Taming Tardiness on Parallel Machines: Online Scheduling with Limited Job Information

Shaik Mohammed Salman*, Alessandro Vittorio Papadopoulos*, Saad Mubeen*, and Thomas Nolte*
*Mälardalen University,, Västerås, Sweden

***Abstract*—We consider the problem of scheduling n jobs on $m \geq 2$ parallel machines in online settings with the objective of minimizing total tardiness. Since no bounded competitive algorithms exist to minimize the general problem of weighted total tardiness of the form $\sum w_j T_j$, we consider an objective of the form $\sum w_j(T_j + d_j)$, where w_j, T_j, and d_j are the weight, tardiness, and deadline of each job, respectively and develop competitive algorithms dependent on jobs' processing times.**

I. Introduction

We consider an online scheduling problem where n jobs arrive at arbitrary times and should be completed before or close to their deadlines on a set of m parallel machines. At any given time, a job can be scheduled on at most one machine. All jobs that arrive must run to completion. They cannot be abandoned or rejected. Preemption and possibly migration are allowed. A natural objective for this problem is to minimize tardiness (T_j), which is the difference between the completion time of a job and its deadline if the job completes after its deadline. A generalization of this objective is to minimize the weighted total tardiness. In this context, each job has a deadline constraint and a weight. In Graham's $\alpha|\beta|\epsilon$ notation, where α is the machine environment, β describes the job characteristics and constraints, and ϵ specifies the objective function, these settings can be described as the online variants of $P|r_j, p_j, d_j, pmtn|\sum w_j T_j$ and $R|r_j, p_j, d_j, pmtn|\sum w_j T_j$ where r_j, p_j and d_j denote release time, processing time and deadline of each job j. P and R represent identical and unrelated machines' settings, respectively.

This problem is well-known to be NP-hard even on a single machine. Due to the inherent difficulty in developing competitive algorithms for this problem, we consider a modified tardiness objective of the form $\sum w_j(T_j + d_j)$. This modified objective was introduced in the offline version of the problem with unit weights in which all jobs have a common deadline by Kovalyov and Werner [1]. Kolliopoulos and Steiner [2] considered the general version of the problem with arbitrary weights and showed a reduction to the problem of finding an approximate solution to the problem of weighted total completion time. Their result showed that any ρ-approximation algorithm for the problem of minimizing total weighted completion time was an $(\rho + 1)$-approximation algorithm for the problem of minimizing weighted modified total tardiness. Liu et al. [3] extended this idea to online settings in addition to an availability constraint on the machines. They provided $O(1)$-competitive algorithms for clairvoyant scenarios. Specifically, for the single-machine version of the problem with weights, they showed a 2-competitive lower bound and a 3-competitive algorithm as an upper bound. For the multiple-machine version of the unit weight problem, they provided a lower bound of 1.309 and a 3-competitive algorithm. Kolliopoulos and Steiner [2] considered a stochastic version of the problem where the processing times of the jobs are assumed to be random variables with known distributions.

As clairvoyant information is difficult to obtain in several practical applications, this paper examines semi-clairvoyant and prediction-clairvoyant scenarios where the processing time information of a job is limited at its release time. In this context, *limited* refers to the fact that a scheduling algorithm utilizes a proxy that either depends on the knowledge of the range or on an estimation of the job's processing time instead of its true processing time. For these scenarios, we develop competitive algorithms addressing the modified total tardiness objective. Specifically, we make the following contributions.

- In prediction-clairvoyant settings: an $O(\mu \log \hat{P})$-competitive algorithm without migration for parallel identical machines with unit weights.
- In semi-clairvoyant settings: an $O(\log P)$-competitive algorithm for parallel identical machines with unit weights.
- In speed-prediction settings: an $O(\mu)$-competitive algorithm for parallel unrelated machines with general weights and clairvoyant processing times.

