Informatics 2 – Introduction to Algorithms and Data Structures

Tutorial 9 - Approximation Algorithms - Solutions

1. Recall the VERTEX COVER problem from the lectures, which we proved to be NP-complete. We will consider several greedy algorithms for the problem.

   GREEDY1: Repeat the following process until the graph is empty: consider any edge (v_1, v_2) of the graph G and include either v_1 or v_2 in the vertex cover S. Remove the incident edges of the chosen node from the graph.

   GREEDY2: Repeat the following process until the graph is empty: consider any edge (v_1, v_2) of the graph G and include both of v_1 and v_2 in the vertex cover S. Remove the incident edges of both v_1, v_2 from the graph.

   GREEDY3: Start with an empty set S. While S is not a vertex cover, choose a node u with maximum degree and add it to S; remove all the incident edges to u and repeat.

   (a) Show that the approximation ratio of GREEDY1 is $\omega(1)$.

   (b) Show that the approximation ratio of GREEDY2 is at most 2.

   (c) Does GREEDY3 also achieve an approximation ratio of 2? Justify your answer.

   Solution:

   (a) Consider a bipartite graph $G = (L \cup R, E)$ for which $|L| = \{x_1, \ldots, x_k\}$ and R consists of k groups of nodes $R_1, \ldots, R_k$, with $|R_i| = k/i$. For any node $e \in R_i$, there is an edge to i nodes of L, such that for any two nodes $u, v \in R_i$, u and v do not have a common neighbour in L. From this, it follows that each node in L has degree at most k, and each node in R_i has degree exactly i.

   Choosing L is a vertex cover of size k in the graph. Our algorithm might pick only nodes from R, and in particular all of the nodes from each R_i. These are in total $k \sum_{i=1}^{n} 1/i = k \cdot H(k)$ nodes, where $H(k)$ is the k-th harmonic number. Asymptotically, we have that $H(k) = \Theta(\lg k)$. The approximation ratio therefore is $\Omega(\lg k)$. Since $n = \Theta(k \lg k + k)$, this is $\omega(1)$.

   (b) First, let's think of what the algorithm does on the example above. When it selects an edge e it will include both of its endpoints to the vertex cover. This means in particular that the nodes in L will be added. Once a node in L is added, all of the edges incident to that node are "covered", so we will not have to add most of the nodes of R. In particular, for every node $u \in L$, we will add exactly one node of R, and that is the node that shares the edge that made us "discover" u in the first place. This example in fact shows that the approximation ratio of GREEDY2 is at least 2, as on this example it is precisely 2.

To see that 2 is also an upper bound, observe that any vertex cover, including the optimal, must contain at least one of the two endpoints u and v of any edge (u, v). Our algorithm produces a vertex cover that contains both, so it is only a factor of 2 away. In addition, the algorithm does not include any nodes that are not incident to edges to the vertex cover. This establishes the 2-approximation.

(c) GREEDY 3 does not have an approximation ratio of 2. In fact, this also has an approximation ratio of $\omega(1)$, and this follows from the example that we constructed for GREEDY 1. In the first stage, it could pick the node from R_k, as its degree is k, which is maximum. In the next step, it could choose both of the nodes from R_{k-1}, as the degree of each of those is $k-1$, and so on. In the end, it may select exactly the same set of nodes as GREEDY1, resulting in the same approximation ratio.

2. A family is moving house, and they would like to move all of their belongings. To do that, they would like to purchase as few boxes as possible, to transfer all of their belongings. In particular, they have n items, with item i having size $w_i \in [0, 1)$, and each box has size (capacity) 1. They would like to put all of the items in some box, and use the minimum number of boxes possible.

(a) The family consider the following greedy algorithm for their problem: They fix an ordering of their items and open a box. They start putting items in the box until the next item in the order does not fit. When this happens, they open a new box and continue with the same process. The process terminates when all the items have been put in boxes.

Prove that this greedy algorithm has approximation ratio at most 2, i.e., it uses at most twice as many boxes as the optimal solution.

(b) Prove that the problem faced by the family is NP-hard, by providing a polynomial-time reduction from the following problem:

> PARTITION: Given positive integers $\alpha_1, \alpha_2, \ldots, \alpha_n$, decide whether it is possible to find a subset A of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in A} \alpha_i = \sum_{i \notin A} \alpha_i$.

