8 Knapsack

In Chapter 1 we mentioned that some **NP**-hard optimization problems allow approximability to any required degree. In this chapter, we will formalize this notion and will show that the knapsack problem admits such an approximability.

Let Π be an **NP**-hard optimization problem with objective function f_Π. We will say that algorithm $\mathcal{A}$ is an *approximation scheme* for Π if on input (I, ε), where I is an instance of Π and $\varepsilon > 0$ is an error parameter, it outputs a solution s such that:

- $f_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$ if Π is a minimization problem.
- $f_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$ if Π is a maximization problem.

$\mathcal{A}$ will be said to be a *polynomial time approximation scheme*, abbreviated PTAS, if for each *fixed* $\varepsilon > 0$, its running time is bounded by a polynomial in the size of instance I.

The definition given above allows the running time of $\mathcal{A}$ to depend arbitrarily on ε. This is rectified in the following more stringent notion of approximability. If the previous definition is modified to require that the running time of $\mathcal{A}$ be bounded by a polynomial in the size of instance I and $1/\varepsilon$, then $\mathcal{A}$ will be said to be a *fully polynomial approximation scheme*, abbreviated FPTAS.

In a very technical sense, an FPTAS is the best one can hope for an **NP**-hard optimization problem, assuming $\mathbf{P} \neq \mathbf{NP}$; see Section 8.3.1 for a short discussion on this issue. The knapsack problem admits an FPTAS.

Problem 8.1 (Knapsack) Given a set $S = \{a_1, \ldots, a_n\}$ of objects, with specified sizes and profits, $\mathrm{size}(a_i) \in \mathbf{Z}^+$ and $\mathrm{profit}(a_i) \in \mathbf{Z}^+$, and a "knapsack capacity" $B \in \mathbf{Z}^+$, find a subset of objects whose total size is bounded by B and total profit is maximized.

An obvious algorithm for this problem is to sort the objects by decreasing ratio of profit to size, and then greedily pick objects in this order. It is easy to see that as such this algorithm can be made to perform arbitrarily badly (Exercise 8.1).

8.1 A pseudo-polynomial time algorithm for knapsack

Before presenting an FPTAS for knapsack, we need one more concept. For any optimization problem Π, an instance consists of *objects*, such as sets or graphs, and *numbers*, such as cost, profit, size, etc. So far, we have assumed that all numbers occurring in a problem instance I are written in binary. The *size* of the instance, denoted $|I|$, was defined as the number of bits needed to write I under this assumption. Let us say that I_u will denote instance I with all numbers occurring in it written in unary. The *unary size* of instance I, denoted $|I_u|$, is defined as the number of bits needed to write I_u.

An algorithm for problem Π is said to be efficient if its running time on instance I is bounded by a polynomial in $|I|$. Let us consider the following weaker definition. An algorithm for problem Π whose running time on instance I is bounded by a polynomial in $|I_u|$ will be called a *pseudo-polynomial time algorithm.*

The knapsack problem, being **NP**-hard, does not admit a polynomial time algorithm; however, it does admit a pseudo-polynomial time algorithm. This fact is used critically in obtaining an FPTAS for it. All known pseudo-polynomial time algorithms for **NP**-hard problems are based on dynamic programming.

Let P be the profit of the most profitable object, i.e., $P = \max_{a \in S} \text{profit}(a)$. Then nP is a trivial upperbound on the profit that can be achieved by any solution. For each $i \in \{1, \ldots, n\}$ and $p \in \{1, \ldots, nP\}$, let $S_{i,p}$ denote a subset of $\{a_1, \ldots, a_i\}$ whose total profit is exactly p and whose total size is minimized. Let $A(i,p)$ denote the size of the set $S_{i,p}$ ($A(i,p) = \infty$ if no such set exists). Clearly $A(1,p)$ is known for every $p \in \{1, \ldots, nP\}$. The following recurrence helps compute all values $A(i,p)$ in $O(n^2P)$ time:

$$A(i+1,p) = \begin{cases} \min\{A(i,p),\ \text{size}(a_{i+1}) + A(i, p - \text{profit}(a_{i+1}))\} & \text{if profit}(a_{i+1}) < p \\ A(i+1,p) = A(i,p) & \text{otherwise} \end{cases}$$

The maximum profit achievable by objects of total size bounded by B is $\max\{p|\ A(n,p) \leq B\}$. We thus get a pseudo-polynomial algorithm for knapsack.

8.2 An FPTAS for knapsack

Notice that if the profits of objects were small numbers, i.e., they were bounded by a polynomial in n, then this would be a regular polynomial time algorithm, since its running time would be bounded by a polynomial in $|I|$. The key idea behind obtaining an FPTAS is to exploit precisely this fact: we will ignore a certain number of least significant bits of profits of objects (depending on the error parameter ε), so that the modified profits can be viewed as numbers bounded by a polynomial in n and $1/\varepsilon$. This will enable us to find a solution whose profit is at least $(1-\varepsilon)\cdot \mathrm{OPT}$ in time bounded by a polynomial in n and $1/\varepsilon$.

Algorithm 8.2 (FPTAS for knapsack)

1. Given $\varepsilon > 0$, let $K = \frac{\varepsilon P}{n}$.
2. For each object a_i, define $\mathrm{profit}'(a_i) = \left\lfloor \frac{\mathrm{profit}(a_i)}{K} \right\rfloor$.
3. With these as profits of objects, using the dynamic programming algorithm, find the most profitable set, say S'.
4. Output S'.

Lemma 8.3 *Let A denote the set output by the algorithm. Then,*

$$\mathrm{profit}(A) \geq (1-\varepsilon)\cdot \mathrm{OPT}.$$

Proof: Let O denote the optimal set. For any object a, because of rounding down, $K \cdot \mathrm{profit}'(a)$ can be smaller than $\mathrm{profit}(a)$, but by not more than K. Therefore,

$$\mathrm{profit}(O) - K \cdot \mathrm{profit}'(O) \leq nK.$$

The dynamic programming step must return a set at least as good as O under the new profits. Therefore,

$$\begin{aligned}\mathrm{profit}(S') &\geq K \cdot \mathrm{profit}'(O) \geq \mathrm{profit}(O) - nK = \mathrm{OPT} - \varepsilon P \\ &\geq (1-\varepsilon)\cdot \mathrm{OPT}\ ,\end{aligned}$$

where the last inequality follows from the observation that $\mathrm{OPT} \geq P$. □

Theorem 8.4 *Algorithm 8.2 is a fully polynomial approximation scheme for knapsack.*

Proof: By Lemma 8.3, the solution found is within $(1-\varepsilon)$ factor of OPT. Since the running time of the algorithm is $O\left(n^2 \left\lfloor \frac{P}{K} \right\rfloor\right) = O\left(n^2 \left\lfloor \frac{n}{\varepsilon} \right\rfloor\right)$, which is polynomial in n and $1/\varepsilon$, the theorem follows. □