Approximation Algorithms I

The knapsack problem

- Input: nonnegative numbers $p_1, \dots, p_n, a_1, \dots, a_n, b$.

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} p_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_j x_j \le b \\ & x \in \mathbb{Z}_+^n \end{aligned}$$

Additive performance guarantees

Theorem 1. *There is a polynomial-time algorithm A for the knapsack problem such that*

$$A(I) \ge OPT(I) - K \quad \textit{for all instances } I \tag{1}$$

for some constant K if and only if $P = NP$.

Proof:

- Let A be a polynomial-time algorithm satisfying (1).
- Let $I = (p_1, \dots, p_n, a_1, \dots, a_n, b)$ be an instance of the knapsack problem.
- Let $I' = (p_1' := (K+1)p_1, \dots, p_n' := (K+1)p_n, a_1, \dots, a_n, b)$ be a new instance.
- Clearly, x^* is optimal for I iff it is optimal for I'.
- If we apply A to I' we obtain a solution x' such that

$$p'x^* - p'x' \le K.$$

- Hence,

$$px^* - px' = \frac{1}{K+1}(p'x^* - p'x') \le \frac{K}{K+1} < 1.$$

- Since px' and px^* are integer, it follows that $px' = px^*$, that is x' is optimal for I.
- The other direction is trivial. $\square$

- Note that this technique applies to *any* combinatorial optimization problem with linear objective function.

Approximation algorithms

- There are few (known) NP-hard problems for which we can find in polynomial time solutions whose value is close to that of an optimal solution in an absolute sense. (Example: edge coloring.)

- In general, an approximation algorithm for an optimization Π produces, in polynomial time, a feasible solution whose objective function value is within a guaranteed factor of that of an optimal solution.

A first greedy algorithm for the knapsack problem

1. Rearrange indices so that $p_1 \geq p_2 \geq \cdots \geq p_n$.
2. FOR $j = 1$ TO n DO
3. set $x_j := \left\lfloor \frac{b}{a_j} \right\rfloor$ and $b := b - \left\lfloor \frac{b}{a_j} \right\rfloor$.
4. Return x.

- This greedy algorithm can produce solutions that are arbitrarily bad.

- Consider the following example, with $\alpha \geq 2$:

$$\begin{array}{llcll} \max & \alpha x_1 & + & (\alpha - 1)x_2 & \\ \text{s.t.} & \alpha x_1 & + & x_2 & \leq \alpha \\ & & & x_1, x_2 & \in \mathbb{Z}_+ \end{array}$$

- Obviously, $\mathrm{OPT} = \alpha(\alpha - 1)$ and $\mathrm{GREEDY}_1 = \alpha$.

- Hence,
$$\frac{\mathrm{GREEDY}_1}{\mathrm{OPT}} = \frac{1}{\alpha - 1} \to 0.$$

A second greedy algorithm for the knapsack problem

1. Rearrange indices so that $p_1/a_1 \geq p_2/a_2 \geq \cdots \geq p_n/a_n$.
2. FOR $j = 1$ TO n DO
3. set $x_j := \left\lfloor \frac{b}{a_j} \right\rfloor$ and $b := b - \left\lfloor \frac{b}{a_j} \right\rfloor$.
4. Return x.

Theorem 2. *For all instances I of the knapsack problem,*

$$\mathrm{GREEDY}_2(I) \geq \frac{1}{2}\,\mathrm{OPT(I)}.$$

Proof:

- We may assume that $a_1 \leq b$.
- Let x be the greedy solution, and let x^* be an optimal solution.
- Obviously,
$$px \geq p_1 x_1 = p_1 \left\lfloor \frac{b}{a_1} \right\rfloor.$$
- Also,
$$px^* \leq p_1 \frac{b}{a_1} \leq p_1 \left(\left\lfloor \frac{b}{a_1} \right\rfloor + 1 \right) \leq 2p_1 \left\lfloor \frac{b}{a_1} \right\rfloor \leq 2px.$$

