Minimalistic Predictions to Schedule Jobs with Online Precedence Constraints

Alexandra Lassota [*1] **Alexander Lindermayr** [*2] **Nicole Megow** [*2] **Jens Schlöter** [*2]

Abstract

We consider non-clairvoyant scheduling with *online* precedence constraints, where an algorithm is oblivious to any job dependencies and learns about a job only if all of its predecessors have been completed. Given strong impossibility results in classical competitive analysis, we investigate the problem in a learning-augmented setting, where an algorithm has access to predictions without any quality guarantee. We discuss different prediction models: novel problem-specific models as well as general ones, which have been proposed in previous works. We present lower bounds and algorithmic upper bounds for different precedence topologies, and thereby give a structured overview on which and how additional (possibly erroneous) information helps for designing better algorithms. Along the way, we also improve bounds on traditional competitive ratios for existing algorithms.

1. Introduction

Cloud computing is a popular approach to outsource heavy computations to specialized providers (Hayes, 2008). Concepts like Function-as-a-Service (FaaS) offer users on demand the execution of complex computations in a specific domain (Lynn et al., 2017; Shahrad et al., 2019). Such tasks are often decomposed into smaller jobs, which then depend on each other by passing intermediate results. The structure of such tasks heavily relies on the users input and internal dependencies within the users system. It might require diverse jobs to solve different problems with distinct inputs.

From the providers perspective, the goal is thus to schedule jobs with different priorities and interdependencies which become known only when certain jobs are completed and their results can be evaluated. From a more abstract perspective, we face *online precedence constraint scheduling*: new jobs arrive only if certain jobs have been completed but the set of jobs and their dependencies are unknown to the scheduler. As tasks might have different priorities, it is a natural objective to minimize the total (average) weighted completion time of the jobs. We focus on *non-clairvoyant* schedulers that do not know a job's processing requirement in advance (Motwani et al., 1994), and we allow *preemptive* schedules, i.e., jobs can be interrupted and resumed later. We present and analyze (non-)clairvoyant algorithms and prove impossibility results for this problem.

Competitive analysis is a widely used technique to assess the performance of online algorithms (Borodin & El-Yaniv, 1998). The *competitive ratio* of an algorithm is the maximum ratio over all instances between its objective value and the objective value of an *offline* optimal solution. In our setting, an offline optimal solution is the best schedule that can be computed with complete information and unbounded running time on the instance. We say that an algorithm is ρ-competitive if its competitive ratio is at most ρ.

It is not hard to see that for our problem, we cannot hope for good worst-case guarantees: consider an instance of $n-1$ initially visible jobs with zero weight such that exactly one of these jobs triggers at its completion the arrival of a job with positive weight. Since the initial jobs are indistinguishable, in the worst-case, any algorithm completes the positive-weight job last. An offline optimal solution can distinguish the initially visible jobs and immediately processes the one which triggers the positive-weight job. This already shows that no deterministic algorithm can have a better competitive ratio than $\Omega(n)$ for n jobs. Notice that this strong impossibility result holds even for (seemingly) simple precedence graphs that consist of a collection of chains. In practice, such topology is highly relevant as, e.g., a sequential computer program executes a path (chain) of instructions that upon execution depends on the evaluation of control flow structures (cf. (Allen, 1970)).

To overcome such daunting lower bounds, we consider closer-to-real-world approaches to go beyond worst-case analysis. In particular, we study augmenting *algorithms with predictions* (Mitzenmacher & Vassilvitskii, 2020; 2022). The intuition is that in many applications, we can learn cer-

[*]Equal contribution [1]Institute of Mathematics, EPFL, Lausanne, Switzerland [2]Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany. Correspondence to: Jens Schlöter <jschloet@uni-bremen.de>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

tain aspects of the uncertainty by considering historical data such as dependencies between jobs for certain computations and inputs. While these predictions might not reflect the current instance, they can contain enough information to design algorithms that break pessimistic worst-case lower bounds. Besides specifying the type of information, this requires a measure for a prediction's quality. This allows parameterized performance guarantees of algorithms w.r.t. the amount of information a prediction contains. Important performance indicators are *consistency*, that is the competitive ratio for best-possible predictions, and *robustness*, that is an upper bound on the competitive ratio for any prediction.

Despite the immense research interest in learning augmented algorithms (Lindermayr & Megow, 2023), the particular choice of prediction models remains often undiscussed. In this work, we discuss various models and analyze their strengths and weaknesses. In particular, we present the first learning-augmented algorithms for scheduling with (online) precedence constraints. The question driving our research is:

> *Which particular minimal information is required to achieve reasonable performance guarantees for scheduling with online precedence constraints?*

In this work, we refer with reasonable performance guarantees to competitive ratios in $o(n)$, i.e., bounds which improve upon the pessimistic lower bound of $\Omega(n)$ from the setting without additional information. Our starting point is the analysis of the two most common models, *full input* predictions, c.f. (Purohit et al., 2018; Azar et al., 2021; 2022a; Im et al., 2021; Antoniadis et al., 2022; Bernardini et al., 2022; Erlebach et al., 2022) and *action predictions*, c.f. (Antoniadis et al., 2020; Bamas et al., 2020; Lindermayr et al., 2022; Lindermayr & Megow, 2022; Eberle et al., 2022; Anand et al., 2022; Jin & Ma, 2022). Our main focus is on a hierarchy of refined prediction models based on their entropy. That is, one can compute a prediction for a weaker model using a prediction from a stronger one, but not vice versa. We predict quantities related to the weight of unknown jobs which is in contrast to previous work which assumes predictions on the jobs' processing times or machine speeds (except (Cho et al., 2022)).

For each prediction model, we analyze its power and limits by providing efficient algorithms and lower bounds on the best-possible performance guarantees w.r.t. these models and the topological properties of the precedence constraints.

1.1. Problem Definition and Prediction Models

An instance of our problem is composed of a set J of n jobs and a precedence graph $G = (J, E)$, which is an acyclic directed graph (DAG). Every job $j \in J$ has a processing requirement $p_j \geq 0$ and a weight $w_j \geq 0$. An edge $(j', j) \in E$ indicates that j can only be started if j' has been completed. If there is a directed path from j' to j in G, then we say that j is a *successor* of j' and that j' is a *predecessor* of j. If that path consists of a single edge, we call j and j' a *direct* successor and predecessor, respectively. For a fixed precedence graph G, we denote by ω the *width* of G, which is the length of the longest anti-chain in G.

An algorithm can process a job j at a time $t \geq 0$ with a rate $R_j^t \geq 0$, which describes the amount of processing the job receives at time t. The completion time C_j of a job j is the first time t which satisfies $\sum_{t'=0}^{t} R_j^{t'} \geq p_j$. On a single machine a total rate of 1 can be processed at any time t, thus we require $\sum_{j \in J} R_j^t \leq 1$. At any time t in a schedule, let $F_t = \{j \mid C_j > t \text{ and } \forall j' \text{ s.t. } (j', j) \in E \colon C_{j'} < t\}$ denote the set of unfinished jobs without unfinished predecessors in G. We refer to such jobs as *front jobs*. In the online setting, a job is revealed to the algorithm once all predecessors have been completed. The algorithm is completely oblivious to G, and, in particular, it does not know whether a front job has successors. Thus, at any time t an algorithm only sees jobs $j \in F_t$ with weights w_j but *not* their processing times p_j. Note that the sets F_t heavily depend on an algorithm's actions. At the start time $t = 0$, an algorithm sees F_0, and until the completion of the last job, it does not know the total number of jobs. An algorithm can at any time t only process front jobs, hence we further require that $R_j^t = 0$ for all $j \in J \setminus F_t$. The objective of our problem is to minimize $\sum_{j \in J} w_j C_j$. For a fixed instance, we denote the optimal objective value by OPT and for a fixed algorithm, we denote its objective value by ALG.

We study different topologies of precedence graphs. In addition to general DAGs, we consider *in-forests* resp. *out-forests*, where every node has at most one outgoing resp. incoming edge. Further, we study *chains*, which is a precedence graph that is an in-forest and an out-forest simultaneously. If an in- or out-forest has only one connected component, we refer to it as *in-* and *out-tree*, respectively.

Two of the most studied prediction models are: a *full input prediction*, which is a prediction on the set of jobs with processing times and weights, and the complete precedence graph, and an *action prediction*, which is a prediction on a full priority order over all jobs predicted to be part of the instance (*static*) or a prediction on which job to schedule next whenever a machine idles (*adaptive*).

Full input predictions however require a significant amount of information on the input. The same holds for provably optimal action predictions that go beyond simple heuristics which can be computed based on limited information. This might be unrealistic or costly to obtain and/or not necessary. We aim for minimalistic extra information and quantify its power.

The set of front jobs F_0 does not give sufficient information

for obtaining a competitive ratio better than $\Omega(n)$, as shown above. For a job $v \in F_0$, we define the set $S(v)$ consisting of v and its successors, and we let $w(S(v)) := \sum_{u \in S(v)} w_u$. We consider various predictions on the set $S(v)$:

Weight predictions: Predictions $\hat{W}_v$ on the total weight $w(S(v))$ of each front job $v \in F_0$.

Weight order predictions: The *weight order* $\preceq_0$ over F_0 sorts the jobs $v \in F_0$ by non-increasing $w(S(v))$, i.e., $v \preceq_0 u$ implies $w(S(v)) \geq w(S(u))$. We assume access to a prediction $\hat{\preceq}_0$ on $\preceq_0$.

Average predictions: Predictions $\hat{a}_v$ on the average weight $a(S(v)) = \frac{\sum_{u \in S(v)} w_u}{\sum_{u \in S(v)} p_u}$ of each front job $v \in F_0$.

For each of these three models, we distinguish *static* and *adaptive* predictions. Static predictions refer to predictions only on the initial front jobs F_0, and adaptive predictions refer to a setting where we receive access to a new prediction whenever a job becomes visible.

1.2. Our Results

Our results can be separated into two categories. First, we consider the problem of scheduling with online precedence constraints with access to additional *reliable* information. In particular, we consider all the aforementioned prediction models and design upper and lower bounds for the online problem enhanced with access to the respective additional information. We classify the power of the different models when solving the problem on different topologies.

For the second type of results, we drop the assumption that the additional information is accurate and turn our pure online results into learning-augmented algorithms. We define suitable error measures for the different prediction models to capture the accuracy of the predictions, and give more fine-grained competitive ratios depending on these measures. We also extend our algorithms to achieve robustness.

Next, we give an overview of our results for these categories. We state all results for the single machine setting but show in Appendix E that they extend to identical parallel machines.

Reliable additional information Table 1 summarizes our results for the pure online setting enhanced with reliable additional information. Our main results are a 4-competitive algorithms for chains and out-forests with weight predictions, and a H_ω-competitive algorithm for out-forests with adaptive weight order predictions, where H_k is the kth harmonic number. The results show that additional information significantly improves the (worst-case) ratio compared to the setting with no predictions.

Our main non-clairvoyant algorithm, given correct weight predictions, has a competitive ratio of at most 4 for online out-forest precedence constraints on a single machine. This improves even for offline precedence constraints upon previous best-known bounds of 8 (Jäger, 2021) and 10 (Garg et al., 2019) for this problem, although these bounds also hold in more general settings. To achieve this small constant, we generalize the Weighted Round Robin algorithm (WRR) (Motwani et al., 1994; Kim & Chwa, 2003) for non-clairvoyant scheduling *without* precedence constraints, which advances jobs proportional to their weight, to our setting. We handle each out-tree as a super-job and update its remaining weight when a sub-job completes. If the out-tree is a chain, this can be done even if only *static* weight predictions are given. Otherwise, when an out-tree gets divided into multiple remaining out-trees, the distribution of the remaining weight is unknown, thus we have to rely on *adaptive* predictions. Due to the increased dynamics of gaining partial weight of these super-jobs, the original analysis of WRR is not applicable. Instead, we use the dual-fitting technique, which has been previously used for offline precedence constraints (Garg et al., 2019). While their analysis builds on offline information and is infeasible in our model, we prove necessary conditions on an algorithm to enable the dual-fitting, which are fulfilled even in our limited information setting. Surprisingly, we also show that a more compact linear programming (LP) relaxation, which does not consider transitive precedences, is sufficient for our result. In particular, compared to the LP used in (Garg et al., 2019), it allows us to craft simple duals which do not involve gradient-type values of the algorithm's rates.

In the more restricted model of weight order predictions, WRR cannot directly be applied, as even the initial rate computation of the algorithm crucially relies on *precise* weight values (cf. Section 3.1). We observe, however, that WRR's rates at the start of an instance have the same ordering as the known chain order. We show that guessing rates for

Table 1. Summary of bounds on the competitive ratio given reliable information. We denote by P the total processing time and by H_k the kth harmonic number.

Prediction Model	Topology	Bound
Actions	DAG	$\Theta(1)$
Input	DAG	$\Theta(1)$
Adaptive weights	Out-Forests	$\Theta(1)$
Adaptive weights	In-Trees	$\Omega(\sqrt{n})$
Static weights	Out-Trees	$\Omega(n)$
Static weights	Chains	$\Theta(1)$
Adaptive weight order	Out-Forests	$\mathcal{O}(H_\omega)$
Static weight order	Chains	$\mathcal{O}(H_\omega^2\sqrt{P})$
Adaptive averages	Chains	$\Omega(\sqrt{n})$
No Prediction	Chains	$\Omega(n)$

chains in a way that respects the ordering compromises only a factor of at most H_ω in the competitive ratio. If the weight order is adaptive, we show a competitive ratio of $4 \cdot H_\omega$. Otherwise, we give a worse upper bound and evidence that this might be best-possible for this algorithm.

