Greedy Approximations : Set Cover

Algorithm Design & Analysis

Spring 2018
Outline
NP-Completeness Transcended

BRUTE-FORCE SOLUTION: \(O(n!)\)

DYNAMIC PROGRAMMING ALGORITHMS: \(O(n^22^n)\)

SELLING ON EBAY: \(O(1)\)

STILL WORKING ON YOUR ROUTE?

SHUT THE HELL UP.

(xkcd #399)
A Greedy Set Cover Approximation

**Input:** Subsets $S_1, \ldots, S_m$ of set $U = \{s_1, \ldots, s_n\}$; weight $w_i$ for each set $S_i$
A Greedy Set Cover Approximation

**Input:** Subsets $S_1, \ldots, S_m$ of set $U = \{s_1, \ldots, s_n\}$; weight $w_i$ for each set $S_i$

**Problem:** Find minimum weight set cover $C$: minimize $\sum_{S_i \in C} w_i$, subject to $U = \bigcup_{S_i \in C} S_i$
A Greedy Set Cover Approximation

**Input:** Subsets $S_1, \ldots, S_m$ of set $U = \{s_1, \ldots, s_n\}$; weight $w_i$ for each set $S_i$

**Problem:** Find minimum weight set cover $C$: minimize $\sum_{S_i \in C} w_i$, subject to $U = \bigcup_{S_i \in C} S_i$

**Question:** How can we be greedy?

• Idea: Unit Covering Cost: Let $c_i = \frac{w_i}{|S_i|}$
• Now build $C$ by adding the $S_i$ with lowest unit covering cost
• But, covering costs change as $C$ is constructed
• Let $R$ be the set of elements of $U$ not covered by $C$; then set $c_i = \frac{w_i}{|S_i \cap R|}$
• That is, the unit covering costs change over run of algorithm
A Greedy Set Cover Approximation

Input: Subsets $S_1, \ldots, S_m$ of set $U = \{s_1, \ldots, s_n\}$; weight $w_i$ for each set $S_i$

Problem: Find minimum weight set cover $C$: minimize $\sum_{S_i \in C} w_i$, subject to $U = \bigcup_{S_i \in C} S_i$

Question: How can we be greedy?

- **Idea:** Unit Covering Cost: Let $c_i = w_i/|S_i|$
A Greedy Set Cover Approximation

Input: Subsets $S_1, \ldots, S_m$ of set $U = \{s_1, \ldots, s_n\}$; weight $w_i$ for each set $S_i$

Problem: Find minimum weight set cover $C$: minimize $\sum_{S_i \in C} w_i$, subject to $U = \bigcup_{S_i \in C} S_i$

Question: How can we be greedy?

- Idea: Unit Covering Cost: Let $c_i = w_i / |S_i|$
- Now build $C$ by adding the $S_i$ with lowest unit covering cost
A Greedy Set Cover Approximation

Input: Subsets $S_1, \ldots, S_m$ of set $U = \{s_1, \ldots, s_n\}$; weight $w_i$ for each set $S_i$

Problem: Find minimum weight set cover $C$: minimize $\sum_{S_i \in C} w_i$, subject to $U = \bigcup_{S_i \in C} S_i$

Question: How can we be greedy?

- **Idea:** Unit Covering Cost: Let $c_i = w_i / |S_i|$
- Now build $C$ by adding the $S_i$ with lowest unit covering cost
- But, covering costs change as $C$ is constructed
A Greedy Set Cover Approximation

**Input:** Subsets $S_1, \ldots, S_m$ of set $U = \{s_1, \ldots, s_n\}$; weight $w_i$ for each set $S_i$

**Problem:** Find minimum weight set cover $\mathcal{C}$: minimize $\sum_{S_i \in \mathcal{C}} w_i$, subject to $U = \bigcup_{S_i \in \mathcal{C}} S_i$

**Question:** How can we be greedy?

- **Idea:** Unit Covering Cost: Let $c_i = w_i/|S_i|$
- Now build $\mathcal{C}$ by adding the $S_i$ with lowest unit covering cost
- But, covering costs change as $\mathcal{C}$ is constructed
- Let $R$ be the set of elements of $U$ not covered by $\mathcal{C}$; then set $c_i = w_i/(|S_i \cap R|)$
A Greedy Set Cover Approximation

