Speed-Oblivious Online Scheduling: Knowing (Precise) Speeds is not Necessary

Alexander Lindermayr [*1] **Nicole Megow** [*1] **Martin Rapp** [*2]

Abstract

We consider online scheduling on unrelated (heterogeneous) machines in a *speed-oblivious* setting, where an algorithm is unaware of the exact job-dependent processing speeds. We show strong impossibility results for clairvoyant and non-clairvoyant algorithms and overcome them in models inspired by practical settings: (i) we provide competitive *learning-augmented* algorithms, assuming that (possibly erroneous) predictions on the speeds are given, and (ii) we provide competitive algorithms for the *speed-ordered* model, where a single global order of machines according to their unknown job-dependent speeds is known. We prove strong theoretical guarantees and evaluate our findings on a representative heterogeneous multi-core processor. These seem to be the first empirical results for scheduling algorithms with predictions that are evaluated in a non-synthetic hardware environment.

1. Introduction

Heterogeneous processors are getting more and more common in various domains. For several years now, efficiency and performance gains in smartphone chips have depended crucially on the combination of high-performance and low-performance (but energy-efficient) cores (ARM Limited, 2013). Heterogeneity has recently been introduced also to the desktop market with Intel Alder Lake (Q1'2022) (Rotem et al., 2022) and AMD Zen 5 (announced for 2023). Further, jobs differ in their instruction mix and memory access patterns, and hence may not benefit uniformly from the high-performance cores, which typically feature larger caches, out-of-order execution, and a higher CPU frequency. Figure 1 shows job-dependent speed varieties in common benchmark suites (*PARSEC-3.0*, *SPLASH-3*, *Polybench*) running on *big* and *LITTLE* cores of a Kirin 970 smartphone system-on-chip (SoC) with Arm big.LITTLE architecture.

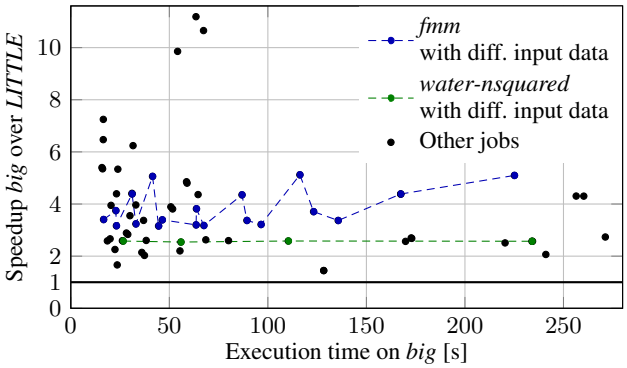


Figure 1. The execution time and speedup of the *big* over *LITTLE* cores on an Arm big.LITTLE heterogeneous processor varies strongly between jobs and different input data. Variations of the speedup w.r.t. input data are large for some jobs (e.g., *water-nsquared*) but small for others (e.g., *fmm*).

These advances show the demand for schedulers that respect *job-dependent heterogeneity*. Formally, the *(processing) speed* s_{ij} of job j on machine i is the amount of processing that j receives when running on i for one time unit. Despite the relevance of values s_{ij} for high-performance scheduling, there is a big discrepancy between how theory and practice handle them: while scheduling theory most commonly assumes that speeds are known to an algorithm, this is typically not the case in practice. Hence, algorithms that perform well in theory are often not applicable in practice.

In this work, we propose new models and algorithms to bridge this gap. In particular, we introduce *speed-oblivious* algorithms, which do not rely on knowing (precise) speeds. Thereby we focus on (non-)clairvoyant scheduling subject to minimizing the total (weighted) completion time.

Formally, an instance of this scheduling problem is composed of a set J of n jobs, a set I of m machines, and a time-discretization. The characteristics of a job $j \in J$ are its processing requirement p_j, its weight w_j, and for every machine $i \in I$ its individual processing speed $s_{ij} > 0$. A job j arrives online at its release date r_j, i.e., an algorithm is unaware of its existence before that time. A schedule

[*]Equal contribution [1]Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany [2]Department of Informatics, Karlsruhe Institute of Technology (KIT), Germany. Correspondence to: Alexander Lindermayr <linderal@uni-bremen.de>.

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

assigns for every unfinished job $j \in J$ and for every machine $i \in I$ at any time $t \geq r_j$ a *(machine) rate* $y_{ijt} \in [0, 1]$, which induces the progress $q_{jt} = \sum_i s_{ij} y_{ijt}$ of j at time t. The completion time C_j of a job j in a fixed schedule is the first point in time t that satisfies $\sum_{t'=r_j}^{t} q_{jt'} \geq p_j$. A schedule is feasible if there exists a progress-preserving actual schedule, where at any infinitesimal time a job is being processed on at most one machine. This applies if the rates satisfy $\sum_{i \in I} y_{ijt} \leq 1$ for all $j \in J$ and $\sum_{j \in J} y_{ijt} \leq 1$ for all $i \in I$ at any time t (Im et al., 2018). The task is to compute a feasible schedule that minimizes $\sum_{j \in J} w_j C_j$.

An algorithm is called *non-migratory*, if it assigns for each job j positive rates only on a single machine i_j, and *migratory* otherwise. Further, it is called *non-preemptive* if for all jobs j, machines i, and times t, a rate $y_{ijt} > 0$ implies $y_{ijt'} = 1$ for all times t' with $t \leq t' \leq C_j$. We say that the machines are *related* if $s_i = s_{ij}$ for all jobs j and machines i, i.e., speeds are not job-dependent.

Models and state-of-the-art in theory Scheduling jobs (offline) on machines with job-dependent heterogeneity (called *unrelated* machine scheduling) to minimize the total weighted completion time is a prominent NP-hard problem; several approximation algorithms are known, e.g., (Hall et al., 1997; Schulz & Skutella, 2002a; Li, 2020; Bansal et al., 2021; Im & Li, 2023). Well-studied *online* models include *online* job arrival (Pruhs et al., 2004), i.e., a job is unknown to an algorithm until its release date r_j, and *non-clairvoyance* (Motwani et al., 1994), i.e., an algorithm has no knowledge about the total processing requirement p_j of a job (as opposed to *clairvoyant* schedulers). In particular, online algorithms cannot revert previous decisions. The performance of an online algorithm is typically evaluated by its *competitive ratio* (Borodin & El-Yaniv, 1998), i.e., the worst-case ratio between the algorithm's objective value and the optimal objective value (given full information upfront) for every instance. We say that an algorithm is ρ-competitive if its competitive ratio is at most ρ. Known online results include (Hall et al., 1997; Chadha et al., 2009; Anand et al., 2012; Im et al., 2014; 2018; Gupta et al., 2020; Jäger, 2021; Bienkowski et al., 2021; Lindermayr & Megow, 2022).

To the best of our knowledge, unrelated machine scheduling has been studied only in a *speed-aware* setting, where an algorithm knows the speeds s_{ij} for available jobs. It is not difficult to see that there are prohibitive lower bounds for speed-oblivious scheduling on (un-)related machines: consider an instance with a single unit-sized job j which makes substantial progress only on one machine. This means that in the worst-case, the first $m - 1$ machines tried by the algorithm have speed ϵ and j makes no substantial progress. Thus, the algorithm spends at least m time units to complete it. Knowing this fast machine upfront allows an optimal solution to complete the job immediately. This implies a competitive ratio of at least $\Omega(m)$ for m machines:

Observation 1.1. *Any speed-oblivious algorithm has a competitive ratio of at least $\Omega(m)$ for minimizing the total (weighted) completion time on m related machines, even if the algorithm is clairvoyant.*

Models and state-of-the-art in practice Practical scheduling algorithms commonly operate in open systems (Feitelson & Rudolph, 1998), where jobs arrive online, are non-clairvoyant, and, in contrast to the assumption in theory, their exact processing speeds on every core are *unknown* upfront. Therefore, state-of-the-practice schedulers usually ignore heterogeneity between jobs, e.g., Linux Energy-Aware Scheduling (The kernel development community, 2019). State-of-the-art schedulers rely on prior knowledge about jobs (Khdr et al., 2015), which is not always available, or rely on predictions of job characteristics to leverage this information gap. Such predictions could be based on prior executions of repeating jobs or on machine-learning-based techniques (Gupta et al., 2018; Rapp et al., 2021). They are often quite precise, but can be highly inaccurate due to varying and unpredictable input data as shown in Figure 1. To the best of our knowledge, all these approaches are evaluated only empirically. In particular, there are no theoretical guarantees on the performance in worst-case scenarios or with respect to a prediction's quality.

1.1. Our Results

We initiate the theoretical study of speed-oblivious algorithms. Since strong lower bounds rule out good worst-case guarantees for speed-oblivious unrelated machine scheduling without further assumptions, we propose two (new) models which are motivated by data-driven machine-learned models and modern heterogeneous hardware architectures:

- **Speed predictions** give algorithms access to values $\hat{s}_{ij}$ for every machine i at the release date of every job j. We measure the accuracy of such a prediction by the *distortion error* μ, where $\mu = \mu_1 \cdot \mu_2$ and
$$\mu_1 = \max_{i \in I, j \in J} \left\{ \frac{\hat{s}_{ij}}{s_{ij}} \right\} \text{ and } \mu_2 = \max_{i \in I, j \in J} \left\{ \frac{s_{ij}}{\hat{s}_{ij}} \right\}.$$
- **Speed-ordered machines** assume an order on I such that for all $i, i' \in I$ and jobs $j \in J$ holds $s_{ij} \geq s_{i'j}$ if and only if $i \leq i'$. Algorithms are aware of this order.

Finally, we compare algorithms for these models with heuristics in experiments on an actual modern heterogeneous chip. These are the first empirical results which show the benefit of learning-augmented algorithms and validate theoretical findings on *real* hardware. In particular, we initiate the investigation in practical applicability of theoretical scheduling algorithms for actual realistic hardware environments.

We now give a more detailed overview of our results.

Learning-augmented algorithms for speed predictions We provide the first learning-augmented algorithms with job-dependent speed predictions and prove error-dependent performance guarantees w.r.t. the distortion error μ. This gives formal evidence on why algorithms perform well in practice, even if the assumed speeds slightly diverge from the true speeds. We further show that a competitive ratio linear in μ is best possible, even for migratory algorithms and related machines. We emphasize that the algorithms do not have access to μ upfront for the given instance.

Theorem 1.2. *For minimizing the total weighted completion time on unrelated machines, there exist speed-oblivious online algorithms with speed predictions that are*

(i) clairvoyant and 8μ-competitive,
(ii) clairvoyant, non-preemptive and $7.216\mu^2$-competitive,
(iii) non-clairvoyant and 108μ-competitive.

For (i), we design a novel and efficient clairvoyant algorithm, which might be of independent interest. It always schedules the subset of jobs that maximizes the total (predicted) density in a feasible job-to-machine assignment, where the density of a job j on machine i is equal to $\frac{w_j s_{ij}}{p_j}$. We show that it is 8-competitive in the speed-aware setting. Interestingly, this algorithm reduces to Smith's rule on a single machine (Smith et al., 1956) and preemptive variants (Schulz & Skutella, 2002b; Megow & Schulz, 2004).

On the technical side, we prove upper bounds on the competitive ratios using the *dual-fitting* technique (Jain et al., 2003; Anand et al., 2012). There, we lower bound the optimal solution by the dual of a linear programming (LP) relaxation, and then show that a specific feasible dual assignment has an objective value which is close to the algorithm's objective value. The main difficulty is therefore to come up with good dual assignment. For (i), we present a new dual setup, which we believe could be helpful for future dual-fitting approaches. The algorithms and proofs for (ii) and (iii) are are inspired by previous work (Greedy WSPT (Gupta et al., 2020), Proportional Fairness (Im et al., 2018)). However, for (iii) we achieve better constants via optimized duals, even for the speed-aware case. In all proofs, we use scalable properties of duals to convert bad decisions due to imprecise predictions into scaled bounds on the competitive ratio.

Novel algorithms for speed-ordered machines The strong lower bound of $\Omega(m)$ on the competitive ratio for speed-oblivious algorithms for m machines crucially relies on accelerating the machine that an algorithm tries last. This argument becomes infeasible in the speed-ordered setting, because the machines are distinguishable upfront. Designing an algorithm is yet still challenging, as precise factors between speeds remain unknown. On the negative side, we show that any constant-competitive algorithm must migrate jobs. This is even true for clairvoyant algorithms and related machines. On the positive side, we present two algorithms:

Theorem 1.3. *There is a clairvoyant speed-oblivious online algorithm for minimizing the total weighted completion time on speed-ordered related machines with a competitive ratio of at most* 8.

We show that this algorithm is not competitive on unrelated machines. Somewhat surprisingly, our non-clairvoyant algorithm achieves non-trivial competitive ratios for both related and unrelated machines, as the following theorem states.

Theorem 1.4. *There is a non-clairvoyant speed-oblivious online algorithm for minimizing the total completion time*

(i) on speed-ordered related machines with a competitive ratio of at most 216*, and*
(ii) on speed-ordered unrelated machines with a competitive ratio of $\Theta(\log(\min\{n, m\}))$.

A crucial observation for deriving these algorithms is that in the speed-ordered setting certain speed-aware algorithms use strategies which can be formulated *even without* precise speed values. An additional challenge is the few-job regime, i.e., there are less jobs than machines, where we have to ensure that the algorithms prefer the fast machines.

1.2. Further Related Work

Uncertainty about machine speeds or, generally, the machine environment, have hardly been studied in scheduling theory. Some works consider scheduling with unknown non-availability periods, i.e., periods with speed 0 (Albers & Schmidt, 2001; Diedrich et al., 2009), permanent breakdowns of a subset of machines (Stein & Zhong, 2020), or more generally arbitrarily changing machine speed for a single machine (Epstein et al., 2012), but not on heterogenous machines. In scheduling with testing, unknown processing requirements of a job (and thus its machine-dependent speed) can be explored by making queries, e.g., (Dürr et al., 2020; Albers & Eckl, 2020; Arantes et al., 2018), but also here heterogenous processors are not considered.

