Lagrangian Duality in Online Scheduling with Resource Augmentation and Speed Scaling

Nguyen Kim Thang*

Abstract

We present an unified approach to study online scheduling problems in the resource augmentation/speed scaling models. Potential function method is extensively used for analyzing algorithms in these models; however, they yields little insight on how to construct potential functions and how to design algorithms for related problems. In the paper, we generalize and strengthen the dual-fitting technique proposed by Anand et al. [1]. The approach consists of considering a possibly non-convex relaxation and its Lagrangian dual; then constructing dual variables such that the Lagrangian dual has objective value within a desired factor of the primal optimum. The competitive ratio follows by the standard Lagrangian weak duality. This approach is simple yet powerful and it is seemingly a right tool to study problems with resource augmentation or speed scaling. We illustrate the approach through the following results.

1. We revisit algorithms EQUI and LAPS in Non-clairvoyant Scheduling to minimize total flow-time. We give simple analyses to prove known facts on the competitiveness of such algorithms. Not only are the analyses much simpler than the previous ones, they also explain why LAPS is a natural extension of EQUI to design a scalable algorithm for the problem.
2. We consider the online scheduling problem to minimize total weighted flow-time plus energy where the energy power $f(s)$ is a function of speed s and is given by s^α for $\alpha \geq 1$. For a single machine, we showed an improved competitive ratio for a non-clairvoyant memoryless algorithm. For unrelated machines, we give an $O(\alpha/\log\alpha)$-competitive algorithm. The currently best algorithm for unrelated machines is $O(\alpha^2)$-competitive.
3. We consider the online scheduling problem on unrelated machines with the objective of minimizing $\sum_{i,j} w_{ij} f(F_j)$ where F_j is the flow-time of job j and f is an arbitrary non-decreasing cost function with some nice properties. We present an algorithm which is $\frac{1}{1-3\epsilon}$-speed, $\frac{2K(\epsilon)}{\epsilon}$-competitive where $K(\epsilon)$ is a function depending on f and ϵ. The algorithm does not need to know the speed $(1+\epsilon)$ a priori. A corollary is a $(1+\epsilon)$-speed, $\frac{k}{\epsilon^{1+1/k}}$-competitive algorithm (which does not know ϵ a priori) for the objective of minimizing the weighted ℓ_k-norm of flow-time.

*IBISC, University of Evry Val d'Essonne, France. Email: thang@ibisc.fr

1 Introduction

We consider online scheduling problems where jobs arrive at unrelated servers/machines over time. Each job j has release date r_j and its processing time p_{ij} and weight w_{ij} on machine i. At the arrival time r_j, job j becomes known to the scheduling algorithm. We distinguish two different models. At time r_j, in the *non-clairvoyant* model only the weights w_{ij}'s becomes known to the scheduler while in the *clairvoyant* model, all parameter of jobs j are available. A scheduler must determine how to process jobs in order to optimize a quality of service without the knowledge about future. In the paper, we study natural qualities of service related to the flow-times of jobs. The *flow-time* of a job is the total amount of time it spends in the system, i.e., the difference of its completion time and its release time.

A popular measure for studying the performance of online algorithms is *competitive ratio*. An algorithm is said to be *c-competitive* if for any instance its objective is within factor c of the optimal offline algorithm's objective. Unfortunately, for many problems, any online algorithm has large competitive ratio even that some heuristics have performance very close to the optimum in practice. To remedy the limitation of pathological instances in worst-case analysis, a popular relaxation *resource augmentation* model was introduced in [24]. In this relaxation, the online algorithm is given extra speed to process jobs and compared to the optimal offline algorithm. This model has successfully provided theoretical evidence for heuristics with good performance in practice. Besides, algorithms could be classified according to their competitive ratios in the model of resource augmentation for practical choices. We say an algorithm is *s-speed c-competitive* if for any input instance the objective value of the algorithm while running at speed s is at most c times the objective value of the optimal offline scheduler while running at unit speed. Ideally, we would like algorithms to be constant competitive when given $(1 + \epsilon)$ times a resource over the optimal offline algorithm for any constant $\epsilon > 0$. Such algorithms are called *scalable*.

The most successful tool until now to analyze online scheduling algorithms with resource augmentation is the potential function method. Potential functions has been designed and show that the corresponding algorithms behave well in an amortized sense. Designing such potential functions is far from trivial and often yields little insight about how to design such potential functions and algorithms for related problems (a generalized variant with additional constraints for example).

Recently, Anand et al. [1] gave a more direct and interesting approach for analyzing online scheduling algorithms with resource augmentation based on the technique of dual fitting for convex programming relaxation. Informally, the technique could be described as follows. Consider a linear (convex) programming relaxation of a given problem and the dual linear program (or Lagrangian dual). Then construct a feasible solution for the dual (given an online algorithm) and prove that its objective value is close to that of the online algorithm. The main advantage of this technique is that the dual variables (which constitute the desired dual solution) often have intuitive interpretations and their construction could be naturally deduced from the algorithm. Consequently, the procedures of analyzing and designing algorithms are more interactive and could be done in a principled manner.

Independently, Gupta et al. [20] gave a principled method to design online algorithms for non-linear programs. Their approach could be seen as an extension of the online primal-dual method for linear programming [10]. Roughly speaking, in the method the dual variables are set in such a way that the increase rate in the dual objective is proportional to the one in the primal objective. This approach is particularly powerful while the primal objective function is convex.

1.1 Approach and Contributions

The main contribution of the paper is to show a principled approach to design/analyze online scheduling algorithms with resource augmentation (or speed scaling) by strengthening the dual fitting technique in [1]. The approach is sharply inspired by the one in [1]. First, consider a mathematical programming relaxation (associated with a given problem) which is *not* necessarily convex and its Lagrangian dual. Then construct dual variables such that the Lagrangian dual has objective value within a desired factor of the primal one (due to some algorithm). Then by the standard Lagrangian weak duality[1] for mathematical programming, the competitive ratio follows.

Lemma 1 (Weak duality) *Consider a possibly non-convex optimization problem*

$$p^* := \min_x f_0(x) \ : \quad f_i(x) \leq 0, \quad i = 1, \ldots, m.$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ for $0 \leq i \leq m$. Let $\mathcal{X}$ be the feasible set of x. Let $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be the Lagragian function

$$L(x, \lambda) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x).$$

Define $d^* = \max_{\lambda \geq 0} \min_{x \in \mathcal{X}} L(x, \lambda)$ where $\lambda \geq 0$ means $\lambda \in \mathbb{R}^m_+$. Then $p^* \geq d^*$.

Weak duality is indeed a direct consequent of the *minimax* inequality

$$\max_{\lambda \in \mathcal{Y}} \min_{x \in \mathcal{X}} L(x, \lambda) \leq \min_{x \in \mathcal{X}} \max_{\lambda \in \mathcal{Y}} L(x, \lambda)$$

where $\mathcal{X}$ and $\mathcal{Y}$ are feasible sets of x and λ. Intuitively, our approach could be considered as a one-shot game between an algorithm and an adversary. The algorithm chooses dual variables λ^* in such a way that whatever the choice of the adversary, the value $\min_{x \in \mathcal{X}} L(x, \lambda^*)$ is always within a desirable factor c of the objective due to the algorithm. In the model, the adversary has less resource than the algorithm. For example, if the algorithm processes jobs with unit rate then the adversary can run only with rate $(1 - \epsilon)$. We extensively use that advantage in proving bounds for the dual objective.

In high level, our approach is the same as the one in [1] except that the relaxation is possibly non-convex. However, the flexibility of our approach provides many advantages. First, a problem could be more directly and naturally formulated as a non-convex program. For example, the online scheduling problem to minimize total weighted flow time plus energy could be naturally formulated by a non-convex relaxation (Section 4) while it is unclear how to formalize the problem by a convex program. Consequently, the analysis is usually simpler, cleaner and the performance guarantee is improved. Inversely, the simplicity of the analysis gives insights on the problems and so (simple) algorithms could be designed. Second, as it is not constrained to be a convex optimization program, additional constraints for generalized variants of a problem could be easily incorporated (for example, from a single machine to unrelated machines). Thereby an algorithm for generalized variants could be derived based on the previous ones for the basic problem and the ideas of the analyses remain essentially the same.

We illustrate the advantages of the approach through the following results.

[1] For completeness, the proof of weak duality is given in the appendix

1. In Section 3, we revisit algorithms EQUI and LAPS_ϵ in Non-clairvoyant Scheduling to minimize total flow-time. We give simple analyses to prove known facts that EQUI is $\frac{1}{1/2-\epsilon}$-speed, $\frac{1}{\epsilon}$-competitive [16] and $\mathsf{LAPS}\epsilon$ is $\frac{1}{1-2\epsilon}$-speed, $\frac{2}{\epsilon^2}$-competitive [17]. Not only are the analyses much simpler than the previous ones, they also explain why LAPS_ϵ is a natural extension of EQUI to design a scalable algorithm for the problem.

2. In Section 4, we consider the online scheduling problem to minimize total weighted flow-time plus energy where the energy power $f(s)$ is a function of speed s and is given by s^α for $\alpha \geq 1$. For a single machine, we showed an improved competitive ratio $O(2^\alpha)$ for a non-clairvoyant memoryless algorithm (its performance was previously known to be $O(3^\alpha)$). For unrelated machines, we give an $O(\alpha/\log\alpha)$-competitive algorithm. This bound matches to the currently best algorithm for a single machine [8]. The currently best algorithm for unrelated machines is $O(\alpha^2)$-competitive [1].

3. In Section 5, we consider the online scheduling problem on unrelated machines with the objective of minimizing $\sum_{i,j} w_{ij} f(F_j)$ where F_j is the flow-time of job j and f is an arbitrary non-decreasing cost function with some nice properties (for example, f is in class $\mathcal{C}^1$ and f' is non-decreasing). We derive an algorithm which is $\frac{1}{1-3\epsilon}$-speed, $\frac{2K(\epsilon)}{\epsilon}$-competitive where $K(\epsilon)$ is a function depending on f and ϵ. The algorithm does not need to know the speed $(1+\epsilon)$ a priori. A corollary is a $(1+\epsilon)$-speed, $\frac{k}{\epsilon^{1+1/k}}$-competitive algorithm (which does not know ϵ a priori) for the objective of minimizing the weighted ℓ_k-norm of flow-time. That answers an open question in [21] and marginally improves the currently best known algorithm which is $(1+\epsilon)$-speed, $\frac{k}{\epsilon^{2+1/k}}$-competitive [1].

Besides, using the approach, some related problems and direct generalizations of the above problems could be proved. Those promising extensions will be discussed at the end of each section and in the conclusion.

1.2 Related work

The online problems of minimizing objectives related to (weighted) flow-times of jobs have been extensively studying. For the basic problem of minimizing total flow-time on single machine, it is well-known that Shortest Remaining Processing Time (SRPT) is the optimal algorithm. However, that is the only constant competitive algorithm. Bansal and Chan [3] showed that no algorithm is constant competitive for the problem of minimizing total weighted flow-time on single machine. In fact, no bounded competitive ratio holds for parallel machines setting [14, 19].

The strong lower bounds motivate the use of resource augmentation, originally introduced by Kalyanasundaram and Pruhs [24], which circumvents the persimist worst-case paradigm. In the same paper, the authors gave an $O(1/\epsilon)$-competitive algorithm, called SETF, for the objective of minimizing flow-time on a single machine in the non-clairvoyant setting. In this setting, without resource augmentation the competitive ratios of every deterministic and randomized algorithms are $\Omega(n^{1/3})$ and $\Omega(\log n)$, respectively [28]. Edmonds [16] considered algorithm EQUI and showed that it was $(2+\epsilon)$-speed, $2/\epsilon$-competitive. Later on, Edmonds and Pruhs [17] proposed a generalized algorithm called LAPS_ϵ. They proved that LAPS_ϵ is $(1+2\epsilon)$-speed, $4/\epsilon^2$-competitive for minimizing the objective of total flow-time (even with sublinear non-decreasing speedup curves).

In the clairvoyant setting, Bansal and Pruhs [4] proved that the Highest Density First (HDF) algorithm is $(1+\epsilon)$-speed, $O(1/\epsilon)$-competitive for the objective of weighted ℓ_k-norm of flow-time

on a single machine. Chadha et al. [11] gave the first $(1+\epsilon)$-speed, $O(1/\epsilon^2)$-competitive algorithm for minimizing weighted flow time on unrelated machines. Recently, using the approach based on linear programming and dual-fitting, Anand et al. [1] derived another simple algorithm which is $(1+\epsilon)$-speed, $O(1/\epsilon)$-competitive. Moreover, the authors extended this to an $(1+\epsilon)$-speed, $O(k/\epsilon^{2+1/k})$-competitive algorithm for the objective of weighted ℓ_k-norm of flow-time. Note that the latter needs to know the speed $(1+\epsilon)$ a priori.

