Minimum (weighted) Set Cover

Mínima Cobertura por conjuntos (sem custos)

- **Instância:** Um conjunto universo $U = \{u_1, \ldots, u_n\}$ e uma família $\mathcal{S} = \{S_1, \ldots, S_m\}$ de subconjuntos de U que cobrem U.
  Ou seja, $\mathcal{S}$ é uma cobertura de U: $\bigcup_{i=1}^{m} S_i = U$.
- **Objetivo:** Obter menor número k de subconjuntos $S_{i_1}, \ldots, S_{i_k}$ em $\mathcal{S}$ que cobrem U, ou seja, $S_{i_1} \cup \ldots \cup S_{i_k} = U$.

Mínima Cobertura por conjuntos (com custos)

- **Instância:** Idem + cada subconjunto S_i tem um custo $c(S_i)$
- **Objetivo:** Obter uma cobertura $S_{i_1}, \ldots, S_{i_k}$ de U de custo mínimo ($c(S_{i_1}) + \ldots + c(S_{i_k})$ mínimo).
- **Generalização:** Basta tomar custo 1 para cada conjunto.

Algoritmo MinCC-Chvátal (com custos e sem custos)

Algoritmo MinCC-Chvátal (U, $\mathcal{S}$, c)

1. **se** $U = \emptyset$, **então retorne** $\emptyset$
2. **faça** $U' = U$ **e** $\mathcal{C} = \emptyset$
3. **enquanto** $(U' \neq \emptyset)$ **faça**
4. **seja** Z o subc. em $\mathcal{S}$ com $|Z \cap U'| \neq \emptyset$ e $c(Z)/|Z \cap U'|$ mínimo
5. **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{Z\}$ **e** $U' \leftarrow U' - Z$
6. **retorne** $\mathcal{C}$

Algoritmo MinCC-Chvátal (U, $\mathcal{S}$)

1. **se** $U = \emptyset$, **então retorne** $\emptyset$
2. **faça** $U' = U$ **e** $\mathcal{C} = \emptyset$
3. **enquanto** $(U' \neq \emptyset)$ **faça**
4. **seja** Z o subc. em $\mathcal{S}$ com $|Z \cap U'| \neq \emptyset$ e $1/|Z \cap U'|$ mínimo
5. **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{Z\}$ **e** $U' \leftarrow U' - Z$
6. **retorne** $\mathcal{C}$

Tempo $O(n \cdot m)$

Algoritmo MinCC-Chvátal (com custos e sem custos)

Algoritmo MinCC-Chvátal (U, $\mathcal{S}$, c)

1. **se** $U = \emptyset$, **então retorne** $\emptyset$
2. **faça** $U' = U$ **e** $\mathcal{C} = \emptyset$
3. **enquanto** $(U' \neq \emptyset)$ **faça**
4. **seja** Z o subc. em $\mathcal{S}$ com $|Z \cap U'| \neq \emptyset$ e $c(Z)/|Z \cap U'|$ mínimo
5. **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{Z\}$ **e** $U' \leftarrow U' - Z$
6. **retorne** $\mathcal{C}$

Algoritmo MinCC-Chvátal (U, $\mathcal{S}$)

1. **se** $U = \emptyset$, **então retorne** $\emptyset$
2. **faça** $U' = U$ **e** $\mathcal{C} = \emptyset$
3. **enquanto** $(U' \neq \emptyset)$ **faça**
4. **seja** Z o subc. em $\mathcal{S}$ com $|Z \cap U'|$ máximo
5. **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{Z\}$ **e** $U' \leftarrow U' - Z$
6. **retorne** $\mathcal{C}$

Tempo $O(n \cdot m)$

Algoritmo MinCC-Chvátal é H_n-aproximativo

Algoritmo MinCC-Chvátal $(U, \mathcal{S}, c)$

1. **se** $U = \emptyset$, **então retorne** $\emptyset$
2. **faça** $U' = U$ **e** $\mathcal{C} = \emptyset$
3. **enquanto** $(U' \neq \emptyset)$ **faça**
4. **seja** Z o subc. em $\mathcal{S}$ com $|Z \cap U'| \neq \emptyset$ e $c(Z)/|Z \cap U'|$ mínimo
5. **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{Z\}$ **e** $U' \leftarrow U' - Z$
6. **retorne** $\mathcal{C}$

Algoritmo MinCC-Chvátal é H_n-aproximativo

- $c_{ef}(Z) = c(Z)/|Z \cap U'|$ **e** preço$(u_i) = c_{ef}(Z)$: Z é 1^o a cobrir u.
- $\sum_{u_i \in Z \cap U'}$ preço$(u_i) = c(Z) \implies \sum_{u_i \in U}$ preço$(u_i) = c(\mathcal{C})$
- $c_{ef}(Z) \leq opt/|U'|$, pois o custo efetivo mínimo é $\leq$ a média
- $u_1, \ldots, u_n$ ordem cobertos: preço$(u_i) \leq opt/|U'| \leq opt/(n-i+1)$

$$c(\mathcal{C}) = \sum_{u_i \in U} \text{preço}(u_i) \leq \sum_{u_i \in U} \frac{opt}{n-i+1} = opt \cdot \sum_{k=1}^{n} \frac{1}{k} = H_n \cdot opt$$