Mitigating pessimistic lower bounds of classic worst-case analysis via untrusted predictions (Mitzenmacher & Vassilvitskii, 2022; Lindermayr & Megow, 2023) has been successfully applied to various scheduling problems (Purohit et al., 2018; Lattanzi et al., 2020; Azar et al., 2021; 2022; Im et al., 2021; Li & Xian, 2021; Lindermayr & Megow, 2022; Antoniadis et al., 2022; Dinitz et al., 2022). While all these results concentrate on the uncertainty of online arrival and non-clairvoyance, Balkanski et al. (2022) consider a robust scheduling problem where machine speeds are only predicted and jobs have to be grouped to be scheduled together before knowing the true machine speeds; such problems without predictions were introduced in (Eberle et al., 2021;

Stein & Zhong, 2020). In contrast, in our model an algorithm will never learn about a job's true speed(s) before its completion and, further, the speeds might be job-dependent.

2. Algorithms with Speed Predictions

In this section, we investigate the model with speed predictions. We first rule out any sublinear error-dependency.

Theorem 2.1. *Any speed-oblivious algorithm with speed predictions has a competitive ratio of at least $\Omega(\min\{\mu, m\})$ for minimizing the total (weighted) completion time, even if the algorithm is clairvoyant and machines are related.*

Proof. Let $\mu_1, \mu_2 \geq 1$ and $\mu = \mu_1 \cdot \mu_2$. Consider an instance $J = \{j\}$ with $p_j = 2\mu$ and $m \geq 2\mu$ machines such that $\hat{s}_i = \mu_1$ for all $1 \leq i \leq m$. The algorithm cannot distinguish the machines. For the first $2\mu - 1$ machines i on which the algorithm processes j, the adversary fixes $s_i = 1$. Thus, at time $2\mu - 1$, the remaining processing requirement of j is at least $2\mu - (2\mu - 1) = 1$ and there exists a machine i' on which j has not been processed yet. Thus, the adversary can set $s_{i'} = \mu$ and complete j on i' within two time units, implying a competitive ratio of at least $\Omega(\min\{\mu, m\})$. □

Observe that this construction already works for two machines when migration is forbidden.

2.1. A Clairvoyant Algorithm

We firstly present a novel migratory algorithm for the clairvoyant setting with known processing requirements for both the speed-aware setting as well as speed predictions. Sequencing jobs by Smith's rule by non-increasing density $\frac{w_j}{p_j}$ (aka Weighted-Shortest-Processing-Time, WSPT) is optimal on a single machine (Smith et al., 1956). In the online setting with release dates, this policy is 2-competitive when applied preemptively on the available unfinished jobs (Schulz & Skutella, 2002b). It can be extended to identical parallel machines (Megow & Schulz, 2004), by processing at any time the (at most) m jobs with highest densities. However, this approach is infeasible on unrelated machines, because jobs can have different densities on every machine.

Inspired by the power of densities, we compute a subset of at most m jobs that instead maximizes the total density, that is, the sum of the densities of the job-to-machine assignment. This can be done efficiently by computing at any time t a matching M_t between alive jobs $j \in J(t) = \{j \in J \mid r_j \leq t \leq C_j\}$ and machines $i \in I$ with edge weights $\hat{\delta}_{ij} = \frac{w_j \hat{s}_{ij}}{p_j}$ using, e.g., the Hungarian algorithm (Kuhn, 1955). In the analysis, we crucially exploit the local optimality of any two matched job-machine pairs via exchange arguments.

Algorithm 1 Maximum Density

Require: time t, speed (predictions) $\{\hat{s}_{ij}\}$

1: Construct a complete bipartite graph $G_t = I \cup J(t)$ where an edge $(i, j) \in I \times J(t)$ has weight $\hat{\delta}_{ij} = \frac{w_j \hat{s}_{ij}}{p_j}$.
2: Compute a maximum-weight matching M_t for G_t.
3: Schedule jobs to machines according to M_t at time t.

Theorem 2.2. *Algorithm 1 has a competitive ratio of at most 8μ for minimizing the total weighted completion time on unrelated machines with speed predictions.*

This theorem implies immediately the following corollary for the speed-aware setting ($\mu = 1$).

Corollary 2.3. *Algorithm 1 has a competitive ratio of at most 8 for minimizing the total weighted completion time on unrelated machines in the speed-aware setting.*

The remaining section is devoted to proof of Theorem 2.2, which uses a dual-fitting argumentation. To this end, we state the standard migratory linear programming relaxation for our objective function (Schulz & Skutella, 2002a). In fact, we state a variant where the machines of an optimal solution run at a lower speed of $\frac{1}{\alpha}$ for $\alpha \geq 1$ (Im et al., 2018).

$$\begin{aligned}
\min \quad & \sum_{i \in I} \sum_{j \in J} \sum_{t \geq 0} w_j \cdot t \cdot \frac{x_{ijt} s_{ij}}{p_j} && && (\mathrm{LP}_\alpha) \\
\text{s.t.} \quad & \sum_{i \in I} \sum_{t \geq 0} \frac{x_{ijt} s_{ij}}{p_j} \geq 1 && \forall j \in J \\
& \sum_{j \in J} \alpha \cdot x_{ijt} \leq 1 && \forall i \in I, t \geq 0 \\
& \sum_{i \in I} \alpha \cdot x_{ijt} \leq 1 && \forall j \in J, t \geq r_j \\
& x_{ijt} \geq 0 && \forall i \in I, j \in J, t \geq r_j \\
& x_{ijt} = 0 && \forall i \in I, j \in J, t < r_j
\end{aligned}$$

Let OPT_α denote the optimal objective value in this restricted setting. The dual of (LP_α) can be written as follows. (From now on we omit obvious set constraints in the notation for an improved readability.)

$$\begin{aligned}
\max \quad & \sum_j a_j - \sum_{i,t} b_{it} - \sum_{j, t \geq r_j} c_{jt} && && (\mathrm{DLP}_\alpha) \\
\text{s.t.} \quad & \frac{a_j s_{ij}}{p_j} - \alpha b_{it} - \alpha c_{jt} \leq w_j \frac{s_{ij} t}{p_j} && \forall i, j, t \geq r_j \\
& a_j, b_{it}, c_{jt'} \geq 0 && \forall i, j, t \; \forall t' \geq r_j
\end{aligned}$$

Fix an instance and the algorithm's schedule. Let $\kappa \geq 1$ be a constant. We define for every machine i and any time t

$$\beta_{it} = \begin{cases} \hat{\delta}_{ij} & \text{if } i \text{ is matched to } j \in J(t) \text{ in } M_t \\ 0 & \text{otherwise,} \end{cases}$$

and for every job j and any time t

$$\gamma_{jt} = \begin{cases} \hat{\delta}_{ij} & \text{if } j \text{ is matched to } i \in I \text{ in } M_t \\ 0 & \text{otherwise.} \end{cases}$$

Consider the following values:

- $\bar{a}_j = w_j C_j$ for every job j,
- $\bar{b}_{it} = \frac{1}{\kappa} \sum_{t' \geq t} \beta_{it'}$ for every machine i and time t, and
- $\bar{c}_{jt} = \frac{1}{\kappa} \sum_{t' \geq t} \gamma_{jt'}$ for every job j and time $t \geq r_j$.

We show in Lemma 2.5 that these values define a feasible solution for the dual problem (DLP$_\alpha$), and that the corresponding dual objective value is at least a certain fraction of the algorithm's solution value (Lemma 2.4). Weak LP duality then implies Theorem 2.2. Let $\text{ALG} = \sum_j w_j C_j$.

Lemma 2.4. $(1 - \frac{2\mu_1}{\kappa})\text{ALG} \leq \sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} - \sum_{j,t \geq r_j} \bar{c}_{jt}$

In the following, let U_t be the set of unfinished jobs at time t, i.e., all jobs j with $t \leq C_j$, and let $W_t = \sum_{j \in U_t} w_j$.

Proof. Fix a time t and a job j. If $j \in U_t$, let $i_1^j, \ldots, i_{z(j)}^j$ be the sequence of individual machine assignments of j between time t and C_j. Let $\hat{\delta}(i,j) := \hat{\delta}_{ij}$. Note that

$$\sum_{\ell=1}^{z(j)} \hat{\delta}(i_\ell^j, j) = \sum_{\ell=1}^{z(j)} \hat{s}_{i_\ell^j, j} \frac{w_j}{p_j} \leq \mu_1 \sum_{\ell=1}^{z(j)} s_{i_\ell^j, j} \frac{w_j}{p_j} \leq \mu_1 w_j.$$

Therefore, $\sum_i \bar{b}_{it} = \frac{1}{\kappa} \sum_{j \in U_t} \sum_{\ell=1}^{z(j)} \hat{\delta}(i_\ell^j, j) \leq \frac{\mu_1}{\kappa} W_t$. Similarly, $\bar{c}_{jt} = \frac{1}{\kappa} \sum_{\ell=1}^{z(j)} \hat{\delta}(i_\ell^j, j) \leq \frac{\mu_1}{\kappa} w_j$. If $j \in J \setminus U_t$, then, $\bar{c}_{jt} = 0$. Hence, $\sum_{j \in J} \bar{c}_{jt} \leq \frac{\mu_1}{\kappa} W_t$. Finally, we conclude $\sum_{i,t} \bar{b}_{it} \leq \frac{\mu_1}{\kappa} \text{ALG}$ and $\sum_{j,t \geq r_j} \bar{c}_{jt} \leq \frac{\mu_1}{\kappa} \text{ALG}$. □

Lemma 2.5. *Assigning $a_j = \bar{a}_j$, $b_{it} = \bar{b}_{it}$ and $c_{jt} = \bar{c}_{jt}$ is feasible for* (DLP$_\alpha$) *if $\alpha = \mu_2 \kappa$.*

Proof. First note that the dual assignment is non-negative. Let $i \in I, j \in J$ and $t \geq r_j$. The definition of $\bar{a}_j$ yields $\bar{a}_j \frac{s_{ij}}{p_j} - w_j t \frac{s_{ij}}{p_j} \leq \sum_{t'=t}^{C_j} \frac{w_j s_{ij}}{p_j}$. By using the fact that $\frac{w_j s_{ij}}{p_j} \leq \mu_2 \frac{w_j \hat{s}_{ij}}{p_j}$, the definitions of $\bar{b}_{it}$ and $\bar{c}_{jt}$, and the value of α, it remains to validate for every $t \leq t' \leq C_j$ that $\frac{w_j \hat{s}_{ij}}{p_j} = \hat{\delta}_{ij} \leq \beta_{it'} + \gamma_{jt'}$. We distinguish five cases:

(i) If $(i,j) \in M_{t'}$, then $\hat{\delta}_{ij} = \beta_{it'} = \gamma_{jt'}$.

(ii) If $(i,j') \in M_{t'}$ and $(i',j) \in M_{t'}$ s.t. $i' \neq i$ (and thus $j' \neq j$), we know by the optimality of $M_{t'}$ that

$$\hat{\delta}_{ij} \leq \hat{\delta}_{ij} + \hat{\delta}_{i'j'} \leq \hat{\delta}_{i'j} + \hat{\delta}_{ij'} = \gamma_{jt'} + \beta_{it'}.$$

Algorithm 2 Greedy WSPT

Require: speed predictions $\{\hat{s}_{ij}\}$
  function UponJobArrival(job j)
    Assign job j to machine $g(j) = \arg\min_{i \in I} \hat{Q}_{ij}$.
  end function
  function UponMachineIdle(machine i, time t)
    Start processing the job j with largest $\hat{\delta}_{ij}$ among all alive jobs assigned to i which satisfy $\hat{r}_{ij} \leq t$.
  end function

(iii) If $(i',j) \in M_{t'}$ and i is not matched in $M_{t'}$, we conclude $\hat{\delta}_{ij} \leq \hat{\delta}_{i'j} = \gamma_{jt'}$.

(iv) If $(i,j') \in M_{t'}$ and j is not matched in $M_{t'}$, we conclude $\hat{\delta}_{ij} \leq \hat{\delta}_{ij'} = \beta_{it'}$.

(v) The case where $\hat{s}_{ij} > 0, w_j > 0$, but both i and j are unmatched in $M_{t'}$ contradicts the optimality of $M_{t'}$, as $t' \leq C_j$. Else holds $\hat{\delta}_{ij} = 0$, and we conclude since the right side of the inequality is non-negative. □

Proof of Theorem 2.2. Weak LP duality implies that the optimal objective value of (DLP$_\alpha$) is greater or equal to the optimal objective value of (LP$_\alpha$). Being the objective value of a relaxation, the latter is a lower bound on OPT_α, which in turn is at most αOPT by scaling completion times, where OPT denotes the optimal objective value of the original problem. This implies via Lemma 2.4 and Lemma 2.5

$$\mu_2 \kappa \cdot \text{OPT} \geq \text{OPT}_{\mu_2 \kappa} \geq \sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} - \sum_{j,t \geq r_j} \bar{c}_{jt}$$
$$\geq \left(1 - \frac{2\mu_1}{\kappa}\right) \cdot \text{ALG}.$$

Choosing $\kappa = 4\mu_1$, we conclude $\text{ALG} \leq 8\mu \cdot \text{OPT}$. □

2.2. A Clairvoyant Non-Preemptive Algorithm

In many applications, job migration or preemption are not possible. In this section, we show that the non-preemptive Greedy WSPT algorithm by (Gupta et al., 2020) achieves an error-dependent competitive ratio when using predicted speeds to make decisions (Algorithm 2). The intuition of this algorithm is to greedily assign arriving jobs to machines, where they are then scheduled in WSPT order, i.e., on machine i by non-decreasing $\frac{w_j s_{ij}}{p_j}$. The greedy job-to-machine assignment intuitively minimizes the increase of the objective value that scheduling the job on a machine incurs in the current state. Additionally, the execution of job j is delayed depending on its processing time $\frac{p_j}{s_{ij}}$ on the assigned machine i. This is necessary due to simple lower bounds in the non-preemptive setting (Lu et al., 2003).

