Primal-dual and dual-fitting analysis of online scheduling algorithms for generalized flow-time problems

Spyros Angelopoulos[1], Giorgio Lucarelli[2,3], and Nguyen Kim Thang[4]

[1]CNRS and Sorbonne Universités, UPMC Univ Paris 06, Paris, France
spyros.angelopoulos@lip6.fr
[2]LIG, University Grenoble-Alpes, France
[3]LCOMS, Université de Lorraine, Metz, France
giorgio.lucarelli@univ-lorraine.fr
[4]IBISC, Univ Evry, Université Paris-Saclay, 91025, Evry, France
thang@ibisc.fr

Abstract

We study online scheduling problems on a single processor that can be viewed as extensions of the well-studied problem of minimizing total weighted flow time. In particular, we provide a framework of analysis that is derived by duality properties, does not rely on potential functions and is applicable to a variety of scheduling problems. A key ingredient in our approach is bypassing the need for "black-box" rounding of fractional solutions, which yields improved competitive ratios.

We begin with an interpretation of Highest-Density-First (HDF) as a primal-dual algorithm, and a corresponding proof that HDF is optimal for total fractional weighted flow time (and thus scalable for the integral objective). Building upon the salient ideas of the proof, we show how to apply and extend this analysis to the more general problem of minimizing $\sum_j w_j g(F_j)$, where w_j is the job weight, F_j is the flow time and g is a non-decreasing cost function. Among other results, we present improved competitive ratios for the setting in which g is a concave function, and the setting of same-density jobs but general cost functions. We further apply our framework of analysis to online weighted completion time with general cost functions as well as scheduling under polyhedral constraints.

1 Introduction

We consider online scheduling problems in which a set of jobs $\mathcal{J}$ arriving over time must be executed on a single processor. In particular, each job $j \in \mathcal{J}$ is characterized by its *processing time* $p_j > 0$ and its *weight* $w_j > 0$, which become known after its *release time* $r_j \geq 0$. The *density* of j is $\delta_j = w_j/p_j$, whereas, given a schedule, its *completion time*, C_j,

1

is defined as the first time $t \geq r_j$ such that p_j units of j have been processed. The *flow time* of j is then defined as $F_j = C_j - r_j$, and represents the time elapsed after the release of job j and up to its completion. A natural optimization objective is to design schedules that minimize the *total weighted flow time*, namely the sum $\sum_{j \in \mathcal{J}} w_j F_j$ of all processed jobs. A related objective is to minimize the weighted sum of completion times, as given by the expression $\sum_{j \in \mathcal{J}} w_j C_j$. We assume that *preemption* of jobs is allowed, i.e., the jobs can be interrupted and be resumed at a later point without any penalty.

Im *et al.* [18] studied a generalization of the total weighted flow time problem, in which jobs may incur non-linear contributions to the objective. More formally, they defined the *Generalized Flow Time Problem* (GFP) in which the objective is to minimize the sum $\sum_{j \in \mathcal{J}} w_j g(F_j)$, where $g : \mathbb{R}^+ \to \mathbb{R}^+$ is a given non-decreasing cost function with $g(0) = 0$. This extension captures many interesting and natural variants of flow time with real-life applications. Moreover, it is an appropriate formulation of the setting in which we aim to simultaneously optimize several objectives. We define the *Generalized Completion Time Problem* (GCP) along the same lines, with the only difference being the objective function, which equals to $\sum_{j \in J} w_j g(C_j)$.

A further generalization of the above problems, introduced in [5], associates each job j with a non-decreasing cost function $g_j : \mathbb{R}^+ \to \mathbb{R}^+$ and $g_j(0) = 0$; in the *Job-Dependent Generalized Flow Time Problem* (JDGFP), the objective is to minimize the sum $\sum_{j \in \mathcal{J}} w_j g_j(F_j)$. This problem formulation is quite powerful, and captures many natural scheduling objectives such as weighted tardiness. No competitive algorithms are known in the online setting; approximation algorithms are studied in [5, 14].

Very recently, Im *et al.* [15] introduced and studied a general scheduling problem called *Packing Scheduling Problem* (PSP). Here, at any time t, the scheduler may assign rates $\{x_j(t)\}$ to each job $j \in \mathcal{J}$. In addition, we are given a matrix B of non-negative entries. The goal is to minimize the total weighted flow time subject to packing constraints $\{B\mathbf{x} \leq 1, \mathbf{x} \geq 0\}$. This formulates applications in which each job j is associated with a resource-demand vector $\mathbf{b}_j = (b_{1j}, b_{2j}, \ldots, b_{Mj})$ so that it requires an amount b_{ij} of the i-th resource.

In this paper, we present a general framework based on LP-duality principles, for online scheduling with generalized flow time objectives. Since no online algorithm even for total weighted flow time is constant competitive [3], we study the effect of *resource augmentation*, introduced by Kalyanasundaram and Pruhs [20]. More precisely, given some optimization objective (e.g. total flow time), an algorithm is said to be α-speed β-competitive if it is β-competitive with respect to an offline optimal scheduling algorithm of speed $1/\alpha$ (here $\alpha \geq 1$).

Related work. It is well-known that the algorithm Shortest Remaining Processing Time (SRPT) is optimal for online total (unweighted) flow time. Becchetti *et al.* [6] showed that the natural algorithm Highest-Density-First (HDF) is $(1+\epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive for total weighted flow time. At each time, HDF processes the job of highest density.

Concerning the online GFP, Im *et al.* [18] showed that HDF is $(2 + \epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive algorithm for general non decreasing functions g. On the negative side, they showed that no *oblivious* algorithm is $O(1)$-competitive with speed augmentation $2 - \epsilon$, for any $\epsilon > 0$ (an oblivious algorithm does not know the function g). In the case in which g is a twice-differentiable, concave function, they showed that the algorithm Weighted Late Arrival

Processor Sharing (WLAPS) is $(1 + \epsilon)$-speed $O(\frac{1}{\epsilon^2})$-competitive. For equal-density jobs and general cost functions [18] prove that FIFO is $(1 + \epsilon)$-speed $\frac{4}{\epsilon^2}$-competitive. Fox *et al.* [10] studied the problem of convex cost functions in the *non-clairvoyant* variant providing a $(2+\epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive algorithm; in this variant, the scheduler learns the processing time of a job only when the job is completed (see for example [8, 9, 11, 12, 20, 21]). Bansal and Pruhs [4] considered a special class of convex functions, namely the weighted ℓ_k norms of flow time, with $1 < k < \infty$, and they showed that HDF is $(1 + \epsilon)$-speed $O(\frac{1}{\epsilon^2})$-competitive. Moreover, they showed how to transform this result in order to obtain an 1-speed $O(1)$-competitive algorithm for the weighted ℓ_k norms of completion time.

Most of the above works rely to techniques based on amortized analysis (see also [17] for a survey). More recently, techniques based on LP duality have been applied in the context of online scheduling for generalized flow time problems. Gupta *et al.* [13] gave a primal-dual algorithm for a class of non-linear load balancing problems. Devanur and Huang [7] used a duality approach for the problem of minimizing the sum of energy and weighted flow time on unrelated machines. Of particular relevance to our paper is the work of Antoniadis *et al.* [2], which gives an optimal *offline* energy and fractional weighted flow trade-off schedule for a speed-scalable processor with discrete speeds, and uses an approach based on primal-dual properties (similar geometric interpretations arise in the context of our work, in the online setting). Anand *et al.* [1] were the first to propose an approach to online scheduling by linear/convex programming and dual fitting. Nguyen [22] presented a framework based on Lagrangian duality for online scheduling problems beyond linear and convex programming. The framework is applied to several problems related to flow time such as energy plus flow time, ℓ_p-norm of flow time. Im *et al.* [15] applied dual fitting in the context of PSP. For the weighted flow time objective, they gave a non-clairvoyant algorithm that is $O(\log n)$-speed $O(\log n)$-competitive, where n denotes the number of jobs. They also showed that for any constant $\epsilon > 0$, any $O(n^{1-\epsilon})$-competitive algorithm requires speed augmentation compared to the offline optimum.

We note that a common approach in obtaining a competitive, resource-augmented scheduling algorithm for flow time and related problems is by first deriving an algorithm that is competitive for the *fractional* objective [6, 17]. An informal interpretation of the fractional objective is that a job contributes to the objective proportionally to the amount of its remaining work (see Section 2 for a formal definition). It is known that any α-speed β-competitive algorithm for fractional GFP can be converted, in "black-box" fashion, to a $(1 + \epsilon)\alpha$-speed $\frac{1+\epsilon}{\epsilon}\beta$-competitive algorithm for (integral) GFP, for $0 < \epsilon \leq 1$ [10]. Fractional objectives are often considered as interesting problems in their own (as in [2]).

Contribution. We present a framework for the design and analysis of algorithms for generalized flow time problems that is based on primal-dual and dual-fitting techniques. Our proofs are based on intuitive geometric interpretations of the primal/dual objectives; in particular, we do not rely on potential functions. An interesting feature in our primal-dual approach, that differs from previous ones, is that when a new job arrives, we may update the dual variables for jobs that already have been scheduled without affecting the past portion (primal solution) of the schedule. Another important ingredient of our analysis consists in relating, in a direct manner, the primal integral and fractional dual objectives, without passing through the fractional primal. This allows us to bypass the "black-box" transformation

$g(\cdot)$	$\sum w_j g(F_j)$		$\sum w_j g(C_j)$	$\sum w_j g_j(F_j)$
	same density	arbitrary density		
linear		HDF, $(1+\epsilon, \frac{1+\epsilon}{\epsilon})$ [6]		
convex	**FIFO, $(1+\epsilon, \frac{1+\epsilon}{\epsilon})$**	WSETF, $(2+\epsilon, O(\frac{1}{\epsilon}))$ [10] oblivious, $(2-\epsilon, \Omega(1))$ [10] non-clairvoyant, $(\sqrt{2}-\epsilon, \Omega(1))$ [10]		
concave differentiable		WLAPS, $(1+\epsilon, O(\frac{1}{\epsilon^2}))$ [18]		$(1+\epsilon, \frac{4(1+\epsilon)^2}{\epsilon^2})$
concave	**LIFO, $(1+\epsilon, \frac{1+\epsilon}{\epsilon})$**	**HDF, $(1+\epsilon, \frac{1+\epsilon}{\epsilon})$**		
general	FIFO, $(1+\epsilon, \frac{4}{\epsilon^2})$ [18] **FIFO, $(1+\epsilon, \frac{1+\epsilon}{\epsilon})$**	HDF, $(2+\epsilon, O(\frac{1}{\epsilon}))$ [18] $(7/6-\epsilon, \Omega(1))$ [18]	**HDF, $(1+\epsilon, \frac{1+\epsilon}{\epsilon})$**	

Table 1: Summary of results for generalized flow time and completion time problems on a single machine. The (α, β) notation describes an algorithm which is α-speed β-competitive. Our contribution (excluding our result on the PSP problem) is shown in bold.

of fractional to integral solutions [6, 18], which has been the canonical approach up to now. As a result, we obtain an improvement to the competitive ratio by a factor of $O(\frac{1}{\epsilon})$, for $(1 + \epsilon)$-speed.

In Section 3 we begin with an interpretation of HDF as a primal-dual algorithm for total weighted flow time. Our analysis, albeit significantly more complicated than the known combinatorial one [6], yields insights about more complex problems. One may draw parallels with the successful application of dual fitting in approximation algorithms; for instance, the dual-fitting analysis of set cover that has led to improved algorithms for metric facility location [19]. We also note that our approach differs from [7] (in which the objective is to minimize the sum of energy and weighted flow time), even though the two settings are seemingly similar. More precisely, the relaxation considered in [7] consists only of covering constraints, whereas for minimizing weighted flow time, one has to consider both covering and packing constraints in the primal LP.

In Sections 4 and 5 we expand the salient ideas behind the above analysis of HDF and derive a framework which is applicable to more complicated objectives. More precisely, we show that HDF is $(1 + \epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive for GFP with concave functions, improving the $(1 + \epsilon)$-speed $O(\frac{1}{\epsilon^2})$-competitive analysis of WLAPS [18], and removing the assumption that g is twice-differentiable. For GFP with general cost functions and jobs of the same density, we show that FIFO is $(1 + \epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive, which improves again the analysis in [18] by a factor of $O(\frac{1}{\epsilon})$ in the competitive ratio. For the special case of GFP with equal-density jobs and convex (resp. concave) cost functions we show that FIFO (resp. LIFO) are fractionally optimal, and $(1 + \epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive for the integral objective.

