Online Unrelated Machine Load Balancing with Predictions Revisited

Shi Li [*1] **Jiayi Xian** [*1]

Abstract

We study the online load balancing problem with machine learned predictions, and give results that improve upon and extend those in a recent paper by Lattanzi et al. (2020). First, we design deterministic and randomized online rounding algorithms for the problem in the unrelated machine setting, with $O\left(\frac{\log m}{\log\log m}\right)$- and $O\left(\frac{\log\log m}{\log\log\log m}\right)$-competitive ratios. They respectively improve upon the previous ratios of $O(\log m)$ and $O(\log^3 \log m)$, and match the lower bounds given by Lattanzi et al. Second, we extend their prediction scheme from the identical machine restricted assignment setting to the unrelated machine setting. With the knowledge of two vectors over machines, a dual vector and a weight vector, we can construct a good fractional assignment online, that can be passed to an online rounding algorithm. Finally, we consider the learning model introduced by Lavastida et al. (2020), and show that under the model, the two vectors can be learned efficiently with a few samples of instances.

1. Introduction

Inspired by the tremendous success of modern machine learning techniques, there is a recent surge of interest in using machine learned predictions to design algorithms for online combinatorial optimization problems. In contrast to the worst case analysis of online algorithms, we are given some predicted information about the problem instance we need to solve online, usually learned from previous instances of the same nature. The prediction should be useful and simple: It should allow the algorithm to achieve a better performance than when no information is given, but on the other hand, it should be simple enough so that it can be learned easily. As predictions are often error-prone, ideally the performance of the algorithm should deteriorate smoothly as a function of some error measurement, but at the same time is never worse than the worst-case guarantee, no matter how bad the prediction is. This has led to the area of *learning augmented online algorithm*, in which many classic problems have been studied (See Section 1.2).

In this paper, we study the classic online load balancing problem in the general *unrelated machine* setting under this model. In the offline problem, we are given m machines M, n jobs J, and $p_{i,j} \in (0, \infty]$ for every $i \in M, j \in J$, which indicates the time needed to process job j if it is assigned to machine i. The goal of the problem is to assign the jobs to machines so as to minimize the makespan, i.e, the maximum over all $i \in M$, the sum of $p_{i,j}$'s over all jobs j assigned to i. In the online version of the problem, M is given upfront, but jobs in J come one by one. When a job $j \in J$ arrives, it reveals the vector $(p_{i,j})_{i\in M} \in (0, \infty]^M$. The online algorithm has to irrevocably assign j to a machine upon its arrival. When no predictions are given, the problem admits an $O(\log m)$-competitive ratio (Azar et al., 1995; Aspnes et al., 1997), which is tight (Azar et al., 1995).

An extensively studied special case of the problem is the *identical machine restricted assignment* setting. [1] There is an intrinsic size $p_j > 0$ for every job $j \in J$ and for every machine $i \in M$, we have $p_{i,j} \in \{p_j, \infty\}$. So, every job j has a set of *permissible machines* which it can be assigned to, and the processing time of j is always p_j on a permissible machine. The lower bound $\Omega(\log m)$ of Azar et al. (1995) on the competitive ratio is indeed for this special case.

Lattanzi et al. (2020) initiated the study of online load balancing with learned predictions. Their result contains two components. First, given a load balancing instance in the identical machine restricted assignment setting, they leveraged the proportion allocation scheme of Agrawal et al. (2018) to show that there is a weight vector $w \in \mathbb{R}^M_{>0}$ over the machines, such that the fractional assignment $(x_{i,j})_{i\in M, j\in J}$ obtained by assigning each job j to its permissible machines proportionally to the weights is $(1+\epsilon)$-approximately optimum. Thus if the weight vector w is

[*]Equal contribution [1]Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY, USA. Research is supported in part by NSF grant CCF-1844890. Correspondence to: Shi Li <shil@buffalo.edu>, Jiayi Xian <jxian@buffalo.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

[1]Usually the model is simply called the restricted assignment setting. We use the longer name since later we shall define another setting called the related machine restricted assignment setting.

given as the prediction, the algorithm has access to the fractional assignment $(x_{i,j})_{i\in M,j\in J}$ online. That is, the vector $(x_{i,j})_{i\in M}$ is revealed upon the arrival of j. The vector w can be specified using m real numbers, i.e, one number per-machine. In a typical application, the number m of machines is much smaller than the number n of jobs.

To complement the first component, Lattanzi et al. (2020) designed an online randomized rounding algorithm that achieves a competitive ratio of $O(\log^3 \log m)$. Namely, the makespan of the schedule produced by the algorithm is at most $O(\log^3 \log m)$ times that of the fractional assignment $(x_{i,j})_{i\in M,j\in J}$. Combining it with the first component leads to an online algorithm with $O(\log^3 \log m)$ competitive ratio, given w as the prediction. This is exponentially better than the worst guarantee of $O(\log m)$. Their algorithm is robust: when the predicted vector w has a multiplicative error of $\eta > 1$, the performance worsens by a multiplicative factor of $O(\lceil \log \eta \rceil)$, but is no worse than the worst-case guarantee of $O(\log m)$. We remark that while the proportional allocation scheme (the first component of Lattanzi et al.) works only for the identical machine restricted assignment setting, the online rounding algorithm (the second component) works for the general unrelated machine setting.

On the negative side, Lattanzi et al. showed lower bounds of $\Omega\left(\frac{\log m}{\log\log m}\right)$ and $\Omega\left(\frac{\log\log m}{\log\log\log m}\right)$ respectively on the competitive ratio of any deterministic and randomized online rounding algorithm.

Learnability of Predictions It is natural to measure the complexity of the predicted information using its bit complexity, i.e, the number of bits needed to represent it. This is due to the following phenomenon in computational learning theory: To learn a function from a family of N hypothesis functions, the number of samples needed is typically proportional to $\log N$.

This notion of learnability of predictions was made formal by Lavastida et al. in a recent paper (2020). In their model, it is assumed that the problem instance is generated from some unknown but structured distribution. For the prediction to be learnable, there should be some algorithm that can learn it after seeing a small number of sampled instances from the distribution. For the load balancing problem in the identical machine restricted assignment setting, Lavastida et al. showed that under some mild conditions, an algorithm can learn a vector $w \in \mathbb{R}^M_{>0}$ after seeing $\mathrm{poly}(m, \frac{1}{\epsilon})$ samples, such that the proportional allocation scheme according to w gives a $(1 + O(\epsilon))$-approximate fractional solution.

1.1. Our Results

Our contribution contains three parts. First we develop tight deterministic and randomized online rounding algorithms for the unrelated machine load balancing problem, improving upon the results of Lattanzi et al. (2020). Second we extend the prediction introduced by Lattanzi et al. from the identical machine restricted assignment setting to the unrelated machine setting. Finally, we show that our prediction is learnable under the model of Lavastida et al. (2020).