$\square$

- This analysis is tight.
- Consider the following example:

$$\begin{array}{lrcll} \max & 2\alpha x_1 & + & 2(\alpha-1)x_2 & \\ \text{s.t.} & \alpha x_1 & + & (\alpha-1)x_2 & \leq 2(\alpha-1) \\ & & x_1, x_2 & & \in \mathbb{Z}_+ \end{array}$$

- Obviously, $p_1/a_1 \geq p_2/a_2$, and $\text{GREEDY}_2 = 2\alpha$ whereas $\text{OPT} = 4(\alpha - 1)$. Hence,
$$\frac{\text{GREEDY}_2}{\text{OPT}} = \frac{2\alpha}{4(\alpha-1)} \to \frac{1}{2}.$$

The 0/1-knapsack problem

- Input: nonnegative numbers $p_1, \ldots, p_n, a_1, \ldots, a_n, b$.

$$\begin{array}{rl} \max & \displaystyle\sum_{j=1}^{n} p_j x_j \\ \text{s.t.} & \displaystyle\sum_{j=1}^{n} a_j x_j \leq b \\ & x \in \{0,1\}^n \end{array}$$

A greedy algorithm for the 0/1-knapsack problem

1. Rearrange indices so that $p_1/a_1 \geq p_2/a_2 \geq \cdots \geq p_n/a_n$.
2. FOR $j = 1$ TO n DO
3. IF $a_j > b$, THEN $x_j := 0$
4. ELSE $x_j := 1$ and $b := b - a_j$.

5. Return x.

- The greedy algorithm can be arbitrarily bad for the 0/1-knapsack problem.
- Consider the following example:

$$\begin{array}{lcrcl} \max & & x_1 & + & \alpha x_2 & \\ \text{s.t.} & & x_1 & + & \alpha x_2 & \leq \alpha \\ & & & & x_1, x_2 & \in \{0,1\} \end{array}$$

- Note that $\mathrm{OPT} = \alpha$, whereas $\mathrm{GREEDY}_2 = 1$.
- Hence,
$$\frac{\mathrm{GREEDY}_2}{\mathrm{OPT}} = \frac{1}{\alpha} \to 0.$$

Theorem 3. *Given an instance I of the 0/1 knapsack problem, let*

$$A(I) := \max\left\{\mathrm{GREEDY}_2(I), p_{\max}\right\},$$

where $p_{\max}$ is the maximum profit of an item. Then

$$A(I) \geq \frac{1}{2}\,\mathrm{OPT}(I).$$

Proof:

- Let j be the first item not included by the greedy algorithm.
- The profit at that point is
$$\bar{p}_j := \sum_{i=1}^{j-1} p_i \leq \mathrm{GREEDY}_2.$$
- The overall occupancy at this point is
$$\bar{a}_j := \sum_{i=1}^{j-1} \leq b.$$
- We will show that
$$\mathrm{OPT} \leq \bar{p}_j + p_j.$$
(If this is true, we are done.)

- Let x^* be an optimal solution. Then:

$$\begin{aligned}
\sum_{i=1}^{n} p_i x_i^* &\le \sum_{i=1}^{j-1} p_i x_i^* + \sum_{i=j}^{n} \frac{p_j a_i}{a_j} x_i^* \\
&= \frac{p_j}{a_j} \sum_{i=1}^{n} a_i x_i^* + \sum_{i=1}^{j-1} \big(p_i - \frac{p_j}{a_j} a_i\big) x_i^* \\
&\le \frac{p_j}{a_j} b + \sum_{i=1}^{j-1} \big(p_i - \frac{p_j}{a_j} a_i\big) \\
&= \sum_{i=1}^{j-1} p_i + \frac{p_j}{a_j} \big(b - \sum_{i=1}^{j-1} a_i\big) \\
&= \bar{p}_j + \frac{p_j}{a_j} \big(b - \bar{a}_j\big)
\end{aligned}$$

- Since $\bar{a}_j + a_j > b$, we obtain

$$\mathrm{OPT} = \sum_{i=1}^{n} p_i x_i^* \le \bar{p}_j + \frac{p_j}{a_j} \big(b - \bar{a}_j\big) < \bar{p}_j + p_j.$$

$\square$

- Recall that there is an algorithm that solves the 0/1-knapsack problem in $\mathrm{O}(n^2 p_{\max})$ time:

- Let $f(i, q)$ be the size of the subset of $\{1, \ldots, i\}$ whose total profit is q and whose total size is minimal.

