COM S 783 / OR&IE 634 Approximation Algorithms	Fall Semester, 2003

Lecture 4: Sept 10, 2003

Lecturer: Yuval Rabani *Scribe: Alexa Sharp*

1 Knapsack

Recall the Knapsack problem, where you are given n items with sizes $s_i \geq 0$, profits $p_i \geq 0$ and a knapsack size B. The goal is to find a maximum profit subset of items such that the combined size of the subset is no more than B. In the previous lecture we presented a few different algorithms, the last of which used, as a subroutine, a pseudo-polynomial time algorithm PP that solves Knapsack exactly in time $O(n^2 P \log(SB))$, where P is the maximum profit, S the maximum size and B the knapsack size.

The algorithm consists of multiple parts. First we define the algorithm A_k, which takes as input a Knapsack instance I and creates new input I_k by scaling the profits down by a factor of k. This yields new profits $p_i' = \lfloor p_i/k \rfloor$. It then runs PP on I_k, and returns this set of items as a solution to the input I (in particular, the profit returned by A_k is sum of the p_is of the items returned by PP). Next we define the algorithm A_ϵ which sorts the items in non-increasing order by p_i/s_i, computes a value k, then runs A_k. The computed value of k is

$$k = \frac{V}{2(1+\frac{1}{\epsilon})n}$$

where $V = p_1 + p_2 + \cdots + p_{i+1}$ and i is the largest index such that $s_1 + s_2 + \cdots + s_i \leq B$.

Theorem 1.1 [1] *The algorithm A_ϵ runs in time $O(\frac{1}{\epsilon}n^3 \log(SB))$ and is a $(\frac{1}{1+\epsilon})$-approximation.*

Proof: Our definition of k ensures that for all i, $0 \leq \lfloor \frac{p_i}{k} \rfloor \leq 2(1+\frac{1}{\epsilon})n$ and yields a maximum profit $P \leq 2(1+\frac{1}{\epsilon})n$, which means the algorithm A_ϵ runs in time $O(\frac{1}{\epsilon}n^3 \log(SB))$ as required. Thus it only remains to prove the approximation bound.

To this end, let us first note that

$$\begin{aligned} p_1 + p_2 + \ldots + p_i &\leq OPT \\ p_{i+1} \leq OPT & \end{aligned}$$

which together imply that $V \leq 2 \cdot OPT$. Let $S_i = s_1 + s_2 + \cdots + s_i$. Then $S_i \leq B$ but $s_{i+1} > B - S_i$ because the $i+1^{st}$ dense item does not fit. An important observation is that up to this point of filling the knapsack (up to size S_i), optimal cannot be doing any better than this greedy algorithm. We have taken the densest items available and placed them in the knapsack. So, at best, optimal could take all these i items, and then fill the remainder of the knapsack with the density of the $i+1^{st}$ item (that didn't fit in the knapsack). I.e. $OPT \leq p_1 + p_2 + \cdots + p_i + (B - S_i)\frac{p_{i+1}}{s_{i+1}} < p_1 + p_2 + \cdots + p_i + s_{i+1}\frac{p_{i+1}}{s_{i+1}} = V$. Thus

$$OPT \leq V \leq 2 \cdot OPT$$

Also note that $A_\epsilon(I) \geq k \cdot OPT(I_k)$, since $OPT(I_k)$ returns the scaled down profits, which even when multiplied by k, may not recover the original profits used by $A_\epsilon(I)$. Let $i_1, i_2, \ldots, i_m$ be the items selected by $OPT(I)$, with profits $p_{i_1}, p_{i_2}, \ldots, p_{i_m}$. Denote the profits of the items in I_k by $p'_{i_j} = \lfloor p_{i_j}/k \rfloor$. Then