II. Preliminaries

Each job j arrives at a time r_j and has a deadline d_j such that $r_j \leq d_j$. The job has an actual processing time p_j. Upon arrival, either an estimated processing time $\tilde{p_j}$ or its job class l_j is available. Let C_j denote the time at which a job completes its execution. Reusing terminology and notation from [4], we let $\mu_1 = \max_j \frac{p_j}{\tilde{p_j}}$ be maximum underestimation error among all the arriving jobs. Similarly, we let $\mu_2 = \max_j \frac{\tilde{p_j}}{p_j}$ be maximum overestimation error among all the arriving jobs and $\mu = \mu_1 \cdot \mu_2$ be the distortion parameter.

We let P be the ratio between the maximal actual processing time and the minimal actual processing time among the jobs, i.e., $P = \frac{\max_j p_j}{\min_j p_j}$. Additionally, we let $\hat{P}$ be the ratio between the maximal estimated processing time and the minimal estimated time among the jobs, i.e., $\hat{P} = \frac{\max_j \tilde{p_j}}{\min_j \tilde{p_j}}$. Furthermore, we use the following definitions.

Definition 1 (Predicted job class)**.** We define the predicted class $\tilde{\ell_j}$ of a job j, as an integer k such that $\tilde{p_j} \in [2^k, 2^{k+1}]$.

Definition 2 (Job class)**.** We define the class ℓ_j of a job j, as an integer k such that $p_j \in [2^k, 2^{k+1}]$.

Definition 3 (Tardiness)**.** we define tardiness T_j of a job j as $\max(C_j - d_j, 0)$.

Definition 4 (Modified Tardiness)**.** we define modified tardiness $\tilde{T_j}$ of a job j as $T_j + d_j$.

Definition 5 (Total Modified Tardiness)**.** we define total modified tardiness as $\sum \tilde{T_j}$.

Definition 6 (Total Completion Time)**.** we define total completion time as $\sum C_j$.

Definition 7 (Total Flow Time)**.** we define total flow time as $\sum C_j - r_j$.

III. Prediction-Clairvoyant Scheduling on Parallel Machines

In this section, we focus on our first contribution, which addresses the problem of minimizing the total modified tardiness on parallel identical machines utilizing predicted job processing times. For this, we reuse the prediction-based algorithm by Azar et al. [4] that was developed for the problem of minimizing the total flow time with a competitive ratio of $O(\mu \log \hat{P})$. This algorithm satisfies the consistency and robustness properties [5] desired from prediction-based online algorithms. When the value of the distortion parameter is close to one due to high-quality predictions, the algorithm has a competitive ratio $O(\log \hat{P})$, which matches the lower bound for clairvoyant settings, satisfying the consistency property. We prove that this algorithm has an identical competitive ratio for the objective of modified total tardiness. Formally, we prove the following result.

Theorem 1. *Algorithm 1 is $O(\mu \log \hat{P})$-competitive for minimizing modified total tardiness on parallel identical machines with predicted job processing times.*

A. Algorithm

Algorithm 1 utilizes job classes based on the predicted processing time to prioritize jobs. When a new job arrives, it is placed at the top of the stack of an available machine or a machine currently processing a job of a higher class. Otherwise, it is added to a central queue. It assigns higher priority to jobs with lower classes. Higher-class jobs wait in the central queue until a machine becomes available or a lower-class job completes processing. Preempted jobs are returned to the machine stack on which they were preempted and resumed later on the same machine.

B. Analysis

Algorithm 1 was originally developed to minimize total flow time, and we reuse it without modifications for the objective of total modified tardiness. To show that this algorithm indeed works for our objective, we use a proof method similar to the one used in [2] and [3]. We first utilize the result that reducing the problem of the total completion time minimization to the problem of total flow time minimization increases the competitive ratio by a constant multiplicative factor of 2. We then utilize the result that reducing the tardiness minimization problem to the problem of total completion time minimization problem increases the competitive ratio by an additive constant 1. Lastly, we plug in the asymptotic bound of Algorithm 1. Formally, we need the following lemmas to prove Theorem 1.

