Approximation-III

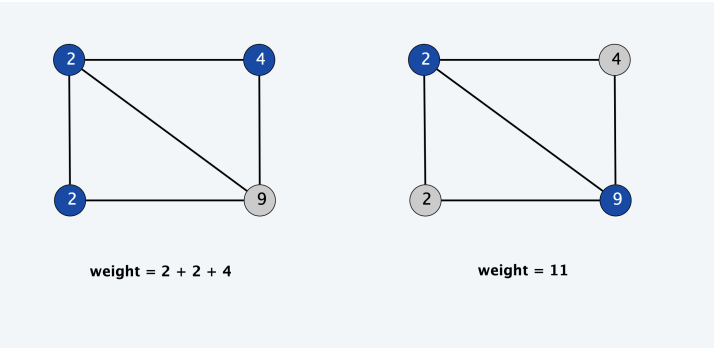
Pricing Method, LP and PTAS

Fahad Panolan

Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad, India

8 December 2022

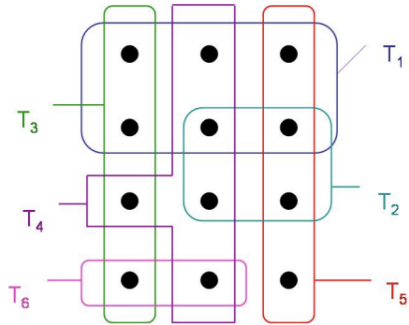
Weighted Vertex Cover



Input: A graph G and $w\colon V(G) \to \mathbb{R}_{\geq 0}$
Output: A minimum weight vertex cover of G.

Image credit: Algorithm Design by Kleinberg and Tardos

Weighted Set Cover



Input: A universe U, family $\mathcal{F}$ of subsets of U, and $w\colon \mathcal{F} \to \mathbb{R}_{\geq 0}$
Output: A minimum weight subfamily of $\mathcal{F}$ covering U.

Image credit: Lecture notes by Amit Chakrabarti

Recall

- 2-approximation for (Unweighted) Vertex Cover.
- $O(\log d^*)$-approximation for Weighted Set Cover.
- $O(\log \Delta)$-approximation for Weighted Vertex Cover

Recall

- 2-approximation for (Unweighted) Vertex Cover.
- $O(\log d^*)$-approximation for Weighted Set Cover.
- $O(\log \Delta)$-approximation for Weighted Vertex Cover

We used pricing method to analyse the algorithm for Weighted Set Cover.

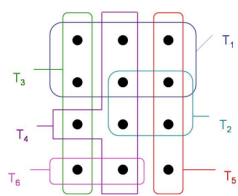
Recall

- 2-approximation for (Unweighted) Vertex Cover.
- $O(\log d^*)$-approximation for Weighted Set Cover.
- $O(\log \Delta)$-approximation for Weighted Vertex Cover

We used pricing method to analyse the algorithm for Weighted Set Cover.

Excursion Problem modeled as Weighted Set Cover

- Universe $U =$ we.
- Sets in $\mathcal{F}$ are groups who like to go together in a car/bus/train etc.
- Weight of the sets corresponds to the cost of the travel.

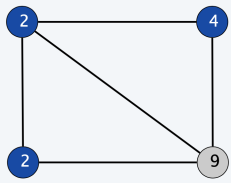


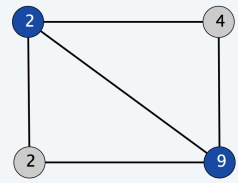
Pricing Method

with fairness and cheating

Example: 2-approximation algorithm for Vertex Cover.

Weighted Vertex Cover





Input: A graph G and $w \colon V(G) \to \mathbb{R}_{\geq 0}$
Output: A minimum weight vertex cover of G.

Image credit: Algorithm Design by Kleinberg and Tardos

Pricing:

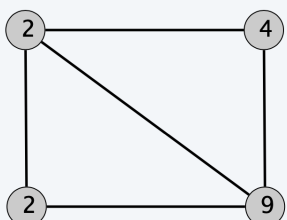
- Each edge must be covered by some vertex and there is a cost for picking a vertex

- Each edge e, pays a price $p_e \geq 0$.