For the objective of total flow-time plus energy on a single machine, Bansal et al. [8] gave a $(3+\epsilon)$-competitive algorithm. Besides, they also proved a $(2+\epsilon)$-competitive algorithm for minimizing total *fractional* weighted flow-time plus energy. Their results hold for a general class of convex power functions. Those results also imply an $O(\alpha/\log\alpha)$-competitive algorithm for weighted flow-time plus energy when the energy function is s^α. Again, always based on linear programming and dual-fitting, Anand et al. [1] proved an $O(\alpha^2)$-competitive algorithm for unrelated machines. The total (weighted) flow-time plus energy in non-clairvoyant setting has been also considered [12, 27]. Chan et al. [13] proved that a memoryless non-clairvoyant algorithm, which a variant of algorithm EQUI with a policy on speed, was $O(3^\alpha)$ competitive.

The objective of minimizing $\sum_{i,j} w_{ij} f(F_j)$ for general cost function f aims to capture multiple standard objectives in literature (weighted ℓ_k-norm of flow-time, weighted tardiness). A competitive algorithm for a general cost function could be useful particularly in scheduling with multiple objectives or in setting where objectives may compete with each other [2, 26]. For the offline version on a single machine, Bansal and Pruhs presented a polynomial time $O(\log\log P)$-approximation algorithm [5, 6] where P is the ratio of the maximum to minimum job size. Im et al. [23] showed that the HDF algorithm is $(2+\epsilon)$-speed, $O(1)$-competitive for arbitrary non-decreasing cost function f on a single machine. They also gave a scalable algorithm when f is a concave and twice differentiable.

Almost all of competitive algorithms with resource augmentation are proved by potential functions. Those clever functions are used to show that a particular algorithm is locally competitive in an amortized sense. Recently, a technique to construct potential functions for online scheduling has been systematically formalized and given in [22]. However, that still yields little insight about how to design algorithms and construct potential functions for related problems or for non-trivial generalized variants.

Anand et al. [1] was the first who proposed studying online scheduling with resource augmentation by linear (convex) programming and dual fitting. By this elegant approach, they gave simple algorithms and simple analysis with improved performance for problems where the analyses based on potential functions are complex or it is unclear how to design such functions. Our approach is greatly inspired by the one in [1].

Independently, Gupta et al. [20] gave a principled method to design online algorithms for non-linear programs. They showed the application of the method to online speed-scaling problems. Subsequently, [25] have applied the method to design an α^α-competitive for the problem of minimizing the consumed energy plus lost values.

2 Preliminaries

In unrelated machine environment, we are given a set of m machines and jobs arrive over time. A job j is released at time r_j and requires p_{ij} units of processing time if it is scheduled on machine i. The machines are allowed to process jobs preemptively. The *flow-time* of a job j is $F_j = C_j - r_j$

where C_j is its the completion time. If a job j is assigned to machine i then its weighted flow-time is $w_{ij}F_{ij}$. Consider a scheduling algorithm. A job j is *pending* at time t if it is not completed by the algorithm, i.e., $r_j \leq t < C_j$. At time t, we denote $q_{ij}(t)$ the *remaining* processing time of job j on machine i. The *total weight* of pending jobs assigned to machine i at time t is denoted as $W_i(t)$. In case where all jobs have unit weight, we use $N_i(t)$ (number of pending jobs) instead of $W_i(t)$. The *residual density* of a pending job j assigned to machine i at time t is $\delta_{ij}(t) = w_{ij}/q_{ij}(t)$. The *density* of a job j on machine i is $\delta_{ij}(r_j)$. We distinguish two different models: the *non-clairvoyant* model in which at the arrival of job j, only the weights w_{ij}'s becomes known to the scheduler; and the *clairvoyant* model in which all parameter of jobs j are available at its release time. Note that when only a single machine is considered, for simplicity the notations remain the same except that the machine index (usually i) will be dropped.

3 Non-clairvoyant Scheduling

The problem. In this section, we study the non-clairvoyant online scheduling problem with the objective of minimizing the total flow-time on a single machine. Let $x_j(t)$ be the variable that represents the processing rate of the machine on job j at time t for every job j. Let C_j be a variable representing the completion time of j. The relaxation could be formulated as the following mathematical program.

$$\begin{aligned}
\min \quad & \sum_j \frac{C_j - r_j}{p_j} \int_{r_j}^{C_j} x_j(t)dt \\
\text{subject to} \quad & \int_{r_j}^{C_j} x_j(t)dt = p_j \qquad \forall j \\
& \sum_{j=1}^{n} x_j(t) \leq 1 \qquad \forall t \\
& x_j(t) \geq 0 \qquad \forall j, t \\
& x_j(t) = 0 \qquad \forall j, \forall t \notin [r_j, C_j]
\end{aligned}$$

Note that the last constraints are redundant but they are kept in order to make the relaxation clear. The dual of that program is $\max \min_{x,C} L(x, C, \lambda, \gamma, \mu)$ where L is the Lagrangian

$$\begin{aligned}
&\sum_j \int_{r_j}^{C_j} \frac{C_j - r_j}{p_j} x_j(t)dt + \sum_j \lambda_j \left(p_j - \int_{r_j}^{C_j} x_j(t)dt \right) \\
&\qquad + \int_0^\infty \left(1 - \sum_j x_j(t) \right) \gamma(t)dt - \sum_j \int_0^\infty x_j(t)\mu_j(t)dt \\
&= \sum_j \lambda_j p_j - \sum_j \int_{r_j}^{C_j} x_j(t) \cdot \left(\lambda_j + \gamma(t) - \frac{C_j - r_j}{p_j} \right) dt + \int_0^\infty \gamma(t)dt - \sum_j \int_0^\infty x_j(t)\mu_j(t)dt
\end{aligned}$$

3.1 EQUI

Algorithm EQUI. The processor shares its resource equally to the pending jobs.

Let $q_1 \leq \ldots \leq q_n$ be remaining processing times of pending jobs at some time t. Assume that no new job is released after t, then the remaining time before completion for the first job is nq_1, that for the second job is $nq_1 + (n-1)(q_2 - q_1)$. By recurrence, the remaining time before completion for job j is $q_1 + \ldots + q_{j-1} + (n+1-j)q_j$ for $1 \leq j \leq n$.

Suppose that at time t, a new job arrives with processing time q such that $q_k \leq q < q_{k+1}$ for some index k. Then the flow time of the new job, assuming that no new job is released after t, is $q_1 + \ldots + q_{k-1} + (n+1-k)q$. Moreover, due to the arrival of the new job, the completion time of job k' is increased by $q_{k'}$ for $k' \leq k$; and by q for $k' > k$. Hence, the marginal increase of the total flow time due to the arrival of the new job is bounded by twice the flow time of that job.

Dual variables Choose $\gamma(t) = 0$, $\mu_j(t) = 0$ for every j, t and $\lambda_j = \lambda_j^E$ such that $\lambda_j^E p_j$ equals the flow time of j (due to the algorithm) assuming that no new job arrives after r_j. By the observation on the flow time of jobs in EQUI, we have that $\sum_j \lambda_j^E p_j \leq \mathcal{F}^E \leq 2\sum_j \lambda_j^E p_j$ where $\mathcal{F}^E$ is the total flow-time due to EQUI.

Lemma 2 *It holds that* $\frac{1}{p_j}\Big(\lambda_j^E p_j - (t - r_j)\Big) \leq N^E(t)$ *for* $t \geq r_j$ *where* $N^E(t)$ *is the number of pending jobs at time* t *by algorithm* EQUI.

Proof Observe that if some request arrives between time r_j and t, the left-hand side remains unchanged while the right hand-side is non-decreasing. Therefore, it is sufficient to prove the inequality assuming that no job is released after r_j. Consider $t \leq C_j^E$ (since otherwise the inequality trivially holds since the left-hand side is negative). Rename jobs in non-decreasing order of the remaining processing times at r_j, i.e., $q_1(r_j) \leq \ldots \leq q_n(r_j)$. Note that $p_j = q_j(r_j)$. Suppose that k is the pending job with smallest index at time t, i.e., jobs $1, \ldots, k-1$ have been completed. We have that

$$\frac{1}{p_j}\Big(\lambda_j^E p_j - (t - r_j)\Big) = \frac{1}{p_j}\Big(q_k(t) + \ldots + q_{j-1}(t) + (n-j)q_j(t)\Big) \leq N^E(t)$$

where the last inequality follows since $q_k(t) \leq \ldots \leq q_{j-1}(t) \leq q_j(t) \leq p_j$. □

Theorem 1 ([16]) *Algorithm* EQUI *is* $\frac{1}{1/2-\epsilon}$*-speed,* $\frac{1}{\epsilon}$*-competitive for the problem of minimizing total flow time.*

Proof As the adversary has only the speed $(1/2 - \epsilon)$, the processing rate of adversary $\sum_j x_j(t) \leq 1/2 - \epsilon$ for all t. By the choice of dual variables corresponding to EQUI, we have

$$\min_{x,C} L \geq \frac{\mathcal{F}^E}{2} - \int_0^\infty \sum_j x_j(t) N^E(t) \geq \frac{\mathcal{F}^E}{2} - \left(\frac{1}{2} - \epsilon\right)\int_0^\infty N^E(t) = \epsilon \mathcal{F}^E$$

where the first inequality is due to Lemma 2; the second inequality follows by $\sum_j x_j(t) \leq 1/2 - \epsilon$. Hence, the competitive ratio of EQUI is at most $1/\epsilon$. □

3.2 LAPS$_\beta$

Inspecting the analysis of EQUI, one realizes that in order to get a scalable algorithm, the machine should share its power only to a small fraction of pending jobs instead of all such jobs. This observation naturally leads to algorithm LAPS introduced in [17].

Algorithm LAPS_β. Let $0 < \beta \leq 1$. The processor shares its resource equally to the $\beta N^L(t)$ jobs with the latest arrival times where $N^L(t)$ is the number of pending jobs at time t.

Note that in the definition of the algorithm, $\beta N^L(t)$ is not necessarily an integer. However, that algorithm is equivalent to the following procedure. First, choose the $\lceil \beta N^L(t) \rceil$ most recent jobs. Then among such jobs, the machine shares its power to the $\lfloor \beta N^L(t) \rfloor$ most recent ones proportional to 1 and to the last job proportional to $(\beta N^L(t) - \lfloor \beta N^L(t) \rfloor)$. For the ease and simplicity of the exposition, we consider the version described in the definition.

Let j be a job released at time r_j. We will bound the marginal increase of the total flow time due to the arrival of j. Assuming that no new jobs are released after r_j and let C_k^L be the completion time of k for every pending job k at time r_j. Rename jobs in the increasing order of C_k^L's. By the algorithm, the completion time of job j is bounded by the following

$$C_j^L \geq r_j + \sum_{k<j} q_k(r_j) + \beta N^L(C_k^L) \cdot p_j$$

since at any moment during $[t, C_j^L]$, the processor shares its resource to at least $\beta N(C_k^L)$ jobs. Besides, due to the arrival of j, the completion time of a job $k > j$ increase by an amount p_j and the total completion times of jobs $\{k : k < j\}$ increases by at most $\frac{1}{\beta}\sum_{k<j} q_k(r_j)$. Therefore, the marginal increase of the total flow time due to job j is at most

$$(C_j^L - r_j) + \frac{1}{\beta}\sum_{k<j} q_k(r_j) + N^L(C_k^L) \cdot p_j \leq \left(1 + \frac{1}{\beta}\right)(C_j^L - r_j)$$

Dual variables. Choose $\gamma(t) = 0$, $\mu_j(t) = 0$ for every p, t and $\lambda_j = \lambda_j^L$ such that $\frac{1+\beta}{\beta}\lambda_j^L p_j$ equals the increase of the total flow time due to job j, assuming that no new job arrives after r_j. By the observation above, we have that $\lambda_j^L p_j \leq C_j^L - r_j$ and $\sum_j \lambda_j^L p_j = \frac{\beta}{1+\beta}\mathcal{F}^L$ where $\mathcal{F}^L$ is the total flow-time due to LAPS_β.