Algorithm 1 Distortion Oblivious Non-Migrative Scheduling Algorithm

```
function UponJobRelease(j)
    if exists an idle machine or a machine that currently processes a job of a class higher than ℓ̂_j then
        Insert j to the top of that machine stack.
    else
        Insert j to the pool.
    end if
end function
function UponJobCompletion(j)
    Denote by m_j the machine j was processed on.
    Pop j from the stack of m_j, and let j' be the next job in that stack.
    if the job with lowest class in the pool j'' has class strictly less than that of j' or j' does not exist then
        Remove j'' from the pool and insert it to the top of the stack of m_j.
    end if
end function
```

Lemma 1. *(Theorem 1 from [2]) Consider a member $\alpha_0|\beta_0|\sum_j w_j C_j$ of the family of scheduling problems $\alpha|\beta|\sum_j w_j C_j$ for which there is a γ-approximation algorithm. Then the same algorithm achieves a $(\gamma+1)$-approximation for the problem $\alpha_0|\beta_0|\sum_j w_j(T_j + d_j)$.*

Proof. See [2] for a comprehensive proof. □

As two of the algorithms we consider in this paper address the problem of total flow time, we use the results from [6] that provide a reduction from the problem of minimizing total completion time to the problem of minimizing total flow time. Combining Theorem 7.3 and lemma 7.6 from [6], we have the following result.

Lemma 2. *Consider the problem of minimizing the total weighted flow time, for which a s-speed, c-competitive algorithm exists. The same algorithm is $2cs$-competitive for the problem of minimizing total weighted completion time.*

Proof. See chapter 7 in [6] for a comprehensive proof. □

The next lemma states the competitiveness of Algorithm 1 for the objective of total flow time.

Lemma 3. *(Theorem 2 from [4]) Algorithm 1 is $O(\mu \log \hat{P})$-competitive for inputs with distortion μ for the problem of minimizing total flow time on parallel identical machines.*

Proof. See [4] for a comprehensive proof. □

Proof of Theorem 1 With the preceding lemmas in place, we are now ready to prove Theorem 1. Let $C_j(alg)$ and $T_j(alg)$ be the completion times and tardiness of n jobs, respectively, due to the scheduling policy of Algorithm 1. Let $C_j(opt)$ and $T_j(opt)$ be the completion times and tardiness due to an optimal algorithm for the same instance of jobs.

By definition, we have the following equality.

$$\sum (T_j + d_j) = \sum \max(C_j, d_j) \tag{1}$$

The left-hand side of Eq. (1) represents the total modified tardiness as the sum of tardiness of individual jobs and their respective deadlines. This is equivalent to the sum of the maximum between the completion time $C_j(alg)$ and the deadline d_j of each job j.

Splitting the sum of the right-hand side into two separate sums, we have the following inequality.

$$\sum \max(C_j(alg), dj) \leq \sum C_j(alg) + \sum d_j \tag{2}$$

Similarly, we have the following lower bound.

$$\sum \max(C_j(opt), dj) \leq \sum C_j(opt) + \sum d_j \tag{3}$$

From the definition of competitive ratio, we have the following inequality bounding the total completion time.

$$\sum C_j(alg) + \sum d_j \leq c \cdot \sum C_j(opt) + \sum d_j \tag{4}$$

Since Algorithm 1 minimizes total flow time, from Lemma 2, we know that any algorithm that is c-competitive for minimizing total flow time bounds the total completion time by a multiplicative factor of 2c of the optimal completion time. Using this fact, we have the following inequality.

$$\sum \max(C_j(alg), dj) \leq 2 \cdot c \cdot \sum C_j(opt) + \sum d_j \tag{5}$$

Dividing the above equation with the respective lower bounds, we get

$$\frac{\sum \max(C_j(alg), dj)}{\sum \max(C_j(opt), dj)} \leq \frac{2 \cdot c \cdot \sum C_j(opt) + \sum d_j}{\sum C_j(opt) + \sum d_j} \tag{6}$$

