathbb{Z}_{\geqslant 0}^E$, one can find an integral flow prediction that minimizes the prediction error on this collection, i.e.,*

$$\hat{f} = \arg\min \left\{ \frac{1}{k} \sum_{i \in [k]} \|f - f_{c_i}^*\|_1 \;\middle|\; f : E \to \mathbb{Z}_{\geqslant 0} \text{ satisfying flow conservation} \right\},$$

in time $O(T(k \cdot |E|))$, where $T(m) \leqslant m^{1+o(1)}$ denotes the min-cost flow complexity in graphs with m edges.

Theorem 5. *Let $G = (V, E)$ be a directed graph, with source and sink $s, t \in V$, and let $c_1, c_2, \ldots, c_k \in \mathbb{Z}_{\geqslant 0}^E$, for $k = \Theta(c_{\max}^2 |E|^3 \log(c_{\max}|E|))$, be independent samples from a distribution $\mathcal{D}$, where $c_{\max} = \max_{c \in \mathrm{supp}(\mathcal{D}), e \in E} c(e)$. Let $\hat{f} \in \mathbb{Z}_{\geqslant 0}^E$ be a flow that minimizes the prediction error on this collection of samples, as in Theorem 4. Then, with high probability over the choice of the samples, the expected ℓ_1 error of $\hat{f}$ over $\mathcal{D}$ is approximately minimum possible, i.e.,*

$$\mathbb{E}_{c \sim \mathcal{D}} \|\hat{f} - f_c^*\|_1 \leqslant \min_f \mathbb{E}_{c \sim \mathcal{D}} \|f - f_c^*\|_1 + O(1),$$

where the minimum is taken over functions $f \in \mathbb{Z}_{\geqslant 0}^E$ satisfying the flow conservation property.

1.2. Related work

Maximum flow algorithms. There are numerous algorithms for the maximum flow problem. The Ford-Fulkerson method [12] is a starting point for many of them, and its vanilla version runs in weakly polynomial $O(|E| \cdot \mathrm{val}(f_c^*))$ time for integral edge capacities. The strongly polynomial time algorithms, which also work for rational edge capacities, can be roughly split into three groups: augmenting paths algorithms (e.g., [9, 7, 13]), push-relabel algorithms (e.g., [14]), and pseudoflow algorithms (e.g., [15]). Each of these groups contains algorithms with running time $\widetilde{O}(|V| \cdot |E|)$ that are widely used in practice, see, e.g., [2, 11]. A long line of research on Laplacian solvers and interior-point methods, initiated by [26], culminated recently with a (weakly polynomial) almost-linear $O(|E|^{1+o(1)})$ time algorithm [4].

In the light of this new development, it may seem that our learning-augmented algorithm is only relevant for very small prediction errors, namely $\|f - f_{c_i}^*\|_1 \leqslant |E|^{1+o(1)}$. However, at this point it is not yet clear if the new almost-linear time algorithm will lead to practical developments.[3]

Learning-augmented algorithms. The idea of using predictions to improve the performance of algorithms is not a new one, see, e.g., [22]. However, the recent systematic study of such methods – under the umbrella term of *learning-augmented algorithms*, or simply *algorithms with predictions* – seems to have started with the works of Lykouris and Vassilvitskii [21], and Purohit, Svitkina, and Kumar [24]. Since then, the field developed rapidly, see [23] for a survey. So far, the majority

[2] This is a standard assumption for PAC-learnability results in the literature on (static) algorithms with predictions, see, e.g., [6, 25, 5].

[3] See https://codeforces.com/blog/entry/100510 for a relevant discussion with an author of [4].

of the works focus on online algorithms, where predictions help reduce uncertainty about the yet unseen part of the instance. There are, however, also works on, e.g., data structures [20], streaming algorithms [17], and sublinear algorithms [8]. Apart from a simple example of binary search [21], until recently there were no works on improving algorithms running times using predictions. This has changed with the work of Dinitz et al. [6], the recent followup works of Chen et al. [3], and Sakaue and Oki [25], as well as the work of Ergun et al. [10] (which is however concerned with approximation algorithms and uses a significantly different approach).

Learning-augmented weighted bipartite matching. A direct inspiration for our approach is the work of Dinitz et al. [6]. They study the maximum weighted bipartite matching problem and predict the dual[4] solution. They give a learning-augmented algorithm that solves the matching problem in $O(|E|\sqrt{|V|}\cdot\min\{\eta,\sqrt{|V|}\})$ time, where η is the ℓ_1 error of the predicted dual solution – our Theorem 2 is an analogue of that result. They also show that, given an oracle access to a joint distribution over edge weights, one can efficiently learn a prediction minimizing the expected ℓ_1 error over the distribution – our Theorems 4 and 5 are together an analogue of that result.