Tight Online Rounding Algorithms We develop online rounding algorithms for the unrelated machine load balancing problem that match the two lower bounds of Lattanzi et al. (2020). That is, we give a deterministic online rounding algorithm with competitive ratio $O\left(\frac{\log m}{\log\log m}\right)$, and a randomized online rounding algorithm with competitive ratio $O\left(\frac{\log\log m}{\log\log\log m}\right)$ (Theorem 2.1 and 2.2).

So, if a fractional solution is given online, a deterministic algorithm can do slightly better (by a $\log\log m$ factor) than when it is not given. Our algorithm is obtained by derandomizing the simple independent rounding $O\left(\frac{\log m}{\log\log m}\right)$-competitive algorithm, using conditional expectation.

The main contribution of this part is the $O\left(\frac{\log\log m}{\log\log\log m}\right)$-competitive randomized online rounding algorithm, which improves upon the $O(\log^3 \log m)$-competitive ratio of Lattanzi et al. (2020), and matches their lower bound of $\Omega\left(\frac{\log\log m}{\log\log\log m}\right)$. Our algorithm uses some ideas from that of Lattanzi et al., but is much simpler. As in their algorithm, we break jobs into small and big ones, depending on whether a job j is mostly assigned to machines i with $p_{i,j} \geq \Omega(T/\log m)$ or $p_{i,j} \leq O(T/\log m)$ in the fractional solution x (T is the makespan of x). For small jobs j, independent rounding incurs only an $O(1)$-factor loss. For big jobs j, we do an initial rounding to make sure positive $x_{i,j}$ values are at least $\Omega\left(\frac{1}{\log m}\right)$. Then we try to round $x_{i,j}$ values further to 0 or 1. If assigning a job j makes a machine overloaded, then we say j failed. Similar to Lattanzi et al. (2020), we show that in the graph induced by the failed jobs and their support machines, every connected components has $\mathrm{poly}\log m$ machines. Then we can use our deterministic algorithm to handle each connected component separately, resulting in the $O\left(\frac{\log\log m}{\log\log\log m}\right)$ competitive ratio.

Predictions for Unrelated Machine Load Balancing In the second part of our paper, we generalize the first component of Lattanzi et al. (2020) to the unrelated machine setting. Likewise, our goal is to define a prediction about the online instance, so that given the prediction, the algorithm can produce a good fractional assignment $(x_{i,j})_{i,j}$ online. We show that it suffices for the prediction to contain two vectors $\beta, w \in \mathbb{R}^M_{>0}$, each having aspect ratio $\exp\left(O(m \log \frac{m}{\epsilon})\right)$, to obtain an $(1 + \epsilon)$-approximate fractional solution online.

To obtain the result, we introduce an intermediate setting between the identical machine restricted assignment and unrelated machine settings, that we call the *related machine restricted assignment* setting: Each job j has a size $p_j \in \mathbb{R}_{>0}$ and each machine i has a *speed* $s_j \in \mathbb{R}_{>0}$. For every $i \in M, j \in J$ we have $p_{i,j} \in \left\{\frac{p_j}{s_i}, \infty\right\}$. With a simple modification to the analysis in Lattanzi et al. (2020), one can show that the weight vector framework developed in Lattanzi et al. can be applied to this setting as well.

Our vector β in the prediction is used to reduce the instance in the unrelated machine setting to one in the related machine restricted assignment setting, and the vector w gives the weight for the latter instance. To establish the reduction, we resort to the dual of the LP relaxation for the problem. In the dual, each machine i has a variable β_i and each job j has a variable α_j. The complementary slackness condition says that if the optimum primal solution has $x_{i,j} > 0$ for some (i, j) pair, then we must have $\alpha_j = p_{i,j}\beta_i$. Thus, if we view β_i as the speed of machine i and α_j as the size of job j, then we can restrict ourselves to the pairs with $p_{i,j} = \frac{\alpha_j}{\beta_i}$, leading to the related machine restricted assignment setting. To address the issue raised by 0 values in the dual solution (α, β), we only take positive α and β values and remove their correspondent jobs and machines. Then we focus on the residual instance to fill the other α and β values. So, the vector β is constructed piece by piece. The results are formally stated in Theorem 2.4 and Corollary 2.5.

Learnability of Predictions Then we proceed to consider the learnability of the prediction under the model introduced by Lavastida et al. (2020). In their model, for each job j, the vector $(p_{i,j})_{i\in M}$ is generated from some distribution $\mathcal{D}_j$, and the process for all jobs j are independent. It is also assumed that individual $p_{i,j}$ values are reasonably small compared to the expected optimum makespan of the instance. We show that under their model, we can learn a pair (β, w) such that their induced fractional solution is $(1 + \epsilon)$-approximate w.h.p. The analysis is based on concentration bounds and union bound. Due to the page limit, the result is deferred to the supplementary material.

1.2. Related Work

Much work has been done in the area of learning augmented online algorithm. The focus of model is on the consistency and robustness of algorithms. Namely, when the prediction is perfect, the algorithm has to perform better (ideally, much better) than the best online algorithm without using predictions. On the other hand, the algorithm is never worse than the worst-case guarantee, even if the prediction is arbitrarily bad. Many problems have been studied under this model, including caching (Lykouris & Vassilvtiskii, 2018; Rohatgi, 2020; Jiang et al., 2020; Wei, 2020), ski-rental (Gollapudi & Panigrahi, 2019; Anand et al., 2020), scheduling (Kumar et al., 2018; Lattanzi et al., 2020), secretary and online matching (Antoniadis et al., 2020), linear optimization (Bhaskara et al., 2020) and primal-dual method (Bamas et al., 2020).

A very similar model studied in the literature is the online algorithm with advice model (Boyar et al., 2016). There is an oracle that knows the whole input sequence. The online algorithm is allowed to query the oracle about the input. The goal is to understand the minimum number of bits needed from the oracle in order for the algorithm to achieve certain competitive ratio. Both models are in a broader theme of studying online algorithms beyond worst case analysis. Other models in the theme include the random arrival order (Karp et al., 1990; Mahdian & Yan, 2011; Devanur et al., 2013), stochastic distribution (Devanur, 2011; Manshadi et al., 2011) and semi-online (Schild et al., 2019) models.

There is a rich literature on the load balancing problem. The classic result of Lenstra et al. (1990) gives a 2-approximation for the offline problem in the unrelated machine setting, which remains the best approximation algorithm for the problem. Much work has been done for the identical machine restricted assignment setting (Svensson, 2012; Chakrabarty et al., 2015; Jansen & Rohwedder, 2017; 2020).

For the online load balancing problem, Azar et al. (1995) developed an $O(\log m)$-competitive algorithm for the identical machine restricted assignment setting, and proved that the ratio is tight. Aspnes et al. (1997) extended the $O(\log m)$ competitive ratio to the unrelated machine setting. Thus, our understanding of the competitive ratio for online load balancing in the two settings is complete.