Solution: This problem is the well-known online bin packing problem and the proposed algorithm is the NEXTFIT algorithm. For the proof, let B_i denote the capacity occupied by the items in bin i and assume for simplicity that the algorithm opens an even number m of bins during its execution. It holds that $B_1 + B_2 > 1$, as bin 2 was opened because the first item that was put in bin 2 would not fit into bin 1. Similarly, we have that $B_3 + B_4 > 1$, ..., $B_{m-1} + B_m > 1$. Adding up all of these inequalities we get (since we have $m/2$ inequalities):

$$\sum_{i=1}^{m} B_i > m/2$$

By definition, it holds that $\sum_{i=1}^{m} B_i = \sum_{i=1}^{n} w_i$, while at the same time it obviously holds that the value of the optimal solution is at least $\sum_{i=1}^{n} w_i$. This implies that the approximation ratio of the algorithm is at most 2.

For the NP-hardness of the problem, we can design a polynomial-time reduction from PARTITION. We consider the decision version of the bin packing problem, where we

are also given an integer k and we would like to decide if we can pack the items in at most k bins. We set $w_i = 2\alpha_i / \sum_{j=1}^{n} \alpha_j$ and $k = 2$.

Assume that have a solution to an instance of partition PARTITION, i.e., a set $A \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in A} \alpha_i = \sum_{i \neq A} \alpha_i$. In the instance of BIN PACKING, we place item i in the first box if any only if $i \in A$ in the solution to PARTITION, and to the second box otherwise. The total weight of box 1 is therefore $\sum_{i \in A} w_i = \sum_{i \in A} \left(2\alpha_i / \sum_{j=1}^{n} \alpha_j \right) \leq 1$, and similarly for the second box.

Conversely, suppose that we have a solution to BIN PACKING. This means that for each of the two boxes, we have $\sum_{i \in \text{box}} w_i = \sum_{i \in \text{box}} \left(2\alpha_i / \sum_{j=1}^{n} \alpha_j \right) \leq 1$. This implies that $\sum_{i \in \text{box}} \left(\alpha_i / \sum_{j=1}^{n} \alpha_j \right) \leq 1/2 \Rightarrow \sum_{i \in \text{box}} \alpha_i \leq \left(\sum_{j=1}^{n} \alpha_j \right) /2$. In the instance of PARTITION, this means that

$$\sum_{i \in A} \alpha_i \leq \left(\sum_{j=1}^{n} \alpha_j \right) /2 \text{ and } \sum_{i \notin A} \alpha_i \leq \left(\sum_{j=1}^{n} \alpha_j \right) /2,$$

which is only possible when $\sum_{i \in A} \alpha_i = \sum_{i \notin A} \alpha_i$, i.e., A is a solution to the instance of PARTITION.

3. *(Optional, more advanced)* Consider the SET COVER problem:

   Definition 1 (WEIGHTED SET COVER)**.** We are given a set of elements $\mathcal{U}$ and a collection of subsets of $\mathcal{U}$, each with an associated weight w_i, whose union is $\mathcal{U}$. The goal is to find set of subsets of $\mathcal{U}$ whose union is equal to $\mathcal{U}$, of minimum total weight, i.e., a set of sets $S = S_1, \ldots, S_k$ such that for every element $j \in U$, there exists a set $S_i \in S$ such that $j \in S_i$., such that $\sum_{i \in S} w_i$ is minimised.

   Show that SET COVER is a generalization of VERTEX COVER.

   Consider the following greedy algorithm for SET COVER: Greedily select sets based on the following "efficiency measure": Let's assume that we have already covered some elements of $\mathcal{U}$ by our selection of sets, and let R be the remaining elements of $\mathcal{U}$ to be covered. Among the sets that have not been yet selected, we will select that which minimises

   $$\frac{w_i}{|S_i \cap R|}.$$

   The idea is that we want to select sets that have small weight, but also cover lots of elements[1]. The pseudocode for the algorithm is the following.

Algorithm 1 GREEDY WEIGHTED SET COVER

```
Start with R = U and C = ∅.                ▷ Note that C is a set of sets.
while R ≠ ∅ do
    Select a set S_i ∈ arg min w_i/|S_i ∩ R|.
    Add S_i to C.
    Remove the elements of S_i from R      ▷ These elements have now been covered.
Return C.
```

[1] This might be reminiscent of the greedy algorithm we used for 0/1-Knapsack, where we wanted elements that have large value but small weight, and we took the ratio v_i/w_i as our measure of efficiency.

Prove that the approximation ratio of this greedy algorithm is $O(\lg n)$, where $n = |\mathcal{U}|$.