The most apparent difference between their approach and ours is that they use a predicted dual solution and we use a predicted primal solution. The reason they state for focusing on the dual solution is that the primal solution is very volatile to small changes in the input. Let us note that this argument clearly applies to weighted problems (in particular, e.g., to the minimum cost flow problem) but it is not clear if it also applies to the maximum flow problem. Moreover, it is also not clear if the dual solution is indeed less volatile, even for weighted problems.

The second important difference is that they do not impose any constraints on predictions, while we require that the predicted solution satisfies the flow conservation property. This difference has the following consequences. First, their learning algorithm can be very simple – the best possible prediction is just a coordinate-wise median over the solutions for the samples – while we need to solve the minimum cost flow problem instead. Second, turning a prediction into a feasible solution is also harder for us, as we want to maintain the flow conservation property. On the other hand, once we have a feasible solution, the remaining part of our maximum flow algorithm is simple and easy to analyse, in contrast with their tailored primal-dual analysis for the analogous part of their algorithm.

[4]Recall that the matching problem can be formulated as a linear program, and every linear program has a corresponding dual program.

Other algorithmic speedups using predictions. In their recent work Chen et al. [3] improve the running time of Dinitz et al. for the matching problem, and extend their framework to a couple of other graph problems: the negative weights single-source shortest paths problem, the degree-constrained subgraph problem, and the minimum cost 0-1 flow problem. For all these problems they use predicted dual solutions. They also propose general learnability theorems, which imply a result similar to Lemma 8 (see also the discussion below the proof of Lemma 8 for a comparison of such results with Theorem 5).

Sakaue and Oki [25] propose a general framework for augmenting discrete optimization problems with predictions. Their approach leads to faster algorithms for three problems: weighted bipartite matching, weighted matroid intersection, and discrete energy minimization for computer vision. It is notable that their guarantees hold with respect to the ℓ_∞ prediction error, compared to the ℓ_1 error in previous works [6, 3] and ours. Interestingly, their framework allows working with both primal and dual predictions. They analyse what properties of a problem make it more suitable for one type of predictions or the other, and they end up using primal predictions for the discrete energy minimization problem.

Ergun et al. [10] study k-means clustering, and use predictions to achieve in near-linear time approximation factors that are impossible to achieve without predictions even in polynomial time. They also propose a very general learnability framework, based on relating the VC-dimension of a class of functions to their computational complexity, which implies bounds similar to ours, but with a worse sample complexity.

Dropping flow conservation constraints. In the concurrent and independent work [5], Davies, Moseley, Vassilvitskii, and Wang propose a similar framework for speeding up maximum flow computations by using predictions. The key difference is that in their setup the prediction does not necessarily need to satisfy the flow conservation constraints. They prove analogs of our Theorems 2–5 for their notion of prediction. In particular, they also use the ℓ_1 distance between the predicted and the maximum flow as the prediction error, and they achieve the same running time as in our Theorem 2.

Their analog of our Theorem 2 also follows the two step approach consisting of a feasibility step and an optimization step. Their optimization step is identical to ours, the key technical difference lies in the feasibility step. In this step, our algorithm only needs to fix violated capacity constraints, while their algorihtm fixes both capacity and flow conservation constraints. It first fixes the capacity constraints by just decreasing the flow on oversaturated edges – possibly violating additional flow conservation constaints, which however are going to be fixed next. Then, it fixes the flow conservation constraints using a reduction to a maximum flow computation in an auxiliary graph – similar to our proof of Theorem 3.

On the other hand, learning (Theorem 4) in their setup is much easier. Since the prediction for each edge can optimized independently, it boils to down to selecting – for each edge – the median of optimal flow values along that edge among samples.

1.3. Limitation and open problem

We do prove that a prediction with a small ℓ_1 error can be used to speed up maximum flow computation, and that given a distribution over flow networks one can learn a prediction minimizing the ℓ_1 error. However, we do not answer the question of what makes a distribution have such a minimum that is actually small. There seems to be no standard approach to address this type of representation error question, and the related works [6, 10, 3] do not address it either.

2. Warm-starting Ford-Fulkerson

Theorem 2. *Given a directed graph $G = (V, E)$, source and sink $s, t \in V$, edge capacities $c : E \to \mathbb{Z}_{\geqslant 0}$, and a predicted flow function $f : E \to \mathbb{Z}_{\geqslant 0}$ satisfying flow conservation, one can compute a maximum (s,t)-flow in G, in time*

$$O(|E| \cdot \|f - f_c^*\|_1).$$

Proof. At first, the predicted flow f does not necessarily satisfy the capacity constraints imposed by c, i.e., for some edges $e \in E$ it might happen that $f(e) > c(e)$. The algorithm consists of two steps. In the first step, it turns f into $\bar{f}$ that satisfies the capacity constraints, while maintaining the flow conservation property. In other words, $\bar{f}$ is a feasible flow. Then, in the second step, the algorithm augments $\bar{f}$ to an optimal flow.