Lemma 3 *It holds that* $\frac{1}{p_j}\left(\lambda_j^L p_j - (t - r_j)\right) \leq \beta N^L(t)$ *for* $t \geq r_j$.

Proof Observe that if some request arrives between time r_j and t, the left-hand side remains unchanged while the right hand-side is non-decreasing. Therefore, it is sufficient to prove the inequality assuming that no job released after r_j. Consider $t \leq C_j^L$ (since otherwise the inequality trivially holds since the left-hand side is negative). Rename jobs in non-decreasing order completion times. Suppose that u is the pending job with smallest index at time t, i.e., jobs $1, \ldots, u-1$ have been completed. Let Q be the total work (processing units) of jobs $\{k : k > j\}$ that will be done in $[t, C_j^L]$. During $[t, C_j^L]$, the processor shares its resource to at most $\beta N^L(t)$ jobs. Therefore, at time t the total remaining work of jobs $\{k : u \leq k \leq j\}$ plus Q is at most $\beta N^L(t) q_j(t) \leq \beta N^L(t) p_j$ since otherwise, there must have been a job $k < j$ completed after C_j^L (contradiction to the definition of job indices). Therefore

$$\frac{1}{p_j}\left(\lambda_j^L p_j - (t - r_j)\right) \leq \frac{1}{p_j}(C_j^L - t) \leq \beta N^L(t)$$

□

Theorem 2 ([17]) *Algorithm LAPS_ϵ is $\frac{1}{1-2\epsilon}$-speed, $\frac{2}{\epsilon^2}$-competitive for the problem of minimizing total flow time.*

Proof Note that now β is chosen as ϵ. By the choice of dual variables corresponding to EQUI, we have

$$\begin{aligned}\min_{x,C} L &\geq \frac{\epsilon}{1+\epsilon}\mathcal{F}^L - \epsilon\int_0^\infty \sum_j x_j(t)N^L(t) \\ &\geq \frac{\epsilon}{1+\epsilon}\mathcal{F}^L - \epsilon\frac{1-\epsilon}{1+\epsilon}\int_0^\infty N^L(t) = \frac{\epsilon^2}{1+\epsilon}\mathcal{F}^L\end{aligned}$$

where the first inequality is due to Lemma 3; the second inequality follows by $\sum_j x_j(t) \leq 1-2\epsilon \leq \frac{1-\epsilon}{1+\epsilon}$. Hence, the competitive ratio of LAPS_ϵ is at most $2/\epsilon^2$. □

3.3 Extensions

A straightforward generalized variant to minimize the total weighted flow-time could be easily proved using the same analyses. Besides, online scheduling on a multiprocessor system with arbitrary speedup curves could be considered. In the model, jobs have varying degrees of parallelizability, some jobs may be sped up significantly when simultaneously process on multiple machines, while other jobs may be sped up by very little. It was shown that EQUI was $(2+\epsilon)$-speed $O(1)$-competitive [16] and LAPS was scalable [17]. By our approach, these results (together with related results in the model) could be reproved by simple analyses. Recently, algorithm LAPS was proved to be scalable for the online broadcast scheduling problem to minimize the total flow-time [9]. Again, the proof could be done by an analysis similar to the one in the previous section. Consequently, related results on broadcast scheduling follow in an unified manner. Those proofs will be appeared in the full version of this paper.

4 Weighted Flowtime plus Energy

The problem. In this section, we study the online scheduling with the objective of minimizing the total weighted flow-time plus energy. The energy power function is given by s^α where s is the speed of the machine and $\alpha \geq 1$ is a constant. In Section 4.1, we consider non-clairvoyant algorithms on a single machine and Section 4.2, we consider algorithms on unrelated machines.

4.1 Non-clairvoyant Scheduling on Single Machine

Algorithm. At time t, the machine maintains a speed $s(t) = \beta W(t)^{1/\alpha}$ where $W(t)$ is the total weight of pending jobs and β is a constant to be defined later. At any time, the machine shares its resource to pending jobs proportional to their weights.

Analysis. Let $s_j(t)$ be the variable that represents the speed of job j at time t for every job j. Let C_j be a variable representing the completion time of j. The problem could be relaxed as the following mathematical program.

$$\begin{aligned}
\text{minimize } & \int_0^\infty \Big(\sum_j s_j(t)\Big)^\alpha dt + \sum_j \Big(\int_{r_j}^{C_j} s_j(t)dt\Big)\delta_j(C_j - r_j) \\
\text{subject to } & \int_{r_j}^{C_j} s_j(t)dt \geq p_j \qquad \forall j \\
& s_j(t) \geq 0 \qquad \forall j, t \geq r_j \\
& s_j(t) = 0 \qquad \forall j, \forall t \notin [r_j, C_j].
\end{aligned}$$

The dual of that program is $\max \min_{s,C} L$ where L is the associated Lagrangian function. Set all dual variables except the ones corresponding to the first constraints equal 0, the dual becomes

$$\begin{aligned}
\min_{s,C} L &= \int_0^\infty \Big(\sum_j s_j(t)\Big)^\alpha dt + \sum_j \int_{r_j}^{C_j} \delta_j(C_j - r_j)s_j(t)dt + \sum_j \lambda_j \Big(p_j - \int_{r_j}^{C_j} s_j(t)dt\Big) \\
&= \sum_j \lambda_j p_j - \max_{s,C} \sum_j \int_{r_j}^{C_j} s_j(t)\Big(\lambda_j - s(t)^{\alpha-1} - \delta_j(C_j - r_j)\Big)dt
\end{aligned}$$

where the speed of the machine $s(t) = \sum_j s_j(t)$.

Dual variables. Choose λ_j such that $\frac{\alpha+\beta(\alpha-1)}{\beta(\alpha-1)}\lambda_j p_j$ equals the marginal increase of the total weighted flow-time due to the arrival of job j.

For simplicity of the notations, denote $q_k = q_k(r_j)$ for every pending job k. Note that $q_j = p_j$. At r_j, rename jobs in non-increasing order of their residual densities, i.e., $q_1/w_1 \leq \ldots \leq q_n/w_n$ (note that q_k/w_k is the inverse of job k's residual density). Denote $W_k = w_k + \ldots + w_n$ for $1 \leq k \leq n$. Assuming that no new job arrives after r_j, by the algorithm we have

$$\begin{aligned}
\beta(C_1 - r_j) &= \frac{q_1}{\frac{w_1}{W_1}W_1^{1/\alpha}} = W_1^{\frac{\alpha-1}{\alpha}}\frac{q_1}{w_1} \\
\beta(C_2 - C_1) &= W_2^{\frac{\alpha-1}{\alpha}}\left(\frac{q_2}{w_2} - \frac{q_1}{w_1}\right) \\
&\ldots \\
\beta(C_k - C_{k-1}) &= W_k^{\frac{\alpha-1}{\alpha}}\left(\frac{q_k}{w_k} - \frac{q_{k-1}}{w_{k-1}}\right)
\end{aligned} \tag{1}$$

Hence, assuming that no new jobs are released after r_j, we have

$$C_k - r_j = \frac{1}{\beta}\left[\frac{q_k}{w_k}W_k^{\frac{\alpha-1}{\alpha}} + \left(W_{k-1}^{\frac{\alpha-1}{\alpha}} - W_k^{\frac{\alpha-1}{\alpha}}\right)\frac{q_{k-1}}{w_{k-1}} + \ldots + \left(W_1^{\frac{\alpha-1}{\alpha}} - W_2^{\frac{\alpha-1}{\alpha}}\right)\frac{q_1}{w_1}\right] \tag{2}$$

Lemma 4 *We have that $\lambda_j \leq \delta_j(C_j - r_j)$ where C_j is the completion time of job j assuming that no new job is released after r_j.*

Proof Let C_k for $1 \leq k \leq n$ be the completion time of pending jobs at time r_j assuming no new job arrives after that time. Let C'_k be the completion time of job $k \neq j$ without job j in the instance.

In other words, C'_k is the completion time of job k in case j is not released. First, observe that function $(a+x)^{\frac{\alpha-1}{\alpha}} - (b+x)^{\frac{\alpha-1}{\alpha}}$ is decreasing for $a \geq b > 0$ and $x \geq 0$ (the derivative of the function is negative). Using that inequality and (2), we have

$$\beta(C_k - C'_k) \leq \begin{cases} \left(W_k^{\frac{\alpha-1}{\alpha}} - (W_k - w_j)^{\frac{\alpha-1}{\alpha}} \right) \frac{q_k}{w_k} & \forall k < j \\ \left(W_j^{\frac{\alpha-1}{\alpha}} - W_{j+1}^{\frac{\alpha-1}{\alpha}} \right) \frac{q_j}{w_j} & \forall k > j \end{cases}$$

Note that $1 - \beta x \geq (1-x)^\beta \geq 1 - x$ for $\beta < 1$ and $0 \leq x \leq 1$. Applying these inequalities for $x = w_j/W_k$ and $x = w_k/W_k$ where $k < j$ and $\beta = (\alpha-1)/\alpha$, we have

$$\frac{1 - \left(1 - \frac{w_j}{W_k}\right)^{\frac{\alpha-1}{\alpha}}}{1 - \left(1 - \frac{w_k}{W_k}\right)^{\frac{\alpha-1}{\alpha}}} \leq \frac{\alpha}{\alpha-1} \frac{w_j}{w_k}$$

$$\Rightarrow \quad \left[W_k^{\frac{\alpha-1}{\alpha}} - (W_k - w_j)^{\frac{\alpha-1}{\alpha}} \right] q_k \leq \frac{\alpha}{\alpha-1} \left[W_k^{\frac{\alpha-1}{\alpha}} - (W_k - w_k)^{\frac{\alpha-1}{\alpha}} \right] \frac{q_k}{w_k} w_j$$

$$\Rightarrow \quad w_k(C_k - C'_k) \leq \frac{\alpha}{\beta(\alpha-1)} w_j \left[W_k^{\frac{\alpha-1}{\alpha}} - (W_k - w_k)^{\frac{\alpha-1}{\alpha}} \right] \frac{q_k}{w_k}$$

Besides, $W_j^{\frac{\alpha-1}{\alpha}} W_{j+1}^{\frac{1}{\alpha}} - W_{j+1} \leq W_j - W_{j+1} = w_j$. So $W_j^{\frac{\alpha-1}{\alpha}} - W_{j+1}^{\frac{\alpha-1}{\alpha}} \leq w_j / W_{j+1}^{\frac{1}{\alpha}}$. Together with the inequality above, the marginal increase of the total weighted flowtime due to the arrival of job j is

$$\begin{aligned} & w_j(C_j - r_j) + \sum_{k<j} w_k(C_k - C'_k) + \sum_{k>j} w_k(C_k - C'_k) \\ & \leq w_j(C_j - r_j) + \sum_{k<j} \frac{\alpha}{\beta(\alpha-1)} w_j \left[W_k^{\frac{\alpha-1}{\alpha}} - (W_k - w_k)^{\frac{\alpha-1}{\alpha}} \right] \frac{q_k}{w_k} + \frac{1}{\beta} \sum_{k>j} w_k \frac{w_j}{W_{j+1}^{\frac{1}{\alpha}}} \frac{p_j}{w_j} \\ & \leq w_j(C_j - r_j) + w_j \sum_{k<j} \frac{\alpha}{\beta(\alpha-1)} \left[W_k^{\frac{\alpha-1}{\alpha}} - (W_k - w_k)^{\frac{\alpha-1}{\alpha}} \right] \frac{q_k}{w_k} + \frac{1}{\beta} w_j W_{j+1}^{\frac{\alpha-1}{\alpha}} \frac{p_j}{w_j} \\ & \leq w_j(C_j - r_j) + \frac{\alpha}{\beta(\alpha-1)} w_j(C_j - r_j) = \frac{\alpha + \beta(\alpha-1)}{\beta(\alpha-1)} w_j(C_j - r_j) \end{aligned}$$

where the last inequality is due to (2). By the definition of λ_j, the lemma follows. □

Monotonicity. Consider two instances $\mathcal{I}$ and $\mathcal{I}'$ such that they are identical except that there is only a job $j \in \mathcal{I} \setminus \mathcal{I}'$. Moreover, assume that no job is released r_j in either of the instances, i.e., $r_j = \max\{r_k : k \in \mathcal{I}\} = \max\{r_k : k \in \mathcal{I}'\}$.

