PES UNIVERSITY

Department of Computer Science and Engineering

PES UNIVERSITY

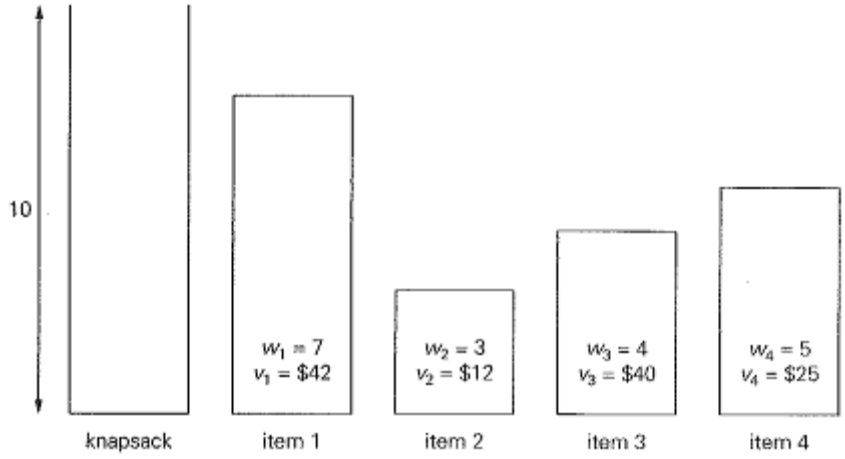
UE19CS251: Design and Analysis of Algorithms (4-0-0-4-4)

Exhaustive Search: Knapsack Problem

Dr. Shylaja S S

Knapsack Problem

Given n items of known weights $w_1, \ldots, w_n$ and values $v_1, \ldots, v_n$, and a knapsack of capacity W, find the most valuable subset of the items that fit into the knapsack. If you do not like the idea of putting yourself in the shoes of a thief who wants to steal the most valuable loot that fits into his knapsack, think about a transport plane that has to deliver the most valuable set of items to a remote location without exceeding the plane's capacity. Fig. 1 presents a small instance of the knapsack problem.



(a)

Subset	Total weight	Total value
∅	0	$0
{1}	7	$42
{2}	3	$12
{3}	4	$40
{4}	5	$25
{1, 2}	10	$36
{1, 3}	11	not feasible
{1, 4}	12	not feasible
{2, 3}	7	$52
{2, 4}	8	$37
{3, 4}	**9**	**$65**
{1, 2, 3}	14	not feasible
{1, 2, 4}	15	not feasible
{1, 3, 4}	16	not feasible
{2, 3, 4}	12	not feasible
{1, 2, 3, 4}	19	not feasible

(b)

Fig. 1: (a) Instance of the knapsack problem. (b) Its solution by exhaustive search. (The information about the optimal selection is in bold.)

The exhaustive-search approach to this problem leads to generating all the subsets of the set of n items given, computing the total weight of each subset to identify feasible subsets (i.e., the ones with the total weight not exceeding the knapsack's capacity), and finding a subset of the largest value among them. As an example, the solution to the instance of Fig. 1a is given in Fig. 1b. Since the number of subsets of an n-element set is 2^n, the exhaustive search leads to a $\Omega(2^n)$ algorithm no matter how efficiently individual subsets are generated.

Thus, for both the traveling salesman and knapsack problems, exhaustive search leads to algorithms that are extremely inefficient on every input. In fact, these two problems are the best-known examples of so-called NP-hard problems. No polynomial-time algorithm is known for any NP-hard problem. Moreover, most computer scientists believe that such algorithms do not exist, although this very important conjecture has never been proven. More sophisticated approaches - backtracking and branch-and-bound enable us to solve some but not all instances of these (and similar) problems in less than exponential time. Alternatively, we can use one of many approximation algorithms.