Greedy Approximations: The Pricing Method

Algorithm Design & Analysis

Spring 2019
Outline
NP-Completeness Transcended

(xkcd #399)
Weighted Vertex Covers via the Pricing Method

The Problem: Given a graph $G = (V, E)$ with vertex weights $\{w_v : v \in V\}$, find a vertex cover $S$ minimizing $w(S) = \sum_{v \in S} w_v$
Weighted Vertex Covers via the Pricing Method

The Problem: Given a graph $G = (V, E)$ with vertex weights \( \{w_v : v \in V\} \), find a vertex cover $S$ minimizing $w(S) = \sum_{v \in S} w_v$

Idea: An edge $e = \{u, v\}$ pays a vertex $v$ some price $p_e$ to cover it.

Claim: For any vertex cover $S$ and fair prices \( \{p_e : e \in E\} \) of prices is fair if, for each $v \in V$, \( \sum_{e = \{u, v\}} p_e \leq w_v \) (v is not overcharging).

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 any minimum-weight vertex cover $S^*$ satisfies $\sum_{e \in E} p_e \leq w(S^*)$.

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

The Problem: Given a graph $G = (V, E)$ with vertex weights \{\(w_v : v \in V\}\), find a vertex cover $S$ minimizing $w(S) = \sum_{v \in S} w_v$

Idea: An edge $e = \{u, v\}$ 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 $v \in V$
Weighted Vertex Covers via the Pricing Method

The Problem: Given a graph $G = (V, E)$ with vertex weights \{w_v : v \in V\}, find a vertex cover $S$ minimizing $w(S) = \sum_{v \in S} w_v$

Idea: An edge $e = \{u, v\}$ 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)
Weighted Vertex Covers via the Pricing Method

The Problem: Given a graph $G = (V, E)$ with vertex weights $\{w_v : v \in V\}$, find a vertex cover $S$ minimizing $w(S) = \sum_{v \in S} w_v$

Idea: An edge $e = \{u, v\}$ 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

The Problem: Given a graph $G = (V, E)$ with vertex weights $\{w_v : v \in V\}$, find a vertex cover $S$ minimizing $w(S) = \sum_{v \in S} w_v$

Idea: An edge $e = \{u, v\}$ 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

The Problem: Given a graph $G = (V, E)$ with vertex weights 
$\{w_v : v \in V\}$, find a vertex cover $S$ minimizing $w(S) = \sum_{v \in S} w_v$

Idea: An edge $e = \{u, v\}$ 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 any minimum-weight vertex cover $S^*$ satisfies $\sum_{e \in E} p_e \leq w(S^*)$
Weighted Vertex Covers via the Pricing Method

The Problem: Given a graph $G = (V, E)$ with vertex weights \( \{w_v : v \in V\} \), find a vertex cover $S$ minimizing $w(S) = \sum_{v \in S} w_v$

Idea: An edge $e = \{u, v\}$ 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 any minimum-weight vertex cover $S^*$ satisfies $\sum_{e \in E} p_e \leq w(S^*)$

That is, the sum of edge prices is a lower bound on the weight of a minimum weight vertex cover
A Price-Setting Greedy Algorithm

Idea:
Simultaneously build a vertex cover $S$ while greedily setting (fair) prices; show that $\sum_{v \in S} w_v \leq 2 \sum_{e \in E} p_e$

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

Algorithm 1
```
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
        end
    end
    Return set $S$ of all tight nodes
end procedure
```

Observe:
The set $S$ returned is a cover and the prices remain fair
A Price-Setting Greedy Algorithm

**Idea:** Simultaneously build a vertex cover $S$ while greedily setting (fair) prices; show that $\sum_{v \in S} w_v \leq 2 \sum_{e \in E} p_e$
A Price-Setting Greedy Algorithm

Idea: Simultaneously build a vertex cover $S$ while greedily setting (fair) prices; show that $\sum_{v \in S} w_v \leq 2 \sum_{e \in E} p_e$

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 $S$ while greedily setting (fair) prices; show that $\sum_{v \in S} w_v \leq 2 \sum_{e \in E} p_e$

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

Algorithm 4 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 $S$ while greedily setting (fair) prices; show that $\sum_{v \in S} w_v \leq 2 \sum_{e \in E} p_e$

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

---

**Algorithm 5 PriceFixing**

```plaintext
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
```

---

**Observe:** The set $S$ returned is a cover and the prices remain fair
**Claim:** The $S$ and $p[\_\_\_\_]$ returned by PriceFixing satisfy

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

**Claim:** The $S$ and $p[−]$ returned by PriceFixing satisfy

$$w(S) \leq 2 \sum_{e \in E} p_e$$

**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:

$$w(S) \leq 2 w(S^\ast)$$
How Good is PriceFixing?

