ADFOCS 2015 Pruhs Homework 1

We will consider the following problem (and some variations thereof): A problem instance consists of n tasks. Task i has a release time r_i, a deadline $d_i > r_i$, and work $w_i > 0$. In the online version of the problem, the scheduler learns about a task only at its release time; at this time, the scheduler also learns the exact work requirement and the deadline of the task. We assume that time is continuous. A schedule specifies for each time a task to be run and a speed at which to run the task. The speed is the amount of work performed on the task per unit time. A task with work w run at a constant speed s thus takes $\frac{w}{s}$ time to complete. More generally, the work done on a task during a time period is the integral over that time period of the speed at which the task is run. A schedule is *feasible* if for each task i, work at least w_i is done on task i during $[r_i, d_i]$. Note that the times at which work is performed on task i do not have to be contiguous. If a task is run at speed s, then the power is $P(s) = s^2$. The energy used during a time period is the integral of the power over that time period. The objective is to minimize the total energy used.

1. (Warmup) Assume for the moment that the speed at each time was fixed a priori. So there may not be a feasible schedule. Convince yourself that if a feasible schedule exists, then a feasible schedule can be produced by always running the released unfinished job with an earliest deadline. This can formally be shown using an exchange argument, but you do not need to do a formal proof. So we can conclude from this that the job selection policy can without loss of generality be to run the earliest deadline job, and we need only focus on algorithms for setting the speed.

2. (Warmup) Consider the offline YDS algorithm defined below. Let $w(t_1, t_2)$ denote the work that has release time at least t_1 and has deadline at most t_2. The *intensity* $I(t_1, t_2)$ of the time interval $[t_1, t_2]$ is defined to be $w(t_1, t_2)/(t_2 - t_1)$.

   Algorithm YDS: The algorithm repeats the following steps until all jobs are scheduled:

   - Let $[t_1, t_2]$ be the maximum intensity time interval.
   - The processor will run at speed $I(t_1, t_2)$ during $[t_1, t_2]$ and schedule all the jobs comprising $w(t_1, t_2)$, always running the released, unfinished task with the earliest deadline.
   - Then the instance is modified as if the times $[t_1, t_2]$ didn't exist. That is, all deadlines $d_i > t_1$ are reduced to $\max(t_1, d_i - (t_2 - t_1))$, and all release times $r_i > t_1$ are reduced to $\max(t_1, r_i - (t_2 - t_1))$.

   Explain why YDS is optimal with respect to the objective of minimizing the maximum of all times t of the power used at time t. Hint: There is a very simple proof. The purpose of this problem is just to get you to understand the YDS algorithm.

3. Express the energy minimization problem as a convex program, and then use the KKT conditions to show that YDS also produces an energy optimal schedule. Below find a review of the KKT conditions and a few hints.

   KKT Conditions: Consider a convex program

$$\begin{aligned} \min\ & f_0(x) \\ f_i(x) \quad & \leq 0 \qquad i = 1, \ldots, n \end{aligned}$$

   Assume that this program is strictly feasible, that is, there is some point x where $f_i(x) < 0$ for $i = 1, \ldots, n$. Assume that the f_i are all differentiable. Then a necessary condition for a solution x be be optimal is the existence of a Lagrangian multiplier λ_i associated with each function $f_i(x)$ such that:

$$f_i(x) \leq 0 \qquad i = 1, \ldots, n \tag{1}$$

$$\lambda_i \geq 0 \qquad i = 1, \ldots, n \tag{2}$$

$$\lambda_i f_i(x) = 0 \qquad i = 1, \ldots, n \tag{3}$$

$$\nabla f_0(x) + \sum_{i=1}^{n} \lambda_i \nabla f_i(x) = 0 \tag{4}$$

Hint: To state the energy minimization problem as a convex program, break time into intervals $t_0, \ldots t_m$ at release times and deadlines of the tasks. Why can you assume that the processor runs at constant speed throughout any such interval? Introduce a variable $x_{i,j}$ that represents the work done on task j during time $[t_i, t_{i+1}]$. The (interval indexed) program has as its objective energy, and two types of constraints: that every job is run enough, and that the $x_{i,j}$'s are positive. Associate a dual variable δ_j with the constraint saying that job j must be finished, and $\gamma_{i,j}$ with the constraint that $x_{i,j}$'s is positive. You will need to find settings of the dual variables so that the primal solution produce by YDS satisfies the KKT conditions. Complementary slackness will be useful.

4. Consider the following online algorithm OA and show that it is O(1)-competitive with respect to energy using the standard potential function under the simplifying assumption that all jobs have the same deadline. Conceptually the analysis for arbitrary deadlines is the same, just more notation heavy. Some review and copious hints follow. Don't worry about optimizing/minimizing the competitive ratio.

   Algorithm OA: Maintain the invariant that at all times t, the task with the earliest deadline is run at speed $\max_t w(t)/t$, where $w(t)$ is the unfinished work that has deadline within the next t units of time. An alternative description is that OA's schedule for the future is always the optimal energy YDS schedule based on the current state.

   Review of Amortized Local Competitiveness: Let $G_a(t)$ and $G_o(t)$ be the objective value (energy in this problem) accumulated by an online algorithm A and optimal up to time t, respectively. To prove that an online scheduling algorithm A is c-competitive using an amortized local competitiveness argument, it suffices to give a potential function $\Phi(t)$ such that the following conditions hold.

   Boundary condition: Φ is zero before any job is released and Φ is non-negative after all jobs are finished.

   Completion condition: Φ does not increase when either the algorithm or the adversary completes a job.

   Arrival condition: For all job arrivals, Φ does not increase.

