Online Primal-Dual for Non-linear Optimization with Applications to Speed Scaling

Anupam Gupta[1,⋆], Ravishankar Krishnaswamy[2], and Kirk Pruhs[3,⋆⋆]

[1] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213
[2] Computer Science Department, Princeton University, Princeton, NJ 08542
[3] Computer Science Department, University of Pittsburgh, Pittsburgh, PA 15260

Abstract. We give a principled method to design online algorithms (for potentially non-linear problems) using a mathematical programming formulation of the problem, and also to analyze the competitiveness of the resulting algorithm using the dual program. This method can be viewed as an extension of the online primal-dual method for linear programming problems, to nonlinear programs. We show the application of this method to two online speed-scaling problems: one involving scheduling jobs on a speed scalable processor so as to minimize energy plus an arbitrary sum scheduling objective, and one involving routing virtual circuit connection requests in a network of speed scalable routers so as to minimize the aggregate power or energy used by the routers. This analysis shows that competitive algorithms exist for problems that had resisted analysis using the dominant potential function approach in the speed-scaling literature, and provides alternate cleaner analysis for other known results. This gives us another tool in the design and analysis of primal-dual algorithms for online problems.

1 Introduction

Speed scalable devices are now a ubiquitous energy management technology. Such devices can be run in high speed and power modes that are energy inefficient, or in low speed and power modes that are more energy efficient, where energy efficiency is the resulting processing speed divided by power investment. The resulting optimization problems involve determining when this improvement in the quality of service provided by running at high speed justifies the resulting inefficient use of significant energy. As the relationship between speed and power in current (and any conceivable) technologies is non-linear, so are the resulting optimization problems. This non-linearity explains in part why we have generally not been able to show that online algorithms are competitive in such settings by reasoning directly about optimal solutions. The dominant algorithm analysis tool in speed-scaling settings has been potential functions. But one has to often guess the "right" potential function; moreover, there are situations where the use of potential functions is problematic, most notably when there does not seem to be a simple algebraic expression for the "right" potential for an arbitrary configuration.

⋆ Research partly supported by NSF awards CCF-0964474 and CCF-1016799.
⋆⋆ Supported in part by NSF grants CCF-0830558, CCF-1115575, CNS-1253218 and an IBM Faculty Award.

T. Erlebach and G. Persiano (Eds.): WAOA 2012, LNCS 7846, pp. 173–186, 2013.

One motivation of our research is to see if one can use duality to analyze online algorithms for speed-scaling problems for which algorithm analysis using potential functions seems problematic.

The first problem we consider involves scheduling jobs on a speed scalable processor online, with an objective of the form $\mathcal{E} + \beta\mathcal{S}$, where $\mathcal{E}$ is the energy used by the processor, $\mathcal{S}$ is a scheduling objective that is the sum over jobs of a scheduling cost of the individual jobs, and β expresses the relative value of saving energy versus decreasing the scheduling objective. The input consists of a collection of jobs that arrive over time. The j^{th} job arrives at time r_j, and has size/work p_j. There is a convex function $P(s) = s^\alpha$ specifying the dynamic power used by the processor as a function of speed s, which may be any nonnegative real number. The value of α is typically around 3 for CMOS based processors. A *fractional sum scheduling objective* $\mathcal{S}$ is of the form $\sum_j \sum_t \frac{y_{jt}}{p_j} C_{jt}$, where C_{jt} is the cost of completing a unit of work of job j at time t, and y_{jt} is the amount of work completed at time t (such that $\sum_t y_{jt} = p_j$). The corresponding *integer* sum scheduling objective is of the form $\sum_j \sum_t z_{jt} C_{jt}$, where z_{jt} indicates if job j was finished at the time instant t. For instance, a single job arrives at time r_j with $C_{jt} = (t + 1 - r_j)$, and is executed over times $t \in \{r_j, r_j + 1, r_j + 2, \cdots, r_j + p_j - 1\}$ at a uniform rate $y_{jt} = 1$, the fractional sum objective is $\sum_{t=r_j}^{r_j+p_j-1} \frac{1}{p_j}(t - r_j + 1) = \frac{p_j+1}{2}$, whereas the integer sum objective is p_j (since $z_{j(r_j+p_j-1)} = 1$ and all other $z_{jt} = 0$). The energy cost $\mathcal{E}$ in both cases is the non-linear expression $\sum_t (\sum_j y_{jt})^2$.

For linear sum scheduling objectives (where the cost C_{jt} for finishing a job j at time t is a increasing linear function of the flow time $t - r_j$, such as the one above), a potential function based on an algebraic expression for the future cost of the online scheduling algorithm (starting from a particular configuration) has proved to be widely applicable for analyzing natural speed-scaling scheduling algorithms [1]. However, the seeming lack of simple algebraic expressions for future online costs of natural online algorithms for nonlinear sum scheduling objectives (e.g., $C_{jt} = (t - r_j + 1)^2$, which gives the sum of the squares of flow time) explains in part why, despite some effort, we have not been able to analyze algorithms for such problems. It is easy to convert a solution for the fractional scheduling objective to the integer objective while losing at most $\max\{(1 + \epsilon)^\alpha, 1/\epsilon\}$ by speeding up each job by $1 + \epsilon$, so that the fractional objective pays at least ϵ for each instant the sped-up job has not yet been completed, whereas the power cost is increased only by $(1 + \epsilon)^\alpha$[2,3], so we will focus on the fractional scheduling objectives from now on.

The second problem that we consider involves online routing of virtual circuit connection requests in a network of speed scalable routers with the objective of minimizing the aggregate power used by the routers. The j^{th} request consists of a source s_j, a sink t_j, and a flow requirement f_j. In the unsplittable flow version of the problem the online algorithm must route f_j units of flow along a single (s_j, t_j)-path. In the splittable flow version of this problem, the online algorithm may partition the f_j units of flow among some collection of (s_j, t_j)-paths. In either case, we assume speed scalable network elements (routers, or links, or both), where element e use powers ℓ_e^α, where the load ℓ_e is the sum of the flows through the element. The objective of total aggregate power is then $\sum_e \ell_e^\alpha$. This problem was introduced in [4], where a poly-log-approximate *offline*

polynomial-time algorithm is given for unsplittable routing; this algorithm classifies the requests by geometrically increasing demands, and randomly rounds a convex program for each demand class.