In addition, we apply our framework to the following problems: i) online GCP: here, we show that HDF is optimal for the fractional objective, and $(1 + \epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive for the integral one; and ii) online PSP assuming a matrix B of strictly positive elements: here, we derive an adaptation of HDF which we prove is 1-competitive and which requires resource augmentation $\max_j \frac{B_j}{b_j}$, with $B_j = \max_i b_{ij}$ and $b_j = \min_i b_{ij}$.

Last, in Section 6 we extend ideas of [16], using, in addition, the Lagrangian relaxation of a non-convex formulation for the online JDGFT problem. We thus obtain a non-oblivious

$(1+\epsilon)$-speed $\frac{4(1+\epsilon)^2}{\epsilon^2}$-competitive algorithm, assuming each function g_j is concave and differentiable. Note that this result does not rely on our framework.

Table 1 summarizes the results of this paper in comparison to previous work.

Notation. Let z be a job that is released at time τ. For a given scheduling algorithm, we denote by P_τ the set of pending jobs at time τ (i.e., jobs released up to and including τ but not yet completed), and by $C_{\max}^\tau$ the last completion time among jobs in P_τ, assuming no jobs are released after τ. We also define R_τ as the set of all jobs released up to and included τ (which may or may not have been completed at τ) and $\mathcal{J}_\tau$ as the set of all jobs that have been completed up to time τ.

2 Linear programming relaxation

In order to give a linear programming relaxation of GFP, we pass through the corresponding fractional variant of the objective. Formally, let $q_j(t)$ be the remaining processing time of job j at time t (in a schedule). The *fractional remaining weight* of j at time t is defined as $w_j q_j(t)/p_j$. The fractional objective of GFP is now defined as $\sum_j \int_{r_j}^{\infty} w_j \frac{q_j(t)}{p_j} g'(t - r_j)$. Note that the fractional objective is a lower bound to the integral one, since

$$\sum_j \int_{r_j}^{\infty} w_j \frac{q_j(t)}{p_j} g'(t - r_j) < \sum_j \int_{r_j}^{C_j} w_j g'(t - r_j) < \sum_j w_j g(C_j - r_j)$$

where the last inequality is due to the assumption that g is non-decreasing. An advantage of fractional GFP is that it admits a linear-programming formulation (in fact, the same holds even for the stronger problem JDGFP (see Appendix A)). Let $x_j(t) \in [0, 1]$ be a variable that indicates the execution rate of $j \in \mathcal{J}$ at time t. The primal and dual LPs are:

$$\min \sum_{j \in \mathcal{J}} \delta_j \int_{r_j}^{\infty} g(t - r_j) x_j(t) dt \quad (P) \qquad \max \sum_{j \in \mathcal{J}} \lambda_j p_j - \int_0^{\infty} \gamma(t) dt \qquad (D)$$

$$\int_{r_j}^{\infty} x_j(t) dt \geq p_j \quad \forall j \in \mathcal{J} \qquad (1) \qquad \lambda_j - \gamma(t) \leq \delta_j g(t - r_j) \qquad \forall j \in \mathcal{J}, t \geq r_j \quad (3)$$

$$\sum_{j \in \mathcal{J}} x_j(t) \leq 1 \quad \forall t \geq 0 \qquad (2) \qquad \lambda_j, \gamma(t) \geq 0 \qquad \forall j \in \mathcal{J}, \forall t \geq 0$$

$$x_j(t) \geq 0 \quad \forall j \in \mathcal{J}, t \geq 0$$

In this paper, we avoid the use of the "black-box" transformation (see for example [6, 17]) from fractional to integral GFP. However, we also consider the fractional objective of GFP as a lower bound for the integral one. Specifically, we will prove the performance of an algorithm by comparing its integral objective to that of a feasible dual solution (D). Note that by weak duality the latter is upper-bounded by the optimal solution of (P), which equals to the optimal solution for fractional GFP and hence it is a lower bound of the optimum solution for integral GFP.

Moreover, we will analyze algorithms that are α-speed β-competitive. In other words, we compare the performance of our algorithm to an offline optimum with speed $1/\alpha$ ($\alpha, \beta > 1$). In turn, the cost of this offline optimum is the objective of a variant of (P) in which constraints (2) are replaced by constraints $\sum_{j \in \mathcal{J}} x_j(t) \leq 1/\alpha$ for all $t \geq 0$. The corresponding dual is the same as (D), with the only difference that the objective is equal to $\sum_{j \in \mathcal{J}} \lambda_j p_j - \frac{1}{\alpha} \int_0^\infty \gamma(t) dt$. We denote these modified primal and dual LP's by (P_α) and (D_α), respectively. In order to prove that the algorithm is α-speed β-competitive, it will then be sufficient to show that there is a feasible dual solution to (D_α) for which the algorithm's cost is at most β times the objective of the solution.

3   A primal-dual interpretation of HDF for $\sum_j w_j F_j$

In this section we give an alternative statement of HDF as a primal-dual algorithm for the total weighted flow time problem. We consider the primal and dual programs defined in the previous section where the function g is considered to be the identity function, i.e., $g(x) = x$. We begin with an intuitive understanding of the complementary slackness (CS) conditions. In particular, the primal CS condition states that for a given job j and time t, if $x_j(t) > 0$, i.e., if the algorithm were to execute job j at time t, then it should be that $\gamma(t) = \lambda_j - \delta_j(t - r_j)$. We would like then the dual variable $\gamma(t)$ to be such that we obtain some information about which job to schedule at time t. To this end, for any job $j \in \mathcal{J}$, we define the line $\gamma_j(t) = \lambda_j - \delta_j(t - r_j)$, with domain $[r_j, \infty)$. The slope of this line is equal to the negative density of the job, i.e, $-\delta_j$. Our algorithm will always choose $\gamma(t)$ to be equal to $\max\{0, \max_{j \in \mathcal{J}: r_j \leq t}\{\gamma_j(t)\}\}$ for every $t \geq 0$. We say that at time t the line γ_j (or the job j) is *dominant* if $\gamma_j(t) = \gamma(t)$; informally, γ_j is above $\gamma_{j'}$, for all $j' \neq j$. We can thus restate the primal CS condition as a *dominance* condition: if a job j is executed at time t, then γ_j must be dominant at t.

We will consider a class of scheduling algorithms, denoted by $\mathcal{A}$, that comply to the following rules: i) the processor is never idle if there are pending jobs; and ii) if at time τ a new job z is released, the algorithm will first decide an ordering on the set P_τ of all pending jobs at time τ. Then for every $t \geq \tau$, it schedules all jobs in P_τ according to the above ordering, unless a new job arrives after τ.

We now proceed to give a primal-dual algorithm in the class $\mathcal{A}$ (which will turn out to be identical to HDF). The algorithm will use the dominance condition so as to decide how to update the dual variables λ_j, and, on the primal side, which job to execute at each time. Note that once we define the λ_j's, the lines γ_j's as well as $\gamma(t)$ are well-defined, as we emphasized earlier. In our scheme we change the primal and dual variables only upon arrival of a new job, say at time τ. We also modify the dual variables for jobs in $\mathcal{J}_\tau$, i.e., jobs that have already completed in the past (before time τ) without however affecting the primal variables of the past, so as to comply with the online nature of the problem.

By induction, suppose that the primal-dual algorithm $A \in \mathcal{A}$ satisfies the dominance condition up to time τ, upon which a new job z arrives. Let q_j be the remaining processing time of each job $j \in P_\tau$ at time τ and $|P_\tau| = k$. Each $j \in P_\tau$ has a corresponding line γ_j, once λ_j is defined. To satisfy CS conditions, each line γ_j must be defined such that to be dominant for a total period of time at least q_j, in $[\tau, \infty)$. The crucial observation is that, if
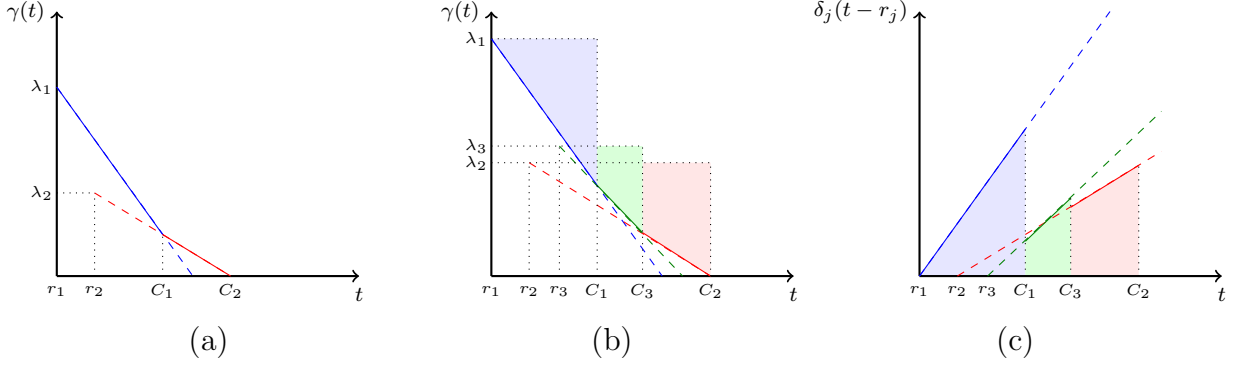
6

Figure 1: Figure (a) depicts the situation right before τ: the two lines γ_1, γ_2 correspond to two pending jobs prior to the release of z. In addition, $\gamma(t)$ is the upper envelope of the two lines. Figure (b) illustrates the situation after the release of a third job z at time $\tau = r_3$; the area of the shaded regions is the dual objective. In Figure (c), the area of the shaded regions is the primal fractional objective for the three jobs of Figure (b).

a line γ_j is dominant at times t_1, t_2, it must also be dominant in the entire interval $[t_1, t_2]$. This implies that for two jobs $j_1, j_2 \in P_\tau$, such that j_1 (resp. j_2) is dominant at time t_1 (resp. t_2), if $t_1 < t_2$ then the slope of γ_{j_1} must be smaller than the slope of γ_{j_2} (i.e., $-\delta_{j_1} \leq -\delta_{j_2}$). We derive that A must make the same decisions as HDF. Consequently, the algorithm A orders the jobs in P_τ in non-decreasing order of the slopes of the corresponding lines γ_j (note that the slope of the lines is independent of the λ_j's). For every job $j \in P_\tau$, define $C_j = \tau + \sum_{j' \prec j} q_{j'}$, where the precedence is according to the above ordering of A. These are the completion times of jobs in P_τ in A's schedule, if no new jobs are released after time τ; so we set the primal variables $x_j(t) = 1$ for all $t \in (C_{j-1}, C_j]$. Procedure 1 formalizes the choice of λ_j for all $j \in P_\tau$; intuitively, it ensures that if a job $j \in P_\tau$ is executed at time $t > \tau$ then γ_j is dominant at t (see Figure 1 for an illustration).

Procedure 1 Assignment of dual variables λ_j for all $j \in P_\tau$ at the arrival of a new job z.

1: Consider the jobs in P_τ in increasing order of completion times if no new jobs are released after time τ, i.e., $C_1 < C_2 < \ldots < C_k$
2: Choose λ_k such that $\gamma_k(C_k) = 0$
3: **for** each pending job $j = k - 1$ to 1 **do**
4:     Choose λ_j such that $\gamma_j(C_j) = \gamma_{j+1}(C_j)$

We first observe a monotonicity property of λ_j's.

Lemma 1. *By Procedure 1, the value of the dual variable λ_j, $j \in P_\tau$, can be only increased after the arrival of a new job z at time τ.*

Proof. Let λ'_j and λ_j be the value of the dual variable of j before and after the arrival of z, respectively. We consider the following two cases.

Job j is delayed by z. Let C_j be the new completion time of job j. So before the arrival of z, the completion of j was $C_j - p_z$. Thus, by Procedure 1, we have that $\gamma_j(C_j) =$

7

$\gamma'_j(C_j - p_z)$ where γ'_j is the corresponding line of j before the arrival of z. Thus, $\lambda_j - \delta_j(C_j - r_j) = \lambda'_j - \delta_j(C_j - p_z - r_j)$, that is $\lambda_j > \lambda'_j$.

Job j is not delayed by z. In this case, Procedure 1 increases the dual variable of j by $\gamma_z(C_z - p_z - r_z) - \gamma_z(C_z - r_z) > 0$. Hence, $\lambda_j > \lambda'_j$.