   Running condition: At any time t when no job arrives or is completed,

$$\frac{\mathtt{d}G_a(t)}{\mathtt{dt}} + \frac{\mathtt{d}\Phi(t)}{\mathtt{dt}} \leq c \cdot \frac{\mathtt{d}G_o(t)}{\mathtt{dt}} \tag{5}$$

   Integrating these conditions over time one gets that $G_a - \Phi(0) + \Phi(\infty) \leq c \cdot G_o$. Informally the standard potential function is the future cost to the online algorithm if no more jobs arrive and if the remaining work for each job is how far the online algorithm is behind on that job.

   Some notation: Let t be the time until the common deadline. Let w_a be the amount of unfinished work OA has, and w_o be the amount of unfinished work that optimal has. Let s_a be the speed on OA and s_o be the speed of optimal. Note that all of w_a, w_o, s_a and s_o are functions of t, which we suppress for notational simplicity.

   (a) Why is OA's future cost if no more jobs arrive equal to ts_a^2, and equal to w_a^2/t? Thus the standard potential function is $\Phi = \beta[\max(0, w_a - w_o)]^2/t$, for some constant β.

   (b) Why does Φ satisfy the boundary, completion and arrival conditions? So let us turn to the running condition for the rest of the problem.

   (c) Why is $s_o \geq w_o/t$?

   (d) Why is $\frac{\mathtt{d}G_a(t)}{\mathtt{dt}} = s_a^2$? Why is $\frac{\mathtt{d}G_o(t)}{\mathtt{dt}} = s_o^2$?

   (e) Why does the running condition obviously hold when $w_o \geq w_a$? So from now on let us assume $w_a > w_o$.

   (f) What is $\frac{\mathtt{d}\Phi(t)}{\mathtt{dt}}$? Be very careful about the signs when you differentiate, it is easy to make sign errors.

(g) Now prove the running condition for some constant c using the facts that $s_a = w_a/t$, $w_a - w_o \leq w_a$, $s_o \geq w_o/t$, and Young's inequality that $ab \leq a^2/2 + b^2/2$.

5. Here we give another online algorithm PD using the online primal dual technique. Your goal is to show using dual fitting that PD is 4-competitive. A review and copious hints follow. We can think of this problem as being modeled by the following convex program

$$\begin{aligned} \min \quad & \sum_t \left(\sum_j x_{jt} \right)^2 \\ \text{subject to} \quad & \sum_t x_{jt} \geq 1 \qquad j = 1, \ldots, n \end{aligned} \tag{6}$$

where x_{jt} denotes the extent to which job j is run at time t. Here we are assuming without loss of generality that each job has one unit of work. Note that an online algorithm can view the constraints as arriving online.

The Lagrangian dual of the primal is then

$$g(\lambda) = \min_{x \geq 0} \left(\sum_j \lambda_j + \sum_t \left(\sum_j x_{jt} \right)^2 - \sum_{j,t} \lambda_j \, x_{jt} \right) \tag{7}$$

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 $\sum_t \left(\sum_j x_{jt} \right)^\alpha$ as in the primal, a fixed cost of λ_j is paid for each job j, and a *payment* (or negative cost) of λ_j is obtained for each unit of job j assigned.

Online PD Algorithm Description: Now the algorithm works as follows when a new job j arrives: until a unit fraction is scheduled, job j is scheduled on all feasible times for which the increase in the cost will be the least. More formally, the value of all the primal variables x_{jt} for all the times t that minimize $2 \sum_{i \leq j} x_{it}$ are increased until all the work from job j is scheduled. Let $\widetilde{x}$ denote the final value of the x_{jt} variables for the online algorithm.

Now, for the purpose of analysis, let $\widehat{\lambda}_j$ to be half the rate of increase of the objective value when algorithm PD assigned the last infinitesimal portion of job j, and then let $\widehat{x}$ be the value of the minimizing x variables in $g(\widehat{\lambda})$

(a) (Warmup) Show that if there is a common deadline for all jobs that the algorithm PD and the algorithm OA always are running at the same speed.

(b) (Warmup) Give an instance where PD and OA will run at different speeds at some times.

(c) Express the value of $\widehat{\lambda}_j$ in terms of the variables $\widetilde{x_{i,t}}$, where t is some time that the algorithm PD runs j.

(d) Give a greedy algorithm to compute $g(\widehat{\lambda})$. Give the values of the $\widehat{x}_{jt}$ variables computed by this algorithm in terms of the $\widehat{\lambda}_j$ terms. Hint: Only one variable of the form $\widehat{x}_{*t}$ will be positive. Call this variable $\widehat{x}_{\psi(t)t}$.

(e) What is special about the job $\psi(t)$ for the algorithm PD? That is, if you watched PD's execution, how could you identify the job $\psi(t)$?

(f) Show that $\sum_j \widehat{\lambda}_j$ is at least half of the cost of the algorithm PD. Hint: $\sum_j \widehat{\lambda}_j = \sum_{j,t} \widehat{\lambda}_j \widetilde{x}_{jt}$. Previously, you determined how to relate the $\widehat{\lambda}$ terms to the $\widetilde{x}$ terms.

(g) Show that $\sum_t \left(\sum_j \widehat{x_{jt}} \right)^2 - \sum_{j,t} \widehat{\lambda_j} \, \widehat{x_{jt}}$ is equal to minus one quarter of the cost of the algorithm PD. Hint: Use the fact that you know how to express the $\widehat{x}$ terms in terms of the $\widehat{\lambda}$ terms, which in turn you know how to relate to the $\widetilde{x}$ terms.

(h) Put all this together to conclude that the algorithm PD is 4-competitive.