Learning-augmentation We extend our algorithmic results by designing suitable error measures for the different prediction models and proving error-dependent competitive ratios. Finally, we show how existing techniques can be used to give these algorithms a robustness of $\mathcal{O}(\omega)$ at the loss of only a constant factor in the error-dependent guarantee. Note that a robustness $\mathcal{O}(\omega)$ matches the lower bound for the online problem without access to additional information.

1.3. Further Related Work

Scheduling jobs with precedence constraints to minimize the sum of (weighted) completion times has been one of the most studied scheduling problems for more than thirty years. The offline problem is known to be NP-hard, even for a single machine (Lawler, 1978; Lenstra & Kan, 1978), and on two machines, even when precedence constraints form chains (Du et al., 1991; Timkovsky, 2003). Several polynomial-time algorithms based on different linear programming formulations achieve an approximation ratio of 2 on a single machine, whereas special cases are even solvable optimally; we refer to (Correa & Schulz, 2005; Ambühl & Mastrolilli, 2009) for comprehensive overviews. For scheduling on m parallel identical machines, the best known approximation factor is $3 - 1/m$ (Hall et al., 1997).

For scheduling with online precedence constraints, strong and immediate lower bounds rule out a competitive ratio better than $\Omega(n)$ for the min-sum objective. Therefore, online scheduling has been mainly studied in a setting where jobs arrive online but once a job arrives its processing time, weight, and relation to other (already arrived) jobs is revealed (Hall et al., 1997; Chakrabarti et al., 1996; Bienkowski et al., 2021). Similarly, we are not aware of any previous learning-augmented algorithm for online precedence constraints and/or weight predictions. Previous work on minimizing the total completion time focussed on the setting where jobs either arrive online and/or are clairvoyant (Purohit et al., 2018; Im et al., 2021; Lindermayr & Megow, 2022; Dinitz et al., 2022; Bampis et al., 2022).

2. Robustness via Time-Sharing

Before we move to concrete algorithmic results, we quickly argue that any ρ-competitive algorithm for scheduling with online precedence constraints of width ω can be extended to a $\mathcal{O}(\min\{\rho, \omega\})$-competitive algorithm. In particular, if ρ depends on a prediction's quality, this ensures that this algorithm is robust against arbitrarily bad predictions.

To this end, consider the algorithm which at any time t shares the machine equally among all front jobs F_t, i.e., gives every job $j \in F_t$ rate $R_j^t = \frac{1}{|F_t|} \geq \frac{1}{\omega}$. For a fixed job j, compared to its completion time in a fixed optimal schedule, the completion time in the algorithm's schedule can be delayed by at most a factor of ω. We conclude:

Proposition 2.1. *There is an ω-competitive non-clairvoyant single-machine algorithm for minimizing the total weighted completion time of jobs with online precedence constraints.*

We can now use the *time-sharing technique* to combine this ω-competitive algorithm with any other algorithm for scheduling online precedence constraints while retaining the better competitive ratio of both up to a factor of 2.

Theorem 2.2 ((Purohit et al., 2018; Lindermayr & Megow, 2022))**.** *Given two deterministic algorithms with competitive ratios $\rho_\mathcal{A}$ and $\rho_\mathcal{B}$ for minimizing the total weighted completion time with online precedence constraints on identical machines, there exists an algorithm for the same problem with a competitive ratio of at most $2 \cdot \min\{\rho_\mathcal{A}, \rho_\mathcal{B}\}$.*

In (Purohit et al., 2018; Lindermayr & Megow, 2022) there is an additional monotonicity requirement which we claim to be unnecessary; see Appendix F.

3. Weight Value Predictions

We begin with problem-specific prediction models, starting with weight value predictions. We first prove strong lower bounds for algorithms with access to static weight predictions on out-trees and adaptive predictions on in-trees. Then, we give 4-competitive algorithms for accurate static predictions on chains, and adaptive weight predictions on out-forest precedence constraints, and finally extend these results to obtain robust algorithms with error dependency.

The lower bound for out-trees adds a dummy root r to the pure online lower bound composed of $\Omega(n)$ zero weight jobs, where exactly one hides a valuable job. In the static prediction setting we thus only receive a prediction for r, which does not help any algorithm to improve.

Observation 3.1. *Any algorithm which has only access to static weight predictions has a competitive ratio of at least $\Omega(n)$, even if the precedence constraint graph is an out-tree.*

For in-trees and adaptive weight predictions, we prove the following lower bound in Appendix A.1.

Lemma 3.2. *Any algorithm which has only access to adaptive weight predictions has a competitive ratio of at least $\Omega(\sqrt{n})$, even for in-tree precedence constraints.*

3.1. Algorithms for Reliable Information

We present algorithms assuming access to *correct* static or adaptive weight predictions and prove their competitiveness

Algorithm 1 Weighted Round Robin on Chains

Require: Chains $\mathcal{C}$, initial total weight W_c for each $c \in \mathcal{C}$.
1: $t \leftarrow 0$ and $W_c(t) \leftarrow W_c$ for every $c \in \mathcal{C}$.
2: **while** there are unfinished jobs **do**
3: Process the front job j_c of every $c \in \mathcal{C}$ at rate $\frac{W_c(t)}{\sum_c W_c(t)}$.
4: If j_c completes, update $W_c(t+1) \leftarrow W_c(t) - w_{j_c}$, else $W_c(t+1) \leftarrow W_c(t)$.
5: $t \leftarrow t+1$
6: **end while**

on online chain and out-forest precedence constraints using a unified analysis framework. This uses a dual-fitting argumentation inspired by an analysis of an algorithm for *known* precedence constraints (Garg et al., 2019). The framework only requires a condition on the rates at which an algorithm processes front jobs, hence it is independent of the considered prediction model. Let U_t refer to the set of unfinished jobs at time t, i.e., $U_t = \bigcup_{v \in F_t} S(v)$. Denote by $w(J')$ the total weight of jobs in a set J'. We write $W(t)$ for $w(U_t)$.

Theorem 3.3. *If an algorithm for online out-forest precedence constraints satisfies at every time t and for each $j \in F_t$ that $w(S(j)) \leq \rho \cdot R_j^t \cdot W(t)$, where R_j^t is the processing rate of j at time t, it is at most 4ρ-competitive for minimizing the total weighted completion time on a single machine.*

We first present algorithms for weight predictions and derive results using Theorem 3.3, and finally prove the theorem.

Static Weight Values for Chains We give an algorithm for correct static weight predictions. As Observation 3.1 rules out well-performing algorithms for out-tree precedence constraints with static weight predictions, we focus on chains. Correct static weight predictions mean access to the total weight W_c of every chain c in the set of chains $\mathcal{C}$.

Algorithm 1, essentially, executes a classical weighted round robin algorithm where the rate at which the front job of a chain c is executed at time t is proportional to the total weight of unfinished jobs in that chain, $W_c(t)$. As this definition is infeasible for unfinished chains with $W_c(t) = 0$, we process these in an arbitrary order in the end. As they have no weight, this does not negatively affect the objective.

Despite initially only having access to the weights $W_c(t)$ for $t = 0$, the algorithm can compute $W_c(t)$ for any $t > 0$ by subtracting the weight of finished jobs of c from the initial W_c (cf. Line 5). Thus, $W_c(t) = w(S(j))$ holds for any time t and every $j \in F_t$, where c is the corresponding chain of job j. Further, $W(t) = \sum_c W_c(t)$. We conclude that, for any t and $j \in F_t$, it holds $R_j^t = \frac{w(S(j))}{W(t)}$. Using Theorem 3.3 with $\rho = 1$, we derive the following result:

Theorem 3.4. *Given correct weight predictions, Algorithm 1 is a non-clairvoyant 4-competitive algorithm for minimizing the total weighted completion time of jobs with online chain precedence constraints on a single machine.*

We remark that Algorithm 1 crucially relies on access to the *precise* weight values. Even if all chain weights were just overpredicted by the same constant factor $\alpha > 1$, the algorithm would not be constant competitive anymore. While the algorithm would compute the same initial rates as it does with access to the correct weights, it cannot precisely recompute the rates in Line 4. We show in Appendix A.2 that this leads to a competitive ratio of at least $\Omega(\frac{\alpha-1}{\alpha} n^{\frac{1}{3}})$.

Adaptive Weight Values for Out-Forests Observation 3.1 states that static weight predictions are not sufficient to obtain $\mathcal{O}(1)$-competitive algorithms for out-forests. The reason is that we, in contrast to chains, cannot recompute $\hat{W}_j$ whenever a new front job j appears. For adaptive predictions, however, we do not need to recompute $\hat{W}_j$, as we simply receive a new prediction. Thus, we can process every front job $j \in F_t$ with rate $R_j^t = \frac{\hat{W}_j}{\sum_{j' \in F_t} \hat{W}_{j'}}$. For correct predictions, Theorem 3.3 directly implies the following.

Theorem 3.5. *Given correct adaptive weight predictions, there exists a non-clairvoyant 4-competitive algorithm for minimizing the total weighted completion time of jobs with online out-forest precedence constraints on a single machine.*

Full Proof of Theorem 3.3 Fix an algorithm satisfying the conditions of Theorem 3.3. Let ALG be the objective value of the algorithm's schedule for a fixed instance. We introduce a linear programming relaxation similar to the one in (Garg et al., 2019) for our problem on a machine running at lower speed $\frac{1}{\alpha}$, for some $\alpha \geq 1$. Let OPT_α denote the optimal objective value for the problem with speed $\frac{1}{\alpha}$. As the completion time of every job is linear in the machine speed, we have $\text{OPT}_\alpha \leq \alpha \cdot \text{OPT}$. The variable $x_{j,t}$ denotes the fractional assignment of job j at time t. The first constraint ensures that every job receives enough amount of processing to complete, the second constraint restricts the avaible rate per time to $\frac{1}{\alpha}$, and the final constraint asserts that no job can be completed before its predecessors.

$$
\begin{aligned}
\min \quad & \sum_{j,t} w_j \cdot t \cdot \frac{x_{j,t}}{p_j} && && (\text{LP}_\alpha) \\
\text{s.t.} \quad & \sum_t \frac{x_{j,t}}{p_j} \geq 1 && \forall j \\
& \sum_j \alpha \cdot x_{j,t} \leq 1 && \forall t \\
& \sum_{s \leq t} \frac{x_{j,s}}{p_j} \geq \sum_{s \leq t} \frac{x_{j',s}}{p_{j'}} && \forall t, \forall (j,j') \in E \\
& x_{j,t} \geq 0 && \forall j, t
\end{aligned}
$$

The dual of (LP_α) can be written as follows.

$$\begin{aligned} \max \quad & \sum_j a_j - \sum_t b_t & & (\text{DLP}_\alpha) \\ \text{s.t.} \quad & \sum_{s \geq t} \left(\sum_{j':(j,j') \in E} c_{s,j \to j'} - \sum_{j':(j',j) \in E} c_{s,j' \to j} \right) & & \\ & \leq \alpha \cdot b_t \cdot p_j - a_j + w_j \cdot t & \forall j, t & \quad (1) \\ & a_j, b_t, c_{t,j \to j'} \geq 0 & \forall t, \forall (j,j') \in E & \end{aligned}$$

Let $\kappa > 1$ be a constant which we fix later. We define a variable assignment for (DLP_α) as follows: $\bar{a}_j = \sum_{s \geq 0} \bar{a}_{j,s}$ for every job j, where $\bar{a}_{j,s} = w_j$ if $s \leq C_j$ and $\bar{a}_{j,s} = 0$ otherwise, $\bar{b}_t = \frac{1}{\kappa} \cdot W(t)$ for every time t, and $\bar{c}_{t,j' \to j} = w(S(j))$ if $j, j' \in U_t$, and $\bar{c}_{t,j' \to j} = 0$ otherwise, for every time t and edge $(j', j) \in E$.

We show in the following that the variables $(\bar{a}_j, \bar{b}_t, \bar{c}_{t,j \to j'})$ define a feasible solution for (DLP_α) and achieve an objective value close to ALG. Weak duality then implies Theorem 3.3. First, consider the objective value.

Lemma 3.6. $\sum_j \bar{a}_j - \sum_t \bar{b}_t = (1 - \frac{1}{\kappa})\text{ALG}$.

Proof. Note that $\bar{a}_j = w_j C_j$, and thus $\sum_j \bar{a}_j = \text{ALG}$. Also, since the weight w_j of a job j is contained in $W(t)$ if $t \leq C_j$, we conclude $\sum_t \bar{b}_t = \frac{1}{\kappa}\text{ALG}$. □

Second, we show that the duals are feasible for (DLP_α).