First step: feasibility. Recall that every integral flow decomposes into cycles and s-t paths[5] (see, e.g., [1, Theorem 3.5]). The algorithm initializes $\bar{f} = f$. While there is an edge $e \in E$ with $\bar{f}(e) > c(e)$, the algorithm uses, e.g., depth-first search to find a cycle or an s-t path containing e (at least one of them is guaranteed to exist because of the integral flow decomposition), and decreases the flow $\bar{f}$ along this cycle/path by one unit. This keeps the invariant that $\bar{f}$ satisfies the flow conservation property. When the process is done, $\bar{f}$ also satisfies all the capacity constraints.

Second step: optimization. Now, the algorithm constructs the *residual network* with respect to $\bar{f}$, i.e., the flow network $G_{\bar{f}} = (V, E_{\bar{f}})$ with edge set $E_{\bar{f}} = \{(u,v) \mid (u,v) \in E \text{ or } (v,u) \in E\}$ and residual capacities[6] $c_{\bar{f}}(u,v) = (c(u,v) - \bar{f}(u,v)) + \bar{f}(v,u)$. Here, for notational simplicity, we assume that $c(u,v) = \bar{f}(u,v) = 0$ if $(u,v) \notin E$. Then, the algorithm runs the Ford-Fulkerson method [12] on $G_{\bar{f}}$ to find a maximum flow $f^*_{c_{\bar{f}}}$ in time $O(|E| \cdot \mathrm{val}(f^*_{c_{\bar{f}}}))$. Finally, $\bar{f} + f^*_{c_{\bar{f}}}$ is a maximum flow for the original edge capacities c, see, e.g., [1, Property 2.6].

Running time analysis. Let $\delta = \sum_{e \in E} \max\{f(e) - c(e), 0\}$ be the total amount by which the flow prediction violates the capacity constraints. The algorithm makes at most δ iterations in the first step, and each iteration decreases the flow value $\mathrm{val}(\bar{f})$ by at most one. We conclude that the first step runs in $O(|E| \cdot \delta)$ time, and that $\mathrm{val}(f) - \mathrm{val}(\bar{f}) \leqslant \delta$.

The second step of the algorithm runs in time

$$O(|E| \cdot \mathrm{val}(f^*_{c_{\bar{f}}})) = O(|E| \cdot (\mathrm{val}(f^*_c) - \mathrm{val}(\bar{f}))) =$$
$$= O(|E| \cdot ((\mathrm{val}(f^*_c) - \mathrm{val}(f)) + (\mathrm{val}(f) - \mathrm{val}(\bar{f})))).$$

Let $\eta = \|f - f_c^*\|_1$ denote the prediction error. It is easy to see that $|\mathrm{val}(f^*_c) - \mathrm{val}(f)| \leqslant \eta$, and that $\delta \leqslant \eta$, so, in particular, $\mathrm{val}(f) - \mathrm{val}(\bar{f}) \leqslant \delta \leqslant \eta$. Therefore, the running time of both steps of the algorithm can be bounded by $O(|E| \cdot \eta)$. □

2.1. Alternative variant of the first step

In this section we give an alternative variant of the first step of the above algorithm. The asymptotic running time remains the same, but, as we explained when

[5] I.e., there exists a collection $p_1, \ldots, p_k$ such that each p_i is either a (simple) cycle or a (simple) s-t path in G, and $f(e) = \#\{i \in [k] \mid e \in p_i\}$ for every edge $e \in E$.

[6] The amount of extra flow that can be sent from u to v equals the sum of remaining capacity for edge (u,v), i.e., $c(u,v) - \bar{f}(u,v)$, and the flow sent from v to u, which can be reversed, i.e., $\bar{f}(v,u)$. Usually we expect only one of these two summands to be non-zero.

introducing Theorem 3 in Section 1.1, this alternative algorithm might be more efficient in practice, because it allows delegating most of the work to any existing well-optimized maximum flow solver. See Algorithm 1.

