Sishir Subedi
Assignment -1
February 14, 2018
CSCE -669 Computational Optimization

Problem description - The KNAPSACK problem:

Given a set $Z = \{z_1, z_2, ..., z_n\}$ of n items, where each item z_i has a size s_i and a value v_i, and a knapsack of size S. Pick a subset of the items that can fit into the kanpsack and maximizes the value.

The KNAPSACK problem is NP-hard:

To show the knapsack problem is NP-hard we will use a known NP-Complete PARTITION problem which is described as the following:
Input:Given a set $X = \{a_1, a_2, ..., a_n\}$ of n items, where each a_i is an interger, partition set X into two disjoint sets $X^{'}$ and $X^{"}$ such that the sum of the integers in each subset equals to each other i.e.

$$\sum_{a_i \in X'} a_i = \sum_{a_i \in X^{"}} a_i$$

Now, we take an instance of a knapsack problem where we have the value of items and size of the items in knapsack is equal i.e. $s_i = v_i$ for each item i in a set $Z = \{z_1, z_2, ..., z_n\}$. Now, we assume that each item in a set Z is also equal to each items set X of partition problem i.e.$\sum_{z_i \in Z} v_i = \sum_{a_i \in X} a_i$. Now when there is two disjoint sets $X^{'}$ and $X^{"}$ then one of the sets in partition i.e. $s_i = v_i = a_i \in X^{'}$, and size of the knapsack S will be equal to sum of elements in set $X^{'}$.

$$\sum_{z_i \in Z} v_i = V = \sum_{z_i \in Z} s_i = S$$
$$= \sum_{a_i \in X'} a_i = \sum_{a_i \in X''} a_i = 1/2 \sum_{a_i \in X} a_i$$

Thus, finding a subset of items in knapsack problem will be equivalent of searching a disjoint set $X^{"}$ whose sum will be equivalent to $X^{'}$.

$$\sum_{a_i \in X'} s_i = 1/2 \sum_{a_i \in X} a_i$$

This reduction from known NP-Complete problem shows that the knapsack is NP-Complete problem.

Develop a simple polynomial time approximation algorithm for the KNAPSACK problem that has a constant approximation ration.

Let $Z = \{z_1, z_2, ..., z_n\}$ of n items, where for each i, the item z_i has a size s_i and a value v_i, and the size of a knapsack is S. We can purpose the following greedy algorithm to pick a subset of the items with maximum value that can fit into the knapsack.

Greedy Algorithm:
1. Sort n items in a decreasing order of $v_1/s_1 \geq v_2/s_2 \geq ... \geq v_n/s_n$
2. Then fill k items in a knapsack until total size is greater than S i.e.

$$\sum_{i=1}^{k} v_i \leq S$$

Here, the algorithm will perform poorly in a case where there exists an item with value of S. For example lets consider we have:
1. v_1 is 2 and s_1 is 1, then ratio is 2
2. v_2 is S and s_2 is S, then ratio is 1
In this case algorithm will pick v_1 which is not good. We can modify this algorithm as the following:

Modified Greedy Algorithm:
1. Sort n items in a decreasing order of $v_1/s_1 \geq v_2/s_2 \geq ... \geq v_n/s_n$
2. Then fill $k-1$ items in a knapsack until total size is greater than S i.e.

$$\sum_{i=1}^{k-1} v_i < S \quad and \quad \sum_{i=1}^{k} v_i > S$$

3. choose max of:

$$max(\sum_{i=1}^{k-1} v_i; v_k)$$

Here, let $V_{k-1} = \sum_{i=1}^{k-1} v_i$ be the value obtained after putting $k-1$ items in the knapsack. We consider a case for fractional knapsack then optimal value for fractional knapsack (FOPT) will be as following:

$$FOPT = \sum_{i=1}^{k-1} v_i + \frac{S - \sum_{i=1}^{k-1} s_i}{s_k} v_k$$

From above we know that the optimal solution (OPT) for interger knapsack is bounded by $OPT < \sum_{i=1}^{k-1} v_i + v_k$ because $\sum_{i=1}^{k} v_i > S$. Since algorithm choses $max(\sum_{i=1}^{k-1} v_i; v_k)$, we can have either

$$\sum_{i=1}^{k-1} v_i \geq OPT/2 \quad or \quad v_k \geq OPT/2$$

Therefore, modified greedy algorithm is a 2 approximation for the integer knapsack problem. Time complexity of this algorithm is bounded by the complexity of sorting algorithm used to sort items.

Develop a pseudo polynomial time approximation algorithm for the KNAPSACK problem.

We consider a dynamic programming solution for a pseudo polynomial time approximation algorithm for knapsack problem. The components of the dynamic program can be described as the following:
Subproblem: For subset of items $\{1, 2, ...i\}$ in set Z the maximum value is V_i.

Substructure:
case I: Optimal solution for items $\{1, 2, ...i\}$ does not contain item i. In this case, the optimal value is the value obtained from a subset $\{1, 2, ...(i-1)\}$.
case II: Optimal solution for items $\{1, 2, ...i\}$ contains item i. In this case, the optimal value is the value obtained from a subset $\{1, 2, ...(i-1)\} + v_i$.

