IOE 691: Approximation Algorithms — Date: 02/01/2017

Lecture Notes: Dynamic Programming (Knapsack and Bin Packing)

Instructor: Viswanath Nagarajan — Scribe: Fatemeh Navidi

1 Knapsack Problem

Recall the knapsack problem from last lecture:

Definition 1.1 *The input is a bound B and a set of n items, where item i has size s_i and value v_i. We want to find a subset of items $S \subseteq [n]$ such that it maximizes $\sum_{i \in S} v_i$ subject to $\sum_{i \in S} s_i \leq B$.*

We assume that there is no item with size greater than B, because we cannot choose such items anyway. Now, recall that we stated an exact algorithm for Knapsack problem when values are integer with running time $poly(n, \sum_{i=1}^{n} v_i)$. We refer to this algorithm as $ExactKnapsack(s_1, ..., s_n, v_1, ..., v_n, B)$.

Here we provide an FPTAS for the knapsack problem.

Definition 1.2 *An FPTAS for problem P finds for any $\epsilon > 0$, a $(1-\epsilon)$-approximate solution in time $poly(N, \frac{1}{\epsilon})$, where N is the input size of P.*

This is stronger than a PTAS which may have an exponential (or worse) dependence on $\frac{1}{\epsilon}$.

Algorithm 1 $(s_1, ..., s_n, v_1, ..., v_n, B, \epsilon)$

1: $M \leftarrow \max_i v_i$
2: $v'_i \leftarrow \lfloor \frac{v_i}{\epsilon M/n} \rfloor$
3: $A \leftarrow ExactKnapsack(s_1, ..., s_n, v'_1, ..., v'_n, B)$
4: **return** solution A

Theorem 1.1 *Algorithm 1 is an FPTAS for the knapsack problem.*

Proof: Note that the new values v' are integer and $\sum_{i=1}^{n} v'_i = \sum_{i=1}^{n} \lfloor \frac{v_i}{\epsilon M/n} \rfloor \leq \sum_{i=1}^{n} \frac{n}{\epsilon} \leq \frac{n^2}{\epsilon}$. So, the running time of $ExactKnapsack$ is $poly(\frac{n^2}{\epsilon}) = poly(n, \frac{1}{\epsilon})$, which means the running time of Algorithm1 is also $poly(n, \frac{1}{\epsilon})$. Now, we only need to show that if OPT denotes the optimal solution of Knapsack problem for the instance with sizes s_i and values v_i, then our solution A has value at least $(1-\epsilon)OPT$.

We will show that the optimal solution to the knapsack instance with values v' has value $\overline{OPT} \geq (1-\epsilon)\frac{n}{\epsilon M} \cdot OPT$. Since A is an optimal solution to this instance, we would obtain $\sum_{i \in A} v_i \geq \frac{\epsilon M}{n} \sum_{i \in A} v'_i = \frac{\epsilon M}{n} \overline{OPT} \geq (1-\epsilon)OPT$.

Let S be the subset of items in the optimal solution under values $\{v_i\}_{i \in [n]}$. We have:

$$OPT = \sum_{i \in S} v_i = \sum_{i \in S} \frac{v_i}{\frac{\epsilon M}{n}} \frac{\epsilon M}{n} \leq \sum_{i \in S} (v'_i + 1) \frac{\epsilon M}{n} = (\sum_{i \in S} v'_i) \frac{\epsilon M}{n} + \epsilon M \leq \frac{\epsilon M}{n} \overline{OPT} + \epsilon OPT$$

Where the last inequality is true because the single item with maximum value is a feasible solution. Now by rearranging inequalities we have $\overline{OPT} \geq (1-\epsilon)\frac{n}{\epsilon M} \cdot OPT$ as needed. ■

2 Bin Packing Problem

Definition 2.1 *In Bin Packing problem we have n items with sizes $s_i \in [0,1]$ and we want to pack them into bins with capacity 1. The goal is to minimize the number of bins used to pack all items.*

Theorem 2.1 *It is NP-hard to approximate the Bin Packing problem to a factor better than $\frac{3}{2}$ under assumption of $P \neq NP$.*

Proof: We show this by a reduction from the Subset Sum problem which is known to be NP-complete. In the Subset-Sum problem we are given a set of n numbers $a_1, a_2, ..., a_n$ and the goal is to specify whether there exists $T \subseteq [n]$ such that $\sum_{i\in T} a_i = \sum_{i\notin T} a_i$.

Consider an arbitrary instance $a_1, a_2, ..., a_n$ for Subset Sum problem. We make an instance for Bin Packing Problem with n items of sizes $s_i = \frac{2a_i}{\sum_{j\in[n]} a_j}$. Note that $\sum_{i=1}^{n} s_i = 2$, so we need at least two bins for the bin-packing instance.

If the subset-sum instance was a yes-instance then there is a $T \subseteq [n]$ such that $\sum_{i\in T} a_i = \sum_{i\notin T} a_i$, i.e. $\sum_{i\in T} s_i = \sum_{i\notin T} s_i$. This means $\sum_{i\in T} s_i = \sum_{i\notin T} s_i = 1$. So the optimal solution for such instance is 2.

If there is a solution to the bin-packing instance with value $k < 3$, it means that we need at most two bins. But we know that $\sum_{i\in[n]} s_i = 2$. So the full capacity of each bin has been used. Let T be the subset of items in the first bin, then we have $\sum_{i\in T} s_i = 1 = \sum_{i\notin T} s_i$. This means that $\sum_{i\in T} a_i = \sum_{i\notin T} a_i$ and the subset-sum instance is a yes-instance.

