

CSE 548 Homework Assignment 3

Due in Class on Thursday, October 29

October 21, 2009

There are four problems. Generic format of the solution is as follows:

1. Discuss your idea briefly.
2. Give the proof of optimality, in terms of well constructed lemmas and theorems.
3. Argue the time complexity of your algorithm.

Problems

1. Suppose you have n video streams that need to be sent, one after another, over a communication link. Stream i consists of a total of b_i bits that need to be sent, at a constant rate, over a period of t_i seconds. (Thus, the rate for sending stream i is b_i/t_i .) You cannot send two streams at the same time, so you need to determine a schedule for the streams: an order in which to send them. Whichever order you choose, there cannot be any delays between the end of one stream and the start of the next. Suppose your schedule starts at time 0 (and therefore ends at time $\sum_{i=0}^n t_i$, whichever order you choose). We assume that all the values b and t are positive integers.

Now, because you are just one user, the link does not want you taking up too much bandwidth, so it imposes the following constraint, using a fixed parameter r :

(*) For each natural number $t > 0$, the total number of bits you send over the time interval from 0 to t cannot exceed rt .

Note that this constraint is only imposed for time intervals that start at 0, not for time intervals that start at any other value. (This scheme is a variation of the so-called leaky bucket algorithm for rate limiting or traffic shaping in networking.)

We say that a schedule is valid if it satisfies the constraint (*) imposed by the link.

The problem Given a set of n streams, each specified by its number of bits b_i and its time duration t_i , as well as the link parameter r , determine whether there exists a valid schedule. Consider an example. Suppose we have $n = 3$ streams with

$$(b_1, t_1) = (2000, 1), (b_2, t_2) = (6000, 2), (b_3, t_3) = (2000, 1)$$

and suppose the link's parameter $r = 5000$. Then the schedule that runs the order 1, 2, 3 is valid since constrain (*) is satisfied:

1. at $t = 1$: the whole first stream has been sent.
2. at $t = 2$: half the second stream has been sent, and $2000 + 3000 < 5000 \cdot 2$
3. at $t = 3, 4$: similar calculations hold.

Questions

- a. Consider the following claim: *there exists a valid schedule if and only if each stream i satisfies $b_i \leq rt_i$.*
Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

- b. Given an algorithm that takes the set of n streams, each specified by its time duration t_i , as well as the link parameter r and determines whether there exists a valid schedule. The running time of your algorithm must be a polynomial in n .
2. Consider the following scheduling problem. You have a set of n computational jobs. Job i runs for time l_i , and only one job can be run at a time (and cannot be preempted). Moreover, there is a set of precedence constraints on the jobs, of the form job i must be done before job j . The precedence constraints form a directed acyclic graph, such as that shown in Figure 1. Any feasible schedule must satisfy all precedence constraints.

With each job i , there is associated a loss function $f_i(t)$ that measures the badness of putting off a job so that it does not complete until time t . The loss function for each job is monotonically increasing; that is, if $t_1 < t_2$, then $f_i(t_1) \leq f_i(t_2)$. However, these functions are otherwise arbitrary! For example, a real-time job might be fine (zero loss) if it completes in at most $t = 5$ seconds but would cause a system failure (high loss) if it completes after this time. For a batch job, the loss function may increase a little bit every second, to discourage the system from putting it off too long.

Your goal is to find a feasible schedule that minimizes the loss of the lossiest job. More precisely, let S be a feasible schedule, and let job i complete at time e_i in S . Then the goal is to construct S so as to minimize $\max_{1 \leq i \leq n} f_i(e_i)$.

3. As we develop sound monetization models, social networks are poised to be the next big thing. Suppose you are developing an application for a professional social network called *pro-matroid.com*. It provides a networking platform for professionals from various industries. With the economy poised to boom, you foresee potential in using this network to match job hunters and employers. You plan to do so by providing an application to the employers.

Prospective employees post their resume and the employers post jobs on the network. Job seekers also post an expected remuneration.

You already have an algorithm that matches applicants to jobs purely based on the skill sets and resume. After running this algorithm, you have pairings of jobs and potential matches. Remember an applicant might be a match for more than one job. Only thing remaining to be optimized is the remuneration.

Now develop an algorithm that given this match, finds the cheapest pairing of jobs-applicants. The application would present this to the employers which they can add to their social network profiles.

Prove the correctness of your algorithm using Matroids.

4. Synchronization in distributed wireless sensor networks is a major concern. Due to topographical constraints, you have a weird routing protocol. Under this protocol, the messages start from a node and travel down a binary tree to eventually reach the leaves that represents the different sinks in the network. Nodes of this tree represent the sensor nodes acting as relays.

A potential problem with having many sinks is that they will receive replicas of the information generated by the node at the root. This introduces the need of synchronization as all sinks must receive the message at the same time, for any co-ordinated response in real time.

So imagine the binary tree on which the message gets routed as made up of edges having weights. Weight l_e of edge e represents the delay incurred on that hop. So the total delay from the root (from where the message emanates) to a sink will be the sum of edges from the root to the sink.

To ensure that the message reaches all sinks (leaves) at the same time, we must increase the weight (delay) of some of the edges. (We can not decrease the weights). This basically corresponds to buffering the packet at some intermediate nodes.

The goal is to achieve synchronization in terms of packet deliver time in a way that minimizes the total delay. Devise an algorithm that achieves the goal of simultaneous delivery at all nodes with minimum total delay.

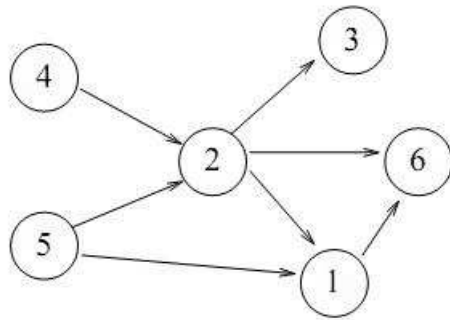


Figure 1: sample DAG for Problem 2