2. Preliminaries and Formal Statements of Our Results

2.1. Problem Definition and Notations

In the offline unrelated machine load balancing problem, we are given a set J of n jobs and a set M of m machines; one should think that m is much smaller than n. For every job j and machine i, $p_{i,j} \in (0, \infty]$ is the processing time of job j on machine i; if $p_{i,j} = \infty$, then j can not be assigned to i. The goal is to find an assignment $\sigma : J \to M$ such that $\max_{i\in M} \sum_{j\in\sigma^{-1}(i)} p_{i,j}$ is minimized. In the online problem, M is given upfront, but jobs J arrive one by one. When a job j arrives, $(p_{i,j})_{i\in M}$ is revealed and we have to irrevocably decide the machine $\sigma(j)$ that j is assigned to.

We use $E := \{(i \in M, j \in J) : p_{i,j} \neq \infty\}$ to denote the set of allowed machine-job pairs. For every $j \in J$, define $M_j := \{i : (i, j) \in E\}$ to be the set of machines j can be assigned to, and for every $i \in M$, $J_i := \{j : (i, j) \in E\}$ to be the set of jobs that can be assigned to i.

Throughout the paper, we always use ϵ to denote an accuracy parameter in $(0, 1)$ that controls the losses incurred in various places. We make the following assumption when defining the prediction for unrelated machine load balancing. It increases the optimum makespan by at most a multiplicative factor of $1 + \epsilon$. See Section A for the proof.

Assumption 1. *For every job $j \in J$ and two machines $i, i' \in M_j$, we have $\frac{p_{i',j}}{p_{i,j}} < \frac{m}{\epsilon}$.*

Throughout the paper, by "with high probability", we mean with probability at least $1 - 1/\mathrm{poly}(m)$, for any $\mathrm{poly}(m)$ factor we desire for.

2.2. Primal and Dual Linear Programs

Now we describe the primal linear programming relaxation for the problem, and its dual. In the primal LP relaxation (P-LP), T' is the makespan of the assignment we try to minimize, and $x_{i,j}, (i,j) \in E$ indicates whether a job j is assigned to a machine i or not (in the correspondent integer program). (1) requires all jobs to be scheduled. (2) bounds the makespan of the schedule. All x variables are non-negative (constraint (3)).

$$\min \qquad T' \qquad\qquad \text{(P-LP)}$$

$$\begin{aligned} \textstyle\sum_{i \in M_j} x_{i,j} &= 1 & \forall j \in J \qquad (1)\\ \textstyle\sum_{j \in J_i} p_{i,j} x_{i,j} &\le T' & \forall i \in M \qquad (2)\\ x_{i,j} &\ge 0 & \forall (i,j) \in E \qquad (3) \end{aligned}$$

The LP as stated has integrality gap m. Consider the case where one job of size 1 that can be assigned to all the m machines. The LP value is $\frac{1}{m}$ but the optimum makespan is 1. To overcome the issue, for any $x \in [0,1]^E$ with $\sum_{i \in M_j} x_{i,j} = 1$ for every $j \in J$, we define the makespan of x to be

$$\mathrm{mspn}(x) := \max \begin{cases} \max_{i \in M} \sum_{j \in J_i} p_{i,j} x_{i,j} \\ \max_{(i,j) \in E : x_{i,j} > 0} p_{i,j} \end{cases}.$$

The first quantity inside the max operator is the value of x to the LP, and the second one is the max $p_{i,j}$ over all (i,j)'s in the support of x. By guessing the value of the second quantity, the x with the smallest $\mathrm{mspn}(x)$ can be found efficiently, and the value is clearly at most the makespan of the optimum (integral) assignment.

The dual of (P-LP) is (D-LP), where α_j's and β_i's correspond to constraints (1) and (2) in (P-LP) respectively, and (4) and (5) correspond to variables $x_{i,j}$'s and T' in (P-LP) respectively. In the optimum solution (α, β), we must have $\alpha_j = \min_{i \in M_j} p_{i,j}\beta_i$ for every $j \in J$, since this maximizes $\sum_{j \in J} \alpha_j$ while maintaining (4). We can treat each β_i as the per-unit-time cost of using the machine i. Then α_j is the minimum cost for processing job j, and $\sum_{j \in J} \alpha_j$ is minimum total cost needed to process all jobs. As $\sum_{i \in M} \beta_i = 1$ (constraint (5)), the budget we can use for processing jobs is exactly the makespan. Therefore, $\sum_{j \in J} \alpha_j$ gives a lower bound on the optimum makespan. Later, we shall use β_i as the *speed* of machine i to convert the an instance to one in the *related machine restricted assignment* setting, described in Section 2.4.

$$\max \qquad \textstyle\sum_{j \in J} \alpha_j \qquad\qquad \text{(D-LP)}$$

$$\begin{aligned} \alpha_j - p_{i,j}\beta_i &\le 0 & \forall (i,j) \in E \qquad (4)\\ \textstyle\sum_{i \in M} \beta_i &= 1 & (5)\\ \beta_i &\ge 0 & \forall i \in M \qquad (6) \end{aligned}$$

2.3. Online Rounding Algorithm

In the setting of an online rounding algorithm, when a job j arrives, in addition to the $(p_{i,j})_{i \in M}$ vector, we are also given a non-negative vector $(x_{i,j})_{i \in M_j}$ such that $\sum_{i \in M_j} x_{i,j} = 1$. As usual, upon the arrival of j, our algorithm has to irrevocably assign j to a machine in M_j. The algorithm is called an online rounding algorithm since it converts the fractional assignment x to an integral one. We make the following assumption, and see Section A for its justification.

Assumption 2. *We are given $T = \mathrm{mspn}(x)$ upfront.*

Our algorithm is α-competitive if the makespan of the assignment it produces is at most αT. Our main results are:

Theorem 2.1. *There is an $O\left(\frac{\log m}{\log\log m}\right)$-competitive deterministic online rounding algorithm for the unrelated machine load balancing problem.*

Theorem 2.2. *There is an $O\left(\frac{\log\log m}{\log\log\log m}\right)$-competitive randomized online rounding algorithm for the unrelated machine load balancing problem. The algorithm succeeds with high probability.*

2.4. Identical and Related Machine Restricted Assignment Settings

We will deal with two special settings for the load balancing problem. The first special case is the *identical machine restricted assignment* setting. In the setting, there is an intrinsic size $p_j \in \mathbb{R}_{>0}$ for every $j \in J$. For every $i \in M$, we have $p_{i,j} \in \{p_j, \infty\}$. The second special case, which is more general than the first one, is the *related machine restricted assignment* setting. Now additionally every machine is given a speed $s_i \in \mathbb{R}_{>0}$. Then for every i, j we have $p_{i,j} \in \{\frac{p_j}{s_i}, \infty\}$. So, in the second setting, when a job j can be assigned to a machine i, its processing time is the size of j divided by the speed of i.