$$\begin{aligned} OPT(I) - k \cdot OPT(I_k) &\overset{(*)}{\leq} p_{i_1} + p_{i_2} + \cdots p_{i_m} - k(p'_{i_1} + p'_{i_2} + \cdots p'_{i_m}) \\ &= \sum_{j=1}^{m} (p_{i_j} - k\lfloor \frac{p_{i_j}}{k} \rfloor) \leq km \leq kn \end{aligned}$$

where inequality $(*)$ follows from the fact that the optimal solution for I is a feasible solution for I_k. Therefore

$$\begin{aligned} & OPT(I) - A_\epsilon(I) \leq kn = \frac{V}{2(1+\frac{1}{\epsilon})} \leq \frac{OPT(I)}{1+\frac{1}{\epsilon}} \\ \Rightarrow\ & OPT(I)(1+\frac{1}{\epsilon}) - A_\epsilon(I)(1+\frac{1}{\epsilon}) \leq OPT(I) \\ \Rightarrow\ & OPT(I)\frac{1}{\epsilon} \leq A_\epsilon(I)(1+\frac{1}{\epsilon}) \\ \Rightarrow\ & \frac{OPT(I)}{A_\epsilon(I)} \leq \frac{1+\frac{1}{\epsilon}}{\frac{1}{\epsilon}} \Rightarrow \frac{OPT(I)}{A_\epsilon(I)} \leq 1+\epsilon \end{aligned}$$

□

The analysis thus far has relied on the existence of a pseudo-polynomial algorithm PP that solves Knapsack exactly. Before continuing, we should guarantee such an algorithm exists.

Claim 1.2 *There is an algorithm that solves Knapsack exactly in time $O(n^2 P \log(SB))$, where P is the maximum profit, S is the maximum size and B is the knapsack bound.*

Proof: We will use a dynamic programming algorithm to compute the optimal solution. Define the variable $X_{i,j}$ to be the feasible solution of total profit j that packs items selected only from the set $\{1, \ldots, i\}$. If no such solution exists we mark the variable with a $*$, and if multiple solutions exist we select the solution with the smallest total size (doing so is justified in the next lemma). In particular,

$$X_{i,j} = \begin{cases} * & \text{if no such solution exists} \\ \text{sol'n of smallest total size} & \text{otherwise} \end{cases}$$

The final desired solution is $\max_{\{j | X_{n,j} \neq *\}} X_{n,j}$.

In order to calculate an arbitrary $X_{i,j}$, we observe that we either take item i to be in our optimal solution, or we do not. Thus for $1 \leq i \leq n, 0 \leq j \leq nP$, we have the recurrence

$$X_{i,j} = \begin{cases} \text{smallest non-}*\text{ sol'n among } X_{i-1,j} \text{ and } X_{i-1,j-p_i} \cup \{i\} & \text{if } \sum_{l \in X_{i-1,j-p_i}} s_l \leq B - s_i \\ X_{i-1,j} & \text{otherwise} \end{cases}$$

For base cases we have

$$\begin{aligned} X_{i,0} &= \emptyset \qquad \forall i \\ X_{i,j} &= \begin{cases} \emptyset & \text{if } j = 0 \\ \{1\} & \text{if } j = p_1 \\ * & \text{otherwise} \end{cases} \end{aligned}$$

Putting the recurrence and base cases together, our algorithm is

Algorithm 1.3 *Set the base cases according to the formula. Then for each $i = 2, \ldots, n$ and each $j = 1, \ldots, nP$, set X_{ij} according to the recurrence relation.*

There are n^2P calls to the recurrence relation assignment, which in turn makes a constant number of operations on integers bounded above by S and B. Therefore, the running time of this algorithm is $O(n^2P \log(SB))$.

The correctness of the algorithm follows from the observations above and the following lemma, which we will not prove.