Splitting the right-hand side of the above inequality into two separate terms, we get

$$\frac{2 \cdot c \cdot \sum C_j(opt)}{\sum C_j(opt) + \sum d_j} \leq 2 \cdot c \tag{7}$$

Similarly,

$$\frac{\sum d_j}{\sum C_j(opt) + \sum d_j} \leq 1 \tag{8}$$

Combining Eq. (7) and Eq. (8) and rearranging the terms in equation Eq. (6), we get

$$\sum \max(C_j(alg), d_j) \leq (2 \cdot c + 1) \cdot \sum \max(C_j(opt), d_j) \tag{9}$$

From Lemma 3, we can replace the constant c with asymptotic bound $O(\mu \log \hat{P})$ of Algorithm 1. Ignoring the constant factors, we have the following inequality.

$$\sum \max(C_j(alg), d_j) \leq O(\mu \log \hat{P}) \cdot \sum \max(C_j(opt), d_j) \tag{10}$$

Using the definition from Eq. (1) and rewriting the above equation, we get

$$\sum (T_j(alg) + d_j) \leq O(\mu \log \hat{P}) \cdot \sum (T_j(opt) + d_j) \tag{11}$$

The claim follows. □

IV. Semi-clairvoyant Scheduling on Parallel Machines

This section focuses on our contribution related to the semi-clairvoyant settings. This is an alternative approach to address the problem of the unavailability of precise processing times at the time of their release. Here, a scheduler utilizes the approximate knowledge of the processing time of a job in terms of its class to determine the order in which the jobs are processed instead of its actual processing time. A job's class is an integer value that identifies the processing time range of the job. While one can assume that a job's class is known precisely, it is possible to consider the scenario in which a job's class is also predicted. For each of these scenarios, the lowest-class-first (LCF) algorithm developed by Bechetti et al. [7] has been proven to be $O(\log P)$ and $O(\mu \log \hat{P})$ competitive for the total flow time objective, respectively. These bounds match proven lower bounds for the total flow time objective. We reuse this algorithm to minimize modified total tardiness. Formally, we prove the following results.

Theorem 2. *Algorithm 2 is $O(\log P)$-competitive for the problem of minimizing modified total tardiness on parallel identical machines in semi-clairvoyant settings.*

Theorem 3. *Algorithm 2 is $O(\mu \log \hat{P})$-competitive for the problem of minimizing modified total tardiness on parallel identical machines in semi-clairvoyant settings with job class predictions.*

A. Algorithm

Algorithm 2 assumes that each job reveals its actual (predicted) class on its arrival. When a new job arrives, it is assigned to an available machine or a machine currently processing a job of a higher class. Otherwise, it is added to a central queue. This approach assigns higher priority to jobs with lower classes, while higher-class jobs wait in the central queue until a machine becomes available or a lower-class job completes processing.

Algorithm 2 Lowest Class First Scheduling Algorithm

function UPONJOBRELEASE(j)
  if exists an idle machine or a machine that currently processes a job k of a class higher than $\ell_j(\hat{\ell}_j)$ **then**
    Preempt k and insert it into the pool.
    Run j.
  else
    Insert j into the pool.
  end if
end function
function UPONJOBCOMPLETION(j)
  Denote by m_j the machine j was processed on.
  if there exists a job in the pool j'' **then**
    Remove j'' with the lowest class from the pool and run on m_j.
  end if
end function

B. Analysis

To show that Algorithm 2 is competitive for the objective of modified tardiness minimization, our analysis follows an identical approach to our proof of Theorem 1. Specifically, we utilize the reduction in Lemma 2 that bounds the total completion time by a factor of 2c of the optimal total completion time when using an algorithm designed to minimize total flow time. We then utilize the reduction in Lemma 1 that bounds the total modified tardiness by a factor of $c+1$. Finally, we plug in the asymptotic upper bound for flow time minimization. Formally, to prove Theorem 2, we need an additional result stated in Lemma 4.