To make this precise, for every $j \in J$, let $M_i(j)$ be the set of jobs, excluding job j, which are assigned to machine i at

Algorithm 3 Proportional Fairness

Require: time t, speed predictions $\{\hat{s}_{ij}\}$
Use solution $\{y_{ijt}\}_{i,j}$ of (CP_t) as rates at time t.

time r_j, but have not been started yet. As this definition is ambiguous if there are two jobs j and j' with $r_j = r_{j'}$ being assigned to i, we assume that we assign them in the order of their index. For all machines i, jobs j and a constant $\theta > 0$, which we will set $\theta = \frac{2}{3}$, we define $\hat{r}_{ij} = \max\{r_j, \theta \frac{p_j}{\hat{s}_{ij}}\}$ and $\hat{Q}_{ij}$ as

$$w_j \left(\hat{r}_{ij} + \frac{\hat{r}_{ij}}{\theta} + \frac{p_j}{\hat{s}_{ij}} + \sum_{\substack{j' \in M_i(j) \\ \hat{\delta}_{ij'} \geq \hat{\delta}_{ij}}} \frac{p_{j'}}{\hat{s}_{ij'}} \right) + \frac{p_j}{\hat{s}_{ij}} \sum_{\substack{j' \in M_i(j) \\ \hat{\delta}_{ij'} < \hat{\delta}_{ij}}} w_{j'}.$$

We prove in Appendix A.1 the following theorem.

Theorem 2.6. *Algorithm 2 has a competitive ratio of at most $\frac{368}{51}\mu^2 < 7.216\mu^2$ for minimizing the total weighted completion time on unrelated machines with speed predictions.*

2.3. A Non-Clairvoyant Algorithm

In the non-clairvoyant setting, any constant-competitive algorithm for minimizing the total completion time on unrelated machines has to migrate and preempt jobs (Motwani et al., 1994; Gupta et al., 2012). Since such algorithms cannot compute densities, a common strategy is to run all jobs simultaneously at a rate proportional to their weight (Motwani et al., 1994; Kim & Chwa, 2003). On unrelated machines with job-dependent speeds, the Proportional Fairness Algorithm (PF) develops this idea further by respecting job-dependent speeds (Im et al., 2018). It is known that PF has a competitive ratio of at most 128 for minimizing the total weighted completion time (Im et al., 2018). In the following, we show that PF has a linear error-dependency in μ when computing rates via predicted speeds. As a byproduct, we slightly improve the upper bound on the speed-aware competitive ratio of PF via optimized duals to 108.

Theorem 2.7. *Algorithm 3 has a competitive ratio of at most 108μ for minimizing the total weighted completion time on unrelated machines with predicted speeds.*

At every time t, Algorithm 3 schedules jobs $J(t)$ with rates computed via the following convex program (CP_t) with variables $\hat{y}_{ijt}$ for every machine i and job $j \in J(t)$.

$$\begin{aligned} \max \quad & \sum_{j \in J(t)} w_j \log \left(\sum_{i \in I} \hat{s}_{ij} \hat{y}_{ijt} \right) && (\text{CP}_t) \\ \text{s.t.} \quad & \sum_{j \in J(t)} \hat{y}_{ijt} \leq 1 && \forall i \in I \\ & \sum_{i \in I} \hat{y}_{ijt} \leq 1 && \forall j \in J(t) \\ & \hat{y}_{ijt} \geq 0 && \forall i \in I, j \in J(t) \end{aligned}$$

We now give an overview over the proof of Theorem 2.7 and defer further details to Appendix A.2.

Fix an instance and PF's schedule. Let $\kappa \geq 1$ and $0 < \lambda < 1$ be constants which we fix later. In the following, we assume by scaling that all weights are integers. For every time t, let Z^t be the sorted (ascending) list of length W_t composed of w_j copies of $\frac{q_{jt}}{p_j}$ for every $j \in U_t$. We define ζ_t as the value at the index $\lfloor \lambda W_t \rfloor$ in Z^t. Let $\{\eta_{it}\}_{i,t}$ and $\{\theta_{jt}\}_{j \in J(t),t}$ be the KKT multipliers of the first two constraint sets of the optimal solution $\{y_{ijt}\}_{i,j}$. Let $\mathbb{1}[\varphi]$ be the indicator variable of the formula φ, and consider the following duals: $\bar{a}_j = \sum_{t'=0}^{C_j} w_j \cdot \mathbb{1}\left[\frac{q_{jt'}}{p_j} \leq \zeta_{t'}\right]$ for every job j, $\bar{b}_{it} = \frac{1}{\kappa} \sum_{t' \geq t} \zeta_{t'} \eta_{it'}$ for every machine i and time t, and $\bar{c}_{jt} = \frac{1}{\kappa} \sum_{t'=t}^{C_j} \zeta_{t'} \theta_{jt'}$ for every job j and time $t \geq r_j$.

We show that this assignment has an objective value which lower bounds a fraction of PF's objective value, and that it is feasible for (DLP_α) for some values of α.

Lemma 2.8. $(\lambda - \frac{4}{(1-\lambda)\kappa})\text{ALG} \leq \sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} - \sum_{j,t \geq r_j} \bar{c}_{jt}$

Lemma 2.9. *Assigning $a_j = \bar{a}_j$, $b_{it} = \bar{b}_{it}$ and $c_{jt} = \bar{c}_{jt}$ is feasible for (DLP_α) if $\alpha = \kappa\mu$.*

Both lemmas and weak LP duality yield Theorem 2.7 when choosing $\kappa = 36$ and $\lambda = \frac{2}{3}$.

3. Algorithms for Speed-Ordered Machines

This section contains our results on speed-ordered machines. In the first subsection, we present a clairvoyant algorithm, and in the second subsection a non-clairvoyant algorithm. But first, we observe in Appendix B that in this model migration is necessary for speed-oblivious algorithms.

Theorem 3.1. *Any non-migratory speed-oblivious algorithm has a competitive ratio of at least $\Omega(m)$ for minimizing the total completion time on m speed-ordered machines, even if it is clairvoyant and the machines are related.*

3.1. A Clairvoyant Algorithm

Our clairvoyant algorithm for speed-ordered related machines is motivated by the following observation. If the

Algorithm 4 Maximum Density for speed-ordered machines

Require: time t, speed-ordered machines $s_1 \geq \ldots \geq s_m$

1: $\sigma_t \leftarrow$ order of $J(t)$ with non-increasing $\frac{w_j}{p_j}$.
2: $M_t = \{(k, \sigma_t(k))\}_{k \in [\ell]}$ where $\ell = \min\{m, |J(t)|\}$
3: Schedule jobs to machines according to M_t at time t.

Algorithm 5 Round Robin for speed-ordered machines

Require: time t, speed-ordered machines $s_{1j} \geq \ldots \geq s_{mj}$
Use rates $y_{ijt} = |J(t)|^{-1} \cdot \mathbb{1}[i \leq |J(t)|]$ at time t.

machines are related and speed-ordered, Algorithm 1, given correct speed predictions, will assign jobs by non-increasing order of $\frac{w_j}{p_j}$ to machines in speed order, because this clearly maximizes the total scheduled density, i.e., sum of assigned $\frac{w_j s_i}{p_j}$. Algorithm 4 can therefore emulate this schedule of maximum density *without* having to compute a maximum matching, and thus does not require (predicted) speeds. These observations also suggest that the analysis must be similar. Indeed, we can use a similar dual-fitting as for Theorem 2.2 to prove the following theorem in Appendix B.1, where we mainly present a new proof for the dual feasibility.

Theorem 3.2. *Algorithm 4 has a competitive ratio of at most* 8 *for minimizing the total weighted completion time on speed-ordered related machines.*

Note that the above observation does not hold for unrelated machines. Indeed, in Appendix B.1 we show that Algorithm 4 is at least $\Omega(n)$-competitive on unrelated machines.

3.2. A Non-Clairvoyant Algorithm

The non-clairvoyant setting is more difficult. This is because the schedules of speed-aware algorithms, such as PF, are not as easy to describe, as it was the case for clairvoyant algorithms. However, for unit weights, related machines and many alive jobs, i.e., $|J(t)| \geq m$, one solution of (CP_t) is to schedule all jobs on all machines with the same rate, i.e., do Round Robin on every machine. We can describe this schedule without knowing anything about the speeds. However, in the few-job regime, i.e., $|J(t)| < m$, this approach violates the packing constraints of the jobs, i.e., $\sum_i y_{ijt} > 1$. This is where the speed order comes into play: we partition a job's available rate only to the $|J(t)|$ fastest machines. For the final algorithm (Algorithm 5), we prove below a guarantee for unrelated machines, and a constant upper bound for related machines in Appendix B.2.2.

Theorem 3.3. *Algorithm 5 has a competitive ratio of at most* $\mathcal{O}(\log(\min\{n, m\}))$ *for minimizing the total completion time on speed-ordered unrelated machines.*

In Appendix B.2.1 we show that this bound is tight for Algorithm 5, even if all speeds are from $\{0, 1\}$. We prove Theorem 3.3 via dual-fitting based on (DLP_α), where $w_j = 1$ for every job j. Fix an instance and the algorithm's schedule. For every time t, we write $m_t = \min\{m, |J(t)|\}$, and we define $\beta_{it} = \frac{1}{i} \cdot |J(t)| \cdot \mathbb{1}[i \leq |J(t)|]$ for every machine i, and $\gamma_{jt} = \mathbb{1}[j \in J(t)]$ for every job j.

Let $\kappa = \Theta(\log(\min\{n, m\}))$. Intuitively, this factor upper bounds $\sum_{i=1}^{m_t} \frac{1}{i}$, which will be necessary when handling $\sum_i \beta_{it}$. For related machines, we can alter the definition of β_{it} and thus have a constant κ, which eventually implies a constant upper bound on the competitive ratio.

For every time t, consider the sorted (ascending) list Z^t composed of values $\frac{q_{jt}}{p_j}$ for every $j \in U_t$. We define ζ_t as the value at the index $\lfloor \frac{1}{2}|U_t| \rfloor$ in Z^t. Consider the following duals: $\bar{a}_j = \sum_{t'=0}^{C_j} \mathbb{1}\left[\frac{q_{jt'}}{p_j} \leq \zeta_{t'}\right]$ for every job j, $\bar{b}_{it} = \frac{1}{\kappa} \sum_{t' \geq t} \beta_{it'} \zeta_{t'}$ for every machine i and time t, and $\bar{c}_{jt} = \frac{1}{\kappa} \sum_{t' \geq t} \gamma_{jt'} \zeta_{t'}$ for every job j and time $t \geq r_j$. We prove the following bound on ALG in Appendix B.2.1.

Lemma 3.4. $\Omega(1) \cdot \text{ALG} \leq \sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} - \sum_{j, t \geq r_j} \bar{c}_{jt}$

This lemma, weak LP duality, and the feasibility of the crafted duals (Lemma 3.5) imply Theorem 3.3 for $\alpha = \kappa$.

Lemma 3.5. *Assigning* $a_j = \bar{a}_j$, $b_{it} = \bar{b}_{it}$ *and* $c_{jt} = \bar{c}_{jt}$ *is feasible for* (DLP_α) *if* $\alpha = \kappa$.

Proof. First observe that the dual assignment is non-negative. Let $i \in I, j \in J$ and $t \geq r_j$. Since the rates of Algorithm 5 imply $q_{jt} = \sum_{\ell=1}^{m_t} \frac{s_{\ell j}}{|J(t)|}$, we have

$$\frac{\bar{a}_j s_{ij}}{p_j} - \frac{s_{ij} \cdot t}{p_j} \leq \sum_{t'=t}^{C_j} \frac{s_{ij}}{p_j} \cdot \mathbb{1}\left[\frac{q_{jt'}}{p_j} \leq \zeta_{t'}\right] \leq \sum_{t'=t}^{C_j} \frac{s_{ij}}{q_{jt'}} \zeta_{t'}$$
$$\leq \sum_{t'=t}^{C_j} \frac{s_{ij}}{\sum_{\ell=1}^{m_{t'}} \frac{s_{\ell j}}{|J(t')|}} \zeta_{t'}. \tag{1}$$

Consider any time t' with $t \leq t' \leq C_j$. If $i \leq |J(t')|$, by the speed order, $\sum_{\ell=1}^{m_{t'}} s_{\ell j} \geq \sum_{\ell=1}^{i} s_{\ell j} \geq i \cdot s_{ij}$, and thus

$$\frac{s_{ij}}{\sum_{\ell=1}^{m_{t'}} s_{\ell j}} \cdot |J(t')| \cdot \zeta_{t'} \leq \frac{1}{i} \cdot |J(t')| \cdot \zeta_{t'} = \beta_{it'} \cdot \zeta_{t'}.$$

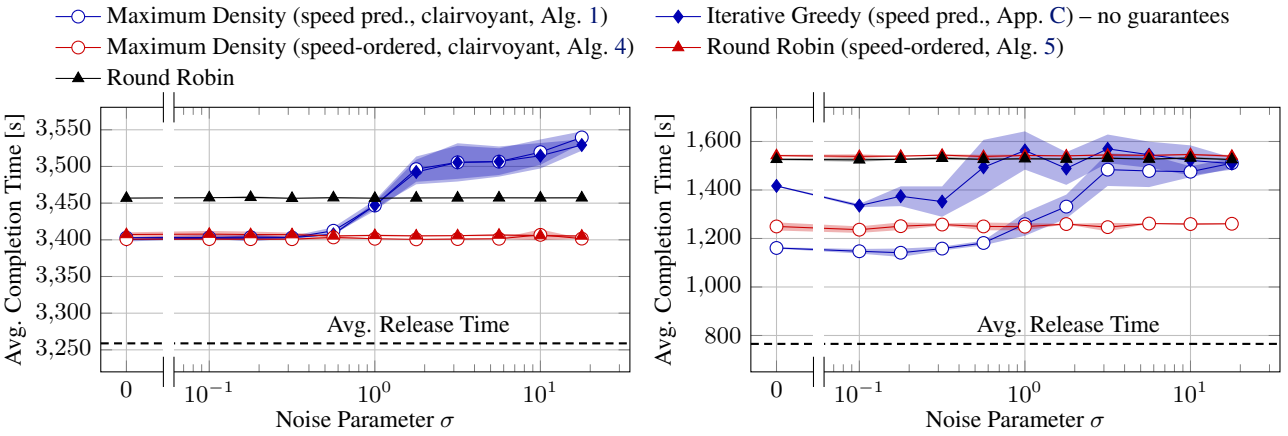
Otherwise, that is, $i > |J(t')|$, we conclude by the speed order, $\sum_{\ell=1}^{m_{t'}} s_{\ell j} \geq \sum_{\ell=1}^{|J(t')|} s_{\ell j} \geq |J(t')| \cdot s_{ij}$. Therefore,

$$\frac{s_{ij}}{\sum_{\ell=1}^{m_{t'}} s_{\ell j}} \cdot |J(t')| \cdot \zeta_{t'} \leq \frac{|J(t')|}{|J(t')|} \cdot \zeta_{t'} = \gamma_{jt'} \cdot \zeta_{t'},$$

because $t' \leq C_j$. Put together, (1) is at most

$$\sum_{t'=t}^{C_j} \beta_{it'} \zeta_{t'} + \sum_{t'=t}^{C_j} \gamma_{jt'} \zeta_{t'} \leq \kappa(\bar{b}_{it} + \bar{c}_{jt}),$$

which verifies the dual constraint. □



(a) Low load: 1 task / min.