Lemma 1.4 *Let $X, Y \subseteq \{1, \ldots, i\}$ such that $profit(X) = profit(Y) = j$ and $size(X) \leq size(Y)$. Then $\forall Z \subseteq \{i+1, \ldots, n\}$, $size(X \cup Z) \leq size(Y \cup Z)$.*

□

2 Combinatorial Bounds

Combinatorial bounds are difficult to classify, but they can generally be thought of as bounds that rely on simple combinatorial structures to bound solutions. We have already seen a combinatorial bound in the metric travelling salesman problem.

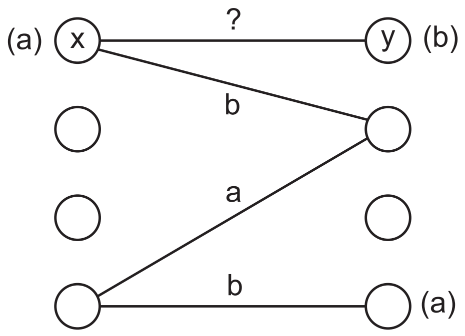
Definition 2.1 *Let G be an undirected loopless graph. The* chromatic index *$\chi'(G)$ is the minimum number of colours needed to colour the edges of G such that no two adjacent edges share a colour.*

We define $\Delta = \Delta(G)$ to be the maximum degree of any vertex in G. Obviously, $\chi'(G) \geq \Delta$. The question we are going to explore is, how good is this bound?

Theorem 2.2 (Konig, 1916 [2]) *If $G = (V, E)$ is bipartite then $\chi'(G) = \Delta$.*

Proof: (By induction on $|E|$). The base case of no edges is trivial. For the inductive step, suppose we remove an edge $e = (x, y)$ from G to create a modified graph G'. By induction we can colour the remaining graph with Δ colours. Our goal is to show we can colour the edge e with one of these Δ colours and still have a valid colouring.

We know that both x and y use at most $\Delta - 1$ colours each since their degrees in G' are at most $\Delta - 1$. Let α be a colour missing at x and let β be a colour missing at y. If $\alpha = \beta$ then we can colour e with this colour and we are done. Otherwise, $\alpha \neq \beta$.



If β is missing at x then we colour e with β (similarly if α is missing we colour e with α). Thus there is only a problem if a β-coloured edge leaves x (and an α-coloured edge leaves y). In this case, consider a *β-α-alternating path* starting from x. This path cannot end at x because all cycles in bipartite graphs have even lengh, which would imply that this path would end on an α-coloured edge. This would contradict α being missing at x. Similarly, the alternating path cannot end at y or else there would be a β-coloured edge going into y. Therefore the path does not end at x or y, so we can swap the colours α and β along this alternating path. This is still a valid colouring since at each node along the path we swap both colours (except the two endpoints, where only one of the colours appear) . After the swap β does not touch x or y, leaving us free to colour the edge e with colour β. $\square$

For non-bipartite graphs this result no longer holds. For example, consider the triangle graph K_3. The maximum degree $\Delta(K_3)$ is 2, but the chromatic index $\chi'(K_3)$ is 3.

Theorem 2.3 (Shannon's Theorem, 1949 [3]) $\chi'(G) \leq \lfloor \frac{3}{2}\Delta \rfloor$. *Furthermore, there is a polynomial time algorithm to colour G with this many colours.*

The proof of Shannon's Theorem will be presented in the following lecture. Here we just have a few observations. First, this is thought to possibly be the first approximation algorithm. Second, for even Δ, $\frac{3}{2}\Delta$ is an integer and the graph K_3 gives a tight lower bound for this approximation. For odd Δ, the Peterson graph, with $\Delta = 3$ and $\chi' = 4$ gives a lower bound of $\frac{4}{3}$ for the approximation.

References

[1] O.H. Ibarra and C.E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22:463-468, 1975.

[2] D. Konig. Uber Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Math. Ann.*, 77, 453-465, 1916.

[3] C.E. Shannon. A theorem on colouring lines of a network. *J. Math. Phys.*, 28:148-151, 1949.