Institut for Matematik og Datalogi
Syddansk Universitet

November 22, 2010
JFB

Introduction to Computer Science
E10 – Lecture 16

Lecture, November 22, 8:15–10, U37

We covered NP Problems from chapter 12 and sections 6.6 and 6.7 of chapter 6.

Lecture, November 25, 14:15–16, U71

Lene Monrad Favrholdt will lecture on on-line algorithms.

Lecture, November 29, 8:15–10, U37

We will cover the first five sectioins of chapter 7 in the textbook.

Announcement

There will be a ”pizza-meeting” for all students of IMADA on Monday, November 29th at 16:15 in room U49. At the meeting IMADA will give general information on the bachelor and candidate studies, and specific information on the elective courses planned for the next semester. The meeting will end with a free pizza, beer, and soft drink session.

Discussion section: December 8, 12:15-14, in U26

Discuss the following problems in groups of 3 to 4. (Think about these problems before coming to discussion section.)

Note that some of the material presented by Lene Monrad Favrholdt is described below, and some is available via Susanne Albers' lecture notes, on the course's homepage.

1. The List Scheduling algorithm (LS) places each new job on the machine with lowest load (currently). We showed in class that the algorithm has a competitve ratio of $2 - \frac{1}{m}$, where m is the number of machines. (Note the competitive ratio is the worst case ratio, over all possible input sequences, of the value the on-line algorithm achieves on the input sequence to the value the optimal off-line algorithm achieves on the same input sequence.) In order to show that the competitive ratio was this high, we showed that if the algorithm gets many small jobs ($m(m-1)$ of length 1) followed by one large job (a job of length m), LS will pack them so each machine has the same number of the small ones, so the last job will be placed on some machine that already has a large load. One might hope that holding a single machine free for such large jobs could help.

   Consider the following algorithm: Machine M_1 is kept mostly free for large jobs. The other machines are $M_2, M_3, ..., M_m$. Suppose you are trying to obtain a competitive ratio of C. When handling a job J, place it on the least loaded of machines $M_2, M_3, ..., M_m$ if placing it there does not bring the competitive ratio above C (if J was the last job). Otherwise, place it on machine M_1. Show that this strategy cannot lead to a better competitive ratio than $2 - \frac{1}{m}$.

2. If the number of machines is $m = 2$, then LS is 3/2-competitive and is optimal, i.e., no on-line algorithm has a better competitive ratio.

   Assume now, that one of the two machines is twice as fast as the other. Consider a variant of LS, LS2, which places a job J on the machine where it will finish first. This algorithm is 3/2-competitive. Show that this is optimal. (You can do it using a sequence with two jobs.)

   There are also other optimal algorithms for this problem. What other algorithm (very simple) also has competitive ratio 3/2?

3. Consider the dual bin-packing problem, which is the problem of putting (as many as possible) items into n bins, which all have size 1. First-Fit (FF), is the algorithm which puts an item (which arrived on-line) in the first bin in which it fits (no bin may be filled to more than 1). An *accommodation sequence* is a sequence which can fit in the n bins. Show that with an accommodating sequence, FF has a competitive ratio between $\frac{1}{2}$ and $\frac{4}{5}$, i.e., with such a sequence, FF packs at least

half of the items in the bins, and there exists such a sequence where FF only packs $\frac{4}{5}$ of the items. You can do this using only items of size $\frac{1}{2}$ and $\frac{1}{3}$.

Can you come further down than 4/5 by using items of size $\frac{1}{5}$, too?

4. In the classical bin-packing problem, you are not given a limit on how many bins there are and must pack all items. The goal is to use as few bins as possible. Show that FF has a competitive ratio between $\frac{5}{3}$ and 2. (Hint: you can show that it is at least $\frac{5}{3}$ using a sequence consisting of items of sizes $\frac{1}{7} + \frac{1}{2000}, \frac{1}{3} + \frac{1}{2000}, \frac{1}{2} + \frac{1}{2000}$.)

   Improve the $\frac{5}{3}$ to about 1.69 by using items of sizes $\frac{1}{43}+\frac{1}{8000}, \frac{1}{7}+\frac{1}{8000}, \frac{1}{3}+\frac{1}{8000}, \frac{1}{2} + \frac{1}{8000}$).

5. Show that FWF (the Flush-When-Full algorithm for paging) has a competitive ratio of at least k, where k is the cache size (show that there exist families of sequences, one sequence for each natural number, where asymptotically FWF faults k times as often as the optimal offline algorithm).

6. Show that FWF is never better than LRU.