(b) High load: 4 tasks / min.

Figure 2. Real experiments on a *HiKey 970* board. The experiments are each repeated 3 times with the same workload but different random noise for speed predictions. Shaded areas show the standard deviation.

4. Experimental Evaluation

Setup We perform experiments on real hardware running representative jobs, which enables us to perform a realistic evaluation. The setup uses a *HiKey 970* board (Linaro 96Boards) with a *Kirin 970* Arm big.LITTLE SoC featuring 4 *big* cores and 4 *LITTLE* cores, running Android 8.0. This is a representative smartphone platform. The *big* cores always offer a higher performance than the *LITTLE* cores (speed-ordered) because they support out-of-order execution at higher frequency and larger caches (see also Figure 1, all speedups are > 1). Our workload comprises 100 randomly selected single-threaded jobs from the well-established *PARSEC-3.0* (Zhan et al., 2016), *SPLASH-3* (Sakalis et al., 2016), and Polybench (Yuki & Pouchet, 2015) benchmark suites. These benchmarks represent various use cases from video transcoding, rendering, compression, etc. The arrival times are drawn from a Poisson distribution with varying rate parameter to study different system loads. We characterized all jobs offline to get accurate speed s_{ij} and job volume p_j values. Speed predictions are created with controllable error by $\hat{s}_{ij} = s_{ij} \cdot y_{ij}$, where y_{ij} follows a log-normal distribution $ln(y_{ij}) \sim \mathcal{N}(0, \sigma^2)$. Note that the predictions do not consider slowdown effects on real hardware, e.g., due to shared resource contention, adding additional inaccuracy.

We consider all algorithms presented in previous sections. Additionally, we consider Round Robin (RR), which distributes a job evenly over all machines, and Iterative Greedy, which at any time iteratively schedules the job j on machine i which has the maximum $\hat{s}_{ij}$ among all unassigned alive jobs and free machines. We show in Appendix C that Iterative Greedy is not competitive (lower bound of $\Omega(n)$).

Additionally, we perform synthetic experiments (Appendix C), which use similar workload and core configurations, but are only simulated. An advantage is that rates must not be transformed to actual schedules. The results are in line with the results of our hardware experiments.

Results Figure 2 presents the results of the hardware experiments. We exclude PF because it produces fractional schedules which are often difficult to convert into real schedules (Im et al., 2018), and Greedy WSPT, because, given incorrect predictions, it significantly underperforms in synthetic experiments. We repeat each experiment 3 times with the same workload (jobs and arrival times) but different random noisy speed predictions and plot the average and standard deviation of the average completion times.

Under low system load (Figure 2a), the number of active jobs is mostly ≤ 4, i.e., it is mostly feasible to only use the *big* cores (load analysis in Appendix C). Consequently, the algorithms that exploit the speed-ordered property (red) consistently perform best. Algorithms with speed predictions (blue) perform equally well for accurate predictions but their performance deteriorates for very noisy predictions. RR always uses all cores and thus shows a low performance.

Under high system load (Figure 2b), the number of active jobs is mostly > 4, thus, *LITTLE* cores have to be used. RR and speed-ordered RR perform similarly, as both mostly use the same cores. For low prediction noise ($\sigma < 1$), Maximum Density performs best, but also requires most information (speed predictions and clairvoyant). For higher prediction noise, speed-ordered Maximum Density is better because too noisy speed predictions result in bad schedules. Iterative Greedy performs best among the non-clairvoyant algorithms, but does not offer any theoretical guarantees.

Speed predictions are beneficial in the average case if they are relatively accurate. With inaccurate predictions, relying on the speed-ordering instead is beneficial. *In summary, our experiments show the power of speed predictions and*

speed-ordering for online scheduling in real-world settings.

5. Conclusion and Future Directions

We initiated research on speed-oblivious algorithms with two models motivated by real-world observations. Future directions include settling the asymptotic competitive ratio for (non-)clairvoyant speed-oblivious algorithms on speed-ordered unrelated machines, shrinking the upper bound of PF to a small constant, and investigating speed-oblivious algorithms for other objective functions such as the total flow time, potentially also in the speed-scaling model.

Acknowledgements

This research has been funded by the German Science Foundation (DFG) under contract 146371743 – TRR 89 Invasive Computing.

References

Albers, S. and Eckl, A. Explorable uncertainty in scheduling with non-uniform testing times. In *WAOA*, volume 12806 of *Lecture Notes in Computer Science*, pp. 127–142. Springer, 2020.

Albers, S. and Schmidt, G. Scheduling with unexpected machine breakdowns. *Discret. Appl. Math.*, 110(2-3): 85–99, 2001.

Anand, S., Garg, N., and Kumar, A. Resource augmentation for weighted flow-time explained by dual fitting. In *SODA*, pp. 1228–1241. SIAM, 2012.

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.

Arantes, L., Bampis, E., Kononov, A. V., Letsios, M., Lucarelli, G., and Sens, P. Scheduling under uncertainty: A query-based approach. In *IJCAI*, pp. 4646–4652, 2018.

ARM Limited. big.LITTLE Technology: The Future of Mobile, 2013.

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, 2022.

Balkanski, E., Ou, T., Stein, C., and Wei, H. Scheduling with speed predictions. *CoRR*, abs/2205.01247, 2022.

Bansal, N., Srinivasan, A., and Svensson, O. Lift-and-round to improve weighted completion time on unrelated machines. *SIAM J. Comput.*, 50(3), 2021.

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.

Chadha, J. S., Garg, N., Kumar, A., and Muralidhara, V. N. A competitive algorithm for minimizing weighted flow time on unrelated machines with speed augmentation. In *STOC*, pp. 679–684. ACM, 2009.

Diedrich, F., Jansen, K., Schwarz, U. M., and Trystram, D. A survey on approximation algorithms for scheduling with machine unavailability. In *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pp. 50–64. Springer, 2009.

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.

Dürr, C., Erlebach, T., Megow, N., and Meißner, J. An adversarial model for scheduling with testing. *Algorithmica*, 82(12):3630–3675, 2020.

Eberle, F., Hoeksma, R., Megow, N., Nölke, L., Schewior, K., and Simon, B. Speed-robust scheduling - sand, bricks, and rocks. In *IPCO*, volume 12707 of *Lecture Notes in Computer Science*, pp. 283–296. Springer, 2021.

Epstein, L., Levin, A., Marchetti-Spaccamela, A., Megow, N., Mestre, J., Skutella, M., and Stougie, L. Universal sequencing on an unreliable machine. *SIAM J. Comput.*, 41(3):565–586, 2012.

Feitelson, D. G. and Rudolph, L. Metrics and benchmarking for parallel job scheduling. In *JSSPP*, volume 1459 of *Lecture Notes in Computer Science*, pp. 1–24. Springer, 1998.

Gupta, A., Im, S., Krishnaswamy, R., Moseley, B., and Pruhs, K. Scheduling heterogeneous processors isn't as easy as you think. In *SODA*, pp. 1242–1253. SIAM, 2012.

Gupta, U., Babu, M., Ayoub, R., Kishinevsky, M., Paterna, F., and Ogras, Ü. Y. STAFF: online learning with stabilized adaptive forgetting factor and feature selection algorithm. In *DAC*, pp. 177:1–177:6. ACM, 2018.

Gupta, V., Moseley, B., Uetz, M., and Xie, Q. Greed works - online algorithms for unrelated machine stochastic scheduling. *Math. Oper. Res.*, 45(2):497–516, 2020.

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

Im, S. and Li, S. Improved approximations for unrelated machine scheduling. In *SODA*, pp. 2917–2946. SIAM, 2023.

Im, S., Kulkarni, J., Munagala, K., and Pruhs, K. Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. In *FOCS*, pp. 531–540. IEEE Computer Society, 2014.

Im, S., Kulkarni, J., and Munagala, K. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. *J. ACM*, 65(1): 3:1–3:33, 2018.

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.

Jain, K., Mahdian, M., Markakis, E., Saberi, A., and Vazirani, V. V. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 50 (6):795–824, 2003.

Khdr, H., Pagani, S., Shafique, M., and Henkel, J. Thermal constrained resource management for mixed ILP-TLP workloads in dark silicon chips. In *DAC*, pp. 179:1–179:6. ACM, 2015.

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

Kuhn, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2): 83–97, 1955.

Lattanzi, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Online scheduling via learned weights. In *SODA*, pp. 1859–1877. SIAM, 2020.

Li, S. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. *SIAM J. Comput.*, 49(4), 2020.

Li, S. and Xian, J. Online unrelated machine load balancing with predictions revisited. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6523–6532. PMLR, 2021.

Linaro 96Boards. Hikey970. https://96boards.org/product/hikey970/.

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/.

Lu, X., Sitters, R., and Stougie, L. A class of on-line scheduling algorithms to minimize total completion time. *Oper. Res. Lett.*, 31(3):232–236, 2003.

Megow, N. and Schulz, A. S. On-line scheduling to minimize average completion time revisited. *Oper. Res. Lett.*, 32(5):485–490, 2004.

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

Motwani, R., Phillips, S. J., and Torng, E. Non-clairvoyant scheduling. *Theor. Comput. Sci.*, 130(1):17–47, 1994.

Pruhs, K., Sgall, J., and Torng, E. Online scheduling. In *Handbook of Scheduling*. Chapman and Hall/CRC, 2004.

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

Rapp, M., Pathania, A., Mitra, T., and Henkel, J. Neural network-based performance prediction for task migration on S-NUCA many-cores. *IEEE Trans. Computers*, 70 (10):1691–1704, 2021.

Rotem, E., Yoaz, A., Rappoport, L., Robinson, S. J., Mandelblat, J. Y., Gihon, A., Weissmann, E., Chabukswar, R., Basin, V., Fenger, R., et al. Intel Alder Lake CPU Architectures. *IEEE Micro*, 42(3):13–19, 2022.

Sakalis, C., Leonardsson, C., Kaxiras, S., and Ros, A. Splash-3: A properly synchronized benchmark suite for contemporary research. In *ISPASS*, pp. 101–111. IEEE Computer Society, 2016.

Schulz, A. S. and Skutella, M. Scheduling unrelated machines by randomized rounding. *SIAM J. Discret. Math.*, 15(4):450–469, 2002a.

Schulz, A. S. and Skutella, M. The power of α-points in preemptive single machine scheduling. *Journal of Scheduling*, 5(2):121–133, 2002b.

Smith, W. E. et al. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2): 59–66, 1956.

Stein, C. and Zhong, M. Scheduling when you do not know the number of machines. *ACM Trans. Algorithms*, 16(1): 9:1–9:20, 2020.

The kernel development community. Energy Aware Scheduling – The Linux Kernel Documentation, 2019. https://www.kernel.org/doc/html/v5.3/scheduler/sched-energy.html.

Yuki, T. and Pouchet, L.-N. Polybench 4.0, 2015.

Zhan, X., Bao, Y., Bienia, C., and Li, K. PARSEC3.0: A multicore benchmark suite with network stacks and SPLASH-2X. *SIGARCH Comput. Archit. News*, 44(5): 1–16, 2016.

A. Details on Algorithms with Speed Predictions

A.1. Full Analysis of Greedy WSPT with Speed Predictions

In this section, we present an error-dependent competitive ratio for Greedy WSPT with speed predictions and eventually prove Theorem 2.6. The analysis is inspired by (Gupta et al., 2020), but uses a different approach for proving the feasibility of the crafted duals. In particular, we need less scaling parameters than Gupta et al.