For convenience, we use $\mathsf{P}|\text{restricted}$ and $\mathsf{Q}|\text{restricted}$ to denote the identical machine and related machine restricted assignment settings respectively. Notice that the general setting is called the *unrelated machine* setting.

2.5. Proportional Allocation Scheme

Agrawal et al. (2018) considered a proportional allocation scheme for assigning jobs to machines fractionally in the $\mathsf{P}|\mathrm{restricted}$ setting to maximize the throughput of a scheduling. Later Lattanzi et al. (2020) applied the scheme to the load balancing problem in the same setting. Given the sets $(M_j)_{j\in J}$ of permissible machines for all jobs $j \in J$, a weight vector $w \in \mathbb{R}^M_{>0}$, the fractional solution $x^{(w)}$ assigns each job j to its permissible machines proportionally to the weights. Namely, for every $(i, j) \in E$, we have $x^{(w)}_{i,j} = \frac{w_i}{\sum_{i'\in M_j} w_{i'}}$. Lattanzi et al. showed that for a load balancing instance in the $\mathsf{P}|\mathrm{restricted}$ setting, there is a vector w such that $x^{(w)}$ is $(1-\epsilon)$-approximate.

We generalizes the result to the $\mathsf{Q}|\mathrm{restricted}$ setting in Lemma 2.3. From now on, we use $\mathrm{powers}_{r,K}$ for any real $r > 1$ and integer $K \geq 1$, to denote the set $\{r^0, r^1, r^2, \cdots, r^K\}$.

Lemma 2.3. *Given a load balancing instance in the $\mathsf{Q}|\mathrm{restricted}$ setting, for any $\epsilon \in (0, 1)$, there is a weight vector $w \in \mathrm{powers}^M_{1+\epsilon,K}$ for some $K = O\left(\frac{m}{\epsilon}\log\frac{m}{\epsilon}\right)$ such that $x^{(w)}$ is a $(1+\epsilon)^3$-approximate solution to (P-LP).*

2.6. Prediction for Unrelated Machine Load Balancing

Our main theorem in deriving the prediction on unrelated machine load balancing is the following one, which says given a vector $\beta \in \mathbb{R}^M_{>0}$ with mild precision requirement, we can reduce a load balancing instance in the unrelated machine setting to one in the $\mathsf{Q}|\mathrm{restricted}$ setting.

Theorem 2.4. *Assume we are given an unrelated machine load balancing instance, and $\epsilon \in (0, 1)$. There exists a $\beta \in \mathrm{powers}^M_{1+\epsilon,K}$ for some $K = O\left(\frac{m}{\epsilon}\log\frac{m}{\epsilon}\right)$, $\alpha_j = \min_{i\in M_j} p_{i,j}\beta_i$ for every $j \in J$, and an optimum solution $x \in [0, 1]^E$ to (P-LP) such that the following holds. For every $(i, j) \in E$ with $x_{i,j} > 0$ we have $p_{i,j}\beta_i \leq (1 + \epsilon)\alpha_j$.*

To see why the theorem gives an instance in the $\mathsf{Q}|\mathrm{restricted}$ setting, we can let α_j be the size of j and β_i be the speed of i. Then $x_{i,j} > 0$ implies $p_{i,j} \in \left[\frac{\alpha_j}{\beta_i}, \frac{(1+\epsilon)\alpha_j}{\beta_i}\right]$. Thus, if we restrict ourselves to the support of x, the processing times are approximated by size-to-speed ratios.

Let $T^* = \min_x \mathrm{mspn}(x)$, where x is over all $x \in [0, 1]^E$ satisfying $\sum_{i\in M_j} x_{i,j} = 1, \forall j \in J$. For simplicity we make following assumption when defining our prediction:

Assumption 3. *T^* is known to us and every $(i, j) \in E$ has $p_{i,j} \leq T^*$.*

Indeed T^* can be a part of the prediction, and often it is easier to learn T^* (approximately) than the other parameters. By removing pairs (i, j) with $p_{i,j} > T^*$ from E, the second part of the assumption is guaranteed.

The prediction contains two vectors in $\mathrm{powers}^M_{1+\epsilon,K}$ for $K = O\left(\frac{m}{\epsilon}\log\frac{m}{\epsilon}\right)$: a dual vector β and a weight vector w. For a fixed pair (β, w) and a job $j \in J$, we define $x^{(\beta,w)}_{i,j} \in [0, 1]$ for each i as follows. Let $\alpha_j := \min_{i\in M_j} p_{i,j}\beta_i$, and $M'_j := \{i \in M_j : p_{i,j}\beta_i \leq (1 + \epsilon)\alpha_j\}$. Then $x^{(\beta,w)}_{i,j} := \frac{w_i}{\sum_{i'\in M'_j} w_{i'}}$ if $i \in M'_j$ and $x^{(\beta,w)}_{i,j} := 0$ if $i \in M_j \setminus M'_j$. So, for any $j \in J$, $(x^{(\beta,w)}_{i,j})_i$ is determined by $(p_{i,j})_i$, β and w, a crucial property for our online algorithm. The following corollary gives our prediction:

Corollary 2.5. *Given an unrelated machine load balancing instance, there are $\beta, w \in \mathrm{powers}^M_{1+\epsilon,K}$ for some $K = O\left(\frac{m}{\epsilon}\log\frac{m}{\epsilon}\right)$ such that $x^{(\beta,w)}$ is $(1 + \epsilon)^4$-approximate to (P-LP).*

Organization The sections indexed by capital letters are in the supplementary material. We give the deterministic and randomized online rounding algorithms, which prove Theorem 2.1 and Theorem 2.2, in Sections 3 and 4 respectively. In Section 5, we prove Theorem 2.4 that reduces the unrelated machine load balancing instance to one in the $\mathsf{Q}|\mathrm{restricted}$ setting. We defer the proof of Corollary 2.5 to Section C, where we also show how to handle the case when the prediction has errors. We give the formal theorem (Theorem D.1) on the learnability of our prediction and its proof in Section D. All omitted proofs can be found in Section B.

3. $O\left(\frac{\log m}{\log\log m}\right)$-Competitive Deterministic Online Rounding Algorithm

In this section, we obtain a deterministic $O\left(\frac{\log m}{\log\log m}\right)$-competitive rounding algorithm. Recall that in the setting, when a job $j \in [n]$ arrives, it reveals $(p_{i,j})_{i\in M}$ and $(x_{i,j})_{i\in M}$. By Assumption 2, we are given $T = \mathrm{mspn}(x)$ upfront, which guarantees $\sum_{j\in J} x_{i,j}p_{i,j} \leq T$ for every $i \in M$, and if $x_{i,j} > 0$ for some i, j, then $p_{i,j} \leq T$. So, we assume $p_{i,j} \leq T$ for every $(i, j) \in E$.

Our algorithm is based on de-randomizing the simple dependent rounding algorithm which assigns each job j to a machine i with probability $x_{i,j}$ independently. Via Chernoff bound and union bound, we can show the algorithm achieves an $O\left(\frac{\log m}{\log\log m}\right)$-competitive ratio with high probability. To de-randomize it, we use the idea of conditional expectation.

