Lagrangian Duality in Online Scheduling with Resource Augmentation and Speed Scaling

Kim Thang Nguyen*

IBISC, University of Evry Val d'Essonne, France

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.

* Supported by the French National Agency (ANR) project COCA ANR-09-JCJC-0066-01, and the GdR RO.

H.L. Bodlaender and G.F. Italiano (Eds.): ESA 2013, LNCS 8125, pp. 755–766, 2013.

1 Introduction

We consider online scheduling problems where jobs arrive at unrelated 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 [22]. 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. [18] 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 [9]. 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 for mathematical programming, the competitive ratio follows.

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

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

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. 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 [14] and $\mathsf{LAPS}\epsilon$ is $\frac{1}{1-2\epsilon}$-speed, $\frac{2}{\epsilon^2}$-competitive [15]. 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 [5]. 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 [19] 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, related problems and direct generalizations of the above problems could be proved.

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 [13,17].

The strong lower bounds motivate the use of resource augmentation, originally introduced by Kalyanasundaram and Pruhs [22], 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 [26]. Edmonds [14] considered algorithm EQUI and showed that it was $(2+\epsilon)$-speed, $2/\epsilon$-competitive. Later on, Edmonds and Pruhs [15] 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 [6] 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. [10] 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. [5] 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 [11,25]. Chan et al. [12] 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 [2,24]. For the offline version on a single machine, Bansal and Pruhs [7] presented a polynomial time $O(\log \log P)$-approximation algorithm [7,8] where P is the ratio of the maximum to minimum job size. Im et al. [21] 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 principle approach to construct potential functions for online scheduling has been systematically formalized and given in [20] for many problems. However, it does not apply to all, for example the problem we consider in Section 5. More importantly, 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. [18] 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, [23] 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. We notice again that in our approach the programs do *not* need to be convex.

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

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

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

Remark that the weak duality holds also for functions instead of variables. In the setting, one could see the dual $\max_{\lambda,\gamma,\mu} \min_{x,C} L(x, C, \lambda, \gamma, \mu)$ as an optimization problem over functions $x_j(t)$ and others (calculus of variations); or as an optimization over variables (x, t) and others (by a transformation $x_j(t) \mapsto (x_j, t)$).

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 $q_1 + \ldots + q_{j-1} + (n-j)q_j$ 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 ([14]). *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 [15].

Algorithm LAPS$_\beta$ 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.

Theorem 2 ([15]). *Algorithm LAPS$_\epsilon$ is $\frac{1}{1-2\epsilon}$-speed, $\frac{2}{\epsilon^2}$-competitive for the problem of minimizing total flow time.*

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.

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

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.

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

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 $z f'(z) \leq K_4 f(z)$ $\forall z \geq 0$.

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

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

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.

Theorem 5. *The algorithm is* $\frac{1}{1-3\epsilon}$*-speed and* $\frac{2K(\epsilon)}{\epsilon}$*-competitive where* $K(\epsilon) = \max\{K_1(\epsilon), 3K_2(\epsilon)K_3(\epsilon)K_4\}$.

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.*

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 [4,25] or the capacitated machine model [16].

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.

Acknowledgment. We would like to thank Kirk Pruhs and anonymous reviewers for pointing out related references and useful comments.

References

1. Anand, S., Garg, N., Kumar, A.: Resource augmentation for weighted flow-time explained by dual fitting. In: Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms, pp. 1228–1241 (2012)
2. Azar, Y., Epstein, L., Richter, Y., Woeginger, G.J.: All-norm approximation algorithms. J. Algorithms 52(2), 120–133 (2004)
3. Bansal, N., Chan, H.-L.: Weighted flow time does not admit $o(1)$-competitive algorithms. In: Proc. 20th ACM-SIAM Symposium on Discrete Algorithms, pp. 1238–1244 (2009)
4. Bansal, N., Chan, H.-L., Lam, T.-W., Lee, L.-K.: Scheduling for speed bounded processors. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 409–420. Springer, Heidelberg (2008)
5. Bansal, N., Chan, H.-L., Pruhs, K.: Speed scaling with an arbitrary power function. In: Proc. 20th ACM-SIAM Symposium on Discrete Algorithms, pp. 693–701 (2009)
6. Bansal, N., Pruhs, K.R.: Server scheduling in the weighted ℓ_p norm. In: Farach-Colton, M. (ed.) LATIN 2004. LNCS, vol. 2976, pp. 434–443. Springer, Heidelberg (2004)
7. Bansal, N., Pruhs, K.: The geometry of scheduling. In: Proc. 51th Symposium on Foundations of Computer Science, pp. 407–414 (2010)