Theorem 2.6. *Algorithm 2 has a competitive ratio of at most $\frac{368}{51}\mu^2 < 7.216\mu^2$ for minimizing the total weighted completion time on unrelated machines with speed predictions.*

Fix an instance and the algorithm's schedule. Let $\kappa \geq 1$ and $0 < \theta < 1$ be constants. We assume w.l.o.g. by scaling the instance that all processing requirements and release dates are integer multiples of κ. Recall that $\hat{\delta}_{ij} = \frac{w_j \hat{s}_{ij}}{p_j}$ and $\hat{r}_{ij} = \max\{r_j, \theta \frac{p_j}{\hat{s}_{ij}}\}$. We write for every job j and machine i

$$Q_{ij} = w_j \left(\hat{r}_{ij} + \mu_1 \frac{\hat{r}_{ij}}{\theta} + \frac{p_j}{s_{ij}} + \sum_{\substack{j' \in M_i(j) \\ \hat{\delta}_{ij'} \geq \hat{\delta}_{ij}}} \frac{p_{j'}}{s_{ij'}} \right) + \frac{p_j}{s_{ij}} \sum_{\substack{j' \in M_i(j) \\ \hat{\delta}_{ij'} < \hat{\delta}_{ij}}} w_{j'}.$$

Also, recall that the algorithm uses the values $\hat{Q}_{ij}$ to assign a job j at time r_j to machine $g(j) = \arg\min_i \hat{Q}_{ij}$:

$$\hat{Q}_{ij} = w_j \left(\hat{r}_{ij} + \frac{\hat{r}_{ij}}{\theta} + \frac{p_j}{\hat{s}_{ij}} + \sum_{\substack{j' \in M_i(j) \\ \hat{\delta}_{ij'} \geq \hat{\delta}_{ij}}} \frac{p_{j'}}{\hat{s}_{ij'}} \right) + \frac{p_j}{\hat{s}_{ij}} \sum_{\substack{j' \in M_i(j) \\ \hat{\delta}_{ij'} < \hat{\delta}_{ij}}} w_{j'}.$$

We now introduce a linear programming relaxation of our problem. As we consider a non-preemptive scheduling problem here, we can define a stronger linear program relaxation than (LP$_\alpha$) (Schulz & Skutella, 2002a):

$$\begin{aligned} \min \quad & \sum_{i,j,t} w_j \cdot x_{ijt} \cdot \left(\frac{1}{2} + \frac{s_{ij}}{p_j} \cdot \left(t + \frac{1}{2} \right) \right) && \text{(NP-LP)} \\ \text{s.t.} \quad & \sum_{i, t \geq r_j} \frac{x_{ijt} s_{ij}}{p_j} \geq 1 && \forall j \\ & \sum_j x_{ijt} \leq 1 && \forall i, t \\ & x_{ijt} \geq 0 && \forall i, j, t \\ & x_{ijt} = 0 && \forall i, j, t < r_j \end{aligned}$$

This relaxation has an integrality gap of 2 (Schulz & Skutella, 2002a). The dual of (NP-LP) can be written as follows:

$$\begin{aligned} \max \quad & \sum_j a_j - \sum_{i,t} b_{it} && \text{(NP-DLP)} \\ \text{s.t.} \quad & \frac{a_j s_{ij}}{p_j} - b_{it} \leq w_j \left(s_{ij} \frac{t + 1/2}{p_j} + \frac{1}{2} \right) \quad \forall i, j, t \geq r_j && (2) \\ & a_j, b_{it} \geq 0 \qquad \forall i, j, t \end{aligned}$$

We define a solution for (NP-DLP) which depends on the schedule produced by the algorithm. Let $U_i(t) = \{j \in J \mid g(j) = i \wedge t < C_j\}$. Note that $U_i(t)$ includes unreleased jobs at time t. Consider the following dual assignment:

- $\bar{a}_j = Q_{g(j)j}$ for every job j and
- $\bar{b}_{it} = \mu \cdot \sum_{j \in U_i(\kappa \cdot t)} w_j$ for every machine i and time t.

We first show that the objective value of (NP-DLP) for $(\bar{a}_j, \bar{b}_{it})$ is close to the objective value of the algorithm.

Lemma A.1. $\sum_j \bar{a}_j \geq \text{ALG}$

Proof. Consider the algorithm's schedule. Let $x_i(t)$ denote the amount of time (not volume) the currently processed job on machine i requires to complete. If there is no job running on machine i at time t, we define $x_i(t) = 0$. We now calculate the contribution of some job j to the algorithm's objective value ALG. Suppose that j gets assigned to $g(j) = i$. Then, j might delay other jobs with smaller predicted density which have been already assigned to i, i.e., are part of $M_i(j)$. Further, j might be delayed by jobs which have higher predicted density and are part of $M_i(j)$. Finally, j's completion time cannot be less than $\hat{r}_{ij} + \frac{p_j}{s_{ij}}$ due to the definition of the algorithm, and this value might be delayed further by $x_i(\hat{r}_{ij})$. In total, we conclude that the contribution of j to ALG is at most

$$w_j \left(\hat{r}_{ij} + x_i(\hat{r}_{ij}) + \frac{p_j}{s_{ij}} + \sum_{\substack{j' \in M_i(j) \\ \hat{\delta}_{ij'} \geq \hat{\delta}_{ij}}} \frac{p_{j'}}{s_{ij'}} \right) + \frac{p_j}{s_{ij}} \sum_{\substack{j' \in M_i(j) \\ \hat{\delta}_{ij'} < \hat{\delta}_{ij}}} w_{j'}.$$

This value is indeed at most Q_{ij}, because if at time $\hat{r}_{ij}$ some job k is being processed, it must be that $\hat{r}_{ik} \leq \hat{r}_{ij}$, and thus

$$x_i(\hat{r}_{ij}) \leq \frac{p_k}{s_{ik}} \leq \mu_1 \frac{p_k}{\hat{s}_{ik}} \leq \mu_1 \frac{\hat{r}_{ik}}{\theta} \leq \mu_1 \frac{\hat{r}_{ij}}{\theta}.$$

The statement then follows by summation of all jobs and the observation that this contribution only affects jobs that were handled before job j. □

Lemma A.2. $\sum_{i,t} \bar{b}_{it} = \frac{\mu}{\kappa} \text{ALG}$

Proof. Since we assumed that all release dates and processing times in J are integer multiples of κ, all all job completions occur at integer multiples of κ. Thus, $\sum_t \sum_{j \in U_i(\kappa \cdot t)} w_j = \frac{1}{\kappa} \sum_t \sum_{j \in U_i(t)} w_j$ for every machine i, and we conclude

$$\sum_{i,t} \bar{b}_{it} = \mu \sum_{i,t} \sum_{j \in U_i(\kappa \cdot t)} w_j = \frac{1}{\kappa} \sum_{i,t} \sum_{j \in U_i(t)} w_j = \frac{\mu}{\kappa} \cdot \text{ALG}.$$

□

These two lemmas give the following corollary.

Corollary A.3. $\sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} \geq \left(1 - \frac{\mu}{\kappa}\right) \cdot \text{ALG}.$

Second, we show that scaling the crafted duals makes them feasible for (NP-DLP).

Lemma A.4. *Assigning $a_j = \bar{a}_j / \lambda$ and $b_{it} = \bar{b}_{it} / \lambda$ gives a feasible solution for* (NP-DLP) *for a constant $\lambda > 0$ that satisfies $\lambda \geq 2\mu(2 + \theta)$ and $\lambda \geq \mu_1(\frac{1}{\theta} + \mu_2 \cdot \kappa)$.*

Proof. Since our defined variables are non-negative by definition, it suffices to show that this assignment satisfies (2). Fix a job j, a machine i and a time $t \geq r_j$. We assume that no new job arrives after j, since such a job may only increase $\bar{b}_{it}$ while $\bar{a}_j$ stays unchanged. We define a partition of $M_i(j)$ into high priority and low priority jobs with respect to j, and into completed and unfinished jobs with respect to time $\kappa \cdot t$:

- $H_U = \{j' \in M_i(j) : \hat{\delta}_{ij'} \geq \hat{\delta}_{ij} \wedge C_{j'} > \kappa \cdot t\}$ and $H_C = \{j' \in M_i(j) : \hat{\delta}_{ij'} \geq \hat{\delta}_{ij} \wedge C_{j'} \leq \kappa \cdot t\}$,
- $L_U = \{j' \in M_i(j) : \hat{\delta}_{ij'} < \hat{\delta}_{ij} \wedge C_{j'} > \kappa \cdot t\}$ and $L_C = \{j' \in M_i(j) : \hat{\delta}_{ij'} < \hat{\delta}_{ij} \wedge C_{j'} \leq \kappa \cdot t\}$.

We write $H = H_C \cup H_U$, $L = L_C \cup L_U$ and $\delta_{ij} = \frac{w_j s_{ij}}{p_j}$. Due to the choice of $g(j)$ in the algorithm, $\hat{Q}_{g(j)j} \leq \hat{Q}_{i'j}$ for every machine i'. Hence, we have $\bar{a}_j = Q_{g(j)j} \leq \mu_1 \cdot \hat{Q}_{g(j)j} \leq \mu_1 \cdot \hat{Q}_{ij}$, and using that,

$$\begin{aligned}
\frac{\bar{a}_j \cdot s_{ij}}{\lambda p_j} &\leq \mu_1 \frac{\hat{Q}_{ij} \cdot s_{ij}}{\lambda p_j} \\
&= \delta_{ij} \frac{\mu_1}{\lambda} \left(\hat{r}_{ij} + \frac{\hat{r}_{ij}}{\theta} + \frac{p_j}{\hat{s}_{ij}} + \sum_{j' \in H} \frac{p_{j'}}{\hat{s}_{ij'}} \right) + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in L} w_{j'} \\
&\leq \delta_{ij} \frac{\mu_1}{\lambda} \left(\left(1 + \frac{1}{\theta}\right) r_j + \sum_{j' \in H} \frac{p_{j'}}{\hat{s}_{ij'}} \right) + \mu_1 \frac{s_{ij} w_j}{\lambda p_j} (2 + \theta) \frac{p_j}{\hat{s}_{ij}} + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in L} w_{j'} \\
&\leq \delta_{ij} \frac{\mu_1}{\lambda} \left(\left(1 + \frac{1}{\theta}\right) r_j + \sum_{j' \in H} \frac{p_{j'}}{\hat{s}_{ij'}} \right) + \mu \frac{w_j}{\lambda} (2 + \theta) + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in L} w_{j'} \\
&\leq \delta_{ij} \frac{\mu_1}{\lambda} \left(\left(1 + \frac{1}{\theta}\right) r_j + \sum_{j' \in H} \frac{p_{j'}}{\hat{s}_{ij'}} \right) + \frac{w_j}{2} + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in L} w_{j'},
\end{aligned}$$

where the second inequality is due to $(1 + \frac{1}{\theta})\hat{r}_{ij} \leq (1 + \frac{1}{\theta}) r_j + (1 + \theta) \frac{p_j}{\hat{s}_{ij}}$, which follows from the definition of $\hat{r}_{ij}$, and the last inequality requires $\lambda \geq 2\mu(2 + \theta)$. Thus, asserting the dual constraint (2) reduces to proving

$$\delta_{ij} \frac{\mu_1}{\lambda} \left(\left(1 + \frac{1}{\theta}\right) r_j + \sum_{j' \in H} \frac{p_{j'}}{\hat{s}_{ij'}} \right) + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in L} w_{j'} \leq \delta_{ij} t + \frac{\bar{b}_{it}}{\lambda}.$$

To this end, first note that for all $j' \in L$ holds

$$w_{j'} \frac{\hat{s}_{ij'}}{p_{j'}} = \hat{\delta}_{ij'} < \hat{\delta}_{ij} = \frac{w_j \hat{s}_{ij}}{p_j} = \frac{\delta_{ij} \hat{s}_{ij}}{s_{ij}} \implies \frac{s_{ij}}{\delta_{ij} \hat{s}_{ij}} w_{j'} \leq \frac{\hat{s}_{ij'}}{p_{j'}}, \tag{3}$$

and for all $j' \in H$

$$\delta_{ij} \leq \mu_2 \cdot \hat{\delta}_{ij} \leq \mu_2 \cdot \hat{\delta}_{ij'} = \frac{w_{j'} \hat{s}_{ij'}}{p_{j'}} \implies \delta_{ij} \frac{p'_j}{\hat{s}_{ij'}} \leq w_{j'}. \tag{4}$$

Using these two inequalities gives

$$\begin{aligned}
&\delta_{ij} \frac{\mu_1}{\lambda} \left(\left(1 + \frac{1}{\theta}\right) r_j + \sum_{j' \in H_C} \frac{p_{j'}}{\hat{s}_{ij'}} + \sum_{j' \in H_U} \frac{p_{j'}}{\hat{s}_{ij'}} \right) + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in L_C} w_{j'} + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in L_U} w_{j'} \\
&= \delta_{ij} \frac{\mu_1}{\lambda} \left(\left(1 + \frac{1}{\theta}\right) r_j + \sum_{j' \in H_C} \frac{p_{j'}}{\hat{s}_{ij'}} + \frac{s_{ij}}{\delta_{ij} \hat{s}_{ij}} \sum_{j' \in L_C} w_{j'} \right) + \delta_{ij} \frac{\mu_1}{\lambda} \sum_{j' \in H_U} \frac{p_{j'}}{\hat{s}_{ij'}} + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in L_U} w_{j'} \\
&\leq \delta_{ij} \frac{\mu_1}{\lambda} \left(\left(1 + \frac{1}{\theta}\right) r_j + \sum_{j' \in H_C} \frac{p_{j'}}{\hat{s}_{ij'}} + \sum_{j' \in L_C} \frac{p_{j'}}{\hat{s}_{ij'}} \right) + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in H_U} w_{j'} + \frac{\mu_1}{\lambda} \frac{s_{ij}}{\hat{s}_{ij}} \sum_{j' \in L_U} w_{j'} \\
&\leq \delta_{ij} \frac{\mu_1}{\lambda} \left(\frac{r_j}{\theta} + \mu_2 \left(r_j + \sum_{j' \in M_i(j) : \kappa \cdot t \geq C_{j'}} \frac{p_{j'}}{s_{ij'}} \right) \right) + \frac{\mu_1}{\lambda} \mu_2 \sum_{j' \in M_i(j) : \kappa \cdot t < C_{j'}} w_{j'} \\
&\leq \delta_{ij} \frac{\mu_1}{\lambda} \left(\frac{t}{\theta} + \mu_2 \cdot \kappa \cdot t \right) + \frac{\mu}{\lambda} \sum_{j' \in U_i(\kappa \cdot t)} w_{j'} \\
&\leq \delta_{ij} t + \frac{\bar{b}_{it}}{\lambda}.
\end{aligned}$$

In the first inequality we use (4) and (3). In order to understand the third inequalty, first recall that $M_i(j)$ contains all jobs that are assigned to machine i but unstarted at time r_j. Thus, the total processing duration of these jobs that are completed within time $\kappa \cdot t$ can be at most $\kappa \cdot t - r_j$. The last inequality follows from $\lambda \geq \mu_1(\frac{1}{\theta} + \mu_2 \cdot \kappa)$ and the definition of $\bar{b}_{it}$. □

Proof of Theorem 2.6. We set $\kappa = \frac{23}{6}\mu$, $\theta = \frac{2}{3}$ and $\lambda = \frac{16}{3}\mu^2$. Then, weak duality, Corollary A.3 and Lemma A.4 imply

$$\text{OPT} \geq \sum_j a_j - \sum_{i,t} b_{it} = \frac{1}{\lambda}\left(\sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it}\right) = \left(\frac{1-\mu/\kappa}{\lambda}\right) \cdot \text{ALG}.$$

Since $\kappa > \mu$ and $\lambda > 0$, we conclude that

$$\text{ALG} \leq \frac{\frac{16}{3} \cdot \mu^2}{1 - \frac{6}{23}} \cdot \text{OPT} = \frac{368}{51} \cdot \mu^2 \cdot \text{OPT}.$$

□

A.2. Full Analysis of Proportional Fairness with Speed Predictions

This section contains the detailed analysis of PF with speed predictions, and thus the proof of Theorem 2.7. It is based on the analysis of the speed-aware PF given in (Im et al., 2018).