Dynamic Algorithm:
Input : N items with sizes $s_1, s_2, ..., s_n$ with values $v_1, v_2, ..., v_n$, size of a backpack is S.
Create a array: M[0...N,0...S]
FOR s=0 to S: M[0,s]=0
FOR i = 0 to N:
    FOR s=0 to S:
        IF s[i] $\leq$ w:
            M[i,s] $= max\{M[i-1, s], v_i + M[i-1, s-s_i]\}$
        ELSE:
            M[i,s] $= M[i-1, s]$
$S^* = max\{s : M[N, s] \leq S\}$
$OPT = M[N, S^*]$

The time complexiy of this dynamic algorithm is $O(n^2 S^*)$, which is a pseudopolynomial because of bound by value of S^*.

Develop a fully polynomial time approximation scheme for the KNAPSACK problem.

In a pseudo polynomial time approximation algorithm we notice that if size S^* is small and bounded by N then the algorithm will be polynomial i.e. n^2. However, S^* is not bounded by N and the algorithm is pseudopolynomial. One idea of decreasing complexity of this algorithm is by dividing each item value v_i by k in such a manner that following conditions are kept:
1. maintain integer property i.e. $v_i^{'} = \lfloor \frac{v_i}{k} \rfloor$
2. value of k should be as small as possible so that precision lost due to division and floor function is minimum
3. value of k should be as high as possible so that the time comlexity of the modified dynamic program is bounded by a polynomial.

FPTAS Algorithm Framework:
1.For i=1 to n do $v_i^{'} = \lfloor \frac{v_i}{k} \rfloor$
2.Find optimal solution using dynamic programming for items $\{v_1^{'}, v_2^{'}, ...v_n^{'}\}$ to get set $S^{'}$
3.Output v_i corresponding to $v_i^{'}$ as set S_{appx}

Here, let $X = \{z_1, z_2, ..., z_j\}$ be the set of items with values $V = \{v_1, v_2, ..., v_j\}$ where sum of value of all items in this set is an optimal solution $OPT = \sum_{i=1}^{j} v_i$. Similarly, let $X^{'} = \{z_1^{'}, z_2^{'}, ..., z_{j^{'}}^{'}\}$ be the set of items with values

$V^{'} = \{v_1^{'}, v_2^{'}, ..., v_{j^{'}}^{'}\}$ where sum of value of all items in this set is an optimal solution $OPT^{'} = \sum_{i=1}^{j^{'}} v_i^{'}$ for modified values problem.
Now, first we need to find relationship between $OPT^{'}$ and OPT. To do that we need to express OPT in terms of $v_i^{'} = \lfloor \frac{v_i}{k} \rfloor$.

$OPT = \sum_{i=1}^{j} v_i$
$= k \sum_{i=1}^{j} \frac{v_i}{k}$
$\leq k \sum_{i=1}^{j} (\lfloor \frac{v_i}{k} \rfloor + 1)$
$\leq kn + k \sum_{i=1}^{j} \lfloor \frac{v_i}{k} \rfloor$
$\leq kn + k \sum_{i=1}^{j} v_i^{'}$
$\leq kn + k \sum_{i=1}^{j^{'}} v_i^{'}$, because $OPT \leq OPT^{'}$ i.e. $\sum_{i=1}^{j} v_i^{'} \leq \sum_{i=1}^{j^{'}} v_i^{'}$
$\leq kn + k \sum_{i=1}^{j^{'}} \lfloor \frac{v_i}{k} \rfloor$

$OPT \leq kn + OPT^{'}$ —(i)

So, we have approximation ration of:
$\frac{OPT}{OPT^{'}} \leq \frac{kn}{OPT^{'}} + 1$

i.e. $ratio \leq \frac{kn}{OPT^{'}} + 1$ —(ii)

We have a lower bound for OPT that it is $OPT \geq \frac{\sum_{i=1}^{n} v_i}{n}$ because optimal will be at equal to or larger than the average of all values. So we know that $OPT^{'} \geq OPT \geq \frac{\sum_{i=1}^{n} v_i}{n} \geq \frac{V_o}{n}$.Using this to the equation (i):
$OPT \leq kn + OPT^{'}$
$OPT - kn \leq OPT^{'}$
$\frac{V_o}{n} - kn \leq OPT^{'}$
so we substitute value of $OPT^{'}$ in equation (ii) then we will have
$ratio \leq \frac{kn}{OPT^{'}} + 1$
$ratio \leq \frac{kn}{\frac{V_o}{n} - kn} + 1$
$ratio \leq \frac{kn^2}{V_o - kn^2} + 1$
Here, let $\epsilon \geq \frac{kn^2}{V_o - kn^2}$ so that we have approximation ration bounded as $ratio \leq 1 + \epsilon$, then
$\epsilon(V_o - kn^2) \geq kn^2$
$\epsilon V_o kn^2) \geq kn^2$
$\frac{\epsilon V_o}{n^2(1+\epsilon)} \geq k$
$\frac{V_o}{n^2(1+\frac{1}{\epsilon})} \geq k$
Therefore, we can substitue $k = \frac{V_o}{n^2(1+\frac{1}{\epsilon})}$ in the FPTAS aglorithm above, and the running time algorithm will be $O(n^3(1 + \frac{1}{\epsilon}))$ i.e. $O(\frac{n^3}{\epsilon})$. This provides a solution $OPT^{'}$ for items in $Z^{'}$ set, and its corresponding item in Z will be solution for the knapsack problem by FPTAS algorithm with approximation ratio bounded by ϵ.