Algorithms
Knapsack Problem Handout

1 Introduction

The Knapsack Problem is defined as follows: A thief robbing a store finds n items. The ith item is worth v_i dollars and weighs w_i pounds, where v_i and w_i are positive integers. The thief wants to take as valuable a load as possible, but can carry at most W pounds in his/her knapsack, for some integer W. Which items should the thief steal?

2 Example

Consider an example with $n = 3$ items and a knapsack that can hold 50 pounds. Item 1 weights 10 pounds and is worth 60 dollars. Item 2 weights 20 pounds and is worth 100 dollars. Item 3 weights 30 pounds and is worth 120 dollars. One possible solution is for the thief to pick items 1 and 2 (value \$160); another solution picks items 1 and 3 (\$180). Neither is optimal, can you find the optimal solution? How about if we pick all three?

3 Solution Methods

Two ideas (described in English) for algorithms are described next (in English). For pseudocode, see the next section.

3.1 Exhaustive Approach

The exhaustive approach computes the total weight and value of every possible subset of items and chooses the subset that has the greatest value but whose weight does not exceed W. This requires evaluating all possible subsets; i.e., the power set. Recall that the power set of a set of n items contains 2^n subsets, since each of the n items can either be chosen or not chosen to give a different subset.

3.2 Greedy Heuristic Approach

The heuristic solution to the Knapsack problem looks for a good – but not always the best – combination of items by choosing items with the highest value-to-weight ratio first. Items are first sorted in decreasing order of value-to-weight ratio (if two or more items have the same ratio, the item with the greatest value is chosen first). Then each item is either added to the knapsack if it fits or skipped if not, until either the knapsack is full or all of the items have been tried.

4 Pseudocode for exhaustive and heuristic approaches.

Algorithm 1 Exhaustive Solution

```
Input: knapsack weight capacity $W$, number of items $n$, list of $(weight, value)$ pairs $items$

Let $PowerSet$ = list of all subsets of $items$
$knapsack = \{\ \}$
$bestValue = 0$
for each $subset$ in $PowerSet$ do
    $subsetValue = 0$
    $subsetWeight = 0$
    for each $item$ in $subset$ do
        $subsetValue$ += $item$.value
        $subsetWeight$ += $item$.weight
    if $subsetWeight \leq W$ AND $subsetValue > bestValue$ then
        $bestValue = subsetValue$
        $knapsack = subset$
return $knapsack$
```

Algorithm 2 Value-to-Weight Heuristic

```
Input: weight capacity $W$, number of items $n$, list of $(weight, value)$ pairs $items$

$knapsack = \{\ \}$
$currentW = W$
Sort $items$ in decreasing order by $\frac{item.value}{item.weight}$ and in case of ties, by decreasing order of $item.value$
for each $item$ in $items$ do
    if $item$.weight $\leq currentW$ then
        $knapsack$.add($item$)
        $currentW = currentW$ - $item.weight$
return $knapsack$
```