Lemma 5 *$W(t)$ is monotone, i.e., $W^{\mathcal{I}'}(t) \leq W^{\mathcal{I}}(t)$ for all t.*

Proof For $t < r_j$, the schedules of both instances are identical. We are interested in $t \geq r_j$. For simplicity of the notations, denote $q_k = q_k(r_j)$ for every pending job k. Again, rename jobs in

non-increasing order of their residual densities, i.e., $q_1/w_1 \le \ldots \le q_n/w_n$ (note that q_k/w_k is the inverse of job k's residual density). Denote $W_k = \sum_{u=k}^{n} w_u$ and $W'_k = \sum_{u=k,u\neq j}^{n} w_u$ for $1 \le k \le n$. Let C_k and C'_k be the completion time of job $k \neq j$ for $1 \le k \le n$ in the schedules of instances $\mathcal{I}$ and $\mathcal{I}'$, respectively. By the algorithm, we have an invariant that $C_u < C_v$ and $C'_u < C'_v$ for any jobs $u < v$. In order to prove that $W^{\mathcal{I}'}(t) \le W^{\mathcal{I}}(t)$ for all t, it is sufficient to show that $C'_k < C_k$ for $1 \le k \neq j \le n$.

By (1), for any job $k < j$, it holds that $C_{k'} < C_k$ since $W'_u < W_u$ for $1 \le u \le k$. Consider a job $k > j$. As $W_j > W_{j+1}$ and $p_j/w_j \ge p_{j-1}/w_{j-1}$, we have

$$\beta(C_{j+1} - C_{j-1}) = W_j^{\frac{\alpha-1}{\alpha}} \left(\frac{q_j}{w_j} - \frac{q_{j-1}}{w_{j-1}}\right) + W_{j+1}^{\frac{\alpha-1}{\alpha}} \left(\frac{q_{j+1}}{w_{j+1}} - \frac{q_j}{w_j}\right) > W_{j+1}^{\frac{\alpha-1}{\alpha}} \left(\frac{q_{j+1}}{w_{j+1}} - \frac{q_{j-1}}{w_{j-1}}\right)$$

Therefore, by (1) and the above inequality,

$$\begin{aligned} C'_u - C'_{u-1} &\le C_u - C_{u-1} \qquad \forall 1 \le u \le j-1 \\ C'_{j+1} - C'_{j-1} &< C_{j+1} - C_{j-1} \\ C'_u - C'_{u-1} &= C_u - C_{u-1} \qquad \forall u \ge j+2 \end{aligned}$$

where conventionally $C_0 = r_j$. We deduce that $C'_k < C_k$ for $k > j$. The lemma follows. □

Lemma 6 *It holds that $\lambda_j - \delta_j(t - r_j) \le \frac{1}{\beta} W(t)^{(\alpha-1)/\alpha}$ for all t.*

Proof By the monotonicity of $W(t)$, it is sufficient to prove the inequality assuming that no new job arrives after r_j. For simplicity of the notations, denote again $q_k = q_k(r_j)$ for every pending job k. At r_j, rename jobs in non-increasing order of their residual densities, i.e., $q_1/w_1 \le \ldots \le q_n/w_n$ (note that q_k/w_k is the inverse of job k's density). Denote $W_k = w_k + \ldots + w_n$ for $1 \le k \le n$. Suppose that at time $t < C_j$, job u is the pending one with the smallest index. In other words, $C_{u-1} \le t < C_u$. Note that $[r_j, t) \supset [r_j, C_1) \cup [C_1, C_2) \cup \ldots \cup [C_{u-2}, C_{u-1})$. By (1) we get

$$\beta(t - r_j) \ge \frac{q_{u-1}}{w_{u-1}} W_{u-1}^{\frac{\alpha-1}{\alpha}} + \left(W_{u-2}^{\frac{\alpha-1}{\alpha}} - W_{u-1}^{\frac{\alpha-1}{\alpha}}\right) \frac{q_{u-2}}{w_{u-2}} + \ldots + \left(W_1^{\frac{\alpha-1}{\alpha}} - W_2^{\frac{\alpha-1}{\alpha}}\right) \frac{q_1}{w_1}$$

By Lemma 4 and the above inequality, it holds that

$$\begin{aligned} \lambda_j - \delta_j(t - r_j) &\le \delta_j((C_j - r_j) - (t - r_j)) \\ &\le \frac{\delta_j}{\beta} \left[\frac{q_j}{w_j} W_j^{\frac{\alpha-1}{\alpha}} + \left(W_{j-1}^{\frac{\alpha-1}{\alpha}} - W_j^{\frac{\alpha-1}{\alpha}}\right) \frac{q_{j-1}}{w_{j-1}} + \ldots + \left(W_u^{\frac{\alpha-1}{\alpha}} - W_{u+1}^{\frac{\alpha-1}{\alpha}}\right) \frac{q_u}{w_u}\right] \\ &\le \frac{1}{\beta} W_u^{\frac{\alpha-1}{\alpha}} = \frac{1}{\beta} W(t)^{(\alpha-1)/\alpha} \end{aligned}$$

where the last inequality follows since $q_u/w_u \le \ldots \le q_j/w_j = 1/\delta_j$. □

Theorem 3 *The algorithm is 2^α-competitive for $\beta = 2$.*

Proof Let $\mathcal{F}^*$ be the total weighted flow-time due to the algorithm. By the choice of dual variables, we have

$$\begin{aligned}
\min_{s,C} L &\geq \sum_j \lambda_j p_j - \max_{s,C} \sum_j \int_{r_j}^{C_j} s_j(t)\Big(\lambda_j - s(t)^{\alpha-1} - \delta_j(C_j - r_j)\Big)dt \\
&\geq \frac{\beta(\alpha-1)}{\alpha+\beta(\alpha-1)}\mathcal{F}^* - \max_{s,C} \sum_j \int_{r_j}^{C_j} s_j(t)\Big(\lambda_j - s(t)^{\alpha-1} - \delta_j(t - r_j)\Big)dt \\
&\geq \frac{\beta(\alpha-1)}{\alpha+\beta(\alpha-1)}\mathcal{F}^* - \max_{s,C} \int_0^\infty \Big(\sum_j s_j(t)\Big)\Big(\frac{1}{\beta}W(t)^{\frac{\alpha-1}{\alpha}} - s(t)^{\alpha-1}\Big)dt \\
&\geq \frac{\beta(\alpha-1)}{\alpha+\beta(\alpha-1)}\mathcal{F}^* - \max_{s,C} \int_0^\infty s(t)\Big(\frac{1}{\beta}W(t)^{\frac{\alpha-1}{\alpha}} - s(t)^{\alpha-1}\Big)dt
\end{aligned}$$

where the second inequality holds since $t \leq C_j$; the third inequality is due to Lemma 6. By the first order condition, $s(t)\Big(\frac{1}{\beta}W(t)^{\frac{\alpha-1}{\alpha}} - s(t)^{\alpha-1}\Big)$ is maximized at $s_0(t) = \frac{W(t)^{1/\alpha}}{(\alpha\beta)^{1/(\alpha-1)}}$. Hence,

$$\begin{aligned}
\min_{s,C} L &\geq \frac{\beta(\alpha-1)}{\alpha+\beta(\alpha-1)}\mathcal{F}^* - \frac{\alpha-1}{(\alpha\beta)^{\frac{\alpha}{\alpha-1}}}\int_0^\infty W(t)dt \\
&= \left[\frac{\beta(\alpha-1)}{\alpha+\beta(\alpha-1)} - \frac{\alpha-1}{(\alpha\beta)^{\frac{\alpha}{\alpha-1}}}\right]\mathcal{F}^*
\end{aligned}$$

Besides, the primal value is

$$\mathcal{F}^* + \int_0^\infty s^\alpha(t)dt = (1+\beta^\alpha)\mathcal{F}^*$$

Hence, the competitive ratio is bounded by

$$\frac{1+\beta^\alpha}{\frac{\beta(\alpha-1)}{\alpha+\beta(\alpha-1)} - \frac{\alpha-1}{(\alpha\beta)^{\frac{\alpha}{\alpha-1}}}}.$$

Choose $\beta = 2$, the competitive ratio is $O(2^\alpha)$. □

4.2 Clairvoyant Scheduling on Unrelated Machines

Scheduling policy. At any time t, every machine i sets its speed $s_i(t) = \beta W_i(t)^{1/\alpha}$ where $W_i(t)$ is the total (integral) weight of pending jobs assigned to machine i; and $\beta > 0$ is a constant to be chosen later. At any time, every machine i processes the highest residual density job among the pending ones assigned to i.

Assignment policy. At the arrival of a job j, assign j to machine i that minimizes the marginal increase (due to the scheduling policy) of the total weighted flow-time.

Analysis. Let $s_{ij}(t)$ be the variable that represents the speed of job j on machine i at time t. Let C_j be a variable representing the completion time of j. Let x_{ij} be the variable indicating whether job j is assigned to machine i. The problem could be relaxed as the following program.

$$\begin{aligned}
\text{minimize } & \sum_i \int_0^\infty \Big(\sum_j s_{ij}(t)\Big)^\alpha dt + \sum_{i,j}\Big(\int_{r_j}^{C_j} s_{ij}(t)dt\Big)\delta_{ij}x_{ij}(C_j - r_j) \\
& \qquad + \frac{\alpha}{\beta(\alpha-1)}\sum_{i,j}\Big(\int_{r_j}^{C_j} s_{ij}(t)dt\Big)x_{ij}w_{ij}^{\frac{\alpha-1}{\alpha}} \\
\text{subject to } \quad & x_{ij}\int_{r_j}^{C_j} s_{ij}(t)dt = p_{ij}x_{ij} \qquad \forall j \\
& \sum_i x_{ij} \geq 1 \qquad \forall j \\
& x_{ij} \in \{0,1\} \qquad \forall i,j \\
& s_{ij}(t) \geq 0 \qquad \forall i,j,t \geq r_j \\
& s_{ij}(t) = 0 \qquad \forall j, \forall t \notin [r_j, C_j].
\end{aligned}$$

The third term in the objective is inspired by the objective functions in [1]. The following lemma show that the objective value of any feasible schedule is within a constant factor of the *cost* of the schedule, which is the sum of the weighted flow-time and the energy consumed. The proof again follows the one in [1].

Lemma 7 *Consider a non-migratory schedule $\mathcal{S}$ for an instance $\mathcal{I}$ of the problem. Let x_{ij} and $s_{ij}(t)$ be the corresponding solution to the mathematical program. Then the objective value of such solution for the mathematical program is at most $(1 + \frac{\alpha}{\beta(\alpha-1)})$ the cost of $\mathcal{S}$.*

Proof In the objective function, the first and second terms capture the consumed energy and the total weighted flow-time, respectively. We will show that the last term is bounded by $\frac{\alpha}{\beta(\alpha-1)}$ the cost of $\mathcal{S}$.

Suppose that $\mathcal{S}$ schedules j on machine i and completes j at time T. Then the average speed $\bar{s}_i$ of i during $[r_j, T]$ is at least $p_{ij}/(T - r_j)$. Thus, $T - r_j \geq p_{ij}/\bar{s}_i$. The total energy consumed to complete job j is at least $(T - r_j)\bar{s}_i^\alpha \geq p_{ij}\bar{s}_i^{\alpha-1}$. Therefore, the contribution of j to the cost of $\mathcal{S}$ is at least

$$\begin{aligned}
w_{ij}(T - r_j) + p_{ij}\bar{s}_i^{\alpha-1} &\geq w_{ij}p_{ij}/\bar{s}_i + p_{ij}\bar{s}_i^{\alpha-1} \\
&\geq p_{ij}w_{ij}^{\frac{\alpha-1}{\alpha}}\left((\alpha-1)^{\frac{1}{\alpha}} + (\alpha-1)^{-\frac{\alpha-1}{\alpha}}\right) \geq p_{ij}w_{ij}^{\frac{\alpha-1}{\alpha}}
\end{aligned}$$

where the second inequality is due to the first order condition. Again, as the energy function is convex, the total energy consumed of a schedule is larger than the sum of energy consumed on each individual job. Summing the above inequality for all jobs j, we deduce that the third term in the objective function is bounded by factor $\frac{\alpha}{\beta(\alpha-1)}$ the cost of $\mathcal{S}$. □

The dual of that program is $\max\min_{x,s,C} L$ where L is the Lagrangian associated to the above mathematical program. Set all dual variables except the ones corresponding to the first constraints

equal to 0, the Lagrangian becomes

$$\sum_i \int_0^\infty \Big(\sum_j s_{ij}(t)\Big)^\alpha dt + \sum_j \int_{r_j}^{C_j} \delta_{ij}(C_j - r_j) x_{ij} s_{ij}(t) dt$$
$$+ \frac{\alpha}{\beta(\alpha-1)} \sum_{i,j} \Big(\int_{r_j}^{C_j} s_{ij}(t)dt\Big) x_{ij} w_{ij}^{\frac{\alpha-1}{\alpha}} + \sum_{i,j} \lambda_{ij} x_{ij} \Big(p_{ij} - \int_{r_j}^{C_j} s_{ij}(t)dt\Big)$$

Hence,

$$\min_{x,s,C} L = \min_x \sum_{i,j} \lambda_{ij} p_{ij} x_{ij} - \max_{x,s,C} \sum_{i,j} \int_{r_j}^{C_j} x_{ij} s_{ij}(t) \Big(\lambda_{ij} - s_i(t)^{\alpha-1} - \frac{\alpha}{\beta(\alpha-1)} w_{ij}^{\frac{\alpha-1}{\alpha}} - \delta_{ij}(C_j - r_j)\Big) dt$$

Choose λ_{ij} such that $\lambda_{ij} p_{ij}$ equals the total increase of the weighted flow-times of jobs (different to j) assigned to machine i plus the weighted flow-time of job j if the latter is assigned to i.