Solution: We need to argue two things for the algorithm, (a) that it outputs a feasible solution (i.e., a set cover) and (b) that the weight of the set cover that it outputs is at most twice the size of the minimum possible set cover. Property (a) is straightforward from the definition of the algorithm; the algorithm does not terminate until it covers all the elements of $\mathcal{U}$. Next, we argue about property (b).

To get some intuition, consider an element s in $\mathcal{U}$. How much do we "pay" for this element, in order to cover it? We will define the "cost" or "price" of element s as the quantity we used in our greedy step. In particular, we define

$$c_s = \frac{w_i}{|S_i \cap R|}, \text{ for all } s \in S_i \cap R.$$

We can insert this line between the "Select" and "Remove" commands of our while loop in the pseudocode, without changing the complexity of the algorithm. This is not necessary, but it will be useful as a reference in the analysis. Generally, the idea is that when we select a set S, its weight is distributed over the costs c_s of the elements that this set covers. If we sum over all of the costs, then we get the total weight of the set cover, therefore we have:

$$\sum_{S_i \in \mathcal{C}} w_i = \sum_{s \in \mathcal{U}} c_s.$$

Claim 1. *For every set S_j, it holds that $\sum_{s \in S_j} c_s \leq H(|S_j|) \cdot w_j$, where $H(n) = \sum_{i=1}^{n} \frac{1}{i}$ is the* harmonic function.

Proof. For ease of notation, let us assume that $S_j = \{s_1, \ldots, s_d\}$ and let us assume that these elements are labelled in the order in which they are assigned a "cost" c_{s_i} by the algorithm (ties can be broken arbitrarily). Consider the iteration of the algorithm during which element s_i is covered by our Greedy algorithm, for some $i \leq d$. Since we consider the elements in order, at the beginning of the iteration, it holds that elements $s_i, s_{i+1}, \ldots, s_d$ are uncovered, and therefore it holds that $s_i, s_{i+1}, \ldots, s_d \in R$. In turn, this implies that $|S_j \cap R| \geq d - i + 1$ and if we take the average cost of the set S_j (the inefficiency), we have

$$\frac{w_j}{|S_j \cap R|} \leq \frac{w_k}{d - i + 1}.$$

Since the Greedy algorithm selected a set S_i with the minimum average cost, and the "cost" of element s_i is the average cost of the set S_i that covered it, we have that

$$c_{s_i} = \frac{w_i}{|S_i \cap R|} \leq \frac{w_j}{|S_j \cap R|} \leq \frac{w_j}{d - i + 1}$$

Finally, we add up all of these inequalities for all elements $s \in S_j$, and we have

$$\sum_{s \in S_j} c_s = \sum_{\ell=1}^{d} c_{s_\ell} \leq \sum_{\ell=1}^{d} \frac{w_j}{d - \ell + 1} = w_j \cdot \sum_{\ell=1}^{d} \frac{1}{d - \ell + 1}$$

If we expand the latter quantity, we get $w_j \left(\frac{1}{d} + \frac{1}{d-1} + \ldots + \frac{1}{1} \right) = H(d) \cdot w_j$. □

We will use Claim 1 to prove the approximation ratio of the mechanism. Let $d^* = \max_i |S_i|$ be the maximum size of any set in $\mathcal{S}$. Also let $\mathcal{C}^*$ be the minimum weight set cover and recall that $\mathcal{C}$ is the set cover returned by our Greedy algorithm. Also, for ease of notation, let $w^* = \sum_{S_i \in \mathcal{C}^*} w_i$ be the weight of $\mathcal{C}^*$. From Claim 1, we have that

$$w_i \geq \frac{1}{H(d^*)} \sum_{s \in S_i} c_s$$

Since $\mathcal{C}^*$ is a set cover, we have that

$$\sum_{S_i \in \mathcal{C}^*} \sum_{s \in S_i} c_s \geq \sum_{s \in \mathcal{U}} c_s.$$

Putting everything together, we have that:

$$w^* = \sum_{S_i \in \mathcal{C}^*} w_i \geq \sum_{S_i \in \mathcal{C}^*} \left(\frac{1}{H(d^*)} \sum_{s \in S_i} c_s \right) \geq \frac{1}{H(d^*)} \sum_{s \in \mathcal{U}} c_s = \frac{1}{H(d^*)} \sum_{S_i \in \mathcal{C}} w_i.$$

Therefore, our Greedy algorithm is a $H(d^*)$ approximation. Since $d^* \leq n$, this is also an $H(n)$ approximation algorithm. It hold that $H(n) = \Theta(\log n)$ and therefore we get that the approximation ratio of our Greedy algorithm is $O(\log n)$.