Algorithm 1: Computing maximum flow

Input: flow network $G = (V, E)$, source s, target t, edge capacities c, predicted flow f
Output: maximum flow for edge capacities c

```
/* Step 1: Feasibility */
```
$\widetilde{E} \leftarrow \emptyset$; $\widetilde{G} \leftarrow (V \cup \{\tilde{s}, \tilde{t}\}, \widetilde{E})$; $\delta \leftarrow 0$
foreach $(u, v) \in E$ **do**
  $\widetilde{E} \leftarrow \widetilde{E} \cup \{(v, u)\}$
  if $f(u, v) \leqslant c(u, v)$ **then**
    $\tilde{c}(v, u) \leftarrow f(u, v)$
  else
    $\tilde{c}(v, u) \leftarrow c(u, v)$
    $\delta \leftarrow \delta + f(u, v) - c(u, v)$
    $\widetilde{E} \leftarrow \widetilde{E} \cup \{(\tilde{s}, u), (v, \tilde{t}\}$
    $\tilde{c}(\tilde{s}, u), \tilde{c}(v, \tilde{t}) \leftarrow f(u, v) - c(u, v)$
$\widetilde{E} \leftarrow \widetilde{E} \cup \{(s, t)\}$; $\tilde{c}(s, t) \leftarrow \delta$
$\tilde{f} \leftarrow$ max flow from $\tilde{s}$ to $\tilde{t}$ in $\widetilde{G}$ with capacities $\tilde{c}$
foreach $(u, v) \in E$ **do**
  $\bar{f}(u, v) \leftarrow \min(f(u, v), c(u, v)) - \tilde{f}(v, u)$
```
/* Step 2: Optimization */
```
$G_{\bar{f}} \leftarrow$ residual network with respect to flow $\bar{f}$
$f^*_{c_{\bar{f}}} \leftarrow$ max flow from s to t in $G_{\bar{f}}$
return $\bar{f} + f^*_{c_{\bar{f}}}$

Consider graph $\widetilde{G} = (V, \widetilde{E})$ with $\widetilde{E} = \{(v, u) \in V \times V \mid (u, v) \in E\}$, i.e., a copy of G with reversed edges. Set capacities to $\tilde{c}(v, u) = f(u, v)$. Note that the first step of the original algorithm essentially finds an integral t-s flow $\tilde{f}$ in $\widetilde{G}$ such that

(i) if $f(u, v) > c(u, v)$, then $\tilde{f}(v, u) \geqslant f(u, v) - c(u, v)$, for every $(u, v) \in E$;
(ii) $\mathrm{val}(\tilde{f}) \leqslant \delta$.

At the end of the first step $\bar{f}(u, v) = f(u, v) - \tilde{f}(v, u)$. In this section we give a different way to compute such $\tilde{f}$.

Add to $\widetilde{G}$ edge (s, t), and set $\tilde{c}(s, t) = \delta$. Now, the problem of finding a t-s flow satisfying (i) and (ii) becomes the problem of finding a *circulation*[7] satisfying (i). This problem – of finding a circulation with lower bounds – can be reduced to a problem of finding a maximum flow (without lower bounds) in a graph with the maximum flow value equal to the sum of all lower bounds, see, e.g., [1, Section 6.7]. The reduction works as follows.

First, add to $\widetilde{G}$ two new nodes $\tilde{s}$ and $\tilde{t}$. Next, for each edge $e = (u, v) \in E$ that violates the capacity constraint let $\delta_e = f(u, v) - c(u, v) > 0$ be the excess flow for this edge; add to $\widetilde{G}$ two edges, $(\tilde{s}, u)$ and $(v, \tilde{t})$, set their capacities to $\tilde{c}(\tilde{s}, u) = \tilde{c}(v, \tilde{t}) = \delta_e$, and decrease the capacity of edge (v, u) by δ_e, so that $\tilde{c}(v, u) = c(u, v)$. This ends the description of the graph constructed in the reduction.

Note that the total capacity of edges leaving $\tilde{s}$ equals the total capacity of edges entering $\tilde{t}$, which also equals $\delta = \sum_{e \in E} \max\{f(e) - c(e), 0\}$. As we will see in a moment, the existence of a flow saturating these edges is equivalent to the existence of a circulation satisfying the lower bounds – which is guaranteed to exist because the original first step of the algorithm finds such a circulation.

After constructing $\widetilde{G}$ as above, the alternative first step proceeds as follows. The algorithm computes a maximum $\tilde{s}$-$\tilde{t}$ flow $\tilde{f}$ in $\widetilde{G}$, using a Ford-Fulkerson method. Then, for each edge $e = (u, v) \in E$ that violates the capacity constraint (in the original graph G), the algorithm first removes the saturated edges $(\tilde{s}, u)$ and $(v, \tilde{t})$ from $\widetilde{G}$. Note that now nodes u and v do not satisfy the flow conservation property, namely node v has an excess of δ_e units of incoming flow and node u has a deficit of δ_e units of incoming flow. The algorithm restores the flow conservation property by increasing flow $\tilde{f}(v, u)$ by δ_e units, and therefore it ensures that the lower bound for this edge is satisfied. This procedure essentially proves the equivalence of the existence of a flow saturating the sink and source edges and the existence of a suitable circulation.