2 Our Contributions

Very much in the spirit of the online primal dual technique for linear programs [5], we give a principled method both to design online algorithms (but for potentially nonlinear problems) using a mathematical programming formulation of the problem, and also to analyze the competitiveness of the resulting algorithm using the Lagrangian dual program. We start by considering a mathematical program for the offline problem. We then interpret the online algorithm as solving this mathematical program online, where the constraints arrive one-by-one: in response to the arrival of a new constraint, the online algorithm has to raise some of the primal variables so that the new constraint will be satisfied. We consider the most natural online greedy algorithm: one that raises the primal variables so that the increase in the primal objective is minimized.

For the analysis, we use weak duality: each feasible value of the dual is a lower bound to the optimal primal solution [6]. How should we set the duals? We set the Lagrangian dual variable corresponding to the new constraint to be *proportional to the rate of increase in the primal objective* that the online algorithm incurred at the time that the constraint was satisfied. If we could argue that the value of the dual increased by at least a constant fraction of the increase to the primal, we would be done. But what is the value of the resulting dual? Due to nonlinearity, analyzing the dual for a nonlinear program is more complicated than for a linear program, since in the dual for a nonlinear program, one can not disentangle the objective and the constraints (as one can in the linear case); the dual itself contains a version of the primal objective, and hence copies of the primal variables, within it. Consequently, the arguments for the dual in the nonlinear case not only involve setting the Lagrangian dual variables, but also relating the settings of the copies of the primal variables in the dual with the actual primal variables in the primal. The solutions we find are fractional solutions to the nonlinear program, so for the speed-scaling scheduling (with the fractional objective) and splittable routing problems that we consider, this analysis allows us to conclude that the natural online greedy algorithm is $O_\alpha(1)$-competitive (the subscript means that the constant hidden in the big-O depends on the constant α). As mentioned above, the integer scheduling objective follows from the fractional one at a small loss [2,3]. For the unsplittable routing problem, we can also show that the natural online greedy algorithm is $O_\alpha(1)$-competitive.

Before we give more details about the specific speed-scaling problems that we consider, we would like to emphasize that once we formulate the primal non-linear program in the obvious way, the design of the online algorithm and the variable settings for its Lagrangian dual are naturally derived from this program. The problem-specific aspects are confined to setting the dual variables—which in both our problems is some multiplier δ times the rate of primal change—and the analysis of the dual (which gives the

"right" value of the multiplier δ we should set).[1] Consequently, we feel that our work represents another step towards a principled design and analysis of primal-dual algorithms for online problems. We hope that this principled approach could be applied to a wider class of nonlinear online problems.

2.1 Applications to Speed-Scaling Scheduling and Routing

Our first observation is that fractional speed-scaling scheduling problems with a general sum scheduling objective can be cast more conveniently as the following *Online Generalized Assignment Problem (OnGAP)*. In OnGAP, jobs arrive online one-by-one, and the algorithm must fractionally assign these jobs to one of m machines. When a job j arrives, the online algorithm learns ℓ_{je}, the amount by which the load of machine e would increase for *each unit* of work of job j that is assigned to machine e, and c_{je}, the assignment cost incurred for each unit of work of job j that is assigned to machine e. So if x_{je} is the fraction of job j assigned to machine e, then the assignment cost is $c_{je}x_{je}$, and the load of the machine e increases by $\ell_{je}x_{je}$. The goal is to minimize the sum of the α^{th} powers of the machine loads, plus the total assignment cost. (See (4.1) for the convex programming formulation.)

To cast a speed-scaling problem with a (fractional) sum scheduling objective as OnGAP, think of each unit of time as being a separate machine. The assignment cost c_{je} then models the cost for scheduling a unit of job j at time e. Let us now illustrate this model using some examples:

Speed-Scaling with Deadline Feasibility. In this problem, each job j has a size of ℓ_j and a deadline of d_j. The goal is to devise speed-scaling and scheduling policies so that every job is scheduled within its deadline, and the total energy is minimized. Indeed, we can view this as OnGAP where each time unit is a machine, $\ell_{jt} := \ell_j$ for all times, and $c_{jt} := 0$ for $t \in [r_j, d_j]$ and is infinite otherwise. This problem has been widely studied [7,8,9,10], and different algorithms are shown to be $O_\alpha(1)$-competitive using varied potential functions.

Total Flow plus Energy. Here, given a set of jobs and a speed-scalable processor, the objective is to minimize the sum of (fractional) flowtimes plus energy of the schedule. We can cast this as OnGAP by setting c_{jt} to be $(t - r_j)$ for $t \geq r_j$ and infinite otherwise. This problem has been studied in [11,3,12,13,14,15,16], where different algorithms are shown to be $O_\alpha(1)$-competitive (and some, even $O(1)$-competitive for general power functions) using potential functions.

Total Flow Squared plus Energy. For the objective of sum of fractional flow/response times squared plus energy, we can set the assignment cost c_{je} to be $(e - r_j)^2$ for all times $e \geq r_j$, the release time of job j, and infinite otherwise.

More General Objectives. We can create much more exotic objectives: say, $c_{je} = (e - r_j)^2$ for $j \in [r_j, d_j]$ and ∞ otherwise would give squared flow time with a hard deadline, or we could create blackout dates by setting some c_{je}'s to ∞. For many of these general problems, we provide the first online algorithms with non-trivial competitive ratios.

[1] In the sequel, whenever we use the word dual, we refer to the Lagrangian dual of the primal convex program being considered.