Pricing:

- Each edge must be covered by some vertex and there is a cost for picking a vertex
- Each edge e, pays a price $p_e \geq 0$.

Fairness:

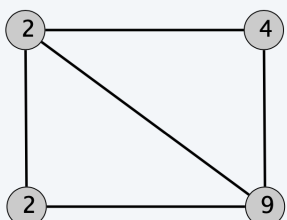


1: For an edge $e = (x, y)$, $p_e \leq \max\{w(x), w(y)\}$

Pricing:

- Each edge must be covered by some vertex and there is a cost for picking a vertex
- Each edge e, pays a price $p_e \geq 0$.

Fairness:



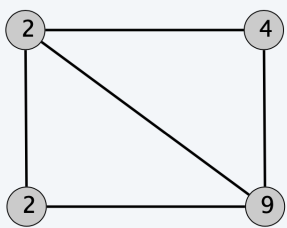
1: For an edge $e = (x, y)$, $p_e \leq \max\{w(x), w(y)\}$

2: For an edge $e = (x, y)$, $p_e \leq \min\{w(x), w(y)\}$

Pricing:

- Each edge must be covered by some vertex and there is a cost for picking a vertex
- Each edge e, pays a price $p_e \geq 0$.

Fairness:



1: For an edge $e = (x, y)$, $p_e \leq \max\{w(x), w(y)\}$

2: For an edge $e = (x, y)$, $p_e \leq \min\{w(x), w(y)\}$

3: For a vertex x, $\sum_{e=(x,z)} p_e \leq w(x)$

Fairness Lemma

Let $\{p_e \ : \ e \in E(G)\}$ be fair prices. Then for any vertex cover S, $\sum_e p_e \leq w(S)$.

Proof:

Fairness Lemma

Let $\{p_e \ : \ e \in E(G)\}$ be fair prices. Then for any vertex cover S, $\sum_e p_e \leq w(S)$.

Proof:

$$\sum_e p_e \ \leq \ \sum_{x \in S} \sum_{e=(x,z)} p_e$$

Fairness Lemma

Let $\{p_e \ : \ e \in E(G)\}$ be fair prices. Then for any vertex cover S, $\sum_e p_e \leq w(S)$.

Proof:

$$\begin{aligned}\sum_e p_e &\leq \sum_{x \in S} \sum_{e=(x,z)} p_e \\ &\leq \sum_{x \in S} w(x)\end{aligned}$$

Fairness Lemma

Let $\{p_e \ : \ e \in E(G)\}$ be fair prices. Then for any vertex cover S, $\sum_e p_e \leq w(S)$.

Proof:

$$\begin{aligned}\sum_e p_e &\leq \sum_{x \in S} \sum_{e=(x,z)} p_e \\ &\leq \sum_{x \in S} w(x) \\ &= w(S)\end{aligned}$$

Fairness Lemma

Let $\{p_e \ : \ e \in E(G)\}$ be fair prices. Then for any vertex cover S, $\sum_e p_e \leq w(S)$.

Proof:

$$\begin{aligned}\sum_e p_e &\leq \sum_{x \in S} \sum_{e=(x,z)} p_e \\ &\leq \sum_{x \in S} w(x) \\ &= w(S)\end{aligned}$$

Corollary: $\sum_e p_e \leq \mathsf{opt}$

Algorithm

Compute prices p_e and find vertex cover simultaneously.

Algorithm

Compute prices p_e and find vertex cover simultaneously.

Definition: A vertex x is tight if $\sum_{e=(x,z)} p_e = w(x)$.

Algorithm

Compute prices p_e and find vertex cover simultaneously.

