COS 598a

February 19, 2001

Assignment 2

Homeworks

Exercise 1 Prove that for $m > 3$, any deterministic online scheduling algorithm must have competitive ratio $\geq 1 + 1/\sqrt{2}$.

Exercise 2 In class, we showed that the given randomized scheduling algorithm has competitive ratio $\leq 4/3$, under the assumption that each new task is no lengthier than the sum of lengths of all previous tasks. Prove the assertion without the latter assumption. (For the solution, see Bartal et al [4]*.)

Research Problems

Problem 1 The radio broadcast problem has a randomized $O(n \log n)$ solutions, and $\Omega(n)$ lower bound. Narrow the gap. Perhaps it is worthwhile to resolve first the following simple question as a warm-up. Is there an $\Omega(\log n)$ lower bound for Exercise 2 in Assignment 1? I don't know the answer.

Problem 2 The best constructive deterministic algorithm for radio broadcast takes $\tilde{O}(n^{3/2})$. Can you do better?

Problem 3 The radio gossip problem has $\tilde{O}(n^{3/2})$ solutions. Is there an $O(n^c)$ solution for some $c < 3/2$? Randomized algorithms are acceptable. Also, can you prove a nontrivial lower bound for this problem?

Problem 4 Optimal online scheduling algorithms are known only for $m = 2, 3$ in the deterministic case, and for $m = 2$ in the randomized case. Narrow the gaps between upper and lower bounds. For example:
(a) Can you improve the bounds for $m = 4$ in the deterministic case** and for $m = 3$ in the randomized case? Can you find a lower bound better than $e/(e-1)$ for large m in the randomized case?
(b) For $m > 7$ there are no known randomized algorithms performing better than the best existing deterministic ones. Can you construct such a randomized algorithm for some $m > 7$. Even more interestingly, for large m, can you design a randomized algorithm

with a performance better than 1.923 (the best known deterministic performance upper bound), or even 1.852 (the known lower bound for deterministic performance)?

Notes

Current information on radio broadcast and gossip is contained in Chrobak, Gasieniec and Rytter[1], on deterministic online scheduling in Albers[2], on randomized online scheduling in Seiden[3]. The optimal randomized online algorithm for $m = 2$ discussed was discovered by Bartal, Fiat, Karloff and Vohra[4]. The lower bound $1 + \frac{1}{(m/(m-1))^m - 1}$ for randomized algorithms can be found in Sgall[5] (and also independently by Chen, van Vliet and Woeginger, see the reference in Seiden [3]).

*All reference numberings refer to the *handout* web page for the course.

** According to Sgall's survey paper "Online scheduling" (available from the same web site as [5]), R. Chandrasekaran claimed a lower bound $\sqrt{3}$ for the $m = 4$ case.

Appendix: Material for Assignment 2

There is some background material (from [4]) needed for doing Exercise 2 in Assignment 2.

We want to show that the best competitive ratio for any randomized online scheduling on 2 processors is exactly 4/3. First we show any randomized algorithm R must have competitive ratio $\geq 4/3$. Let $L_1 = (1,1)$ and $L_2 = (1,1,2)$ be two lists. On input L_1, algorithm R will schedule with certain probability p one task for each processor, and with probability $1-p$ both tasks on one processor. Then the expected performance on L_1 is $p \cdot 1 + (1-p) \cdot 2 = 2 - p$. How does R perform on L_2? It is easy to see that it has to be at least $p \cdot 3/2 + (1-p) \cdot 1 = 1 + p/2$. This implies that the competitive ratio of R must satisfy

$$\sigma_R \geq \max\{2-p, 1+p/2\} \geq \frac{1}{3}(2-p) + \frac{2}{3}(1+p/2) = 4/3.$$

Note that to achieve this ratio, $1-p$ must be $1/3$. The moral is, for a good randomized algorithm, when faced with L_1 needs to choose with some probability a bad current schedule so that it might accomodate future tasks more efficiently.

For an input list starting with $1,1,2$, the previous discussion shows that a randomized algorithm can achieve $4/3$ by using $p = 2/3$ when the second task is being scheduled. But this is just a simple input list. The amazing thing is that there is a general rule to handle *any* list, no matter how complicated, so that the performance at any time is maintained at a level equal to or better than 4/3.

At any time, the *load* on a processor is the sum of lengths of tasks scheduled on it. The *total load* is the sum of loads on both processors. The processor with the heavier load is called the *tall* processor, the other one *short*. The *discrepancy* is the difference between the two loads. The general rule for the algorithm can be described inductively. Suppose n tasks have been scheduled so far with a total load S, and a new task a is to be scheduled. Let $C_1, C_2, \cdots, C_N$ (where $N = 2^n$) be the set of all possible schedules of the initial n tasks, and let p_i be the probability that C_i is produced by our rule. Let b_i be the discrepancy of schedule C_i. The expected discrepancy after scheduing these n tasks is thus $\sum_i p_i b_i$.