8. Bansal, N., Pruhs, K.: Weighted geometric set multi-cover via quasi-uniform sampling. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 145–156. Springer, Heidelberg (2012)
9. Buchbinder, N., Naor, J.: The design of competitive online algorithms via a primal-dual approach. Foundations and Trends in Theoretical Computer Science 3(2-3), 93–263 (2009)
10. Chadha, J.S., Garg, N., Kumar, A., Muralidhara, V.N.: A competitive algorithm for minimizing weighted flow time on unrelatedmachines with speed augmentation. In: Proc. 41st ACM Symposium on Theory of Computing, pp. 679–684 (2009)
11. Chan, H.-L., Edmonds, J., Lam, T.W., Lee, L.-K., Marchetti-Spaccamela, A., Pruhs, K.: Nonclairvoyant speed scaling for flow and energy. Algorithmica 61(3), 507–517 (2011)
12. Chan, S.-H., Lam, T.W., Lee, L.-K., Ting, H.-F., Zhang, P.: Non-clairvoyant scheduling for weighted flow time and energy on speed bounded processors. Chicago J. Theor. Comput. Sci. (2011)
13. Chekuri, C., Khanna, S., Zhu, A.: Algorithms for minimizing weighted flow time. In: Proc. 33rd ACM Symposium on Theory of Computing, pp. 84–93 (2001)
14. Edmonds, J.: Scheduling in the dark. Theor. Comput. Sci. 235(1), 109–141 (2000)
15. Edmonds, J., Pruhs, K.: Scalably scheduling processes with arbitrary speedup curves. ACM Transactions on Algorithms 8(3), 28 (2012)
16. Fox, K., Korupolu, M.: Weighted flowtime on capacitated machines. In: Proc. 24th ACM-SIAM Symposium on Discrete Algorithms, pp. 129–143 (2013)
17. Garg, N., Kumar, A.: Minimizing average flow-time: Upper and lower bounds. In: Proc. 48th Symposium on Foundations of Computer Science, pp. 603–613 (2007)
18. Gupta, A., Krishnaswamy, R., Pruhs, K.: Online primal-dual for non-linear optimization with applications to speed scaling. In: Proc. 10th Workshop on Approximation and Online Algorithms, pp. 173–186 (2012)
19. Im, S.: Online Scheduling Algorithms for Average Flow Time and its Variants. PhD thesis, University of Illinois at Urbana-Champaign (2012)
20. Im, S., Moseley, B., Pruhs, K.: A tutorial on amortized local competitiveness in online scheduling. SIGACT News 42(2), 83–97 (2011)
21. Im, S., Moseley, B., Pruhs, K.: Online scheduling with general cost functions. In: Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms, pp. 1254–1265 (2012)
22. Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. J. ACM 47(4), 617–643 (2000)
23. Kling, P., Pietrzyk, P.: Profitable scheduling on multiple speed-scalable processors. In: Proc. 25th Symposium on Parallelism in Algorithms and Architectures (2013)
24. Kumar, V.S.A., Marathe, M.V., Parthasarathy, S., Srinivasan, A.: A unified approach to scheduling on unrelated parallel machines. J. ACM 56(5) (2009)
25. Lam, T.W., Lee, L.-K., To, I.K., Wong, P.W.H.: Online speed scaling based on active job count to minimize flow plus energy. Algorithmica 65(3), 605–633 (2013)
26. Motwani, R., Phillips, S., Torng, E.: Non-clairvoyant scheduling. Theor. Comput. Sci. 130(1), 17–47 (1994)