CMPUT 675: Approximation Algorithms **Fall 2014**

Lecture 6 (Sep 15): The Set Cover and Knapsack Problems

Lecturer: Zachary Friggstad *Scribe: Bradley Hauer*

6.1 The Set Cover Problem

Definition 1 *In the* Set Cover *problem, we are given X, a set of items, a set $\mathcal{S}$ of subsets of X, and a cost function c such that $c(S) \geq 0$ for all $S \in \mathcal{S}$. The goal is to find a minimum-cost set of sets $\mathcal{C} \subseteq \mathcal{S}$ which covers X (i.e. the union of all the sets in $\mathcal{C}$ is equal to X).*

Algorithm 1 Greedy Set Cover Algorithm (Annotated with Marginal Costs)

Input: A set $\mathcal{S}$ of subsets of X, with c, a non-negative cost function on $\mathcal{S}$
Output: A subset $\mathcal{C}$ of $\mathcal{S}$ which covers X.

$\mathcal{C} \leftarrow \emptyset$
$Y \leftarrow \emptyset$
$z_i \leftarrow 0$ for all $i \in X$
while $Y \neq X$ **do**
  Let S be the element of $\mathcal{S}$ minimizing $\frac{c(S)}{|S-Y|}$ (note that this function gives the *density* of S)
  $z_i \leftarrow \frac{c(S)}{|S-Y|}$ for all $i \in S - Y$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{S\}$
  $Y \leftarrow Y \cup S$
end while
return $\mathcal{C}$

Note: recording the z_i values in the loop has no effect on the output; they are only used in the analysis below.

Before proving the main result, we establish a claim first.

Claim 1 *For each $S \in \mathcal{S}$: $\sum_{i \in S} z_i \leq H_{|S|} \cdot c(S)$*

Proof. Let $i_1, i_2, \ldots, i_{|S|}$ be the order the items of S were covered. Just before i_j is covered, the density of S (recall: density of S is equal to $\frac{c(S)}{|S-Y|}$) is less than or equal to $\frac{c(S)}{|S|-j+1}$ since items $i_j, i_{j+1}, \ldots, i_{|S|}$ were not yet covered by $\mathcal{C}$ at this time. Since i_j is covered by the minimum density set at this time, $z_{i_j} \leq \frac{c(S)}{|S|-j+1}$. Therefore,

$$\sum_{j=1}^{|S|} z_j \leq c(S) \cdot \sum_{j=1}^{|S|} \frac{1}{|S| - j + 1} = c(S) \cdot H_{|S|}.$$

■

Theorem 1 *This in an H_k-approximation, where:*

1. $k = \max_{S \in \mathcal{S}} |S|$
2. $H_k = \sum_{a=1}^{k} \frac{1}{a} = \ln(k) + O(1)$

Proof. Say $\mathcal{C}$ is returned by our algorithm and say $\mathcal{C}^*$ is an optimal solution, with cost OPT.

Notice that each $i \in X$ has z_i assigned some value in the loop exactly once. Namely, in the iteration when i is first covered. Also note that the total z_i value assigned to items in a single loop is exactly the cost of the set S that is added to $\mathcal{C}$ in that iteration.

We bound the cost of $\mathcal{C}$ as follows.

$$\begin{aligned}
\text{cost}(\mathcal{C}) &= \sum_{i \in X} z_i \cdot 1 \\
&\leq \sum_{i \in X} z_i \sum_{S \in \mathcal{C}^* \text{s.t.} i \in S} 1 \quad \text{(Since each item } i \text{ is covered by } \mathcal{C}^*) \\
&= \sum_{S \in \mathcal{C}^*} \sum_{i \in S} z_i \quad \text{(Rearranging)} \\
&\leq \sum_{S \in \mathcal{C}^*} H_{|S|} \cdot c(S) \quad \text{(By Claim 1)} \\
&\leq H_k \cdot OPT
\end{aligned}$$

■

If this analysis seems difficult to intuit, be assured that it will make more sense once we cover linear programming duality. The analysis of the H_k approximation ratio of this greedy algorithm was first provided in [C79].

6.2 The Knapsack Problem

Definition 2 *In the* Knapsack *problem, we are given a set of items, $I = \{1, \ldots, n\}$, each with a weight, $w_i \geq 0$, and a value, $v_i \geq 0$. We are also given a weight capacity, $C \geq 0$. The goal is to find a maximum-value subset $X \subseteq I$ with $\sum_{i \in X} w_i \leq C$.*

We want to prove the following:

Theorem 2 *For all $\epsilon > 0$, there is a $(1 - \epsilon)$-approximation for* Knapsack *which runs in time $O(n^3/\epsilon)$.*

Before proving this, we present an exact algorithm whose running time depends on the largest weight. This will be used as a subroutine in our approximation algorithm.

Theorem 3 *If all values v_i are integers at most D, we can solve* Knapsack *in time $O(n^2 \cdot D)$.*

Proof. We use dynamic programming. Note that $OPT \leq n \cdot D$.

