CS 4820, Spring 2017 Homework 11, Problem 2

Name: Yuxiang Peng

NetId: yp344

Collaborator: jl3455, zl542

(a) A counterexample for GA is an instance with two elements: (v1 = 2, w1 = 1) and (v2 = 2C + 1, w2 = 2C + 1). If the knapsack capacity is 2C + 1, then GA will pick the first element and then the second one won't fit. The value obtained will be only 2, whereas value 2C + 1 would be obtained by taking the second element instead.

A counterexample for EMGA is an instance with 2C + 2 elements: one element with (v1 = 1, w1 = 2C + 1) and 2C + 1 elements each with (vi = 1, wi = 1). If the knapsack capacity is 2C + 1, then EMGA will pick the first element and then all the remaining ones won't fit. The value obtained will be only 2, whereas value 2C + 1 would be obtained by taking all elements except the first one.

(b)Assume that there are no items whose weight is greater than W. Such items will never appear in any solution, and can be eliminated without changing the value of the optimum. We try to prove that GA + EMGA >= OPT from which is follows that max{GA, EMGA} >= ½ * OPT. Let i1, … , ik denote the sequence of items selected by GA (in the order selected) and let i(k+1) be the next element that GA would have selected, if it had not run out of space. The EMGA gets a value at least as high as the value of item i(k+1), and the value of items i1, … , ik is the value achieved by the GA. On the other hand, the combined value of items i1, … , ik, i(k+1) is greater than that of the optimal knapsack solution Sopt, since Sopt has a smaller combined size, and any elements of Sopt that are not among { i1, … , i(k+1)} have a value density that is smaller than the least dense element of that set.

Let $S_0 = S_{OPT} \cap \{i_1, \dots, i_{k+1}\}$, $S_1 = \{i_1, \dots, i_{k+1}\} \backslash S_{OPT}, S_2 = S_{OPT} \backslash \{i_1, \dots, i_{k+1}\}$, if the value density of each item i by pi = vi/wi, and let p* = $p_{i(k+1)}$, from greedy we know that $p_{i(j)}$ >= p* for j = 1,…,k+1 whereas pi<=p* for every i not belongs to $\{i_1, \dots, i_{k+1}\}$. So vi>=p*wi for every i belong to S1, while vi<=p*wi for every i belongs to S2. It shows that:

$\left(\sum_{j=1}^{k+1} v_{i_j}\right) - \left(\sum_{i \in S_{opt}} v_i\right) = \left(\sum_{i \in S_1} v_i\right) - \left(\sum_{i \in S_2} v_i\right) \geq \left(\sum_{i \in S_1} \rho^* w_i\right) - \left(\sum_{i \in S_2} \rho^* w_i\right) = \rho^* \left(\sum_{i \in S_1} w_i - \sum_{i \in S_2} w_i\right)$, $\left(\sum_{j=1}^{k+1} v_{i_j}\right) - \left(\sum_{i \in S_{opt}} v_i\right)$ is positive and because $\sum_{i \in S_1} w_i > W - \sum_{i \in S_0} w_i \geq \sum_{i \in S_2} w_i$, $so\ \rho^* \left(\sum_{i \in S_1} w_i - \sum_{i \in S_2} w_i\right) is\ also\ positive\ and\ v_{i_1} + \cdots + v_{i_{k+1}} > \mathrm{OPT}$, so $\mathrm{GA} + \mathrm{EMGA} \geq \mathrm{OPT}$ is proved.

(c)It is presented that dynamic programming algorithm for the knapsack problem that runs in time O(nv*), where v* is the upper bound of the value of the optimum solution. It is not a polynomial algorithm because the running time is proportional to v*. Compute a set S* obtained by running GA and EMGA to get two sets, and taking the one with the higher total value. Let $v_* = \sum_{i \in S_*} v_i\ and\ k = \frac{\varepsilon v_*}{(1+\varepsilon)n}, for\ every\ item\ i\ let\ v'_i = \frac{v_i}{k}$. Let S is the output of the dynamic program on the instance defined by $\{(v'_i, w_i) | i = 1, \dots, n\}$ and Sopt by $\{(v_i, w_i) | i = 1, \dots, n\}$. Because it outputs the optimal solution for the knapsack instance on $\{(v'_i, w_i) | i = 1, \dots, n\}$. We get that $\left(\sum_{i \in S} v'_i\right) \geq \left(\sum_{i \in S_{OPT}} v'_i\right), \left(\sum_{i \in S} v_i\right) \geq \left(\sum_{i \in S} kv'_i\right) \geq \left(\sum_{i \in S_{OPT}} kv'_i\right) \geq \left(\sum_{i \in S_{OPT}} (v_i - k)\right), \left(\sum_{i \in S} v_i\right) \geq \left(\sum_{i \in S_{OPT}} v_i\right) - nk = \left(\sum_{i \in S_{OPT}} v_i\right) - \frac{\varepsilon v_*}{(1+\varepsilon)} \geq \left(1 - \frac{\varepsilon}{1+\varepsilon}\right)\left(\sum_{i \in S_{OPT}} v_i\right), because\ v_* \leq \sum_{i \in S_{OPT}} v_i\ , so\ \sum_{i \in S} v_i \geq \left(\frac{1}{1+\varepsilon}\right)\left(\sum_{i \in S_{OPT}} v_i\right)$, so S is a $(1+\varepsilon)$ approximation for Sopt. So based on the value of k, the running time is O$(n^2(v^*/v_*)/\varepsilon)$. The computation of the value density of elements and running GA and EMGA for deciding the set S_* takes O(nlogn), based on the algorithm of O($n^3 s/\varepsilon$) presented in class, the overall running time of the algorithm is O($n^2 s/\varepsilon$) where s denotes the maximum number of bits in the binary representation of any of the numbers vi, wi(i=1,…,n).