- $0.5772 < (H_n - \ln n) \leq 1.$ $\qquad H_n = \ln n + 0.5772 + \frac{1}{2n} + O(\frac{1}{n^2})$

Algoritmo MinCC-Chvátal é H_n-aproximativo

Algoritmo MinCC-Chvátal $(U, \mathcal{S}, c)$

1. **se** $U = \emptyset$, **então retorne** $\emptyset$
2. **faça** $U' = U$ **e** $\mathcal{C} = \emptyset$
3. **enquanto** $(U' \neq \emptyset)$ **faça**
4. **seja** Z o subc. em $\mathcal{S}$ com $|Z \cap U'| \neq \emptyset$ e $c(Z)/|Z \cap U'|$ mínimo
5. **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{Z\}$ **e** $U' \leftarrow U' - Z$
6. **retorne** $\mathcal{C}$

Algoritmo MinCC-Chvátal é H_n-aproximativo

- $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$. $\quad 0.5772 < (H_n - \ln n) \leq 1$
- É uma $(1 + \ln n)$-aproximação. Fator de aproximação não constante.
- Set-Cover $\in$ log-APX (Problemas otim c/ fator de aprox logarítmico)
- Dinur & Steurer'2013: $(1 - \varepsilon) \ln n$ inaprox p/ qquer $\varepsilon > 0$, se P$\neq$NP
- Conclusão1: MinCC-Chvátal é praticamente o melhor possível
- Conclusão2: SetCover $\notin$ APX, se P$\neq$NP

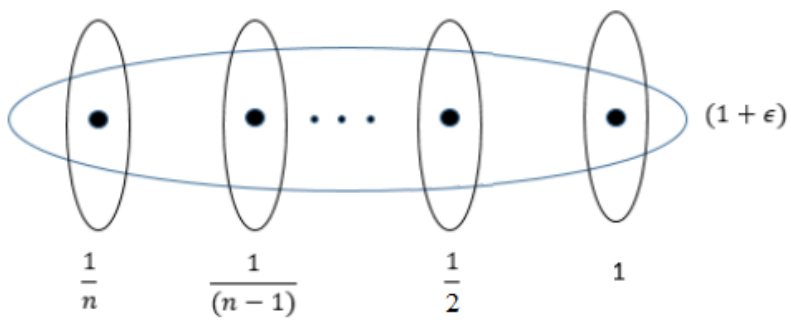
Algoritmo MinCC-Chvátal é H_n-aprox (tight example)

Algoritmo MinCC-Chvátal (U, $\mathcal{S}$, c)

1. **se** $U = \emptyset$, **então retorne** $\emptyset$
2. **faça** $U' = U$ **e** $\mathcal{C} = \emptyset$
3. **enquanto** $(U' \neq \emptyset)$ **faça**
4. **seja** Z o subc. em $\mathcal{S}$ com $|Z \cap U'| \neq \emptyset$ e $c(Z)/|Z \cap U'|$ mínimo
5. **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{Z\}$ **e** $U' \leftarrow U' - Z$
6. **retorne** $\mathcal{C}$

tight example (com custos)

- $opt = 1 + \varepsilon$
- $c(\mathcal{C}) = \frac{1}{n} + \frac{1}{n-1} + \ldots + \frac{1}{2} + 1 \;=\; H_n$



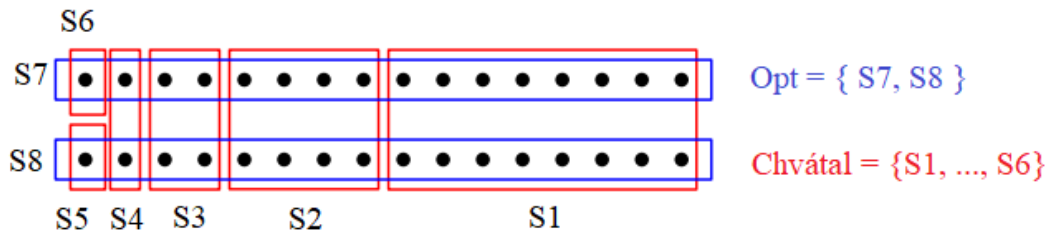
Algoritmo MinCC-Chvátal é H_n-aprox (tight example)

Algoritmo MinCC-Chvátal (U, $\mathcal{S}$, c)

1. **se** $U = \emptyset$, **então retorne** $\emptyset$
2. **faça** $U' = U$ **e** $\mathcal{C} = \emptyset$
3. **enquanto** $(U' \neq \emptyset)$ **faça**
4. **seja** Z o subc. em $\mathcal{S}$ com $|Z \cap U'| \neq \emptyset$ e $c(Z)/|Z \cap U'|$ mínimo
5. **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{Z\}$ **e** $U' \leftarrow U' - Z$
6. **retorne** $\mathcal{C}$

tight example (sem custos)

- Conj tam. $1, 1, 2, 4, 8, \ldots, 2^k$ e dois conj tam. $2^k - 1$
- $n = 2 \cdot (2^k - 1) = 2^{k+1} - 2$. $opt = 2$ conjuntos
- $c(\mathcal{C}) = k + 2 \approx \log_2 n + 1$. $aprox \approx \frac{1}{2} \log_2 n$.
- Existem exemplos mais complicados com $aprox \approx \ln n$



