Institut for Matematik og Datalogi
Syddansk Universitet

May 13, 2019
JFB

On-Line Algorithms – F19 – Lecture 17

Lecture, May 6

We finished section 5 of the article on bin packing with advice, including a very brief review of (introduction to) reductions. We then started on the article "The Advice Complexity of a Class of Hard Online Problems", Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Jesper W. Mikkelsen. *Theory of Computing Systems*, 61(4): 1128-1177, 2017. The publication is available through the course's homepage. We covered Definition 3 for the minimum asymmetric string guessing problem with known history, section 3.1 on covering designs, Definition 6 of AOC, Theorem 3.11 showing how to use covering designs as advice for a minimization problem, Definition 7 of AOC-Completeness, and the Vertex Cover problem as an example.

Lecture, May 14

We will finish the article on the advice complexity of a class of hard online problems. Then we will return to chapter 12 in the textbook, starting with section 12.2.2.

Lecture, May 21

We will continue on chapter 12 in the textbook.

Problems for May 15

1. The first two problems were not finished from the previous note.

2. Go through the reduction from Binary Separation to Bin Packing with the sizes $\langle \frac{7}{16}, (S, \frac{15}{32}), (L, \frac{29}{64}), (S, \frac{59}{128}), S \rangle$ as input. Let $\varepsilon = \frac{1}{256}$. You may assume that your algorithm for Bin Packing is First-Fit.

3. For makespan in the identical machines case, where might advice be useful? (Note, this is a vague, open sort of question.)

4. Consider the following advice for the makespan scheduling problem on N identical machines: The index of one machine which should not be used, except for items with load at least $N/2$.

   (a) Specify how many bits of advice are used, and define an algorithm to use it effectively (specify what the algorithm should do with jobs with different loads).

   (b) Show the effect this advice would have on the sequence used in the lower bound proof of Theorem 12.1 in the textbook. How would your algorithm schedule this sequence and what would the performance ratio be, compared to OPT? (Note that if your algorithm does not do better on this sequence than GREEDY does, you should redefine your algorithm.)

   (c) Give an example input sequence showing that the performance ratio compared to OPT is still at least $2 - \frac{1}{N}$.

   (d) Give an example input sequence showing that the performance ratio compared to OPT is at least 2. Can the performance ratio be even higher?

5. Consider the graph $G = (V, E)$, where $V = \{v_1, v_1, v_3, v_4, v_5, v_6, v_7\}$ and $E = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_1, v_5), (v_1, v_6), (v_7, v_2), (v_7, v_3), (v_7, v_4), (v_7, v_5), (v_7, v_6)\}$. Find an optimal vertex cover.

   Find a $(7, 5, 2)$-covering design of size 3.

   Show how to use this covering design for the graph G. What is your performance ratio?

6. Show that Maximum Matching is in AOC, but Simple Knapsack is not.

7. Show that the Independent Set problem is NP-hard. For Independent Set, the input is a graph, $G = (V, E)$, and the output is a subset $I \subseteq V$ such that for any two vertices $u, v \in I$, there is no edge between them in G, i.e., $(u, v) \notin E$. Such a set I is called an *independent set*. The goal is to output a maximum size independent set. Reduce from 3-SAT.

   Show that the Vertex Cover problem is NP-hard. Reduce from Independent Set.

8. Show that the Cycle Finding problem can be solved in polynomial time. The input is a graph, and the goal is to find a smallest subset of the vertices containing a cycle.

9. Go through the proofs of Theorems 12.2 and 12.3 for $N = 4$.

10. Consider remark 12.1 on page 208. What is meant here? Why is there no problem if the loads can be greater than 1? (Do not try to prove the desired result for loads of at most 1.)

11. Prove that makespan on related machines is NP-hard. Prove the same for restricted machines. Prove that the classical bin packing problem is NP-hard.

12. In the algorithm SLOWFIT$_\Lambda$, what if some other value than 2 is used in $i = \arg\min_k\{l_j(k) + r_{j+1}(k) \leq 2\Lambda\}$? How does Theorem 12.4 change?

13. For makespan on identical machines, show how to use the "slowfit" idea for an algorithm which is 2-competitive when the optimal value is known.