For $0 \leq i \leq n$, and an integer $0 \leq p \leq n \cdot D$, let $f(i,p)$ be the minimum weight of a subset of $\{1,\ldots,i\}$ that has value p (∞ if no such subset exists). The optimum solution value is the largest $p \leq n \cdot D$ such that $f(n,p) \leq C$.

The following recurrence allows us to calculation the $f(i,p)$ values efficiently:

$$f(i,p) = \begin{cases} 0 & i = 0, p = 0 \\ \infty & i = 0, p \neq 0 \\ \min(f(i-1,p), f(i-1,p-v_i) + w_i) & i \neq 0, v_i \leq p \\ f(i-1,p) & i \neq 0, v_i > p \end{cases}$$

For example, if $i \neq 0$ and $v_i \leq p$ then a minimum-weight set $S \subseteq \{1,\ldots,i\}$ with value $= p$ either contains i or it does not. The min in this step of the recurrence tries both possibilities and keeps the best answer.

Since there are $O(n^2 \cdot D)$ different pairs (i,p) with $0 \leq i \leq n$ and $0 \leq p \leq n \cdot D$ then we can calculate all $f(i,p)$ values using dynamic programming in $O(n^2 \cdot D)$ time.

Using standard dynamic programming techniques, we can use this recurrence to construct a set S with weight $\leq C$ and value OPT in $O(n^2 \cdot D)$ time. ■

Of course, since KNAPSACK is NP-hard we do not expect this algorithm can be used to efficiently solve all instances. The instances of KNAPSACK that are produced in the known NP-hardness proofs have values that are exponential in n. Still, we can use this algorithm to approximate the optimum solutions of any KNAPSACK instance. The main idea is to scale the values so the largest is only polynomial in n and $1/\epsilon$ and then round the values down to the nearest integer. This is much like making a rough estimate of a grocery bill while shopping by only adding the dollar values of the items you have in your cart.

This scaling and rounding will be done carefully to ensure that an optimum solution with the new values is a near-optimum solution under the original values. We then use the dynamic programming algorithm described in Theorem 3 on this scaled instance.

Algorithm 2 is our final approximation for KNAPSACK.

Algorithm 2 A $(1-\epsilon)$-Approximation for KNAPSACK

Input: A set of items I, weights $w_i \geq 0$ and values $v_i \geq 0$ for $i \in I$, a capacity $C \geq 0$, and a parameter $\epsilon > 0$
Output: A subset S of I with total weight at most C

Discard any i with $w_i > C$ from I
$\delta \leftarrow M \cdot \epsilon / n$ where $M = \max_{i \in I} v_i$
Let $v'_i \leftarrow \lfloor v_i/\delta \rfloor$
Use the algorithm described by Theorem 3 on the instance with values v'_i to find a set of items S
return S

Proof of Theorem 2. Let S^* be the optimum solution for the original problem (the one with values v_i).

Note the following facts:

- $M \leq OPT$, because the item $i \in I$ with largest value v_i fits in the knapsack by itself.
- $v_i/\delta - 1 \leq v'_i \leq v_i/\delta$ for each item $i \in I$.

The last fact shows $v'_i \leq v_i/\delta \leq M/\delta \leq n/\epsilon$. Therefore, the running time of the step that invokes the dynamic programming algorithm described in Theorem 3 is $O(n^3/\epsilon)$. The remaining steps clearly take less time, so the overall running time is $O(n^3/\epsilon)$.

We conclude by bounding the value of the solution S found by Algorithm 2.

$$\begin{aligned}
\sum_{i \in S} v_i &\geq \delta \cdot \sum_{i \in S} v'_i \\
&\geq \delta \cdot \sum_{i \in S^*} v'_i && \text{(Since } S \text{ is optimal for the instance with values } v'_i\text{)} \\
&\geq \delta \cdot \sum_{i \in S^*} \left(\frac{v_i}{\delta} - 1 \right) \\
&= \sum_{i \in S^*} v_i - |S^*| \cdot \delta \\
&\geq OPT - n \cdot \delta \\
&= OPT - M \cdot \epsilon \\
&\geq OPT - OPT \cdot \epsilon = (1 - \epsilon) \cdot OPT
\end{aligned}$$

■

Theorem 2 was first proved in [IK75]. Since then, quite a few fundamentally different algorithms for Knapsack have been found that achieve this same approximation ratio. The Williamson and Shmoys text has a neat exercise at the end of Chapter 3 on reducing the running time of this approximation to $O(n^2/\epsilon)$.

References

C79 V. Chvátal, A greedy heuristic for the set-covering problem, *Mathematics of Operations Research*, 4:233–235, 1979.

IK75 O. H. Ibarra and C. E. Kim, Fast approximations for the knapsack and sum of subset problems, *Journal of the ACM*, 22:463–468, 1975.