Algoritmo MinCC-Guloso (só sem custos)

Tempo $O(n \cdot m)$

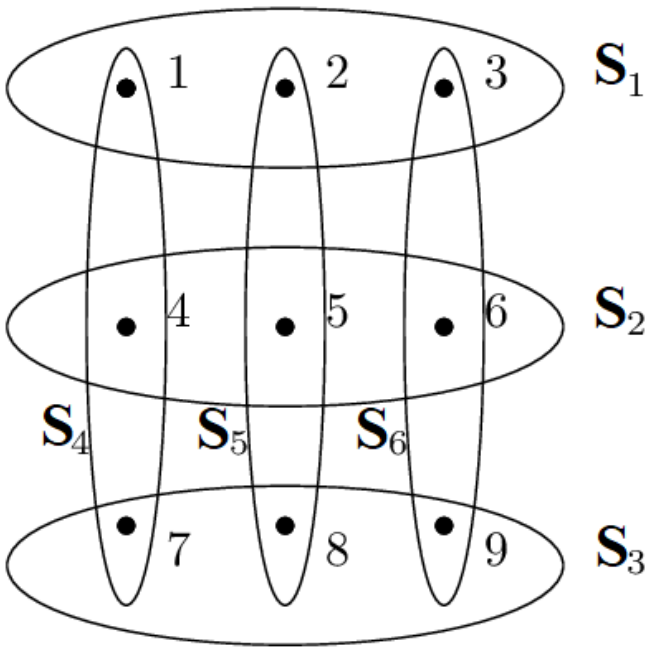
Algoritmo MinCC-Guloso $(U, \mathcal{S})$

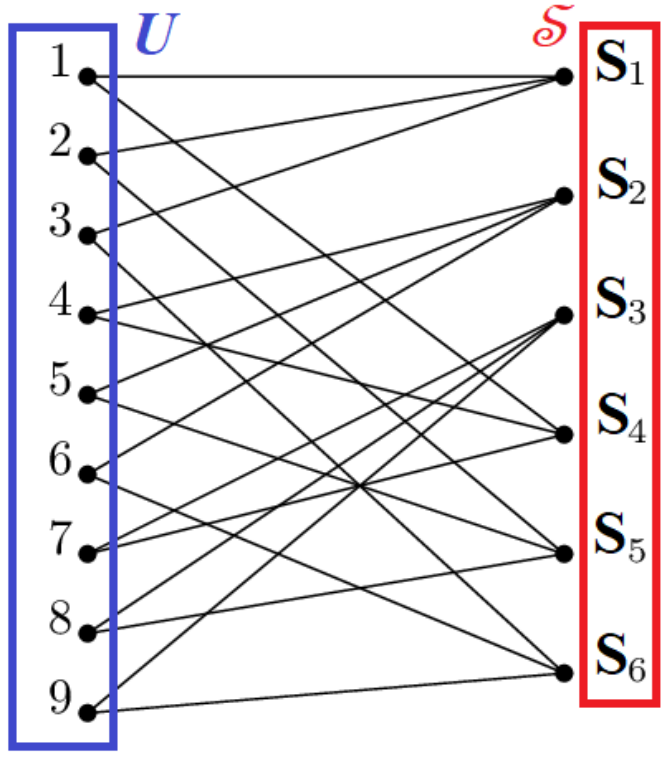
- **se** $U = \emptyset$, **então retorne** $\emptyset$
- **faça** $U' = U$ **e** $I = \emptyset$ **e** $\mathcal{C} = \emptyset$
- **enquanto** $(U' \neq \emptyset)$ **faça**
- **seja** $u \in U'$ e **faça** $I \leftarrow I \cup \{u\}$
- **para cada** $Z \in \mathcal{S}$ que contém u **faça**
- **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{Z\}$
- **faça** $U' \leftarrow U' - Z$
- **retorne** $\mathcal{C}$

MinCC-Guloso é f-aproximativo, onde f é a frequência máxima

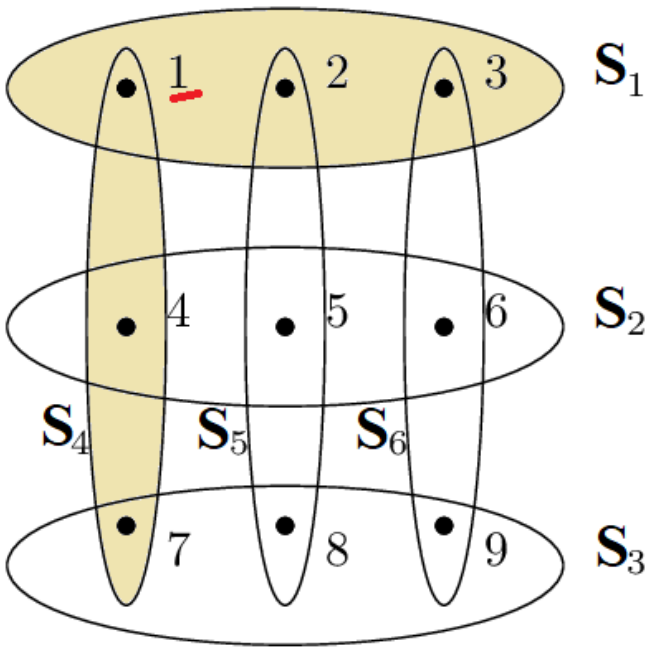
- Seja a frequência $f(u)$ o número de subconjuntos S_i que contém u
- Seja $f = \max_{u \in U}\{f(u)\}$ a frequência máxima.
- MinCC-Guloso é f-aproximativo. Prova nos próximos slides.