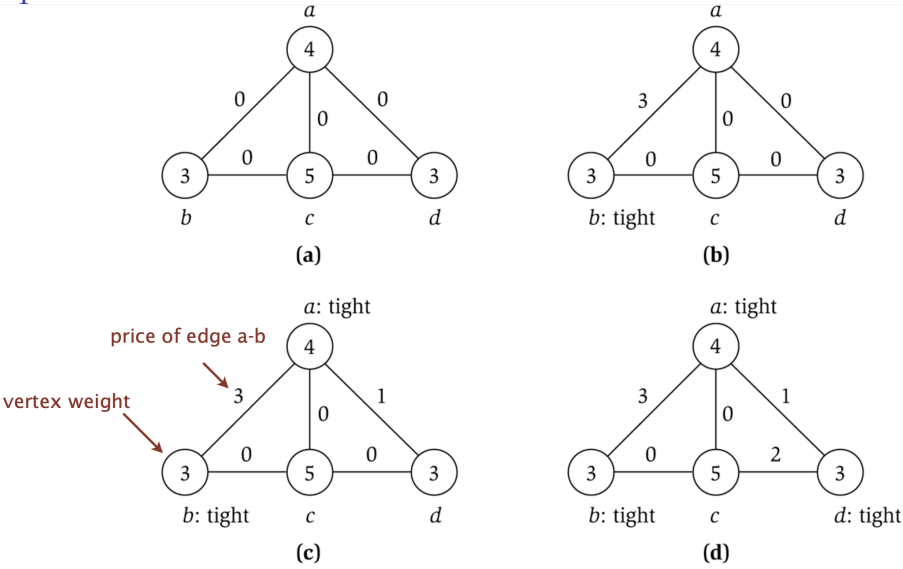
Definition: A vertex x is tight if $\sum_{e=(x,z)} p_e = w(x)$.

Initially $p_e = 0$ for all e

While(there is an edge $e = (x, y)$ such that both x and y are not tight)
    Select such an edge e and increase p_e until x is tight or y is tight.
EndWhile

Return the set S of tight vertices,.

Example


a
4
0
0
0
0
0
3
5
3
b
c
d
(a)
a
4
3
0
0
0
0
3
5
3
b: tight
c
d
(b)
a: tight
price of edge a-b
vertex weight
4
3
1
0
0
0
3
5
3
b: tight
c
d
(c)
a: tight
4
3
1
0
0
2
3
5
3
b: tight
c
d: tight
(d)

Analysis

Claim 1

Output S is a vertex cover.

Analysis

Claim 1

Output S is a vertex cover.

Proof: For each edge e, one its endpoint is tight at the end of the while loop.

Analysis

Claim 2

S is a 2-approximate solution

Analysis

Claim 2

S is a 2-approximate solution

Proof:

- Cheating: For each edge e, we ask to pay $2p_e$.

Analysis

Claim 2

S is a 2-approximate solution

Proof:

- Cheating: For each edge e, we ask to pay $2p_e$.

$$w(S) = \sum_{x \in S} w(x)$$

Analysis

Claim 2

S is a 2-approximate solution

Proof:

- Cheating: For each edge e, we ask to pay $2p_e$.

$$\begin{aligned} w(S) &= \sum_{x \in S} w(x) \\ &= \sum_{x \in S} \sum_{e=(x,z)} p_e \end{aligned}$$

Analysis

Claim 2

S is a 2-approximate solution

Proof:

- Cheating: For each edge e, we ask to pay $2p_e$.

$$\begin{aligned} w(S) &= \sum_{x \in S} w(x) \\ &= \sum_{x \in S} \sum_{e=(x,z)} p_e \\ &\leq \sum_{x \in V(G)} \sum_{e=(x,z)} p_e \end{aligned}$$

Analysis

Claim 2

S is a 2-approximate solution

Proof:

- Cheating: For each edge e, we ask to pay $2p_e$.

$$\begin{aligned} w(S) &= \sum_{x \in S} w(x) \\ &= \sum_{x \in S} \sum_{e=(x,z)} p_e \\ &\leq \sum_{x \in V(G)} \sum_{e=(x,z)} p_e \\ &\leq 2 \sum_{e \in E(G)} p_e \leq 2 \cdot \mathsf{opt} \end{aligned}$$

So far..