For simplicity, we identify J with $[n]$, and index both jobs and times using $[n]$: For any $j \in [n]$, job j arrives at time j.

We now describe the algorithm. Let $a > 0$ be some parameter whose value will be set to $\ln\ln m$ later. For every $i \in M$ and time t, let $L_{i,t}$ denote the total load of machine i at the end of time t. We use L_i to denote $L_{i,n}$. Our algorithm is simple: when job t arrives, we assign it to a machine $i \in M_t$ so that Φ_t, defined as follows, is minimized:

$$\Phi_t := \sum_{i\in M} \exp\left(\frac{aL_{i,t}}{T} + (e^a-1)\Big(1-\frac{1}{T}\sum_{j=1}^{t} x_{i,j}p_{i,j}\Big)\right).$$

To give some intuition behind the definition, we remark that our goal is to guarantee that $\sum_{i\in M}\exp\left(\frac{aL_{i,n}}{T}\right)$ is at most its expected value when jobs are assigned randomly and independently according to x. Then $\exp\left((e^a-1)\frac{1}{T}\sum_{j=t+1}^{n} x_{i,j}p_{i,j}\right)$ is an upper bound on $\mathbb{E}\left[\exp\left(\frac{a(L_{i,n}-L_{i,t})}{T}\right)\right]$ if we assign all jobs $j \in [t+1,n]$ randomly and independently, that is used as an intermediate bound in the proof of Chernoff bound. Then assuming the worst case that $\frac{1}{T}\sum_{j=1}^{n} x_{i,j}p_{i,j} = 1$ for every $i \in M$ leads to the definition of Φ_t. We show that Φ_t is non-increasing:

Lemma 3.1. *For every $t \in [n]$, we have $\Phi_t \le \Phi_{t-1}$.*

Notice that $\Phi_0 = \sum_{i\in M}\exp(e^a-1) = m\cdot\exp(e^a-1)$. Hence, we have $\Phi_n \le m\cdot\exp(e^a-1)$ at the end of the algorithm. For every $i \in M$, we have $\exp\left(\frac{aL_i}{T}\right) \le \Phi_n \le m\cdot\exp(e^a-1)$. That is, $L_i \le \frac{T}{a}(\ln m + e^a - 1)$. Setting $a = \ln\ln m$, we get $L_i \le \frac{T}{\ln\ln m}(2\ln m - 1) = T\cdot O\left(\frac{\log m}{\log\log m}\right)$ for every $i \in M$.

4. $O\left(\frac{\log\log m}{\log\log\log m}\right)$-Competitive Randomized Online Rounding Algorithm

In this section, we give our randomized online rounding algorithm with $O\left(\frac{\log\log m}{\log\log\log m}\right)$-competitive ratio, proving Theorem 2.2. Our goal is to construct a solution of makespan $O\left(\frac{\log\log m}{\log\log\log m}\right)\cdot T$ w.h.p, where $T = \mathrm{mspn}(x)$ is given upfront. Again recall that we have $p_{i,j} \le T$ for every $(i,j) \in E$ and $\sum_{j\in M_i} p_{i,j}x_{i,j} \le T$ for every $i \in M$. We aim at a success probability of $1-\frac{1}{m}$; but it can be boosted to any $1-\frac{1}{\mathrm{poly}(m)}$. Most of the time we describe the algorithm as if it runs offline. Along the way we argue that it can be easily made online.

Let $\rho = \lceil \log m \rceil$. We say a job j is a *big* job if $\sum_{i\in M_j: p_{i,j}\ge T/\rho} x_{i,j} \ge 1/2$, and *small* otherwise. Notice that if j is small, then we have $\sum_{i\in M_j: p_{i,j}<T/\rho} x_{i,j} > 1/2$. Let J^{big} and J^{small} be the set of big and small jobs respectively. Then for a big job $j \in J^{\mathrm{big}}$ and a machine $i \in M_j$ with $p_{i,j} < T/\rho$, we remove (i,j) from E. (Accordingly, we remove j from J_i, i from M_j, change $p_{i,j}$ to ∞ and discard the variable $x_{i,j}$.) We do the same for any small job $j \in J^{\mathrm{small}}$ and machine $i \in M_j$ with $p_{i,j} \ge T/\rho$.

We sum up the properties we have after the operations:

(P1) For every $j \in J^{\mathrm{big}}$ and $i \in M_j$, we have $p_{i,j} \ge T/\rho$.

(P2) For every $j \in J^{\mathrm{small}}$ and $i \in M_j$, we have $p_{i,j} < T/\rho$.

(P3) $\sum_{i\in M_j} x_{i,j} = \frac{1}{2}$, for every $j \in J$. (Notice that we had $\sum_{i\in M_j} x_{i,j} \ge \frac{1}{2}$; the equality can be obtained by decreasing some $x_{i,j}$ values.)

(P4) For every $i \in M$, we have $\sum_{j\in J_i} p_{i,j}x_{i,j} \le T$.

For convenience we let $J_i^{\mathrm{big}} = J^{\mathrm{big}} \cap J_i$ and $J_i^{\mathrm{small}} = J^{\mathrm{small}} \cap J_i$ denote the sets of big and small jobs that can be assigned to i respectively. Clearly, deciding if a job is small or big and modifying E and x can be made online. In the following, we handle small and big jobs separately.

4.1. Dealing with Small Jobs

Small jobs can be handled easily by independent rounding. For any $j \in J^{\mathrm{small}}$, we assign it to a random machine $i \in M_j$ so that i is the chosen machine with probability $2x_{i,j}$; this can be done because of (P3). Clearly the procedure can be made online. Using Chernoff bound we can show that w.h.p every machine i has a total load $O(T)$ of small jobs.

Lemma 4.1. *With probability at least $1-\frac{1}{4m}$, $\forall i \in M$, the total load of small jobs assigned to i is at most $8T$.*

4.2. Dealing with Big Jobs

Now we focus on big jobs J^{big} and assign them to M. We break the algorithm into 3 stages. First, we apply an initial rounding to obtain a fractional solution x' from x, so that every non-zero value in x' is at least $1/\rho$. Second, we randomly assign big jobs to machines according to x', and a job fails if the machine it is assigned to is already overloaded. Finally, we assign all failed jobs using our deterministic algorithm in Theorem 2.1.

In the actual online algorithm, for each job j in the arrival order, we run the procedures for the job in the three stages. Thus, it is crucial that the procedure for a job j do not depend on the knowledge of the jobs that arrive after j, and the overcome of handling these jobs. One can verify this from the description of the algorithm.

