Report...

Marcos Daniel Baroni

November 19, 2014

$$\begin{aligned} \text{maximize} &\sum_{j=1}^{n} p_j x_j \\ \text{subject to} &\sum_{j=1}^{n} w_j x_j \leq b \\ & x_j \in \{0,1\}, \quad j \in \{1,\dots,n\}. \end{aligned}$$

1

2 The Backtracking Approach

[3] [2]

$$\frac{p_1}{w_1} \geqslant \frac{p_2}{w_2} \geqslant \dots \geqslant \frac{p_n}{w_n}$$

3 The Nemhauser-Ullmann Algorithm

A brute force method to solve the knapsack problem is to enumerate all possible subsets over the n items. In order to reduce the search space, a domination concept is used which is usually attributed to Weingartner and Ness [5].

Definition 1 (Domination) *A subset $S \in [n]$ with weight $w(S) = \sum_{i \in S} w_i$ and profit $p(S) = \sum_{i \in S} p_i$ **dominates** another subset $T \subseteq [n]$ if $w(S) \leqslant w(T)$ and $p(S) \geqslant p(T)$.*

For simplicity assume that no two subsets have the same profit. Then no subset dominated by another subset can be an optimal solution to the knapsack problem, regardless of the specified knapsack capacity. Consequently, it suffices to consider those sets that are not dominated by any other set.

Using this concept Nemhauser and Ullmann [4] introduced an elegant algorithm (Algorithm 1) computing a list of all dominating sets in an iterative manner.

Algorithm 1: The Nemhauser-Ullmann Algorithm

Input : a KP instance
Output: list $S(n)$ of all dominating sets

1 $S(0) \leftarrow \emptyset$;
2 **for** $i \leftarrow 1$ **to** n **do**
3 $\quad S'(i) \leftarrow S(i-1) \cup \{s \cup \{i\} \mid s \in S(i-1)\}$;
4 $\quad S(i) \leftarrow \{s \in S'(i) \mid \textit{dominates}(s, S'(i))\}$;
5 **end**

This algorithm can be viewed as a sparse dynamic programming approach which, at each iteration duplicates all subsets in $S(i-1)$ and then adds item i to each of the duplicated subsets (line 3). The $\textit{dominates}(s, S)$ procedure checks if subset s dominates all others subsets in S. The dominated subsets are then *filtered* (line 4). The result is the ordered sequence $S(n)$ of dominating subsets over the items $1, \ldots, n$.

Figure 1 graphically represents profit and weight values for dominating sets of (a) an intermediate solution $S(i)$, (b) its next solution $S(i+1)$ and (c) an optimal solution for a small random instance.

If we denote $q(i)$ the upper bound on the number of dominating sets over items in $1, \ldots, i$, at each iteration the algorithm computes $S(i)$ over $S(i-1)$ in $O(q(i))$ time. The total running time of the algorithm is then $O(n \cdot q(n))$. Now the challenge in the analysis is to estimate the number of dominating sets.

Beier and Vöcking [1] addressed this analysis considering sets of items with adversary weights and randomly drawn profits. They could deduce that for the uniform distribution, for example, the expected number of dominating sets is $E[q] = O(n^3)$ leading to an expected running time of $O(n^4)$.

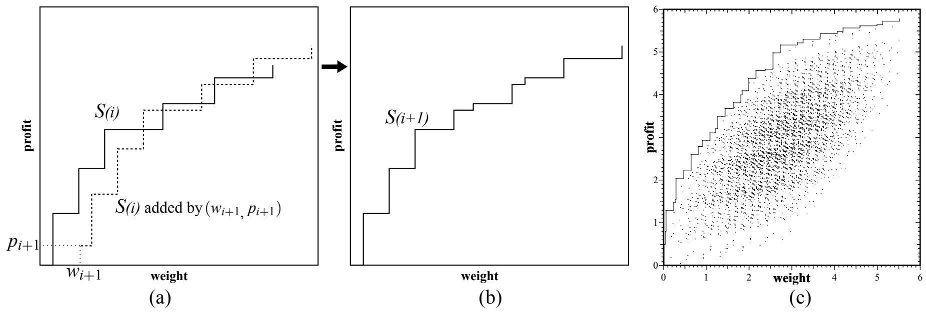

Figure 1: Graphical representation of dominating sets for (a) an intermediate solution $S(i)$, (b) its next solution $S(i+1)$. and (c) an optimal solution for a small random instance.

References

[1] R. Beier and B. Vöcking. Random knapsack in expected polynomial time. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 232–241. ACM, 2003.

[2] E. Horowitz and S. Sahni. *Fundamentals of computer algorithms.* Computer Science Press, 1978.

[3] G. S. Lueker. Average-case analysis of off-line and on-line knapsack problems. *Journal of Algorithms*, 29(2):277–305, 1998.

[4] G. L. Nemhauser and Z. Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15(9):494–505, 1969.

[5] H. M. Weingartner and D. N. Ness. Methods for the solution of the multidimensional 0/1 knapsack problem. *Operations Research*, 15(1):83–103, 1967.