This ends the description of the alternative algorithm. Let us analyse its running time. Graph $\widetilde{G}$ has $O(|E|)$ edges and can be constructed in $O(|E|)$ time. Since $\mathrm{val}(\tilde{f}) = \delta$, the Ford-Fulkerson method runs in $O(|E| \cdot \delta)$ time. Finally, transforming $\tilde{f}$ to $\bar{f}$ takes $O(|E|)$ time. Therefore, the total running time of $O(|E| \cdot \delta)$ remains unchanged compared to the original first step of the algorithm. This proves Theorem 3.

Theorem 3. *Given a directed graph $G = (V, E)$, source and sink $s, t \in V$, edge capacities $c : E \to \mathbb{Z}_{\geqslant 0}$, and a predicted flow function $f : E \to \mathbb{Z}_{\geqslant 0}$ satisfying flow conservation, one can reduce computing a maximum (s, t)-flow in G to computing maximum flows in two graphs, each with $O(|E|)$ edges and the maximum flow value bounded by $O(\|f - f^*_c\|_1)$. The reduction works in*

[7] A circulation is defined similarly to a flow. The only exception is that there are no designated source and sink nodes, and hence the flow conservation property has to be satisfied for all the nodes of the graph (see, e.g., [1, Section 1.2]).

$O(|E|)$ *time.*

Finally, we remark that a similar trick – for handling edges initialized with a flow exceeding their capacities – was already proposed, albeit without provable running time guarantees, in the context of repeatedly solving similar minimum cut instances in a computer vision application [19]. That trick however only allows computing the maximum flow value and a corresponding minimum cut, but not the flow itself.

3. Learning a prediction that minimizes error

Theorem 4. *Given a directed graph $G = (V, E)$, with source and sink $s, t \in V$, and a collection of k lists of edge capacities $c_1, c_2, \ldots, c_k \in \mathbb{Z}^E_{\geqslant 0}$, one can find an integral flow prediction that minimizes the prediction error on this collection, i.e.,*

$$\hat{f} = \arg\min \Big\{ \frac{1}{k} \sum_{i \in [k]} \|f - f^*_{c_i}\|_1 \;\Big|\; f : E \to \mathbb{Z}_{\geqslant 0} \text{ satisfying flow conservation} \Big\},$$

in time $O(T(k \cdot |E|))$, where $T(m) \leqslant m^{1+o(1)}$ denotes the min-cost flow complexity in graphs with m edges.

Proof. The first step of the learning algorithm (see Algorithm 2) is to compute a maximum flow $f^*_{c_i}$ for each $i \in [k]$. This step can be completed in $k \cdot T(|E|) \leqslant O(T(k \cdot |E|))$ time in total.

Now, the goal is to find an integral flow f (satisfying the flow conservation property) that minimizes $\sum_{i \in [k]} \sum_{e \in E} |f(e) - f^*_{c_i}(e)| = \sum_{e \in E} \sum_{i \in [k]} |f(e) - f^*_{c_i}(e)|$. For an edge $e \in E$, let $\text{cost}_e(x) = \sum_{i \in [k]} |x - f^*_{c_i}(e)|$ denote the contribution of $f(e)$ to the minimization objective, which now can be written simply as $\sum_{e \in E} \text{cost}_e(f(e))$.

Let us analyse how the function $\text{cost}_e(x)$ behaves. Let $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_k$ denote the sorted elements of the (multi-)set $\{f^*_{c_1}(e), f^*_{c_2}(e), \ldots, f^*_{c_k}(e)\}$. Clearly, $\text{cost}_e(0) = \sum_{i \in [k]} x_i$. For $x \in [0, x_1]$, the contribution $\text{cost}_e(x)$ is a decreasing linear function with slope $-k$. For $x \in [x_1, x_2]$, the slope is $-k + 2$. More generally, for $x \in [x_i, x_{i+1}]$ the slope is $2i - k$, because increasing the flow by δ increases also by δ each of the first i summands, and decreases by the same amount each of the remaining $(k - i)$ summands in the sum $\sum_{i \in [k]} |x - x_i|$. Hence, $\text{cost}_e(x)$ is piecewise linear and convex, and the overall goal is to find a flow minimizing a separable piecewise linear convex cost function.