In Section 4 we apply our primal-dual approach to solving OnGAP *fractionally*, and show that a natural greedy algorithm is α^α-competitive.[2] This immediately gives us solutions for speed scaling scheduling problems with fractional sum objectives. To analyze our algorithm for OnGAP, we show a dual solution has a particularly nice form, and for our setting of the dual variables, is within α^α of the primal solution. As an immediate consequence, the corresponding online greedy algorithm is α^α-competitive for speed-scaling scheduling problems with *any* fractional sum scheduling objective. Recall that previously, competitive analyses were known only for linear sum scheduling objectives, so this is a substantial improvement. For some speed-scaling problems, our duality analysis is cleaner than the existing potential function analyses; e.g., compare the α^α-competitive analysis of the greedy Optimal Available (OA) algorithm for energy minimization with deadline feasibility constraints given in [8] to our α^α-competitive analysis of our greedy algorithm. Lower bounds in [17,18,8] imply that no deterministic online algorithm can be better than α^α-competitive for OnGAP, and no deterministic online algorithm can be better than α^α-competitive for speed scaling scheduling to minimize energy with feasibility constraints. In Section 5, we make some further comments about the application of these results to speed-scaling problems.

Finally in Section 6, we apply our primal-dual approach to the splittable routing problem in a network of speed scalable routers. In this case, the worst-case settings of the copies of the primal variables in the dual are not easy to reason about. To facilitate this reasoning, we *relax* the dual problem in a novel way, by allowing the copies of the primal variables in the dual to take on different values for different edges. To overcome relaxing the flow-constraints, we alter the relaxed objective function (based on the edge loads of our online algorithm!) to ensure that we can still recover enough dual value. This allows us to show the online greedy algorithm is α^α-competitive with respect to the relaxed dual with the specified settings of the dual variables, and hence with respect to optimal. This extends to unsplittable routings as well.

3 Related Work

An extensive survey/tutorial on the online primal dual technique for linear problems can be found in [5]. A survey of the algorithmic power management literature in general, and the speed-scaling literature in particular, can be found in [19]. Casting the speed-scaling scheduling problems as a load balancing problem is natural in hindsight, but to the best of our knowledge this has not been observed before. This reduction allows the application of techniques from the load balancing literature to speed-scaling problems. The version of OnGAP without assignment costs was studied by [17,18], where the online greedy algorithm is shown to be $O_\alpha(1)$-competitive. In their analysis the online cost is bounded by an algebraic expression involving the product of the online cost and the optimal cost, which is disentangled by use of the Cauchy-Schwartz inequality. While this analysis shares some commonalities with both potential function analysis

[2] In the full version, we show a related greedy algorithm is $O(\alpha)^\alpha$-competitive for OnGAP *integrally* (where each job has to go to a *single* machine). This does not imply anything useful for the speed-scaling scheduling problems. On the other hand, one can use the underlying ideas to convert the splittable energy-aware routing algorithm of Section 6 to unsplittable routings.

and duality analysis, it is probably best considered a distinct technique. Upon some reflection, one can see that their potential function technique can be used to obtain an alternate analysis that achieves the same bounds as we achieve in this paper by duality. Caragiannis [20] gives some refinements to the analysis in [17,18] for OnGAP without assignment costs. An offline $O(1)$-approximation (independent of α) was given by [21,22], via solving the convex program and rounding the solution in a correlated fashion: such an result independent of α is impossible in the online setting. Finally, offline poly-log-approximation algorithms for the virtual circuit routing problem, when routers have a static power component, can be found in [23,4].

Works [24,25] show that various online algorithms are competitive, using potential function analysis, for various scheduling problems with fixed speed processors and for the ℓ_k norms of flow objective. The potential functions used in these analyses were motivated by the desire to have an algebraic expression for the future costs for the online algorithm, but required some ad-hoc features in order for the algebra to work out. Despite efforts by the authors of these papers, it is not clear how to extend these potential functions to work in a speed-scaling setting.

Independently and concurrently with this work, Anand, Garg and Kumar [26] obtained results in a similar vein to the results here. Mostly notably, they showed how to use nonlinear-duality to analyze a greedy algorithm for a multiprocessor speed-scaling problem involving minimizing flow plus energy on unrelated machines. Additionally, [26] showed how duality based analyses could be given for several scheduling algorithms that were analyzed in the literature using potential functions. However, our results are somewhat different in spirit, with our emphasis being more on a principled methodology for algorithm design and setting of the dual variables. For instance the algorithm for the speed-scaling problem in [26] is not derived from the mathematical programming formulation, and the emphasis is more on obtaining a "dual-fitting" analysis for (in some sense) pre-existing algorithms.

4 The Online Generalized Assignment Problem

In this section we consider the problem of Online Generalized Assignment Problem (OnGAP). If x_{je} denotes the extent to which job j is assigned on machine e, then this problem can be expressed by the following mathematical program:

$$\begin{aligned} \min \quad & \sum_e \left(\sum_j \ell_{je} x_{je} \right)^\alpha + \sum_e \sum_j c_{je} x_{je} \\ \text{subject to} \quad & \sum_e x_{je} \geq 1 \qquad j = 1, \ldots, n \end{aligned} \tag{4.1}$$

The dual of the primal relaxation is then

$$g(\lambda) = \min_{x \geq 0} \left(\sum_j \lambda_j + \sum_e \left(\sum_j \ell_{je} x_{je} \right)^\alpha + \sum_{j,e} c_{je} x_{je} - \sum_{j,e} \lambda_j \, x_{je} \right) \tag{4.2}$$

One can think of the dual problem as having the same instance as the primal, but where jobs are allowed to be assigned to extents less than unit. This is compensated for in the

objective function: in addition to the load cost $\sum_e \left(\sum_j \ell_{je} x_{je}\right)^\alpha$ as in the primal, a fixed cost of λ_j is paid for each job j, and a *payment* (or negative cost) of $\lambda_j - c_{je}$ is obtained for each unit of job j assigned. It is well known that each feasible value of the dual is a lower bound to the optimal primal solution; this is *weak duality* [6].