So we have shown that we have a yes-instance for the Subset Sum problem if and only if the optimal bin-packing value is ≤ 2. This shows that a better than $\frac{3}{2}$-approximation is NP-hard. ■

However this hardness only applies to instances of small optimal value. Below we discuss an approximation algorithm which is almost a PTAS.

Lemma 2.1 *Fix any constants $\epsilon, c > 0$. Consider any instance of Bin Packing that satisfies:*

1. *For every size s_i we have $s_i \geq \epsilon$.*
2. *Number of distinct sizes is at most c.*

Then there is an exact algorithm with running time poly$(n^{c^{(1/\epsilon)}})$.

Proof: We define the pattern of a bin as a vector of size at most c, such that its i-th entry denotes the number of i-th size items in this bin. By assumption 1 we can have at most $1/\epsilon$ items in a bin, and by assumption 1 there are at most c choices for each item size. So we can have at most $p = c^{1/\epsilon}$ different patterns. We order all the possible patterns from 1 to p and define $m_1, m_2, ..., m_p$ as the number of bins we fill with each of these patterns and $n_{i,j}$ as the number of i-th size items in pattern j. Then for each i, we have $0 \leq m_i \leq n$. Now, we enumerate all possibilities for $m_1, m_2, ..., m_p$ which are at most $(n+1)^p$ in total. Then from these, we will omit the infeasible solutions, which are the ones in which for any i, $\sum_{j=1}^{p} n_{ij} m_j$ is less than the input number of items of i-th size. Among the feasible solutions, we choose the one that minimizes $\sum_{j=1}^{p} m_j$ which is the number of bins. The runtime is $\mathcal{O}((n+1)^p) = \mathcal{O}(n^{c^{1/\epsilon}})$ steps. ■

We refer to this algorithm as $ExactBinPacking(s_1, ..., s_n)$.

Now, we present the following algorithm for the general instance of Bin Packing problem:

Algorithm 2 $(s_1, ..., s_n, \epsilon)$

1: Remove items with size smaller than ϵ
2: Sort and relabel items such that $s_1 \geq s_2 \geq ... \geq s_{n'}$
3: $K \leftarrow \epsilon \sum_{i=1}^{n'} s_i$
4: Put each K consecutive items in a group, starting with the largest ones
5: Round up each s_i to the maximum size in its group and call it $\overline{s_i}$
6: Run $ExactBinPacking(\overline{s_1}, ..., \overline{s_n})$ to get a solution with L bins.
7: Put the removed items with weight smaller than ϵ in these L bins and at any time put an item in a new bin if and only if it cannot fit in any of the occupied bins.
8: **return** set of occupied bins

Note that the number of distinct sizes in $\bar{s}$ is $\lceil n'/K \rceil = \lceil n'/\epsilon \sum_{i=1}^{n'} s_i \rceil \leq \frac{1}{\epsilon^2}$ as each $s_i > \epsilon$. So, the two assumptions in Lemma 2.1 hold for the instance $\overline{s_1}, ..., \overline{s_{n'}}$, and $ExactBinPacking$ runs in $\mathcal{O}(n^{(1/\epsilon^2)^{1/\epsilon}})$ which is polynomial for fixed $\epsilon > 0$.

Theorem 2.2 *If ALG denotes the solution to the above algorithm then we have* $ALG \leq \frac{OPT}{1-\epsilon} + 1$.

Proof: Let $\overline{OPT}$ be the optimal solution to the instance after we have removed the items with weight smaller than ϵ. Then we claim:

$$L \leq \overline{OPT} + K$$

First we show why this inequality holds. We construct a feasible solution for the instance with rounded up sizes $\bar{s}$ from the solution corresponding to $\overline{OPT}$ by replacing each item with size s_i with an item of round up size $\overline{s_{i+K}}$ if such item exist, and putting the first K items with rounded up sizes in K separate bins. This is feasible, because due to our construction $\overline{s_{i+K}} \leq s_i$. Furthermore, since L is the optimal value to the instance with rounded up sizes, its value should not exceed to value of any feasible solution, and this concludes the inequality.

Now, note that since we have removed a bunch of items to obtain $\overline{OPT}$ we have $\overline{OPT} \leq OPT$. Moreover, since bins have capacity 1, we have $\sum_{i=1}^{n} s_i \leq OPT$ and we can write:

$$L \leq OPT + K = OPT + \epsilon \sum_{i=1}^{n'} s_i \leq OPT + \epsilon \sum_{i=1}^{n} s_i \leq OPT(1+\epsilon)$$

Then when we are adding the items with size smaller than ϵ, we either remain with L bins and won't open any new bin or we open other bins. In the second case when the last bin is being used, at least $1-\epsilon$ of all the previous $ALG - 1$ bins should have been used, otherwise we didn't need to open the last bin. So in this case we have:

$$(ALG-1)(1-\epsilon) \leq \sum_{i=1}^{n} s_i \implies (ALG-1)(1-\epsilon) \leq OPT$$

$$ALG \leq \frac{OPT}{1-\epsilon} + 1$$

But this is in the case that we had to open a new bin. If not, we will have $ALG = L$. So in general we have:

$$ALG \leq \max\{L, \frac{OPT}{1-\epsilon} + 1\} \leq \max\{OPT(1+\epsilon), \frac{OPT}{1-\epsilon} + 1\} = \frac{OPT}{1-\epsilon} + 1$$

■