**Input:** Subsets $S_1, \ldots, S_m$ of set $U = \{s_1, \ldots, s_n\}$; weight $w_i$ for each set $S_i$

**Problem:** Find minimum weight set cover $C$: minimize $\sum_{S_i \in C} w_i$, subject to $U = \bigcup_{S_i \in C} S_i$

**Question:** How can we be greedy?

- **Idea:** Unit Covering Cost: Let $c_i = w_i/|S_i|
- Now build $C$ by adding the $S_i$ with lowest unit covering cost
- But, covering costs change as $C$ is constructed
- Let $R$ be the set of elements of $U$ not covered by $C$; then set $c_i = w_i/(|S_i \cap R|)$
- That is, the unit covering costs change over run of algorithm
A Greedy Set Cover Algorithm

Algorithm 1

procedure GreedySetCover(S₁, ... , Sₙ)
  R ← U
  C = ∅
  while R ≠ ∅
    Select Sᵢ that minimizes wᵢ / (|Sᵢ ∩ R|)
    R ← R − Sᵢ
  Add Sᵢ to C
  return C

// C is a set cover of U
end procedure

• GreedySetCover can be \(O(\log n)\) times larger than optimal set cover
• We'll show that it's no worse
A Greedy Set Cover Algorithm

Algorithm 2 GreedySetCover

procedure GreedySetCover($S_1, \ldots, S_n$)
    $R \leftarrow U$
    $C = \emptyset$
    while $R \neq \emptyset$
        Select $S_i$ that minimizes $w_i / (|S_i \cap R|)$
        $R \leftarrow R - S_i$
        Add $S_i$ to $C$
    return $C$  // $C$ is a set cover of $U$
end procedure
A Greedy Set Cover Algorithm

Algorithm 3 GreedySetCover

procedure \textbf{GREEDYSETCOVER}(S_1, \ldots, S_n)

\begin{itemize}
  \item \( R \leftarrow U \)
  \item \( C = \emptyset \)
  \item \textbf{while} \( R \neq \emptyset \) \textbf{do}
    \begin{itemize}
      \item Select \( S_i \) that minimizes \( w_i/(|S_i \cap R|) \)
      \item \( R \leftarrow R - S_i \)
      \item Add \( S_i \) to \( C \)
    \end{itemize}
  \item return \( C \) // \( C \) is a set cover of \( U \)
\end{itemize}

end procedure

\begin{itemize}
  \item GreedySetCover can be \( O(\log n) \) times larger than optimal set cover
\end{itemize}
A Greedy Set Cover Algorithm

Algorithm 4 GreedySetCover

procedure \textsc{GreedySetCover}(S_1, \ldots, S_n)

\hspace{1em} R \leftarrow U \hspace{1em}
\hspace{1em} C = \emptyset \hspace{1em}

\hspace{1em} while R \neq \emptyset do

\hspace{2em} Select \( S_i \) that minimizes \( \frac{w_i}{|S_i \cap R|} \)

\hspace{2em} \begin{align*}
& \hspace{1em} R \leftarrow R - S_i \\
& \hspace{1em} \text{Add} \ S_i \ \text{to} \ C
\end{align*}

\hspace{1em} \text{return} \ C \hspace{1em} // \ C \ is \ a \ set \ cover \ of \ U

end procedure

\begin{itemize}
\item GreedySetCover can be \( O(\log n) \) times larger than optimal set cover
\item We’ll show that it’s no worse
\end{itemize}
**Set Cover : An Example**

![Diagram of set cover example](image)

**Figure 11.6** An instance of the Set Cover Problem where the weights of sets are either 1 or $1 + \varepsilon$ for some small $\varepsilon > 0$. The greedy algorithm chooses sets of total weight 4, rather than the optimal solution of weight $2 + 2\varepsilon$.