Theorem 2.7. *Algorithm 3 has a competitive ratio of at most* 108μ *for minimizing the total weighted completion time on unrelated machines with predicted speeds.*

Fix an instance and PF's schedule. Let $\kappa \geq 1$ and $0 < \lambda < 1$ be constants which we fix later. Recall that q_{jt} denotes the progress of job j at time t. For every t, consider the sorted (ascending) list Z^t composed of w_j copies of $\frac{q_{jt}}{p_j}$ for every $j \in U_t$. Note that Z^t has length W_t. We define ζ_t as the value at the index $\lfloor \lambda W_t \rfloor$ in Z^t.

We first state the KKT conditions with multipliers $\{\eta_{it}\}_i$ and $\{\theta_{jt}\}_{j \in J(t)}$ of the optimal solution $\{y_{ijt}\}_{i,j}$ of (CP$_t$) the algorithm uses at time t:

$$\frac{\hat{s}_{ij} w_j}{\sum_{i'} \hat{s}_{i'j} y_{i'jt}} \leq \theta_{jt} + \eta_{it} \quad \forall t, \forall i, \forall j \in J(t) \tag{5}$$

$$y_{ijt}\left(\frac{\hat{s}_{ij} w_j}{\sum_{i'} \hat{s}_{i'j} y_{i'jt}} - (\theta_{jt} + \eta_{it})\right) = 0 \quad \forall t, \forall i, \forall j \in J(t) \tag{6}$$

$$\theta_{jt}\left(\sum_i y_{ijt} - 1\right) = 0 \quad \forall t, \forall j \in J(t) \tag{7}$$

$$\eta_{it}\left(\sum_j y_{ijt} - 1\right) = 0 \quad \forall t, \forall i \tag{8}$$

$$\theta_{jt}, \eta_{it} \geq 0 \quad \forall t, \forall i, \forall j \in J(t) \tag{9}$$

We have the following dual assignment:

- $\bar{a}_j = \sum_{t'=0}^{C_j} \bar{a}_{jt'}$, where $\bar{a}_{jt'} = w_j \cdot \mathbb{1}\left[\frac{q_{jt'}}{p_j} \leq \zeta_{t'}\right]$, for every job j,
- $\bar{b}_{it} = \frac{1}{\kappa} \sum_{t' \geq t} \zeta_{t'} \eta_{it'}$ for every machine i and time t, and
- $\bar{c}_{jt} = \frac{1}{\kappa} \sum_{t'=t}^{C_j} \zeta_{t'} \theta_{jt'}$ for every job j and time $t \geq r_j$.

The following three lemmas will conclude that the dual objective value of this assignment is close the algorithm's objective value, and thus prove Lemma 2.8.

Lemma A.5. $\sum_j \bar{a}_j \geq \lambda \cdot \textsc{Alg}$

Proof. Consider a time t and the list Z^t. Observe that $\sum_{j \in U_t} \bar{a}_{jt}$ contains for every job j which satisfies $\frac{q_{jt}}{p_j} \leq \zeta_t$ its weight w_j. By the definitions of Z_t and ζ_t, we conclude that this is at least λW_t, i.e., $\sum_{j \in U_t} \bar{a}_{jt} \geq \lambda W_t$. The statement then follows by summing over all times t. □

Lemma A.6. *At any time t,* $\sum_i \eta_{it} + \sum_{j \in J(t)} \theta_{jt} \leq W_t$.

Proof. At any time t holds

$$\begin{aligned}
\sum_i \eta_{it} + \sum_{j \in J(t)} \theta_{jt} &= \left(\sum_i \eta_{it} \sum_{j \in J(t)} y_{ijt} \right) + \left(\sum_{j \in J(t)} \theta_{jt} \sum_i y_{ijt} \right) \\
&= \sum_i \sum_{j \in J(t)} y_{ijt} (\eta_{it} + \theta_{jt}) \\
&= \sum_i \sum_{j \in J(t)} y_{ijt} \frac{\hat{s}_{ij} w_j}{\sum_{i'} \hat{s}_{i'j} y_{i'jt}} \\
&= \sum_{j \in J(t)} \sum_i \hat{s}_{ij} y_{ijt} \frac{w_j}{\sum_{i'} \hat{s}_{i'j} y_{i'jt}} = \sum_{j \in J(t)} w_j \leq W_t.
\end{aligned}$$

The first equality is due to (7) and (8), and the third equality due to (6). □

Lemma A.7. *At any time t,* $\sum_i \bar{b}_{it} + \sum_{j \in J: t \geq r_j} \bar{c}_{jt} \leq \frac{4}{(1-\lambda)\kappa} W_t$.

Proof. Fix a time t. Observe that for every $t' \geq t$ the definitions of $Z_{t'}$ and $\zeta_{t'}$ imply $(1-\lambda) W_{t'} \leq \sum_{j \in U_{t'}} w_j \cdot \mathbb{1}\left[\frac{q_{jt'}}{p_j} \geq \zeta_{t'}\right]$. Thus,

$$\zeta_{t'} \cdot (1-\lambda) W_{t'} \leq \sum_{j \in U_{t'}} w_j \cdot \zeta_{t'} \cdot \mathbb{1}\left[\frac{q_{jt'}}{p_j} \geq \zeta_{t'}\right] \leq \sum_{j \in U_{t'}} w_j \cdot \frac{q_{jt'}}{p_j} \cdot \mathbb{1}\left[\frac{q_{jt'}}{p_j} \geq \zeta_{t'}\right]. \tag{10}$$

We define a partition $\{M_k\}_{k \geq 1}$ of the time interval $[t, \infty)$ such that the total weight of unfinished jobs at all times during M_k is part of $(\frac{1}{2^k} W_t, \frac{1}{2^{k-1}} W_t]$. Fix a $k \geq 1$. Rearranging (10) and estimating the total weight of unfinished jobs in a partition against both its upper and lower bound yields

$$\begin{aligned}
\sum_{t' \in M_k} \zeta_{t'} &\leq \sum_{t' \in M_k} \frac{1}{1-\lambda} \sum_{j \in U_{t'}} \frac{w_j}{W_{t'}} \cdot \frac{q_{jt'}}{p_j} \cdot \mathbb{1}\left[\frac{q_{jt'}}{p_j} \geq \zeta_{t'}\right] \\
&\leq \frac{1}{1-\lambda} \sum_{t' \in M_k} \sum_{j \in U_{t'}} \frac{w_j}{W_{t'}} \cdot \frac{q_{jt'}}{p_j} \\
&\leq \frac{2^k}{(1-\lambda) W_t} \sum_{t' \in M_k} \sum_{j \in U_{t'}} w_j \cdot \frac{q_{jt'}}{p_j} \\
&\leq \frac{2^k \cdot W_t}{(1-\lambda) W_t \cdot 2^{k-1}} = \frac{2}{1-\lambda}.
\end{aligned}$$

The definitions of $\bar{b}_{it}$ and $\bar{c}_{jt}$ and Lemma A.6 imply

$$\begin{aligned}
\sum_i \bar{b}_{it} + \sum_{j \in J: t \geq r_j} \bar{c}_{jt} &= \left(\sum_i \frac{1}{\kappa} \sum_{t' \geq t} \eta_{it'} \cdot \zeta_{t'} \right) + \left(\sum_{j \in J: t \geq r_j} \frac{1}{\kappa} \sum_{t'=t}^{C_j} \theta_{jt'} \cdot \zeta_{t'} \right) \\
&= \frac{1}{\kappa} \sum_{t' \geq t} \zeta_{t'} \left(\sum_i \eta_{it'} + \sum_{j \in J(t')} \theta_{jt'} \right) \leq \frac{1}{\kappa} \sum_{t' \geq t} \zeta_{t'} W_{t'}.
\end{aligned}$$

By dividing the time after t into the partition $\{M_k\}_{k\geq 1}$ and using our bound on $\sum_{t'\in M_k}\zeta_{t'}$, we conclude that this is at most

$$\frac{1}{\kappa}\sum_{k\geq 1}\sum_{t'\in M_k}\zeta_{t'}W_{t'} \leq \frac{1}{\kappa}\sum_{k\geq 1}\frac{W_t}{2^{k-1}}\sum_{t'\in M_k}\zeta_{t'} \leq \frac{2}{\kappa(1-\lambda)}W_t\sum_{k\geq 1}\frac{1}{2^{k-1}} \leq \frac{4}{\kappa(1-\lambda)}W_t.$$

The last inequality uses a bound on the geometric series. □

Lemma 2.8. $(\lambda - \frac{4}{(1-\lambda)\kappa})\textsc{Alg} \leq \sum_j \bar{a}_j - \sum_{i,t}\bar{b}_{it} - \sum_{j,t\geq r_j}\bar{c}_{jt}$

Proof. Follows directly from Lemmas A.5 and A.7. □

Lemma 2.9. *Assigning $a_j = \bar{a}_j$, $b_{it} = \bar{b}_{it}$ and $c_{jt} = \bar{c}_{jt}$ is feasible for (DLP$_\alpha$) if $\alpha = \kappa\mu$.*

Proof. First observe that for every t and j holds

$$\sum_i \hat{s}_{ij}y_{ijt} \leq \mu_1\sum_i s_{ij}y_{ijt} = \mu_1\cdot q_{jt}. \tag{11}$$

Fix a job j, a machine i and a time $t \geq r_j$.

$$\begin{aligned}
\frac{\bar{a}_j s_{ij}}{p_j} - w_j\cdot\frac{t\cdot s_{ij}}{p_j} &\leq s_{ij}\cdot\sum_{t'=t}^{C_j}\frac{\bar{a}_{jt'}}{p_j}\\
&= s_{ij}\cdot\sum_{t'=t}^{C_j}\frac{w_j}{p_j}\cdot\mathbb{1}\left[\frac{q_{jt'}}{p_j}\leq\zeta_{t'}\right]\\
&= s_{ij}\cdot\sum_{t'=t}^{C_j}\frac{w_j}{\sum_{i'}\hat{s}_{i'j}y_{i'jt'}}\cdot\frac{\sum_{i'}\hat{s}_{i'j}y_{i'jt'}}{q_{jt'}}\cdot\frac{q_{jt'}}{p_j}\cdot\mathbb{1}\left[\frac{q_{jt'}}{p_j}\leq\zeta_{t'}\right]\\
&\leq \mu_1\cdot\mu_2\cdot\sum_{t'=t}^{C_j}\frac{\hat{s}_{ij}w_j}{\sum_{i'}\hat{s}_{i'j}y_{i'jt'}}\cdot\frac{q_{jt'}}{p_j}\cdot\mathbb{1}\left[\frac{q_{jt'}}{p_j}\leq\zeta_{t'}\right]\\
&\leq \mu\cdot\sum_{t'=t}^{C_j}(\eta_{it'}+\theta_{jt'})\cdot\frac{q_{jt'}}{p_j}\cdot\mathbb{1}\left[\frac{q_{jt'}}{p_j}\leq\zeta_{t'}\right]\\
&\leq \mu\cdot\sum_{t'=t}^{C_j}(\eta_{it'}+\theta_{jt'})\cdot\zeta_{t'}\\
&\leq \mu\kappa\cdot\frac{1}{\kappa}\left(\sum_{t'\geq t}\eta_{it'}\cdot\zeta_{t'}\right)+\mu\kappa\cdot\left(\frac{1}{\kappa}\sum_{t'=t}^{C_j}\theta_{jt'}\cdot\zeta_{t'}\right)\\
&= \mu\kappa\cdot\bar{b}_{it}+\mu\kappa\cdot\bar{c}_{jt}.
\end{aligned}$$

The second inequality uses (11) and the third inequality uses (5). Since $\alpha = \kappa\mu$, this dual assignment indeed satisfies the constraint of (DLP$_\alpha$). □

Proof of Theorem 2.7. Weak duality, Lemma 2.8 and Lemma 2.9 imply

$$\kappa\mu\cdot\textsc{Opt} \geq \textsc{Opt}_{\kappa\mu} \geq \sum_j\bar{a}_j - \sum_{i,t}\bar{b}_{it} - \sum_{j,t\geq r_j}\bar{c}_{jt} \geq \left(\lambda - \frac{4}{(1-\lambda)\kappa}\right)\cdot\textsc{Alg}.$$

Setting $\kappa = 36$ and $\lambda = \frac{2}{3}$ implies $\textsc{Alg} \leq 108\mu\cdot\textsc{Opt}$. □

B. Details on Algorithms for Speed-Ordered Machines

Theorem 3.1. *Any non-migratory speed-oblivious algorithm has a competitive ratio of at least $\Omega(m)$ for minimizing the total completion time on m speed-ordered machines, even if it is clairvoyant and the machines are related.*

Proof. Consider the execution of some algorithm on an instance of n jobs with unit-weights and with processing requirements equal to n^2m and $s_1 = n^2m$. If at some point in time, the algorithm starts a job on machines $2, \ldots, m$, the adversary sets $s_2 = \ldots = s_m = 1$ to enforce an objective value of at least $\Omega(n^2m)$, while scheduling all jobs on the first machine gives an objective value of at most $\mathcal{O}(n^2)$. If this does not happen, the algorithm must have scheduled all jobs on the first machine. But then the adversary sets $s_2 = \ldots = s_m = n^2m$ and achieves an objective value of $\mathcal{O}(\frac{n^2}{m})$ by distributing the jobs evenly to all machines, while the algorithm has an objective value of $\Omega(n^2)$. □

B.1. Full Analysis of Maximum Density for Speed-Ordered Related Machines

This section is devoted to the proof of Theorem 3.2, which we firstly restate:

Theorem 3.2. *Algorithm 4 has a competitive ratio of at most 8 for minimizing the total weighted completion time on speed-ordered related machines.*

We use a dual-fitting analysis based on (DLP$_\alpha$) to prove this theorem. Fix an instance and the algorithm's schedule, and observe that the algorithm ensures at every time t that M_t is a matching between alive jobs and machines. Recall that for related machines, $s_i = s_{ij}$ for every job j and every machine i.