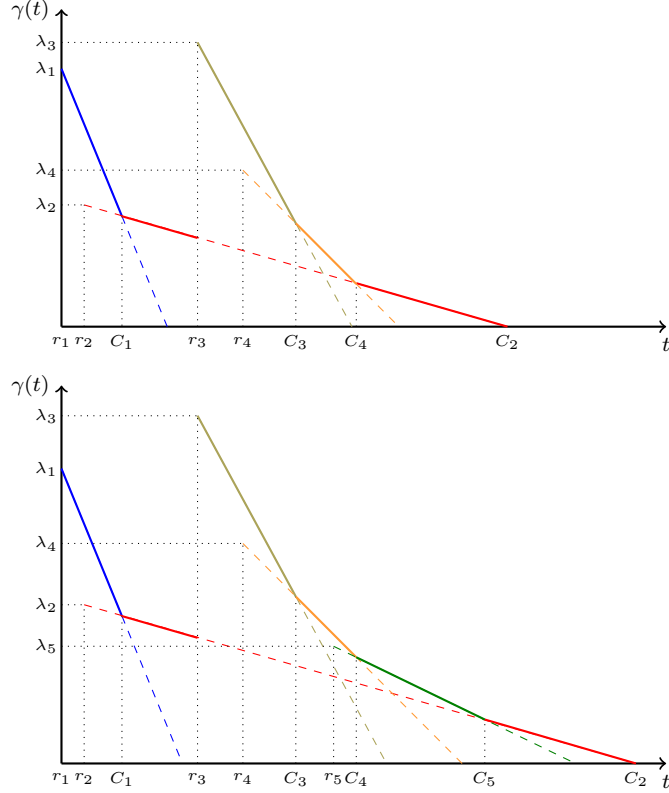The lemma follows. $\qquad\square$

Figure 2: The figure depictes the lines corresponding to the dual variables before (above) and after (below) the arrival of job 5 at time $\tau = r_5$. The set of completed jobs at time τ is $\mathcal{J}_\tau = \{1, 3\}$. Moreover, Procedure 2 defines three disjoint sets: $\{1, 2\}$ with representative 2, $\{5\}$ with representative 5 and $\{3, 4\}$ with representative 4.

The following lemma shows that if no new jobs were to be released after time τ, HDF would guarantee the dominance condition for all times $t \geq \tau$.

Lemma 2 (future dominance). *For λ_j's as defined by Procedure 1, and $A \equiv HDF$, if job $j \in P_\tau$ is executed at time $t \geq \tau$, then γ_j is dominant at t, assuming that no new jobs are released after time τ.*

Proof. Suppose that the jobs in P_τ are in increasing order of their completion times, i.e., $C_1 < C_2 < \ldots < C_k$. In order to prove the lemma, we have to show that

$$\gamma_j(t) \leq \gamma_{j+1}(t) \qquad \forall t \geq C_j \tag{4}$$
$$\gamma_j(t) \geq \gamma_{j+1}(t) \qquad \forall t \in [\tau, C_j] \tag{5}$$

8

Recall that by the choice of λ_j's, we have that $\lambda_j - \delta_j(C_j - r_j) = \lambda_{j+1} - \delta_{j+1}(C_j - r_{j+1})$. Therefore,

$$
\begin{aligned}
\lambda_j - \delta_j(t - r_j) &= \lambda_{j+1} - \delta_{j+1}(C_j - r_{j+1}) + \delta_j(C_j - r_j) - \delta_j(t - r_j) \\
&= \lambda_{j+1} - \delta_{j+1}(t - r_{j+1}) + (\delta_j - \delta_{j+1})(C_j - t)
\end{aligned}
\tag{6}
$$

Since $A \equiv HDF$, we have that $\delta_j - \delta_{j+1} \geq 0$. If $t \geq C_j$ then (4) follows, while if $t < C_j$ then (5) follows. $\qquad\square$

By Lemma 1, Procedure 1 modifies (increases) the λ_j variables of all jobs pending at time τ. In turn, this action may violate the dominance condition *prior to* τ. We thus need a second procedure that will rectify the dominance condition for $t \leq \tau$.

We consider again the jobs in P_τ in increasing order of their completion times, i.e., $C_1 < C_2 < \ldots < C_k$, with $k = |P_\tau|$. We partition R_τ into k disjoint sets $S_1, S_2, \ldots, S_k$. Each set S_j is initialized with the job $j \in P_\tau$, which is called the *representative* element of S_j (we use the same index to denote the set and its representative job). Informally, the set S_j will be constructed in such a way that it will contain all jobs $a \in \mathcal{J}_\tau$ whose corresponding variable λ_a will be increased by the same amount in the procedure. This amount is equal to the increase, say Δ_j, of λ_j, due to Procedure 1 for the representative job of S_j.

Fig. 2 gives an example of the definition of the above sets. We can also observe that the distance between λ_1 and λ_2, as well as, between λ_3 and λ_4 remains the same before and after the arrival of the job 5. However, this is not true for jobs of different sets (see for example λ_1 and λ_3). We then define Procedure 2 that describes formally the increase in the dual variables for jobs in $\mathcal{J}_\tau$.

Procedure 2 Updating of dual variables λ_j for all jobs $j \in \mathcal{J}_\tau$ at the arrival of a new job z.

1: **for** $j = 1$ to k **do**
2:     Add j in S_j
3: **for** each job $a \in \mathcal{J}_\tau$ in decreasing order of completion times (defined under the assumption that no new jobs are released after time τ) **do**
4:     Let b be the job such that $\gamma_a(C_a) = \gamma_b(C_a)$
5:     Let S_j be the set that contains b
6:     Add a in S_j
7: **for** each set S_j, $1 \leq j \leq k$, **do**
8:     Let Δ_j be the increase of λ_j, due to Procedure 1, for the representative of S_j
9:     Increase λ_a by an amount of Δ_j for all $a \in S_j \setminus \{j\}$

Geometrically, the update operation is a vertical translation of the line $\gamma(t)$ for $t < \tau$. The following lemma ensures that the job b selected in Line 4 of Procedure 2 always exists.

Lemma 3. *For each job $a \in \mathcal{J}_\tau$, there is always a job b such that $\gamma_a(C_a) = \gamma_b(C_a)$ (except if a is the last completed job of the schedule). Moreover, Procedure 2 adds b to a set S_j before treating a.*

Proof. Let c be the last job arrived before the completion of a, i.e., $r_c \leq C_a$ and there is no job c' with $r_c < r_{c'} \leq C_a$. Note that a may coincide with c. Since $a \in P_{r_c}$, there is a job b

9

such that $\gamma_a(C_a) = \gamma_b(C_a)$ as this is defined by Line 4 of Procedure 1, except if a is the last completed job of the schedule (Line 2 of Procedure 1). After this point only Procedure 2 can raise this property. However, Procedure 2 always groups the jobs a and b in the same set, let S_j. Since all lines corresponding to jobs in S_j are vertically translated by the same quantity, the first part of the lemma follows.

If $b \in P_\tau$, then the second part of the lemma directly holds. Otherwise, consider again the last job, c, arrived before the completion of a. By the definition of c and the execution of Procedure 1 at time r_c, we know that a completes before b and hence it is treated before a by Procedure 2. $\qquad\square$

The following lemma shows that, if a line γ_j was dominant for a time $t < \tau$ prior to the arrival of the new job at time τ, then it will remain dominant after the application of Procedures 1 and 2.

Lemma 4 (past dominance). *For λ_j's as defined by both Procedure 1 and Procedure 2, and $A \equiv HDF$, if job $j \in \mathcal{J}_\tau \cup P_\tau$ is executed at time $t < \tau$, then γ_j is dominant at t.*

Proof. The proof is based on the following three claims.

Claim 1. *Let j_1 and j_2 be two jobs in P_τ such that $C_{j_1} < C_{j_2}$. If $A \equiv HDF$, then $\Delta_{j_1} \geq \Delta_{j_2}$.*

Proof of claim. Let z be the job released at time τ, with processing time p_z. Consider the following three cases.

(i) $C_z < C_{j_1} < C_{j_2}$. Hence, the completion times of both j_1 and j_2 are delayed by p_z by HDF. In other words, before the arrival of z the completion times of j_1 and j_2 were $C_{j_1} - p_z$ and $C_{j_2} - p_z$, respectively. Moreover, the relative orderings of all jobs in $P_\tau \setminus \{z\}$ (as done by the algorithm) is the same before and after the release of z. Thus, by Procedure 1, we have that $\lambda_{j_1} - \delta_{j_1}(C_{j_1} - r_{j_1}) = \lambda'_{j_1} - \delta_{j_1}(C_{j_1} - p_z - r_{j_1})$, where λ'_{j_1} is the value of the dual variable of j_1 before the arrival of z. Hence, $\Delta_{j_1} = \delta_{j_1} p_z$. Similarly, we obtain that $\Delta_{j_2} = \delta_{j_2} p_z$. Therefore, $\Delta_{j_1} \geq \Delta_{j_2}$ as from the HDF algorithm we infer that $\delta_{j_1} \geq \delta_{j_2}$.

(ii) $C_{j_1} < C_{j_2} < C_z$. In this case, Procedure 1 increases both λ_{j_1} and λ_{j_2} by $\gamma_z(C_z - p_z - r_z) - \gamma_z(C_z - r_z)$. Therefore, $\Delta_{j_1} = \Delta_{j_2} = \delta_z p_z$.

(iii) $C_{j_1} < C_z < C_{j_2}$. As in case (ii), we have $\Delta_{j_1} = \delta_z p_z$. As in case (i), we have $\Delta_{j_2} = \delta_{j_2} p_z$. Therefore, $\Delta_{j_1} \geq \Delta_{j_2}$ as from the HDF algorithm we infer that $\delta_z \geq \delta_{j_2}$.

The claim follows. $\qquad\square$

We call a set S_j *critical* if at least one of the following hold: S_j contains at least one job in $\mathcal{J}_\tau$ (i.e., $S_j \cap \mathcal{J}_\tau \neq \emptyset$) or its representative job has been partially executed before time τ (i.e., $q_j(\tau) < p_j$). Let $\ell \leq k$ be the number of critical sets. Let also $Q_\tau \subseteq P_\tau$ denote the set of representative jobs of critical subsets; hence $|Q_\tau| = \ell$. Note that job z which is released at time τ does not belong in Q_τ.

Claim 2. *Let j_1 and j_2 be two jobs in Q_τ such that $C_{j_1} < C_{j_2}$. Then $r_{j_1} \geq r_{j_2}$.*

10

Proof of claim. By way of contradiction, suppose that there are two jobs j_1 and j_2 in Q_τ such that $C_{j_1} < C_{j_2}$ and $r_{j_1} < r_{j_2}$. Since $C_{j_1} < C_{j_2}$, j_1 has higher density than j_2. Hence, j_2 has not been scheduled before τ because j_1 is active during $[r_{j_2}, \tau]$ and it has higher density. Thus, from the definition of Q_τ, there is a job $a \in S_{j_2} \cap \mathcal{J}_\tau$ for which $\gamma_a(C_a) = \gamma_{j_2}(C_a)$ and $C_a < \tau$. Therefore, at time C_a the job j_2 is the pending job with the highest density, which is a contradiction as we assumed that j_1 is already released by time C_a and that it has higher density than j_2. $\qquad\square$

Let t_j be the first time in which a job in the critical set S_j begins its execution for $1 \le j \le \ell$. Let $i_1, i_2, \ldots, i_\ell$ be job indices such that $t_{i_1} < t_{i_2} < \ldots < t_{i_\ell} \le \tau$. The following lemma shows a structure property of the algorithm schedule.

Claim 3. *During interval $[t_{i_j}, t_{i_{j+1}})$, only jobs in S_j are executed for $1 \le j \le \ell - 1$.*

Proof of claim. Consider interval $[t_{i_j}, t_{i_{j+1}})$ and let a be the job processed at time t_{i_j}. If a is still pending at time τ then by definition a is indeed job i_j and S_{i_j} consists of a singleton job. If a is completed at time $C_a < \tau$ then by Procedure 2 there exists a job b such that $\gamma_b(C_a) = \gamma_a(C_a)$ and jobs a and b belong to the same representative set. By repeating the same argument for job b inductively, we infer that all jobs executed in interval $[t_{i_j}, t_{i_{j+1}})$ belong to the same representative set. $\qquad\square$

We now continue with the proof of the lemma and we show that at each time $t < \tau$, the dominant job remains the same before and after the application of Procedures 1 and 2. By Procedure 2, the dual variables λ_j of all jobs in the same critical set are all increased by the same amount. Moreover, by Claim 3 the jobs in the same critical set are all executed consecutively. Consider two jobs j_1 and j_2 in the same critical set S_j. Assume, without loss of generality, that $\gamma'_{j_1}(t) \ge \gamma'_{j_1}(t)$ for time $t < \tau$, where γ'_{j_1} and γ'_{j_2} are the lines of j_1, j_2 prior to the arrival of z. Then $\gamma_{j_1}(t) = \gamma'_{j_1}(t) + \Delta_j \ge \gamma'_{j_2}(t) + \Delta_j = \gamma_{j_2}(t)$. Therefore, it suffices to consider only jobs that belong in different critical sets.