**Claim:** The $S$ and $p[−]$ returned by PriceFixing satisfy

$$w(S) \leq 2 \sum_{e \in E} p_e$$

**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:

$$w(S) \leq 2w(S^*)$$
The Pricing Method

Observations
The Pricing Method

Observations

- View $p_e$ as the price paid by $e$ to be covered by some vertex that covers $e$
The Pricing Method

Observations

- View $p_e$ as the price paid by $e$ to be covered by some vertex that covers $e$
- The $v$ used to cover $e$ is selected by greedy algorithm, but...
The Pricing Method

Observations

- View $p_e$ as the price paid by $e$ to be covered by some vertex that covers $e$
- The $v$ used to cover $e$ is selected by greedy algorithm, but
- Sum of prices paid by all edges incident with $v$ are bounded
The Pricing Method

Observations

- View $p_e$ as the price paid by $e$ to be covered by some vertex that covers $e$
- The $v$ used to cover $e$ is selected by greedy algorithm, but
- Sum of prices paid by all edges incident with $v$ are bounded
- In other words, the prices paid by edges are not exorbitant; more importantly:
The Pricing Method

Observations

- View $p_e$ as the price paid by $e$ to be covered by some vertex that covers $e$
- The $v$ used to cover $e$ is selected by greedy algorithm, but
- Sum of prices paid by all edges incident with $v$ are bounded
- In other words, the prices paid by edges are not exorbitant; more importantly:
- The total paid by all edges of $G$ using the cover produced by the greedy algorithm can be used to bound the weight of any other cover!
Observations

- View $p_e$ as the price paid by $e$ to be covered by some vertex that covers $e$
- The $v$ used to cover $e$ is selected by greedy algorithm, but
- Sum of prices paid by all edges incident with $v$ are bounded
- In other words, the prices paid by edges are not exorbitant; more importantly:
  - The total paid by all edges of $G$ using the cover produced by the greedy algorithm can be used to bound the weight of any other cover!

This idea of using a pricing method to measure goodness of approximation is quite powerful
A Greedy Set Cover Approximation

Input:

Idea:

• Unit Covering Cost: \( c_i = \frac{w_i}{|S_i|} \)

• 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 \( U = \bigcup_{i=1}^{m} S_i = \{s_1, \ldots, s_n\} \) (every \( s_i \) is in some \( S_j \))
A Greedy Set Cover Approximation

Input:

- Subsets $S_1, \ldots, S_m$ of set $U = \bigcup_{i=1}^{m} S_i = \{s_1, \ldots, s_n\}$ (every $s_i$ is in some $S_j$)
- Weight $w_i$ for each set $S_i$
A Greedy Set Cover Approximation

Input:
- Subsets $S_1, \ldots, S_m$ of set $U = \bigcup_{i=1}^{m} S_i = \{s_1, \ldots, s_n\}$ (every $s_i$ is in some $S_j$)
- 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 $U = \bigcup_{i=1}^{m} S_i = \{s_1, \ldots, s_n\}$ (every $s_i$ is in some $S_j$)
- 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 $U = \bigcup_{i=1}^m S_i = \{s_1, \ldots, s_n\}$ (every $s_i$ is in some $S_j$)
- 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 $U = \bigcup_{i=1}^m S_i = \{s_1, \ldots, s_n\}$ (every $s_i$ is in some $S_j$)
- 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 $U = \bigcup_{i=1}^{m} S_i = \{s_1, \ldots, s_n\}$ (every $s_i$ is in some $S_j$)
- 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 $U = \bigcup_{i=1}^m S_i = \{s_1, \ldots, s_n\}$ (every $s_i$ is in some $S_j$)
- 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 $c_i = w_i/(|S_i \cap R|)$
A Greedy Set Cover Approximation