Lemma 3.7. *Assigning $a_j = \bar{a}_j$, $b_t = \bar{b}_t$ and $c_{t,j \to j'} = \bar{c}_{t,j \to j'}$ is feasible for (DLP_α) if $\alpha \geq \kappa\rho$ and $\frac{w(S(j))}{W(t)} \leq \rho R_j^t$ for any time t for $j \in F_t$ and the algorithm's rates R_j^t.*

Proof. Since our defined variables are non-negative by definition, it suffices to show that this assignment satisfies (1). Fix a job j and a time $t \geq 0$. By observing that $\bar{a}_j - t \cdot w_j \leq \sum_{s \geq t} \bar{a}_{j,s}$, it suffices to verify

$$\sum_{s \geq t} \left(\bar{a}_{j,s} + \sum_{(j,j') \in E} \bar{c}_{s,j \to j'} - \sum_{(j',j) \in E} \bar{c}_{s,j' \to j} \right) \leq \alpha \bar{b}_t p_j. \tag{2}$$

To this end, we consider the terms of the left side for all times $s \geq t$ separately. For any s with $s > C_j$, the left side of (2) is zero, because $\bar{a}_{j,s} = 0$ and $j \notin U_s$.

Otherwise, if $s \leq C_j$, let t_j^* be the first point in time after t when j is available, and let $s \in [0, t_j^*)$. Then, $j \in U_s$, and since each vertex in an out-forest has at most one direct predecessor, there must be a unique job $j_1 \in U_s$ with $(j_1, j) \in E$. Thus, $\bar{c}_{s,j_1 \to j} = w(S(j))$ and $\bar{c}_{s,j \to j'} = w(S(j'))$ for all $(j, j') \in E$. Observe that in out-forests, we have $S(j') \cap S(j'') = \emptyset$ for all $j' \neq j''$ with $(j, j'), (j, j'') \in E$. This implies $\sum_{(j,j') \in E} \bar{c}_{s,j \to j'} = w(S(j)) - w_j$ and $\sum_{(j,j') \in E} \bar{c}_{s,j \to j'} - \sum_{(j_1,j) \in E} \bar{c}_{s,j_1 \to j} = -w_j$. Hence,

$$\bar{a}_{j,s} + \sum_{(j,j') \in E} \bar{c}_{s,j \to j'} - \sum_{(j',j) \in E} \bar{c}_{s,j' \to j} \leq w_j - w_j = 0.$$

Therefore, proving (2) reduces to proving

$$\sum_{s=t_j^*}^{C_j} \left(w_j + \sum_{(j,j') \in E} \bar{c}_{s,j \to j'} - \sum_{(j',j) \in E} \bar{c}_{s,j' \to j} \right) \leq \alpha \bar{b}_t p_j. \tag{3}$$

Now, let $s \in [t_j^*, C_j)$. There cannot be an unfinished job preceding j, thus $\sum_{(j',j) \in E} \bar{c}_{s,j' \to j} = 0$. Observe that if there is a job $j' \in U_s$ with $(j, j') \in E$, the fact that $j \in U_s$ implies $j' \in U_s$, and thus $\bar{c}_{s,j \to j'} = w(S(j'))$ by definition. Using again the fact that the sets $S(j')$ are pairwise disjoint for all direct successors j' of j, i.e., for all $(j, j') \in E$, this yields $\sum_{(j,j') \in E} \bar{c}_{s,j \to j'} = w(S(j)) - w_j$, and further gives

$$w_j + \sum_{(j,j') \in E} \bar{c}_{s,j \to j'} - \sum_{(j',j) \in E} \bar{c}_{s,j' \to j} = w(S(j)).$$

Thus, the left side of (3) is equal to $\sum_{s=t_j^*}^{C_j} w(S(j))$.

The facts that $W(t_1) \geq W(t_2)$ at any $t_1 \leq t_2$ and that j is processed by $R_j^{t'}$ units at any time $t' \in [t_j^*, C_j]$ combined with the assumption $\frac{w(S(j))}{W(t)} \leq \rho \cdot R_j^t$ imply the following:

$$\sum_{s=t_j^*}^{C_j} \frac{w(S(j))}{W(t)} \leq \sum_{s=t_j^*}^{C_j} \frac{w(S(j))}{W(s)} \leq \sum_{s=t_j^*}^{C_j} \rho \cdot R_j^s \leq \rho \cdot p_j.$$

Rearranging it, using the definition of $\bar{b}_t$ and $\alpha \geq \kappa\rho$ gives

$$\sum_{s=t_j^*}^{C_j} w(S(j)) \leq \rho \cdot p_j \cdot W(t) = \rho\kappa \cdot p_j \cdot \bar{b}_t \leq \alpha \cdot p_j \cdot \bar{b}_t,$$

which implies (3) and thus proves the statement. □

Proof of Theorem 3.3. We set $\alpha = \rho\kappa$. Weak LP duality, Lemma 3.7, and Lemma 3.6 imply

$$\rho\kappa \cdot \text{OPT} \geq \text{OPT}_{\rho\kappa} \geq \sum_j \bar{a}_j - \sum_t \bar{b}_t = \left(1 - \frac{1}{\kappa}\right) \cdot \text{ALG}.$$

Choosing $\kappa = 2$, we conclude that $\text{ALG} \leq 4\rho \cdot \text{OPT}$. □

3.2. Learning-Augmented Algorithms

In this section, we extend the algorithms presented in Section 3.1 to achieve a smooth error-dependency in the case of inaccurate predictions, while preserving constant consistency. Further, we use the time-sharing technique (cf. Section 2) to ensure a robustness of $\mathcal{O}(\omega)$.

Static Weight Predictions for Chains Here, the main challenges are as follows: we only have access to the potentially wrong predictions $\hat{W}_c$ on the total chain weight for all $c \in \mathcal{C}$ and, therefore, we execute Algorithm 1 using $\hat{W}_c$ instead of W_c. In particular, the weight of a chain c might be *underpredicted*, $\hat{W}_c < W_c$, or *overpredicted*, $\hat{W}_c > W_c$. This means that $\sum_c \hat{W}_c$ may not be the accurate total weight of the instance and that the recomputation of $W_c(t)$ in Line 5 may be inaccurate. In Appendix A.2, we show how to encode the error due to underpredicted chains in an instance $\mathcal{C}_u$ and the error due to overpredicted chains in an instance $\mathcal{C}_o$, similar to an error-dependency proposed in (Bamas et al., 2020) for online set cover. We prove the following result:

Theorem 3.8. *For minimizing the total weighted completion time of jobs with online chain precedence constraints on a single machine, there is a non-clairvoyant algorithm with predicted chain weights with a competitive ratio of at most*

$$\mathcal{O}(1) \cdot \min\left\{1 + \frac{\textsc{Opt}(\mathcal{C}_o) + \omega \cdot \textsc{Opt}(\mathcal{C}_u)}{\textsc{Opt}}, \omega\right\}.$$

Adaptive Weight Predictions for Out-Forests To capture the quality of an adaptive prediction, we intuitively need to measure its quality over the whole execution. To this end, we use the maximal distortion factor of the weight predictions of every possible front job, which in fact can be any job in J. We prove in Appendix A.3:

Theorem 3.9. *For minimizing the total weighted completion time on a single machine with online out-forest precedence constraint and adaptive weight predictions, there is a non-clairvoyant algorithm with a competitive ratio of at most*

$$\mathcal{O}(1) \cdot \min\left\{\max_{v \in J} \frac{\hat{W}_v}{w(S(v))} \cdot \max_{v \in J} \frac{w(S(v))}{\hat{W}_v}, \omega\right\}.$$

4. Weight Order Predictions

Next we consider static and adaptive weight order predictions. As strong lower bounds hold for in-trees, even for the more powerful adaptive weight predictions (cf. Lemma 3.2), we focus on chains and out-forest precedence constraints.

Further, we introduce an error measure for wrongly predicted orders. A natural function on orders is the *largest inversion*, i.e., the maximum distance between the position of a front job in an order prediction $\hat{\preceq}_t$ and the true order $\preceq_t$. However, if all out-trees have almost the same weight, just perturbed by some small constant, this function indicates a large error for the reverse order, although it will arguably perform nearly as good as the true order. To mitigate this overestimation, we first introduce ϵ-approximate inversions. Formally, for every precision constant $\epsilon > 0$, we define

$$\mathcal{L}(\epsilon) = \max_{t, j \in F_t} \left| \left\{ i \in F_t \middle| \frac{w(S(j))}{1+\epsilon} \geq w(S(i)) \wedge i \hat{\preceq}_t j \right\} \right|.$$

Algorithm 2 Adaptive weight order algorithm

Require: Time t, front jobs F_t, adaptive order $\hat{\preceq}_t$.

1: Process every $j \in F_t$ with rate $(H_{|F_t|} \cdot i_j)^{-1}$, where i_j is the position of j in $\hat{\preceq}_t$.

Note that $\mathcal{L}(\epsilon) \geq 1$ for every $\epsilon > 0$, because $\hat{\preceq}_t$ is reflexive. We define the *ϵ-approximate largest inversion* error as $\max\{1+\epsilon, \mathcal{L}(\epsilon)\}$. We show performance guarantees depending on this error which hold for any $\epsilon > 0$. Therefore, we intuitively get a pareto frontier between the precision $(1+\epsilon)$ and $\mathcal{L}(\epsilon)$, the largest distance of inversions which are worse than the precision. A configurable error with such properties has been applied to other learning-augmented algorithms (Azar et al., 2022b; Bernardini et al., 2022).

4.1. Adaptive Weight Order

We introduce Algorithm 2, which exploits access to the adaptive order $\hat{\preceq}_t$. In a sense, the idea of the algorithm is to emulate Algorithm 1 for weight predictions. Instead of having access to the total remaining weight of every out-tree to computing rates, Algorithm 2 uses $\hat{\preceq}_t$ to approximate the rates. For every front job $j \in F_t$, let i_j be the position of j in $\hat{\preceq}_t$. Recall that H_k denotes the kth harmonic number.

Theorem 4.1. *For any $\epsilon > 0$, Algorithm 2 has a competitive ratio of at most $4H_\omega \cdot \max\{1+\epsilon, \mathcal{L}(\epsilon)\}$ for minimizing the total weighted completion time on a single machine with online out-forest precedence constraints.*

Proof. We first observe that the rates of the algorithm are feasible, because $\sum_{j \in F_t} \frac{1}{H_{|F_t|} \cdot i_j} = \frac{H_{|F_t|}}{H_{|F_t|}} = 1$.

Fix a time t and an $\epsilon > 0$. Assume that $j_1 \hat{\preceq}_t \ldots \hat{\preceq}_t j_{|F_t|}$, and fix a front job $j_i \in F_t$. The algorithm processes j_i at time t with rate $R^t_{j_i} = (H_{|F_t|} \cdot i)^{-1} \geq (H_\omega \cdot i)^{-1}$. Note that showing $\frac{w(S(j_i))}{W(t)} \leq H_\omega \cdot \max\{1+\epsilon, \mathcal{L}(\epsilon)\} \cdot R^t_{j_i}$ implies the theorem via Theorem 3.3. Assume otherwise, i.e., $\frac{w(S(j_i))}{W(t)} > \frac{1}{i} \cdot \max\{1+\epsilon, \mathcal{L}(\epsilon)\}$. For the sake of readability, we define $K_> = \{k \in [i-1] \mid w(S(i_k)) > \frac{w(S(j_i))}{1+\epsilon}\}$ and $K_\leq = \{k \in [i] \mid w(S(i_k)) \leq \frac{w(S(j_i))}{1+\epsilon}\}$. Since in an out-forest the sets $S(j)$ are pairwise disjoint for all front jobs $j \in F_t$,

$$1 \geq \sum_{k \in [i]} \frac{w(S(i_k))}{W(t)} \geq \sum_{k \in K_>} \frac{w(S(i_k))}{W(t)} + \sum_{k \in K_\leq} \frac{w(S(i_k))}{W(t)}.$$

Consider the second sum. First, observe that this sum has at most $\mathcal{L}(\epsilon)$ many terms, including the one for j_i, and that each such term is at most $\frac{w(S(j_i))}{W(t)}$. Then, observe that every term in the first sum is at least $\frac{w(S(j_i))}{(1+\epsilon)W(t)}$. Thus, we can

further lower bound the sum of the two sums by

$$\frac{1}{1+\epsilon}\sum_{k\in K_>}\frac{w(S(j_i))}{W(t)}+\frac{1}{\mathcal{L}(\epsilon)}\sum_{k\in K_\leq}\frac{w(S(j_i))}{W(t)}$$
$$\geq\frac{1}{\max\{1+\epsilon,\mathcal{L}(\epsilon)\}}\sum_{k\in[i]}\frac{w(S(j_i))}{W(t)}>\sum_{k=1}^{i}\frac{1}{i}=1.$$

This is a contradiction. □

Using this theorem, we conclude the following corollary.

Corollary 4.2. *There exists a non-clairvoyant weight-oblivious algorithm for the problem of minimizing the total weighted completion time of n jobs on a single machine with a competitive ratio of at most $\mathcal{O}(\log n)$ when given access to the order of the job's weights.*

4.2. Static Weight Order

If we only have access to $\hat{\preceq}_0$, a natural approach would be to compute the initial rates as used in Algorithm 2 and just not update them. As Observation 3.1 rules out well-performing algorithms for out-trees, we focus on chains. Even for chains, we show in Appendix B that this algorithm has a competitive ratio of at least $\Omega(\omega\cdot H_\omega)$.