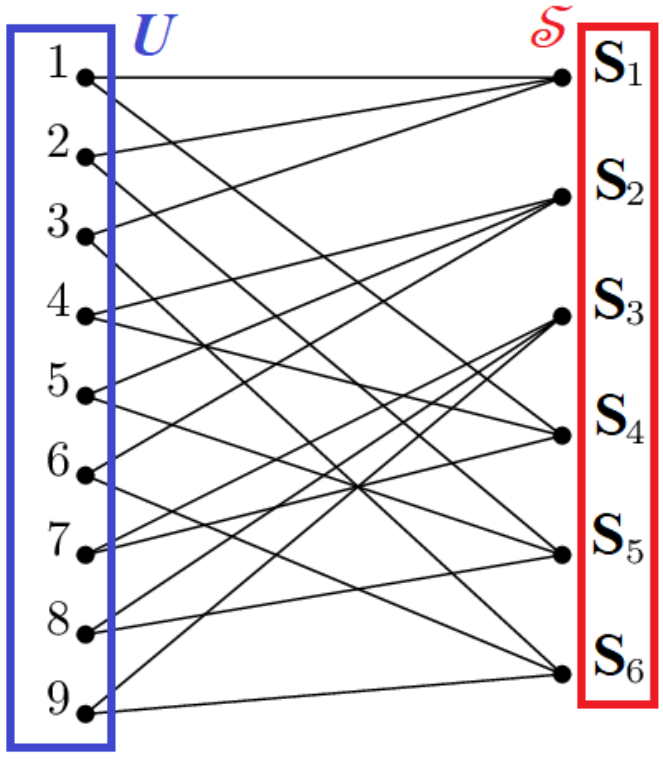
MinCC-Guloso - Exemplo



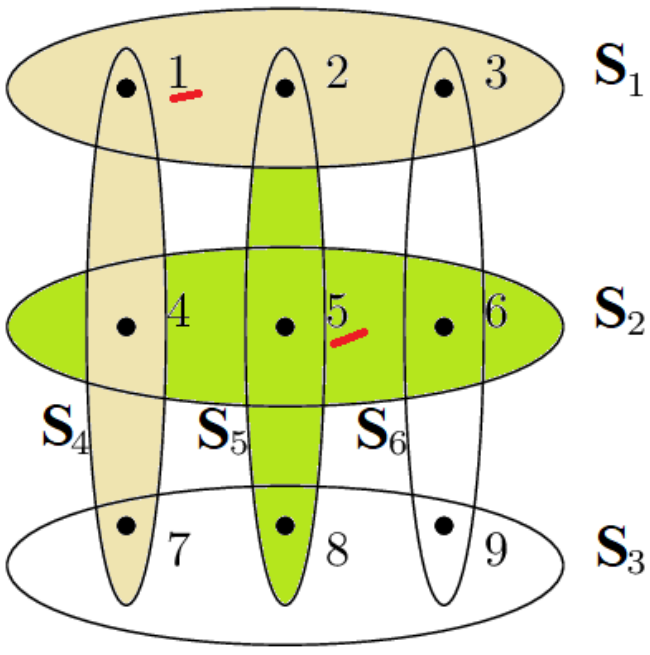


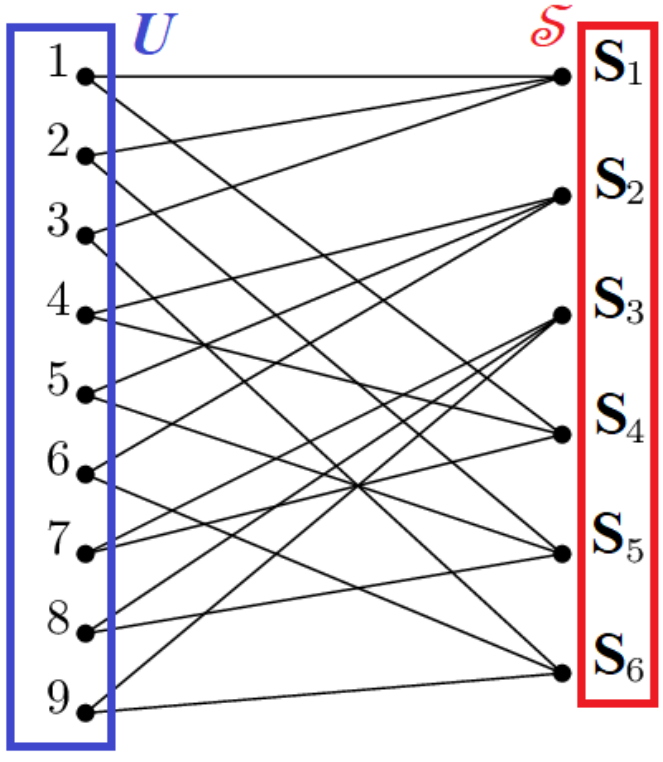
MinCC-Guloso - Exemplo



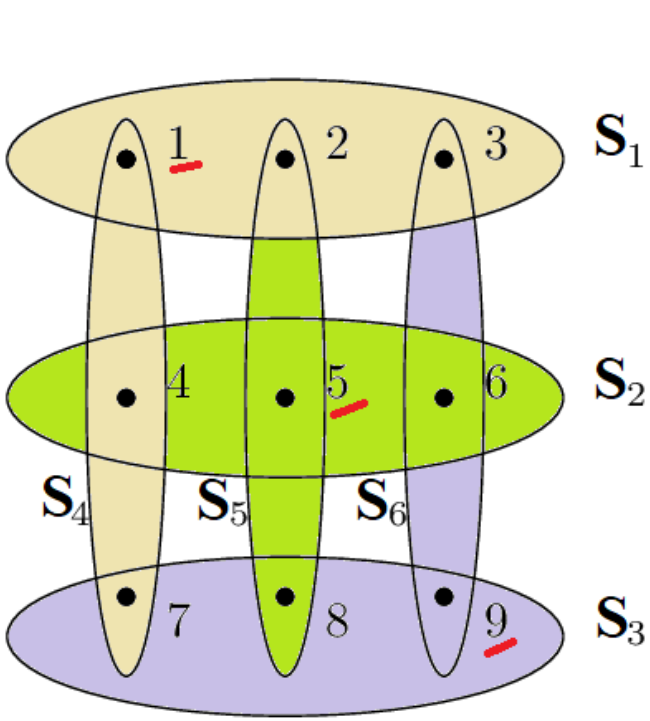


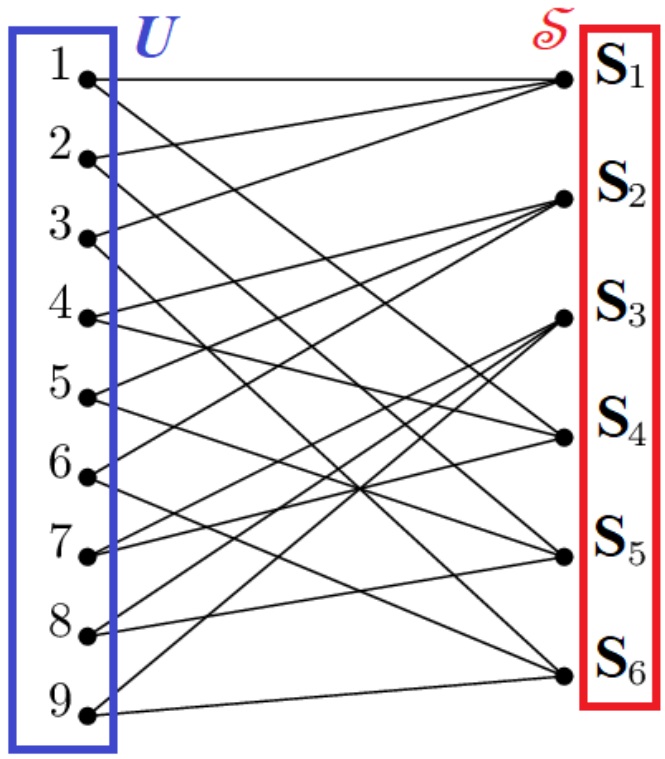
MinCC-Guloso - Exemplo





