Institut for Matematik og Datalogi
Syddansk Universitet

December 20, 2004
JFB

Introduction to Computer Science
E04 – Lecture 14

Lecture, December 6

Brian Vinter lectured on GRID computing. There are notes and program code on the course's Web page.

Lecture, December 13

Lene Monrad Favrholdt will lecture on on-line algorithms. At the end, I will lecture from chapter 11 in the textbook on NP-hardness.

Lecture, December 20

No lecture.

Discussion section: week 50

Discuss the following in groups of three or four.

1. Algoritmen List Scheduling (LS) placerer hvert job på den i øjeblikket mindst belastede maskine. Algoritmen har en competitive ratio på $2 - \frac{1}{m}$. Til forelæsningen så vi, at det skyldes, at en sekvens af mange små jobs vil blive fordelt ligeligt på de m maskiner, hvorefter et evt. langt job vil komme til at "rage langt op over" de små jobs. Man kunne derfor tro, at det ville hjælpe, hvis man holdt en enkelt maskine fri til store jobs.

   Betragt følgende algoritme, hvor maskinen M_1 holdes fri til store jobs. De andre maskiner kaldes $M_2, M_3, \ldots, M_m$. Lad C være den competitive ratio, man ønsker at opnå. For hvert job J undersøger man, om

man kan anbringe J på den mindst belastede maskine blandt $M_2, M_3, \ldots, M_m$, uden at det aktuelle makespan bliver mere end C gange så stort som det bedst mulige makespan. Hvis ja, gøres dette. Hvis nej, placeres J på M_1. Vis, at man med denne strategi ikke kan opnå en bedre competitive ratio end med LS.

2. Hvis der kun er to maskiner, har LS en competitive ratio på $\frac{3}{2}$, og det er optimalt, dvs. ingen online-algoritme har en bedre competitive ratio.

   Antag nu, at den ene maskine er dobbelt så hurtig som den anden. Til dette problem kan vi bruge en variant af LS, som placerer hvert enkelt job på den maskine, hvor det vil blive tidligst færdigt. Dvs. når man skal afgøre, hvilken maskine, der er mest belastet, tager man hensyn til, at den ene er dobbelt så hurtig som den anden. Denne algoritme er $\frac{3}{2}$-competitive. Vis, at det er optimalt. Vink: du behøver kun en sekvens med to jobs.

   Der findes også andre optimale algoritmer for dette problem. Kan du komme i tanker om en anden, meget simpel algoritme, som har competitive ratio $\frac{3}{2}$?

3. Nu ser vi på problemet, hvor det gælder om at pakke så mange elementer som muligt i n kasser, som alle har størrelse 1. First-Fit (FF) er algoritmen, som pakker hvert element i den første kasse, det passer i. Vis, at med accommodating sekvenser har FF en competitive ratio, som ligger mellem $\frac{1}{2}$ og $\frac{4}{5}$. Dvs. vis, at

   - hvis alle elementer i en sekvens kan være i de n bins, så kan FF pakke mindst halvdelen af elementerne i sekvensen, og
   - der findes en sekvens, hvor FF kun pakker $\frac{4}{5}$ af elementerne, selvom alle elementer i sekvensen godt kan pakkes i de n bins. Vink: Se på en sekvens bestående af elementer af to forskellige størrelser: $\frac{1}{2}$ og $\frac{1}{3}$. Kan du komme længere ned end $\frac{4}{5}$ ved også at bruge elementer af størrelse $\frac{1}{5}$?

4. Antag nu, at opgaven er at pakke samtlige elementer i så få kasser af størrelse 1 som muligt. Vis, at FF har en competitive ratio, som ligger mellem $\frac{5}{3}$ og 2. Vink: til at vise, at FFs competitive ratio er mindst $\frac{5}{3}$ kan du f.eks. bruge en sekvens bestående af elementer af størrelse $\frac{1}{7} + \frac{1}{2000}$, $\frac{1}{3} + \frac{1}{2000}$ og $\frac{1}{2} + \frac{1}{2000}$.

5. Discuss how the Shortest Path Problem (defined in lecture – given two cities, find the shortest path between them) and the Traveling Salesman Problem differ.

6. Discuss issues 1, 5, 8 on pages 493–494.

Last assignment due 9:00, Monday, December 20

Late assignments will not be accepted. Working together is not allowed. (You may write this either in English or Danish, but write clearly if you do it by hand.) Explain your answers.

1. Consider problem 4 above, but use items of size $\frac{1}{43} + \frac{1}{8000}$, $\frac{1}{7} + \frac{1}{8000}$, $\frac{1}{3} + \frac{1}{8000}$ and $\frac{1}{2} + \frac{1}{8000}$. You should now be able to get a ratio of at least 1.69. How? 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.

2. Problem 45 on page 492. For this problem, specify the algorithm you use so that it works for lists of any length n. Give the complexity in terms of n.