The above problem can be reduced to the standard minimum cost flow problem [1, Chapter 14]. The reduction works as follows. For notational simplicity, let $x_0 = 0$ and $x_{k+1} = +\infty$. Replace each edge $e \in E$ with $k + 1$ parallel edges $e_0, \ldots, e_k$, and let edge e_i have capacity $x_{i+1} - x_i$ and cost (of sending one unit of flow) equal to $2i - k$. It is easy to observe that any optimal solution to the minimum cost flow problem in the constructed multigraph uses some prefix of the cheapest parallel edges for each $e \in E$, and the total cost of such prefix behaves exactly like cost_e. Since all the introduced capacities are integral, it is guaranteed that there exists an optimal integral solution. The multigraph has $(k + 1) \cdot |E|$ edges, hence the minimum cost flow can be found in $O(T(k \cdot |E|))$. □

Algorithm 2: Learning prediction from samples

Input: flow network $G = (V, E)$, and sampled edge capacity functions $c_1, c_2, \ldots, c_k$
Output: flow f minimizing $\frac{1}{k} \sum_{i \in [k]} \|f - f^*_{c_i}\|_1$
foreach $i \in \{1, 2, \ldots, k\}$ **do**
  compute maximum flow $f^*_{c_i}$ for capacities c_i
foreach $e \in E$ **do**
  $x_1 \leqslant \cdots \leqslant x_k \leftarrow$ sorted set $\{f^*_{c_1}(e), \ldots, f^*_{c_k}(e)\}$
  $x_0 \leftarrow 0$, $x_{k+1} \leftarrow +\infty$
  replace e with $k + 1$ parallel edges $e_0, \ldots, e_k$
  foreach $i \in \{0, \ldots, k\}$ **do**
    capacity of $e_i \leftarrow x_{i+1} - x_i$
    cost of $e_i \leftarrow 2i - k$
return minimum cost flow in the modified graph

4. Sample complexity

Theorem 5. *Let $G = (V, E)$ be a directed graph, with source and sink $s, t \in V$, and let $c_1, c_2, \ldots, c_k \in \mathbb{Z}^E_{\geqslant 0}$, for $k = \Theta(c^2_{\max} |E|^3 \log(c_{\max} |E|))$, be independent samples from a distribution $\mathcal{D}$, where $c_{\max} = \max_{c \in \text{supp}(\mathcal{D}), e \in E} c(e)$. Let $\hat{f} \in \mathbb{Z}^E_{\geqslant 0}$ be a flow that minimizes the prediction error on this collection of samples, as in Theorem 4. Then, with high probability over the choice of the samples, the expected ℓ_1 error of $\hat{f}$ over $\mathcal{D}$ is approximately minimum possible, i.e.,*

$$\mathbb{E}_{c \sim \mathcal{D}} \|\hat{f} - f^*_c\|_1 \leqslant \min_f \mathbb{E}_{c \sim \mathcal{D}} \|f - f^*_c\|_1 + O(1),$$

where the minimum is taken over functions $f \in \mathbb{Z}^E_{\geqslant 0}$ satisfying the flow conservation property.

For a flow prediction $f \in \mathbb{Z}^E_{\geqslant 0}$, let us use

$$\mathrm{cost}_{c_1,\dots,c_k}(f) = \frac{1}{k}\sum_{i\in[k]} \|f - f^*_{c_i}\|_1,$$
$$\mathrm{cost}_{\mathcal{D}}(f) = \mathbb{E}_{c\sim\mathcal{D}}\|f - f^*_c\|_1$$

to denote the ℓ_1 error of f on the samples and on the distribution, respectively. We will use Hoeffding's inequality to prove that the number of samples in Theorem 5 is large enough for $\mathrm{cost}_{c_1,\dots,c_k}(f)$ to be a good approximation of $\mathrm{cost}_{\mathcal{D}}(f)$, with high probability for all f's simultaneously.

Theorem 6 (Hoeffding's inequality [16])**.** *Let $X_1, \dots, X_k$ be independent random variables with values from 0 to U, and let $S = X_1 + \cdots + X_k$ denote their sum. Then, for all $t > 0$,*

$$P(|S - \mathbb{E}S| \geqslant t) \leqslant 2 \cdot \exp(-2t^2/kU^2).$$

To use the inequality, first we need a bound on the values of the considered functions.

Lemma 7. *Any flow prediction f that minimizes the error must satisfy $\|f\|_1 \leqslant 2c_{\max}|E|$.*

Let us note that Lemma 7 is actually nontrivial. Even though $\|f^*_{c_i}\|_\infty \leqslant c_{\max}$ for every $i \in [k]$, it may happen that $\|f\|_\infty > c_{\max}$ because of the flow conservation constraint, e.g., when multiple disjoint paths end at a single node and force a single edge going out of that node to have a flow larger than $c_{\max}$.

