CMSC 641, Design and Analysis of Algorithms, Spring 2010

Sandipan Dey,
Homework Assignment - 9

30/30

April 20, 2010

Bin Packing

✓

(a) Bin Packing is NP Hard

10

Let's reduce Bin Packing from the known NP-Compete problem Subset Sum in couple of steps (using transitivity of reduction). We first define the problems formally and then define a restriction of the Subset Sum problem: Subset Half Sum.

OK, but that was not needed

Bin Packing $= \{<S, j> \mid S = \{s_1, s_2, \ldots, s_n\}, 0 < s_i < 1$, and all of objects $1, \ldots, n$ fit into j unit-sized bins.$\}$

SubSet Sum $= \{<C, k> \mid C = \{c_1, c_2, \ldots, c_n\}$, where c_i is a positive integer, and there is some subset of C that adds up to exactly $k\}$

Subset Half Sum $= \{<C, k> \mid C = \{c_1, c_2, \ldots, c_n\}$, where c_i is a positive integer, and there is some subset of C that adds up to exactly $W/2$, where $W = \sum_{c_i \in C} c_i /2\}$

$$\text{SubSet Sum} \le_m^p \text{Subset Half Sum} \le_m^p \text{Bin Packing}$$

Subset Sum $\le_m^p$ Subset Half Sum

1. $k = \frac{W}{2} \Rightarrow$ we are done.
2. $k < \frac{W}{2} \Rightarrow$ add an element e of weight W_e in the set C. If there is a subset, S that sums to k in the old set, then there is a set $S \cup \{e\}$ that sums to $k + W_e$ in the new set. Also, the new set has total weight $W + W_e$. (We need $k + W_e = \frac{W+W_e}{2} \Rightarrow W_e = W - 2k$).
   Hence, if we add element e with $W_e = W - 2k$, the set $S \cup \{e\}$ sums to

$W - 2k$ which is exact half of the total weight of the new set $(2(W-k))$, this is a problem of Subset Half Sum.

3. $k > \frac{W}{2} \Rightarrow$ again add an element e of weight W_e in the set C. But this time we shall not include the element into the subset. If there is a subset, S that sums to k in the old set, then the same subset S with sum k exists in the new set too. Also, the new set has total weight $W + W_e$. (We need $k = \frac{W+W_e}{2} \Rightarrow W_e = 2k - W$).
   Hence, if we add element e with $W_e = 2k - W$, the set S sums to k which is exact half of the total weight of the new set $(2k)$, this is a problem of Subset Half Sum. ✓

Subset Half Sum $\leq_m^p$ Bin Packing

We keep the same instance set and ask the following question for the decision problem: how many bins of size $\frac{W}{2}$ are required. If the number of bins required is exactly 2, then the Subset Half Sum answer is Yes. It is No otherwise.
If the minimal number of bins is 2 and each bin has capacity $\frac{W}{2}$ then each must contain a subset of weight exactly $\frac{W}{2}$ and each is a solution to Subset Half Sum. If the minimal number of bins is greater than 2 (it cannot be 1 since that is smaller than W), then no subset with weight $\frac{W}{2}$ exists.
Otherwise we would use that subset in the first bin and the remaining $\frac{W}{2}$ weighted items in the second and have a 2-bin solution.

(b)

- Size of a bin = 1.
- The total weight packed in n bins is at most n. ✓
- To pack objects with total weight n, we need at least n bins.
- To pack objects with total weight S, we need at least $\lceil S \rceil$ bins.

(c)

- Let's assume to the contrary.
- Assume the first-fit heuristic leaves two bins that are less than half full. ✓
- Let b_1 be the first bin that is less than half-full and let b_2 be the second bin that is less than half-full.
- According to the first-fit heuristic, an object, s_k that is in b_2 is placed there because b_2 is the first bin that can accommodate it. But, if b_2 ends up less than half full $\Rightarrow s_k < 0.5 \Rightarrow$ it would have fit in b_1, since b_1 has remaining space of more than 0.5.

- Hence, a contradiction with the definition of the first-fit heuristic, in that b_2 is not the first bin that could accommodate s_k.
- Hence, our assumption must have been false, and the first-fit heuristic must leave at most one bin less than half full.

(d)

- Let's assume to the contrary again.
- Assume more than $\lceil 2S \rceil$ bins were used.
- Since the first-fit heuristic fills all but the last bin with weight at least $\frac{1}{2}$ (from (c)), the first $\lceil 2S \rceil$ bins have at least S weight in them and the $\lceil 2S \rceil + 1$ bin has some non-zero weight. This exceeds the total weight S of the set packed, a contradiction.

✓

(e)

✓

- $C*$ be the optimal solution $\Rightarrow |C*| \geq \lceil S \rceil$ (by part (b)).
- C be the first-fit heuristic solution $\Rightarrow |C| \leq \lceil 2S \rceil$ (by part (d)).
- $\Rightarrow$ Approximation factor $= \frac{C}{C*} \leq \frac{\lceil 2S \rceil}{\lceil S \rceil} = 2$.