Consider the critical sets in decreasing order of completion times of their representatives, i.e., $C_1 > C_2 > \ldots > C_\ell$, and let S_j and S_{j+1} be any pair of consecutive critical sets. Consider also the time $t_{i_{j+1}} < \tau$ as it is defined before Claim 3. Let $a \in S_j$ be the last job that is executed before $t_{i_{j+1}}$ and $b \in S_{j+1}$ be the job that is executed at $t_{i_{j+1}}$. By Claim 1, $\Delta_j \le \Delta_{j+1}$ and hence λ_a has been increased at most as much as λ_b did (in Procedure 2). Thus, γ_b is dominant after $t_{i_{j+1}}$ and it remains to show that γ_a is dominant just before $t_{i_{j+1}}$. In order to prove this, it suffices to prove that the job b is not yet released by time $t_{i_{j+1}}$, which means that γ_b does not affect the line γ_a prior to this time. Indeed, if b was released before $t_{i_{j+1}}$, then $C_a = t_{i_{j+1}}$ and $\Delta_a > \Delta_b$ since $A \equiv HDF$ and a was dominant just before $t_{i_{j+1}}$ while b was dominant after $t_{i_{j+1}}$ when considering the situation prior to the arrival of the new job z at time τ. Hence, Procedure 1 would have defined $\gamma_a(C_a) \ne \gamma_b(C_a)$, that is a and b would belong to the same representative set, which is a contradiction. $\qquad\square$

Recall that $C_{\max}^\tau$ denotes the completion time of the last pending job in P_τ (with the usual assumption that no job arrives after time τ). The following lemma states that the dual variable $\gamma(t)$ has been defined in such a way that it is zero for all $t > C_{\max}^\tau$. This will be required in order to establish that the primal and dual solutions have the same objective value.

Lemma 5 (completion). *For λ_j's defined by Procedures 1 and 2, we have that $\gamma(t) = 0$ for every $t > C_{\max}^\tau$.*

Proof. From Lemmas 2 and 4, for every job $j \in \mathcal{J}_\tau \cup P_\tau$, it holds that $\gamma_j(C_{\max}^\tau) \leq \gamma(C_{\max}^\tau)$. By construction we have that $\gamma(C_{\max}^\tau) = 0$, and hence $\gamma_j(C_{\max}^\tau) \leq 0$. Moreover, $\gamma_j(t)$ is non-increasing function of t, that is $\gamma_j(t) \leq \gamma_j(C_{\max}^\tau)$ for every $t \geq C_{\max}^\tau$. Therefore, $\gamma(t) = \max\{0, \max_{j:r_j \leq t}\{\gamma_j(t)\}\} = 0$ for $t \geq C_{\max}^\tau$. $\qquad\square$

The proof of the following theorem is based on Lemmas 2, 4 and 5, and it is a simplified case of the proof of Theorem 2 which is given in the next section. We note that the primal and dual objectives have intuitive geometric interpretations, as shown in Figure 1(b),(c). For each job, its contribution to the dual objective is the area of a trapezoid that is exactly the same as the contribution of the job to the primal objective.

Theorem 1. *The primal-dual algorithm $A \equiv HDF$ is an optimal online algorithm for the total fractional weighted flow time and a $(1+\epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive algorithm for the total (integral) weighted flow time.*

4   A framework for primal-dual algorithms

Building on the primal-dual analysis of HDF for total weighted flow time, we can abstract the essential properties that we need to satisfy in order to obtain online algorithms for other similar problems. For the problems we consider, the primal solution is generated by an online primal-dual algorithm $A \in \mathcal{A}$ which may not necessarily be HDF. In addition, each job j will now correspond to a *curve* γ_j (for the total weighted flow time problem, γ_j is a line), and we will also have a dual variable $\gamma(t)$ that will be set equal to $\max\{0, \max_{j \in \mathcal{J}:r_j \leq t}\{\gamma_j(t)\}\}$ for every $t \geq 0$. Finally, the crux is in maintaining dual variables λ_j, upon release of a new job z at time τ, such that the following properties are satisfied:

($\mathcal{P}$1) **Future dominance.** If the algorithm A executes job j at time $t \geq \tau$, then γ_j is dominant at t.

($\mathcal{P}$2) **Past dominance.** If the algorithm A executes job j at time $t < \tau$, then γ_j remains dominant at t. In addition, the primal solution (i.e., the algorithm's scheduling decisions) for $t < \tau$ does not change due to the release of z.

($\mathcal{P}$3) **Completion.** $\gamma(t) = 0$ for all $t > C_{\max}^\tau$.

Essentially properties ($\mathcal{P}$1), ($\mathcal{P}$2) and ($\mathcal{P}$3) reflect that the statements of Lemmas 2, 4 and 5 are not tied exclusively to the total weighted flow time problem.

Theorem 2. *Any algorithm that satisfies the properties ($\mathcal{P}$1), ($\mathcal{P}$2) and ($\mathcal{P}$3) with respect to a feasible dual solution is an optimal online algorithm for fractional GFP and a $(1+\epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive algorithm for integral GFP.*

Proof. The feasibility of the dual solution is directly implied by the fact that $\lambda_j \geq 0$ (since we only increase these dual variables) and from our definition of $\gamma(t)$ which implies that

the constraints (3) are satisfied and $\gamma(t) \geq 0$. Let $C_{\max}$ be the completion time of the last job. We will assume, without loss of generality, that at time $t \leq C_{\max}$ there is at least one pending job in the schedule; otherwise, there are idle times in the schedule and we can apply the same type of analysis for jobs scheduled between consecutive idle periods.

We will first show that the primal and the dual objectives are equal. Consider a job j and let $[t_1, t_2], [t_2, t_3], \ldots, [t_{k-1}, t_k]$ be the time intervals during which j is executed. Note that $x_j(t) = 1$ for every t in these intervals (and $x_{j'}(t) = 0$ for $j' \neq j$). Hence, the contribution of j to the primal (fractional) objective is

$$\sum_{i=1}^{k-1} \delta_j \int_{t_i}^{t_{i+1}} g(t - r_j) dt$$

By properties ($\mathcal{P}1$) and ($\mathcal{P}2$), the line γ_j is dominant during the same time intervals. Thus, the contribution of job j to the dual is

$$\lambda_j p_j - \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \gamma(t) dt = \lambda_j p_j - \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \left(\lambda_j - \delta_j g(t - r_j) \right) dt$$

$$= \sum_{i=1}^{k-1} \delta_j \int_{t_i}^{t_{i+1}} g(t - r_j) dt$$

since $\sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \lambda_j dt = \lambda_j \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} x_j(t) dt = \lambda_j p_j$. The first part of the theorem follows by summing over all jobs j, and by accounting for the fact that $\int_{C_{\max}}^{\infty} \gamma(t) = 0$ (from property ($\mathcal{P}3$)).

For the second part of the theorem, consider again the time intervals during which a job j is executed. The contribution of j to the integral objective is $w_j g(C_j - r_j) = \delta_j g(C_j - r_j) p_j$. By properties ($\mathcal{P}1$) and ($\mathcal{P}2$), for any $t \in \bigcup_{i=1}^{k-1} [t_i, t_{i+1}]$ we have that $\gamma_j(t) \geq 0$. In particular, it holds for $t_k = C_j$, that is $\lambda_j \geq \delta_j g(C_j - r_j)$. Therefore, the contribution of j to the integral objective is

$$w_j g(C_j - r_j) \leq \lambda_j p_j$$

Since we consider the speed augmentation case, we will use as lower bound of the optimal solution the dual program that uses a smaller speed as explained in Section 2. By properties ($\mathcal{P}1$) and ($\mathcal{P}2$), we have $\gamma(t) = \lambda_j - \delta_j g(t - r_j) \leq \lambda_j$ during the time intervals where the job j is executed. Thus, the contribution of j to the dual objective is at least

$$\lambda_j p_j - \frac{1}{1+\epsilon} \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \gamma(t) dt \geq \lambda_j p_j - \frac{1}{1+\epsilon} \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \lambda_j dt = \frac{\epsilon}{1+\epsilon} \lambda_j p_j,$$

since $\sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \lambda_j dt = \lambda_j \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} x_j(t) dt = \lambda_j p_j$. From property ($\mathcal{P}3$) we have that $\int_{C_{\max}}^{\infty} \gamma(t) dt = 0$. Summing up over all jobs, the theorem follows. $\square$

In what follows in this section, we apply this framework in three different problems.

4.1   Online GCP with general cost functions

In this section, we consider the fractional GCP and we will show that there is an optimal primal-dual algorithm for it. The following is a linear relaxation of the problem.

$$\min \sum_{j \in \mathcal{J}} \delta_j \int_{r_j}^{\infty} g(t) x_j(t) dt$$

$$\int_{r_j}^{\infty} x_j(t) dt \geq p_j \qquad \qquad \forall j \in \mathcal{J}$$

$$\sum_{j \in \mathcal{J}} x_j(t) \leq 1 \qquad \qquad \forall t \geq 0$$

$$x_j(t) \geq 0 \qquad \qquad \forall j \in \mathcal{J}, t \geq 0$$

The dual program reads

$$\max \sum_{j \in \mathcal{J}} \lambda_j p_j - \int_0^{\infty} \gamma(t) dt$$

$$\lambda_j - \gamma(t) \leq \delta_j g(t) \qquad \qquad \forall j \in \mathcal{J}, t \geq r_j$$

$$\lambda_j \geq 0 \qquad \qquad \forall j \in \mathcal{J}$$

$$\gamma(t) \geq 0 \qquad \qquad \forall t \geq 0$$

We define $\gamma_j(t) = \lambda_j - \delta g(t)$, based on the same arguments as in Section 3. In what follows, we use Procedure 3 so to define and update the values of λ_j's for all jobs in R_τ, i.e., all jobs that are released up to time τ (completed or not).

Procedure 3 Assignment and updating of λ_j's for the set R_τ of all jobs released by time τ when a new job is released.

1: Consider jobs in R_τ in increasing order of their completion times if no new jobs are released after time τ, i.e., $C_1 < C_2 < \ldots < C_k$
2: Choose λ_k such that $\gamma_k(C_k) = 0$
3: **for** every job $j = k - 1$ to 1 **do**
4:    Let $j' \in R_\tau$ be the job scheduled right after C_j
5:    Choose the λ_j such that $\gamma_j(C_j) = \gamma_{j'}(C_j)$

The following lemma shows that the dominance condition is always satisfied for all jobs in R_τ due to Procedure 3.

Lemma 6. *For λ_j's as defined by Procedure 3, and algorithm $A \equiv HDF$, the properties ($\mathcal{P}1$) and ($\mathcal{P}2$) hold.*

Proof. Consider the jobs in R_τ in increasing order of their completion times, i.e., $C_1 < C_2 < \ldots < C_k$. Let j and $j + 1$ be two consecutive jobs in this order. By the choice of dual

variables in Procedure 3 for a time $t > \max\{r_j, r_{j+1}\}$ we have

$$
\begin{aligned}
\gamma_j(t) &= \lambda_j - \delta_j g(t) = \lambda_{j+1} - \delta_{j+1} g(C_j) + \delta_j g(C_j) - \delta_j g(t) \\
&= \lambda_{j+1} - \delta_{j+1} g(t) + (\delta_{j+1} - \delta_j)(g(t) - g(C_j)) \\
&= \gamma_{j+1}(t) + (\delta_{j+1} - \delta_j)(g(t) - g(C_j))
\end{aligned}
$$

As we follow HDF, we have that $\delta_{j+1} - \delta_j \leq 0$. Since g is a non-decreasing function, if $t < C_j$ then $\gamma_j(t) \leq \gamma_{j+1}(t)$, while if $t \geq C_j$ then $\gamma_j(t) \geq \gamma_{j+1}(t)$. In other words, γ_j intersects γ_{j+1} at C_j and that is indeed the unique intersection point between the two curves for every $j, j+1$. Note that the uniqueness property does not necessarily hold in the settings with flow time objectives, which explains the simplicity of the dual construction for objectives on completion times compared to the ones on flow times. Then, the dominance property follows using the same arguments as in the proof of Lemma 2.