**Note:** Example can be extended to show $O(\log n)$ factor worse than optimal
A Pricing Model

Idea: Charge each element $s \in U$ the (current) unit cost of the set $S_i$ that first covered it.
A Pricing Model

Idea: Charge each element \( s \in U \) the (current) unit cost of the set \( S_i \) that \textit{first covered it}

- \( s \) gets charged \( c_s = w_i / (|S_i \cap R|) \) for first \( S_i \) in algorithm to cover \( s \)
A Pricing Model

Idea: Charge each element \( s \in U \) the (current) unit cost of the set \( S_i \) that \textit{first covered it}

- \( s \) gets charged \( c_s = w_i / (|S_i \cap R|) \) for first \( S_i \) in algorithm to cover \( s \)
- \textbf{Claim:} \( w(C) = \sum_{S_i \in C} w_i = \sum_{s \in U} c_s \)
A Pricing Model

Idea: Charge each element $s \in U$ the (current) unit cost of the set $S_i$ that first covered it

- $s$ gets charged $c_s = w_i/(|S_i \cap R|)$ for first $S_i$ in algorithm to cover $s$
- **Claim:** $w(C) = \sum_{S_i \in C} w_i = \sum_{s \in U} c_s$
- **Proof:** When $S_i$ is added to $C$ its weight is evenly divided among some elements of $U$
A Pricing Model

Idea: Charge each element $s \in U$ the (current) unit cost of the set $S_i$ that first covered it

- $s$ gets charged $c_s = w_i/(|S_i \cap R|)$ for first $S_i$ in algorithm to cover $s$

- **Claim:** $w(C) = \sum_{S_i \in C} w_i = \sum_{s \in U} c_s$

- **Proof:** When $S_i$ is added to $C$ its weight is evenly divided among some elements of $U$

Goal: Show that for some value $H$ and for every $S_k$:

$$w_k \geq (1/H) \sum_{s \in S_k} c_s \quad \text{[Greedy charges are not too large]}$$
A Pricing Model

Idea: Charge each element \( s \in U \) the (current) unit cost of the set \( S_i \) that first covered it

- \( s \) gets charged \( c_s = w_i / (|S_i \cap R|) \) for first \( S_i \) in algorithm to cover \( s \)
- **Claim:** \( w(C) = \sum_{S_i \in C} w_i = \sum_{s \in U} c_s \)
- **Proof:** When \( S_i \) is added to \( C \) its weight is evenly divided among some elements of \( U \)

Goal: Show that for some value \( H \) and for every \( S_k \):
\[ w_k \geq (1/H) \sum_{s \in S_k} c_s \] [Greedy charges are not too large]
- Then for any set cover \( C^* \), we get

(3)
A Pricing Model

Idea: Charge each element $s \in U$ the (current) unit cost of the set $S_i$ that first covered it

- $s$ gets charged $c_s = w_i / |S_i \cap R|$ for first $S_i$ in algorithm to cover $s$
- **Claim:** $w(C) = \sum_{S_i \in C} w_i = \sum_{s \in U} c_s$
- **Proof:** When $S_i$ is added to $C$ its weight is evenly divided among some elements of $U$

Goal: Show that for some value $H$ and for every $S_k$:

- $w_k \geq (1/H) \sum_{s \in S_k} c_s$ [Greedy charges are not too large]

  - Then for any set cover $C^*$, we get
  
  $$w(C^*) = \sum_{S_i \in C^*} w_i \geq \sum_{S_i \in C^*} (1/H) \sum_{s \in S_i} c_s$$  
  
  (1)
A Pricing Model

**Idea:** Charge each element \( s \in U \) the (current) unit cost of the set \( S_i \) that *first covered it*

- \( s \) gets charged \( c_s = w_i / (|S_i \cap R|) \) for first \( S_i \) in algorithm to cover \( s \)
- **Claim:** \( w(C) = \sum_{S_i \in C} w_i = \sum_{s \in U} c_s \)
- **Proof:** When \( S_i \) is added to \( C \) its weight is evenly divided among some elements of \( U \)

**Goal:** Show that for *some* value \( H \) and for *every* \( S_k \):
\[
w_k \geq \left( \frac{1}{H} \right) \sum_{s \in S_k} c_s \quad \text{[Greedy charges are not too large]} \]

- Then for *any* set cover \( C^* \), we get
\[
w(C^*) = \sum_{S_i \in C^*} w_i \geq \sum_{S_i \in C^*} \left( \frac{1}{H} \right) \sum_{s \in S_i} c_s \quad (1)
\]
\[
\geq \left( \frac{1}{H} \right) \sum_{s \in U} c_s = \left( \frac{1}{H} \right) \sum_{S_i \in C} w_i \quad (2)
\]
A Pricing Model