Let E be the expected discrepancy if we decide (deterministically) to put the new task a into the short processor, and E' be expected discrepancy if a is put into the tall processor. The rule is as follows:

(i) if there exists a p such that $pE + (1-p)E' = (S+a)/3$, then put a into the short processor with probability p, and
(ii) if no such p exists, just put a into the short processor.
In other words, if you can make the expected discrepancy equal to 1/3 of the total load, do it; otherwise schedule the task on the short processor.

It is important to emphasize that the actual current configuration is irrelevant in deciding whether (i) or (ii) is true. Thus, it is possible that currently the tall processor has a very heavy load, but (i) is true and we still have to schedule a in the tall processor with some non-zero probability.

Now we show that the randomized algorithm R specified by this rule achieves $\sigma_R \leq 4/3$. That is, for any list $L = (a_1, a_2, \cdots, a_n)$, the expected makespan

$$T_R(L) \leq (4/3)OPT(L). \tag{1}$$

Call L a *regular list* if every new task is no greater than the sum of all the previous tasks; ie,

$$a_k \leq a_1 + a_2 \cdots + a_{k-1} \tag{2}$$

for all $1 < k \leq n$. Below we will prove the result under the assumption that L is regular. The general case is left for you to prove in Exercise 2 of Assignment 2.

Assume L is regular. Let j be the largest integer such that $\sum_{2 \leq k \leq j} a_k \leq a_1/2$. Clearly, there exists such a j since $j = 1$ satisfies the inequality. The following fact is not hard to verify (do it!)

Fact 1 Algorithm R will put a_1 in one processor, and $a_2, a_3, \cdots, a_j$ in another processor. Thus, the performance of R during the scheduling of the first j tasks is always 1.

If $j = n$, then we are done. Assume $j < n$. Let's see how R will schedule a_{j+1}. Let $S = \sum_{1 \leq k \leq j} a_k$. If a_{j+1} goes to the long processor, then the discrepancy is $E' = a_{j+1} + a_1 - \sum_{2 \leq k \leq j} a_k \geq a_{j+1} + (\sum_{1 \leq k \leq j} a_k)/3 \geq (a_{j+1} + S)/3$, where we have used (2) in the derivation. If a_{j+1} goes to the short processor, then the discrepancy is $E = |a_1 - \sum_{2 \leq k \leq j} a_k - a_{j+1}| \leq (a_{j+1} + S)/3$ (this takes a little calculation; do it!). It follows that there exists a p satisfying $pE + (1-p)E' = (S + a_{j+1})/3$. Therefore, in scheduling a_{j+1}, R uses (i) and the following is true:

Fact 2 After a_{j+1} is scheduled, we have $d_R(a_1, a_2, \cdots, a_{j+1}) = (\sum_{1 \leq i \leq n} a_i)/3$ (recall $d_R(F)$ is the expected discrepancy of the schedule produced by R for input list F).

We will prove that, for all $j+1 \leq m \leq n$,

$$d_R(a_1, a_2, \cdots, a_m) = (\sum_{1 \leq k \leq m} a_k)/3. \tag{3}$$

This implies (1), since $OPT(L) \geq (\sum_{1 \leq i \leq n} a_i)/2$, and hence $T_R(L) = (d_R(L) + \sum_{1 \leq i \leq n} a_i)/2 = (2/3)\sum_{1 \leq i \leq n} a_i \leq (4/3)OPT(L)$.

We prove (3) by induction on m. The base case $m = j+1$ is valid by Fact 2. For the inductive step, let $m > j+1$, and assume that all lesser values have been established.

Let $S = \sum_{1 \leq k \leq m-1} a_k$, and $\{b_i | 1 \leq i \leq 2^{m-1}\}$ be the set of discrepancies of all the possible scheduling of the first $m-1$ tasks, and p_i be the probability for the ith configuration to occur. By the induction hypothesis,

$$\sum_i p_i b_i = S/3.$$

The expected discrepancy after putting a_m in the long and short processors are respectively

$$\begin{aligned} E' &= \sum_i p_i(b_i + a_m), \\ E &= \sum_i p_i |b_i - a_m|. \end{aligned}$$

We will show that

$$E \leq (S + a_m)/3 \leq E'. \tag{4}$$

That would clearly imply the existence of a $0 \leq p \leq 1$ satisfying $pE + (1-p)E' = (S + a_m)/3 = (\sum_{1 \leq k \leq m} a_k)/3$. Since R uses such a p to schedule a_m, we have $d_R(a_1, a_2, \cdots, a_m) = pE + (1-p)E' = (\sum_{1 \leq k \leq m} a_k)/3$ which is (3), and thus would conclude the inductive step.

Clearly, $E' = a_m + \sum_i p_i b_i \geq a_m + S/3 \geq (S + a_m)/3$. Let $f(x) = \sum_i p_i |b_i - x|$. It is easy to verify that $f(0) = S/3$ and $f(S) = 2S/3$. Now here comes the pretty argument. Note that f is a piecewise linear function with a non-decreasing slope. Thus, the graph of $(x, f(x))$ between $x \in [0, S]$, as a convex continuous curve, must lie entirely below the line connecting $(0, S/3)$ and $(S, 2S/3)$. This means $f(x) \leq S/3 + x/3$ for $x \in [0, S]$, and hence $E = f(a_m) \leq S/3 + a_m/3$. This finishes the proof of (4).