Monotonicity. Consider two set of jobs $\mathcal{I}$ and $\mathcal{I}'$ assigned to machine i such that they are identical except that there is only a job $j \in \mathcal{I} \setminus \mathcal{I}'$. Moreover, assume that no job is released after time r_j in either of the instances. For every job k, define the *fractional* weight of k at time t as $w_k q_k(t)/p_k$. Let $V_i(t)$ be the total *fractional* weight of pending jobs assigned to machine i. The following lemma in [1] showed the monotonicity of $V_i(t)$.

Lemma 8 ([1]) *$V_i(t)$ is monotone for every machine i, i.e., $V_i^{\mathcal{I}'}(t) \le V_i^{\mathcal{I}}(t)$ for all i, t.*

Lemma 9 *It holds that $\lambda_{ij} - \delta_{ij}(t - r_j) - \frac{\alpha}{\beta(\alpha-1)} w_{ij}^{\frac{\alpha-1}{\alpha}} \le \frac{\alpha}{\beta(\alpha-1)} V_i(t)^{\frac{\alpha-1}{\alpha}}$ for all i, t.*

Proof By Lemma 8, $V_i(t)$ is non-decreasing function. Hence, it is sufficient to prove the inequality for a fixed machine i assuming that no new job is released after r_j. For simplicity of the notations, as machine i is fixed, we drop the index of the machines in all the parameters. Moreover, denote again $q_k = q_k(r_j)$ and $\delta_k = \delta_k(r_j)$ for every pending job k. At r_j, rename jobs in non-increasing order of their residual densities, i.e., $q_1/w_1 \le \ldots \le q_n/w_n$ (note that q_k/w_k is the inverse of job k's residual density). Denote $W_k = w_k + \ldots + w_n$ for $1 \le k \le n$. The marginal increase in the total weighted flow-time due to the arrival of job j is

$$w_j \Big(\frac{q_1}{\beta W_1^{1/\alpha}} + \ldots + \frac{q_j}{\beta W_j^{1/\alpha}}\Big) + W_{j+1} \frac{q_j}{\beta W_j^{1/\alpha}}$$

where the first term is the weighted flow-time of job j and the second one is the increase of the weighted flow-time of other jobs (note that only jobs with density smaller than that of j has their completion times increased). Let C_j^* be the completion time of job j if it is assigned to machine i. We consider different cases.

Case 1: $t \le C_j^*$. Let k be the pending job at t with the smallest index. In other words, the machine has processed all jobs $1, \ldots, k-1$ and a part of job k in interval $[r_j, t]$. By the definition

of λ_j, we have that

$$\begin{aligned}
\lambda_j - \delta_j(t-r_j) &= \delta_j\left(\frac{q_k(t)}{\beta W_k^{1/\alpha}} + \frac{q_{k+1}}{\beta W_{k+1}^{1/\alpha}} + \ldots + \frac{q_j}{\beta W_j^{1/\alpha}}\right) + \frac{W_{j+1}}{\beta W_j^{1/\alpha}} \\
&= \delta_j\left(\frac{w_k(t)}{\delta_k \beta W_k^{1/\alpha}} + \frac{w_{k+1}}{\delta_{k+1}\beta W_{k+1}^{1/\alpha}} + \ldots + \frac{w_j}{\delta_j \beta W_j^{1/\alpha}}\right) + \frac{W_{j+1}}{\beta W_j^{1/\alpha}} \\
&\leq \frac{1}{\beta}\left(\frac{w_k(t)}{W_k^{1/\alpha}} + \frac{w_{k+1}}{W_{k+1}^{1/\alpha}} + \ldots + \frac{w_j}{W_j^{1/\alpha}} + \frac{w_{j+1}}{W_{j+1}^{1/\alpha}} + \ldots + \frac{w_n}{W_n^{1/\alpha}}\right) \\
&\leq \frac{1}{\beta}\int_{w_n}^{V(t)+w_j} \frac{dz}{z^{1/\alpha}} \leq \frac{\alpha}{\beta(\alpha-1)}(V(t)+w_j)^{\frac{\alpha-1}{\alpha}} \leq \frac{\alpha}{\beta(\alpha-1)}\left(V(t)^{\frac{\alpha-1}{\alpha}} + w_j^{\frac{\alpha-1}{\alpha}}\right).
\end{aligned}$$

The second equality is due to the definition of the residual density. The first inequality follows since $\delta_j \leq \delta_{k'}$ for every job $k' \leq j$ and $W_j \geq W_{j+1} \geq \ldots \geq W_n$. The second inequality holds since function $z^{-1/\alpha}$ is decreasing. The last inequality holds because $(\alpha-1)/\alpha < 1$.

Case 2: $t > C_j^*$**.** Let k be the pending job at t with the smallest index. We have

$$\begin{aligned}
\lambda_j - \delta_j(t-r_j) &= \frac{W_{j+1}}{\beta W_j^{1/\alpha}} - \delta_j(t-C_j^*) = \frac{1}{\beta W_j^{1/\alpha}}\left(w_{j+1} + \ldots + w_n\right) - \delta_j(t-C_j^*) \\
&\leq \delta_{j+1}\frac{q_{j+1}}{\beta W_{j+1}^{1/\alpha}} + \ldots + \delta_n \frac{q_n}{\beta W_n^{1/\alpha}} - \delta_j(t-C_j^*) \\
&\leq \delta_k \frac{q_k(t)}{\beta W_k^{1/\alpha}} + \delta_{k+1}\frac{q_{k+1}}{\beta W_{k+1}^{1/\alpha}} + \ldots + \delta_n \frac{q_n}{\beta W_n^{1/\alpha}} \\
&= \delta_k \frac{w_k(t)}{\delta_k \beta W_k^{1/\alpha}} + \delta_{k+1}\frac{w_{k+1}}{\delta_{k+1}\beta W_{k+1}^{1/\alpha}} + \ldots + \delta_n \frac{w_n}{\delta_n \beta W_n^{1/\alpha}} \\
&\leq \frac{1}{\beta}\int_{w_n}^{V(t)} \frac{dz}{z^{1/\alpha}} \leq \frac{\alpha}{\beta(\alpha-1)} V(t)^{\frac{\alpha-1}{\alpha}}
\end{aligned}$$

where the first inequality holds since $W_j \geq W_{j+1} \geq \ldots \geq W_n$; the second inequality is due to $\delta_j \geq \delta_{k'}$ for every job $k' > j$.

Combining both cases, the lemma follows. □

Theorem 4 *The algorithm is* $8(1+\frac{\alpha}{\ln \alpha})$*-competitive for* $\beta = \frac{1}{\alpha-1}(\alpha - 1 + \ln(\alpha-1))^{\frac{\alpha-1}{\alpha}}$.

Proof Let $\mathcal{F}^*$ be the total weighted flow-time due to the algorithm. By the choice of dual variables, we have

$$\begin{aligned}
\min_{x,s,C} L &= \min_{x} \sum_{i,j} \lambda_{ij} p_{ij} x_{ij} - \max_{x,s,C} \sum_{i,j} \int_{r_j}^{C_j} x_{ij} s_{ij}(t) \left(\lambda_{ij} - s_i(t)^{\alpha-1} - \frac{1}{\beta} w_{ij}^{\frac{\alpha-1}{\alpha}} - \delta_j (C_j - r_j) \right) dt \\
&\geq \mathcal{F}^* - \max_{x,s,C} \sum_{i,j} \int_{r_j}^{C_j} x_{ij} s_{ij}(t) \left(\lambda_{ij} - s_i(t)^{\alpha-1} - \frac{1}{\beta} w_{ij}^{\frac{\alpha-1}{\alpha}} - \delta_j (t - r_j) \right) dt \\
&\geq \mathcal{F}^* - \max_{x,s,C} \sum_{i,j} \int_{r_j}^{C_j} x_{ij} s_{ij}(t) \left(\frac{\alpha}{\beta(\alpha-1)} V_i(t)^{\frac{\alpha-1}{\alpha}} - s_i(t)^{\alpha-1} \right) dt \\
&= \mathcal{F}^* - \max_{x,s,C} \sum_{i} \int_{0}^{\infty} \left(\sum_j x_{ij} s_j(t) \right) \left(\frac{\alpha}{\beta(\alpha-1)} V_i(t)^{\frac{\alpha-1}{\alpha}} - s_i(t)^{\alpha-1} \right) dt \\
&\geq \mathcal{F}^* - \max_{x,s,C} \sum_{i} \int_{0}^{\infty} s_i(t) \left(\frac{\alpha}{\beta(\alpha-1)} V_i(t)^{\frac{\alpha-1}{\alpha}} - s_i(t)^{\alpha-1} \right) dt
\end{aligned}$$

where the first inequality follows by the assignment policy (assign job j to machine i that minimizes $\lambda_{ij} p_{ij}$) and $t \leq C_j$; the second inequality is due to Lemma 9. By the first order condition, function $z(\frac{\alpha}{\beta(\alpha-1)} V^{\frac{\alpha-1}{\alpha}} - z^{\alpha-1})$ is maximized at $z_0 = \frac{V^{1/\alpha}}{((\alpha-1)\beta)^{1/(\alpha-1)}}$. We have

$$\begin{aligned}
\min_{x,s,C} L &\geq \mathcal{F}^* - \frac{\alpha-1}{((\alpha-1)\beta)^{\frac{\alpha}{\alpha-1}}} \sum_i \int_0^\infty V_i(t) dt \\
&\geq \mathcal{F}^* - \frac{\alpha-1}{((\alpha-1)\beta)^{\frac{\alpha}{\alpha-1}}} \sum_i \int_0^\infty W_i(t) dt = \left(1 - \frac{\alpha-1}{((\alpha-1)\beta)^{\frac{\alpha}{\alpha-1}}} \right) \mathcal{F}^*
\end{aligned}$$

where the second inequality holds since $V_i(t) \leq W_i(t)$ for every i and t. Besides, the total weighted flow-time plus energy is $\mathcal{F}^* + \int_0^\infty s^\alpha(t) dt = (1 + \beta^\alpha)\mathcal{F}^*$. Therefore the primal is bounded by $\left((1+\beta^\alpha) + \frac{\alpha}{\beta(\alpha-1)}(1+\beta^\alpha) \right) \mathcal{F}^*$ (Lemma 7). Thus, the competitive ratio is at most

$$\frac{(1+\beta^\alpha) + \frac{\alpha}{\beta(\alpha-1)}(1+\beta^\alpha)}{1 - \frac{\alpha-1}{((\alpha-1)\beta)^{\frac{\alpha}{\alpha-1}}}} \tag{3}$$

Choose $\beta = \frac{1}{\alpha-1}(\alpha - 1 + \ln(\alpha-1))^{\frac{\alpha-1}{\alpha}}$. Observe that

$$\left(1 + \frac{\ln(\alpha-1)}{\alpha-1} \right)^{\alpha-1} < e^{\ln(\alpha-1)} = \alpha - 1 \quad \Rightarrow (\alpha - 1 + \ln(\alpha-1))^{\alpha-1} < (\alpha-1)^\alpha \quad \Rightarrow \beta < 1$$

Moreover, $\beta > (\alpha-1)^{-1/\alpha}$. With the chosen β, the denominator of (3) becomes $\frac{\ln(\alpha-1)}{\alpha-1+\ln(\alpha-1)}$ and the nominator is bounded by 8 (since $\alpha^{-1/\alpha} < \beta < 1$ and $\alpha \geq 2$). Hence, the competitive ratio is at most $8(1 + \alpha/\ln\alpha)$. □

4.3 Extensions

Our algorithm works for a more general class of energy power functions (the next section would give an idea on this extension). Besides, the analyses could be generalized for broadcast scheduling with objective for minimizing weighted flow-time plus energy.

