Quality vs Time

Polynomial Time Approximation Schemes

Outline

- PTAS and FPTAS
  - Polynomial Time Algorithms for NP-Hard Problems
- Knapsack
  - PTAS
  - FPTAS (exercise session)

Outline

- PTAS and FPTAS
  - Polynomial Time Algorithms for NP-Hard Problems
- Knapsack
  - PTAS
  - FPTAS (exercise session)

Learning Objectives:
You are able to
1. restate the definition of a PTAS and an FPTAS
2. describe a PTAS algorithm for Knapsack
3. state the runtime of the PTAS algorithm for Knapsack

Polynomial Time Approximation Schemes

Hard Problems:
We believe that NP-hard problems cannot be solved (exactly) in polynomial time.

Approximation:
For example, there is a greedy 2-approximation to Knapsack.

Polynomial Time Approximation Schemes

Hard Problems:
We believe that NP-hard problems cannot be solved (exactly) in polynomial time.

Approximation:
For example, there is a greedy 2-approximation to Knapsack.

Can we do better?

Polynomial Time Approximation Schemes

Hard Problems:
We believe that NP-hard problems cannot be solved (exactly) in polynomial time.

Approximation:
For example, there is a greedy 2-approximation to Knapsack.

Can we do better?

Trade time for approximation ratio:

Polynomial time approximation scheme (PTAS):
A $(1+\epsilon)$-approximation in time $n^{O(f(\epsilon))}$, where $f(\epsilon)$ is allowed to be any function of ϵ.

Polynomial Time Approximation Schemes

Hard Problems:
We believe that NP-hard problems cannot be solved (exactly) in polynomial time.

Approximation:
For example, there is a greedy 2-approximation to Knapsack.

Can we do better?

Trade time for approximation ratio:

Polynomial time approximation scheme (PTAS):
A $(1+\epsilon)$-approximation in time $n^{O(f(\epsilon))}$, where $f(\epsilon)$ is allowed to be any function of ϵ.

Fully polynomial time approximation scheme (FPTAS):
A $(1+\epsilon)$-approximation in time poly$(n, 1/\epsilon)$

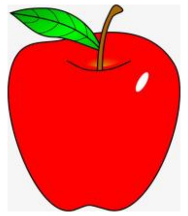
Outline

- PTAS and FPTAS
  - Polynomial Time Algorithms for NP-Hard Problems
- Knapsack
  - PTAS
  - FPTAS (exercise session)

The Knapsack Problem

Sauerkraut
Value v_1: 1
Weight w_1: 7

Chocolate
Value v_2: 4
Weight w_2: 3



Apple
Value: v_3: 5
Weight: w_3: 4

Capacity C

Maximize S $\Sigma_{S\subseteq I} V(S)$

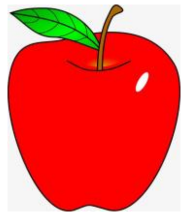
Subject to $W(S) \leq C$

$V(S)$: sum of values
$W(S)$: sum of weights

The Knapsack Problem

Sauerkraut
Value v_1: 1
Weight w_1: 7

Chocolate
Value v_2: 4
Weight w_2: 3



Apple
Value: v_3: 5
Weight: w_3: 4

Set of items I
Bounded Knapsack:
You can pick each item at most once.

Capacity C

Maximize S $\quad \Sigma_{S \subseteq I} V(S)$

Subject to $\quad W(S) \leq C$

$V(S)$: sum of values
$W(S)$: sum of weights

The Knapsack Problem

Plan:
Suppose we know the cost OPT of the optimum solution S^*.

1. We can guess all the items with value at least $\epsilon \cdot \text{OPT}$ in S^*.
2. If the largest value of any item is at most $\epsilon \cdot \text{OPT}$, then greedy is a $(1+\epsilon)$ $-$approximation.

The Knapsack Problem

Plan:
Suppose we know the cost OPT of the optimum solution S^*.