Idea: Charge each element \( s \in U \) the (current) unit cost of the set \( S_i \) that first covered it

- \( s \) gets charged \( c_s = w_i / (|S_i \cap R|) \) for first \( S_i \) in algorithm to cover \( s \)

- **Claim:** \( w(C) = \sum_{S_i \in C} w_i = \sum_{s \in U} c_s \)

- **Proof:** When \( S_i \) is added to \( C \) its weight is evenly divided among some elements of \( U \)

Goal: Show that for some value \( H \) and for every \( S_k \):
\[
w_k \geq (1/H) \sum_{s \in S_k} c_s \quad [\text{Greedy charges are not too large}]\]

- Then for any set cover \( C^* \), we get
\[
w(C^*) = \sum_{S_i \in C^*} w_i \geq \sum_{S_i \in C^*} (1/H) \sum_{s \in S_i} c_s \quad (1)
\]
\[
\geq (1/H) \sum_{s \in U} c_s = (1/H) \sum_{S_i \in C} w_i \quad (2)
\]
\[
= (1/H)w(C) \quad (3)
\]
An Accounting Scheme

Goal: Show that for some value $H$ and for every $S_k$:

$$\sum_{s \in S_k} c_s \leq Hw_k$$
An Accounting Scheme

**Goal:** Show that for some value $H$ and for every $S_k$:

$$\sum_{s \in S_k} c_s \leq Hw_k$$

**Idea:** Consider GreedySetCover from the point of view of $S_k$. 
An Accounting Scheme

**Goal:** Show that for *some* value $H$ and for *every* $S_k$:

$$\sum_{s \in S_k} c_s \leq Hw_k$$

**Idea:** Consider GreedySetCover from the point of view of $S_k$
- Run GreedySetCover to find the order in which sets were added
An Accounting Scheme

**Goal:** Show that for *some* value $H$ and for *every* $S_k$:

$$\sum_{s \in S_k} c_s \leq Hw_k$$

**Idea:** Consider GreedySetCover from the point of view of $S_k$

- Run GreedySetCover to find the order in which sets were added
- Relabel $U$ so that $s_1, \ldots, s_d$ are the elements of $S_k$ *in the order they were covered by* GreedySetCover ($d = |S_k|$)
An Accounting Scheme

**Goal:** Show that for some value $H$ and for every $S_k$:

$$
\sum_{s \in S_k} c_s \leq Hw_k
$$

**Idea:** Consider GreedySetCover from the point of view of $S_k$

- Run GreedySetCover to find the order in which sets were added
- Relabel $U$ so that $s_1, \ldots, s_d$ are the elements of $S_k$ in the order they were covered by GreedySetCover ($d = |S_k|$)
- Rerun GreedySetCover : relabeling has no impact!
An Accounting Scheme

**Goal:** Show that for some value $H$ and for every $S_k$:

$$\sum_{s \in S_k} c_s \leq H w_k$$

**Idea:** Consider GreedySetCover from the point of view of $S_k$
- Run GreedySetCover to find the order in which sets were added
- Relabel $U$ so that $s_1, \ldots, s_d$ are the elements of $S_k$ *in the order they were covered by GreedySetCover* ($d = |S_k|$)
- Rerun GreedySetCover: relabeling has no impact!
- Now try to bound the costs $\{c_j = c_{s_j} : s_j \in S_k\}$
Bounding the Costs

Observations
Bounding the Costs

Observations

- When \( s_j \in S_k \) first covered by some \( S_i \) in GSC, none of \( s_j, \ldots, s_d \) are yet covered
Bounding the Costs

Observations

• When $s_j \in S_k$ first covered by some $S_i$ in GSC, none of $s_j, \ldots, s_d$ are yet covered
• So $|S_k \cap R| \geq (d - j + 1)$, thus $w_k/|S_k \cap R| \leq w_k/(d - j + 1)$
Bounding the Costs

Observations

• When $s_j \in S_k$ first covered by some $S_i$ in GSC, none of $s_j, \ldots, s_d$ are yet covered
• So $|S_k \cap R| \geq (d - j + 1)$, thus $w_k/|S_k \cap R| \leq w_k/(d - j + 1)$
• So $c_j = w_i/|S_i \cap R| \leq w_k/|S_k \cap R| \leq w_k/(d - j + 1)$
Bounding the Costs

