Knapsack

Bag/knapsack of integer capacity B

n items

item i has size s_i and profit/weight w_i

Goal: find a subset of items of maximum profit such that the item subset fits in the bag

Knapsack

X: item set

for $A \subseteq X$

$s(A) = \sum_{i \in A} s_i$

$w(A) = \sum_{i \in A} w_i$

Goal: $\max_{A \subseteq X,\, s(A) \leq B} w(A)$

Knapsack

Q: what if $w(i) = 1$ for all i ?

Q: what is a good heuristic for the general case?

Knapsack

Q: what if $w(i) = 1$ for all i ?

Q: what is a good heuristic for the general case?

Greedy algorithm:

Sort items s.t $w_1/s_1 \geq w_2/s_2 \ldots \geq w_n/s_n$

Insert in sorted order discarding an item if it does not fit

Bad example for Greedy

Two items

$s_1 = \varepsilon$, $w_1 = 2\varepsilon$

$s_2 = B$, $w_2 = B$

Modified Greedy:

Pick best of Greedy and largest profit item

Theorem: Modified Greedy is a 1/2 approximation

Modified Greedy Analysis

Let k be the first item discarded by Greedy

Lemma: $w_1 + w_2 + \ldots + w_k \geq OPT$

$\alpha = (B - (s_1 + s_2 + \ldots + s_{k-1}))/s_k$
(fraction of item k that we could potentially pack)

Stronger lemma:
$w_1 + w_2 + \ldots + w_{k-1} + \alpha\, w_k \geq OPT$

Proof of Lemma

Consider a relaxation. Allow items to be fractionally packed

$x_i \in [0, 1]$ a fractional amount by which item i is packed in the knapsack

$x_i = 1$ implies item i is fully packed

$x_i = 0$ implies item i is not packed

Packing constraint: $\sum_i s_i x_i \leq B$

maximize $\sum_i w_i x_i$

Proof of Lemma

Therefore

$OPT \leq OPT'$

where

$OPT' = \max \sum_i w_i x_i$

s.t

$\sum_i s_i x_i \leq B$

$x_i \in [0,1]$

Lemma: $w_1 + \ldots + w_{k-1} + \alpha\, w_k \geq OPT'$

Proof of Lemma

Lemma: $w_1 + \ldots + w_{k-1} + \alpha\, w_k \geq OPT'$

Proof:

$x_1^* = x_2^* = \ldots x_{k-1}^* = 1$

$x_k^* = \alpha$

$x_i^* = 0$ for $i > k$

is an optimum solution to fractional relaxation (why?)

Corollary

Suppose all $s_i \leq \varepsilon B$

Greedy's solution $\geq (1-\varepsilon)$ OPT

Proof:

for $1 \leq i \leq k$

$w_i/s_i \geq w_k/s_k \Rightarrow w_i \geq s_i\, w_k/s_k$

implies

$w_1 + w_2 + \ldots + w_k \geq (s_1 + \ldots + s_k)\, w_k/s_k$

Corollary

$w_1 + w_2 + \ldots + w_k \geq (s_1+\ldots+s_k)\ w_k/s_k$

but $s_1+s_2+\ldots+s_k > B$

hence

$w_k \leq s_k\ (w_1+\ldots+w_k)/B \leq \varepsilon\ (w_1+\ldots+w_k)$

implies

$w_k \leq \varepsilon\ (w_1+\ldots+w_{k-1})/(1-\varepsilon)$

therefore

$w_1+w_2 + \ldots + w_{k-1} \geq (1-\varepsilon)\ OPT$

General Principle for Packing Probs

Items/object sizes small comparted to capacity

$\Longrightarrow$

problem is easier (to approximate)

Know your problem instance!