Since the claim holds for every j, we deduce that $\gamma_j(t) \geq \gamma_{j'}(t)$ for every $j' \in R_\tau$ and every time t during the execution of job j. $\qquad\square$

Property $(\mathcal{P}3)$ is straightforward from Procedure 3 and hence we obtain the following theorem.

Theorem 3. *The primal-dual algorithm $A \equiv HDF$ is an optimal algorithm for fractional GCP and a $(1+\epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive algorithm for integral GCP.*

4.2 Online GFP with convex/concave cost functions and equal density jobs

In this section, we consider GFP with cost function g that is either a convex or a concave non-decreasing function with $g(0) = 0$. Moreover, we assume that all jobs have the same density, i.e., $\delta_j = \delta$ for each $j \in \mathcal{J}$. For both convex and concave cases, we will show that there is an optimal primal-dual algorithm for minimizing the total fractional cost. For convex functions, this algorithm has to be the FIFO policy, whereas for concave functions the optimal algorithm has to be the LIFO policy.

For both problems, we define $\gamma_j = \lambda_j - \delta g(t - r_j)$ (based on the same arguments as in Section 3 and the constraints (3)). In what follows, we use Procedures 1 and 2 so to define and update the values of λ_j's.

Lemma 7. *For λ_j's as defined by Procedure 1, property $(\mathcal{P}1)$ holds if:*
(i) g is convex, all jobs have equal density and $A \equiv FIFO$,
(ii) g is concave, all jobs have equal density and $A \equiv LIFO$.

Proof. We will follow the proof of Lemma 2 by showing (4) and (5). Similarly all jobs have the same density, we obtain

$$
\begin{aligned}
\lambda_j - \delta g(t - r_j) = \lambda_{j+1} - \delta g(t - r_{j+1}) \\
+ \delta[(g(C_j - r_j) - g(C_j - r_{j+1})) - (g(t - r_j) - g(t - r_{j+1}))]
\end{aligned}
$$

(i) Suppose that g is convex. Since $C_j < C_{j+1}$ and the scheduling algorithm is FIFO, we have that $r_j \leq r_{j+1}$. Thus, $g(C_j - r_j) \geq g(C_j - r_{j+1})$ and $g(t - r_j) \geq g(t - r_{j+1})$, since g

15

is non-decreasing. If $t \geq C_j$, then by convexity it follows that $g(C_j - r_j) - g(C_j - r_{j+1}) \leq g(t - r_j) - g(t - r_{j+1})$, and hence (4) holds. If $t < C_j$, then by convexity it holds that $g(C_j - r_j) - g(C_j - r_{j+1}) \geq g(t - r_j) - g(t - r_{j+1})$, and hence (5) holds.

(ii) Suppose that g is concave. Since $C_j < C_{j+1}$ and the scheduling algorithm is LIFO, we have that $r_j \geq r_{j+1}$. Thus, $g(C_j - r_j) \leq g(C_j - r_{j+1})$ and $g(t - r_j) \leq g(t - r_{j+1})$, since g is non-decreasing. If $t \geq C_j$, then by concavity it follows that $g(C_j - r_j) - g(C_j - r_{j+1}) \leq g(t - r_j) - g(t - r_{j+1})$, and hence (4) holds. If $t < C_j$, then by concavity it holds that $g(C_j - r_j) - g(C_j - r_{j+1}) \geq g(t - r_j) - g(t - r_{j+1})$, and hence (5) holds. $\qquad\square$

Lemma 8. *For λ_j's defined by Procedures 1 and 2, property ($\mathcal{P}2$) holds if*
(i) g is convex, all jobs have equal density and $A \equiv FIFO$,
(ii) g is concave, all jobs have equal density and $A \equiv LIFO$.

Proof. We rely on the following claim. Recall that P_τ denotes the set of the algorithm's pending jobs at the release of z at time τ.

Claim 4. *Let j_1 and j_2 be two jobs in P_τ such that $C_{j_1} < C_{j_2}$. Then $\Delta_{j_1} \geq \Delta_{j_2}$.*

Proof of claim.
(i) Suppose that g is convex and we follow the FIFO policy, then $C_{j_1} < C_{j_2} < C_z$, In this case, both γ_{j_1} and γ_{j_2} are moved up by $\gamma_z(C_z - p_z - r_z) - \gamma_z(C_z - r_z)$. Therefore, $\Delta_{j_1} = \Delta_{j_2} = \delta(g(C_z - r_z) - g(C_z - p_z - r_z))$.

(ii) Suppose that g is concave and we follow the LIFO policy. Then $C_z < C_{j_1} < C_{j_2}$. Hence, the completion times of both j_1 and j_2 are delayed by p_z by the algorithm. In other words, before the release of z the completion times of j_1 and j_2 were $C_{j_1} - p_z$ and $C_{j_2} - p_z$, respectively. Moreover, the relative ordering of all jobs in $P_\tau \setminus \{z\}$ (as done by the algorithm) remains the same before and after the release of z. Thus, by Procedure 1, we have $\lambda_{j_1} - \delta g(C_{j_1} - r_{j_1}) = \lambda'_{j_1} - \delta g(C_{j_1} - p_z - r_{j_1})$, where λ'_{j_1} is the value of the dual variable of j_1 before the release of z. Hence, $\Delta_{j_1} = \delta(g(C_{j_1} - r_{j_1}) - g(C_{j_1} - p_z - r_{j_1}))$. Similarly, $\Delta_{j_2} = \delta(g(C_{j_2} - r_{j_2}) - g(C_{j_2} - p_z - r_{j_2}))$. Therefore, $\Delta_{j_1} \geq \Delta_{j_2}$ since g is concave and by the LIFO algorithm $r_{j_1} \geq r_{j_2}$. $\qquad\square$

Note that Claims 2 and 3 also hold for the problems we study in this section. Therefore, by applying the same arguments as in the proof of Lemma 4, and by replacing Claim 1 with Claim 4, we arrive at the same conclusion. $\qquad\square$

The proof of the following lemma directly follows from Lemmas 7 and 8 as in the linear case.

Lemma 9. *For λ_j's as defined by Procedures 1 and 2, the property ($\mathcal{P}3$) holds if (i) g is convex and all jobs have equal density; or (ii) g is concave and all jobs have equal density.*

By Lemmas 7, 8 and 9, the properties (P_1), (P_2) and (P_3), respectively, are satisfied, and hence we obtain the following theorem.

Theorem 4. *The primal-dual algorithm $A \equiv FIFO$ (resp. $A \equiv LIFO$) is an optimal online algorithm for fractional GFP and a $(1+\epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive algorithm for integral GFP, when we consider convex (reps. concave) cost functions and jobs of equal density.*

16

4.3   PSP with positive constraint coefficients

In this section we study the PSP problem (defined formally in Section 1), assuming constraints of the form $Bx \leq 1$, and $b_{ij} > 0$ for every i, j. We denote by b_j, B_j the smallest and largest element of each column of B, respectively, i.e., $b_j = \min_i b_{ij}$ and $B_j = \max_i b_{ij}$. The primal LP relaxation of the problem, and its dual are as follows:

$$\min \sum_{j \in \mathcal{J}} \delta_j \int_{r_j}^{\infty} (t - r_j) x_j(t) dt$$

$$\int_{r_j}^{\infty} x_j(t) dt \geq p_j \qquad \forall j \in \mathcal{J}$$

$$\sum_{j \in \mathcal{J}} b_{ij} x_j(t) \leq 1 \qquad \forall t \geq 0, \forall 1 \leq i \leq m$$

$$x_j(t) \geq 0 \qquad \forall j \in \mathcal{J}, t \geq 0$$

$$\max \sum_{j \in \mathcal{J}} \lambda_j p_j - \sum_{i=1}^{m} \int_0^{\infty} \gamma_i(t) dt$$

$$\lambda_j - \sum_i b_{ij} \gamma_i(t) \leq \delta_j(t - r_j) \qquad \forall j \in \mathcal{J}, t \geq r_j$$

$$\lambda_j \geq 0 \qquad \forall j \in \mathcal{J}$$

$$\gamma_i(t) \geq 0 \qquad \forall t \geq 0$$

We begin with the intuition behind the analysis, and how one can exploit the ideas of the analysis of HDF of Section 3, in the context of minimum total flow. The essential difference between the two problems is the set of packing constraints that are present in the PSP formulation. This difference manifests itself in the dual with the constraints $\lambda_j - \sum_i b_{ij} \gamma_i(t) \leq \delta_j(t - r_j)$, for all $j \in \mathcal{J}, t \geq 0$. In contrast, the dual LP of the minimum total weighted flow time problem has corresponding constraints $\lambda_j - \gamma(t) \leq \delta_j(t - r_j)$. At this point, one would be motivated to define $\mu(t) = \sum_{i=1}^{m} b_{ij} \gamma_i(t)$, and then view this $\mu(t)$ as the variable $\gamma(t)$ in the dual LP formulation of the minimum total flow problem (and thus one would proceed as in the analysis of HDF). However, a complication arises: it is not clear how to assign the variables $\gamma_i(t)$, for given $\mu(t)$. Because of this, we follow a different way (which will guarantee the feasibility of the $\gamma_i(t)$'s): Instead of satisfying the constraint $\lambda_j - \sum_{i=1}^{m} b_{ij} \gamma_i(t) \leq \delta_j(t - r_j)$ we will satisfy the stronger constraint $\lambda_j - b_j \sum_{i=1}^{m} \gamma_i(t) \leq \delta_j(t - r_j)$. At an intuitive level, we will satisfy the constraints $\lambda_j' - \mu'(t) \leq \delta_j'(t - r_j)$, where $\delta_j' = \delta_j/b_j$ and $\mu'(t)$ and λ_j' are variables. Using the same scheme as in the setting of linear functions we can construct λ_j' (so $\mu_j'(t)$), so that the properties $(\mathcal{P}1), (\mathcal{P}2), (\mathcal{P}3)$ are satisfied. Particularly, if job j is processed at time t then μ_j' is dominant at t, i.e., $\mu_j'(t) = \lambda_j' - \delta_j'(t - r_j) = \mu'(t) = \max_k \mu_k'(t)$. Moreover, $\lambda_j' - \mu'(t) \leq \delta_j'(t - r_j)$ for every job j and $t \geq r_j$.

As a last step, we need to translate the above dual variables to the PSP problem. Define $\lambda_j = \lambda_j' b_j$ for every job j. Let j be the job processed at time t and the constraint i is a tight constraint at time t, i.e., $b_{ij} = B_j$. Set $\gamma_i(t) = \mu'(t)$ and $\gamma_{i'}(t) = 0$ for $i' \neq i$.

Algorithm. The above discussion implies an adaptation of HDF to the PSP problem as follows: At any time t, we schedule job the job j which attains the highest ratio δ_j/b_j among all pending jobs, at rate $x_j(t) = 1/B_j$.

Based on this algorithm, we prove Theorem 5 using similar arguments as for the proof of Theorem 1.

Theorem 5. *For the online PSP problem with constraints $Bx \leq 1$ and $b_{ij} > 0$ for every i, j, an adaptation of HDF is $\max_j\{B_j/b_j\}$-speed 1-competitive for fractional weighted flow time and $\max_j\{(1+\epsilon)B_j/b_j\}$-speed $(1+\epsilon)/\epsilon$-competitive for integral weighted flow time.*

Proof. We first show the feasibility of the dual solution. For every job j and time t, we have

$$\lambda_j/b_j - \delta_j/b_j \cdot (t - r_j) = \mu_j'(t) \leq \mu'(t) = \sum_i \gamma_i(t) \leq \sum_i b_{ij}/b_j \cdot \gamma_i(t)$$

where the last inequality is due to $b_{ij} \geq \min_i b_{ij} = b_j > 0$.