Observations

- When $s_j \in S_k$ first covered by some $S_i$ in GSC, none of $s_j, \ldots, s_d$ are yet covered
- So $|S_k \cap R| \geq (d - j + 1)$, thus $w_k/|S_k \cap R| \leq w_k/(d - j + 1)$
- So $c_j = w_i/|S_i \cap R| \leq w_k/|S_k \cap R| \leq w_k/(d - j + 1)$
- Thus
Bounding the Costs

Observations

• When \( s_j \in S_k \) first covered by some \( S_i \) in GSC, none of \( s_j, \ldots, s_d \) are yet covered.

• So \( |S_k \cap R| \geq (d - j + 1) \), thus \( w_k/|S_k \cap R| \leq w_k/(d - j + 1) \).

• So \( c_j = w_i/|S_i \cap R| \leq w_k/|S_k \cap R| \leq w_k/(d - j + 1) \).

• Thus

\[
\sum_{s \in S_k} c_s = \sum_{j=1}^{d} c_j \leq \sum_{j=1}^{d} w_k/(d-j+1) = w_k \sum_{i=1}^{d} 1/i \quad \text{note: } i = d-j+1
\]
Bounding the Costs

Observations

• When \( s_j \in S_k \) first covered by some \( S_i \) in GSC, none of \( s_j, \ldots, s_d \) are yet covered

• So \( |S_k \cap R| \geq (d - j + 1) \), thus \( w_k/|S_k \cap R| \leq w_k/(d - j + 1) \)

• So \( c_j = w_i/|S_i \cap R| \leq w_k/|S_k \cap R| \leq w_k/(d - j + 1) \)

• Thus

\[
\sum_{s \in S_k} c_s = \sum_{j=1}^{d} c_j \leq \sum_{j=1}^{d} w_k/(d-j+1) = w_k \sum_{i=1}^{d} 1/i \quad \text{(note: } i = d-j+1)\]

• So, letting \( H(d) = \sum_{i=1}^{d} 1/d \) gives \( \sum_{s \in S_k} c_s \leq H(d)w_k \)
Bounding the Costs

Observations

• When \( s_j \in S_k \) first covered by some \( S_i \) in GSC, none of \( s_j, \ldots, s_d \) are yet covered

• So \( |S_k \cap R| \geq (d - j + 1) \), thus \( w_k/|S_k \cap R| \leq w_k/(d - j + 1) \)

• So \( c_j = w_i/|S_i \cap R| \leq w_k/|S_k \cap R| \leq w_k/(d - j + 1) \)

• Thus

\[
\sum_{s \in S_k} c_s = \sum_{j=1}^{d} c_j \leq \sum_{j=1}^{d} \frac{w_k}{d-j+1} = w_k \sum_{i=1}^{d} \frac{1}{i} \quad \text{(note: } i = d-j+1)\]

• So, letting \( H(d) = \sum_{i=1}^{d} 1/d \) gives \( \sum_{s \in S_k} c_s \leq H(d)w_k \)

• Now let \( d^* = \max_{k=1}^{m} |S_k| \), and \( H = H(d^*) \)
Putting It All Together

**Theorem:** GreedySetCover produces a set cover having weight within a factor of $H = H(d^*)$ of the optimum.
**Putting It All Together**

**Theorem:** GreedySetCover produces a set cover having weight within a factor of $H = H(d^*)$ of the optimum.

How big is $H(d^*)$?
**Putting It All Together**

**Theorem:** GreedySetCover produces a set cover having weight within a factor of \( H = H(d^*) \) of the optimum.

How big is \( H(d^*) \)?

**Fact:** \( \ln(n + 1) \leq H(n) \leq 1 + \ln n \), so \( H(d^*) \leq H(n) \in \Theta(\log n) \).
Putting It All Together

**Theorem:** GreedySetCover produces a set cover having weight within a factor of $H = H(d^*)$ of the optimum

How big is $H(d^*)$?