Lemma 4.3. *The variant of Algorithm 2 that computes the rates using $\hat{\preceq}_0$ instead of $\hat{\preceq}_t$ is at least $\Omega(\omega\cdot H_\omega)$-competitive, even if $\hat{\preceq}_0$ equals $\preceq_0$.*

However, the lower bound instance of the lemma requires ω to be "small" compared to the number of jobs, in case of unit jobs, or to $P:=\sum_j p_j$, otherwise. We exploit this to prove the following theorem in Appendix B.

Theorem 4.4. *For any $\epsilon>0$, Algorithm 2 has a competitive ratio of at most $\mathcal{O}(H_\omega^2\sqrt{P}\cdot\max\{1+\epsilon,\mathcal{L}(\epsilon)\})$ when computing rates with $\hat{\preceq}_0$ instead of $\hat{\preceq}_t$ at any time t. For unit jobs, it is $\mathcal{O}(H_\omega^2\sqrt{n}\cdot\max\{1+\epsilon,\mathcal{L}(\epsilon)\})$-competitive.*

5. Average Predictions

Recall that average predictions give access to predicted values $\hat{a}_v$ on $a(S(v))=(\sum_{u\in S(v)}w_u)/(\sum_{u\in S(v)}p_u)$ for each $v\in F_t$. In Appendix C, we show the following lower bound for chains with unit jobs, where average predictions coincide with the average weight of the jobs in the respective chain. The lower bound exploits that we can append jobs of weight zero to a chain in order to manipulate the average weight of the chain until all chains have the same average.

Lemma 5.1. *Any algorithm which has only access to correct adaptive average predictions is at least $\Omega(\sqrt{n})$-competitive even for chain precedence constraints with unit jobs.*

6. Action Predictions

We now turn our focus to *general* prediction models, which are not specifically tailored for our concrete problem. Action predictions induce an optimal algorithm. Hence, following accurate predictions clearly results in an optimal solution. To define an error measure for erroneous static and adaptive action predictions, let $\hat{\sigma}:J\to[n]$ be the order in which a fixed static or adaptive action prediction suggests to process jobs. In case of static action predictions, we receive the predicted order initially, meaning that it might predict a set of jobs $\hat{J}$ different to the actual J. During the analysis, we can simply remove the jobs $\hat{J}\setminus J$ from $\hat{\sigma}$ as they do not have any effect on the schedule for the actual instance. For the jobs in $J\setminus\hat{J}$, we define the static action prediction algorithm to just append them to the end of the order $\hat{\sigma}$ once they are revealed. Thus, we can still treat $\hat{\sigma}$ as a function from J to $[n]$. We analyse an algorithm which follows a static or adaptive action prediction using the permutation error introduced in (Lindermayr & Megow, 2022). To this end, let $\sigma:J\to[n]$ be the order of a fixed optimal solution for instance J, and $\mathcal{I}(J,\hat{\sigma})=\{(j',j)\in J^2\mid\sigma(j')<\sigma(j)\wedge\hat{\sigma}(j')>\hat{\sigma}(j)\}$ be the set of inversions between the permutations σ and $\hat{\sigma}$. Applying the analysis of (Lindermayr & Megow, 2022) yields the following theorem.

Theorem 6.1. *Given static or adaptive action predictions, there exists an efficient $\mathcal{O}(\min\{1+\eta,\omega\})$-competitive non-clairvoyant algorithm for minimizing the total weighted completion time on a single machine with online precedence constraints, where $\eta=\sum_{(j',j)\in\mathcal{I}(J,\hat{\sigma})}(w_{j'}p_j-w_jp_{j'})$.*

7. Full Input Predictions

We can use full input predictions to compute static action predictions $\hat{\sigma}$. In general, computing $\hat{\sigma}$ requires exponential running time as the problem is NP-hard. For special cases, e.g., chains, there are efficient algorithms (Lawler, 1978).

While following $\hat{\sigma}$ allows us to achieve the guarantee of Theorem 6.1, the error η does not directly depend on the predicted input but on an algorithm which computed actions for that input. Thus, we aim at designing error measures depending directly on the "similarity" between the predicted and actual instance. As describing the similarity between two graphs is a notoriously difficult problem on its own, we leave open whether there is a meaningful error for general topologies. However, we give an error measure for chains. The key idea of this error is to capture additional cost that any algorithm pays due to both, *absent predicted* weights and *unexpected actual* weights. This is in the same spirit as the universal cover error for graph problems in (Bernardini et al., 2022). Assuming that the predicted and actual instance only differ in the weights, our error $\Lambda=\Gamma_u+\Gamma_a$ considers the optimal objective values Γ_u and Γ_a for the problem instances that use $\{(w_j-\hat{w}_j)_+\}_j$ and $\{(\hat{w}_j-w_j)_+\}_j$ as

weights, respectively. Then, Γ_u and Γ_a measure the cost for *unexpected* and *absent* weights. In the appendix, we generalize this idea to also capture other differences of the predicted and actual chains and prove the following theorem.

Theorem 7.1. *Given access to an input prediction, there exists an efficient algorithm for minimizing the total weighted completion time of unit-size jobs on a single machine with online chain precedence constraints with a competitive ratio of at most* $\mathcal{O}(\min\{1+\Lambda, \omega\})$*, where* $\Lambda = \Gamma_u + \Gamma_a$.

8. Final Remarks

We initiated the study of learning-augmented algorithms for scheduling with online precedence constraints by considering a hierarchy of prediction models based on their entropy. For several models of the hierarchy, we were able to show that the predicted information is sufficient to break lower bounds for algorithms without predictions. We hope that our approach leads to more discussions on the identification of the "right" prediction model in learning-augmented algorithm design. As a next research step, we suggest investigating the missing bounds for our prediction models, e.g., an upper bound for average predictions, and exploring error measures for full input predictions based on more fine-grained graph distance metrics.

Acknowledgements

This research has been partially funded by the Swiss National Science Foundation project *Complexity of integer Programming* (207365) and by the German Science Foundation (DFG) under contract 146371743 – TRR 89 Invasive Computing.

References

Allen, F. E. Control flow analysis. *ACM Sigplan Notices*, 5 (7):1–19, 1970.

Ambühl, C. and Mastrolilli, M. Single machine precedence constrained scheduling is a vertex cover problem. *Algorithmica*, 53(4):488–503, 2009.

Anand, K., Ge, R., Kumar, A., and Panigrahi, D. Online algorithms with multiple predictions. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 582–598. PMLR, 2022.

Antoniadis, A., Coester, C., Eliás, M., Polak, A., and Simon, B. Online metric algorithms with untrusted predictions. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 345–355. PMLR, 2020.

Antoniadis, A., Ganje, P. J., and Shahkarami, G. A novel prediction setup for online speed-scaling. In *SWAT*, volume 227 of *LIPIcs*, pp. 9:1–9:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Azar, Y., Leonardi, S., and Touitou, N. Flow time scheduling with uncertain processing time. In *STOC*, pp. 1070–1080. ACM, 2021.

Azar, Y., Leonardi, S., and Touitou, N. Distortion-oblivious algorithms for minimizing flow time. In *SODA*, pp. 252–274. SIAM, 2022a.

Azar, Y., Panigrahi, D., and Touitou, N. Online graph algorithms with predictions. In *SODA*, pp. 35–66. SIAM, 2022b.

Bamas, É., Maggiori, A., and Svensson, O. The primal-dual method for learning augmented algorithms. In *NeurIPS*, 2020.

Bampis, E., Dogeas, K., Kononov, A. V., Lucarelli, G., and Pascual, F. Scheduling with untrusted predictions. In *IJCAI*, pp. 4581–4587. ijcai.org, 2022.

Beaumont, O., Bonichon, N., Eyraud-Dubois, L., and Marchal, L. Minimizing weighted mean completion time for malleable tasks scheduling. In *IPDPS*, pp. 273–284. IEEE Computer Society, 2012.

Bernardini, G., Lindermayr, A., Marchetti-Spaccamela, A., Megow, N., Stougie, L., and Sweering, M. A universal error measure for input predictions applied to online graph problems. In *NeurIPS*, 2022.

Bienkowski, M., Kraska, A., and Liu, H. Traveling repairperson, unrelated machines, and other stories about average completion times. In *ICALP*, volume 198 of *LIPIcs*, pp. 28:1–28:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

Borodin, A. and El-Yaniv, R. *Online computation and competitive analysis*. Cambridge University Press, 1998.

Chakrabarti, S., Phillips, C. A., Schulz, A. S., Shmoys, D. B., Stein, C., and Wein, J. Improved scheduling algorithms for minsum criteria. In *ICALP*, volume 1099 of *Lecture Notes in Computer Science*, pp. 646–657. Springer, 1996.

Cho, W., Henderson, S. G., and Shmoys, D. B. Scheduling with predictions. *CoRR*, abs/2212.10433, 2022.

Correa, J. R. and Schulz, A. S. Single-machine scheduling with precedence constraints. *Math. Oper. Res.*, 30(4): 1005–1021, 2005.

Dinitz, M., Im, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Algorithms with prediction portfolios. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Du, J., Leung, J. Y., and Young, G. H. Scheduling chain-structured tasks to minimize makespan and mean flow time. *Inf. Comput.*, 92(2):219–236, 1991.

Eberle, F., Lindermayr, A., Megow, N., Nölke, L., and Schlöter, J. Robustification of online graph exploration methods. In *AAAI*, pp. 9732–9740. AAAI Press, 2022.

Erlebach, T., de Lima, M. S., Megow, N., and Schlöter, J. Learning-augmented query policies for minimum spanning tree with uncertainty. In *ESA*, volume 244 of *LIPIcs*, pp. 49:1–49:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Garg, N., Gupta, A., Kumar, A., and Singla, S. Non-clairvoyant precedence constrained scheduling. In *ICALP*, volume 132 of *LIPIcs*, pp. 63:1–63:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

Hall, L. A., Schulz, A. S., Shmoys, D. B., and Wein, J. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Math. Oper. Res.*, 22(3):513–544, 1997.

Hayes, B. Cloud computing. *Commun. ACM*, 51(7):9–11, 2008.

Im, S., Kumar, R., Qaem, M. M., and Purohit, M. Non-clairvoyant scheduling with predictions. In *SPAA*, pp. 285–294. ACM, 2021.

Jäger, S. J. *Approximation in deterministic and stochastic machine scheduling*. PhD thesis, Technical University of Berlin, Germany, 2021.

Jin, B. and Ma, W. Online bipartite matching with advice: Tight robustness-consistency tradeoffs for the two-stage model. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Kim, J. and Chwa, K. Non-clairvoyant scheduling for weighted flow time. *Inf. Process. Lett.*, 87(1):31–37, 2003.

Lawler, E. L. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, 2:75–90, 1978.

Lenstra, J. K. and Kan, A. H. G. R. Complexity of scheduling under precedence constraints. *Oper. Res.*, 26(1):22–35, 1978.

Lindermayr, A. and Megow, N. Permutation predictions for non-clairvoyant scheduling. In *SPAA*, pp. 357–368. ACM, 2022.

Lindermayr, A. and Megow, N. Repository of papers on algorithms with predictions, 2023. URL https://algorithms-with-predictions.github.io/.

Lindermayr, A., Megow, N., and Simon, B. Double coverage with machine-learned advice. In *ITCS*, volume 215 of *LIPIcs*, pp. 99:1–99:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Lynn, T., Rosati, P., Lejeune, A., and Emeakaroha, V. C. A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms. In *CloudCom*, pp. 162–169. IEEE Computer Society, 2017.

McNaughton, R. Scheduling with deadlines and loss functions. *Management science*, 6(1):1–12, 1959.

Mitzenmacher, M. and Vassilvitskii, S. Algorithms with predictions. In *Beyond the Worst-Case Analysis of Algorithms*, pp. 646–662. Cambridge University Press, 2020.

Mitzenmacher, M. and Vassilvitskii, S. Algorithms with predictions. *Commun. ACM*, 65(7):33–35, 2022.

Motwani, R., Phillips, S., and Torng, E. Nonclairvoyant scheduling. *Theoretical computer science*, 130(1):17–47, 1994.

Purohit, M., Svitkina, Z., and Kumar, R. Improving online algorithms via ML predictions. In *NeurIPS*, pp. 9684–9693, 2018.

Shahrad, M., Balkind, J., and Wentzlaff, D. Architectural implications of function-as-a-service computing. In *MICRO*, pp. 1063–1075. ACM, 2019.

Timkovsky, V. G. Identical parallel machines vs. unit-time shops and preemptions vs. chains in scheduling complexity. *Eur. J. Oper. Res.*, 149(2):355–376, 2003.

A. Weight Value Predictions

This section deals with the weight value prediction model. We first prove lower bounds. Then we present the algorithmic upper bounds.