5 Arbitrary Cost Functions of Flow-time

The problem. In this section, we study the online scheduling on unrelated machines to minimize a general objective $\sum_{i,j} w_{ij} f(F_j)$ where f is a function with certain properties (described below). At the arrival time of a job, the scheduler has to immediately assign it to a machine. Jobs will be entirely processed on their machines and the migration of jobs across machines is not allowed. (In practice, it is not desirable to migrate jobs from a machine to others.)

Properties. $f(0) = f'(0) = 0$ and for any $\epsilon > 0$ arbitrarily small,

($\mathcal{P}1$) there exists a function $K_1(\epsilon)$ such that $f(z_1 + z_2) \leq \frac{1}{1-\epsilon} f(z_1) + K_1(\epsilon) f(z_2)\ \forall z_1, z_2 \geq 0$;

($\mathcal{P}2$) $f'(z)$ is non-decreasing. By this property, we can deduce that

$$\sum_{\ell=1}^{k} a_\ell f'(A_{\ell-1}) \leq f(A_k) \leq \sum_{i=\ell}^{k} a_\ell f'(A_\ell)$$

where $A_\ell = a_1 + \ldots + a_\ell$ and $a_\ell \geq 0$ for every $1 \leq \ell \leq k$.

($\mathcal{P}3$) there exists a function $K_2(\epsilon)$ such that $f'(z_1 + z_2) \leq \frac{1}{1-\epsilon} f'(z_1) + K_2(\epsilon) f'(z_2)\ \forall z_1, z_2 \geq 0$;

($\mathcal{P}4$) there exists a function $K_3(\epsilon)$ such that $f'(z + \frac{z}{K_3(\epsilon)}) \leq \frac{1}{1-\epsilon} f'(z)\ \forall z \geq 0$;

($\mathcal{P}5$) there exists a function $K_4 \geq 1$ such that $z f'(z) \leq K_4 f(z)\ \forall z \geq 0$.

Define $K(\epsilon) = \max\{K_1(\epsilon), 3K_2(\epsilon)K_3(\epsilon)K_4\}$

Scheduling policy. At time t, every machine i schedules the highest residual density job among the ones assigned to i.

Assignment policy. For a job j, recall that $q_{ij}(t)$ is the remaining processing time of j on machine i. Let $Q_j(t)$ be the remaining time of job j from t to its completion time by the algorithm. Let $U_i(t)$ be the set of jobs assigned to machine i and are still pending at t. At the arrival time r_j, job j is assigned to the machine i that minimize $\widetilde{\lambda}_{ij}$, which is defined as

$$\delta_{ij} f\Bigg(\sum_{\substack{u \in U_i(r_j) \\ \delta_u(r_j) \geq \delta_{ij}}} q_u(r_j) + p_{ij} \Bigg) + \sum_{\substack{u \in U_i(r_j) \\ \delta_u(r_j) < \delta_{ij}}} \frac{w_{iu}}{p_{ij}} \Big(f(Q_u(r_j) + p_{ij}) - f(Q_u(r_j)) \Big)$$

where δ_{ij} is the density of job j on machine i, i.e., $\delta_{ij} = \delta_{ij}(r_j)$. Note that $\widetilde{\lambda}_{ij} p_{ij}$ is the marginal increase of the objective function if job j is assigned to machine i.

Analysis. Let x_{ij} be the variable indicating whether job j is assigned to machine i. Besides, let $s_{ij}(t)$ be the rate that a machine i processes job j at time t. The problem could be relaxed as the following mathematical program where variable C_j represents the completion time of job j.

$$
\begin{aligned}
\text{minimize } & \sum_{i,j} \delta_{ij} x_{i,j} \int_{r_j}^{C_j} s_{ij}(t) \Big(f(C_j - r_j) + f(p_{ij}) \Big) dt \\
\text{subject to } \quad & x_{ij} \int_{r_j}^{C_j} s_{ij}(t) dt \geq p_{ij} x_{ij} \qquad \forall j \\
& \sum_j s_{ij}(t) \leq 1 \qquad \forall i, t \\
& \sum_i x_{ij} \geq 1 \qquad \forall j \\
& x_{ij} \in \{0,1\} \qquad \forall i, j \\
& s_{ij}(t) \geq 0 \qquad \forall j, t \geq r_j \\
& s_{ij}(t) = 0 \qquad \forall j, \forall t \notin [r_j, C_j].
\end{aligned}
$$

Note that if job j is assigned to machine i then $C_j - r_j \geq p_{ij}$. Thus, the objective function in the relaxation is bounded by $2 \sum_{i,j} \delta_{i,j} x_{i,j} \int_{r_j}^{C_j} s_{ij}(t) f(C_j - r_j)$ where the latter represents twice the objective of the problem.

The dual of that program is $\max \min_{x,s,C} L$ where L is the Lagrangian function associated to the above relaxation. Set all dual variables except the ones corresponding to the first constraints equal to 0, the Lagrangian function becomes

$$
\sum_{i,j} \delta_{ij} x_{ij} \int_{r_j}^{C_j} s_{ij}(t) \Big(f(C_j - r_j) + f(p_{ij}) \Big) dt + \sum_{i,j} \lambda_{ij} x_{ij} \Big(p_{ij} - \int_{r_j}^{C_j} s_{ij}(t) dt \Big)
$$

Hence, the Lagragian dual is

$$
\max \min_{x,s,C} L \geq \min_{x,s,C} \sum_{i,j} \lambda_{ij} p_{ij} x_{ij} - \sum_{i,j} x_{ij} \int_{r_j}^{C_j} s_{ij}(t) \Big(\lambda_{ij} - \delta_{ij} f(C_j - r_j) - \delta_{ij} f(p_{ij}) \Big) dt \qquad (4)
$$

Dual variables. Choose λ_{ij} such that $\lambda_{ij} := \frac{(1-\epsilon)^2}{K(\epsilon)} \widetilde{\lambda}_{ij}$.

Lemma 10 *It holds that* $\lambda_{ij} - \delta_{ij} f(t - r_j) - \delta_{ij} f(p_{ij}) \leq \frac{1}{K(\epsilon)} \sum_{u \in U_i(t)} w_u f'(Q_u(t))$ *for every* $t \geq r_j$.

The proof of the lemma is in the appendix.

Theorem 5 *The algorithm is* $\frac{1}{1-3\epsilon}$*-speed and* $\frac{2K(\epsilon)}{\epsilon}$*-competitive.*

Proof Let $\mathcal{F}^*$ be the objective value $\sum_{i,j} w_{ij} f(F_j)$ of the algorithm and let C_j^* be the completion time of job j due to the algorithm. Recall that $\frac{K(\epsilon)}{(1-\epsilon)^2} \lambda_{ij} p_{ij}$ is the marginal contribution of job j to

the objective value if it is assigned to machine i. By Lemma 11, we have

$$\begin{aligned}
&\sum_{i,j}\int_{r_j}^{C_j} x_{ij}s_{ij}(t)\Big(\lambda_{ij}-\delta_{ij}f(C_j-r_j)-\delta_{ij}f(p_{ij})\Big)dt\\
&\quad\le \sum_{i,j}\int_{r_j}^{C_j} x_{ij}s_{ij}(t)\Big(\lambda_{ij}-\delta_{ij}f(t-r_j)-\delta_{ij}f(p_{ij})\Big)dt\\
&\quad\le \frac{1}{K(\epsilon)}\sum_i\sum_j\int_{r_j}^{C_j} x_{ij}s_{ij}(t)\Big(\sum_{u\in U_i(t)} w_{iu}f'(Q_u(t))\Big)dt\\
&\quad= \frac{1}{K(\epsilon)}\sum_i\sum_u\int_0^{\infty}\Big(\sum_j x_{ij}s_{ij}(t)\Big)w_{iu}f'(Q_u(t))\cdot\mathbf{1}_{\{Q_u(t)>0\}}dt\\
&\quad= \frac{1}{K(\epsilon)}\sum_i\sum_u\int_{r_u}^{C_u^*}\Big(\sum_j x_{ij}s_{ij}(t)\Big)w_{iu}f'(Q_u(t))dt\\
&\quad= \frac{1}{K(\epsilon)}\sum_i\sum_u\int_{r_u}^{C_u^*}\Big(\sum_j x_{ij}s_{ij}(t)\Big)w_{iu}f'(C_u^*-t)dt\\
&\quad\le \frac{1-3\epsilon}{K(\epsilon)}\sum_{i,u} w_{iu}f(C_u^*-r_u)
\end{aligned}$$

In the first equality, observe that for any job $u\in U_i(t)$, it means that $Q_u(t)>0$. The last inequality follows by the fact that the adversary can only schedule with rate at most $(1-3\epsilon)$.

Therefore, we deduce

$$\begin{aligned}
\max\min_{x,s,C} L &\overset{(4)}{\ge} \min_{x,s,C}\sum_{i,j}\lambda_{ij}p_{ij}x_{ij}-\sum_{i,j}x_{ij}\int_{r_j}^{C_j} s_{ij}(t)\Big(\lambda_{ij}-\delta_{ij}f(C_j)-\delta_{ij}f(p_{ij})\Big)dt\\
&\ge \frac{(1-\epsilon)^2}{K(\epsilon)}\mathcal{F}^*-\frac{1-3\epsilon}{K(\epsilon)}\mathcal{F}^*\ge\frac{\epsilon}{K(\epsilon)}\mathcal{F}^*
\end{aligned}$$

where the second inequality follows since by the algorithm, every job j is assigned to machine i with minimal λ_{ij}, so $\sum_{ij}\lambda_{ij}x_{ij}\ge\frac{(1-\epsilon)^2}{K(\epsilon)}\mathcal{F}^*$. Hence, the competitive ratio is at most $2K(\epsilon)/\epsilon$. □

Corollary 1 *The algorithm is $\frac{1}{1-3\epsilon}$-speed $O(\frac{k}{\epsilon^{1+1/k}})$-competitive for the objective of weighted ℓ_k-norm of flow-time.*

Proof If $f(z)=z^k$ then by simple estimations, $K_1(\epsilon)=\left(\frac{k}{\epsilon}\right)^k$, $K_2(\epsilon)=\left(\frac{k-1}{\epsilon}\right)^{k-1}$, $K_3(\epsilon)=O(k/\epsilon)$ and $K_4=k-1$. Thus, $K(\epsilon)=O\left(\frac{k^{k+1}}{\epsilon^k}\right)$. Applying the above theorem, the competitive ratio of the problem with objective of weighted ℓ_k-norm of flow-time is $O(\frac{k^{(k+1)/k}}{\epsilon^{1+1/k}})=O(\frac{k}{\epsilon^{1+1/k}})$. □

5.1 Extensions

We have considered the problem in the model of non-migration of jobs accros machines. However, the same algorithm is still competitive even migration is allowed. The analysis remains essentially the same. Besides, the objective function could be generalized for different jobs j a function f_j. These functions could be generalized such that they only need to be in $\mathcal{C}^1$ almost everywhere.

6 Conclusion and Further Directions

In the paper, we have proved competitive algorithms in the resource augmentation/speed scaling models for different online scheduling problems using an unified approach. The approach is simple yet powerful in designing and analyzing algorithms. It seems to be a right tool to study problems in the resource augmentation/speed scaling models. Besides the extensions mentioned in previous sections, a future direction is to study online scheduling problems with the objectives of different nature, for example throughput-related objective. Moreover, different constraints might be incorporated, for example the bounded-speed model [7, 27] or the capacitated machine model [18].

An interesting future direction is to investigate different online problems with resource augmentation using the approach. Moreover, the min max game between algorithms and adversaries may give insights not only for designing algorithms but also for constructing counter-examples. In the latter, an adversary will choose variables such that the Lagrangian function has value far from optimum whatever the variable choices of any algorithm. That direction is worthy to study and that may gives ideas for seeking the *reverse engineer* goal described in [15]: can hard instances of problems be used to design algorithms?

References

[1] S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1228–1241, 2012.

[2] Yossi Azar, Leah Epstein, Yossi Richter, and Gerhard J. Woeginger. All-norm approximation algorithms. *J. Algorithms*, 52(2):120–133, 2004.

[3] Nikhil Bansal and Ho-Leung Chan. Weighted flow time does not admit $o(1)$-competitive algorithms. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms*, pages 1238–1244, 2009.