Let $\kappa \geq 1$ be a constant. We define for every machine i and any time t

$$\beta_{it} = \begin{cases} \frac{w_j s_i}{p_j} & \text{if } i \text{ is matched to } j \in J(t) \text{ in } M_t \\ 0 & \text{otherwise,} \end{cases}$$

and for every job j and any time t

$$\gamma_{jt} = \begin{cases} \frac{w_j s_i}{p_j} & \text{if } j \text{ is matched to } i \in I \text{ in } M_t \\ 0 & \text{otherwise.} \end{cases}$$

Using these values, we have the following dual assignment:

- $\bar{a}_j = w_j C_j$ for every job j,
- $\bar{b}_{it} = \frac{1}{\kappa} \sum_{t' \geq t} \beta_{it'}$ for every machine i and time t, and
- $\bar{c}_{jt} = \frac{1}{\kappa} \sum_{t' \geq t} \gamma_{jt'}$ for every job j and time $t \geq r_j$.

We first observe that the dual objective of this assignment is close to algorithm's objective. The proof works analogous to the proof of Lemma 2.4.

Lemma B.1. $(1 - \frac{2}{\kappa})\textsc{Alg} \leq \sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} - \sum_{j, t \geq r_j} \bar{c}_{jt}$

Lemma B.2. *Assigning $a_j = \bar{a}_j$, $b_{it} = \bar{b}_{it}$ and $c_{jt} = \bar{c}_{jt}$ is feasible for (DLP$_\alpha$) if $\alpha = \kappa$ and $s_i = s_{ij}$ for every job j and every machine i.*

Proof. Since the dual assignment is clearly non-negative, we now show that it satisfies the dual constraint. Let $i \in I, j \in J$ and $t \geq r_j$. We first observe that

$$\bar{a}_j \frac{s_i}{p_j} - w_j t \frac{s_i}{p_j} \leq \sum_{t'=t}^{C_j} \frac{w_j s_i}{p_j}.$$

Using $\alpha = \kappa$, it remains to validate for every $t \leq t' \leq C_j$ that $\frac{w_j s_i}{p_j} \leq \beta_{it'} + \gamma_{jt'}$. We distinguish five cases:

(i) If $(i,j) \in M_{t'}$, then $\frac{w_j s_i}{p_j} = \beta_{it'} = \gamma_{jt'}$.

(ii) If $(i,j') \in M_{t'}$ and $(i',j) \in M_{t'}$ s.t. $i \neq i'$, we have two cases. If $i < i'$, it must be that $\sigma_{t'}(j') < \sigma_{t'}(j)$ and, thus, $\frac{w_{j'}}{p_{j'}} \geq \frac{w_j}{p_j}$. But then, $\frac{w_j s_i}{p_j} \leq \frac{w_{j'} s_i}{p_{j'}}$. Otherwise, that is, $i > i'$, we know by the speed order that $s_i \leq s_{i'}$, and, thus, $\frac{w_j s_i}{p_j} \leq \frac{w_j s_{i'}}{p_j}$. Put together,

$$\frac{w_j s_i}{p_j} \leq \frac{w_{j'} s_i}{p_{j'}} + \frac{w_j s_{i'}}{p_j} = \beta_{it'} + \gamma_{jt'}.$$

(iii) If $(i',j) \in M_{t'}$ and i is not matched in $M_{t'}$, it follows $i' < i$, which gives $\frac{w_j s_i}{p_j} \leq \frac{w_j s_{i'}}{p_j} = \gamma_{jt'}$.

(iv) If $(i,j') \in M_{t'}$ and j is not matched in $M_{t'}$, it follows $\sigma_{t'}(j') < \sigma_{t'}(j)$, and hence $\frac{w_j}{p_j} \leq \frac{w_{j'}}{p_{j'}}$. This immediately concludes $\frac{w_j s_i}{p_j} \leq \frac{w_{j'} s_i}{p_{j'}} = \beta_{it'}$.

(v) The case where both i and j are unmatched in $M_{t'}$ contradicts the definition of $M_{t'}$ in Algorithm 4.

□

Proof of Theorem 3.2. Weak duality, Lemma B.2 and Lemma B.1 imply

$$\kappa \cdot \textsc{Opt} \geq \textsc{Opt}_\kappa \geq \sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} - \sum_{j,t \geq r_j} \bar{c}_{jt} \geq \left(1 - \frac{2}{\kappa}\right) \cdot \textsc{Alg}.$$

Using $\kappa = 4$ concludes $\textsc{Alg} \leq \frac{\kappa}{1-2/\kappa} \cdot \textsc{Opt} = 8 \cdot \textsc{Opt}$. □

We finally observe that Algorithm 4 cannot achieve a good competitive ratio if speeds are job-dependent.

Lemma B.3. *Algorithm 4 has a competitive ratio of at least $\Omega(n)$ for minimizing the total weighted completion time on speed-ordered unrelated machines, even on two machines and if $w_j = 1$ for all jobs j.*

Proof. Let $0 < \epsilon < 1$. Consider an instance composed of n jobs and 2 machines, where $w_j = 1$ for all jobs j, $p_1 = 1$ and $p_j = 1 + \epsilon$ for all $2 \leq j \leq n$. The processing speeds are given by $s_{11} = s_{21} = \epsilon$, and $s_{1j} = 1$ and $s_{2j} = \epsilon$ for all $2 \leq j \leq n$. Note that the machines are speed-ordered. Algorithm 4 completes at time $\frac{1}{\epsilon}$ job 1 on machine 1 before any other job. Thus, $\textsc{Alg} \geq \frac{n}{\epsilon}$. Another solution is to schedule jobs $2, \ldots, n$ on machine 1, and job 1 on machine 2, giving an objective of at most $n^2 + \frac{1}{\epsilon}$. For $\epsilon < n^{-2}$, this concludes that $\frac{\textsc{Alg}}{\textsc{Opt}} \geq \Omega(n)$. □

B.2. Round Robin for Speed-Ordered Machines

B.2.1. Missing Details for the Analysis for Unrelated Machines

This section contains missing details for the proof of Theorem 3.3, which we firstly restate:

Theorem 3.3. *Algorithm 5 has a competitive ratio of at most $\mathcal{O}(\log(\min\{n,m\}))$ for minimizing the total completion time on speed-ordered unrelated machines.*

Proposition B.4. *At any time t, $\sum_i \beta_{it} \leq \mathcal{O}(\log(\min\{n,m\})) \cdot |U_t|$.*

Proof. At any time t,

$$\sum_{i \in I} \beta_{it} = \sum_{i=1}^{m_t} \frac{1}{i} \cdot |J(t)| \leq |U_t| \sum_{i=1}^{m_t} \frac{1}{i} \leq \mathcal{O}(\log(\min\{n,m\})) \cdot |U_t|,$$

where in the last inequality we use that $m_t = \min\{m, |J(t)|\} \leq \min\{m,n\}$. □

Proposition B.5. *At any time t, $\sum_{j \in J: r_j \geq t} \gamma_{jt} \leq |U_t|$.*

Lemma B.6. $\sum_j \bar{a}_j \geq \frac{1}{2} \cdot \textsc{Alg}$.

Proof. Analogous to the proof of Lemma A.5. □

Lemma B.7. *At any time t,* $\sum_i \bar{b}_{it} \leq \mathcal{O}(1) \cdot |U_t|$.

Proof. Analogous to the proof of Lemma A.7 when using Proposition B.4 and the fact that $\kappa = \Theta(\log(\min\{m, n\}))$. □

Lemma B.8. *At any time t,* $\sum_{j \in J: r_j \geq t} \bar{c}_{jt} \leq \mathcal{O}(1) \cdot |U_t|$.

Proof. Analogous to the proof of Lemma A.7 when using Proposition B.5. □

Observe that Lemma B.6, Lemma B.7 and Lemma B.8 imply Lemma 3.4. It remains the proof of Theorem 3.3:

Proof of Theorem 3.3. Weak duality, Lemma 3.4 and Lemma 3.5 imply

$$\kappa \cdot \textsc{Opt} \geq \textsc{Opt}_\kappa \geq \sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} - \sum_{j, t \geq r_j} \bar{c}_{jt} \geq \Omega(1) \cdot \textsc{Alg}.$$

We conclude the proof by noting that $\kappa = \Theta(\log(\min\{m, n\}))$. □

We finally observe that this competitive ratio is tight for Algorithm 5 on speed-ordered unrelated machines.

Lemma B.9. *Algorithm 4 has a competitive ratio of at least* $\Omega(\log(\min\{n, m\}))$ *for minimizing the total completion time on speed-ordered unrelated machines, even if processing speeds are exclusively from* $\{0, 1\}$.

Proof. Consider an instance of m unit-sized jobs $[m]$ and m machines $[m]$. Every job $j \in [m]$ has on machine $i \in [m]$ a processing speed equal to $s_{ij} = \mathbb{1}[i \leq m - j + 1]$. First observe that $\textsc{Opt} \leq m$, because we can process and complete every job $j \in [m]$ excusively on machine $m - j + 1$ at time 1. We now calculate the algorithm's objective value. To this end, we argue that in the algorithm's schedule holds $C_j = 1 + \sum_{i=1}^{j-1} \frac{1}{m-i+1}$ for every job j. Then, $\textsc{Alg} = \sum_{j=1}^{m} C_j = \Omega(m \log m)$ concludes the statement.

We first observe that $C_1 = 1$, because job 1 receives in interval $I_1 = [0, C_1)$ on every machine a rate equal to $\frac{1}{m}$. We now argue iteratively for $j = 2, \ldots, m$ that $C_j = 1 + \sum_{i=1}^{j-1} \frac{1}{m-i+1}$. Consequently, in interval $I_j = [C_{j-1}, C_j)$ must be exactly jobs $j, \ldots, m$ alive. Fix a job j with $2 \leq j \leq m$ and let $2 \leq i \leq j$. Since j receives progress on exactly $m - j + 1$ machines, there are $m - i + 1$ alive jobs in I_i, and I_i has length $\frac{1}{m-i+2}$, its total progress in I_i is equal to $\frac{m-j+1}{(m-i+1)(m-i+2)}$. Further, j's progress is equal to $\frac{m-j+1}{m}$ in I_1. Summing over all intervals I_i with $1 \leq i \leq j$ concludes that j's progress until the end of I_j is equal to

$$\frac{m - j + 1}{m} + \sum_{i=2}^{j} \frac{m - j + 1}{(m - i + 1)(m - i + 2)} = 1,$$

asserting that $1 + \sum_{i=1}^{j-1} \frac{1}{m-i+1}$ is indeed j's competion time in the algorithm's schedule. □

B.2.2. Full Analysis of Round Robin for Speed-Ordered Related Machines

Theorem B.10. *Algorithm 5 has a competitive ratio of at most 216 for minimizing the total completion time on speed-ordered related machines.*

We prove this theorem using a dual-fitting proof based on (DLP$_\alpha$), where $w_j = 1$ and $s_i = s_{ij}$ for every job j and every machine i. Fix an instance and the algorithm's schedule. For every time t we write $m_t = \min\{m, |J(t)|\}$. We define for every machine i and any time t

$$\beta_{it} = \frac{s_i}{\sum_{\ell=1}^{m_t} s_\ell} \cdot |J(t)| \cdot \mathbb{1}[i \leq |J(t)|],$$

and $\gamma_{jt} = \mathbb{1}[j \in J(t)]$ for every job j and any time t.

Observe the following bounds when summing up these values:

Proposition B.11. *At any time t,* $\sum_i \beta_{it} \leq |U_t|$.

Proposition B.12. *At any time t,* $\sum_{j \in J(t)} \gamma_{jt} \leq |U_t|$.