MinCC-Guloso - Exemplo





Algoritmo MinCC-Guloso é f-aproximativo

- Seja a frequência $f(u)$ o número de subconjuntos S_i que contém u
- Seja $f = \max_{u \in U}\{f(u)\}$ a frequência máxima.

MinCC-Guloso é f-aproximativo, onde f é a frequência máxima

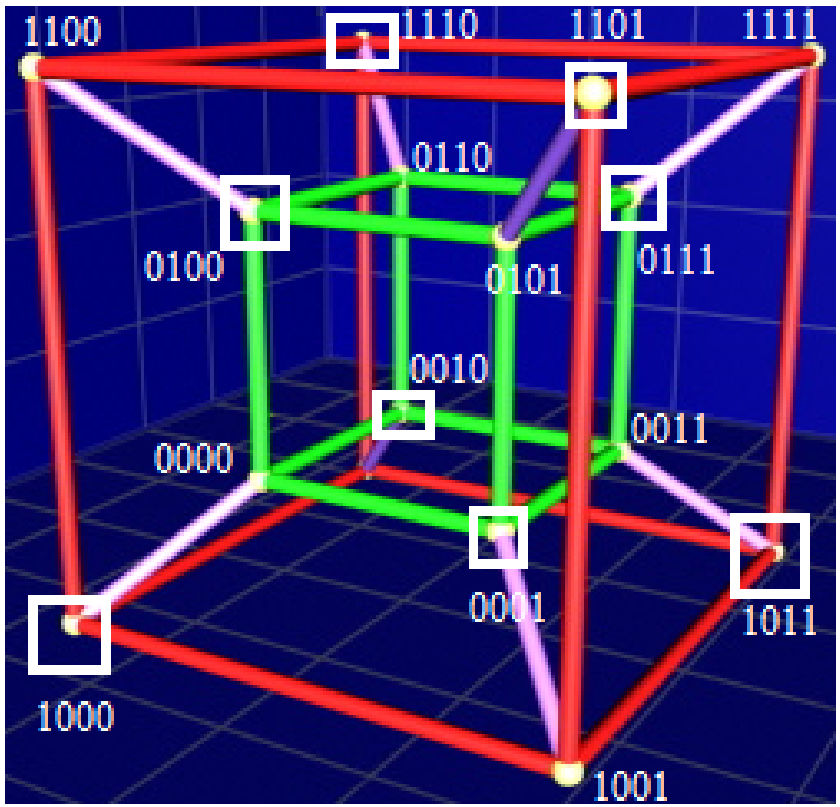
PROVA:

- Conjunto I é independente: $u_i, u_j \in I \Rightarrow \nexists S_k \in \mathcal{S} : u_i, u_j \in S_k$
- Portanto: $opt(U, \mathcal{S}) \geq |I|$
- Além disso, $\mathcal{C}$ tem todos os subc. que contém os elementos de I
- $|\mathcal{C}| \leq f \cdot |I| \leq f \cdot opt(U, \mathcal{S})$

MinCC-Guloso é f-aproximativo (e não menos !!)

Tight Example: hipercubo f-dimensional

- Elementos de U são as palavras de 0's e 1's de tamanho f
- Se duas diferem em apenas uma letra, forme um conjunto de $\mathcal{S}$
- Frequência é o próprio f. Exemplo abaixo: $f = 4$.
- $opt = 2^{f-1}$, mas MinCC-Guloso retorna $f \cdot 2^{f-1}$ (todos os subc.)



Minimum Vertex Cover

Mínima Cobertura por vértices

- **Instância:** Um grafo $G = (V, E)$.
- **Objetivo:** Obter menor número k de vértices que cobrem as arestas de G (toda aresta tem um vértice na cobertura).

Algoritmo MinCV-Guloso (G)

- **se** $E(G) = \emptyset$, **então retorne** $\emptyset$
- **faça** $E' = E(G)$ **e** $I = \emptyset$ **e** $\mathcal{C} = \emptyset$
- **enquanto** $(E' \neq \emptyset)$ **faça**
- **seja** $e = uv \in E'$ e **faça** $I \leftarrow I \cup \{e\}$
- **faça** $\mathcal{C} \leftarrow \mathcal{C} \cup \{u, v\}$
- **faça** $E' \leftarrow E' - \{\text{arestas de } u \text{ e de } v\}$
- **retorne** $\mathcal{C}$

As arestas em I formam um emparelhamento. Cada aresta em I deve ser coberta por um vértice distinto: $opt \geq |I|$. Como $|\mathcal{C}| = 2 \cdot |I| \leq 2 \cdot opt$, MinCV-Guloso é 2-aproximativo.

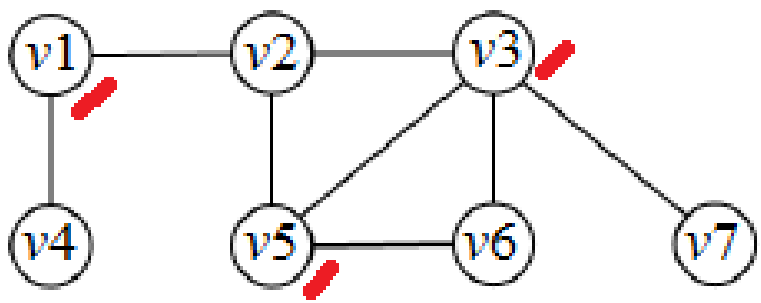
Minimum Vertex Cover versus Minimum Set Cover

Redução para Set Cover, dado um grafo G

- Faça $U = E(G)$.
- Para cada vértice $v \in V(G)$, crie conjunto S_v das arestas de v em G
- Cobrir as arestas com vértices de G é o mesmo que cobrir elementos de U com subconjuntos S_v.

Algoritmo MinCV-Guloso $\equiv$ MinCC-Guloso

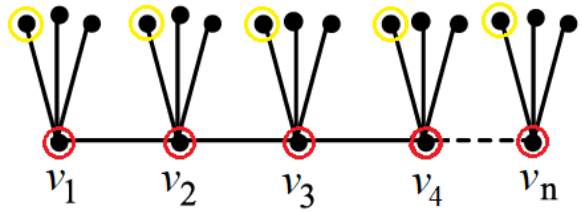
- Frequência $f = 2$, pois cada aresta $e = v_1v_2$ está nos dois conjuntos S_{v_1} e S_{v_2}



$S_{v1} = \{v_1v_2, v_1v_4\}$,$S_{v2} = \{v_1v_2, v_2v_3, v_2v_5\}$,$S_{v3} = \{v_2v_3, v_3v_5, v_3v_6, v_3v_7\}$,
$S_{v4} = \{v_1v_4\}$, $S_{v5} = \{v_2v_5, v_3v_5, v_5v_6\}$, $S_{v6} = \{v_3v_6, v_5v_6\}$, $S_{v7} = \{v_3v_7\}$.