A.1. Lower Bounds

Lemma 3.2. *Any algorithm which has only access to adaptive weight predictions has a competitive ratio of at least $\Omega(\sqrt{n})$, even for in-tree precedence constraints.*

Proof. Consider an in-tree instance with unit-size jobs and root r of weight 0. There are $\sqrt{n}$ chains of length 2 with leaf weights 0 and inner weights 1 which are connected to r. Further, there are $n - 2\sqrt{n} - 1$ leaves with weight 0, which are connected to a node v with weight 1, which itself is a child of r. Note that the weight prediction for all potential front jobs except r is always 1. Thus, even the adaptive predictions do not help, and we can assume that the algorithm first processes the children of v, giving a total objective of at least $\Omega((n - 2\sqrt{n} - 1)^2 + (n - 2\sqrt{n} - 1)\sqrt{n}) = \Omega(n\sqrt{n})$, while processing the other leaves first yields a value of at most $\mathcal{O}((2\sqrt{n})^2 + (2\sqrt{n} + n - 2\sqrt{n})) = \mathcal{O}(n)$. □

A.2. A Learning-Augmented Algorithm for Chains

We observe that WRR (cf. Algorithm 1) for chains with static weight predictions crucially relies on having access to exactly precise predicted chain weights. In particular, we show that even if the initial weights of all chains are just overpredicted by the same constant factor $\alpha > 1$, the algorithm is not constant competitive anymore. The intuitive reason for this is that the recomputation of the chain weights in Line 4 can slightly overestimate the remaining weight of a large number of chains, which can slow down important chains that contribute a lot to the objective value by a large factor.

Lemma A.1. *Given wrong weight predictions with $\hat{W}_c = \alpha \cdot W_c$ for all chains $c \in \mathcal{C}$ and a constant factor $\alpha > 1$, Algorithm 1 has a competitive ratio of at least $\Omega(\frac{(\alpha-1)}{\alpha} n^{\frac{1}{3}})$ for minimizing the total weighted completion time of jobs with online out-forest precedence constraints on a single machine.*

Proof. Consider an instance consisting of $k = n^{\frac{1}{3}}$ chains with only unit jobs, i.e., all jobs have processing times of 1. Each chain c has a predicted weight of $\hat{W}_c = \alpha$ and an actual weight of $W_c = 1$. The first chain consists of $\ell = n^{\frac{2}{3}}$ jobs and has its total weight of 1 on the last job in the chain and all previous jobs have a weight of zero. The remaining $(k-1)$ chains each consist of at least $n^{\frac{1}{3}}$ jobs with the total weight of 1 being on the very first job of the respective chain and all other jobs having a weight of zero.

The optimal solution for this instance is to first process the front jobs with weight 1 in an arbitrary order, then process the ℓ jobs of the first chain and finally process all remaining jobs in an arbitrary order. This leads to an objective value of $\textsc{Opt} = \frac{k \cdot (k-1)}{2} + k + \ell \in \mathcal{O}(n^{\frac{2}{3}})$.

Algorithm 1 starts to process each chain with a rate of $\frac{1}{k}$ until the first jobs of each chain is completed after k time units. Afterwards, the first chain is processed with rate $\frac{\alpha}{(\alpha-1)\cdot(k-1)+\alpha} \leq \frac{\alpha}{(\alpha-1)\cdot k}$, as the algorithm thinks that the other chains still have a remaining weight of $\alpha - 1$. The algorithm processes the first chain at this rate until it completes. This leads to an objective value of $\textsc{Alg} \geq k \cdot (k-1) + \frac{(\alpha-1)\cdot k}{\alpha} \cdot \ell \geq k \cdot (k-1) + \frac{\alpha-1}{\alpha} \cdot n \in \Omega(\frac{\alpha-1}{\alpha} \cdot n)$.

Putting the bounds for $\textsc{Opt}$ and $\textsc{Alg}$ together, we get a competitive ratio of $\frac{\textsc{Alg}}{\textsc{Opt}} \in \Omega(\frac{\alpha-1}{\alpha} \cdot n^{1/3})$. □

As a consequence of this lemma, the algorithm cannot be $\mathcal{O}(\eta)$-competitive for the very intuitive multiplicative error measure $\eta = \max_{c \in C} \max\{\frac{W_c}{\hat{W}_c}, \frac{\hat{W}_c}{W_c}\}$. Instead, we use more involved error measures and continue by giving a formal proof of the following theorem.

Theorem 3.8. *For minimizing the total weighted completion time of jobs with online chain precedence constraints on a single machine, there is a non-clairvoyant algorithm with predicted chain weights with a competitive ratio of at most*

$$\mathcal{O}(1) \cdot \min\left\{1 + \frac{\textsc{Opt}(\mathcal{C}_o) + \omega \cdot \textsc{Opt}(\mathcal{C}_u)}{\textsc{Opt}}, \omega\right\}.$$

In order to give the proof, we first formally define the *predicted instance* (including $\mathcal{C}_o$ and $\mathcal{C}_u$).

Algorithm 3 Learning-augmented WRR on Chains

1: Execute Algorithm 1 with rate $\frac{1}{2}$ using the predicted chain weights.
2: Execute the algorithm of Proposition 2.1 with rate $\frac{1}{2}$.

Definition A.2 (predicted instances)**.** The *predicted instance* $\hat{\mathcal{C}}$, the *underpredicted subinstance* $\mathcal{C}_u$, and the *overpredicted subinstance* $\mathcal{C}_o$ are constructed by considering for every $c = [j_1, \dots, j_\ell] \in \mathcal{C}$ the following cases:

(i) if $\hat{W}_c = W_c$, then the chain $\hat{c} = c$ with job weights $\hat{w}_j = w_j$ for all $j \in c$ is added to $\hat{\mathcal{C}}$.

(ii) if $\hat{W}_c < W_c$, then the chain $\hat{c} = [j_1, \dots, j_k]$, where k is the smallest index s.t. $\hat{W}_c \leq \sum_{i=1}^{k} w_i$, with weights $\hat{w}_{j_i} = w_{j_i}$ for all $1 \leq i \leq k-1$ and $\hat{w}_{j_k} = \hat{W}_c - \sum_{i=1}^{k-1} w_i$ is added to $\hat{\mathcal{C}}$. Additionally, a chain $c_u = [\perp, j_{k+1}, \dots, j_\ell]$ with weights $\hat{w}_{j_i} = w_{j_i}$ for all $k+1 \leq i \leq \ell$ and $\hat{w}_\perp = \sum_{i=1}^{k} w_i - \hat{W}_c$ is added to $\mathcal{C}_u$, where the processing requirement of $\perp$ is equal to the total processing requirement of $\hat{c}$.

(iii) if $\hat{W}_c > W_c$, then chain $\hat{c} = [j_1, \dots, j_\ell]$ with weights $\hat{w}_{j_i} = w_{j_i}$ for all $1 \leq i \leq \ell - 1$ and $\hat{w}_{j_\ell} = \hat{W}_c - \sum_{i=1}^{\ell-1} w_i$ is added to $\hat{\mathcal{C}}$. Additionally, a chain $c_o = [\top]$ is added to $\mathcal{C}_o$, where the weight of $\top$ is equal to $\hat{W}_c - \sum_{i=1}^{\ell} w_i$ and its processing requirement is equal to the total processing requirement of $\hat{c}$.

Finally, $\mathcal{C}_p$ is a copy of $\hat{\mathcal{C}}$ where for every overpredicted chain $\hat{c} = [j_1, \dots, j_\ell] \in \hat{\mathcal{C}}$ the weight of its last job j_ℓ is set to w_{j_ℓ}, the weight of the job in the actual instance. This weight is strictly smaller than the weight $\hat{w}_{j_\ell} = \hat{W}_c - \sum_{i=1}^{\ell-1} w_i$ of the job in instance $\hat{\mathcal{C}}$.

Note that for every $\hat{c} \in \hat{\mathcal{C}}$, we have $\sum_{j \in \hat{c}} \hat{w}_j = \hat{W}_c$, i.e., the predicted weights are correct for $\hat{\mathcal{C}}$. In the following, we nevertheless call a chain $\hat{c} \in \hat{\mathcal{C}}$ overpredicted resp. underpredicted if that is true for its corresponding chain in $\mathcal{C}$. Since every job j of $\mathcal{C}_p$ is also part of $\mathcal{C}$ with the same processing requirement and a weight of at most w_j and the chains in $\mathcal{C}_p$ are prefixes of the chains in $\mathcal{C}$, we conclude:

Proposition A.3. $\textsc{Opt}(\mathcal{C}_p) \leq \textsc{Opt}(\mathcal{C})$.

We use the algorithm of Proposition 2.1 in combination with the times-haring of Theorem 2.2 to define Algorithm 3.

Lemma A.4. *Algorithm 3 with predicted chain weights achieves an objective value of at most*

$$\mathcal{O}(1) \cdot \textsc{Opt}(\mathcal{C}_p) + \mathcal{O}(1) \cdot (\textsc{Opt}(\mathcal{C}_o) + \omega \cdot \textsc{Opt}(\mathcal{C}_u)).$$

Proof. We first argue that $\textsc{Opt}(\hat{\mathcal{C}}) \leq \mathcal{O}(1) \cdot \textsc{Opt}(\mathcal{C}_p) + \mathcal{O}(1) \cdot \textsc{Opt}(\mathcal{C}_o)$. To this end, consider the instance $\mathcal{C}_p \cup \mathcal{C}_o$. Every correctly predicted or underpredicted chain in $\hat{\mathcal{C}}$ is contained as an identical copy in $\mathcal{C}_p$. For every overpredicted chain $\hat{c} \in \hat{\mathcal{C}}$ with weight $\hat{W}_{\hat{c}}$ in $\hat{\mathcal{C}}$, all jobs of $\hat{c}$ are contained with a total weight of $W_{\hat{c}}$ in $\mathcal{C}_p$ and the remaining weight of $\hat{W}_{\hat{c}} - W_{\hat{c}}$ is contained in $\mathcal{C}_o$. Additionally, it is ensured by the processing requirement of the jobs in $\mathcal{C}_o$ that their weight can only be gained when processing at least the total processing requirement of $\hat{c}$. This implies that the time to gain weight $\hat{W}_{\hat{c}} - W_{\hat{c}}$ of every overpredicted chain $\hat{c} \in \hat{\mathcal{C}}$ in $\mathcal{C}_p \cup \mathcal{C}_o$ takes as least as long as in $\hat{\mathcal{C}}$, and thus $\textsc{Opt}(\hat{\mathcal{C}}) \leq \textsc{Opt}(\mathcal{C}_p \cup \mathcal{C}_o)$. Finally, it is not hard to see that $\textsc{Opt}(\mathcal{C}_p \cup \mathcal{C}_o) \leq 2 \cdot \textsc{Opt}(\mathcal{C}_p) + 2 \cdot \textsc{Opt}(\mathcal{C}_o)$ as $\textsc{Opt}(\mathcal{C}_p)$ and $\textsc{Opt}(\mathcal{C}_o)$ can be executed in parallel by preemptively sharing the machine, yielding the claimed bound.

We now show that $\textsc{Alg} \leq \mathcal{O}(1) \cdot (\textsc{Opt}(\hat{\mathcal{C}}) + \omega \cdot \textsc{Opt}(\mathcal{C}_u))$ for Algorithm 3, which implies the statement. First consider the execution of Algorithm 1 in the first line. We may assume that the algorithm processes the artificial job added to each overpredicted chain in $\hat{\mathcal{C}}$, as it only increases its objective. Further, Algorithm 1 stops processing an underpredicted chain $c \in \mathcal{C}$ when a total weight of $\hat{W}_c$ has been completed on c and only finishes them at the very end of the schedule. This concludes that the total objective of Algorithm 1 *without* the weighted completion times of the jobs processed in Line 7 is at most $\mathcal{O}(1) \cdot \textsc{Opt}(\hat{\mathcal{C}})$. But, due to Proposition 2.1 and line two of the algorithm, we conclude that Algorithm 3 always processes such chains that are only completed in Line 7 of Algorithm 1 with a rate of at least $\frac{1}{2\omega}$ and, thus, delays the completion the jobs in these chains by a factor of at most 2ω compared to an optimal solution. By observing that the total weight of jobs processed in Line 7 by Algorithm 1 is exactly equal to the total weight of chains in $\mathcal{C}_u$, and the fact that the jobs in a chain $c \in \mathcal{C}_u$ can only be processed after time equal to the total processing requirement of the corresponding chain in $\hat{\mathcal{C}}$, we conclude the stated bound. □

A.3. Adaptive weight predictions for out-forests

Algorithm 4 Weighted Round Robin on out-forests

Require: Out-forest T and adaptive weight predictions.
1: $t \leftarrow 0$
2: **while** $F_t \neq \emptyset$ **do**
3: Process every $v \in F_t$ with rate $R_j^t = \frac{\hat{W}_j}{\sum_{i \in F_t} \hat{W}_i}$
4: $t \leftarrow t + 1$
5: **end while**

Theorem 3.9. *For minimizing the total weighted completion time on a single machine with online out-forest precedence constraint and adaptive weight predictions, there is a non-clairvoyant algorithm with a competitive ratio of at most*

$$\mathcal{O}(1) \cdot \min \left\{ \max_{v \in J} \frac{\hat{W}_v}{w(S(v))} \cdot \max_{v \in J} \frac{w(S(v))}{\hat{W}_v}, \omega \right\}.$$

Proof. To prove the theorem, we show that Algorithm 4 is $\mathcal{O}(\eta)$-competitive for

$$\eta = \max_{v \in J} \frac{\hat{W}_v}{W_v} \cdot \max_{v \in J} \frac{W_v}{\hat{W}_v}.$$

Then, Proposition 2.1 and Theorem 2.2 imply the theorem.