Online Greedy Algorithm Description: Let δ be a constant that we will later set to $\frac{1}{\alpha^{\alpha-1}}$. Now the algorithm works as follows when a new job j arrives: until a unit fraction is scheduled, job j is scheduled on all machines for which the increase in the cost will be the least, assuming that energy costs are discounted by a factor of δ. More formally, the value of all the primal variables x_{je} for all the machines e that minimize

$$\delta \cdot \alpha \cdot \ell_{je} (\sum_{i \le j} \ell_{ie} x_{ie})^{\alpha-1} + c_{je} \tag{4.3}$$

are increased until all the work from job j is scheduled, i.e., $\sum_e x_{je} = 1$. Notice that $\alpha \cdot \ell_{je} \left(\sum_{i \le j} \ell_{ie} x_{ie}\right)^{\alpha-1}$ is the rate at which the load cost is increasing for machine e, and c_{je} is the rate that assignment costs are increasing for machine e. In other words, our algorithm fractionally assigns the job to the machines on which the overall objective increases at the least rate. Furthermore, observe that if the algorithm begins assigning the job to some machine e, it does not stop raising the primal variable x_{je} until the job is fully assigned[3]. By this monotonicity property, it is clear that all machines e for which $x_{je} > 0$ have the same value of the above derivative when j is fully assigned. Now, for the purpose of analysis, we set the value $\widehat{\lambda}_j$ to be the rate of increase of the objective value when we assigned the last infinitesimal portion of job j. More formally, if e is any machine on which job j is run, i.e., if $x_{je} > 0$, then

$$\widehat{\lambda}_j := \delta \cdot \alpha \cdot \ell_{je} \left(\sum_{i \le j} \ell_{ie} x_{ie} \right)^{\alpha-1} + c_{je} \tag{4.4}$$

Intuitively, $\widehat{\lambda}_j$ is a surrogate for the total increase in objective value due to our fractional assignment of job j (we assign a total of 1 unit of job j, and λ_j is set to be the rate at which objective value increases). Let $\widetilde{x}$ denote the final value of the x_{je} variables for the online algorithm.

Algorithm Analysis. To establish the desired competitive ratio of $O(\alpha^\alpha)$, note that it is sufficient (by weak duality) to show that $g(\widehat{\lambda})$ is at least $\frac{1}{\alpha^\alpha}$ times the cost of the online solution. To this end, let $\widehat{x}$ be the value of the minimizing x variables in $g(\widehat{\lambda})$, namely

$$\widehat{x} = \arg\min_{x \succeq 0} \left(\sum_j \widehat{\lambda}_j + \sum_e \left(\sum_j \ell_{je} x_{je} \right)^\alpha - \sum_{j,e} \left(\widehat{\lambda}_j - c_{je} \right) x_{je} \right)$$

Observe that the values $\widehat{x}$ could be very different from the values $\widetilde{x}$, and indeed the next few lemmas try to characterize these values. Lemma 1 notes that $\widehat{x}$ only has one job $\varphi(e)$ on each machine e, and Lemma 2 shows how to determine $\varphi(e)$ and $\widehat{x}_{\varphi(e)e}$. Then, in Lemma 3, we show that a feasible choice for the job $\varphi(e)$ is the latest arriving job for which the online algorithm scheduled some bit of work on machine e; Let us denote this latest job by $\psi(e)$. Formally, $\psi(e) = \max\{j \text{ s.t } \widetilde{x}_{je} > 0\}$.

[3] It may however increase x_{je} and $x_{je'}$ at different rates so as to balance the derivatives where e and e' are both machines which minimize equation 4.3

Lemma 1. *There is a minimizing solution $\widehat{x}$ such that if $\widehat{x}_{je} > 0$, then $\widehat{x}_{ie} = 0$ for $i \neq j$.*

Proof. Suppose for some machine e, there exist distinct jobs i and k such that $\widehat{x}_{ie} > 0$ and $\widehat{x}_{ke} > 0$. Then by the usual argument of either increasing or decreasing these variables along the line that keeps their sum constant, we can keep the convex term $(\sum_j \ell_{je}\widehat{x}_{je})^\alpha$ term fixed and not increase the linear term $\sum_j (\widehat{\lambda}_j - c_{je})\,\widehat{x}_{je}$. This allows us to either set $\widehat{x}_{ie}$ or $\widehat{x}_{ke}$ to zero without increasing the objective. ∎

Lemma 2. *Define $\varphi(e) = \arg\max_j \frac{(\widehat{\lambda}_j - c_{je})}{\ell_{je}}$. Then $\widehat{x}_{\varphi(e)e} = \frac{1}{\ell_{\varphi(e)e}} \left(\frac{\widehat{\lambda}_{\varphi(e)} - c_{\varphi(e)e}}{\alpha \ell_{\varphi(e)e}} \right)^{1/(\alpha-1)}$ and $\widehat{x}_{je} = 0$ for $j \neq \varphi(e)$. Moreover, the contribution of machine e towards $g(\widehat{\lambda})$ is exactly $(1-\alpha)\left(\frac{\widehat{\lambda}_{\varphi(e)} - c_{\varphi(e)e}}{\alpha \ell_{\varphi(e)e}} \right)^{\alpha/(\alpha-1)}$.*

Proof. By Lemma 1 we know that in $\widehat{x}$ there is at most one job (say j, if any) run on machine e. Then the contribution of this machine to the value of $g(\widehat{\lambda})$ is

$$(\ell_{je}\widehat{x}_{je})^\alpha - (\widehat{\lambda}_j - c_{je})\widehat{x}_{je} \tag{4.5}$$