Lemma 4. *(Theorem 15 from [7]) Algorithm 2 is $O(\log P)$-competitive for the problem of minimizing total flow time on parallel identical machines.*

Proof. See [7] and [8] for a comprehensive proof. □

Proof of Theorem 2 Let $C_j(alg)$ and $T_j(alg)$ be the completion times and tardiness of n jobs, respectively, due to the scheduling policy of Algorithm 2. Let $C_j(opt)$ and $T_j(opt)$ be the completion times and tardiness due to an optimal algorithm for the same instance of jobs.

The left-hand side of Eq. (1) represents the total modified tardiness as the sum of tardiness of individual jobs and their respective deadlines. This is equivalent to the sum of the maximum between the completion time $C_j(alg)$ and the deadline d_j of each job j.

Splitting the sum of the right-hand side of Eq. (1) into two separate sums, we have the following inequality.

$$\sum \max(C_j(alg), dj) \leq \sum C_j(alg) + \sum d_j \tag{12}$$

Similarly, we have the following lower bound.

$$\sum \max(C_j(opt), dj) \leq \sum C_j(opt) + \sum d_j \tag{13}$$

From the definition of competitive ratio, we have the following inequality bounding the total completion time.

$$\sum C_j(alg) + \sum d_j \leq c \cdot \sum C_j(opt) + \sum d_j \tag{14}$$

Since Algorithm 2 minimizes total flow time, from Lemma 2, we know that any algorithm that is c-competitive for minimizing total flow time bounds the total completion time by a multiplicative factor of 2c of the optimal completion time. Using this fact, we have the following inequality.

$$\sum \max(C_j(alg), dj) \leq 2 \cdot c \cdot \sum C_j(opt) + \sum d_j \tag{15}$$

Dividing the above equation with the respective lower bounds, we get

$$\frac{\sum \max(C_j(alg), dj)}{\sum \max(C_j(opt), dj)} \leq \frac{2 \cdot c \cdot \sum C_j(opt) + \sum d_j}{\sum C_j(opt) + \sum d_j} \tag{16}$$

Splitting the right-hand side of the above inequality into two separate terms, we get

$$\frac{2 \cdot c \cdot \sum C_j(opt)}{\sum C_j(opt) + \sum d_j} \leq 2 \cdot c \tag{17}$$

Similarly,

$$\frac{\sum d_j}{\sum C_j(opt) + \sum d_j} \leq 1 \tag{18}$$

Combining Eq. (17) and Eq. (18) and rearranging the terms in equation Eq. (6), we get

$$\sum \max(C_j(alg), d_j) \leq (2 \cdot c + 1) \cdot \sum \max(C_j(opt), d_j) \tag{19}$$

Using the result from Lemma 4 and ignoring the constant factors, we have the following inequality.

$$\sum \max(C_j(alg), d_j) \leq O(\mu \log P) \cdot \sum \max(C_j(opt), d_j) \tag{20}$$

Using the definition from Eq. (1) and rewriting the above equation, we get

$$\sum (T_j(alg) + d_j) \leq O(\mu \log P) \cdot \sum (T_j(opt) + d_j) \tag{21}$$

The claim follows. □

To prove Theorem 3, we need the following result.

Lemma 5. *(Theorem 14 from [4]) Algorithm 2 is $O(\mu \log \hat{P})$-competitive for inputs with distortion μ for the problem of minimizing total flow time on parallel identical machines with predicted job classes.*

Proof. See [4] for a comprehensive proof. □

Proof of Theorem 3 Our proof is identical to the proofs of Theorem 1 and Theorem 2. By plugging in the result of Lemma 5, the claim follows. □

V. Scheduling to Minimize Weighted Modified Tardiness on Unrelated Machines

We now consider a generalization that aims to minimize weighted modified total tardiness on unrelated machines. Here, the processing speed of the jobs is different on different machines. For this, we rely on the algorithm and analysis by Megow et al. [9].