[4] Nikhil Bansal and Kirk Pruhs. Server scheduling in the weighted ℓ_p norm. In *Proc. 6th Latin American Symposium on Theoretical Informatics*, pages 434–443, 2004.

[5] Nikhil Bansal and Kirk Pruhs. The geometry of scheduling. In *Proc. 51th Symposium on Foundations of Computer Science*, pages 407–414, 2010.

[6] Nikhil Bansal and Kirk Pruhs. Weighted geometric set multi-cover via quasi-uniform sampling. In *Proc. 20th European Symposium on Algorithms*, pages 145–156, 2012.

[7] Nikhil Bansal, Ho-Leung Chan, Tak Wah Lam, and Lap-Kei Lee. Scheduling for speed bounded processors. In *Proc. 35th Colloquium on Automata, Languages and Programming*, pages 409–420, 2008.

[8] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs. Speed scaling with an arbitrary power function. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms*, pages 693–701, 2009.

[9] Nikhil Bansal, Ravishankar Krishnaswamy, and Viswanath Nagarajan. Better scalable algorithms for broadcast scheduling. In *Proc. 37th Colloquium on Automata, Languages and Programming*, pages 324–335, 2010.

[10] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2009.

[11] Jivitej S. Chadha, Naveen Garg, Amit Kumar, and V. N. Muralidhara. A competitive algorithm for minimizing weighted flow time on unrelatedmachines with speed augmentation. In *Proc. 41st ACM Symposium on Theory of Computing*, pages 679–684, 2009.

[12] Ho-Leung Chan, Jeff Edmonds, Tak Wah Lam, Lap-Kei Lee, Alberto Marchetti-Spaccamela, and Kirk Pruhs. Nonclairvoyant speed scaling for flow and energy. *Algorithmica*, 61(3):507–517, 2011.

[13] Sze-Hang Chan, Tak Wah Lam, Lap-Kei Lee, Hing-Fung Ting, and Pan Zhang. Nonclairvoyant scheduling for weighted flow time and energy on speed bounded processors. *Chicago J. Theor. Comput. Sci.*, 2011, 2011.

[14] Chandra Chekuri, Sanjeev Khanna, and An Zhu. Algorithms for minimizing weighted flow time. In *Proc. 33rd ACM Symposium on Theory of Computing*, pages 84–93, 2001.

[15] Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In *Proc. 44th ACM Symposium on Theory of Computing*, pages 137–144, 2012.

[16] Jeff Edmonds. Scheduling in the dark. *Theor. Comput. Sci.*, 235(1):109–141, 2000.

[17] Jeff Edmonds and Kirk Pruhs. Scalably scheduling processes with arbitrary speedup curves. *ACM Transactions on Algorithms*, 8(3):28, 2012.

[18] Kyle Fox and Madhukar Korupolu. Weighted flowtime on capacitated machines. In *Proc. 24th ACM-SIAM Symposium on Discrete Algorithms*, pages 129–143, 2013.

[19] Naveen Garg and Amit Kumar. Minimizing average flow-time : Upper and lower bounds. In *Proc. 48th Symposium on Foundations of Computer Science*, pages 603–613, 2007.

[20] Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *Proc. 10th Workshop on Approximation and Online Algorithms*, pages 173–186, 2012.

[21] Sungjin Im. *Online Scheduling Algorithms for Average Flow Time and its Variants*. PhD thesis, University of Illinois at Urbana-Champaign, 2012.

[22] Sungjin Im, Benjamin Moseley, and Kirk Pruhs. A tutorial on amortized local competitiveness in online scheduling. *SIGACT News*, 42(2):83–97, 2011.

[23] Sungjin Im, Benjamin Moseley, and Kirk Pruhs. Online scheduling with general cost functions. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1254–1265, 2012.

[24] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4): 617–643, 2000.

[25] Peter Kling and Peter Pietrzyk. Profitable scheduling on multiple speed-scalable processors. In *Proc. 25th Symposium on Parallelism in Algorithms and Architectures*, 2013.

[26] V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. A unified approach to scheduling on unrelated parallel machines. *J. ACM*, 56(5), 2009.

[27] Tak Wah Lam, Lap-Kei Lee, Isaac Kar-Keung To, and Prudence W. H. Wong. Online speed scaling based on active job count to minimize flow plus energy. *Algorithmica*, 65(3):605–633, 2013.

[28] Rajeev Motwani, Steven Phillips, and Eric Torng. Non-clairvoyant scheduling. *Theor. Comput. Sci.*, 130(1):17–47, 1994.

Appendix

Lemma 1 (Weak duality) *Consider a possibly non-convex optimization problem*

$$p^* := \min_x f_0(x) \ : \quad f_i(x) \leq 0, \quad i = 1, \ldots, m.$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ *for* $0 \leq i \leq m$. *Let* $\mathcal{X}$ *be the feasible set of* x. *Let* $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ *be the Lagragian function*

$$L(x, \lambda) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x).$$

Define $d^* = \max_{\lambda \geq 0} \min_{x \in \mathcal{X}} L(x, \lambda)$ *where* $\lambda \geq 0$ *means* $\lambda \in \mathbb{R}^m_+$. *Then* $p^* \geq d^*$.

Proof We observe that, for every feasible $x \in \mathcal{X}$, and every $\lambda \geq 0$, $f_0(x)$ is bounded below by $L(x, \lambda)$:

$$\forall x \in \mathcal{X}, \ \forall \lambda \geq 0 : \ f_0(x) \geq L(x, \lambda)$$

Define a function $g : \mathbb{R}^m \to \mathbb{R}$ such that

$$g(\lambda) := \min_z L(z, \lambda) = \min_z f_0(z) + \sum_{i=1}^{m} \lambda_i f_i(z)$$

As g is defined as a point-wise minimum, it is a concave function.

We have, for any x and λ, $L(x, \lambda) \geq g(\lambda)$. Combining with the previous inequality, we get

$$\forall x \in \mathcal{X} : \ f_0(x) \geq g(\lambda)$$

Taking the minimum over x, we obtain $\forall \lambda \geq 0 : \ p^* \geq g(\lambda)$. Therefore,

$$p^* \geq \max_{\lambda \geq 0} g(\lambda) = d^*.$$

□

5 Arbitrary Cost Functions of Flow-time

Lemma 11 *It holds that* $\lambda_{ij} - \delta_{ij} f(t - r_j) - \delta_{ij} f(p_{ij}) \leq \frac{1}{K(\epsilon)} \sum_{u \in U(t)} w_u f'(Q_u(t))$ *for every* $t \geq r_j$.

Let $U_i^1(t) \subset U_i(t)$ be the set of pending jobs u assigned to machine i and $\delta_{iu}(r_j) \geq \delta_{ij}$. Let $U_i^2(t) = U_i(t) \setminus U_i^1(t)$. Recall that

$$\frac{K(\epsilon)}{(1-\epsilon)^2}\lambda_{ij} = \delta_{ij} f\Bigg(\sum_{u \in U_i^1(r_j)} q_u(r_j) + p_{ij}\Bigg) + \sum_{u \in U_i^2(r_j)} \frac{w_{iu}}{p_{ij}} \Bigg(f(Q_u(r_j) + p_{ij}) - f(Q_u(r_j))\Bigg).$$

By increasing property of function f',

$$\begin{aligned}\sum_{u \in U_i^2(r_j)} \frac{w_{iu}}{p_{ij}} \Bigg(f(Q_u(r_j) + p_{ij}) - f(Q_u(r_j))\Bigg) &\leq \sum_{u \in U_i^2(r_j)} \frac{w_{iu}}{p_{ij}} p_{ij} f'(Q_u(r_j) + p_{ij}) \\ &= \sum_{u \in U_i^2(r_j)} w_{iu} f'(Q_u(r_j) + p_{ij})\end{aligned}$$

Hence,

$$\frac{K(\epsilon)}{(1-\epsilon)^2}\lambda_{ij} \leq \delta_{ij} f\Bigg(\sum_{u \in U_i^1(r_j)} q_u(r_j) + p_{ij}\Bigg) + \sum_{u \in U_i^2(r_j)} w_{iu} f'(Q_u(r_j) + p_{ij}) \tag{5}$$

To prove Lemma 11, observe that if some new job is assigned to machine i after r_j, then the right-hand side of the lemma inequality increases while the left-hand side remains unchanged (as λ_{ij} is unchanged). Hence, it is sufficient to prove Lemma 11 assuming that no new job is released after r_j. Let t_0 be the completion time of job j if j is scheduled in machine i by the algorithm. Lemma 11 follows Lemma 12 and Lemma 13.

Lemma 12 *If no new job is released after* r_j *then* $\lambda_{ij} - \delta_{ij} f(t - r_j) - \delta_{ij} f(p_{ij}) \leq \frac{1}{K(\epsilon)} \sum_{u \in U_i(t)} w_{iu} f'(Q_u(t))$ *for every* $r_j \leq t \leq t_0$.

Proof We prove the inequality in the lemma for a fixed machine i. For simplicity, we drop the machine index i of all parameters and variables (which correspond to the ones on machine i). First, we prove the following claim.

Claim 1 *It holds that*

$$\sum_{u \in U^2(t)} w_u f'(Q_u(r_j) + p_j) \leq \frac{1}{(1-\epsilon)^2} \left(\sum_{u \in U^2(t)} w_u f'(Q_u(t))\right) + \frac{K(\epsilon)}{1-\epsilon} \delta_j \Bigg(f(t - r_j) + f(p_j)\Bigg)$$

Proof of claim Observe that $U^2(r_j) = U^2(t)$ and $q_u(t) = q_u(r_j)$ for $u \in U^2(r_j)$ since no such job u is scheduled in interval $[r_j, t]$. Let V' be the set of jobs $u \in U^2(t)$ such that $t - r_j \leq \frac{1}{K_3(\epsilon)}(Q_u(t) + p_j)$.

Let $V'' = U^2(t) \setminus V'$. We have

$$\begin{aligned}
\sum_{u\in U^2(r_j)} w_u f'\Big(Q_u(r_j)+p_j\Big) &= \sum_{u\in U^2(r_j)} w_u f'\Big((t-r_j)+Q_u(t)+p_j\Big)\\
&= \sum_{u\in V'} w_u f'\Big((t-r_j)+Q_u(t)+p_j\Big) + \sum_{u\in V''} w_u f'\Big((t-r_j)+Q_u(t)+p_j\Big)\\
&\overset{(\mathcal{P}4)}{\le} \frac{1}{1-\epsilon}\sum_{u\in V'} w_u f'(Q_u(t)+p_j) + \sum_{u\in V''} w_u f'\Big(t-r_j+Q_u(t)+p_j\Big)\\
&\overset{(\mathcal{P}3)}{\le} \frac{1}{1-\epsilon}\sum_{u\in V'} w_u f'(Q_u(t)+p_j) + \frac{1}{1-\epsilon}\sum_{u\in V''} w_u f'(Q_u(t)+p_j) + K_2(\epsilon)\sum_{u\in V''} w_u f'(t-r_j)\\
&\le \frac{1}{1-\epsilon}\sum_{u\in V'\cup V''} w_u f'(Q_u(t)+p_j) + K_2(\epsilon)\delta_j \sum_{u\in V''} q_u(t) f'(t-r_j)\\
&\le \frac{1}{1-\epsilon}\sum_{u\in U^2(r_j)} w_u f'(Q_u(t)+p_j) + K_2(\epsilon)K_3(\epsilon)\delta_j(t-r_j)f'(t-r_j)\\
&\overset{(\mathcal{P}5)}{\le} \frac{1}{1-\epsilon}\sum_{u\in U^2(r_j)} w_u f'(Q_u(t)+p_j) + K_2(\epsilon)K_3(\epsilon)K_4\delta_j f(t-r_j). \qquad (6)
\end{aligned}$$

The third inequality follows because $\delta_j \ge \delta_u(r_j)$ for all $u \in U^2(t)$. The fourth inequality holds since $\sum_{u\in V''} q_u(t) \le \max_{u\in V''} Q_u(t) \le K_3(\epsilon)(t-r_j)$ (by definition of V'').