1. We can guess all the items with value at least $\epsilon \cdot \text{OPT}$ in S^*.
2. If the largest value of any item is at most $\epsilon \cdot \text{OPT}$, then greedy is a $(1+\epsilon)$ −approximation.

Guess the value OPT, by trying all values $(1+\epsilon)^i$, for $i := 1, \dots, \log_{1+\epsilon} \Sigma_j v_j$

Combine guessing and greedy to obtain a PTAS

The Knapsack Problem

Lemma:
If no item has value larger than $\epsilon \cdot \mathrm{OPT}$, then Greedy is a $(1-\epsilon)$-approximation.

The Knapsack Problem

Greedy:
Iteratively pick the item i with the best marginal gain v_i/w_i

Observation:
Order items descending according to the marginal gain. Suppose Greedy takes the first $k-1$ items.
Then OPT $\leq \Sigma_{i=1}^{k} v_i$

Proof (sketch):
If the greedy choices exactly fit the capacity, there is no better way to choose. Therefore, we can only lose the value of the best remaining item, i.e.,
OPT $\leq \Sigma_{i=1}^{k-1} v_i + v_k = \Sigma_{i=1}^{k} v_i$

The Knapsack Problem

Lemma:
If no item has value larger than $\epsilon \cdot \text{OPT}$, then Greedy is a $(1-\epsilon)$-approximation.

Proof:
Let v_{max} be the most valuable item.
$\text{OPT} \leq \Sigma_{i=1}^{k-1} v_i + v_k \leq \Sigma_{i=1}^{k-1} v_i + v_{\text{max}}$

$(1-\epsilon) \cdot \text{OPT} \leq \Sigma_{i=1}^{k-1} v_i$

Observation:
Order items descending according to the marginal gain. Suppose Greedy takes the first $k-1$ items.
Then $\text{OPT} \leq \Sigma_{i=1}^{k} v_i$

Proof (sketch):
If the greedy choices exactly fit the capacity, there is no better way to choose. Therefore, we can only lose the value of the best remaining item, i.e.,
$\text{OPT} \leq \Sigma_{i=1}^{k-1} v_i + v_k = \Sigma_{i=1}^{k} v_i$

The Knapsack Problem

Observation:
The optimum solution has at most $1/\epsilon$ items with value at least $\epsilon \cdot \mathrm{OPT}$

Otherwise the value of OPT is larger than OPT.

The Knapsack Problem

Algorithm (that knows OPT):
Let $\bar{I}$ be the set of items with value at most $\epsilon \cdot \text{OPT}$

For all possible set of items H, s.t., $|H| \leq 1/\epsilon$
    Consider H as a partial solution.
    Run Greedy on items in $\bar{I} \setminus H$.

Return the best solution found this way.

The Knapsack Problem

Algorithm (that knows OPT):
Let $\bar{I}$ be the set of items with value at most $\epsilon \cdot \mathrm{OPT}$

For all possible set of items H, s.t., $|H| \leq 1/\epsilon$
    Consider H as a partial solution.
    Run Greedy on items in $\bar{I} \setminus H$.

Return the best solution found this way.

Guess the set of "valuable items"

The Knapsack Problem

Algorithm (that knows OPT):
Let $\bar{I}$ be the set of items with value at most $\epsilon \cdot \mathrm{OPT}$

For all possible set of items H, s.t., $|H| \leq 1/\epsilon$
  Consider H as a partial solution.
  Run Greedy on items in $\bar{I} \setminus H$.

Return the best solution found this way.

Guess the set of "valuable items"

Select the rest greedily

The Knapsack Problem

Consider the iteration where H is ”guessed correctly”, i.e., it contains the same items with value at least $\epsilon \cdot \mathrm{OPT}$ as the optimal solution.

Now, the only loss can come from running greedy in $\bar{I} \setminus H$.