Let $\kappa \geq 1$ and $0 < \lambda < 1$ be constants. For every t, consider the sorted (ascending) list Z^t composed of values $\frac{q_{jt}}{p_j}$ for every $j \in U_t$. We define ζ_t as the value at the index $\lfloor \lambda |U_t| \rfloor$ in Z^t. Consider the following duals:

- $\bar{a}_j = \sum_{t'=0}^{C_j} \mathbb{1}\left[\frac{q_{jt'}}{p_j} \leq \zeta_{t'}\right]$ for every job j,
- $\bar{b}_{it} = \frac{1}{\kappa} \sum_{t' \geq t} \beta_{it'} \zeta_{t'}$ for every i and t, and
- $\bar{c}_{jt} = \frac{1}{\kappa} \sum_{t' \geq t} \gamma_{jt'} \zeta_{t'}$ for every j and $t \geq r_j$.

Lemma B.13. $\sum_j \bar{a}_j \geq \lambda \cdot \text{ALG}$.

Proof. Analogous to the proof of Lemma A.5. □

Lemma B.14. *At any time t, $\sum_i \bar{b}_{it} \leq \frac{4}{(1-\lambda)\kappa} |U_t|$.*

Proof. Analogous to the proof of Lemma A.7 when using Proposition B.11. □

Lemma B.15. *At any time t, $\sum_{j \in J: r_j \geq t} \bar{c}_{jt} \leq \frac{4}{(1-\lambda)\kappa} |U_t|$.*

Proof. Analogous to the proof of Lemma A.7 when using Proposition B.12. □

Lemmas B.13 to B.15 conclude the following bound between ALG and the objective value of the crafted duals.

Lemma B.16. $(\lambda - \frac{8}{(1-\lambda)\kappa})\text{ALG} \leq \sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} - \sum_{j,t \geq r_j} \bar{c}_{jt}$

We finally prove that the crafted duals are feasible under certain conditions.

Lemma B.17. *Assigning $a_j = \bar{a}_j$, $b_{it} = \bar{b}_{it}$ and $c_{jt} = \bar{c}_{jt}$ is feasible for (DLP$_\alpha$) if $\alpha = \kappa$ and $s_i = s_{ij}$ for all machines i and jobs j.*

Proof. First observe that the dual assignment is non-negative. Let $i \in I, j \in J$ and $t \geq r_j$. Since the rates of Algorithm 5 imply $q_{jt} = \sum_{\ell=1}^{m_t} \frac{s_\ell}{|J(t)|}$, we have

$$\frac{\bar{a}_j s_i}{p_j} - \frac{s_i \cdot t}{p_j} \leq \sum_{t'=t}^{C_j} \frac{s_i}{p_j} \cdot \mathbb{1}\left[\frac{q_{jt'}}{p_j} \leq \zeta_{t'}\right] = \sum_{t'=t}^{C_j} \frac{s_i}{q_{jt'}} \cdot \frac{q_{jt'}}{p_j} \cdot \mathbb{1}\left[\frac{q_{jt'}}{p_j} \leq \zeta_{t'}\right] \leq \sum_{t'=t}^{C_j} \frac{s_i}{\sum_{\ell=1}^{m_{t'}} \frac{s_\ell}{|J(t')|}} \cdot \zeta_{t'} \tag{12}$$

Consider any time t' with $t \leq t' \leq C_j$. If $i \leq |J(t')|$, the definition of $\beta_{it'}$ yields

$$\frac{s_i}{\sum_{\ell=1}^{m_{t'}} s_\ell} \cdot |J(t')| \cdot \zeta_{t'} = \beta_{it'} \cdot \zeta_{t'}.$$

Otherwise, $i > |J(t')|$, the fact that $s_1 \geq \ldots \geq s_m$ implies $\sum_{\ell=1}^{m_{t'}} s_\ell \geq \sum_{\ell=1}^{|J(t')|} s_\ell \geq |J(t')| \cdot s_i$, and thus

$$\frac{s_i}{\sum_{\ell=1}^{m_{t'}} s_\ell} \cdot |J(t')| \cdot \zeta_{t'} \leq \frac{|J(t')|}{|J(t')|} \cdot \zeta_{t'} = \gamma_{jt'} \cdot \zeta_{t'},$$

because $t' \leq C_j$. Put together, (12) is at most

$$\sum_{t'=t}^{C_j} \beta_{it'} \zeta_{t'} + \sum_{t'=t}^{C_j} \gamma_{jt'} \zeta_{t'} \leq \kappa(\bar{b}_{it} + \bar{c}_{jt}),$$

which verifies the dual constraint. □

Proof of Theorem B.10. Weak duality, Lemma B.16 and Lemma B.17 imply

$$\kappa \cdot \text{OPT} \geq \text{OPT}_\kappa \geq \sum_j \bar{a}_j - \sum_{i,t} \bar{b}_{it} - \sum_{j,t \geq r_j} \bar{c}_{jt} \geq \left(\lambda - \frac{8}{(1-\lambda)\kappa}\right) \cdot \text{ALG}.$$

Setting $\kappa = 72$ and $\lambda = \frac{2}{3}$ concludes $\text{ALG} \leq 216 \cdot \text{OPT}$. □

C. Further Details on Experimental Results

C.1. The Iterative Greedy Algorithm

Algorithm 6 Iterative Greedy

Require: time t, speed predictions $\{\hat{s}_{ij}\}$
1: $I' \leftarrow I, J' \leftarrow J(t)$
2: **while** $I' \neq \emptyset \wedge J' \neq \emptyset$ **do**
3: $(i,j) = \arg\max_{i \in I', j \in J'} w_j \hat{s}_{ij}$
4: $I' \leftarrow I' \setminus \{i\}, J' \leftarrow J' \setminus \{j\}$
5: Schedule job j on machine i with rate $y_{ijt} = 1$ at time t.
6: **end while**

Lemma C.1. *Algorithm 6 has a competitive ratio of at least $\Omega(n)$ for minimizing the total completion time on unrelated machines, even if $s_{ij} = \hat{s}_{ij}$ for all jobs j and machines i.*

Proof. Let $\epsilon > 0$ and $n > m \geq 2$ such that $\frac{n-1}{m-1}$ is an integer. Consider a unit-weight instance of one job with $p_1 = \frac{n-1}{m-1}$, $s_{11} = 1+\epsilon$ and $s_{i1} = 1$ for $2 \leq i \leq m$, and $n-1$ jobs with $p_j = \epsilon$ and $s_{1j} = 1$ and $s_{ij} = \epsilon$ for $2 \leq j \leq n, 2 \leq i \leq m$. Algorithm 6 first schedules job 1 on machine 1, and the $n-1$ others on the remaining $m-1$ machines. Since the completion time of job 1 is equal to $\frac{n-1}{(1+\epsilon)(m-1)}$, jobs $2, \ldots, n$ will complete at time at least $\frac{n-1}{m-1}$ only on machines $2, \ldots, m$ if $\epsilon < \frac{m}{n-m-1}$, hence this allocation will remain until the end of the instance. This implies a total completion time of $\Omega(\frac{n^2}{m})$ for jobs $2, \ldots, n$. Another solution is to schedule all jobs $2, \ldots, n$ on machine 1 with a total completion time of at most $\mathcal{O}(\epsilon n^2)$, and job 1 latest at time $\mathcal{O}(\frac{n}{m})$ on any other machine. This implies that Algorithm 6 has a competitive ratio of at least $\Omega(n)$. □

C.2. Implementation Details

We implemented the schedulers as separate applications running in *userspace*, scheduling jobs via Linux *affinity masks*, which indicate for each process on which core it may be executed. The schedulers compute a schedule every 2 s based on the currently active jobs and their (predicted) characteristics. The schedulers are provided with the process IDs of the tasks in the workload, potentially along with predictions, and only manage these processes via affinity masks. Other processes may run on any core, but their load is negligible.

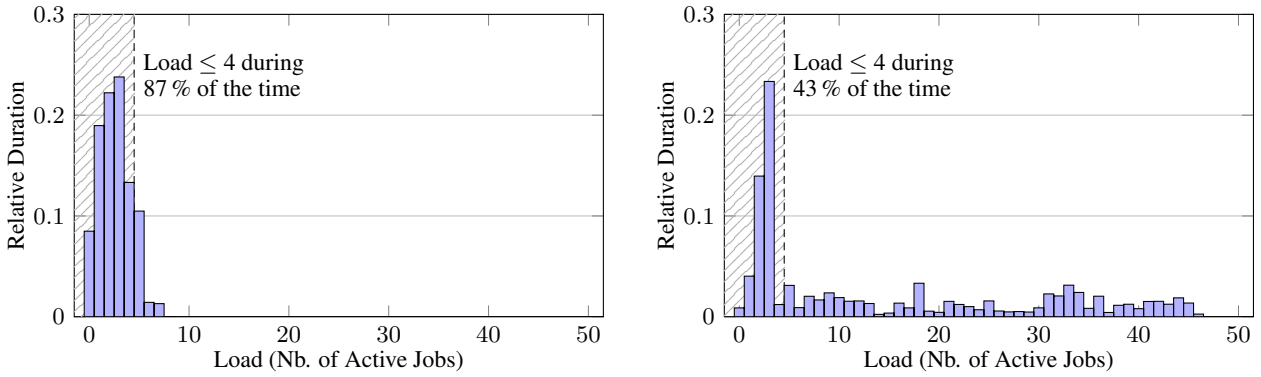
We use *native* input set for the *PARSEC-3.0* jobs. For the *SPLASH-3* jobs, we use both the *large* input set and custom input sizes to study the impact of different input data (see Figure 1). The *Polybench* jobs use their standard hard-coded inputs. We discard jobs that execute for less than 30 s on a *big* core to reduce measurement noise, and discard jobs that use more than 512 MB RAM because the *HiKey 970* board has only 6 GB RAM and Android does not support swap. Overall, this results in 43 combinations of jobs and input data, i.e., some jobs are repeated in the workloads.

C.3. Further Analysis of the Hardware Results

Figure 3 shows the distribution of system load during the experiments with speed-ordered Round Robin (Algorithm 5). At low job arrival rate (1 task/min), the system load is ≤ 4 during 87 % of the time. This means that during the majority of the time, it is possible to only use the *big* cores, explaining why speed predictions or clairvoyance bring little benefit over the speed-ordered setting as in Figure 2a. In contrast, the system load is ≤ 4 during 43 % of the time at a high job arrival rate (4 tasks/min), reaching up to 46. Accurate speed and job volume predictions are much more beneficial in this case, explaining the larger differences between algorithms in Figure 2b.

C.4. Simulation Experiments

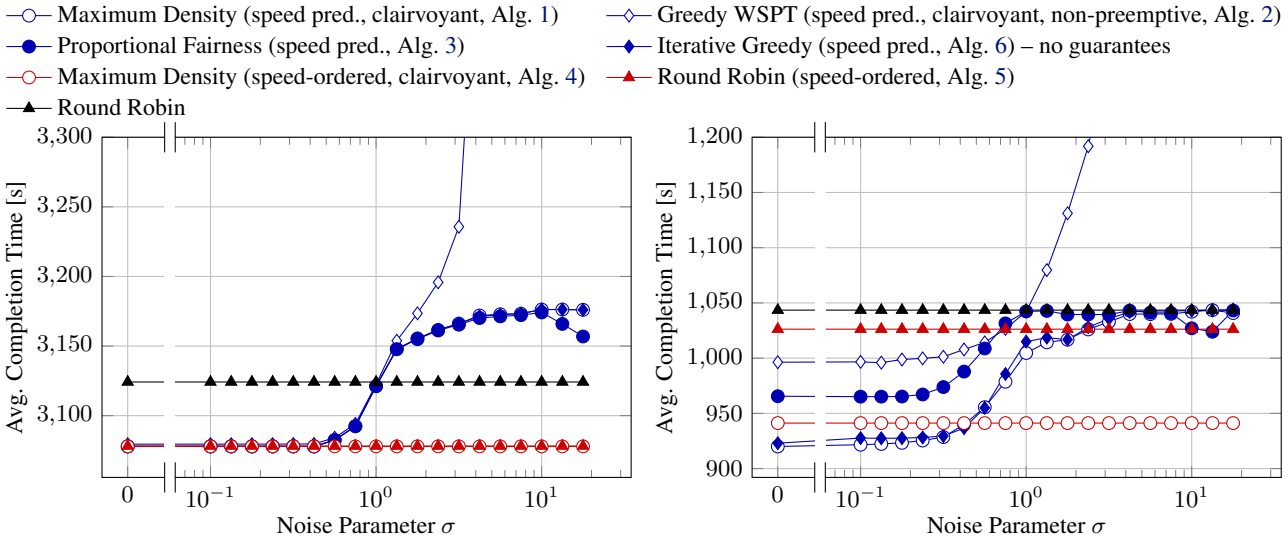
The experiments on real hardware are slow, hence we can only study a limited set of workload scenarios. We therefore additionally test many different scenarios in synthetic experiments in simulation. These synthetic experiments also model an 8-core processor. We create 20 random workloads with 100 synthetic jobs, whose arrival times are drawn from a Poisson distribution and with random characteristics: 4 *LITTLE* cores with speed 1, 4 *big* cores with job-dependent speeds from $\mathcal{U}(2,6)$, and $p_j \sim \mathcal{U}(60,600)$. Speed predictions are same as in the hardware experiments, i.e., $\hat{s}_{ij} = s_{ij} \cdot y_{ij}$.



(a) Low load: 1 task / min.

(b) High load: 4 tasks / min.

Figure 3. Distribution of the system load with speed-ordered Round Robin (Algorithm 5).



(a) Low load: 1 task / min.

(b) High load: 4 tasks / min.

Figure 4. Synthetic experiments. The experiments are each repeated 20 times with different random workloads.

Figure 4 shows the results of the synthetic experiments, including the fractional schedulers Greedy WSPT and PF. Unlike the real experiments, we are not restricted to a single workload and instead run 20 different workloads and plot the average results. Inaccurate speed predictions in Greedy WSPT result in large idle times, greatly deteriorating the performance. PF performs similar to or worse than Maximum Density, depending on the system load. The other algorithms perform similar to the experiments on the real platform, confirming that the results are not depending on a specific workload.