**Fact:** $\ln(n + 1) \leq H(n) \leq 1 + \ln n$, so $H(d^*) \leq H(n) \in \Theta(\log n)$.

*Figure 11.7* Upper and lower bounds for the Harmonic Function $H(n)$. 
The Pricing Method

Observations
The Pricing Method

Observations

- View $c_s$ as the price paid by $s$ to be covered by $S_i$
The Pricing Method

Observations

- View $c_s$ as the price paid by $s$ to be covered by $S_i$
- $S_i$ used to cover $s$ is selected by greedy algorithm, but
The Pricing Method

Observations

• View $c_s$ as the price paid by $s$ to be covered by $S_i$
• $S_i$ used to cover $s$ is selected by greedy algorithm, but
• Sum of prices paid by elements of each $S_k$ are bounded by a constant multiple of $w_k$
The Pricing Method

Observations

- View $c_s$ as the price paid by $s$ to be covered by $S_i$
- $S_i$ used to cover $s$ is selected by greedy algorithm, but
- Sum of prices paid by elements of each $S_k$ are bounded by a constant multiple of $w_k$
- In other words, the prices paid by elements are not exorbitant; more importantly:
The Pricing Method

Observations

- View $c_s$ as the price paid by $s$ to be covered by $S_i$
- $S_i$ used to cover $s$ is selected by greedy algorithm, but
- Sum of prices paid by elements of each $S_k$ are bounded by a constant multiple of $w_k$
- In other words, the prices paid by elements are not exorbitant; more importantly:
- The total paid by all elements in $U$ using the cover produced by the greedy algorithm can be used to bound the value of any other cover!
The Pricing Method

Observations

- View $c_s$ as the price paid by $s$ to be covered by $S_i$
- $S_i$ used to cover $s$ is selected by greedy algorithm, but
- Sum of prices paid by elements of each $S_k$ are bounded by a constant multiple of $w_k$
- In other words, the prices paid by elements are not exorbitant; more importantly:
  - The total paid by all elements in $U$ using the cover produced by the greedy algorithm can be used to bound the value of any other cover!

This idea of using a pricing method to measure goodness of approximation is quite powerful
Approximation Via Reduction: Weighted Vertex Cover

The Problem: Given a graph $G = (V, E)$ with vertex weights $w_v$, find a vertex cover of low weight.
The Problem: Given a graph $G = (V, E)$ with vertex weights $w_v$, find a vertex cover of low weight

Recall that SETCOVER can be used to solve VERTEXCOVER (even with weights)

Perhaps we can do better?
The Problem: Given a graph $G = (V, E)$ with vertex weights $w_v$, find a vertex cover of low weight.

Recall that SETCOVER can be used to solve VERTEXCOVER (even with weights).

- For $G = (V, E)$, $U = E$ and the sets are $S_v = \{e \in E : e = \{u, v\}\}$
Approximation Via Reduction: Weighted Vertex Cover

The Problem: Given a graph $G = (V, E)$ with vertex weights $w_v$, find a vertex cover of low weight

Recall that SETCOVER can be used to solve VERTEXCOVER (even with weights)

- For $G = (V, E)$, $U = E$ and the sets are $S_v = \{e \in E : e = \{u, v\}\}$
- A set cover of $U$ by $S_{v_1}, \ldots S_{v_k}$ corresponds to a vertex cover of $E$ by $v_1, \ldots, v_k$
The Problem: Given a graph \( G = (V, E) \) with vertex weights \( w_v \), find a vertex cover of low weight.

Recall that SETCOVER can be used to solve VERTEXCOVER (even with weights).

- For \( G = (V, E) \), \( U = E \) and the sets are \( S_v = \{ e \in E : e = \{u, v\} \} \).
- A set cover of \( U \) by \( S_{v_1}, \ldots, S_{v_k} \) corresponds to a vertex cover of \( E \) by \( v_1, \ldots, v_k \).
- Thus a minimum weight vertex cover of \( G \) corresponds to a minimum weight set cover of \( U \).
The Problem: Given a graph $G = (V, E)$ with vertex weights $w_v$, find a vertex cover of low weight. Recall that SETCOVER can be used to solve VERTEXCOVER (even with weights).