Proof of Lemma 7. For every $c \in \mathrm{supp}(\mathcal{D})$, we have $\|c\|_1 \leqslant c_{\max}|E|$, and, since $0 \leqslant f^*_c \leqslant c$, then also $\|f^*_c\|_1 \leqslant c_{\max}|E|$. Moreover, by the triangle inequality, $\|f - f^*_c\|_1 + \|f^*_c\|_1 \geqslant \|f\|_1$, and thus $\|f - f^*_c\|_1 \geqslant \|f\|_1 - c_{\max}|E|$. If $\|f\|_1 > 2c_{\max}|E|$, then $\|f - f^*_c\|_1 > c_{\max}|E|$ for every $c \in \mathrm{supp}(\mathcal{D})$, and thus also $\mathrm{cost}_{\mathcal{D}}(f) = \mathbb{E}_{c\sim\mathcal{D}}\|f - f^*_c\|_1 > c_{\max}|E|$.

At the same time, if we consider the all-zero vector as a flow prediction, we have $\|0 - f^*_c\|_1 = \|f^*_c\|_1 \leqslant c_{\max}|E|$, for every $c \in \mathrm{supp}(\mathcal{D})$, and thus also $\mathrm{cost}_{\mathcal{D}}(0) = \mathbb{E}_{c\sim\mathcal{D}}\|0 - f^*_c\|_1 \leqslant c_{\max}|E|$. It follows that f could not minimize the error if $\|f\|_1 > 2c_{\max}|E|$. □

Now we are ready to apply Hoeffding's inequality in order to prove the following lemma.

Lemma 8. *With high probability over the choice of the samples, for all $f \in \mathbb{Z}^E_{\geqslant 0}$ satisfying the flow conservation property and such that $\|f\|_1 \leqslant 2c_{\max}|E|$ it holds that*

$$|\mathrm{cost}_{c_1,\dots,c_k}(f) - \mathrm{cost}_{\mathcal{D}}(f)| \leqslant 1.$$

Proof. For a fixed f, satisfying the conditions of the lemma, let $X_i = \frac{1}{k}\|f - f^*_{c_i}\|_1$. We have that $\|f - f^*_{c_i}\|_1 \leqslant \|f\|_1 + \|f^*_{c_i}\|_1 \leqslant (2+1)\cdot c_{\max}|E|$, so the random variable X_i has values from 0 to $3c_{\max}|E|/k$. Clearly, $\mathrm{cost}_{c_1,\dots,c_k}(f) = X_1 + \cdots + X_k$, and $\mathbb{E}\,\mathrm{cost}_{c_1,\dots,c_k}(f) = \mathrm{cost}_{\mathcal{D}}(f)$. Applying Hoeffding's inequality, with $t = 1$, we get that

$$\begin{aligned} &P(|\mathrm{cost}_{c_1,\dots,c_k}(f) - \mathrm{cost}_{\mathcal{D}}(f)| \geqslant 1) \\ &\quad\leqslant 2\cdot\exp\Big(-\frac{2}{k\cdot(3c_{\max}|E|/k)^2}\Big) \\ &\quad= \exp(-\Theta(k/(c_{\max}|E|)^2)) \\ &\quad= \exp(-\Theta(|E|\log(c_{\max}|E|))). \end{aligned}$$

Let $\mathcal{F}$ denote the set of all f's satisfying the conditions of the lemma. We can upper-bound the number of such f's by $|\mathcal{F}| \leqslant (2c_{\max}|E| + 1)^{|E|} = \exp(\Theta(|E|\log(c_{\max}|E|)))$. To finish the proof, note that

$$\begin{aligned} &|\mathcal{F}|\cdot\mathrm{poly}(c_{\max}|E|) \\ &\quad\leqslant \exp(\Theta(|E|\log(c_{\max}|E|)))\cdot\exp(\Theta(\log(c_{\max}|E|))) \\ &\quad= \exp(\Theta(|E|\log(c_{\max}|E|))), \end{aligned}$$

and hence we can take the union bound to conclude that with high probability it holds that $|\mathrm{cost}_{c_1,\dots,c_k}(f) - \mathrm{cost}_{\mathcal{D}}(f)| \leqslant 1$ for all $f \in \mathcal{F}$ simultaneously. □

Let us remark that the above proof of Lemma 8 crucially relies on the fact that it suffices to consider a finite set of possible flow predictions – because they are integral and bounded – and therefore we can use the union bound. Dinitz et al. [6, Section 3.3 in their supplemental material] give a proof of an analogous result regarding learning optimal dual solution for the weighted bipartite matching problem. Their proof is more complex than ours, it uses the notion of pseudo-dimension, but thanks to that it works also for fractional predictions. We note that it is possible to prove alternative version of Lemma 8, in the spirit of Dinitz et al., that would generalize to fractional flows but would lose a small factor $\log|E|$ in the sample complexity.