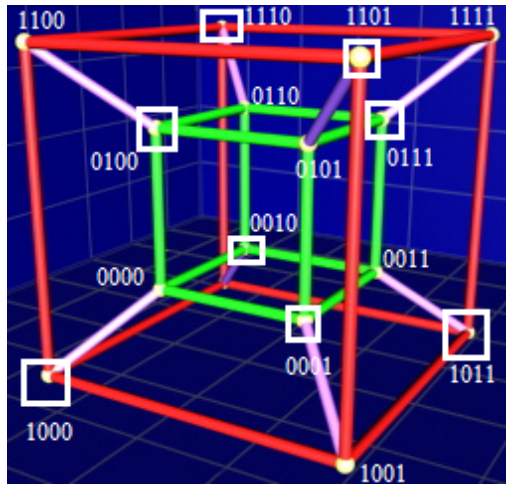
MinCV-Guloso é 2-aproximativo (e não menos !!)

Tight Example: caterpillar tree



Tight Example: hipercubo f-dimensional

- Vértices de G são as palavras de 0's e 1's de tamanho f
- Se dois diferem em apenas uma letra, adicione uma aresta em G
- $opt = 2^{f-1}$, mas MinCV-Guloso retorna 2^f (todos os vértices.)



Conclusão geral (até agora)

* **Escalonamento:** Algoritmo 2-aproximativo.
* **TSPM:** Alg 1.5-aproximativo. Parece ser o melhor possível, na prática.
* **Weighted Set Cover:** Algoritmos $(\ln n+1)$-aprox. e f-aprox. Parecem os melhores possíveis (parece não ter aprox para fator constante).
* **Vertex Cover:** Algoritmo 2-aprox. Parece o melhor possível (parece não ter PTAS).
* **Mochila:** FPTAS. É o melhor possível.

- Problemas NP-Difíceis com FPTAS (Mochila)
- Problemas NP-Difíceis com PTAS, provav **sem** FPTAS (falta provar)
- Problemas NP-Difíceis com α-aprox (p/ α const), provav **sem** PTAS (TSP Métrico, Vertex Cover - falta provar)
- Problemas NP-Difíceis com $\alpha(n)$-aprox (p/ $\alpha(n)$ **não** const), que provav **não** tem para α const (Set Cover)
- Problemas NP-Difíceis que provav **não** tem alg poli $\alpha(n)$-aprox para nenhuma função computável $\alpha(n)$ poli (Caixeiro viajante)

Voltando ao Weighted Set Cover - Método Primal

Min CC - Cobertura por conjuntos (weighted)

- **Instância:** Conj. univ. $U = \{u_1, \ldots, u_n\}$, família $\mathcal{S} = \{S_1, \ldots, S_m\}$ de subconjuntos de U e um custo $c(S_i)$ para cada conjunto S_i
- **Objetivo:** Obter uma cobertura $S_{i_1}, \ldots, S_{i_k}$ de U de custo mínimo ($c(S_{i_1}) + \ldots + c(S_{i_k})$ mínimo).
- **Generalização:** custo 1 para cada conjunto $\rightarrow$ unweighted.

Método Primal - Formulação de Programação Inteira

- Vetores c e x, indexados por $\mathcal{S}$. **Ex:** c_S e x_S para cada $S \in \mathcal{S}$
- **Minimizar** $c \cdot x = \sum_{S \in \mathcal{S}} c_S x_S$, **restrito a**:
- **Para cada** $S \in \mathcal{S}$: $x_S \in \{0, 1\}$
- **Para cada** $u \in U$: $\sum_{S \in \mathcal{S}:\ u \in S} x_S \geq 1$

- É viável; basta tomar $x_S = 1$ para todo conjunto $S \in \mathcal{S}$
- Obtém soℓ. exata $opt(U, \mathcal{S}, c) = c \cdot x_{opt}$, mas problema NP-Difícil.

Voltando ao Weighted Set Cover - Método Primal

Min CC - Cobertura por conjuntos (weighted)

- **Instância:** Conj. univ. $U = \{u_1, \ldots, u_n\}$, família $\mathcal{S} = \{S_1, \ldots, S_m\}$ de subconjuntos de U e um custo $c(S_i)$ para cada conjunto S_i
- **Objetivo:** Obter uma cobertura $S_{i_1}, \ldots, S_{i_k}$ de U de custo mínimo ($c(S_{i_1}) + \ldots + c(S_{i_k})$ mínimo).
- **Generalização:** custo 1 para cada conjunto $\rightarrow$ unweighted.

Método Primal - Relaxação (Programação Linear)

- Vetores c e x, indexados por $\mathcal{S}$. **Ex:** c_S e x_S para cada $S \in \mathcal{S}$
- **Minimizar** $c \cdot x = \sum_{S \in \mathcal{S}} c_S x_S$, **restrito a**:
- **Para cada** $S \in \mathcal{S}$: $x_S \in [0, 1]$
- **Para cada** $u \in U$: $\sum_{S \in \mathcal{S}:\ u \in S} x_S \geq 1$

- É viável; basta tomar $x_S = 1$ para todo conjunto $S \in \mathcal{S}$
- $opt(U, \mathcal{S}, c) \geq c \cdot x_{opt}$, pois a relaxação pode obter valores menores.