- $O(\log d^\star)$-approximation for Weighted Set Cover.

So far..

- $O(\log d^\star)$-approximation for Weighted Set Cover.
- 2-approximation for Weighted Set Cover.

So far..

- $O(\log d^\star)$-approximation for Weighted Set Cover.
- 2-approximation for Weighted Set Cover.
- So, Weighted Set Cover instances derived from Weighted Set Cover has a 2-approximation algorithm.

Linear Programming

Weighted Set Cover: ILP

Variable x_S for each set $S \in \mathcal{F}$.

Weighted Set Cover: ILP

Variable x_S for each set $S \in \mathcal{F}$.

$$\min \quad \sum_{S \in \mathcal{F}} w(S) \cdot x_S$$

Weighted Set Cover: ILP

Variable x_S for each set $S \in \mathcal{F}$.

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{x \in S} x_S \geq 1 \qquad \text{for all } x \in U \end{aligned}$$

Weighted Set Cover: ILP

Variable x_S for each set $S \in \mathcal{F}$.

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{x \in S} x_S \geq 1 \qquad \text{for all } x \in U \\ & x_S \in \{0,1\} \qquad \text{for all } S \in \mathcal{F} \end{aligned}$$

Feasible solutions of ILP is in one-one correspondence with the feasible solutions of Weighted Set Cover.

Weighted Set Cover: LP relaxation

Variable x_S for each set $S \in \mathcal{F}$.

$$\begin{aligned} \text{min} \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{x \in S} x_S \geq 1 \qquad \text{for all } x \in U \\ & x_S \geq 0 \qquad \text{for all } S \in \mathcal{F} \end{aligned}$$

Weighted Set Cover: LP relaxation

Variable x_S for each set $S \in \mathcal{F}$.

$$\begin{aligned} \text{min} \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{x \in S} x_S \geq 1 \qquad \text{for all } x \in U \\ & x_S \geq 0 \qquad \text{for all } S \in \mathcal{F} \end{aligned}$$

Feasible solutions of LP is in one-one correspondence with the feasible solutions of Weighted Set Cover?

Weighted Set Cover: LP relaxation

Variable x_S for each set $S \in \mathcal{F}$.

$$\begin{aligned} \text{min} \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{x \in S} x_S \geq 1 \qquad \text{for all } x \in U \\ & x_S \geq 0 \qquad \text{for all } S \in \mathcal{F} \end{aligned}$$

Weighted Set Cover: LP relaxation

Variable x_S for each set $S \in \mathcal{F}$.

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{x \in S} x_S \geq 1 \qquad \text{for all } x \in U \\ & x_S \geq 0 \qquad \text{for all } S \in \mathcal{F} \end{aligned}$$

~~Feasible solutions of LP is in one-one correspondence with the feasible solutions of Weighted Set Cover?~~

Linear Programming

Simplex algorithm. [Dantzig 1947] Can solve LP in practice.

Ellipsoid algorithm. [Khachiyan 1979] Can solve LP in poly-time.

Interior point algorithms. [Karmarkar 1984, Renegar 1988, ...] Can solve LP both in polynomial time and in practice.

Weighted Set Cover: LP relaxation

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{u \in S} x_S \geq 1 \qquad \text{for all } u \in U \\ & x_S \geq 0 \qquad \text{for all } S \in \mathcal{F} \end{aligned}$$

Weighted Set Cover: LP relaxation

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{u \in S} x_S \geq 1 \qquad \text{for all } u \in U \\ & x_S \geq 0 \qquad \text{for all } S \in \mathcal{F} \end{aligned}$$

- optimal value of LP $\leq$ optimal value of ILP.

Weighted Set Cover: LP relaxation

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{u \in S} x_S \geq 1 \qquad \text{for all } u \in U \\ & x_S \geq 0 \qquad \text{for all } S \in \mathcal{F} \end{aligned}$$

- optimal value of LP $\leq$ optimal value of ILP.
- LP solution may not corresponds to set cover.