We will now bound the integral and fractional primal cost (with unit speed) by the dual cost (with speed b_j/B_j). Consider a job j and let $[t_1, t_2], [t_2, t_3], \ldots, [t_{m-1}, t_m]$ be the time intervals during which j is executed. Note that $x_j(t) = 1/B_j$ for every t during the intervals (and $x_{j'}(t) = 0$ for $j' \neq j$). The contribution of job j to the fractional primal objective is

$$\sum_{a=1}^{m-1} \int_{t_a}^{t_{a+1}} \delta_j(t - r_j)x_j(t)dt = \sum_{a=1}^{m-1} \int_{t_a}^{t_{a+1}} \delta_j'(t - r_j)b_j/B_j dt$$

The contribution of j to the integral primal objective is

$$w_j(C_j - r_j) = p_j\delta_j(C_j - r_j) \leq p_j\lambda_j$$

since the curve $\lambda_j' - \delta_j'(t - r_j) \geq 0$ for every t during which j is executed, particularly for $t = C_j$ (and note that $\lambda_j = b_j\lambda_j'$ and $\delta_j = b_j\delta_j'$).

Assuming that job j is processed on the machine with speed b_j/B_j, the contribution of job j to the dual is

$$\lambda_j p_j - \frac{b_j}{B_j}\sum_{a=1}^{m-1}\int_{t_a}^{t_{a+1}}\sum_i \gamma_i(t)dt = \sum_{a=1}^{m-1}\int_{t_a}^{t_{a+1}}\left(\lambda_j x_j(t) - \frac{b_j}{B_j}\sum_i \gamma_i(t)\right)dt$$

$$= \sum_{a=1}^{m-1}\int_{t_a}^{t_{a+1}}\frac{b_j}{B_j}\left(\lambda_j' - \mu'(t)\right)dt = \sum_{a=1}^{m-1}\int_{t_a}^{t_{a+1}}\frac{b_j}{B_j}\delta_j'(t - r_j)dt$$

where the first equality is due to $\sum_{a=1}^{m-1}\int_{t_a}^{t_{a+1}} = p_j$; the second equality follows by $x_j(t) = 1/B_j$ and the definitions of dual variables; and the last equality holds by the dominance property. Hence, the contribution of job j to the fractional primal cost and that to the dual one (with speed b_j/B_j) are equal. As the latter holds for every job, the first statement follows.

18

We can bound differently the contribution of j to the dual.

$$\lambda_j p_j - \frac{b_j}{(1+\epsilon)B_j} \sum_{a=1}^{m-1} \int_{t_a}^{t_{a+1}} \sum_i \gamma_i(t) dt \geq \lambda_j p_j - \frac{1}{(1+\epsilon)B_j} \sum_{a=1}^{m-1} \int_{t_a}^{t_{a+1}} \lambda_j dt$$

$$= \lambda_j p_j - \frac{\lambda_j}{1+\epsilon} \sum_{a=1}^{m-1} \int_{t_a}^{t_{a+1}} x_j(t) dt = \frac{\epsilon}{1+\epsilon} \lambda_j p_j$$

where the first inequality due to $\lambda_j/b_j = \lambda'_j \geq \mu'(t) = \sum_i \gamma_i(t)$ for every t during which j is processed; the first equality holds since job j is processed at rate $x_j(t) = 1/B_j$; and the last equality follows $\int x_j(t) dt = p_j$. Therefore, the contribution of job j to the integral primal cost is at most $(1+\epsilon)/\epsilon$ that to the dual objective. Again, summing over all jobs, the second statement holds. $\qquad\square$

5 A generalized framework using dual-fitting

In this section we relax certain properties as established in Section 4 in order to generalize our framework and apply it to the integral variant of more problems. Our analysis here is based on the dual-fitting paradigm, since the analysis of Section 3 provides us with intuition about the geometric interpretation of the primal and dual objectives. We consider, as concrete applications, GFP for given cost functions g. We again associate with each job j the curve γ_j and set $\gamma(t) = \max\{0, \max_{j\in\mathcal{J}:r_j \leq t}\{\gamma_j(t)\}\}$. Then, we need to define how to update the dual variables λ_j, upon release of a new job z at time τ, such that the following properties are satisfied:

(Q1) If the algorithm A schedules job j at time $t \geq \tau$ then $\gamma_j(t) \geq 0$ and $\lambda_j \geq \gamma_{j'}(t)$ for every other pending job j' at time t.

(Q2) If the algorithm A schedules job j at time $t < \tau$, then $\gamma_j(t) \geq 0$ and $\lambda_j \geq \gamma_{j'}(t)$ for every other pending job j' at time t. In addition, the primal solution for $t < \tau$ is not affected by the release of z.

(Q3) $\gamma(t) = 0$ for all $t > C^\tau_{\max}$.

Note that (Q1) is relaxed with respect to property (P1) of Section 4, since it describes a weaker dominance condition. Informally, (Q1) guarantees that for any time t the job that is scheduled at t does not have negative contributions in the dual. On the other hand, property (Q2) is the counterpart of (Q1), for times $t < \tau$ (similar to the relation between (P1) and (P2)). Finally, note that even though the relaxed properties do not guarantee anymore the optimality for the fractional objectives, the following theorem (Theorem 6) establishes exactly the same result as Theorem 2 for the *integral* objectives. This is because in the second part of the proof of Theorem 2 we only require that when j is executed at time t then $\lambda_j \geq \gamma_{j'}(t)$ for every other pending job j' at t, which is in fact guaranteed by properties (Q1) and (Q2).

Theorem 6. *Any algorithm that satisfies the properties (Q1), (Q2) and (Q3) with respect to a feasible dual solution is a $(1+\epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive algorithm for integral GFP with general cost functions g.*

Proof. The feasibility of the dual solution is straightforward. We denote by $C_{\max}$ the completion time of the last job. As in the proof of Theorem 2, we can assume, without loss of generality, that at time $t \leq C_{\max}$ there is at least one pending job in the schedule.

Consider now a job j and let $[t_1, t_2], [t_2, t_3], \ldots, [t_{k-1}, t_k]$ be the time intervals during which j is executed. Note that $x_j(t) = 1$ for every t during the intervals (and $x_{j'}(t) = 0$ for $j' \neq j$). Hence, the contribution of j to the primal objective is

$$\sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \delta_j g(t - r_j) dt \leq \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \lambda_j dt = \lambda_j p_j$$

where the inequality follows from properties $(\mathcal{Q}1)$ and $(\mathcal{Q}2)$, and specifically due to the constraint that $\gamma_j(t) \geq 0$.

From properties $(\mathcal{Q}1)$ and $(\mathcal{Q}2)$, we have $\lambda_j \geq \gamma(t)$ during the same time intervals. Since we assume that the optimal solution uses a speed of $\frac{1}{1+\epsilon}$, the contribution of job j to the dual objective is at least

$$\lambda_j p_j - \frac{1}{1+\epsilon} \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \gamma(t) dt \geq \lambda_j p_j - \frac{1}{1+\epsilon} \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} \lambda_j dt = \frac{\epsilon}{1+\epsilon} \lambda_j p_j$$

In addition, from property $(\mathcal{Q}3)$ we have $\int_{C_{\max}}^{\infty} \gamma(t) dt = 0$. Summing up over all jobs j, the theorem follows. $\qquad \square$

5.1   Online GFP with general cost functions and equal-density jobs

We will analyze the FIFO algorithm using dual fitting. We will use a single procedure, namely Procedure 4, for the assignment of the λ_j variables for each job j released by time τ. We denote this set of jobs by R_τ, and $k = |R_\tau|$.

Procedure 4 Assignment and updating of λ_j's for the set R_τ of all jobs released by time τ when a new job is released.

1: Consider jobs in R_τ in increasing order of completion times if no new jobs are released after time τ, i.e., $C_1 < C_2 < \ldots < C_k$
2: Choose λ_k such that $\gamma_k(C_k) = 0$
3: **for** $j = k - 1$ to 1 **do**
4:    Choose the maximum possible λ_j such that for every $t \geq C_j$, $\gamma_j(t) \leq \gamma_{j+1}(t)$
5:    **if** $\gamma_j(C_j) < 0$ **then**
6:       Choose λ_j such that $\gamma_j(C_j) = 0$

We will need first the following simple proposition:

Proposition 1. *Let $i, j \in \mathcal{J}$ be two jobs such that $r_i \leq r_j$ and suppose that there is a time $t_0 \geq r_j$ such that $\gamma_i(t_0) \geq \gamma_j(t_0)$. Then, $\lambda_i \geq \lambda_j$.*

Proof. By the statement of the proposition we have that $\lambda_i - \delta g(t_0 - r_i) \geq \lambda_j - \delta g(t_0 - r_j)$. Since the function g is non-decreasing and $r_i \leq r_j$, it must be that $\lambda_i \geq \lambda_j$. $\qquad \square$

The following lemma is instrumental in establishing the desired properties.

Lemma 10. *For every job j in R_τ, $\lambda_j \geq \gamma_i(C_{j-1})$.*

Proof. Consider the jobs in R_τ in increasing order of completion times. We will prove the lemma by considering two cases: for jobs $i < j$, and for jobs $i > j$ (according to the above ordering). In both cases we apply induction on i.

Case 1: $i \leq j$. For base case $i = j$, the inequality becomes $\lambda_j \geq \gamma_j(C_{j-1})$ which trivially holds since $\lambda_j \geq \gamma_j(t)$ for all t.

For the induction step, suppose that the lemma is true for a job $i < j$, that is $\lambda_j \geq \gamma_i(C_{j-1})$. Recall that job $i - 1 \in R_\tau$ is the last job that is completed before i. If λ_{i-1} had not been set in line 6 of Procedure 4 then it has been set in line 4 of the procedure; thus, from the induction hypothesis, $\gamma_{i-1}(C_{i-1}) \leq \gamma_i(C_{i-1}) \leq \lambda_j$. On the other hand, if λ_{i-1} is set in line 6 of Procedure 4 then $\gamma_{i-1}(C_{i-1}) = 0$. Hence $\lambda_j > 0 = \gamma_{i-1}(C_{i-1}) \geq \gamma_{i-1}(C_{j-1})$ where the last inequality is due to the fact that γ_{i-1} is a non-increasing function.

Case 2: $i \geq j$. We will show the stronger claim that $\lambda_j \geq \lambda_i$ for every $i \geq j$. This suffices since $\lambda_i \geq \gamma_i(t)$ for every t. The base case in the induction, namely the case $i = j$ is trivial.

Assume that the claim is true for a job $i > j$, that is $\lambda_j \geq \lambda_i$. We need to show that $\lambda_j \geq \gamma_{i+1}(C_{j-1})$. If λ_i is not set in line 6 of Procedure 4 (and thus set in line 4 instead), then there is time t_0 such that $\gamma_i(t_0) = \gamma_{i+1}(t_0)$ by the maximality in the choice λ_j. On the other hand, if λ_i is set in line 6 of Procedure 4 then there is at least one time t_1, with $C_j \leq t_1 \leq t_0$ such that $\gamma_i(t_1) \geq \gamma_{i+1}(t_1)$. In both cases, from Proposition 1 we deduce that $\lambda_i \geq \lambda_{i+1}$. Moreover, $\lambda_i \leq \lambda_j$ by the induction hypothesis. The claim follows. $\square$

Lemma 11. *Procedure 4 satisfies properties $(\mathcal{Q}1)$, $(\mathcal{Q}2)$ and $(\mathcal{Q}3)$.*

Proof. From Lemma 10 it follows that properties $(\mathcal{Q}1)$ and $(\mathcal{Q}2)$ are satisfied. More precisely, observe that by line 6 of Procedure 4 we have $\gamma_j(t) \geq 0$ for any $t \leq C_j$. It remains to show that for any time t at which a job j is executed, we have $\lambda_j \geq \gamma(t) = \max_{i \in \mathcal{J}} \gamma_i(t)$. Since g is non-decreasing, it follows that $\gamma_j(t)$ is non-increasing, hence it suffices to show the above only for $t = C_{j-1}$; this is indeed established by Lemma 10.

Moreover, concerning property $(\mathcal{Q}3)$, we observe that for all jobs $j \in R_\tau$ for which λ_j is set in line 6 of Procedure 4, the property follows straightforwardly. For all remaining jobs in R_τ, the property holds using the same arguments as in Lemma 5 for the minimum flow time problem. $\square$

The above lemma in conjunction with Theorem 6 lead to the following result.

Theorem 7. *FIFO is a $(1 + \epsilon)$-speed $\frac{1+\epsilon}{\epsilon}$-competitive for integral GFP with general cost functions and equal-density jobs.*

5.2 Online GFP with concave cost functions

We will analyze the HDF algorithm using dual fitting. As in Section 3, we will employ two procedures for maintaining the dual variables λ_j. The first one is Procedure 5 which updates the λ_j's for $j \in P_\tau$. The second procedure updates the λ_j's for $j \in \mathcal{J}_\tau$; this procedure is identical to Procedure 2 of Section 3.