Improving upon Greedy

Obs: Greedy $\geq$ (1-ε) OPT if wmax $\leq$ ε OPT

Fix an optimum solution S^*
# of items in S^* with weight $\geq$ ε OPT $\leq$ 1/ε

Improving upon Greedy

Obs: Greedy $\geq$ (1-ε) OPT if wmax $\leq \varepsilon$ OPT

N: item set

Guess the $h = \lceil 1/\varepsilon \rceil$ items in opt soln, say S_h

Pack S_h in knapsack

Let α be the least weight in S_h

Remove all items $i \in N - S_h$ such that $w_i > \alpha$

Apply Greedy on remaining items and remaining capacity

$B' = B - \sum_{i \in S_h} s_i$

Improving upon Greedy

Theorem: Guess_h+Greedy yields a $(1-\varepsilon)$ approximation

Proof: exercise

Running time? $O(n \log n + n^{1+1/\varepsilon})$

(dominated by time to guess (enumerate) S_h)

Polynomial time approximation scheme (PTAS)

Collection of algorithms:

For every *fixed* $\varepsilon > 0$

- a $(1+\varepsilon)$ approximation (for minimization) and $(1-\varepsilon)$ for maximization
- running time: polynomial in input size

$\text{poly}(1/\varepsilon)\ \text{poly}(n)$

$n^{\text{poly}(1/\varepsilon)}$

$f(1/\varepsilon)\ \text{poly}(n)$

$n^{f(1/\varepsilon)}$

f can be exponential or worse: $2^{1/\varepsilon}$ …

An even better alg for Knapsack?

Assume sizes, profits, B are integers

$W = \sum_i w_i$

Exercise: use dynamic programming to obtain an exact algorithm in $O(nB)$ time or in $O(nW)$ time

Pseudo-polynomial time algorithms

Polynomial in input size if numbers are in *unary*

Scaling and rounding

If W or B is *small* (polynomial in n) algs yield exact solution!

Can we transform any instance into one with *small* P/W?

Clearly have to lose some in approximation, otherwise exact algorithm for NP-hard problem

Scaling and Rounding for Knapsack

Scale profits of items: $w'_i = w_i \, n/(\varepsilon \, wmax)$
Round profits: $w''_i = \lfloor w'_i \rfloor$

Observation: w''_i is an integer in $[0, n/\varepsilon]$

$W'' = \sum_i w''_i \leq n^2/\varepsilon$

FPTAS for Knapsack

Use dynamic prog to solve exactly using w''_i

Running time: $O(n^3/\varepsilon)$

Let A be the set of items packed by dyn prog

Output A for the original problem

Theorem: $w(A) \geq (1-\varepsilon)$ OPT

Proof of Theorem

let $\alpha = n/(\varepsilon \, wmax)$ (the scaling factor)

for any subset X of items

$\alpha\, w(X) - |X| \leq w''(X) \leq \alpha\, w(X)$

$\Rightarrow$

$\alpha\, w(X) - n \leq w''(X) \leq \alpha\, w(X)$

Proof of Theorem

Let A^* be some optimum solution for original instance

Clearly A^* is a feasible solution for the instance with modified weights w''

therefore

$w''(A^*) \leq w''(A)$

we have that $w''(A^*) \leq \alpha\, w(A^*)$

and $w''(A) \geq \alpha\, w(A) - n$

Proof of Theorem

$w''(A^*) \le w''(A)$

we have that $w''(A) \le \alpha\, w(A)$
and $w''(A^*) \ge \alpha\, w(A^*) – n$

$\alpha\, w(A^*) – n \le \alpha\, w(A)$
$\Rightarrow$
$w(A) \ge w(A^*) – n/$\alpha
$\quad \ge w(A^*) - \varepsilon\, wmax$

Proof of Theorem

$w(A) \geq w(A^*) - n/\alpha$

$\geq w(A^*) - \varepsilon\, wmax$

$\geq OPT - \varepsilon\, wmax$

$\geq OPT - \varepsilon\, OPT$

$\geq (1-\varepsilon)\, OPT$

Conclusion: $(1-\varepsilon)$ approx in $O(n^3/\varepsilon)$ time

FPTAS implementation in $O(n/\varepsilon + 1/\varepsilon^4)$ time [Lawler]

FPTAS

FPTAS: fully polynomial time approximation scheme

PTAS with running time poly(n, 1/ε)

FPTAS and strong NP-hardness

Theorem: If Π is a strongly NP-hard problem then Π cannot have an FPTAS

(Why?)

Corollary: Π has an FPTAS implies Π has a pseudo-polynomial time algorithm

Observation: converse of corollary is false

There exist problems that have pseudo-poly algs but do not have an FPTAS

Example

Observation: converse of corollary is false

There exist problems that have pseudo-poly algs but do not have an FPTAS

Consider multiple knapsack problem with 2 bins

Prove that it admits a pseudo-poly time algorithm

Prove that an FPTAS for it will imply an exact algorithm for the Partition problem