Weighted Set Cover: LP relaxation

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{F}} w(S) \cdot x_S \\ \text{subject to} \quad & \\ & \sum_{u \in S} x_S \geq 1 \qquad \text{for all } u \in U \\ & x_S \geq 0 \qquad \text{for all } S \in \mathcal{F} \end{aligned}$$

- optimal value of LP $\leq$ optimal value of ILP.
- LP solution may not corresponds to set cover.
- Solve LP and get an optimum solution $x^\star$

Rounding

- Let f be the maximum frequency of an element.

Rounding

- Let f be the maximum frequency of an element.
- Let $Q = \{S : x_S \geq \frac{1}{f}\}$

Rounding

- Let f be the maximum frequency of an element.
- Let $Q = \{S : x_S \geq \frac{1}{f}\}$

Q is a set cover.

$w(Q) \leq f \cdot \sum_{S \in \mathcal{F}} w(S) \cdot x^\star_S \leq f \cdot \mathsf{opt}$

Polynomial Time Approximation Scheme

PTAS

An algorithm that takes an instance of an optimization problem and a parameter $\epsilon > 0$ and produces $(1 + \epsilon)$-approximate solution.

Knapsack problem

Knapsack problem

- Given n objects and a knapsack of capacity W.
- Item i has value $v_i > 0$ and weighs $w_i > 0$.
- Goal: fill knapsack so that maximize the total value.

Knapsack problem

- Given n objects and a knapsack of capacity W.
- Item i has value $v_i > 0$ and weighs $w_i > 0$.
- Goal: fill knapsack so that maximize the total value.

item	value	weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

original instance (W = 11)

Items $\{3, 4\}$ has value 40.

Knapsack problem

- A dynamic programming algorithm with running time $O(nW)$.

Knapsack problem

- A dynamic programming algorithm with running time $O(nW)$.
- Not polynomial in the input size!!

Knapsack problem

- A dynamic programming algorithm with running time $O(nW)$.
- Not polynomial in the input size!!
- It is NP-Complete

Algorithm

Theorem: $O(n^2 v_{max})$ time algorithm, where $v_{max} = \max_i v_i$.

Algorithm

Theorem: $O(n^2 v_{max})$ time algorithm, where $v_{max} = \max_i v_i$.

- Def: $T[i, v] =$ min weight for which we can obtain a solution of value $\geq v$ using a subset of items $1, \ldots, i$.

Algorithm

Theorem: $O(n^2 v_{max})$ time algorithm, where $v_{max} = \max_i v_i$.

- Def: $T[i, v]$ = min weight for which we can obtain a solution of value $\geq v$ using a subset of items $1, \ldots, i$.
- Design a dynamic programming algorithm to compute $T[i, v]$ for all possible values of i and $v \leq \sum v_i$

Algorithm

Theorem: $O(n^2 v_{max})$ time algorithm, where $v_{max} = \max_i v_i$.

- Def: $T[i, v] =$ min weight for which we can obtain a solution of value $\geq v$ using a subset of items $1, \ldots, i$.
- Design a dynamic programming algorithm to compute $T[i, v]$ for all possible values of i and $v \leq \sum v_i$

$$T[i,v] = \begin{cases} 0 & \text{if } v \leq 0 \\ \infty & \text{if } i = 0 \text{ and } v > 0 \\ \min\{T[i-1,v], w_i + T[i-1, v-v_i]\} & \text{otherwise} \end{cases}$$

Algorithm

Theorem: $O(n^2 v_{max})$ time algorithm, where $v_{max} = \max_i v_i$.

- Def: $T[i, v] =$ min weight for which we can obtain a solution of value $\geq v$ using a subset of items $1, \ldots, i$.
- Design a dynamic programming algorithm to compute $T[i, v]$ for all possible values of i and $v \leq \sum v_i$

$$T[i,v] = \begin{cases} 0 & \text{if } v \leq 0 \\ \infty & \text{if } i = 0 \text{ and } v > 0 \\ \min\{T[i-1,v], w_i + T[i-1, v-v_i]\} & \text{otherwise} \end{cases}$$

Not polynomial time!!