The Knapsack Problem

Consider the iteration where H is ”guessed correctly”, i.e., it contains the same items with value at least $\epsilon \cdot \text{OPT}$ as the optimal solution.

Now, the only loss can come from running greedy in $\bar{I} \setminus H$.

Lemma:
If no item has value larger than $\epsilon \cdot \text{OPT}$, then Greedy is a $(1 - \epsilon)$-approximation.

The Knapsack Problem

Consider the iteration where H is ”guessed correctly”, i.e., it contains the same items with value at least $\epsilon \cdot \text{OPT}$ as the optimal solution.

Now, the only loss can come from running greedy in $\bar{I} \setminus H$.

We get a $(1-\epsilon)$-approximation algorithm when we know OPT.

Lemma:
If no item has value larger than $\epsilon \cdot \text{OPT}$, then Greedy is a
$(1-\epsilon)$-approximation.

Exponential Guessing

We have a $(1-\epsilon)$-approximation when we know OPT

Guess all values $(1+\epsilon)^i$ up to $\Sigma_j^n v_j = F$ as the value of the optimum. Clearly OPT cannot be larger than the sum of all values.

Notice that $F = (1+\epsilon)^{O(\log F)}$

Exponential Guessing

We have a $(1-\epsilon)$-approximation when we know OPT

When we guess $i = \text{ceil}(\log \text{OPT})$, we can use the algorithm that knows OPT with another $(1+\epsilon)$-factor error.

Guess all values $(1+\epsilon)^i$ up to $\Sigma_j^n v_j = F$ as the value of the optimum. Clearly OPT cannot be larger than the sum of all values.

Notice that $F = (1+\epsilon)^{O(\log F)}$

Exponential Guessing

We have a $(1-\epsilon)$-approximation when we know OPT

When we guess $i = \text{ceil}(\log \text{OPT})$, we can use the algorithm that knows OPT with another $(1+\epsilon)$-factor error.

Guess all values $(1+\epsilon)^i$ up to $\Sigma_j^n v_j = F$ as the value of the optimum. Clearly OPT cannot be larger than the sum of all values.

Notice that $F = (1+\epsilon)^{O(\log F)}$

We get a $(1 - O(\epsilon))$-approximation without knowing OPT.

Notice that we can always re-write $\epsilon' = O(\epsilon)$ and obtain a $(1+\epsilon')$-approximation for a small ϵ'

The Runtime

Theorem:
There is a PTAS for the Knapsack problem.

Algorithm:
Guess H with size at most $1/\epsilon$ and run greedy on the rest.

The Runtime

$$F = \sum_i v_i$$

Theorem:
There is a PTAS for the Knapsack problem.

Algorithm:
Guess H with size at most $1/\epsilon$ and run greedy on the rest.

Runtime:

1. There at most $\binom{n+1}{(1/\epsilon)} \leq (n+1)^{1/\epsilon}$ subsets of n items. The +1 comes from allowing to not take an item.
2. Greedy takes polynomial time
3. Roughly $\log F$ guesses for OPT

Dominated by the $(n+1)^{1/\epsilon}$ term.

The Runtime

Remark:
The input is roughly n bits. We can describe a number of size 2^n with n bits.
Therefore, $\log F$ can be polynomial in n.
Gives at most a constant factor to the exponent.

Runtime:

1. There at most $\binom{n+1}{(1/\epsilon)} \leq (n+1)^{1/\epsilon}$ subsets of n items. The +1 comes from allowing to not take an item.
2. Greedy takes polynomial time
3. Roughly $\log F$ guesses for OPT

Dominated by the $(n+1)^{1/\epsilon}$ term.

Wrap-up

Runtime:

Need roughly $n^{\frac{1}{\epsilon}}$ guesses.

Algorithm:
Guess the set of valuable items in the optimal solution.

Run greedy on the rest.

Get a $(1+\epsilon)$-approximation.

Theorem:
There is a PTAS for the Knapsack problem.