We consider the model where an algorithm can access a predicted processing speed for each machine and job. we represent this as $\hat{s_{ij}}$ while s_{ij} is the actual processing speed. Additionally, we assume a clairvoyant model, i.e., each job's exact (normalized) processing time is known upon arrival. Similar to the distortion parameter considered for the case of predicted job processing times, we let $\mu_1 = \max \frac{s_{ij}}{\hat{s_{ij}}}$ be the maximum underestimation error among all the arriving jobs. Similarly, we let $\mu_2 = \max \frac{\hat{s_{ij}}}{s_{ij}}$ be maximum overestimation error among all the speed predictions and $\mu = \mu_1 \cdot \mu_2$ be the distortion parameter.

In this section, we prove the following result.

Theorem 4. *Algorithm 3 is $O(\mu)$-competitive for the problem of minimizing weighted modified total tardiness on unrelated machines with speed predictions.*

A. Algorithm

The Maximum Density Scheduling Algorithm prioritizes jobs based on their density. The density is calculated for each active job by considering its weight, predicted speeds, and known processing times. The algorithm determines the maximum number of jobs to run based on available machines and the number of currently active jobs. It selects at most M jobs that maximize the sum of the densities to run.

Algorithm 3 Maximum Density Scheduling Algorithm

1: **function** UponJobRelease(j)
2: Compute density $\hat{\delta}_{ij} = \frac{w_j \hat{s}_{ij}}{p_j}$ for all active jobs
3: Find $k = \min(M, |J(t)|)$ jobs that maximize $(\sum_k \hat{\delta}_{ij})$
4: Run k jobs
5: **end function**
6: **function** UponJobCompletion(j)
7: Compute density $\hat{\delta}_{ij} = \frac{w_j \hat{s}_{ij}}{p_j}$ for all active jobs
8: Find $k = \min(M, |J(t)|)$ jobs that maximize $(\sum_k \hat{\delta}_{ij})$
9: Run k jobs
10: **end function**

B. Analysis

To prove Theorem 4, we need the following result.

Lemma 6. *(Theorem 2.2 from [9]) Algorithm 3 has a competitive ratio of at most 8μ for minimizing the total weighted completion time on unrelated machines with speed predictions.*

Proof. See [9] for a comprehensive proof. □

Proof of Theorem 4 Let $C_j(alg)$ and $T_j(alg)$ be the completion times and tardiness of n jobs, respectively, due to the scheduling policy of Algorithm 1. Let $C_j(opt)$ and $T_j(opt)$ be the completion times and tardiness due to an optimal algorithm for the same instance of jobs.

By definition, we have the following equality.

$$\sum w_j(T_j(alg) + d_j) = \sum w_j \max(C_j(alg), dj) \quad (22)$$

Similarly,

$$\sum w_j(T_j(opt) + d_j) = \sum w_j \max(C_j(opt), dj) \quad (23)$$

The left-hand side of Eq. (22) represents the total weighted modified tardiness as the sum of weighted tardiness of individual jobs and their respective deadlines. This is equivalent to the weighted sum of the maximum between the completion time $C_j(alg)$ and the deadline d_j of each job j.

We have the following inequality by splitting the sum of the right-hand side into two separate sums.

$$\sum w_j \max(C_j(alg), dj) \leq \sum w_j C_j(alg) + \sum w_j d_j \quad (24)$$

Similarly, we have the following lower bound.

$$\sum w_j \max(C_j(opt), dj) \leq \sum w_j C_j(opt) + \sum w_j d_j \quad (25)$$

From the definition of competitive ratio, we have the following inequality bounding the total completion time.

$$\sum w_j \max(C_j(opt), dj) \leq c \cdot \sum w_j C_j(opt) + \sum w_j d_j \quad (26)$$