21

The intuition behind Procedure 5 is to ensure property ($\mathcal{Q}1$) that is, $\gamma_j(t) \geq 0$ and $\lambda_j \geq \gamma_{j'}(t)$ for all $j' \in P_\tau$, which in some sense is the "hard" property to maintain. Specifically, for given job j there is a set of jobs A (initialized in line 4) for which the property does not hold. The while loop in the procedure decreases the λ values of jobs in A so as to rectify this situation (see line 6(ii)). However, this decrement may, in turn, invalidate this property for some jobs b (see line 6(i)). These jobs are then added in the set of "problematic" jobs A and we continue until no problematic jobs are left. One can formally argue that this procedure terminates.

Procedure 5 Assignment of dual variables λ_j for all $j \in P_\tau$ at the arrival of a new job.

1: Consider the jobs in P_τ in increasing order of completion times if no new jobs are released after time τ, i.e., $C_1 < C_2 < \ldots < C_k$
2: For every $1 \leq j \leq k$ choose λ_j such that $\gamma_j(C_k) = 0$
3: **for** $j = 2$ to k **do**
4:     Define $A := \{\text{jobs } 1 \leq a \leq j - 1 : \gamma_a(C_{j-1}) > \lambda_j\}$
5:     **while** $A \neq \emptyset$ **do**
6:         Continuously reduce λ_a by the same amount for all jobs $a \in A$ until:
        (i) $\exists\, a \in A$ and $b \in P_\tau \setminus A$ with $b < a$ such that $\lambda_a = \gamma_b(C_{a-1})$; then $A \leftarrow A \cup \{b\}$
        (ii) $\exists\, a \in A$ such that $\gamma_a(C_{j-1}) = \lambda_j$; then $A \leftarrow A \setminus \{a\}$

We will first need to show the following technical lemma:

Lemma 12. *Let $a, b \in \mathcal{J}$ such that $\delta_a \geq \delta_b$ and suppose that there is a time $t_0 \geq \max\{r_a, r_b\}$ such that $\gamma_a(t_0) \geq \gamma_b(t_0)$. Then $\lambda_a \geq \gamma_b(t)$ for every $t \geq \max\{r_a, r_b\}$.*

Proof. By assumption, we have $\lambda_a - \delta_a g(t_0 - r_a) \geq \lambda_b - \delta_b g(t_0 - r_b)$. If $r_a \leq r_b$ then $g(t_0 - r_a) \geq g(t_0 - r_b)$ so $\delta_a g(t_0 - r_a) \geq \delta_b g(t_0 - r_b)$. Hence $\lambda_a \geq \lambda_b \geq \gamma_b(t)$ for every t. Remains then to consider the case $r_a > r_b$. Since $\gamma_b(t)$ is non-increasing, suffices to prove that $\lambda_a \geq \gamma_b(r_a)$. Since $\lambda_a \geq \lambda_b - \delta_b g(t_0 - r_b) + \delta_a g(t_0 - r_a)$, it will suffice to show that

$$\lambda_b - \delta_b g(t_0 - r_b) + \delta_a g(t_0 - r_a) \geq \lambda_b - \delta_b g(r_a - r_b)$$
$$\Leftrightarrow \qquad \delta_a g(t_0 - r_a) \geq \delta_b(g(t_0 - r_b) - g(r_a - r_b))$$

Note that g is concave and $g(0) = 0$ so g is sub-additive (sub-linear). Therefore, the right-hand side is upper bounded by $\delta_b g(t_0 - r_a)$, which is at most the left-hand side. $\square$

The following lemma is related to Procedure 5, which rectifies property ($\mathcal{Q}1$) iteratively. It is not hard to see that this procedure terminates, since whenever a job a is added to the set A, then λ_a is decreased. Note that a job may be added and removed several times however, λ_a cannot be smaller than zero.

Lemma 13. *At the end of the for-loop for job j (line 3) of Procedure 5:*
(i) For any two jobs $a, b \in P_\tau$ such that $1 \leq a < b \leq j$ (i.e, $\delta_a \geq \delta_b$), we have $\lambda_a \geq \gamma_b(t)$ for all $t \in [C_{a-1}, C_a]$; and
(ii) For any two jobs $a, b \in P_\tau$ such that $1 \leq b < a \leq j$ (i.e, $\delta_a \leq \delta_b$), we have $\lambda_a \geq \gamma_b(t)$ for all $t \in [C_{a-1}, C_a]$.

Proof.

(i) Proof by induction on j. For the base case ($j = 1$) note that in line 2 of the procedure, $\gamma_j(C_k) = 0$ for every $j \in P_\tau$; hence from Lemma 12 the base case is satisfied. Suppose that the claim holds at the end of the for-loop for job $j - 1$; we will call this for-loop the *iteration* for job $j - 1$. Consider the iteration of job j. Suppose that during this iteration, λ_a has decreased more than λ_b has, since otherwise the claim follows directly from the induction hypothesis. This implies that at some moment during the execution of the while loop, $a \in A$ and $b \notin A$. At that moment, we deduce that $\gamma_b(C_{j-1}) \leq \lambda_j < \gamma_a(C_{j-1})$. However, by line 6(ii), λ_a stops decreasing at the point in which $\lambda_j = \gamma_a(C_{j-1})$. Therefore, at the end of the iteration of job j, we have $\gamma_b(C_{j-1}) \leq \gamma_a(C_{j-1})$. Applying Lemma 12 by choosing $t_0 = C_{j-1}$, it holds that $\lambda_a \geq \gamma_b(t)$ for $t \in [C_{a-1}, C_a]$.

(ii) The proof is again by induction on j. The base case, $j = 1$, holds since no other jobs in P_τ have higher density than job 1. Assume that the statement holds at the end of the for-loop for job $j - 1$. During the iteration of the for-loop for job j, a set A contains jobs $a < j$ such that $\lambda_j < \gamma_a(C_{j-1})$. By the procedure, λ_a for every $a \in A$ is decreased until $\lambda_j = \gamma_a(C_{j-1})$. So at the end of the iteration, $\lambda_j \geq \gamma_a(C_{j-1})$ for every $a < j$. It remains to show that at the end of the iteration, $\lambda_a \geq \gamma_b(C_{a-1})$ for jobs $b < a < j$. Let $b < a < j$ be two arbitrary jobs. By the induction hypothesis, $\lambda_a \geq \gamma_b(C_{a-1})$ holds before the iteration. During the iteration, if λ_a is not modified then the inequality remains true (since λ_b is not increased). Otherwise, a must be added to A at some moment. As λ_a is decreased so probably at some later moment $\lambda_a = \gamma_b(C_{a-1})$. However, job b will be added to A and the λ-values of both jobs will be decreased by the same amount. Therefore, at the end of the iteration $\lambda_a \geq \gamma_b(C_{a-1})$ (the induction step is done). $\square$

Lemma 14. *Procedures 5 and 2 combined satisfy properties* ($\mathcal{Q}1$),($\mathcal{Q}2$) *and* ($\mathcal{Q}3$).

Proof. We will first show that at the end of Procedure 5 the dual solution satisfies property ($\mathcal{Q}1$). From Lemma 13, given $j \in P_\tau$, we have $\lambda_j \geq \gamma_a(t)$ for every job $a \in P_\tau$ and for every time $t \in [C_{j-1}, C_j]$. It remains to show that $\gamma_j(t) \geq 0$ for $t \in [C_{j-1}, C_j]$. Since γ_j is non-increasing, suffices to show that $\gamma_j(C_j) \geq 0$. After line 2 of the procedure, $\gamma_j(C_j) \geq \gamma_j(C_k) \geq 0$. Note that subsequently λ_j may be decreased; however, we will argue that $\gamma_j(C_j) \geq 0$. Job j may be added in A in either line 2 or in line 6(i) of the procedure, thus λ_j may be decreased only if there is a job $j' > j$ such that $\lambda_{j'} < \gamma_j(C_{j'-1})$. Since $C_{j'-1} \geq C_j$ and $\gamma_j(t)$ is a decreasing in t we have that $\lambda_{j'} < \gamma_j(C_j)$. Moreover, λ_j will continue decreasing until $\lambda_{j'} = \gamma_j(C_{j'-1})$ (line 6(ii)). Note that λ_j may be decreased in many iterations and thus the condition $\lambda_{j'} = \gamma_j(C_{j'-1})$ may hold for more than one job j'; let j^* denote the job of the last iteration for which $\lambda_{j^*} = \gamma_j(C_{j^*-1})$. Since $\lambda_{j^*} \geq 0$ we obtain $\gamma_j(C_j) \geq \lambda_{j^*} \geq 0$. Hence we showed that at the end of Procedure 5 the dual solution satisfies property ($\mathcal{Q}1$).

Moreover, property ($\mathcal{Q}2$) is a relaxed variant of ($\mathcal{P}2$), therefore Procedure 2 guarantees ($\mathcal{Q}2$). In addition, Procedures 5 and 2 satisfy ($\mathcal{Q}3$), using very similar arguments as in the proof of Lemma 5. $\square$

The above lemma in conjunction with Theorem 6 lead to the following result.

Theorem 8. *HDF is a* $(1 + \epsilon)$*-speed* $\frac{1+\epsilon}{\epsilon}$*-competitive for integral GFP with concave cost functions.*

6 Online JDGFP with differentiable concave cost functions

We consider the online JDGFP, assuming that for each job j, the cost function g_j is concave and differentiable. Instead of analyzing the fractional objectives and rounding to integral ones (as in previous sections), we study directly the integral objective by considering a non-convex relaxation. Let $x_j(t)$ be the variable indicating the execution rate of job j at time t. Let C_j be the variable representing the completion time of job j. We have the following non-convex relaxation:

$$\min \sum_{j \in \mathcal{J}} \delta_j \int_{r_j}^{C_j} g_j(C_j - r_j) x_j(t) dt$$

$$\int_{r_j}^{C_j} x_j(t) dt = p_j \qquad\qquad \forall j \in \mathcal{J}$$

$$\sum_{j \in \mathcal{J}} x_j(t) \leq 1 \qquad\qquad \forall t \geq 0$$

$$x_j(t) \geq 0 \qquad\qquad \forall j \in \mathcal{J}, t \geq 0$$

By associating the dual variables λ_j and $\gamma(t)$ with the first and second constraints respectively, we obtain the Lagrangian dual program $\max_{\lambda,\gamma} \min_{x,C} L(\lambda, \gamma, x, C)$ where the Lagrangian function $L(\lambda, \gamma, x, C)$ is equal to

$$L(\lambda, \gamma, x, C) = \sum_j \delta_j \int_{r_j}^{C_j} g_j(C_j - r_j) x_j(t) dt + \sum_j \lambda_j \left(p_j - \int_{r_j}^{C_j} x_j(t) dt \right)$$

$$+ \int_0^\infty \gamma(t) \left(\sum_j x_j(t) - 1 \right) dt.$$

Next, we describe the algorithm and its analysis, which is inspired by [15, 16].

Algorithm. Let $k = \lceil 2/\epsilon \rceil$. We write $j' \prec j$ if $r_{j'} \leq r_j$, breaking ties arbitrarily. Let $G_j(t) = \sum_{a \preceq j} w_a g'_a(t - r_a)$ where the sum is taken over pending jobs a at time t. Note that as g_a's are concave functions, $G_j(t)$ is non-increasing (with respect to t). Informally, each term $w_a g'_a(t - r_a)$ represents the rate of the contribution of pending job a at time t to the total cost. Thus $G_j(t)$ stands for the contribution rate of pending jobs released before j (including j) at time t to the total cost. Whenever it is clear from context, we omit the time parameter. At time t, the algorithm processes job j at rate proportional to $G_j(t)^k - G_{j+1}(t)^k$. In other words, the rate of job j at time t is $\nu_j(t) = (G_j(t)^k - G_{j+1}(t)^k)/G(t)^k$, where $G(t) = \sum_a w_a g'_a(t - r_a)$, and the sum is taken over pending jobs a at time t.

Next, we define the dual variables. Define $\gamma(t) = 0$. Moreover, define λ_j such that $\lambda_j p_j = \frac{1}{k+1} \int_{r_j}^{C_j} \left(\nu_j(t) \sum_{a \preceq j} w_a g'_a(t - r_a) + w_j g'_j(t - r_j) \sum_{a \prec j} \nu_a(t) \right) dt$, where C_j is the completion time of job j. In the following we will bound the total cost of the algorithm by the Lagrangian dual value. Let $\mathcal{F}$ denote the total cost of the algorithm. We will use the definitions of the dual variables to derive first some essential properties (Lemma 15 and 16).