Algorithm

Theorem: $O(n^2 v_{max})$ time algorithm, where $v_{max} = \max_i v_i$.

- Def: $T[i, v] =$ min weight for which we can obtain a solution of value $\geq v$ using a subset of items $1, \ldots, i$.
- Design a dynamic programming algorithm to compute $T[i, v]$ for all possible values of i and $v \leq \sum v_i$

$$T[i,v] = \begin{cases} 0 & \text{if } v \leq 0 \\ \infty & \text{if } i = 0 \text{ and } v > 0 \\ \min\{T[i-1,v], w_i + T[i-1, v-v_i]\} & \text{otherwise} \end{cases}$$

Not polynomial time!!

Polynomail time when the values are small integers.

Knapsack: PTAS

Knapsack: PTAS

- Scale all the values to lie in a small range.
- Run the above algorithm with these small values.
- Retun the output.

Knapsack: PTAS

- Scale all the values to lie in a small range.
- Run the above algorithm with these small values.
- Retun the output.

item	value	weight
1	934221	1
2	5956342	2
3	17810013	5
4	21217800	6
5	27343199	7

original instance (W = 11)

item	value	weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

rounded instance (W = 11)

Knapsack: PTAS

Knapsack: PTAS

- $\theta = \frac{\epsilon \cdot v_{max}}{2n}$, where v_{max} is the largest value in the original instance.

Knapsack: PTAS

- $\theta = \frac{\epsilon \cdot v_{max}}{2n}$, where v_{max} is the largest value in the original instance.
- $\widehat{v}_i = \lceil \frac{v_i}{\theta} \rceil$.

Knapsack: PTAS

- $\theta = \frac{\epsilon \cdot v_{max}}{2n}$, where v_{max} is the largest value in the original instance.
- $\widehat{v}_i = \lceil \frac{v_i}{\theta} \rceil$.
- Solve w.r.t $\widehat{v}$. Running time $O(n^3/\epsilon)$.

Knapsack: PTAS

- $\theta = \frac{\epsilon \cdot v_{max}}{2n}$, where v_{max} is the largest value in the original instance.
- $\widehat{v}_i = \lceil \frac{v_i}{\theta} \rceil$.
- Solve w.r.t $\widehat{v}$. Running time $O(n^3/\epsilon)$.

Let $\bar{v}_i = \widehat{v} \cdot \theta = \lceil \frac{v_i}{\theta} \rceil \cdot \theta$

Observation: Optimal solutions to problem with v are equivalent to optimal solutions to problem with $\widehat{v}$.

Knapsack: PTAS

- $\theta = \frac{\epsilon \cdot v_{max}}{2n}$, where v_{max} is the largest value in the original instance.
- $\widehat{v}_i = \lceil \frac{v_i}{\theta} \rceil$.
- Solve w.r.t $\widehat{v}$. Running time $O(n^3/\epsilon)$.

Let $\bar{v}_i = \widehat{v} \cdot \theta = \lceil \frac{v_i}{\theta} \rceil \cdot \theta$

Observation: Optimal solutions to problem with v are equivalent to optimal solutions to problem with $\widehat{v}$.

Intuition: $\bar{v}$ close to v. So optimal solution using $\bar{v}$ is nearly optimal for v.

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$\sum_{i\in S^\star} v_i \quad \leq \quad \sum_{i\in S^\star} \bar{v}_i$$

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$\begin{aligned}\sum_{i\in S^\star} v_i &\leq \sum_{i\in S^\star} \bar{v}_i \\ &\leq \sum_{i\in S} \bar{v}_i\end{aligned}$$

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$\begin{aligned}\sum_{i\in S^\star} v_i &\leq \sum_{i\in S^\star} \bar{v}_i \\ &\leq \sum_{i\in S} \bar{v}_i \\ &\leq \sum_{i\in S} (v_i + \theta)\end{aligned}$$