With Lemma 8 at hand, it takes a standard argument to prove Theorem 5.

Proof of Theorem 5. Let $\hat{f}$ and $\tilde{f}$ be flow predictions that minimize the error on the samples and on the whole distribution, respectively. By Lemma 7, $\|\hat{f}\|_1, \|\tilde{f}\|_1 \leqslant 2c_{\max}|E|$, and hence Lemma 8 applies. Note that it is crucial that Lemma 8 holds with high probability for all f's, because $\hat{f}$ is chosen after the samples are drawn from $\mathcal{D}$. We finish the proof with the following chain of

inequalities.

$$\operatorname{cost}_{\mathcal{D}}(\hat{f}) \overset{\text{Lemma 8}}{\leqslant} \operatorname{cost}_{c_1,\dots,c_k}(\hat{f}) + 1 \leqslant \operatorname{cost}_{c_1,\dots,c_k}(\bar{f}) + 1 \overset{\text{Lemma 8}}{\leqslant} \operatorname{cost}_{\mathcal{D}}(\bar{f}) + 2.$$

($\hat{f}$ minimizes the error on $c_1, \dots, c_k$)

□

Acknowledgments

We would like to thank Alexandra Lassota, Sai Ganesh Nagarajan, and Moritz Venzin for helpful discussion.

Funding

Part of this work was done while Adam Polak was at École Polytechnique Fédérale de Lausanne supported by the Swiss National Science Foundation projects *Lattice Algorithms and Integer Programming* (185030) and *Complexity of Integer Programming* (CRFS-2_207365). During the preparation of this paper Maksym Zub was a participant of the tutoring programme under the Excellence Initiative at the Jagiellonian University.

None of the funding sources had any involvement in study design, in the writing of the report, or in the decision to submit the article for publication.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows – theory, algorithms and applications*. Prentice Hall, 1993.

[2] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004. doi:10.1109/TPAMI.2004.60.

[3] 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.

[4] 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*, pages 612–623. IEEE, 2022. doi:10.1109/FOCS54457.2022.00064.

[5] 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.

[6] Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In *Advances in Neural Information Processing Systems*, volume 34, pages 10393–10406. Curran Associates, Inc., 2021. URL: https://papers.nips.cc/paper/2021/hash/5616060fb8ae85d93f334e7267307664-Abstract.html.

[7] Yefim Dinitz. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl.*, 11:1277–1280, 1970.

[8] Talya Eden, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, and Tal Wagner. Learning-based support estimation in sublinear time. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=tilovEHA3YS.

[9] Jack R. Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972. doi:10.1145/321694.321699.

[10] 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.

[11] Barak Fishbain, Dorit S. Hochbaum, and Stefan Müller. A competitive study of the pseudoflow algorithm for the minimum s-t cut problem in vision applications. *J. Real Time Image Process.*, 11(3):589–609, 2016. doi:10.1007/s11554-013-0344-3.

[12] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

[13] Zvi Galil and Amnon Naamad. An $O(EV \log^2 V)$ algorithm for the maximal flow problem. *J. Comput. Syst. Sci.*, 21(2):203–217, 1980. doi:10.1016/0022-0000(80)90035-5.

[14] Andrew V. Goldberg and Robert Endre Tarjan. A new approach to the maximum flow problem. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, 1986*, pages 136–146. ACM, 1986. doi:10.1145/12130.12144.

[15] Dorit S. Hochbaum. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Oper. Res.*, 56(4):992–1009, 2008. doi:10.1287/opre.1080.0524.

[16] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi:10.1080/01621459.1963.10500830.

[17] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=r1lohoCqY7.

[18] Olivier Juan and Yuri Boykov. Active graph cuts. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 1023–1029. IEEE Computer Society, 2006. doi:10.1109/CVPR.2006.47.

[19] Pushmeet Kohli and Philip H. S. Torr. Efficiently solving dynamic markov random fields using graph cuts. In *10th IEEE International Conference on Computer Vision (ICCV 2005)*, pages 922–929. IEEE Computer Society, 2005. doi:10.1109/ICCV.2005.81.

[20] 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.

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

[22] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *Proceedings 8th ACM Conference on Electronic Commerce (EC-2007)*, pages 288–294. ACM, 2007. doi:10.1145/1250910.1250952.

[23] 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.

[24] 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.

[25] Shinsaku Sakaue and Taihei Oki. Discrete-convex-analysis-based framework for warm-starting algorithms with predictions. In *NeurIPS*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/844e61124d9e1f58632bf0c8968ad728-Abstract-Conference.html.

[26] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, 2004*, pages 81–90. ACM, 2004. doi:10.1145/1007352.1007372.