Since $\widehat{x}$ is a minimizer for $g(\widehat{\lambda})$, we know that the partial derivative of the above term evaluates to zero. This gives $\alpha \ell_{je} \cdot (\ell_{je}\widehat{x}_{je})^{\alpha-1} - \left(\widehat{\lambda}_j - c_{je}\right) = 0$, or equivalently, $\widehat{x}_{je} = \frac{1}{\ell_{je}} \left(\frac{\widehat{\lambda}_j - c_{je}}{\alpha \ell_{je}} \right)^{1/(\alpha-1)}$. Substituting into this value of $\widehat{x}_{je}$ into equation (4.5), the contribution of machine e towards the dual $g(\widehat{\lambda})$ is

$$\left(\frac{\widehat{\lambda}_j - c_{je}}{\alpha \ell_{je}} \right)^{\alpha/(\alpha-1)} - \frac{(\widehat{\lambda}_j - c_{je})}{\ell_{je}} \left(\frac{\widehat{\lambda}_j - c_{je}}{\alpha \ell_{je}} \right)^{1/(\alpha-1)} = (1-\alpha) \left(\frac{\widehat{\lambda}_j - c_{je}}{\alpha \ell_{je}} \right)^{\alpha/(\alpha-1)}$$

Hence, for each machine e, we want to choose that the job j that minimizes this expression, which is also the job j that maximizes the expression $(\widehat{\lambda}_j - c_{je})/\ell_{je}$ since $\alpha > 1$. This is precisely the job $\varphi(e)$ and the proof is hence complete. ∎

Lemma 3. *For all machines e, job $\psi(e)$ is a feasible choice for $\varphi(e)$.*

Proof. The line of reasoning is the following:

$$\varphi(e) = \arg\max_j \frac{(\widehat{\lambda}_j - c_{je})}{\ell_{je}} = \arg\max_j \left(\delta \cdot \alpha \cdot \left(\sum_{i \leq j} \ell_{je} x_{ie} \right)^{\alpha-1} \right)$$
$$= \arg\max_j \left(\left(\sum_{i \leq j} \ell_{ie} x_{ie} \right)^{\alpha-1} \right) = \psi(e)\,.$$

The first equality is the definition of $\varphi(e)$. For the second, observe that for any job k,

$$\widehat{\lambda}_k \leq \delta \cdot \alpha \cdot \ell_{ke} (\sum_{i \leq k} \ell_{ie} x_{ie})^{\alpha-1} + c_{ke} \implies \frac{\widehat{\lambda}_k - c_{ke}}{\ell_{ke}} \leq \delta\, \alpha\, (\sum_{i \leq k} \ell_{ie} x_{ie})^{\alpha-1}\,.$$

The above expression is monotone increasing in $\sum_{i \leq k} \ell_{ie} x_{ie}$, the load due to jobs up to and including k. Moreover, it is maximized by the last job assigned fractionally to e. Since the last job is $\psi(e)$, the last equality follows. ∎

Theorem 1. *The online greedy algorithm is α^α-competitive.*

Proof. By weak duality it is sufficient to show that $g(\widehat{\lambda}) \geq \text{ON}/\alpha^\alpha$. Applying Lemma 2 to the expression for $g(\widehat{\lambda})$ (equation (4.2)), and using Lemma 3 to replace $\varphi(e)$ by $\psi(e)$, we get that

$$g(\widehat{\lambda}) = \left(\sum_j \widehat{\lambda}_j + \sum_e (1-\alpha) \left(\frac{\widehat{\lambda}_{\psi(e)} - c_{\psi(e)e}}{\alpha \ell_{\psi(e)e}} \right)^{\alpha/(\alpha-1)} \right) \tag{4.6}$$

Now we consider only the first term $\sum_j \widehat{\lambda}_j$ and evaluate it.

$$\sum_j \widehat{\lambda}_j = \sum_{j,e} \widehat{\lambda}_j \widetilde{x}_{je} \tag{4.7}$$

$$= \sum_e \sum_j \left(\delta \cdot \alpha \cdot \ell_{je} \left(\sum_{i \leq j} \ell_{ie} \widetilde{x}_{ie} \right)^{\alpha-1} + c_{je} \right) \widetilde{x}_{je} \tag{4.8}$$

$$= (\delta \cdot \alpha) \sum_e \sum_j \ell_{je} \widetilde{x}_{je} \left(\sum_{i \leq j} \ell_{ie} \widetilde{x}_{ie} \right)^{\alpha-1} + \sum_{j,e} \widetilde{x}_{je} c_{je} \tag{4.9}$$

$$\geq \delta \sum_e \left(\sum_j \ell_{je} \widetilde{x}_{je} \right)^\alpha + \sum_{j,e} \widetilde{x}_{je} c_{je} \tag{4.10}$$

If we consider the second term of (4.6), and plug in the value of $\widehat{\lambda}_{\psi(e)}$, it evaluates to $(1-\alpha)\delta^{\alpha/(\alpha-1)} \sum_e \left(\sum_j \ell_{je} \widetilde{x}_{je} \right)^\alpha$. Putting the above two estimates together, we get

$$g(\widehat{\lambda}) \geq \delta \sum_e \left(\sum_j \ell_{je} \widetilde{x}_{je} \right)^\alpha + \sum_{j,e} \widetilde{x}_{je} c_{je} + (1-\alpha)\delta^{\alpha/(\alpha-1)} \sum_e \left(\sum_j \ell_{je} \widetilde{x}_{je} \right)^\alpha \tag{4.11}$$

$$= \left(\delta + (1-\alpha)\delta^{\alpha/(\alpha-1)} \right) \sum_e \left(\sum_j \widetilde{x}_{je} \ell_{je} \right)^\alpha + \sum_{j,e} \widetilde{x}_{je} c_{je} \geq \text{ON}/\alpha^\alpha \tag{4.12}$$

The last step is by setting $\delta = 1/\alpha^{\alpha-1}$ which maximizes $\left(\delta + (1-\alpha)\delta^{\alpha/(\alpha-1)} \right)$. ∎

As observed, e.g., in [18], an $O(\alpha)^\alpha$ result is the best possible, even for the (fractional) OnGAP problem without any assignment costs. In the full version, we show how to obtain an $O(\alpha)^\alpha$-competitive algorithm for *integer* solutions to OnGAP by a similar greedy algorithm, and a similar but slightly more careful analysis.