Stage 1: round small x values For every job $j \in J^{\mathrm{big}}$ and machine $i \in M_j$, we randomly set $x'_{i,j}$ so that:

$$x'_{i,j} = \begin{cases} x_{i,j} & \text{if } x_{i,j} \ge 1/\rho \\ \begin{cases} 1/\rho & \text{with probability } \rho x_{i,j} \\ 0 & \text{with probability } 1-\rho x_{i,j} \end{cases} & \text{if } x_{i,j} < 1/\rho \end{cases}.$$

We correlate the variables $\{x'_{i,j}\}_{i\in M_j}$ for the same $j \in J^{\mathrm{big}}$, so that we always have $\sum_{i\in M_j} x'_{i,j} - \sum_{i\in M_j} x_{i,j} \in (-\frac{1}{\rho}, \frac{1}{\rho})$, which is equivalent to

$$\textstyle\sum_{i\in M_j} x'_{i,j} \in \left(\frac{1}{2}-\frac{1}{\rho}, \frac{1}{2}+\frac{1}{\rho}\right). \tag{7}$$

This is possible since all truly-random variables take values in $\{0, \frac{1}{\rho}\}$. On the other hand, we guarantee that the random processes for all jobs $j \in J^{\mathrm{big}}$ are independent.

Notice that $\mathbb{E}[x'_{i,j}] = x_{i,j}$ for every $j \in J^{\mathrm{big}}$ and $i \in M_j$. Also, we have $x'_{i,j} = 0$ or $x'_{i,j} \geq 1/\rho$. The following lemma is a simple application of Chernoff bound.

Lemma 4.2. *With probability at least $1 - \frac{1}{4m}$, for every $i \in M$, we have $\sum_{j \in J_i^{\mathrm{big}}} p_{i,j} x'_{i,j} \leq 5T$.*

From now on, we assume the events in Lemma 4.2 happen.

Stage 2: attempt to assign big jobs The procedure in this stage is formally defined in Algorithm 1. Notice that Step 3 is well-defined as (7) says $\sum_{i \in M_j} x'_{i,j} \geq \frac{1}{2} - \frac{1}{\rho} \geq \frac{1}{3}$.

Algorithm 1 Algorithm in Stage 2

```
1: for every $i \in M$ do $L_i \leftarrow 0$ and let $i$ be unmarked
2: for every $j \in J^{\mathrm{big}}$ in order of arrival do
3:     choose a machine $i \in M_j$ randomly, with the only requirement that the probability $i$ is chosen is at most $3x'_{i,j}$
4:     if $L_i \leq \frac{15 \log\log m}{\log\log\log m} \cdot T$ then
5:         assign $j$ to $i$, and let $L_i \leftarrow L_i + p_{i,j}$
6:     else
7:         claim that $j$ fails to be assigned, and mark $i$
```

Let J^{failed} be the set of jobs in J^{big} that failed, and let M^{marked} be the set of the marked machines at the end of Stage 2. Notice that for any machine i, we have $L_i \leq \frac{15 \log\log m}{\log\log\log m} \cdot T \leq O\left(\frac{\log\log m}{\log\log\log m}\right) \cdot T$. So, in this stage, every machine gets a load of at most $O\left(\frac{\log\log m}{\log\log\log m}\right) \cdot T$.

Lemma 4.3. *For every $i \in M$, we have $\Pr[i \in M^{\mathrm{marked}}] \leq \frac{1}{700\rho^{12}}$.*

Stage 3: schedule J^{failed} deterministically In stage 3, we schedule J^{failed} using our deterministic algorithm in Theorem 2.1, with x' (instead of x) being the fractional solution given online. As in Lattanzi et al. (2020), we show that w.h.p in the graph defined by J^{failed}, M and the support of x', every connected components contains at most $\mathrm{poly}\log(m)$ machines. Then we can make the deterministic algorithm $O\left(\frac{\log \hat{m}}{\log\log \hat{m}}\right)$-competitive, where $\hat{m} = \mathrm{poly}\log(m)$ is the maximum number of machines in any connected component of the graph. Therefore in this stage, each machine i gets a load of $O\left(\frac{\log \hat{m}}{\log\log \hat{m}}\right) \cdot T = O\left(\frac{\log\log m}{\log\log\log m}\right) \cdot T$ w.h.p.

Throughout this section, for any graph $\hat{G} = (\hat{V}, \hat{E})$ and an integer $p \geq 1$, we use $\hat{G}^p$ to denote the graph over $\hat{V}$, in which there is an edge between $u, v \in \hat{V}$ if and only if there is a path of at most p edges in $\hat{G}$ connecting u and v. For any graph $\hat{G} = (\hat{V}, \hat{E})$ and a subset $\hat{U} \subseteq \hat{V}$ of vertices, we use $\hat{G}[\hat{U}]$ to denote the sub-graph of $\hat{G}$ induced by $\hat{U}$.

We let $G' = (M \uplus J^{\mathrm{big}}, E')$ be the support bipartite graph of x': for any $j \in J^{\mathrm{big}}$ and $i \in M_j$, we have $(i,j) \in E'$ if and only if $x'_{i,j} > 0$ (which implies $x'_{i,j} \geq 1/\rho$). The following claim is immediate:

Claim 4.4. *In G', every job $j \in J^{\mathrm{big}}$ has degree at most $\rho/2 + 1$, and every machine $i \in M$ has degree at most $5\rho^2$. $G'^2[M]$ has maximum degree at most $5\rho^3/2$, $G'^4[M]$ has maximum degree at most $25\rho^6/4$, and $G'^8[M]$ has maximum degree at most $625\rho^{12}/16$.*

As we mentioned, it remains to prove the following lemma:

Lemma 4.5. *With probability at least $1 - \frac{1}{4m}$, every connected component of the graph $G'[M \cup J^{\mathrm{failed}}]$ contains at most $\rho(5\rho^3/2 + 1)^2 = \mathrm{poly}\log(m)$ machines.*

To show the lemma, we prove that the negation of the event in Lemma 4.5 implies some event that happens with probability at most $\frac{1}{4m}$.

Let H be the following graph over M. For every pair of distinct machines $i, i' \in M^{\mathrm{marked}}$, we have $(i,i') \in H$ if $(i,i') \in G'^4$. For every $i \in M \backslash M^{\mathrm{marked}}$ and $i' \in M^{\mathrm{marked}}$, we have $(i,i') \in H$ if $(i,i') \in G'^2$; that is, i and i' share a common neighbor in G'. Notice that there are no edges between any two unmarked machines in H.

Lemma 4.6. *The machines in any connected component of $G'[M \cup J^{\mathrm{failed}}]$ are in a same connected component of H.*

Lemma 4.7. *The marked machines in any connected component of H are in a same connected component of $G'^4[M^{\mathrm{marked}}]$.*

The above two lemmas imply the following:

Lemma 4.8. *If some connected component of $G'[M \cup J^{\mathrm{failed}}]$ contains $\rho(5\rho^3/2 + 1)^2$ machines, then some connected component of $G'^4[M^{\mathrm{marked}}]$ has size at least $\rho(5\rho^3/2 + 1)$.*

We say a subset $M' \subseteq M$ of machines is *interesting* if $G'^8[M']$ is connected but M' is an independent set in G'^2.