Input:
- Subsets \(S_1, \ldots, S_m\) of set \(U = \bigcup_{i=1}^{m} S_i = \{s_1, \ldots, s_n\}\) (every \(s_i\) is in some \(S_j\))
- 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 6

GreedySetCover

procedure

GreedySetCover($S_1, \ldots, S_n$)

$R \leftarrow \emptyset$

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

• GreedySetCover can be $O(\log n)$ times larger than optimal set cover

• We’ll show that it’s no worse

Algorithm 7 GreedySetCover

procedure GreedySetCover($S_1, \ldots, S_n$)
    $R \leftarrow U$
    $C = \emptyset$
    while $R \neq \emptyset$ do
        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
Algorithm 8 GreedySetCover

```
procedure GREEDYSETCOVER(S_1, \ldots, S_n)
    R ← \emptyset
    C = \emptyset
    while R ≠ \emptyset do
        Select S_i that minimizes \( w_i / (|S_i \cap R|) \)
        R ← R − S_i
        Add S_i 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
A Greedy Set Cover Algorithm

Algorithm 9 GreedySetCover

procedure GREEDYSETCOVER(S₁, . . . , Sₙ)
    \( R \leftarrow U \)
    \( C = \emptyset \)
    \( \text{while } R \neq \emptyset \text{ do} \)
        Select \( S_i \) that minimizes \( w_i / (|S_i \cap R|) \)
        \( R \leftarrow R - S_i \)
        Add \( S_i \) to \( C \)
    \( \text{return } C \) // \( C \) 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
Set Cover: An Example

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

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 *first covered it*: Pricing!!

\[ s \text{ gets charged } c_s = \frac{w_i}{|S_i \cap R|} \text{ for first } S_i \text{ in algorithm to cover } s \]

**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 the as-yet uncovered elements of $S_i$

**Goal:**

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

\[ w_k \geq \frac{1}{H} \sum_{s \in S_k} c_s \]

[Greedy charges are not too large]

Then for any set cover $C^*$, we get

\[ w(C^*) = \sum_{S_i \in C^*} w_i \geq \sum_{S_i \in C^*} \left( \frac{1}{H} \sum_{s \in S_i} c_s \right) \geq \left( \frac{1}{H} \right) \sum_{s \in U} c_s = \left( \frac{1}{H} \right) w(C) \tag{3} \]
**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 \( s \)
A Pricing Model

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

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

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

- $s$ 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 the as-yet uncovered elements of $S_i$
A Pricing Model

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

- \( s \) gets charged \( c_s = \frac{w_i}{|S_i \cap R|} \) 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 the as-yet uncovered elements of \( S_i \)

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]
A Pricing Model

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

- \( s \) gets charged \( c_s = \frac{w_i}{|S_i \cap R|} \) 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 the as-yet uncovered elements of \( S_i \)

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
**A Pricing Model**

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

- \( s \) gets charged \( c_s = \frac{w_i}{|S_i \cap R|} \) 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 the as-yet uncovered elements of \( S_i \)

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

Then for *any* set cover \( C^* \), we get
\[
    w(C^*) = \sum_{S_i \in C^*} w_i \geq \sum_{S_i \in C^*} \left( \frac{1}{H} \right) \sum_{s \in S_i} c_s \quad (1)
\]
\[
    \geq \left( \frac{1}{H} \right) \sum_{s \in U} c_s = \left( \frac{1}{H} \right) \sum_{S_i \in C} w_i \quad (2)
\]
\[
    = \left( \frac{1}{H} \right) w(C) \quad (3)
\]
Goal: Show that for some value $H$ and for every $S_k$:

\[ \sum_{s \in S_k} c_s \leq H \]
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
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

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

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

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

• Now let \( d^* = \max_{k=1}^{k=m} |S_k| \), 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 $H(d^*) \leq H(n) \in \Theta(\log n)$.

![Graph](image-url)

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

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.

But we did much better with a more finely-tuned pricing method!
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 $S_v = \{ e \in E : e = \{u, v\} \}$

- Thus a minimum weight vertex cover of $G$ corresponds to a minimum weight set cover of $U$.

- So GREEDYSETCOVER can be used to get an $O(\log n)$ approximation for VERTEXCOVER.

- But we did much better with a more finely-tuned pricing method!
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 \( v_1, \ldots, v_k \)
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 $U$
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
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.
- But we did much better with a more finely-tuned pricing method!