Let V be the set of jobs $u \in U^2(t)$ such that $p_j \le Q_u(t)/K_3(\epsilon)$. Similarly, we have

$$\begin{aligned}
\sum_{u\in U^2(t)} w_u f'(Q_u(t)+p_j) &= \sum_{u\in V} w_u f'(Q_u(t)+p_j) + \sum_{u\in U^2(t)\setminus V} w_u f'(Q_u(t)+p_j)\\
&\overset{(\mathcal{P}4)}{\le} \frac{1}{1-\epsilon}\sum_{u\in V} w_u f'(Q_u(t)) + \sum_{u\in U^2(t)\setminus V} w_u f'(Q_u(t)+p_j)\\
&\overset{(\mathcal{P}3)}{\le} \frac{1}{1-\epsilon}\left(\sum_{u\in U^2(t)} w_u f'(Q_u(t))\right) + K_2(\epsilon)\sum_{u\in U^2(t)\setminus V} w_u f'(p_j)\\
&\le \frac{1}{1-\epsilon}\left(\sum_{u\in U^2(t)} w_u f'(Q_u(t))\right) + K_2(\epsilon)\delta_j f'(p_j)\sum_{u\in U^2(t)\setminus V} q_u(t)\\
&< \frac{1}{1-\epsilon}\left(\sum_{u\in U^2(t)} w_u f'(Q_u(t))\right) + K_2(\epsilon)K_3(\epsilon)\delta_j f'(p_j)p_j\\
&\overset{(\mathcal{P}5)}{\le} \frac{1}{1-\epsilon}\left(\sum_{u\in U^2(t)} w_u f'(Q_u(t))\right) + K_2(\epsilon)K_3(\epsilon)K_4\delta_j f(p_j) \qquad (7)
\end{aligned}$$

where the third inequality follows since $\delta_u(r_j) \le \delta_j$ and $q_u(t) = q_u(r_j)$ for $u \in U^2(t) = U^2(r_j)$; the fourth inequality is due to the fact that $\sum_{u\in U_i^2(t)\setminus V} q_u(t)$ is bounded by the maximal $Q_u(t)$ for $u \in U_i^2(t) \setminus V$, which is bounded by $K_3(\epsilon)p_j$ (by definition of V).

Combining (7) and (6), we get

$$\sum_{u\in U^2(r_j)} w_u f'\Big(Q_u(r_j)+p_j\Big) \le \frac{K(\epsilon)}{1-\epsilon}\delta_j\Big(f(t-r_j)+f(p_j)\Big)+\frac{1}{(1-\epsilon)^2}\sum_{u\in U^2(t)} w_u f'(Q_u(t))$$

□

We are now proving the lemma. Observe that machine i schedules jobs with rate 1, it holds that

$$\sum_{u\in U^1(r_j)} q_u(r_j)+p_j = (t-r_j)+\sum_{u\in U^1(t)} q_u(t)+q_j(t)$$

Therefore,

$$\begin{aligned}
f\Big(\sum_{u\in U^1(r_j)} q_u(r_j)+p_j\Big) &= f\Big((t-r_j)+\sum_{u\in U^1(t)} q_u(t)+q_j(t)\Big)\\
&\overset{(\mathcal{P}1)}{\le} K_1(\epsilon)f(t-r_j)+\frac{1}{1-\epsilon}f\Big(\sum_{u\in U^1(t)} q_u(t)+q_j(t)\Big)\\
&\overset{(\mathcal{P}1)}{\le} K_1(\epsilon)f(t-r_j)+\frac{K_1(\epsilon)}{1-\epsilon}f(p_j)+\frac{1}{(1-\epsilon)^2}f\Big(\sum_{u\in U^1(t)} q_u(t)\Big)\\
&\overset{(\mathcal{P}2)}{\le} \frac{K_1(\epsilon)}{1-\epsilon}\Big(f(t-r_j)+f(p_j)\Big)+\frac{1}{(1-\epsilon)^2}\sum_{u\in U^1(t)} q_u(t)f'(Q_u(t)) \qquad (8)
\end{aligned}$$

Hence, using (5) and (8), we get

$$\begin{aligned}
\frac{K(\epsilon)}{(1-\epsilon)^2}\lambda_j &\le \frac{K_1(\epsilon)}{1-\epsilon}\delta_j\Big(f(t-r_j)+f(p_j)\Big)+\frac{1}{(1-\epsilon)^2}\sum_{u\in U^1(t)}\delta_j q_u(t)f'(Q_u(t))\\
&\quad+\sum_{u\in U^2(r_j)} w_u f'(Q_u(r_j)+p_j)\\
&\overset{\text{(Claim 1)}}{\le} \frac{2K(\epsilon)}{1-\epsilon}\delta_j\Big(f(t-r_j)+f(p_j)\Big)+\frac{1}{(1-\epsilon)^2}\sum_{u\in U^1(t)\cup U^2(t)} w_u f'(Q_u(t)).
\end{aligned}$$

where the second inequality is due to $\delta_u \ge \delta_j$ for every $u \in U^1(t)$. Rearranging the terms, the lemma follows (for $\epsilon \le 1/2$). □

Lemma 13 *If no new job is released after r_j then $\lambda_{ij}-\delta_{ij}f(t-r_j)-\delta_{ij}f(p_{ij}) \le \frac{1}{K(\epsilon)}\sum_{u\in U_i(t)} w_{iu}f'(Q_u(t))$ for every $t > t_0$.*

Proof Again we drop the machine index to simplify the notations. First we argue the following claim.

Claim 2 *It holds that*

$$\sum_{u\in U^2(r_j)} w_u f'(Q_u(r_j)) \le \frac{3K_2(\epsilon)K_3(\epsilon)K_4}{1-\epsilon}\delta_j f(t-r_j)-\frac{1}{1-\epsilon}\delta_j f(t_0-r_j)+\frac{1}{1-\epsilon}\sum_{u\in U(t)} w_u f'(Q_u(t))$$

Proof of claim Let $W = \{u_1, \ldots, u_a\} \subset U^2(r_j)$ be the set of jobs processed by the algorithm in interval $[t_0, t]$ where all jobs in W but probably u_a have been completed. It means that at time t, the machine is processing job a or has just completed job $(a-1)$. Hence, $U(t) = U^2(r_j) \setminus W \cup a$. For simplicity, denote $q_u = q_u(r_j)$ for all pending jobs u at time r_j. Recall that $q_u(t)$ is the remaining of job u at time t. We have

$$
\begin{aligned}
&\sum_{u \in U^2(r_j)} w_u f'(Q_u(r_j)) \\
&\quad \leq \delta_j \sum_{b=1}^{a-1} q_{u_b} f'\Big(t_0 - r_j + q_{u_1} + \ldots + q_{u_b}\Big) \\
&\qquad + \delta_j \Big(q_{u_a} - q_{u_a}(t)\Big) f'\Big(t_0 - r_j + q_{u_1} + \ldots + q_{u_{a-1}} + q_{u_a} - q_{u_a}(t)\Big) \\
&\qquad + \sum_{u \in U^2(r_j) \setminus W \cup a} w_u f'\Big(t - r_j + Q_u(t)\Big) \\
&\quad \overset{(\mathcal{P}3)}{\leq} \frac{1}{1-\epsilon} \delta_j \left[\sum_{b=1}^{a-1} q_{u_b} f'(t_0 - r_j + q_{u_1} + \ldots + q_{u_{b-1}}) + (q_{u_a} - q_{u_a}(t)) f'(t_0 - r_j + q_{u_1} + \ldots + q_{u_{a-1}})\right] \\
&\qquad + K_2(\epsilon) \delta_j \left[\sum_{b=1}^{a-1} q_{u_b} f'(q_{u_b}) + \Big(q_{u_a} - q_{u_a}(t)\Big) f'(q_{u_a} - q_{u_a}(t))\right] \\
&\qquad + \sum_{u \in U(t)} w_u f'(t - r_j + Q_u(t)) \\
&\quad \overset{(\mathcal{P}2)}{\leq} \frac{1}{1-\epsilon} \delta_j \left[\sum_{b=1}^{a-1} q_{u_b} f'(t_0 - r_j + q_{u_1} + \ldots + q_{u_{b-1}}) + (q_{u_a} - q_{u_a}(t)) f'(t_0 - r_j + q_{u_1} + \ldots + q_{u_b})\right] \\
&\qquad + K_2(\epsilon) \delta_j \left(\sum_{b=1}^{a-1} q_{u_b} + q_{u_a} - q_{u_a}(t)\right) f'\left(\sum_{b=1}^{a-1} q_{u_b} + q_{u_a} - q_{u_a}(t)\right) \\
&\qquad + \sum_{u \in U(t)} w_u f'(t - r_j + Q_u(t)) \\
&\overset{(\mathcal{P}2)\&(\mathcal{P}5)}{\leq} \frac{1}{1-\epsilon} \delta_j \left[f(t - r_j) - f(t_0 - r_j)\right] + K_2(\epsilon) K_4 \delta_j f(t - r_j) \\
&\qquad + \sum_{u \in U(t)} w_u f'(t - r_j + Q_u(t)) \qquad (9)
\end{aligned}
$$

where in the inequalities, note that $q_{u_1} + \ldots + q_{u_{a-1}} + q_{u_a} - q_{u_a}(t) = t - t_0 \leq t - r_j$.

Let W' be the set of jobs $u \in U(t)$ such that $t - r_j \leq \frac{1}{K_3(\epsilon)} Q_u(t)$. Let $W'' = U(t) \setminus W'$. Then

we have

$$\begin{aligned}
\sum_{u\in U(t)} w_u f'(t-r_j+Q_u(t)) &= \sum_{u\in W'} w_u f'(t-r_j+Q_u(t)) + \sum_{u\in W''} w_u f'(t-r_j+Q_u(t)) \\
&\overset{(\mathcal{P}4)}{\leq} \frac{1}{1-\epsilon}\sum_{u\in W'} w_u f'(Q_u(t)) + \sum_{u\in W''} w_u f'(t-r_j+Q_u(t)) \\
&\overset{(\mathcal{P}3)}{\leq} \frac{1}{1-\epsilon}\sum_{u\in W'} w_u f'(Q_u(t)) + \frac{1}{1-\epsilon}\sum_{u\in W''} w_u f'(Q_u(t)) + K_2(\epsilon)\sum_{u\in W''} w_u f'(t-r_j) \\
&\leq \frac{1}{1-\epsilon}\sum_{u\in W'\cup W''} w_u f'(Q_u(t)) + K_2(\epsilon)\delta_j \sum_{u\in W''} q_u f'(t-r_j) \\
&\leq \frac{1}{1-\epsilon}\sum_{u\in W'\cup W''} w_u f'(Q_u(t)) + 2K_2(\epsilon)K_3(\epsilon)\delta_j(t-r_j)f'(t-r_j) \\
&\overset{(\mathcal{P}5)}{\leq} \frac{1}{1-\epsilon}\sum_{u\in W'\cup W''} w_u f'(Q_u(t)) + 2K_2(\epsilon)K_3(\epsilon)K_4\delta_j f(t-r_j) \qquad (10)
\end{aligned}$$

The third inequality follows by $\delta_j \geq \delta_u$ for $u \in U^2(t)$. In the fourth inequality, note that $\sum_{u\in W''} q_u \leq (t_0 - r_j) + \sum_{u\in W''} q_u(t) \leq (t - r_j) + \max_{u\in W''} Q_u(t) \leq 2K_3(\epsilon)(t-r_j)$ (by definition of W'').

Using (9) and (10), we deduce that

$$\sum_{u\in U^2(r_j)} w_u f'(Q_u(r_j)) \leq \frac{3K_2(\epsilon)K_3(\epsilon)K_4}{1-\epsilon}\delta_j f(t-r_j) - \frac{1}{1-\epsilon}\delta_j f(t_0-r_j) + \frac{1}{1-\epsilon}\sum_{u\in U(t)} w_u f'(Q_u(t))$$

□

We are now proving the lemma. By (5), we have

$$\begin{aligned}
\frac{K(\epsilon)}{(1-\epsilon)^2}\lambda_j &\leq \delta_j f(t_0-r_j) + \sum_{u\in U^2(r_j)} w_u f'(Q_u(r_j)+p_j) \\
&\overset{(\mathcal{P}3)}{\leq} \delta_j f(t_0-r_j) + \frac{1}{1-\epsilon}\sum_{u\in U^2(r_j)} w_u f'(Q_u(r_j)) + K_2(\epsilon)\delta_j f'(p_j)p_j \\
&\overset{\text{(Claim 2)}}{\leq} \frac{3K_2(\epsilon)K_3(\epsilon)K_4}{(1-\epsilon)^2}\delta_j f(t-r_j) + \frac{1}{(1-\epsilon)^2}\sum_{u\in U(t)} w_u f'(Q_u(t)) + K_2(\epsilon)K_4\delta_j f(p_j)
\end{aligned}$$

Rearranging the terms, the lemma follows. □