Lemma 15. $(k+1)\sum_j \lambda_j p_j = \int_0^\infty G(t)dt = \mathcal{F}$.

Proof. We first show that $(k+1)\sum_j \lambda_j p_j = \mathcal{F}$. Consider the term $w_j g_j'(t-r_j)$ in the sum $(k+1)\sum_j \lambda_j p_j$ for $r_j \le t \le C_j$. The coefficient of this term in the sum is equal to $\sum_{a \prec j} \nu_a(t) + \sum_{a \succeq j} \nu_a(t) = 1$. Therefore,

$$(k+1)\sum_j \lambda_j p_j = \sum_j \int_{r_j}^{C_j} w_j g_j'(t-r_j)dt = \sum_j w_j g_j(C_j - r_j).$$

Last, note that the identity $\int_0^\infty G(t)dt = \sum_j w_j g_j(C_j - r_j)$ is straightforward from the definition of $G(t)$. $\qquad\square$

Lemma 16. *For any time $\tau \ge r_j$, it holds that $\lambda_j \le \delta_j g_j(\tau - r_j) + \frac{1}{k}G(\tau)$.*

Proof. The proof is by the same scheme as in [16]. We nevertheless present a complete proof, which follows from the following two claims, namely Claim 5 and Claim 6.

Claim 5. *For any time $\tau \ge r_j$,*

$$\frac{1}{p_j}\int_{r_j}^\tau \left(\nu_j(t)\sum_{a \preceq j} w_a g_a'(t-r_a) + w_j g_j'(t-r_j)\sum_{a \prec j} \nu_a(t) \right) dt \le (k+1)\delta_j g_j(\tau - r_j)$$

Proof of claim. At time t,

$$\nu_j(t)\sum_{a \preceq j} w_a g_a'(t-r_a) = \frac{G_j(t)^k - G_{j+1}(t)^k}{G(t)^k}G_j(t) \le k w_j g_j'(t-r_j)$$

where the inequality is due to the convexity of function z^k. Moreover, note that $\sum_{a \prec j} \nu_a \le 1$. Therefore,

$$\int_{r_j}^\tau \left(\nu_j(t)\sum_{a \preceq j} w_a g_a'(t-r_a) + w_j g_j'(t-r_j)\sum_{a \prec j} \nu_a(t) \right) dt$$
$$\le \int_{r_j}^\tau (k+1)w_j g_j'(t-r_j)dt = (k+1)w_j g_j(\tau - r_j)$$

The claim follows. $\qquad\square$

Claim 6. *For any time $\tau \ge r_j$,*

$$\frac{1}{p_j}\int_\tau^{C_j} \left(\nu_j(t)\sum_{a \preceq j} w_a g_a'(t-r_a) + w_j g_j'(t-r_j)\sum_{a \prec j} \nu_a(t) \right) dt \le \left(1 + \frac{1}{k}\right)G(t)$$

Proof of claim. We have

$$\frac{1}{p_j} \int_\tau^{C_j} \left(\nu_j(t) \sum_{a \preceq j} w_a g_a'(t - r_a) + w_j g_j'(t - r_j) \sum_{a \prec j} \nu_a(t) \right) dt$$

$$= \frac{1}{p_j} \int_\tau^{C_j} \nu_j(t) \left(\sum_{a \preceq j} w_a g_a'(t - r_a) + w_j g_j'(t - r_j) \frac{G_{j+1}(t)^k}{G_j(t)^k - G_{j+1}(t)^k} \right) dt$$

$$\leq \frac{1}{p_j} \int_\tau^{C_j} \nu_j(t) \left(\sum_{a \preceq j} w_a g_a'(t - r_a) + \frac{1}{k} G_{j+1}(t) \right) dt$$

$$\leq \frac{1}{p_j} \left(1 + \frac{1}{k} \right) G_j(\tau) \int_\tau^{C_j} \nu_j(t) dt = \frac{1}{p_j} \left(1 + \frac{1}{k} \right) G_j(\tau) p_j$$

$$= \left(1 + \frac{1}{k} \right) G_j(\tau) \leq \left(1 + \frac{1}{k} \right) G(\tau)$$

where the first inequality is due to the convexity of z^k and the second inequality follows from the fact that $G_j(t)$ is non-increasing in t. Hence, the claim follows. □

The proof of the lemma follows by the above two claims. □

Theorem 9. *The algorithm is $(1 + \epsilon)$-speed $\frac{4(1+\epsilon)^2}{\epsilon^2}$-competitive for integral JDGFP.*

Proof. With our choice of dual variables, the Lagrangian dual objective is

$$\min_{x,C} \sum_j \lambda_j p_j - \int_0^\infty \gamma(t) dt - \sum_j \int_{r_j}^{C_j} x_j(t) \left(\lambda_j - \gamma(t) - \delta_j g_j(C_j - r_j) \right) dt$$

$$\geq \min_{x,C} \sum_j \lambda_j p_j - \sum_j \int_{r_j}^{C_j} x_j(t) \left(\lambda_j - \delta_j g_j(t - r_j) \right) dt$$

$$\geq \min_{x,C} \sum_j \lambda_j p_j - \int_0^\infty \sum_j x_j(t) \frac{G(t)}{k} dt$$

The first inequality is due to $\gamma(t) = 0$ for every t and $t \leq C_j$ in the integral term corresponding to j so $g_j(t - r_j) \leq g_j(C_j - r_j)$ for every job j. The second inequality holds by Lemma 16.

In the resource augmentation model, the offline optimum has a machine of speed $\frac{1}{1+\epsilon}$, i.e., $\sum_j x_j(t) \leq \frac{1}{1+\epsilon}$ (while the algorithm has unit-speed machine). Recall that $k = \lceil 2/\epsilon \rceil$. Therefore, the Lagrangian dual is at least

$$\sum_j \lambda_j p_j - \frac{1}{1+\epsilon} \int_0^\infty \frac{G(t)}{k} dt = \frac{1}{k+1} \mathcal{F} - \frac{1}{1+\epsilon} \frac{1}{k} \mathcal{F} \geq \frac{\epsilon^2}{4(1+\epsilon)^2} \mathcal{F}$$

The algorithm has cost $\mathcal{F}$ so the theorem follows. □

7 Conclusion

In this work we applied primal-dual and dual-fitting techniques in the analysis of online algorithms for generalized flow-time scheduling problems. This approach yields proofs that

are derived from duality principles, unlike previous approaches that are predominantly based on potential functions. More importantly, we showed how to exploit duality in order to bypass a canonical rounding of fractional solutions that has been, up to now, a standard tool in the area of online scheduling. As a result, we obtained, at least for some objective functions, improved competitive ratios.

A promising direction for future work is to apply our framework to *non-clairvoyant* problems. It would be very interesting to obtain a primal-dual analysis of Shortest Elapsed Time First (SETF) which is is known to be scalable [20]; moreover, this algorithm has been analyzed in [10] in the context of the online GFP with convex/concave cost functions. We believe that one can use duality to argue that SETF is the non-clairvoyant counterpart of HDF; more precisely, we believe that one can derive SETF as a primal-dual algorithm in a similar manner as the discussion of HDF in Section 3. A more challenging task is to bound the primal and dual objectives, which appears to be substantially harder than in the clairvoyant setting. A further open question is extending the results of this paper to multiple machines; here, one potentially needs to define the dual variable $\gamma(t)$ with respect to as many curves per job as machines. Last, it would be very interesting to extend the framework in order to allow for algorithms that are not necessarily scalable. To this direction, one needs to further relax the conditions of the proposed framework, so as to exploit the speed augmentation and remedy the problematic situations in which primal job contributions correspond to a negative contribution in the dual.

Acknowledgements

References

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

[2] A. Antoniadis, N. Barcelo, M. Consuegra, P. Kling, M. Nugent, K. Pruhs, and M. Scquizzato. Efficient computation of optimal energy and fractional weighted flow trade-off schedules. In *STACS*, volume 25 of *LIPIcs*, pages 63–74, 2014.

[3] N. Bansal and H.-L. Chan. Weighted flow time does not admit $o(1)$-competitive algorithms. In *SODA*, pages 1238–1244, 2009.

[4] N. Bansal and K. Pruhs. Server scheduling in the weighted ℓ_p norm. In *LATIN*, volume 2976 of *LNCS*, pages 434–443, 2004.

[5] N. Bansal and K. Pruhs. The geometry of scheduling. In *FOCS*, pages 407–414, 2010.

[6] L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, and K. Pruhs. Online weighted flow time and deadline scheduling. *J. Discrete Algorithms*, 4(3):339–352, 2006.

[7] N. R. Devanur and Z. Huang. Primal dual gives almost optimal energy efficient online algorithms. In *SODA*, pages 1123–1140, 2014.

[8] J. Edmonds. Scheduling in the dark. *Theoretical Computer Science*, 235(1):109–141, 2000.

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

[10] K. Fox, S. Im, J. Kulkarni, and B. Moseley. Online non-clairvoyant scheduling to simultaneously minimize all convex functions. In *APPROX-RANDOM*, volume 8096 of *LNCS*, pages 142–157, 2013.

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

[12] A. Gupta, R. Krishnaswamy, and K. Pruhs. Nonclairvoyantly scheduling power-heterogeneous processors. In *Green Computing Conference*, pages 165–173, 2010.

[13] A. Gupta, R. Krishnaswamy, and K. Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *WAOA*, volume 7846 of *LNCS*, pages 173–186, 2012.

[14] W. Höhn, J. Mestre, and A. Wiese. How unsplittable-flow-covering helps scheduling with job-dependent cost functions. In *ICALP*, volume 8572 of *LNCS*, pages 625–636, 2014.

[15] S. Im, J. Kulkarni, and K. Munagala. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. In *STOC*, pages 313–322, 2014.

[16] S. Im, J. Kulkarni, K. Munagala, and K. Pruhs. Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. In *FOCS*, pages 531–540, 2014.

[17] S. Im, B. Moseley, and K. Pruhs. A tutorial on amortized local competitiveness in online scheduling. *SIGACT News*, 42(2):83–97, 2011.

[18] S. Im, B. Moseley, and K. Pruhs. Online scheduling with general cost functions. *SIAM Journal on Computing*, 43(1):126–143, 2014.

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

[20] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.

[21] R. Motwani, S. Phillips, and E. Torng. Non-clairvoyant scheduling. *Theoretical Computer Science*, 130(1):17–47, 1994.

[22] N. K. Thang. Lagrangian duality in online scheduling with resource augmentation and speed scaling. In *ESA*, volume 8125 of *LNCS*, pages 755–766, 2013.

Appendix

A LP-formulation of fractional JDGFP

We argue that the following LP is a relaxation of JDGFP.

$$\min \sum_{j \in \mathcal{J}} \delta_j \int_{r_j}^{\infty} g_j(t - r_j) x_j(t) dt$$

$$\int_{r_j}^{\infty} x_j(t) dt \geq p_j \qquad\qquad \forall j \in \mathcal{J}$$

$$\sum_{j \in \mathcal{J}} x_j(t) \leq 1 \qquad\qquad \forall t \geq 0$$

$$x_j(t) \geq 0 \qquad\qquad \forall j \in \mathcal{J}, t \geq 0$$

Consider a job $j \in \mathcal{J}$. By definition, the fractional weighted cost of a job $j \in \mathcal{J}$ is equal to

$$
\int_{r_j}^{\infty} w_j(t) g_j'(t - r_j) dt \;\;=\;\; \frac{w_j}{p_j} \int_{r_j}^{\infty} q_j(t) g_j'(t - r_j) dt
$$

$$
=\;\; \frac{w_j}{p_j} \int_{r_j}^{\infty} \left(g_j'(t - r_j) \int_{u=t}^{\infty} x_j(u) du \right) dt
$$

By changing the order of the integrals we get

$$
\int_{r_j}^{\infty} w_j(t) g_j'(t - r_j) dt \;\;=\;\; \frac{w_j}{p_j} \int_{u=r_j}^{\infty} \left(x_j(u) \int_{t=r_j}^{u} g_j'(t - r_j) dt \right) du
$$

$$
=\;\; \frac{w_j}{p_j} \int_{u=r_j}^{\infty} g_j(u - r_j) x_j(u) du
$$

$\square$