- For $G = (V, E)$, $U = E$ and the sets are $S_v = \{ e \in E : e = \{u, v\} \}$
- A set cover of $U$ by $S_{v_1}, \ldots S_{v_k}$ corresponds to a vertex cover of $E$ by $v_1, \ldots, v_k$
- Thus a minimum weight vertex cover of $G$ corresponds to a minimum weight set cover of $U$
- So GREEDYSETCOVER can be used to get a $O(\log n)$ approximation for VERTEXCOVER
Approximation Via Reduction: Weighted Vertex Cover

**The Problem:** Given a graph $G = (V, E)$ with vertex weights $w_v$, find a vertex cover of low weight.

Recall that SETCOVER can be used to solve VERTEXCOVER (even with weights)

- For $G = (V, E)$, $U = E$ and the sets are $S_v = \{e \in E : e = \{u, v\}\}$
- A set cover of $U$ by $S_{v_1}, \ldots S_{v_k}$ corresponds to a vertex cover of $E$ by $v_1, \ldots, v_k$
- Thus a minimum weight vertex cover of $G$ corresponds to a minimum weight set cover of $U$
- So GREEDYSETCOVER can be used to get a $O(\log n)$ approximation for VERTEXCOVER
- Perhaps we can do better?
Weighted Vertex Covers via the Pricing Method

**Idea:** An edge $e$ pays a vertex $v$ some price $p_e$ to cover it.

\[ \sum_{e \in E} p_e \leq \sum_{v \in S} \sum_{e \in \{u, v\}} p_e \leq \sum_{v \in S} w_v = w(S) \]

So, in particular, if $S^*$ is a minimum weight vertex cover, we have

\[ \sum_{e \in E} p_e \leq w(S^*) \]

That is, the sum of edge prices is a lower bound on the weight of a minimum weight vertex cover.
Weighted Vertex Covers via the Pricing Method

Idea: An edge $e$ pays a vertex $v$ some price $p_e$ to cover it.

- The set $\{p_e : e \in E\}$ of prices is *fair* if, for each $v \in V$
  $$\sum_{e=\{u,v\}} p_e \leq w_v \; (v \text{ is not overcharging})$$

Claim: For any vertex cover $S$ and fair prices $\{p_e : e \in E\}$:
$$\sum_{e \in E} p_e \leq w(S)$$

Proof:
$$\sum_{e \in E} p_e \leq \sum_{v \in S} \sum_{e=\{u,v\}} p_e \leq \sum_{v \in S} w_v = w(S)$$

So, in particular, if $S^*$ is a minimum weight vertex cover, we have
$$\sum_{e \in E} p_e \leq w(S^*)$$

That is, the sum of edge prices is a lower bound on the weight of a minimum weight vertex cover.
Weighted Vertex Covers via the Pricing Method

Idea: An edge $e$ pays a vertex $v$ some price $p_e$ to cover it.

- The set $\{p_e : e \in E\}$ of prices is *fair* if, for each $v \in V$
  \[ \sum_{e=\{u,v\}} p_e \leq w_v \]  (v is not *overcharging*)

Claim: For any vertex cover $S$ and fair prices $\{p_e : e \in E\}$:
\[ \sum_{e \in E} p_e \leq w(S) \]
Weighted Vertex Covers via the Pricing Method

Idea: An edge $e$ pays a vertex $v$ some price $p_e$ to cover it.

- The set $\{p_e : e \in E\}$ of prices is fair if, for each $v \in V$
  \[ \sum_{e=\{u,v\}} p_e \leq w_v \] (v is not overcharging)

Claim: For any vertex cover $S$ and fair prices $\{p_e : e \in E\}$:
\[ \sum_{e \in E} p_e \leq w(S) \]

Proof:
\[ \sum_{e \in E} p_e \leq \sum_{v \in S} \sum_{e=\{u,v\}} p_e \leq \sum_{v \in S} w_v = w(S) \]
Weighted Vertex Covers via the Pricing Method

Idea: An edge \( e \) pays a vertex \( v \) some price \( p_e \) to cover it.

