- **Algorithm**

  We first guess k items in the optimal solution. Then for each possible combination we run the greedy algorithm on the remaining Knapsack problem by packing items i in U/X in non-increasing order of density p_i/s_i until the knapsack runs out of space.

- **The algorithm terminates in polynomial time O(n^{k+1}).**

  The brutal force search has O(n^k) possibilities because for each of the k slots we have n choices. For each combination of k items, the greedy algorithm runs in O(n), resulting in O(n^{k+1}) total time. Because k is some fixed constant, the algorithm runs in polynomial time.

- **The algorithm has an approximation ratio of $(1-\epsilon)$.**

  If an optimal solution has no more than k items, the approximation solution returns an exact solution because it has considered the optimal solution during the brute-force search.

  Otherwise, the optimal solution contains at least k items. We sort the packed items in decreasing order of density. When we pick up the first k items during the brutal force search and use greedy algorithm to fill the rest of the space, the approximation algorithm generates a result within the ratio as shown below.

  Suppose the mth item is the first one that the greedy algorithm fails to fit into the knapsack. We denote its space s_m and profit p_m. Because the greedy algorithm cannot fit item m into the remaining empty space of the knapsack,

$$s_m < S_e \tag{1}$$

  where S_e represents the unpacked empty space within B. In addition, s_m must also satisfy

$$s_m \le \frac{B}{k+1} \tag{2}$$

  because if every item in a solution is greater than $\frac{B}{k+1}$, then a solution with more than k items will result in that the total space exceeds B, which is impossible.

  For any solution, we denote the set of that k items as O while G the rest of the items that not in O. A solution by approximation $APPRO$ is thus composed of three parts: the first k items in the sorted item list, items that are also in the optimal solution from $(k+1)$th to $(m-1)$th, and a few items that are not in the optimal solution,

$$APPRO = P_1 + P_2 + P_3. \tag{3}$$

  If we denote the size of the three parts as S_1, S_2, and S_3, then

$$B = S_1 + S_2 + S_3 + S_e. \tag{4}$$

  The total space of all items in part three is

$$S_3 = B - S_e - \sum_{i=1}^{m-1} s_i. \tag{5}$$

  And the total profit of all items in G is the sum of the later two parts,

$$P_G = P_2 + P_3 \ge \sum_{i=k+1}^{m-1} p_i + S_G * \frac{p_m}{s_m} \tag{6}$$

due to the fact that item m has the least density $\frac{p_m}{s_m}$ among all items in G.

The total profit in the optimal solution is also composed of three parts similar to that in the approximated solution. Along with the previous equations, we obtain the expression for the optimal solution,

$$\begin{aligned} OPT &= \sum_{i=1}^{k} p_i + \sum_{i=k+1}^{m-1} p_i + \sum_{i=m}^{|O|+|G|} p_i \\ &\leq P_1 + (P_G - S_3 * \frac{p_m}{s_m}) + (B - \sum_{i=1}^{m-1} s_i) * \frac{p_m}{s_m} \\ &= P_1 + (P_2 + P_3 - S_3 * \frac{p_m}{s_m}) + (B - S_1 - S_2) * \frac{p_m}{s_m} \\ &= (P_1 + P_2 + P_3) + (B - S_1 - S_2 - S_3) * \frac{p_m}{s_m} \\ &= APPRO + S_e * \frac{p_m}{s_m}. \end{aligned} \tag{7}$$

Then, according to Equation 1,

$$\begin{aligned} OPT &\leq APPRO + S_e * \frac{p_m}{s_m} \\ &= APPRO + p_m. \end{aligned} \tag{8}$$

We notice that p_m is bounded by

$$\begin{aligned} p_m &= s_m * \frac{p_m}{s_m} \\ &< \frac{B}{k+1} * \frac{OPT}{B} \\ &= \frac{OPT}{k+1}. \end{aligned} \tag{9}$$

To sum up the above two cases, we obtain that

$$(1 - \frac{1}{k+1}) * OPT < APPRO \leq OPT. \tag{10}$$

Substituting with $\epsilon = \frac{1}{k+1}$, we realize that the approximation ratio is $(1 - \epsilon)$.

- **The Algorithm is a PTAS to the Knapsack problem.**

  When ϵ is a constant, the algorithm runs in polynomial time O(n^ϵ). Its approximation ratio $(1 - \epsilon)$ only depends on ϵ.