ADFOCS 2015 Pruhs Homework 2

1. The goal of this problem is to give a relatively simple proof that the online hallucination algorithm is poly-log competitive for multicast routing, that is when all the sinks are the same. Recall that the main issue was to show that the optimal algorithm can not do more than poly-log better by using edges before the online algorithm could. To accomplish this, explain how to take any solution, and produce a solution that is at most poly-log more expensive and doesn't use any edges before the online algorithm adds them to the Steiner tree or the hallucination backbone. Don't worry about the degree of the polynomial in the poly-log term. Here we are assuming that α is a constant like 2 or 3.

   A review and hints follow:

   Problem Definition: Recall that the input to Energy Efficient Routing Problem is an undirected multi-graph $G = (V, E)$ with $|V| = n$ vertices, m edges, and k request-pairs $\{(s_i, t_i)\}_{i=1}^{k}$ that must be routed in G. The demand for each request pair is 1. Here we consider the multicast setting where each t_i is a common sink t. The cost for routing f units of flow over any edge is: zero if $f = 0$, and $\sigma + f^\alpha$ if $f > 0$. The power incurred by any solution is naturally split into two parts: (i) *static power* which is σ times the number of edges with positive flow, and (ii) *dynamic power* which is $\sum_{e \in E} f_e^\alpha$ where f_e denotes the flow on edge $e \in E$. A useful parameter throughout the paper is $q := \sigma^{1/\alpha}$, which is the amount of flow on an edge for which the static and dynamic power are equal.

   Online Hallucination Algorithm: In response to request-pair (s_i, t_i) the algorithm takes the following steps:
   1. Augmenting the Steiner backbone: We run an online algorithm for Steiner forest to connect s_i and t_i, where the cost of edges is the static power σ. Let $G_S(i)$ be the Steiner forest maintained after the i^{th} request-pair.
   2. Augmenting the Hallucination backbone: The request-pair (s_i, t_i) hallucinates a demand of q with probability $p = \min(1, 32 \log k/q)$. We then find the cheapest unsplittable routing of q units of hallucinated flow between s_i and t_i. The edges used in this routing are added to the hallucinated backbone $G_H(i-1)$ to obtain $G_H(i)$.
   3. Routing: We route one unit of actual flow between s_i and t_i in the graph $G_F(i) = G_S(i) \cup G_H(i)$, to minimize the increase in dynamic power.

   Explain how to take any solution, and produce a solution that is at most poly-log more expensive and doesn't use any edges before the online algorithm adds them to the Steiner tree or the hallucination backbone. Don't worry about the degree of the polynomial in the poly-log term. Here we are assuming that α is a constant like 2 or 3.

   Hint: Consider the offline algorithm specifically designed for multicast routing given in lecture. To refresh your memory, this algorithm powered on a Steiner tree, aggregated sources in groups of q in the Steiner tree, and then routed the flow from the aggregated sources to the common sink. Use Chernoff bounds. Basically all you need is that if there are at least $\log n$ many fair independent coin flips, then probability that the number of heads is $< .01 \log n$ is at most 1 over some polynomial of high degree.

2. Another common energy/power related objective is temperature, which I didn't talk about. So the purpose of this problem is to getting you thinking about temperature using the standard model of cooling, namely Newton's law of cooling.

   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.

Cooling is a complex phenomenon that cannot be captured completely accurately by any simple model. For tractability we require a simple first-order approximation. Our key assumptions are that heat is lost via conduction, and the ambient temperature of the environment surrounding the device is constant. This is likely a reasonable first-order approximation in some, but certainly not all, settings. Then we appeal to Newton's law of cooling, which states that the rate of cooling is proportional to the difference in temperature between the object and the ambient environmental temperature. Without loss of generality we may assume that the temperature scale is translated so that the ambient temperature is zero. If we assume that the net change in temperature is the sum of the decrease due to cooling as described above and an increase proportional to the electrical power applied to the device, a first-order approximation for rate of change $T'(t)$ of the temperature $T(t)$ at time t is then given by the equation:

$$T'(t) = P(t) - bT(t)$$

where $P(t)$ is the supplied power at time t, and $b \geq 0$ is the *cooling parameter* of the device. We will assume that the initial temperature of the processor is 0.