Lemma 4.9. *Suppose we have a set M^* of at least $\rho(5\rho^3/2 + 1)$ machines such that $G'^4[M^*]$ is connected. Then there is an interesting set $M' \subseteq M^*$ of size at least ρ.*

Lemma 4.10. *With probability at least $1 - \frac{1}{4m}$, every interesting set M' of size ρ contains an unmarked machine.*

Now we have all the ingredients to prove Lemma 4.5.

Proof of Lemma 4.5. By Lemma 4.8, if some connected component of $G'[M \cup J^{\mathrm{failed}}]$ contains at least $\rho(5\rho^3/2 +$

Algorithm 2 construction of initial β

```
1: $x_{i,j} \leftarrow 0, \forall (i,j) \in E, r \leftarrow \frac{\max_{(i,j)\in E} p_{i,j}}{\min_{(i,j)\in E} p_{i,j}}, U \leftarrow 1$
2: while $J \neq \emptyset$ do
3:     solve (P-LP) to obtain $\tilde{x}$ and (D-LP) to obtain $(\tilde{\alpha}, \tilde{\beta})$
4:     $M' \leftarrow \{i \in M : \tilde{\beta}_i > 0\}, J' \leftarrow \{j \in J : \tilde{\alpha}_j > 0\}$
5:     scale $(\tilde{\alpha}, \tilde{\beta})$ so that $\max_{i\in M'} \tilde{\beta}_i$ becomes $U$
6:     $U \leftarrow \frac{\min_{i\in M'} \tilde{\beta}_i}{r}$
7:     let $\alpha_j \leftarrow \tilde{\alpha}_j, \forall j \in J'$, $\beta_i \leftarrow \tilde{\beta}_i, \forall i \in M'$, and
       $x_{i,j} \leftarrow \tilde{x}_{i,j}, \forall j \in J', i \in M', (i,j) \in E$
8:     $M \leftarrow M \setminus M'$ and $J \leftarrow J \setminus J'$
```

$1)^2$ machines, then $G'^4[M^{\text{marked}}]$ has a connected component of size at least $\rho(5\rho^3/2+1)$. Let $M^* \subseteq M^{\text{marked}}$ be the machines in the component. Then by Lemma 4.9, there is an interesting set $M' \subseteq M^* \subseteq M^{\text{marked}}$ of size at least ρ. We can remove machines from M' while keeping $G'^8[M']$ connected to make $|M'| = \rho$.

So, if some component of $G'[M \cup J^{\text{failed}}]$ contains at least $\rho(5\rho^3/2+1)^2$ machines, there is an interesting set $M' \subseteq M^{\text{marked}}$ of size ρ. By Lemma 4.10, the latter event happens with probability at most $\frac{1}{4m}$, so does the former. □

5. Reduction of Unrelated Machine Setting to Related Machine Restricted Assignment Setting: Proof of Theorem 2.4

In this section, we prove Theorem 2.4 that reduces the instance in the unrelated machine setting to one in the Q|restricted setting. The proof of Corollary 2.5 and the handling of errors can be found in Section C.

We can see that if we let (α, β) be the optimum dual solution, then β satisfies all the properties of theorem except that $\beta \in \text{powers}^M_{1+\epsilon,K}$. However this is an important property that can not be ignored. First, if some machine i has $\beta_i = 0$, we will have an invalid instance in the Q|restricted setting. Specifically, all adjacent jobs of machines with β values being 0 have α values being 0. Then we do not have any restriction on how x assigns these jobs to machines. Second, without the property, we could not bound the aspect ratio of β, which determines its bit-complexity after descretization.

Indeed, our algorithm constructs the vector β piece by piece. We take the optimum solution $(\tilde{\alpha}, \tilde{\beta})$ to (D-LP) and copy the non-zero values of $\tilde{\beta}$ to β. Then we remove the jobs and machines with positive $\tilde{\alpha}$ and $\tilde{\beta}$ values. Then we continue to fill the other β values by considering the residual instance. The initial $\beta \in \mathbb{R}^M_{>0}$ is constructed in Algorithm 2. Finally we modify β to make its aspect ratio small, and discretize it.

In Algorithm 2, r is fixed to be the ratio between the max and min $p_{i,j}$ values, U serves as an upper bound on the value of future β values; it decreases as the algorithm proceeds. So, our final β (α and x, resp.) is the combination of all the $\tilde{\beta}$'s ($\tilde{\alpha}$'s and $\tilde{x}$'s, resp.) constructed in all iterations. The way we scale $(\tilde{\alpha}, \tilde{\beta})$ in Step 5 and update U in Step 6 guarantees that the β values assigned in later iterations are at most $\frac{1}{r}$ times the β values assigned in earlier iterations.

Focus on each iteration of Loop 2 in Algorithm 2. We obtain a primal optimum solution $\tilde{x}$ and a dual optimum solution $(\tilde{\alpha}, \tilde{\beta})$. Notice that $\tilde{\alpha}_j = \min_{i\in M_j} p_{i,j}\tilde{\beta}_i$ for every $j \in J$. By complementary slackness conditions we have that $\tilde{x}_{i,j} > 0$ implies $\tilde{\alpha}_j = p_{i,j}\tilde{\beta}_i$. So, in the solution x, any $j \in J'$ is completely assigned to M', and any $j \in J \setminus J'$ is completely assigned to $M \setminus M'$. Since we copied $\tilde{x}$ values between M' and J' to x, x assigns each job $j \in J'$ to an extension of 1 and the makespan of every machine $i \in M'$ is at most T^* (Recall that T^* is the optimum fractional makespan). Moreover, we are guaranteed that the residual instance restricted to $J \setminus J'$ and $M \setminus M'$ admits a primal LP solution of value at most T^*. So the value of (P-LP) can only go down as the algorithm proceeds. So, the final x we constructed is optimum to (P-LP). Moreover, $x_{i,j} > 0$ implies $\alpha_i = p_{i,j}\beta_j$. Also, at the end of the algorithm we have $M = \emptyset$, since otherwise $\bigcup_{i\in M} J_i$ are still in J (assuming no J_i is empty).

We then show $\alpha_j = \min_{i\in M_i} p_{i,j}\beta_i$. Focus on the job j and the iteration in which β_j is assigned; in the iteration we have $\tilde{\alpha}_j = \min_{i\in M'} p_{i,j}\tilde{\beta}_i$. The β values of machines assigned in previous iterations are at least r times bigger than β_j. All machines in M_j will have β values assigned by the end of the iteration. Then $\alpha_j = \min_{i\in M_i} p_{i,j}\beta_i$ follows from the definition of r.