- The set \( \{p_e : e \in E\} \) of prices is fair if, for each \( v \in V \)

\[ \sum_{e=\{u,v\}} p_e \leq w_v \] (\( v \) is not overcharging)

Claim: For any vertex cover \( S \) and fair prices \( \{p_e : e \in E\} \):

\[ \sum_{e \in E} p_e \leq w(S) \]

Proof:

\[ \sum_{e \in E} p_e \leq \sum_{v \in S} \sum_{e=\{u,v\}} p_e \leq \sum_{v \in S} w_v = w(S) \]

So, in particular, if \( S^* \) is a minimum weight vertex cover, we have

\[ \sum_{e \in E} p_e \leq w(S^*) \]
Weighted Vertex Covers via the Pricing Method

Idea: An edge \( e \) pays a vertex \( v \) some price \( p_e \) to cover it.

- The set \( \{ p_e : e \in E \} \) of prices is fair if, for each \( v \in V \)
  \[ \sum_{e=\{u,v\}} p_e \leq w_v \text{ (} v \text{ is not overcharging)} \]

Claim: For any vertex cover \( S \) and fair prices \( \{ p_e : e \in E \} \):
  \[ \sum_{e \in E} p_e \leq w(S) \]

Proof:

\[
\sum_{e \in E} p_e \leq \sum_{v \in S} \sum_{e=\{u,v\}} p_e \leq \sum_{v \in S} w_v = w(S)
\]

So, in particular, if \( S^* \) is a minimum weight vertex cover, we have
\[ \sum_{e \in E} p_e \leq w(S^*) \]

That is, the sum of edge prices is a lower bound on the weight of a minimum weight vertex cover
Idea: Simultaneously build a vertex cover while greedily setting prices; show that small multiple of price sum bounds cover weight
A Price-Setting Greedy Algorithm

Idea: Simultaneously build a vertex cover while greedily setting prices; show that small multiple of price sum bounds cover weight

Def’n: A vertex $v$ is tight if $\sum_{e=\{u,v\}} p_e = w_v$
A Price-Setting Greedy Algorithm

Idea: Simultaneously build a vertex cover while greedily setting prices; show that small multiple of price sum bounds cover weight

Def’n: A vertex $v$ is tight if $\sum_{e=\{u,v\}} p_e = w_v$

Algorithm 7 PriceFixing

```
procedure PRICEFIXING($G = (V, E), w[-]$)
    Set all prices $p[e]$ to 0
    while Some edge $e$ has neither vertex tight do
        Select such an edge $e = \{u, v\}$
        Increase $p[e]$ until first of $u$ or $v$ becomes tight
    Return set $S$ of all tight nodes
end procedure
```
A Price-Setting Greedy Algorithm

Idea: Simultaneously build a vertex cover while greedily setting prices; show that small multiple of price sum bounds cover weight

Def’n: A vertex \( v \) is tight if \( \sum_{e=\{u,v\}} p_e = w_v \)

Algorithm 8 PriceFixing

\begin{verbatim}
procedure PRICEFIXING(G = (V, E), w[−])
    Set all prices \( p[e] \) to 0
    while Some edge \( e \) has neither vertex tight do
        Select such an edge \( e = \{u, v\} \)
        Increase \( p[e] \) until first of \( u \) or \( v \) becomes tight
    Return set \( S \) of all tight nodes
end procedure
\end{verbatim}

Observe: Tight vertices form a cover: every edge has at least one tight vertex; also, the prices are fair
How Good is PriceFixing?

**Claim:** The $S$ and $p[-]$ returned by PriceFixing satisfy $w(S) \leq 2 \sum_{e \in E} p_e$ and the prices are fair.
How Good is PriceFixing?

Claim: The $S$ and $p[\rightarrow]$ returned by PriceFixing satisfy $w(S) \leq 2 \sum_{e \in E} p_e$ and the prices are fair.

Proof:

$$w(S) = \sum_{v \in S} w_v = \sum_{v \in S} \sum_{e = \{u, v\}} p_e \leq 2 \sum_{e \in E} p_e$$
How Good is PriceFixing?

Claim: The $S$ and $p[\_\_\_]$ returned by PriceFixing satisfy $w(S) \leq 2 \sum_{e \in E} p_e$ and the prices are fair.

Proof:

$$w(S) = \sum_{v \in S} w_v = \sum_{v \in S} \sum_{e=\{u,v\}} p_e \leq 2 \sum_{e \in E} p_e$$

Corollary: The weight of $S$ is within a factor of 2 of optimal: $w(S) \leq 2w(S^*)$