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$\begin{aligned}
\sum_{i\in S^\star} v_i &\leq \sum_{i\in S^\star} \bar{v}_i \\
&\leq \sum_{i\in S} \bar{v}_i \\
&\leq \sum_{i\in S} (v_i + \theta) \\
&\leq n\theta + \sum_{i\in S} v_i
\end{aligned}$$

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$
\begin{aligned}
\sum_{i\in S^\star} v_i &\leq \sum_{i\in S^\star} \bar{v}_i \\
&\leq \sum_{i\in S} \bar{v}_i \\
&\leq \sum_{i\in S} (v_i + \theta) \\
&\leq n\theta + \sum_{i\in S} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i
\end{aligned}
$$

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$
\begin{aligned}
\sum_{i\in S^\star} v_i &\leq \sum_{i\in S^\star} \bar{v}_i \\
&\leq \sum_{i\in S} \bar{v}_i \\
&\leq \sum_{i\in S} (v_i + \theta) \\
&\leq n\theta + \sum_{i\in S} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i
\end{aligned}
$$

$$
v_{max} \leq \sum_{i\in S^\star} v_i
$$

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$
\begin{aligned}
\sum_{i\in S^\star} v_i &\leq \sum_{i\in S^\star} \bar{v}_i \\
&\leq \sum_{i\in S} \bar{v}_i \\
&\leq \sum_{i\in S} (v_i + \theta) \\
&\leq n\theta + \sum_{i\in S} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i
\end{aligned}
$$

$$
\begin{aligned}
v_{max} &\leq \sum_{i\in S^\star} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i
\end{aligned}
$$

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$
\begin{aligned}
\sum_{i\in S^\star} v_i &\leq \sum_{i\in S^\star} \bar{v}_i \\
&\leq \sum_{i\in S} \bar{v}_i \\
&\leq \sum_{i\in S} (v_i + \theta) \\
&\leq n\theta + \sum_{i\in S} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i
\end{aligned}
$$

$$
\begin{aligned}
v_{max} &\leq \sum_{i\in S^\star} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i \\
&\leq \frac{1}{2} v_{max} + \sum_{i\in S} v_i
\end{aligned}
$$

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$
\begin{aligned}
\sum_{i\in S^\star} v_i &\leq \sum_{i\in S^\star} \bar{v}_i \\
&\leq \sum_{i\in S} \bar{v}_i \\
&\leq \sum_{i\in S} (v_i + \theta) \\
&\leq n\theta + \sum_{i\in S} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i
\end{aligned}
$$

$$
\begin{aligned}
v_{max} &\leq \sum_{i\in S^\star} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i \\
&\leq \frac{1}{2} v_{max} + \sum_{i\in S} v_i \\
&\leq 2\sum_{i\in S} v_i
\end{aligned}
$$

Analysis: approximation ratio

- Let S be the output of our algorithm and $S^\star$ be an optimum solution.
- To prove: $(1+\epsilon)\sum_{i\in S} v_i \geq \sum_{i\in S^\star} v_i$.

$$
\begin{aligned}
\sum_{i\in S^\star} v_i &\leq \sum_{i\in S^\star} \bar{v}_i \\
&\leq \sum_{i\in S} \bar{v}_i \\
&\leq \sum_{i\in S} (v_i + \theta) \\
&\leq n\theta + \sum_{i\in S} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i
\end{aligned}
$$

$$
\begin{aligned}
v_{max} &\leq \sum_{i\in S^\star} v_i \\
&\leq \frac{1}{2}\epsilon v_{max} + \sum_{i\in S} v_i \\
&\leq \frac{1}{2} v_{max} + \sum_{i\in S} v_i \\
&\leq 2\sum_{i\in S} v_i
\end{aligned}
$$

$$\sum_{i\in S^\star} v_i \leq (1+\epsilon)\sum_{i\in S} v_i$$

Summary

- Pricing Methods
- Linear Programming and rounding
- PTAS

Summary

- Pricing Methods
- Linear Programming and rounding
- PTAS

Thank You.