So far we have $\beta \in \mathbb{R}^M_{>0}$ but we need $\beta \in \text{powers}^M_{1+\epsilon,K}$. To gurantee this, we reduce the aspect ratio of β . We sort all β values from the smallest to biggest. If we see two adjacent machines i_1, i_2 in the ordering with $\frac{\beta_{i_2}}{\beta_{i_1}} > \frac{m}{\epsilon}$, then we can scale all β values of jobs after i_1 down by the same factor so that $\frac{\beta_{i_2}}{\beta_{i_1}}$ becomes $\frac{m}{\epsilon}$. We update α_j values accordingly. By Assumption 1, this operation will not change whether a machine i is the one that minimizes $\beta_i p_{i,j}$ or not for any job j. We repeat the operation until we can not find such an adjacent pair. Then we have that $\frac{\max_{i\in M}\beta_i}{\min_{i\in M}\beta_i} \leq \left(\frac{m}{\epsilon}\right)^{m-1}$. By scaling, we assume the smallest β value is 1.

Finally, we round each β_j values down to its nearest integer power of $1+\epsilon$. So after the rounding we have $\beta_j \in \text{powers}(1+\epsilon, K)$ for every j, where $K = \left\lceil \log_{1+\epsilon}\left(\frac{m}{\epsilon}\right)^{m-1} \right\rceil = O\left(\frac{m}{\epsilon}\log\frac{m}{\epsilon}\right)$. We update α_j's accordingly. Before the rounding, $x_{i,j} > 0$ implies $\alpha_j = p_{i,j}\beta_i$. After the rounding, it implies $\alpha_j \leq p_{i,j}\beta_i < (1+\epsilon)\alpha_j$. This finishes the proof of Theorem 2.4.

References

Agrawal, S., Zadimoghaddam, M., and Mirrokni, V. Proportional allocation: Simple, distributed, and diverse matching with high entropy. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 99–108, 2018.

Anand, K., Ge, R., and Panigrahi, D. Customizing ML predictions for online algorithms. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 303–313. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/anand20a.html.

Antoniadis, A., Gouleakis, T., Kleer, P., and Kolev, P. Secretary and online matching problems with machine learned advice. In *Advances in Neural Information Processing Systems*, 2020.

Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., and Waarts, O. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44 (3):486–504, May 1997. ISSN 0004-5411. doi: 10.1145/258128.258201. URL https://doi.org/10.1145/258128.258201.

Azar, Y., Naor, J., and Rom, R. The competitiveness of on-line assignments. *Journal of Algorithms*, 18(2):221 – 237, 1995. ISSN 0196-6774. doi: https://doi.org/10.1006/jagm.1995.1008. URL http://www.sciencedirect.com/science/article/pii/S0196677485710085.

Bamas, E., Maggiori, A., and Svensson, O. The primal-dual method for learning augmented algorithms. In *Advances in Neural Information Processing Systems*, 2020. URL https://papers.nips.cc/paper/2020/file/e834cb114d33f729dbc9c7fb0c6bb607-Paper.pdf.

Bhaskara, A., Cutkosky, A., Kumar, R., and Purohit, M. Online learning with imperfect hints. In *Advances in Neural Information Processing Systems*, 2020.

Boyar, J., Favrholdt, L. M., Kudahl, C., Larsen, K. S., and Mikkelsen, J. W. Online algorithms with advice: A survey. *SIGACT News*, 47(3):93–129, August 2016. ISSN 0163-5700. doi: 10.1145/2993749.2993766. URL https://doi.org/10.1145/2993749.2993766.

Chakrabarty, D., Khanna, S., and Li, S. *On (1,ϵ)-Restricted Assignment Makespan Minimization*. 2015.

Devanur, N. R. Online algorithms with stochastic input. *SIGecom Exch.*, 10(2):40–49, June 2011. doi: 10.1145/1998549.1998558. URL https://doi.org/10.1145/1998549.1998558.

Devanur, N. R., Jain, K., and Kleinberg, R. D. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pp. 101–107, USA, 2013. Society for Industrial and Applied Mathematics. ISBN 9781611972511.

Gollapudi, S. and Panigrahi, D. Online algorithms for rent-or-buy with expert advice. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2319–2327. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/gollapudi19a.html.

Jansen, K. and Rohwedder, L. On the configuration-lp of the restricted assignment problem. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pp. 2670–2678, USA, 2017. Society for Industrial and Applied Mathematics.

Jansen, K. and Rohwedder, L. A quasi-polynomial approximation for the restricted assignment problem. *SIAM Journal on Computing*, 49(6):1083–1108, 2020. doi: 10.1137/19M128257X. URL https://doi.org/10.1137/19M128257X.

Jiang, Z., Panigrahi, D., and Sun, K. Online algorithms for weighted paging with predictions. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming*, ICALP '20, pp. 69:1–69:18, USA, 2020.

Karp, R. M., Vazirani, U. V., and Vazirani, V. V. An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, pp. 352–358, New York, NY, USA, 1990. Association for Computing Machinery. ISBN 0897913612. doi: 10.1145/100216.100262. URL https://doi.org/10.1145/100216.100262.

Kumar, R., Purohit, M., and Svitkina, Z. Improving online algorithms via ml predictions. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 9684–9693, Red Hook, NY, USA, 2018. Curran Associates Inc.

Lattanzi, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Online scheduling via learned weights. SODA '20, pp. 1859–1877, USA, 2020. Society for Industrial and Applied Mathematics.

Lavastida, T., Moseley, B., Ravi, R., and Xu, C. Learnable and instance-robust predictions for online matching, flows and load balancing. *ArXiv*, abs/2011.11743, 2020.

Lenstra, J. K., Shmoys, D. B., and Tardos, É. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990. URL https://doi.org/10.1007/BF01585745.

Lykouris, T. and Vassilvtiskii, S. Competitive caching with machine learned advice. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3296–3305, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/lykouris18a.html.

Mahdian, M. and Yan, Q. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, pp. 597–606, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306911. doi: 10.1145/1993636.1993716. URL https://doi.org/10.1145/1993636.1993716.

Manshadi, V. H., Gharan, S. O., and Saberi, A. Online stochastic matching: Online actions based on offline statistics. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pp. 1285–1294, USA, 2011. Society for Industrial and Applied Mathematics.

Rohatgi, D. Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, pp. 1834–1845, USA, 2020. Society for Industrial and Applied Mathematics.

Schild, A., Vee, E., Purohit, M., Ravikumar, R. K., and Svitkina, Z. Semi-online bipartite matching. 2019.

Svensson, O. Santa claus schedules jobs on unrelated machines. *SIAM Journal on Computing*, 41(5):1318–1341, 2012. doi: 10.1137/110851201. URL https://doi.org/10.1137/110851201.

Wei, A. Better and Simpler Learning-Augmented Online Caching. In Byrka, J. and Meka, R. (eds.), *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, volume 176 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 60:1–60:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. ISBN 978-3-95977-164-1. doi: 10.4230/LIPIcs.APPROX/RANDOM.2020.60. URL https://drops.dagstuhl.de/opus/volltexte/2020/12663.