5 Application to Speed Scaling

We now discuss the application of our results for OnGAP to some well-studied speed-scaling problems. Normally one thinks of the online scheduling algorithm as having two components: a *job selection policy* to determine the job to run, and a *speed-scaling policy* to determine the processor speed. However, one gets a different view when one thinks of the online scheduler as solving online the following mathematical programming formulation of the problem (which is an instance of the fractional OnGAP problem):

$$\min \quad \sum_t \left(\sum_j p_j x_{jt} \right)^\alpha + \sum_j \sum_t C_{jt}\, x_{jt}$$

$$\text{subject to} \quad \sum_t x_{jt} \geq 1 \qquad j = 1, \ldots, n$$

Here the variables x_{jt} specify how much work from job j is run at time t, and the objective captures the fractional sum scheduling objective defined in the Introduction. (Think of $y_{jt} := x_{jt} \cdot p_j$.) The arrival of a job j corresponds to the arrival of a constraint specifying that job j must be completed. Greedily raising the primal variables corresponds to committing to complete the work of job j in the cheapest possible way, given the previous commitments. This greedy algorithm has the advantage that, at the release time of a job, it can commit to the client exactly the times that each portion of the job will be run. One can certainly imagine situations when this information would be useful to the client. The speed-scaling algorithms analyzed in the literature for total (possibly weighted) flow scheduling objectives, are some variation of the *balancing* speed-scaling algorithm that sets the power equal to the (fractional) number/weight of unfinished jobs; so for these prior algorithms, when a job is run generally depends on jobs that arrive in the future.

As mentioned earlier, this algorithm for the *fractional* sum objective can be converted to the *integer* scheduling objective by speeding up the processor by a $(1+\epsilon)$ factor and using known techniques [2,3]: the eventual competitive ratio is $\min(\alpha^\alpha(1+\epsilon)^\alpha, \frac{1}{\epsilon})$. One price we pay for the fact that we can handle *any* sum scheduling objective is that our analysis is sub-optimal for specific problems, such as when the scheduling objective is total flow plus energy. Notice that notion of integral solutions for OnGAP do not apply for the integral versions of these energy minimization scheduling problems, since the notions of integrality are different: integrality for OnGAP means each job must be assigned to a single machine (i.e., a single time unit, when we cast OnGAP as an energy minimization scheduling problem), which is different from the concept of integrality for the scheduling objective.

6 Routing with Speed Scalable Routers

Our analysis will follow the same general approach as for OnGAP: we define dual variables $\widehat{\lambda}_j$ for the demand pairs, but now the minimization problem (which is over flow paths, and not just job assignments) is not so straight-forward: the different edges on a path p might want to set $f(p)$ to different values. So we do something seemingly bad: we *relax* the dual to decouple the variables, and allow each (edge, path) pair to choose its own "flow" value $f(p,e)$. And which of these should we use as our surrogate for $f(p)$? We use a convex combination $\sum_{e \in p} h_e \, f(p,e)$—where the multipliers $h(e)$ are chosen *based on the primal loads*(!), hence capturing the importance of edges.

6.1 The Algorithm and Analysis

We first consider the splittable flow version of the problem. Therefore, we can assume without loss of generality that all flow requirements are unit, and all sources and sinks are distinct (so we can associate a unique request $j(p)$ with each path p). This will also

allow us to order paths according to in when flow was sent along the paths. We now model the problem as follows:

$$\begin{aligned} \min \quad & \sum_e \left(\sum_j \sum_{p \ni e : p \in P_j} f(p) \right)^\alpha \\ \text{subject to} \quad & \sum_{p \in P_j} f(p) \geq 1 \qquad j = 1, \ldots, n \end{aligned}$$

where P_j is the set of all (s_j, t_j) paths, and $f(p)$ is a non-negative real variable denoting the amount of flow routed on the path p. In this case, the dual function is:

$$g(\lambda) = \min_{f(p)} \left(\sum_j \lambda_j + \sum_e \left(\sum_j \sum_{p \ni e : p \in P_j} f(p) \right)^\alpha - \sum_{j, p \in P_j} \lambda_j f(p) \right)$$

One can think of the dual as a routing problem with the same instance, but without the constraints that at least a unit of flow must be routed for each request. In the objective, in addition to energy costs, a fixed cost of λ_j is paid for each request j, and a payment of λ_j is received for each unit of flow routed from s_j to t_j.

Description of the Online Greedy Algorithm: For request j, flow is continuously routed along the paths that will increase costs the least until enough flow is routed to satisfy the request. That is, flow is routed along all (s_j, t_j) paths p that minimize $\sum_{e \in p} \alpha \cdot \left(\sum_{q \leq p : q \ni e} f(q) \right)^{\alpha - 1}$. For analysis purposes, after the flow for request j is routed, we define (where δ is a constant later set to $\frac{1}{\alpha^{\alpha-1}}$):

$$\widehat{\lambda}_j = \alpha \delta \left(\sum_{e \in p} \sum_{q \leq p : q \ni e} f(q) \right)^{\alpha - 1}$$

where p is *any* path along which flow for request j was routed.

The Analysis: Unfortunately, unlike the previous section for load balancing, it is not so clear how to compute the dual $g(\widehat{\lambda})$ or its minimizer since the variables cannot be nicely decoupled as we did there (per machine). In order to circumvent this difficulty, we consider the following *relaxed* function $\widehat{g}(\widehat{\lambda}, h)$, which does not enforce the constraint that flow must be routed along paths. This enables us to decouple variables and then argue about the objective value. Indeed, let $\widetilde{f}(p)$ be the final flow on path p for the routing produced by the online algorithm. Let $h(e) = \alpha \sum_{p \ni e} \widetilde{f}(p)^{\alpha - 1}$ be the incremental cost of routing additional flow along edge e, and $h(p) = \sum_{e \in p} h(e)$ be the incremental cost of routing additional flow along path p. We then define:

$$\widehat{g}(\widehat{\lambda}, h) = \min_{f(p,e)} \left(\sum_j \widehat{\lambda}_j + \sum_e \left(\sum_j \sum_{p \ni e : p \in P_j} f(p, e) \right)^\alpha - \sum_j \widehat{\lambda}_j \sum_{P \in P_j} \sum_{e \in p} \frac{h(e)}{h(p)} f(p, e) \right)$$