Dividing the above equation with the respective lower bounds, we get

$$\frac{\sum w_j \max(C_j(alg), dj)}{\sum w_j \max(C_j(opt), dj)} \leq \frac{c \cdot \sum w_j C_j(opt) + \sum w_j d_j}{\sum w_j C_j(opt) + \sum w_j d_j} \quad (27)$$

Splitting the right-hand side of the above inequality into two separate terms, we get

$$\frac{c \cdot \sum w_j C_j(opt)}{\sum w_j C_j(opt) + \sum w_j d_j} \leq c \quad (28)$$

Similarly,

$$\frac{\sum w_j d_j}{\sum w_j C_j(opt) + \sum w_j d_j} \leq 1 \quad (29)$$

Combining Eq. (28) and Eq. (29) and rearranging the terms in equation Eq. (6), we get

$$\sum w_j \max(C_j(alg), d_j) \leq (c+1) \cdot \sum w_j \max(C_j(opt), d_j) \quad (30)$$

Using the result from Lemma 6, we can replace the constant c with asymptotic bound $O(\mu)$ of Algorithm 3. Ignoring the constant factors, we have the following inequality.

$$\sum w_j \max(C_j(alg), d_j) \leq O(\mu) \sum w_j \max(C_j(opt), d_j) \tag{31}$$

Using the definition of Eq. (22) and Eq. (23) and rewriting the above equation, we get

$$\sum w_j (T_j(alg) + d_j) \leq O(\mu) \sum w_j (T_j(opt) + d_j) \tag{32}$$

The claim follows. □

VI. Conclusion

We considered the problem of minimizing modified total tardiness on parallel machines with different levels of information about job processing times and provided algorithms with competitive guarantees. One natural direction is to address the generalized version of the problem by including weights on identical machines. Another direction is to relax the requirement of clairvoyance for scheduling on unrelated machines.

VII. Acknowledgement

The research leading to these results has received funding from the Knowledge Foundation (KKS), under the projects FIESTA (Project No. 20190034) and SACSys (Project No. 20190021), and under the Swedish Research Council (VR), under the project PSI (Project No. 2020-05094).

References

[1] M. Y. Kovalyov and F. Werner, "Approximation schemes for scheduling jobs with common due date on parallel machines to minimize total tardiness," *Journal of Heuristics*, vol. 8, pp. 415–428, 2002.

[2] S. G. Kolliopoulos and G. Steiner, "Approximation algorithms for scheduling problems with a modified total weighted tardiness objective," *Operations research letters*, vol. 35, no. 5, pp. 685–692, 2007.

[3] M. Liu, Y. Xu, C. Chu, and F. Zheng, "Online scheduling to minimize modified total tardiness with an availability constraint," *Theoretical computer science*, vol. 410, no. 47-49, pp. 5039–5046, 2009.

[4] Y. Azar, E. Peretz, and N. Touitou, "Distortion-Oblivious Algorithms for Scheduling on Multiple Machines," in *Leibniz International Proceedings in Informatics, LIPIcs*, vol. 248, 12 2022, pp. 16:1–16:18.

[5] S. Im, R. Kumar, M. M. Qaem, and M. Purohit, "Non-clairvoyant scheduling with predictions," *ACM Transactions on Parallel Computing*, vol. 10, no. 4, pp. 1–26, 2023.

[6] N. Bansal, *Algorithms for flow time scheduling*. Carnegie Mellon University, 2003.

[7] L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, and K. Pruhs, "Semi-clairvoyant scheduling," *Theoretical computer science*, vol. 324, no. 2-3, pp. 325–335, 2004.

[8] S. Leonardi, "A simpler proof of preemptive total flow time approximation on parallel machines," in *Efficient Approximation and Online Algorithms: Recent Progress on Classical Combinatorial Optimization Problems and New Applications*. Springer, 2006, pp. 203–212.

[9] A. Lindermayr, N. Megow, and M. Rapp, "Speed-oblivious online scheduling: knowing (precise) speeds is not necessary," in *International Conference on Machine Learning*. PMLR, 2023, pp. 21 312–21 334.