We consider the objective of minimizing the maximum temperature of the processor. Our goal is to show that YDS is $O(1)$-approximate with respect to maximum temperature.

(a) Show that the maximum temperature reach by a particular schedule is within a constant factor of the most energy used in any time interval of length $c = 1/b$.

(b) Use this fact to show that the online algorithm OA is not $O(1)$-approximate with respect to maximum temperature. OA was defined in homework 1. Hint: OA was defined in homework 1. It is sufficient to consider a simple instance with a common deadline.

(c) We first require some preliminary definitions and observations. Let $E(I)$ denote an interval of length c in the YDS schedule that uses the most energy. Say $E(I)$ uses energy E. Let ϵ be a small positive constant. Let the speed s_0 be defined as

$$s_0 = (\epsilon E/c)^{1/2}$$

We call a task *slow* if it runs at a speed strictly less than s_0 in YDS(I). This notion is well-defined because each task runs at constant speed (when it is run) in the YDS schedule. The rest of the tasks are called *fast*. Let $s(t)$ denote the speed at time t in YDS(I). Define an *island* to be a maximal interval of time where $s(t) \geq s_0$. Why are all tasks that YDS executes in an island also executed in that island by optimal?

(d) Why is most of the energy in $E(I)$ used by fast tasks?

(e) If there is an island of length at least c then why does optimal have to have an interval of length c where it uses at least a constant fraction of E energy?

(f) Show that if all islands are of length less than c then why does optimal have to have an interval of length c where it uses at least a constant fraction of E energy? Hint: Consider the islands that intersect $E(I)$.

(g) Conclude that YDS is $O(1)$-approximate with respect to maximum temperature.

3. Here we are consider another temperature related scheduling problem. The goal of this problem is learn a bit about calculus of variations, which is useful occasionally.

Introduction to relevant calculus of variations material: Suppose that we want to find the "nice" function $Y(x)$ that maximizes (or minimizes) the value of a functional

$$J(Y) = \int_{x_0}^{x_1} F(x, Y(x), Y'(x))dx$$

subject to the constraint that $Y(x_0) = y_0$ and $Y(x_1) = y_1$. Here $Y'(x)$ is the derivative of $Y(x)$ with respect to x. So here F and the points x_0, x_1, y_0 and y_1 are given. And the goal is to find the function $Y(x)$ that maximizes the integral. Then $Y(x)$ must satisfy:

$$F_Y(x, Y(x), Y'(x)) - \frac{d}{dx} F_{Y'}(x, Y(x), Y'(x)) = 0$$

Here F_Y is derivative of F with respect to Y, and $F_{Y'}$ is derivative of F with respect to Y'.

The problem: Assume that you have a speed scalable processor with allowable speeds in the range $[0, \infty)$. Assume that the power is the speed squared. Assume that the temperature is governed by Newton's law of cooling, namely

$$T'(t) = P(t) - T(t)/2$$

Here $T(t)$ is the temperature at time t, $T'(t)$ is derivative of $T(t)$ with respect to t, and $P(t)$ is the power at time t. Assume that we start at time 0 with temperature 0 and the temperature at time 1 must be 1. The problem is to determine at what speed/power to run to maximize the amount of work that can be processed between time 0 and 1 while meeting the constraints on starting and ending temperatures.

(a) Explain how the speed $s(t)$ at time t can be determined from a given temperature curve $T(t)$.

(b) Write the the work processed as an integral over a function F of T, and T'.

(c) Calculate F_T, the derivative of F with respect to T.

(d) Calculate $F_{T'}$, the derivative of F with respect to T'.

(e) Calculate the derivative of $F_{T'}$ with respect to t. This introduces a T'' term, which is the second derivative of T with respect to t.

(f) Write down the resulting second order differential equation that T must satisfy.

(g) Verify that $T(t) = (e^{t/2} - e^{-t})/(e^{-1/2} - e^{-1})$ is a solution to this differential equation and satisfies the required starting and ending temperatures.