Conceptually, $f(p, e)$ can be viewed as the load placed on edge e by request $j(p)$. In $\widehat{g}(\widehat{\lambda}, h)$, the scheduler has the option of increasing the load on individual edges $e \in p \in P_j$, but the income from edge e will be a factor of $\frac{h(e)}{h(p)}$ less than the income achieved in $g(\widehat{\lambda})$. In Lemma 4 we prove that $\widehat{g}(\widehat{\lambda}, h)$ is a lower bound for $g(\widehat{\lambda})$. Lemma 5 shows how the minimizer and value of $\widehat{g}(\widehat{\lambda}, h)$ can be computed, and Lemma 6 shows how to bound some of the dual variables in terms of the final online primal solution.

Lemma 4. *For the above setting of $h(\cdot)$, $\widehat{g}(\widehat{\lambda}, h) \leq g(\widehat{\lambda})$.*

Proof. We show that there is a feasible value of $\widehat{g}(\widehat{\lambda}, h)$ that is less than $g(\widehat{\lambda})$. Let the value of $f(p,e)$ in $\widehat{g}(\widehat{\lambda}, h)$ be the same as the value of $f(p)$ in $g(\widehat{\lambda})$. Plugging these values for $f(p,e)$ into the expression for $\widehat{g}(\widehat{\lambda}, h)$, and simplifying, we get:

$$\begin{aligned}\widehat{g}(\widehat{\lambda}, h) &\leq \sum_j \widehat{\lambda}_j + \sum_e \left(\sum_j \sum_{p \ni e : p \in P_j} f(p) \right)^\alpha - \sum_j \widehat{\lambda}_j \sum_{P \in P_j} f(p) \sum_{e \in p} \frac{h(e)}{h(p)} \\ &= \sum_j \widehat{\lambda}_j + \sum_e \left(\sum_j \sum_{p \ni e : p \in P_j} f(p) \right)^\alpha - \sum_j \widehat{\lambda}_j \sum_{P \in P_j} f(p) = g(\widehat{\lambda})\end{aligned}$$

The first equality holds by the definitions of $h(e)$ and $h(p)$, and the second equality holds by the optimality of $f(p)$. ∎

Lemma 5. *There is a minimizer $\widehat{f}$ of $\widehat{g}(\widehat{\lambda}, h)$ s.t for any edge e, there is a single path $p(e)$ such that $\widehat{f}(p,e)$ is positive, and $\widehat{f}(p(e), e) = \left(\frac{\widehat{\lambda}_{j(p(e))} h(e)}{\alpha \cdot h(p(e))} \right)^{1/(\alpha-1)}$.*

Lemma 6. $\widehat{\lambda}_{j(p(e))} \leq \delta \cdot h(p(e))$

Proof. $\widehat{\lambda}_{j(p(e))}$ is δ times the rate at which the energy cost was increasing for the online algorithm when it routed the last bit of flow for request $j(p(e))$. $h(p(e))$ is the rate of at which the energy cost would increase for the online algorithm if additional flow was pushed along $p(e)$ after the last request was satisfied. If $p(e)$ was a path on which the online algorithm routed flow, then the result follows from the fact the online algorithm never decreases the flow on any edge. If $p(e)$ was not a path on which the online algorithm routed flow, then the lemma follows from the fact that, when the online algorithm was routing flow for request $j(p(e))$, $p(e)$ was more costly than the selected paths (and this cost can't decrease subsequently, by the monotonicity of the online algorithm). ∎

Theorem 2. *The online greedy algorithm is α^α competitive.*

Proof. We will show that $\widehat{g}(\widehat{\lambda}, h)$ is at least ON/α^α, which is sufficient since $\widehat{g}(\widehat{\lambda}, h)$ is a lower bound to $g(\widehat{\lambda})$ by Lemma 4, and since $g(\widehat{\lambda})$ is a lower bound to optimal.

$$\widehat{g}(\widehat{\lambda}, h) = \min_{f(p,e)} \left(\sum_j \widehat{\lambda}_j + \sum_e \left(\sum_j \sum_{p \ni e : p \in P_j} f(p,e) \right)^\alpha - \sum_j \widehat{\lambda}_j \sum_{P \in P_j} \sum_{e \in p} \frac{h(e)}{h(p)} f(p,e) \right) \tag{6.13}$$

$$= \sum_j \widehat{\lambda}_j - (\alpha - 1) \sum_e \left(\frac{\widehat{\lambda}_{j(p(e))} h(e)}{\alpha \cdot h(p(e))} \right)^{\alpha/(\alpha-1)} \tag{6.14}$$

$$\geq \sum_j \widehat{\lambda}_j - (\alpha - 1) \sum_e \left(\frac{\delta \cdot h(e)}{\alpha} \right)^{\alpha/(\alpha-1)} \tag{6.15}$$

$$= \sum_j \widehat{\lambda}_j - (\alpha - 1) \delta^{\alpha/(\alpha-1)} \sum_e \left(\sum_{p \ni e} \widetilde{f}(p) \right)^\alpha \tag{6.16}$$

$$= \sum_j \widehat{\lambda}_j \sum_{p \in P_j} \widetilde{f}(p) - (\alpha-1)\delta^{\alpha/(\alpha-1)} \sum_e \left(\sum_{p \ni e} \widetilde{f}(p) \right)^\alpha \tag{6.17}$$

$$= \delta\alpha \sum_j \sum_{p \in P_j} \widetilde{f}(p) \left(\sum_{e \in p} \sum_{q \le p : q \ni e} \widetilde{f}(q) \right)^{\alpha-1} - (\alpha-1)\delta^{\alpha/(\alpha-1)} \sum_e \left(\sum_{p \ni e} \widetilde{f}(p) \right)^\alpha \tag{6.18}$$

$$\geq \delta \sum_e \left(\sum_{p \ni e} \widetilde{f}(p) \right)^\alpha - (\alpha-1)\delta^{\alpha/(\alpha-1)} \sum_e \left(\sum_{p \ni e} \widetilde{f}(p) \right)^\alpha \tag{6.19}$$

$$= \frac{1}{\alpha^\alpha} \sum_e \left(\sum_{p \ni e} \widetilde{f}(p) \right)^\alpha \geq \text{ON}/\alpha^\alpha \tag{6.20}$$

