Exercises on Approximation Algorithms

Exercise 6.5 [LN]: Proof by contradiction. Suppose there is an instance of KNAPSACK for which this is not true. Suppose that for this instance, after ordering on non-increasing p_j/w_j-ratio, the items $1, 2, \ldots, k-1$ fit entirely in the knapsack and item k fits only partially (where partially may even be a 0-fraction); i.e., the solution is represented by the vector x with $x_i = 1$, $0 \leq i \leq k-1$, $x_k = B - \frac{\sum_{i=1}^{k-1} w_i}{w_k}$ and $x_i = 0$, otherwise.

If this is not the optimal solution, then there exists a better solution. From all of these take the one x' that diverges least from x in L_1-distance ($\sum_{i=1}^{n} |x'_i - x_i|$). In this solution there must be some j, $0 \leq j \leq k$ with $x'_j < x_j$. Then it must be that for some $i > j$ and $i \geq k$, $x'_i > x_i$, since in an optimal LP-solution the capacity of the knapsack will always be fully utilized. Now we will take a bit from the x'_i and reallocate the space becoming available to raise x'_j, and find out that this improves the solution. So, we set x'_i to $x'_i - \epsilon$, by which we sacrifice ϵp_i in objective value. The space becoming available is ϵw_i. Thus we use this to raise x'_j with $\epsilon \frac{w_i}{w_j}$, gaining in objective value $\epsilon \frac{w_i}{w_j} p_j$.

$$\epsilon \frac{w_i}{w_j} p_j - \epsilon \underset{=}{\frac{p_i}{}} \epsilon w_i \left(\frac{p_j}{w_j} - \frac{p_i}{w_i} \right) \geq 0,$$

given that $i > j$ and the items have been ordered on non-increasing ratio. Thus, either we have a solution with better value, which is a contradiction or we have solution with equal value, which is closer to x in L_1-distance, which is a contradiction. □

Exercise 6.6. [LN]: An example with which the worst-case bound of 1/2 can be approximated arbitrarily closely is the following one: Take $n = 3$ with $p_1 = w_1 = 1 + \frac{1}{2}B$, $p_2 = w_2 = p_3 = w_3 = \frac{1}{2}B$, and B the capacity. The performance of Approximation algorithm 3 amounts to $(\frac{1}{2}B + 1)/b$, which goes to 1/2 if B goes to infinity.

Exercise 6.7. [LN]: See the proof of Thm. 17.12 in [PS].

Exercise 6.9. [LN]: Take the instance with $2m - 1$ jobs, given in the order:

- Jobs $j = 1, \ldots, m-1$ have all $p_j = m - 1$;
- Jobs $j = m, \ldots, 2(m-1)$ have all $p_j = 1$;
- Job $2m - 1$ has $p_{2m-1} = m$.

List Scheduling will create makespan $2m - 1$, whereas optimal clearly is m.

Exercise 6.11. [LN]: Notice that the flow time of job j is $F_j = C_j - r_j$. Hence, $\sum F_j = \sum C_j - \sum r_j$ and since $\sum r_j$ is fixed minimising $\sum F_j$ is equal to minimising $\sum C_j$.

Exercise 6.12. [LN]: The single machine version is solved by the *First-In-First-Out rule*, processing always the job with the earliest release date. The 2 machine case is NP-hard by a reduction from MAKESPAN. Give all jobs in a *Makespan* instance release time 0 and keep the constant K. With all jobs having release time 0, $\max F_j$ is simply equal to $\max C_j$, the makespan.

Exercise 17.1 [PS]: Consider the graph that consists of a single odd circuit with $2n+1$ vertices. In this case the vertex cover approximation algorithm will select all vertices, whereas $n+1$ would have sufficed. Still, $\frac{2n+1}{n+1} \leq 2$. However, this vertex cover according to the transformation we used between VERTEX COVER and INDEPENDENT SET would produce an empty independent set, which is certainly an independent set, but the optimal one has n vertices. Argue similarly for CLIQUE.

Exercise 17.3 [PS]: **a**) Easy reduction from HAMILTON CIRCUIT.

b) Again Christofides' TREE+MATCHING-algorithm works, where now no shortcuts are made, simply because they do not exist. Prove the ratio as in the TSP.

Exercise 17.4.a [PS]: I consider this problem as a 2-machine scheduling problem minimising makespan. I call the algorithm using k here $A(k)$. We distinguish two cases. First if one of the k largest jobs completes last. Then we have the optimal solution. Otherwise one of the $n-k$ smaller jobs completes last, say job ℓ with processing time p_ℓ. Clearly, $p_\ell \leq p_k$, the processing time of the k-th largest job. Hence, using the same upper bound and lower bound as used for the analysis of List Scheduling we have that

$$C_\ell = S_\ell + p_\ell \leq \frac{1}{2}\sum_{j=1}^{n} p_j + \frac{1}{2}p_\ell$$

Hence,

$$\begin{aligned}
\frac{Z^{A(k)}}{Z^{OPT}} &\leq 1 + \frac{\frac{1}{2}p_\ell}{\frac{1}{2}\sum_{j=1}^{n} p_j} \\
&= 1 + \frac{p_\ell}{\sum_{j=1}^{n} p_j} \\
&\leq 1 + \frac{p_\ell}{\sum_{j=1}^{k} p_j + p_\ell} \\
&\leq 1 + \frac{p_\ell}{kp_\ell + p_\ell} \\
&\leq 1 + \frac{1}{k+1}.
\end{aligned}$$

Hence, $A(k)$ is a $1+\frac{1}{k+1}$ algorithm.

b) To obtain a ratio of $1+\epsilon$ we need to choose k such that $\epsilon \leq \frac{1}{k+1}$ i.e. $k+1 \geq \frac{1}{\epsilon}$ or $k \geq \frac{1}{\epsilon} - 1$. For ϵ we call the smallest such value $k(\epsilon)$: $k(\epsilon) = \frac{1}{\epsilon} - 1$. Then

the algorithm $A(k(\epsilon))$ yields approximation ratio $1+\epsilon$ and its running time is $O(2^{1/\epsilon}+n)$. Therefore for all possible values of ϵ the algorithms $A(k(\epsilon))$ form a PTAS.