By Theorem 3.3, it suffices to show that $\frac{w(S(j))}{W(t)} \leq \eta \cdot R_j^t$ holds for any point in time t during the execution of Algorithm 4 and any $j \in F_t$, where R_j^t is the rate with which the algorithm processes j at point in time t. Consider a fixed point in time t and an arbitrary $j \in F_t$. Then, the algorithm processes j with rate $R_j^t = \frac{\hat{W}_j}{\sum_{i \in F_t} \hat{W}_i}$ by definition. We can conclude

$$\begin{aligned}
\frac{w(S(j))}{W(t)} = \frac{w(S(j))}{\sum_{i \in F_t} w(S(i))} &= \frac{\hat{W}_j \cdot \frac{w(S(j))}{\hat{W}_j}}{\sum_{i \in F_t} \hat{W}_i \cdot \frac{w(S(i))}{\hat{W}_i}} \\
&\leq \frac{\left(\max_{v \in J} \frac{w(S(v))}{\hat{W}_v}\right) \cdot \hat{W}_j}{\left(\min_{v \in J} \frac{w(S(v))}{\hat{W}_v}\right) \cdot \sum_{i \in F_t} \hat{W}_i} \\
&= \max_{v \in J} \frac{\hat{W}_v}{w(S(v))} \cdot \max_{v \in J} \frac{w(S(v))}{\hat{W}_v} \cdot \frac{\hat{W}_j}{\sum_{i \in F_t} \hat{W}_i} \\
&= \eta \cdot R_j^t.
\end{aligned}$$

□

B. Static Weight Order

Lemma 4.3. *The variant of Algorithm 2 that computes the rates using $\hat{\preceq}_0$ instead of $\hat{\preceq}_t$ is at least $\Omega(\omega \cdot H_\omega)$-competitive, even if $\hat{\preceq}_0$ equals $\preceq_0$.*

Proof. Consider an instance with ω chains, each with a total weight of one. Then, $\preceq_0$ is just an arbitrary order of the chains. Recall that the algorithm starts processing the chains c with rate $(H_\omega \cdot i_c)^{-1}$, where i_c is the position of c in the order. We define the first $\omega - 1$ chains to have their total weight of one at the very first job and afterwards only jobs of weight zero. Chain ω, the slowest chain, has its total weight on the last job. We define the chains c to contain a total of $d \cdot H_\omega \cdot i_c$ jobs with unit processing times, for some common integer d. This means that the algorithm finishes all chains at the same time. The optimal solution value for this instance is $\omega \cdot (\omega + 1) + \omega - 1 + d \cdot H_\omega \cdot \omega$, where $\omega \cdot (\omega + 1)$ is the optimal sum of completion times for the first $\omega - 1$ chains, $d \cdot H_\omega \cdot \omega$ is the cost for processing the last chain, and $\omega - 1$ is the cost for

delaying the last chains by the $\omega - 1$ time units needed to process the first jobs of the first $\omega - 1$ chains. The solution value of the algorithm is at least $d \cdot H_\omega^2 \cdot \omega^2$ as this is the cost for just processing the last chain. Thus, for large d, the competitive ratio tends to $H_\omega \cdot \omega$. □

Theorem 4.4. *For any $\epsilon > 0$, Algorithm 2 has a competitive ratio of at most $\mathcal{O}(H_\omega^2 \sqrt{P} \cdot \max\{1+\epsilon, \mathcal{L}(\epsilon)\})$ when computing rates with $\hat{\preceq}_0$ instead of $\hat{\preceq}_t$ at any time t. For unit jobs, it is $\mathcal{O}(H_\omega^2 \sqrt{n} \cdot \max\{1+\epsilon, \mathcal{L}(\epsilon)\})$-competitive.*

Proof. Recall that P denotes the sum over all job processing times in the instance. For a subset of chains S, let $\text{OPT}(S)$ denote the optimal objective value for the subinstance induced by S. For a single chain c, $\text{OPT}(c)$ is just the cost for processing chain c with rate 1 on a single machine. Clearly, $\text{OPT}(S) \geq \sum_{c \in S} \text{OPT}(c)$. Let $\text{ALG}(S)$ denote the sum of weighted completion times of the jobs that belong to chains in S in the schedule computed by the algorithm.

In the first part of the proof, we assume that all chains c_i have a total processing time of at most $\sqrt{P}$. This only decreases the objective value of OPT. For ALG, we will analyze the additional cost caused by longer chains afterwards. In a sense, we assume that ALG, for each chain c, has to pay all weight that appears after $\sqrt{P}$ processing times units of the chain two times: Once artificially after exactly $\sqrt{P}$ time units of the chain have been processed and once at the point during the processing where the weight actually appears. This assumption clearly only increases ALG. In the first part of the proof, we analyze only the artificial cost for such weights and ignore the actual cost. In the context of our algorithm this is equivalent to assuming the chains have total processing times of at most $\sqrt{P}$. In the second part of the proof we will analyze the actual cost for the jobs that appear after $\sqrt{P}$ time units in their chain.

First Part. Assume $c_1 \hat{\preceq}_0 c_2 \hat{\preceq}_0 \ldots \hat{\preceq}_0 c_\omega$. Therefore, the algorithm processes chain c_i with rate $(H_\omega \cdot i)^{-1}$. This directly implies $\text{ALG}(c_i) = H_\omega \cdot i \cdot \text{OPT}(c_i)$ and, thus,

$$\text{ALG} = \sum_{i=1}^{\omega} H_\omega \cdot i \cdot \text{OPT}(c_i).$$

Let $\mathcal{C}_k = \{c_1, \ldots, c_k\}$ for every $k \in [\omega]$. We first analyze $\text{ALG}(\mathcal{C}_{3 \cdot \mathcal{L}(\epsilon)})$. For the chains in $\mathcal{C}_{3 \cdot \mathcal{L}(\epsilon)}$, we get

$$\begin{aligned}
\text{ALG}(\mathcal{C}_{3 \cdot \mathcal{L}(\epsilon)}) &= \sum_{i=1}^{3 \cdot \mathcal{L}(\epsilon)} H_\omega \cdot i \cdot \text{OPT}(c_i) \\
&\leq H_\omega \cdot 3 \cdot \mathcal{L}(\epsilon) \cdot \sum_{i=1}^{3 \cdot \mathcal{L}(\epsilon)} \text{OPT}(c_i) \\
&\leq H_\omega \cdot 3 \cdot \mathcal{L}(\epsilon) \cdot \text{OPT},
\end{aligned}$$

meaning that, for $\mathcal{C}_{3 \cdot \mathcal{L}(\epsilon)}$, we achieve the desired competitive ratio.

Next, consider the chains in $\mathcal{C} \setminus \mathcal{C}_{3 \cdot \mathcal{L}(\epsilon)}$, i.e., the chains c_i with $i > 3 \cdot \mathcal{L}(\epsilon)$. To analyze the cost for these chains C_i, we continue by lower bounding $\text{OPT}(c_i)$. To that end, consider $\text{OPT}(\mathcal{C}_i)$. The definition of $\mathcal{L}(\epsilon)$ implies that there are at most $\mathcal{L}(\epsilon)$ chains $c_j \in \mathcal{C}_i$ with $W_{c_i} \geq (1+\epsilon) W_{c_j}$. For all other chains c_j in $\mathcal{C}_i$, we have $\frac{W_{c_i}}{(1+\epsilon)} < W_{c_j}$. Thus, there are $i - \mathcal{L}(\epsilon)$ chains in $\mathcal{C}_i$ with a weight of at least $\frac{W_{c_i}}{(1+\epsilon)}$. Since we consider chains with $i > 3 \cdot \mathcal{L}(\epsilon)$, it holds $i - \mathcal{L}(\epsilon) \geq 1$. We can lower bound $\text{OPT}(\mathcal{C}_i)$ by assuming that all such chains consist only of a single job with weight $\frac{W_{c_i}}{(1+\epsilon)}$ and ignoring the up-to $\mathcal{L}(\epsilon)$ other chains. These assumptions only decrease $\text{OPT}(\mathcal{C}_i)$. Since in this relaxation all jobs have an equal

weight and length, an optimal solution for it processes the jobs in an arbitrary order, giving

$$\begin{aligned}
\textsc{Opt}(\mathcal{C}_i) &\geq \sum_{j=1}^{i-\mathcal{L}(\epsilon)} j \cdot \frac{W_{c_i}}{1+\epsilon} \\
&= \frac{(i-\mathcal{L}(\epsilon)+1)\cdot(i-\mathcal{L}(\epsilon))\cdot W_{c_i}}{2\cdot(1+\epsilon)} \\
&= \frac{((i+1)\cdot i+\mathcal{L}(\epsilon)^2-2\cdot i\cdot\mathcal{L}(\epsilon)-\mathcal{L}(\epsilon))\cdot W_{c_i}}{2\cdot(1+\epsilon)} \\
&\geq \frac{(i+1)\cdot i-3\cdot i\cdot\mathcal{L}(\epsilon)}{2\cdot(1+\epsilon)}\cdot W_{c_i}.
\end{aligned}$$

Since we still assume that each chain has a total processing time of at most $\sqrt{P}$, we can observe $\textsc{Opt}(c_i) \leq \sqrt{P}\cdot W_{c_i}$. This yields:

$$\frac{2\cdot(1+\epsilon)}{(i+1)\cdot i-3\cdot i\cdot\mathcal{L}(\epsilon)}\cdot\sqrt{P}\cdot\textsc{Opt}(\mathcal{C}_i) \geq \textsc{Opt}(c_i).$$

We can therefore conclude

$$\begin{aligned}
\textsc{Alg}(\mathcal{C}\setminus\mathcal{C}_{3\cdot\mathcal{L}(\epsilon)}) &= \sum_{i=3\cdot\mathcal{L}(\epsilon)+1}^{\omega} H_\omega\cdot i\cdot\textsc{Opt}(c_i) \\
&\leq \sum_{i=3\cdot\mathcal{L}(\epsilon)+1}^{\omega} H_\omega\cdot\frac{2\cdot(1+\epsilon)}{(i+1)-3\cdot\mathcal{L}(\epsilon)}\cdot\sqrt{P}\cdot\textsc{Opt}(\mathcal{C}_i) \\
&\leq 2\cdot(1+\epsilon)\cdot H_\omega\cdot\sqrt{P}\cdot\textsc{Opt}\sum_{i=3\cdot\mathcal{L}(\epsilon)+1}^{\omega}\frac{1}{(i+1)-3\cdot\mathcal{L}(\epsilon)} \\
&\leq 2\cdot(1+\epsilon)\cdot H_\omega\cdot H_{\omega-3\mathcal{L}(\epsilon)+1}\cdot\sqrt{P}\cdot\textsc{Opt} \\
&\leq 2\cdot(1+\epsilon)\cdot H_\omega^2\cdot\sqrt{P}\cdot\textsc{Opt}.
\end{aligned}$$

We can finish the first part of the proof by combining the bounds for $\textsc{Alg}(\mathcal{C}\setminus\mathcal{C}_{3\cdot\mathcal{L}(\epsilon)})$ and $\textsc{Alg}(\mathcal{C}_{3\cdot\mathcal{L}(\epsilon)})$:

$$\begin{aligned}
\textsc{Alg}(\mathcal{C}) &= \textsc{Alg}(\mathcal{C}\setminus\mathcal{C}_{3\cdot\mathcal{L}(\epsilon)})+\textsc{Alg}(\mathcal{C}_{3\cdot\mathcal{L}(\epsilon)}) \\
&\leq 5\cdot H_\omega^2\cdot\sqrt{P}\cdot\max\{1+\epsilon,\mathcal{L}(\epsilon)\}\cdot\textsc{Opt}.
\end{aligned}$$

Second Part. It remains to analyze the additional cost incurred by chains with a total processing time of more than $\sqrt{P}$. To that end, consider the set J_L of jobs that, in any schedule, cannot be started before $\sqrt{P}$ time units have past. For a job $j \in J_L$, the predecessors of j in the chain of j must have a total processing time of at least $\sqrt{P}$.

Let $\textsc{Alg}(J_L)$ and $\textsc{Opt}(J_L)$ denote the weighted completion times of the jobs in J_L in the optimal solution and the schedule computed by $\textsc{Alg}$, respectively. Then,

$$\frac{\textsc{Alg}(J_L)}{\textsc{Opt}(J_L)} \leq \frac{\sum_{j\in J_L} P\cdot w_j}{\sum_{j\in J_L}\sqrt{P}\cdot w_j} = \sqrt{P}.$$

Thus, the additional cost of the jobs in J_L asymptotically does not worsen the competitive ratio. □

C. Average Predictions

Lemma 5.1. *Any algorithm which has only access to correct adaptive average predictions is at least $\Omega(\sqrt{n})$-competitive even for chain precedence constraints with unit jobs.*

Proof. Consider an instance composed of $\sqrt{n} \in \mathbb{N}$ chains of unit jobs, where the first two jobs of the first chain have weights 1 resp. $n - \sqrt{n}$, followed by $n - \sqrt{n} - 1$ zero weight jobs. The other $\sqrt{n} - 1$ chains are single jobs with weight 1. For an algorithm, all chains look identical since the first jobs have weight 1 and the average of every chain is equal to 1. Therefore, an adversary can ensure that the algorithm processes the first chain last, giving an objective value of $\sum_{i=1}^{\sqrt{n}} i + (\sqrt{n}+1)(n-\sqrt{n}) = \Omega(n\sqrt{n})$, while a solution which schedules the heavy weight job initially achieves an objective value of at most $1 + 2(n-\sqrt{n}) + \sum_{i=1}^{\sqrt{n}-1}(3+i) = \mathcal{O}(n)$. The adaptivity of the predictions does not help for this lower bound as the algorithm would only receive meaningful updates once it finishes the first job of the first chain, which is too late. □

D. Input Predictions

This section is devoted to the proof of Theorem 7.1.