The equality in line (6.13) is the definition of $\widehat{g}(\widehat{\lambda}, h)$. The equality in line (6.14) follows from Lemma 5. The inequality in line (6.15) follows from Lemma 6. The equality in line (6.16) follows from the definition of $h(e)$. The equality in line (6.17) follows from the feasibility of $\widetilde{f}$. The equality in line (6.18) follows from the definition of $\widehat{\lambda}$. The equality in line (6.19) follows from the definition of δ. ■

7 Conclusion

The online primal-dual technique (surveyed in [5]) has proven to be a widely and systematically applicable method to analyze online algorithms for problems expressible by linear programs. This paper develops an analogous technique to analyze online algorithms for problems expressible by *nonlinear* programs. The main difference is that in the nonlinear setting one can not disentangle the objective and the constraints in the dual, and hence the arguments for the dual have a somewhat different feel to them than in the linear setting. We apply this technique to several natural nonlinear covering problems, most notably obtaining competitive analysis for greedy algorithms for uniprocessor speed-scaling problems with arbitrary sum scheduling objectives that researchers were not previously able to analyze using the prevailing potential function based analysis techniques.

References

1. Im, S., Moseley, B., Pruhs, K.: A tutorial on amortized local competitiveness in online scheduling. SIGACT News 42(2), 83–97 (2011)
2. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Pruhs, K.: Online weighted flow time and deadline scheduling. Journal of Discrete Algorithms 4(3), 339–352 (2006)
3. Bansal, N., Pruhs, K., Stein, C.: Speed scaling for weighted flow time. SIAM J. Comput. 39(4), 1294–1308 (2009)
4. Andrews, M., Antonakopoulos, S., Zhang, L.: Energy-aware scheduling algorithms for network stability. In: INFOCOM, pp. 1359–1367 (2011)
5. Buchbinder, N., Naor, J.S.: The design of competitive online algorithms via a primal-dual approach. Foundations and Trends in Theoretical Computer Science 3(2-3), 93–263 (2009)
6. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2004)

7. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced CPU energy. In: Proc. IEEE Symposium on Foundations of Computer Science, pp. 374–382 (1995)
8. Bansal, N., Kimbrel, T., Pruhs, K.: Speed scaling to manage energy and temperature. Journal of the ACM 54(1) (2007)
9. Bansal, N., Bunde, D.P., Chan, H.L., Pruhs, K.: Average rate speed scaling. Algorithmica 60(4), 877–889 (2011)
10. Bansal, N., Chan, H.-L., Pruhs, K., Katz, D.: Improved bounds for speed scaling in devices obeying the cube-root rule. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 144–155. Springer, Heidelberg (2009)
11. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Transactions on Algorithms 3(4), 49 (2007)
12. Bansal, N., Chan, H.L., Pruhs, K.: Speed scaling with an arbitrary power function. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 693–701 (2009)
13. Lam, T., Lee, L., To, I., Wong, P.: Speed scaling functions based for flow time scheduling based on active job count. In: European Symposium on Algorithms, pp. 647–659 (2008)
14. Andrew, L.L.H., Lin, M., Wierman, A.: Optimality, fairness, and robustness in speed scaling designs. In: SIGMETRICS, pp. 37–48 (2010)
15. Chan, H.L., Edmonds, J., Lam, T.W., Lee, L.K., Marchetti-Spaccamela, A., Pruhs, K.: Nonclairvoyant speed scaling for flow and energy. In: Symposium on Theoretical Aspects of Computer Science, pp. 255–264 (2009)
16. Chan, S.H., Lam, T.W., Lee, L.K.: Non-clairvoyant speed scaling for weighted flow time. In: European Symposium on Algorithms, pp. 23–35 (2010)
17. Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts, O.: On-line routing of virtual circuits with applications to load balancing and machine scheduling. Journal of the ACM 44(3), 486–504 (1997)
18. Awerbuch, B., Azar, Y., Grove, E.F., Kao, M.Y., Krishnan, P., Vitter, J.S.: Load balancing in the L_p norm. In: IEEE Symposium on Foundations of Computer Science, pp. 383–391 (1995)
19. Albers, S.: Energy-efficient algorithms. Communications of the ACM 53(5), 86–96 (2010)
20. Caragiannis, I.: Better bounds for online load balancing on unrelated machines. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 972–981. ACM, New York (2008)
21. Azar, Y., Epstein, A.: Convex programming for scheduling unrelated parallel machines. In: ACM Symposium on Theory of Computing, pp. 331–337. ACM, New York (2005)
22. Anil Kumar, V.S., Marathe, M.V., Parthasarathy, S., Srinivasan, A.: A unified approach to scheduling on unrelated parallel machines. Journal of the ACM 56(5), Art. 28, 31 (2009)
23. Andrews, M., Antonakopoulos, S., Zhang, L.: Minimum-cost network design with (dis)economies of scale. In: IEEE Symposium on Foundations of Computer Science, pp. 585–592 (2010)
24. Gupta, A., Im, S., Krishnaswamy, R., Moseley, B., Pruhs, K.: Scheduling jobs with varying parallelizability to reduce variance. In: ACM Symposium on Parallelism in Algorithms and Architectures, pp. 11–20 (2010)
25. Im, S., Moseley, B.: Online scalable algorithm for minimizing lk-norms of weighted flow time on unrelated machines. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 95–108 (2011)
26. Anand, S., Garg, N., Kumar, A.: Resource augmentation for weighted flow-time explained by dual fitting. In: ACM-SIAM Symposium on Discrete Algorithms (2012)