(f)

Implementation

- Sort objects into increasing size.
- Create the intial bin ($i = 1$).
- for all objects in the set S do
  1. if object fits in current bin b_i, add that object to bin b_i.
  2. else create a new bin b_{i+1} and put the object into that bin.
  3. $i = i + 1$.
- Output i as the final number of bins needed to pack all the objects.

not needed! Do you read the problem requirements :-)

Analysis

In order to sort n objects time required is $\theta(nlogn)$. The for loop is executed $\theta(n)$ times. The amount of work to perform per iteration is $\theta(1)$. The rest of the lines are also $\theta(1)$ work. Hence, the total running time of this algorithm is $\theta(nlogn)$.

✓

10

Approximating Max Clique

(a)

Let $G = (V, E)$. First, we demonstrate a clique in $G^{(k)}$ of size l^k where l is the size of the maximum clique in G. Consider the set of verticies, $C'' = \{(v_1, v_2, \ldots, v_k) : v_i \in C\}$, where C is a maximum clique in G. Each vertex in C'' is adjacent to every other vertex since for $(v_1, \ldots, v_k)$ and $(u_1, \ldots, u_k)$ an edge exists if any $u_i \neq v_i$. The size of C'' is l^k and forms a clique in $G^{(k)}$.

✓

We show that there is no larger clique inductively. First, since $G^{(1)} = G$, it is clear for $k = 1$. Now, assume it is true for all $i < k$. Consider a largest clique C^k in $G^{(k)}$. Let $C_a = \{(a, v_2, \ldots, v_k) | (a, v_2, \ldots, v_k) \in C^k\}$ for $a \in V$. By induction, $|C_a| \leq l^{k-1}$. For $a \neq b$, either $(a, b) \in E$ or for every $(a, v_2, ldots, v_k) \in C_a$ and every $(b, u_2, \ldots, u_k) \in C_b$, $(v_2, \ldots, v_k)$ and $(u_2, \ldots, u_k)$ are connected in $G^{(k-1)}$. If a and b are not connected in G, we will take each vertex of C_b and replace its first component with a. The resulting set of vertices is still a clique of $G^{(k-1)}$ of the same size since the projections of C_a and C_b onto their last $(k-1)$ components are disjoint. ea replace C_b are indepen first vertex in a tuple of size k, a. The tuple-verticies that begin with a must form a clique on the remaining $k-1$ elements (and there are l^{k-1} of those by the induction). Further, for some other prefix, b, if $(a, b) \in G$, then the verticies prefixed by BOTH a and b must form a clique on the remaining verticies. That is, they must share the cliques on $(k-1)$-tuples. Therefore, if the prefixes are drawn from m verticies, and the largest clique is l, then $m - l$ edges are missing and $m - l$ vertices must share $(k-1)$-tuple cliques with another vertex giving us a maximum of $m - (m - l) \times l^{(k-1)} = l \times l^{(k-1)} = l^k$ verticies in the k-tuple clique.

✓

(b)

Given an polynomial c-approximation for MAX-CLIQUE, we can generate a ϵ (for any constant ϵ) approximation by running it on graph $G^{(k)}$, where $\epsilon > c^{1/k}$ and then taking the k-th root of the answer returned. This remains polynomial time because the problem is only blown up by a polynomial factor, $k = lg(c)/lg(\epsilon)$ which is a constant.

✓

Weighted Set Covering

10

(a)

The algorithm for greedy setcover can be naturally extended to the weighted set covering problem as well. Here the greedy method needs to ensure that at each stage whenever it picks a new set S from the collection, S not only covers the greatest number of remaining elements that are uncovered, but also the cost of S should be as low as possible. We define the cost effectiveness for the same purpose as shown in the following algorithm:

```
Algorithm: WeightedGreedySetCover($X, F$)
$C \longleftarrow \emptyset$
$U \longleftarrow X$
while $U \neq \emptyset$ do
    Find set $S \in F \setminus C$ that minimizes $\alpha := \frac{\text{cost}(S)}{S \cap U}$        (Cost effectiveness)
    for each $x \in S \cap U$ do
        price($x$) $\longleftarrow \alpha$
    $C \longleftarrow C \cup \{S\}$
    $U \longleftarrow U \setminus S$
return $C$
```

or maximizes $\left(\frac{S \cap U}{\text{cost}(S)}\right)$.

(b)

Following the anaysis from CLRS (p. 1036, 1037 2nd edition), the analysis can be generalized for weighted set covering in the following manner:

$$c_x = \frac{cost(S_i)}{|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|}$$

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} (u_{i-1} - u_i) \cdot \frac{cost(S_i)}{|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|} .$$

Observe that $\dfrac{cost(S_i)}{|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|} \le \dfrac{cost(S)}{|S - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|}$

by the greedy choice of S_i

$$\sum_{x \in S} c_x \le \sum_{i=1}^{k} (H(u_{i-1}) - H(u_i))\, cost(S)$$

$$= cost(S)\, H(|S|)$$

$$\begin{aligned} |C| &= \sum_{x \in X} c_x \\ &\le \sum_{S \in C^*} \sum_{x \in S} c_x \, . \\ &\le \sum_{S \in C^*} cost(S)\, H(|S|) \\ &\le H(\max\{|S| : S \in \mathcal{F}\}) \sum_{S \in C^*} cost(S) \\ &\le |C^*| \cdot H(\max\{|S| : S \in \mathcal{F}\}) \end{aligned}$$

$$\boxed{\frac{|C|}{|C^*|} \le H(\max\{|S| : S \in \mathcal{F}\})}$$

✓