Let $\hat{\mathcal{C}}$ denote the set of predicted chains, and let $\hat{w}_j$ denote the predicted weight of a job j of the instance. All processing requirements are equal to 1, and the algorithm is aware of this. We assume w.l.o.g. that $|\hat{\mathcal{C}}| = |\mathcal{C}|$ by adding chains with zero (predicted) weight, and that predicted and actual chains have the same identities. That is, there exists exactly one predicted chain $c_i \in \hat{\mathcal{C}}$ for each actual chain $\hat{c}_i \in \mathcal{C}$, which an algorithm can match to each other.

Our error measure further requires that there exists for every actual job a predicted counterpart, and vice versa. For a chain c let $|c|$ denote the number of jobs of chain c. We define augmentations of $\mathcal{C}$ and $\hat{\mathcal{C}}$ as follows. Let $\mathcal{C}'$ be composed of all jobs of $\mathcal{C}$, and additionally, for every paired chains $c_i \in \mathcal{C}$ and $\hat{c}_i \in \hat{\mathcal{C}}$:

- if $|\hat{c}_i| > |c_i|$, we add $|\hat{c}_i| - |c_i|$ jobs J_u with weight 0 at the end of c_i in $\mathcal{C}'$. Note that $\textsc{Opt}(\mathcal{C}) = \textsc{Opt}(\mathcal{C}')$.
- if $|c_i| > |\hat{c}_i|$, we add $|c_i| - |\hat{c}_i|$ jobs J_a with predicted weight 0 at the end of $\hat{c}_i$ in $\hat{\mathcal{C}}'$.

Note that this construction ensures $\textsc{Opt}(\hat{\mathcal{C}}) = \textsc{Opt}(\hat{\mathcal{C}}')$.

For the sake of analysis, assume w.l.o.g. that both $\mathcal{C}'$ and $\hat{\mathcal{C}}'$ share the same set of jobs J'. Let $n' = |J'|$. We define $\textsc{Opt}(\{w'_j\}_j)$ as the objective of an optimal solution for J' where a job j has weight w'_j. We further define

$$\textsc{Opt}(\{w'_j\}_j, \{w_j\}_j) = \max\left\{\sum_{j \in J'} w'_j C^*_j \mid \{C^*_j\}_j \text{ is an optimal schedule for } \{w_j\}_j\right\}.$$

Given two fixed augmented instances $\mathcal{C}'$ and $\hat{\mathcal{C}}'$, we define the input prediction error $\Lambda = \Gamma_u + \Gamma_a$:

- a job $j \in J'$ has *unexpected actual* weight if $w_j > \hat{w}_j$. The prediction error due to all unexpected weights can be expressed as $\Gamma_u = \textsc{Opt}(\{\max\{\hat{w}_j, w_j\} - w_j\}_j, \{w_j\}_j)$.
- a job $j \in J'$ has *absent predicted* weight if $\hat{w}_j > w_j$. The prediction error due to all absent weights can be expressed as $\Gamma_a = \textsc{Opt}(\{\max\{w_j, \hat{w}_j\} - \hat{w}_j\}_j, \{\hat{w}_j\}_j)$.

Theorem 7.1. *Given access to an input prediction, there exists an efficient algorithm for minimizing the total weighted completion time of unit-size jobs on a single machine with online chain precedence constraints with a competitive ratio of at most $\mathcal{O}(\min\{1+\Lambda, \omega\})$, where $\Lambda = \Gamma_u + \Gamma_a$.*

Recall that $\mathcal{O}(\omega)$-robustness can be achieved via Proposition 2.1 and Theorem 2.2. Thus, we can prove the theorem by deriving an $\mathcal{O}(1+\Lambda)$-competitive algorithm.

Proof. We analyze the following algorithm:

1) Efficiently compute an optimal solution based on $\hat{\mathcal{C}}$ (Lawler, 1978). This yields an non-preemptive schedule for the predicted instance, i.e., an order of the jobs.
2) Follow the computed solution. The following situations might occur:

a) a chain finishes earlier than expected. In this case, discard the remaining predicted jobs of this chain in the precomputed schedule.
b) a chain continues although there are no more jobs in this chain in the algorithms schedule. In this case, schedule the remaining jobs in an arbitrary order at the end of the precomputed schedule.

Let ALG denote the objective value of this algorithm. We first observe that $\text{ALG} \leq \text{OPT}(\{w_j\}_j, \{\hat{w}_j\}_j)$. To see this, recall that the algorithm first follows an optimal schedule for jobs $J' \setminus J_u$ and then schedules all unexpected jobs J_u at the end due to case b). Since jobs J_u have predicted weight 0 in $\hat{\mathcal{C}}'$, we can assume that an optimal solution for $\hat{\mathcal{C}}'$ first schedules jobs $J' \setminus J_u$ as our algorithm with the same objective value and makespan as our algorithm, and then schedules jobs J_u in any order. Since $\text{OPT}(\{w_j\}_j, \{\hat{w}_j\}_j)$ is an upper bound on the actual objective for any such order, the inequality follows. It further holds that

$$\begin{aligned}
\text{OPT}(\{w_j\}_j, \{\hat{w}_j\}_j) &\leq \text{OPT}(\{\max\{w_j, \hat{w}_j\}\}_j, \{\hat{w}_j\}_j) \\
&= \text{OPT}(\{\hat{w}_j\}_j) + \text{OPT}(\{\max\{w_j, \hat{w}_j\} - \hat{w}_j\}_j, \{\hat{w}_j\}_j) \\
&\leq \text{OPT}(\{\hat{w}_j\}_j, \{w_j\}_j) + \text{OPT}(\{\max\{w_j, \hat{w}_j\} - \hat{w}_j\}_j, \{\hat{w}_j\}_j) \\
&\leq \text{OPT}(\{\max\{\hat{w}_j, w_j\}\}_j, \{w_j\}_j) + \text{OPT}(\{\max\{w_j, \hat{w}_j\} - \hat{w}_j\}_j, \{\hat{w}_j\}_j) \\
&\leq \text{OPT}(\{w_j\}_j) + \text{OPT}(\{\max\{\hat{w}_j, w_j\} - w_j\}_j, \{w_j\}_j) + \text{OPT}(\{\max\{w_j, \hat{w}_j\} - \hat{w}_j\}_j, \{\hat{w}_j\}_j) \\
&= \text{OPT}(\{w_j\}_j) + \Lambda.
\end{aligned}$$

We finally observe that $\text{OPT}(\{w_j\}_j) = \text{OPT}(\mathcal{C})$, as jobs J_a do not influence the objective value of an optimal solution. □

E. Extension to Parallel Machines

We generalize in this section our main results for weight value and adaptive weight order predictions (Theorems 3.4, 3.5 and 4.1) to parallel identical machines. Compared to the single machine, the competitive ratios of these results only increase additively by 2. Formally, we consider the scheduling problem where we are given m parallel identical machines and an algorithm can again assign at any time t to every front job $j \in F_t$ a rate $R_j^t \in [0, 1]$. Opposed to the single machine model, the rates now have to satisfy $\sum_j R_j^t \leq m$ at any time t. Using McNaughton's wrap-around-rule (McNaughton, 1959), we can transform these rates to an actual schedule that processes any job at any point in time on at most one machine.

We prove the following generic result, which is a generalization of Theorem 3.3.

Theorem E.1. *If an algorithm for scheduling weighted jobs with online out-forest precedence constraints on parallel identical machines satisfies at any time t and for every $j \in F_j$ that*

$$R_j^t < 1 \Longrightarrow m \cdot \frac{w(S(j))}{W(t)} \leq \rho \cdot R_j^t,$$

where R_j^t denotes the processing rate of j at time t, it is at most $2 + 4\rho$-competitive for minimizing the total weighted completion time.

We first apply this theorem to derive competitive ratios of algorithms with weight value and adaptive weight order predictions for parallel identical machines, and finally prove Theorem E.1.

E.1. Static Weight Values for Chains

Recall Algorithm 1 for a single machine, which assigns rates proportional to a chains remaining total weight. These rates fulfill by definition that they can be scheduled on a single machine, i.e., sum up to at most 1. On m parallel identical machines, we intuitively have m times as much processing power compared to a single machine. A straightforward approach to exploit this observation would be to scale the single machine rates by a factor of m, i.e., process chain c with rate $m \cdot \frac{W_c(t)}{\sum_c W_c(t)}$. These rates clearly some up to at most m. However, a single chain might receive a rate more than 1, which is disallowed. To circumvent this issue, we cap such rates at 1. The final algorithm (Algorithm 5) therefore first identifies chains with large remaining weight and assigns them rate 1, and then splits the remaining processing power among the remaining chains. For non-clairvoyant scheduling of weighted jobs on parallel identical machines without precedence constraints, this approach is known to be 2-competitive (Beaumont et al., 2012).

Algorithm 5 WDEQ on Chains

Require: Set of chains $\mathcal{C}$, initial total weight W_c of every chain $c \in \mathcal{C}$.

$t \leftarrow 0$
$W_c(t) \leftarrow W_c$ for every chain c.
while $U_t \neq \emptyset$ **do**
  $C \leftarrow \{c_1, \ldots, c_k\}$
  $m' \leftarrow m$
  while $\exists c \in C$ such that $m' \cdot \frac{W_c(t)}{\sum_c W_c(t)} \geq 1$ **do**
    Process the first job $j \in I_t$ in chain c with rate $R_j^t = 1$
    $C \leftarrow C \setminus \{c\}, m' \leftarrow m' - 1$
  end while
  For every chain $c \in C$, process the first job j by $R_j^t = m' \cdot \frac{W_c(t)}{\sum_c W_c(t)}$
  $t \leftarrow t + 1$
  If some job j in chain c finished, set $W_c(t) \leftarrow W_c(t) - w_j$
end while
Schedule remaining jobs arbitrarily.

Algorithm 6 Adaptive weight order algorithm for parallel identical machines

Require: Time t, front jobs F_t, adaptive order $\hat{\preceq}_t$

Process every $j \in F_t$ with rate $R_j^t = \min\left\{1, \frac{m}{H_{|F_t|} \cdot i_j}\right\}$, where i_j is the position of j in $\hat{\preceq}_t$.

Using the same observation as for Algorithm 1, we conclude that every job j's rate, when being less than 1, is at most $m \cdot \frac{w(S(j))}{W(t)}$. Then, Theorem E.1 implies immediately the following corollary.

Corollary E.2. *Algorithm 5 is a non-clairvoyant algorithm for the problem of minimizing the total weighted completion time on m parallel identical machines with online chain precedence constraints with a competitive ratio of at most 6, when given access to accurate static weight predictions.*

E.2. Adaptive Weight Values for Out-Forests

In a similar fashion, we can extend the algorithm for adaptive weight values on out-forests to the parallel machine environment, and thereby generalize Theorem 3.5. To this end, we consider the algorithm which processes every front job $j \in F_t$ with rate $R_j^t = \min\left\{1, m\frac{\hat{W}_j}{\sum_{j' \in F_t} \hat{W}_{j'}}\right\}$. This immediately implies that if $R_j^t < 1$, then $R_j^t = m\frac{\hat{W}_j}{\sum_{j' \in F_t} \hat{W}_{j'}} = m\frac{w(S(j))}{W(t)}$ for correct predictions. We derive the following corollary via Theorem E.1 with $\rho = 1$.

Corollary E.3. *Given correct weight predictions, there exists a non-clairvoyant 6-competitive algorithm for minimizing the total weighted completion time on m parallel identical machines with online out-forest precedence constraints .*

E.3. Adaptive Weight Order for Out-Forests

For adaptive weight order predictions, we essentially use the exact same trick from the previous section and derive Algorithm 6.

Observe that using the proof of Theorem 4.1, we conclude at any time t and for every $j \in F_t$ that if $R_j^t < 1$, then

$$m \cdot \frac{w(S(j))}{W(t)} \leq m \cdot H_\omega \cdot \max\{1 + \epsilon, \mathcal{L}(\epsilon)\} \cdot \frac{1}{H_{|F_t|} \cdot i_j} = H_\omega \cdot \max\{1 + \epsilon, \mathcal{L}(\epsilon)\} \cdot R_j^t.$$

Using Theorem E.1, we conclude our result for adaptive weight order predictions.