- Then

$$f(i+1, q) = \min\big\{f(i, q), a_{i+1} + f(i, q - p_{i+1})\big\}.$$

- We need to compute $\max\{q : f(n, q) \le b\}$.

- In particular, if the profits of items were small numbers (i.e., bounded by a polynomial in n), then this would be a regular polynomial-time algorithm.

An FPTAS for the 0/1-knapsack problem

1. Given $\epsilon > 0$, let $K := \frac{\epsilon p_{\max}}{n}$.

2. FOR $j = 1$ TO n DO $p'_j := \left\lfloor \frac{p_j}{K} \right\rfloor$.

3. Solve the instance $(p'_1, \ldots, p'_n, a_1, \ldots, a_n, b)$ using the dynamic program.

4. Return this solution.

Theorem 4. *This algorithm is a Fully Polynomial-Time Approximation Scheme for the 0/1-knapsack problem.*

That is, given an instance I and an $\epsilon > 0$, it finds in time polynomial in the input size of I and $1/\epsilon$ a solution x' such that

$$px' \geq (1 - \epsilon)px^*.$$

Proof:

- Note that $p_j - K \leq Kp'_j \leq p_j$.
- Hence, $px^* - Kp'x^* \leq nK$.
- Moreover,

$$px' \geq Kp'x' \geq Kp'x^* \geq px^* - nK = px^* - \epsilon p_{\max} \geq (1 - \epsilon)px^*.$$

□

Fully Polynomial Time Approximation Schemes

- Let Π be an optimization problem. Algorithm A is an approximation scheme for Π if on input (I, ϵ), where I is an instance of Π and $\epsilon > 0$ is an error parameter, it outputs a solution of objective function value $A(I)$ such that
  - $A(I) \leq (1 + \epsilon)\mathrm{OPT}(I)$ if Π is a minimization problem.
  - $A(I) \geq (1 - \epsilon)\mathrm{OPT}(I)$ if Π is a maximization problem.
- A is a *polynomial-time approximation scheme (PTAS)*, if for each fixed $\epsilon > 0$, its running time is bounded by a polynomial in the size of I.
- A is a *fully polynomial-time approximation scheme (FPTAS)*, if its running time is bounded by a polynomial in the size of I and $1/\epsilon$.

Theorem 5. *Let p be a polynomial and let Π be an NP-hard minimization problem with integer-valued objective function such that on any instance $I \in \Pi$, $\mathrm{OPT}(I) < p(|I|_u)$. If Π admits an FPTAS, then it also admits a pseudopolynomial-time algorithm.*

Proof:

- Suppose there is an FPTAS with running time $q(|I|, 1/\epsilon)$, for some polynomial q.
- Choose $\epsilon := 1/p(|I|_u)$ and run the FPTAS.
- The solution has objective function value at most

$$(1 + \epsilon)\mathrm{OPT}(I) < \mathrm{OPT}(I) + \epsilon p(|I|_u) = \mathrm{OPT}(I) + 1.$$

- Hence, the solution is optimal.
- The running time is $q(|I|, p(|I|_u))$, i.e., polynomial in $|I|_u$. □

Corollary 6. *Let Π be an NP-hard optimization problem satisfying the assumptions of the previous theorem. If Π is strongly NP-hard, then Π does not admit an FPTAS, assuming $\mathrm{P} \neq \mathrm{NP}$.*