Corollary E.4. *For any $\epsilon > 0$, Algorithm 6 has a competitive ratio of at most $2 + 4H_\omega \cdot \max\{1 + \epsilon, \mathcal{L}(\epsilon)\}$ for minimizing the total weighted completion time on m parallel identical machines with online out-forest precedence constraints.*

E.4. Proof of Theorem E.1

The proof is by dual-fitting and is inspired by (Garg et al., 2019). Fix an instance, an algorithm which fulfills the stated property, and this algorithm's schedule. We first introduce a linear programming relaxation (Garg et al., 2019) for our problem on parallel identical machines which are running at lower speed $\frac{1}{\alpha}$, for some $\alpha \geq 1$. By denoting the value of an optimal solution for the problem with speed $\frac{1}{\alpha}$, we conclude $\text{OPT}_\alpha \leq \alpha \cdot \text{OPT}$.

$$\begin{aligned} \min \quad & \sum_{j,t} w_j \cdot t \cdot \frac{x_{j,t}}{p_j} && && \text{(P-LP}_\alpha\text{)} \\ \text{s.t.} \quad & \sum_t \frac{x_{j,t}}{p_j} \geq 1 && \forall j \\ & \sum_j \alpha \cdot x_{j,t} \leq m && \forall t \\ & \sum_{s \leq t} \frac{x_{j,s}}{p_j} \geq \sum_{s \leq t} \frac{x_{j',s}}{p_{j'}} && \forall t, (j,j') \in E \\ & x_{jt} \geq 0 && \forall j, t \end{aligned}$$

The dual of (P-LP$_\alpha$) can be written as follows.

$$\begin{aligned} \max \quad & \sum_j a_j - m \sum_t b_t && && \text{(P-DLP}_\alpha\text{)} \\ \text{s.t.} \quad & a_j - w_j \cdot t + \sum_{s \geq t} \left(\sum_{(j,j') \in E} c_{s,j \to j'} - \sum_{(j',j) \in E} c_{s,j' \to j} \right) \leq \alpha \cdot b_t \cdot p_j && \forall j, t && (4) \\ & a_j, b_t, c_{t,j \to j'} \geq 0 && \forall j, t, (j,j') \in E \end{aligned}$$

On a single machine, we use the dual linear program to compare the algorithm' objective to the optimal objective at any time. Unfortunately, showing feasibility of the dual variables which encode the algorithm's schedule crucially require to satisfy $m \frac{W_{c_j}(t)}{W(t)} \leq \rho \cdot R_j^t$ for some $\rho \geq 1$ at *any* time and for *every* job j. However, if there is a heavy weight chain, we can only assign it a rate at of at most 1. Thus, in general we cannot satisfy this inequality for any constant ρ. To resolve this issue, we instead use a second lower bound for the optimal objective value, and bound the total objective of the algorithm incurred by rates equal to 1 against it. This intuitively explains the slightly larger competitive ratio compared to the single machine setting.

We say that a job $j \in U_t$ *active* at time t if there exists $j' \in F_t$ such that $j \in S(j')$ and $R_{j'}^t < 1$, otherwise it is *inactive*. We remark that j being active at time t implies $t \leq C_j$. For every time t we denote the set of active jobs by A_t. Note that $A_t \subseteq U_t$. Intuitively, we will use the dual linear program as lower bound for all jobs in A_t, and the following lower bound every time a job is inactive. For every job j, let $chain_j$ be the total processing time of a chain's prefix in G until job j (including j). Note that $\text{OPT} \geq \sum_j w_j \cdot chain_j$.

Let $\kappa > 1$ be a constant which we fix later. We define a dual assignment of (P-DLP$_\alpha$) based on the algorithm's schedule:

- $\bar{a}_j = \sum_{s \geq 0} \bar{a}_{j,s}$, where $\bar{a}_{j,s} \begin{cases} w_j \text{ if } j \text{ is active at time } s \\ 0 \text{ otherwise} \end{cases}$ for every job j,
- $\bar{b}_t = \frac{1}{m \cdot \kappa} \cdot W(t)$ for every time t, and
- $\bar{c}_{t,j' \to j} = \begin{cases} 0 & \text{if } j \notin A_t \text{ or } j' \notin A_t \\ W(S(j)) \end{cases}$ for every time t and edge $(j', j) \in E$.

Lemma E.5. $\sum_j \bar{a}_j - m \sum_t \bar{b}_t + \sum_j w_j \cdot chain_j \geq (1 - \frac{1}{\kappa}) \cdot \text{ALG}$.

Proof. Since the weight w_j of a job j is contained in $W(t)$ if $t \leq C_j$, we conclude $\sum_t \bar{b}_t = \frac{1}{m \cdot \kappa} \text{ALG}$. Now consider a job j and a time $t \leq C_j$. If j is active at time t, by definition $\bar{a}_{j,s} = w_j$. Otherwise, j or some predecessor of j is being processed with rate 1. Hence, we conclude that the number of times before C_j when j is inactive is at most $chain_j$. Therefore $w_j C_j \leq \bar{a}_j + w_j \cdot chain_j$ and $\text{ALG} \leq \sum_j \bar{a}_j + \sum_j w_j \cdot chain_j$. □

Lemma E.6. *Assigning $a_j = \bar{a}_j$, $b_t = \bar{b}_t$ and $c_{t,j \to j'} = \bar{c}_{t,j \to j'}$ gives a feasible solution for* (P-DLP$_\alpha$) *if there exists a $\rho \geq 1$ such that $\alpha \geq \kappa \cdot \rho$ and the algorithm's rates satisfy at any time t for every $j \in F_t$*

$$R_j^t < 1 \Longrightarrow m \cdot \frac{W_{c_j}(t)}{W(t)} \leq \rho \cdot R_j^t. \tag{5}$$

Proof. Since our defined variables are non-negative by definition, it suffices to show that this assignment satisfies (4). Fix a job j and a time $t \geq 0$. Let c be j's chain. By observing that $\bar{a}_j - t \cdot w_j \leq \sum_{s \geq t} \bar{a}_{j,s}$, it suffices to verify

$$\sum_{s \geq t} \left(\bar{a}_{j,s} + \sum_{(j,j') \in E} \bar{c}_{s,j \to j'} - \sum_{(j',j) \in E} \bar{c}_{s,j' \to j} \right) \leq \alpha \cdot \bar{b}_t \cdot p_j. \tag{6}$$

To this end, we consider the terms of the left side for all times $s \geq t$ separately. For any s with $s > C_j$, the left side of (6) is zero, because $\bar{a}_{j,s} = 0$ and $j \notin A_s$. Also note that at any time $s \in [0, C_j]$ when j is inactive, the left side of (6) is again zero. Therefore, we restrict our attention in the following to times where j is active.

Let t_j^* be the first point in time after t when j is available, and let $s \in [0, t_j^*)$ such that j is active at time s. Observe that $j \in A_s$ and since each vertex in an out-forest has at most one direct predecessor, there must be a unique job $j_1 \in A_s$ with $(j_1, j) \in E$. Thus $\bar{c}_{s,j_1 \to j} = w(S(j)) = \sum_{(j',j) \in E} \bar{c}_{s,j' \to j}$ and $\bar{c}_{s,j \to j'} = w(S(j'))$ for every j' with $(j, j') \in E$, because j' must also be unfinished and active, hence $j' \in A_s$. Using the fact that the sets $S(j')$ are pairwise disjoint for all direct successors j' of j, we conclude $\sum_{(j,j') \in E} \bar{c}_{s,j' \to j} = w(S(j)) - w_j$ and, thus,

$$\bar{a}_{j,s} + \sum_{(j,j') \in E} \bar{c}_{s,j \to j'} - \sum_{(j',j) \in E} \bar{c}_{s,j' \to j} \leq w_j - w_j = 0.$$

By defining $T_j = \{t_j^* \leq s \leq C_j \mid j \text{ active at time } s\}$, we conclude that proving (6) reduces to proving

$$\sum_{s \in T_j} \left(w_j + \sum_{(j,j') \in E} \bar{c}_{s,j \to j'} - \sum_{(j',j) \in E} \bar{c}_{s,j' \to j} \right) \leq \alpha \cdot \bar{b}_t \cdot p_j. \tag{7}$$

Now, let $s \in T_j$. There cannot be an unfinished job preceding j, thus $\sum_{(j',j) \in E} \bar{c}_{s,j' \to j} = 0$. Observe that if there is a job $j' \in A_s$ with $(j, j') \in E$, the fact that $j \in A_s$ gives $\bar{c}_{s,j \to j'} = w(S(j'))$. By again exploiting the out-forest topology, this yields

$$w_j + \sum_{(j,j') \in E} \bar{c}_{s,j \to j'} - \sum_{(j',j) \in E} \bar{c}_{s,j' \to j} = w_j + \sum_{(j,j') \in E} w(S(j')) = w(S(j)),$$

and hence implies that the left side of (7) is equal to $\sum_{s \in T_j} w(S(j))$.

The facts that $W(t_1) \geq W(t_2)$ at any $t_1 \leq t_2$ and that j is processed by $R_j^s < 1$ units at any time $s \in T_j$ imply combined with (5) the following inequality:

$$m \cdot \sum_{s \in T_j} \frac{w(S(j))}{W(t)} \leq m \cdot \sum_{s \in T_j} \frac{w(S(j))}{W(s)} \leq \sum_{s \in T_j} \rho \cdot R_j^s \leq \rho \cdot p_j.$$

Rearranging it, using the definition of $\bar{b}_t$ and the bound on α gives

$$\sum_{s \in T_j} w(S(j)) \leq \rho \cdot p_j \cdot \frac{1}{m} \cdot W(t) = \rho \cdot \kappa \cdot p_j \cdot \bar{b}_t \leq \alpha \cdot p_j \cdot \bar{b}_t,$$

which implies (7) and thus proves the statement. □

Proof of Theorem E.1. We set $\alpha = \kappa \cdot \rho$. The facts that $\alpha \cdot \text{OPT} \geq \text{OPT}_\alpha$ and $\sum_j w_j \cdot chain_j \leq \text{OPT}$, weak duality and Lemma E.6 imply

$$(1+\alpha) \cdot \text{OPT} \geq \text{OPT}_\alpha + \text{OPT} \geq \sum_j \bar{a}_j - m \sum_{i,t} \bar{b}_t + \sum_j w_j \cdot chain_j.$$

Using Lemma E.5 concludes

$$(1+\alpha) \cdot \text{OPT} \geq \left(1 - \frac{1}{\kappa}\right) \cdot \text{ALG},$$

which, choosing $\kappa = 2$, can be rearranged to $\text{ALG} \leq \left(\frac{1+2\rho}{1-\frac{1}{2}}\right) \cdot \text{OPT} = (2+4\rho) \cdot \text{OPT}$. □

F. Time-Sharing without Monotonicity

We give a proof of the following theorem. The proofs given in (Purohit et al., 2018; Lindermayr & Megow, 2022) require both algorithms to be monotone, that is, for every two instances on the same set of jobs but with processing requirements $\{p'_j\}_j$ and $\{p_j\}_j$ such that for all jobs holds $p'_j \leq p_j$, the algorithm's objective value for $\{p'_j\}_j$ must be less or equal than the algorithm's objective value for $\{p_j\}_j$. In contrast, we do *not* require monotonicity. This even holds when scheduling jobs on unrelated machines, where jobs receive a different progress on different machines. The only requirement on both algorithms is that they are *progress-aware*. That is, they can track the exact progress jobs have received by them so far at any point in time, which is true for most algorithms in almost every model.

Theorem F.1. *Given two deterministic progress-aware algorithms $\mathcal{A}$ and $\mathcal{B}$ with competitive ratios $\rho_\mathcal{A}$ and $\rho_\mathcal{B}$ for minimizing the total weighted completion time with online precedence constraints, there exists an algorithm for the same problem with a competitive ratio of at most $2 \cdot \min\{\rho_\mathcal{A}, \rho_\mathcal{B}\}$.*

Proof. We consider the algorithm which simulates each of both algorithms $\mathcal{A}$ and $\mathcal{B}$ with a rate equal to $\frac{1}{2}$ on the same set of jobs, i.e., in every time step it executes algorithm $\mathcal{A}$ in the first half of the timestep, and algorithm $\mathcal{B}$ in the second half. Additionally, it keeps track of how much each algorithm advances every job. For any job j, both algorithms ignore the progress in the processing of j made by the other algorithm. In particular, if $\mathcal{A}$ finishes job j, then our algorithm still simulates the processing of j until the total time spend by $\mathcal{B}$ on the (partially simulated) processing of j equals p_j (and vice versa if $\mathcal{B}$ finishes a job before $\mathcal{A}$). We assume that an algorithm only can start successors of a job j once the (partially simulated) processing time spend *by the algorithm* on j is at least p_j. We remark that this requires that the main algorithm is able to manipulate the input for the sub-algorithms $\mathcal{A}$ and $\mathcal{B}$. This ensures that the simulated completion times $\tilde{C}_j^\mathcal{A}$, $\tilde{C}_j^\mathcal{B}$ for both algorithms of every job j are exactly doubled compared to the completion times $C_j^\mathcal{A}$, $C_j^\mathcal{B}$ of j in independent schedules of both algorithms, i.e., $\tilde{C}_j^\mathcal{A} = 2 \cdot C_j^\mathcal{A}$ and $\tilde{C}_j^\mathcal{B} = 2 \cdot C_j^\mathcal{B}$. In the actual combined schedule, job j clearly completed not later than $\min\{\tilde{C}_j^\mathcal{A}, \tilde{C}_j^\mathcal{B}\} = 2 \cdot \min\{C_j^\mathcal{A}, C_j^\mathcal{B}\}$, implying the stated bound on its competitive ratio. □

We finally note that this argument also works when the share each of both algorithms receives is parameterized by some $\lambda \in [0, 1]$.