Voltando ao Weighted Set Cover - Método Primal

Técnica do arredondamento

Algoritmo MinCC-Hochbaum (U, $\mathcal{S}$, c)

1. **seja** x uma solução ótima racional da PL (relax. PI)
2. **para cada** $u \in U$: **seja** $f_u = |\{S \in \mathcal{S} : \ u \in S\}|$
3. **seja** $f \ = \ \max\{f_u : \ u \in U\}$
4. **seja** $\mathcal{C} \ = \ \{S \in \mathcal{S} : \ x_S \geq 1/f\}$
5. **retorne** $\mathcal{C}$

LEMA: $\mathcal{C}$ é uma cobertura de U por conjuntos de $\mathcal{S}$
Prova: Para cada $u \in U$ vale a restrição $\sum_{S \in \mathcal{S}: \ u \in S} \ x_S \ \geq 1$
$\Rightarrow \exists S \in \mathcal{S} \ : u \in S$ e $x_S \geq 1/f_u \geq 1/f$.

TEOREMA: MinCC-Hochbaum é f-**aproximativo**
Prova: Como $f \cdot x_S \geq 1, \forall S \in \mathcal{C}$, então:
$c(\mathcal{C}) = \sum_{S \in \mathcal{C}} c_S \leq f \cdot \sum_{S \in \mathcal{C}} c_S x_S \leq f \cdot \sum_{S \in \mathcal{S}} c_S x_S \leq f \cdot opt(U, \mathcal{S}, c)$

Weighted Vertex Cover - Método Primal

Min CV - Vertex Cover (weighted)

- **Instância:** Grafo G e um custo $c(v)$ para cada vértice v de G
- **Objetivo:** Obter uma cobertura das arestas por vértices $v_1, \ldots, v_k$ de G de custo mínimo $(c(v_1) + \ldots + c(v_k)$ mínimo$)$.
- **Generalização:** custo 1 para cada vértice $\rightarrow$ unweighted.

Método Primal - Formulação de Programação Inteira

- Vetores c e x, indexados por $V(G)$. **Ex:** c_v e x_v para cada $v \in V(G)$
- **Minimizar** $c \cdot x = \sum_{v \in V(G)} c_v x_v$, **restrito a**:
- **Para cada vértice** $v \in V(G)$: $x_v \in \{0, 1\}$
- **Para cada aresta** $uv \in E(G)$: $x_u + x_v \geq 1$

- É viável; basta tomar $x_v = 1$ para todo vértice $v \in V(G)$
- Obtém solução exata $opt(G, c) = c \cdot x_{opt}$, mas Problema NP-Difícil.

Weighted Vertex Cover - Método Primal

Min CV - Vertex Cover (weighted)

- **Instância:** Grafo G e um custo $c(v)$ para cada vértice v de G
- **Objetivo:** Obter uma cobertura das arestas por vértices $v_1, \ldots, v_k$ de G de custo mínimo $(c(v_1) + \ldots + c(v_k)$ mínimo$)$.
- **Generalização:** custo 1 para cada vértice $\rightarrow$ unweighted.

Método Primal - Relaxação (Programação Linear)

- Vetores c e x, indexados por $V(G)$. **Ex:** c_v e x_v para cada $v \in V(G)$
- **Minimizar** $c \cdot x = \sum_{v \in V(G)} c_v x_v$, **restrito a**:
- **Para cada vértice** $v \in V(G)$: $x_v \in [0, 1]$
- **Para cada aresta** $uv \in E(G)$: $x_u + x_v \geq 1$

- É viável; basta tomar $x_v = 1$ para todo vértice $v \in V(G)$
- $opt(G, c) \geq c \cdot x_{opt}$, pois a relaxação pode obter valores menores.

Weighted Vertex Cover - Método Primal

Técnica do arredondamento

Algoritmo MinCV-Hochbaum (G, c)

1. **seja** x uma solução ótima racional da PL (relax. PI)
2. **seja** $\mathcal{C} = \{v \in V(G) : x_v \geq 1/2\}$
3. **retorne** $\mathcal{C}$

LEMA: $\mathcal{C}$ é uma cobertura de $E(G)$ por vértices de G
Prova: Para cada aresta $uv \in E(G)$ vale a restrição $x_u + x_v \geq 1$.
Portanto, $x_u \geq \frac{1}{2}$ ou $x_v \geq \frac{1}{2}$.

TEOREMA: MinCV-Hochbaum é 2-**aproximativo**
Prova: Como $2 \cdot x_v \geq 1, \forall v \in \mathcal{C}$, então:

$$c(\mathcal{C}) = \sum_{v \in \mathcal{C}} c_v \leq 2 \cdot \sum_{v \in \mathcal{C}} c_v x_v \leq 2 \cdot \sum_{v \in V(G)} c_v x_v \leq 2 \cdot opt(G, c)$$

Weighted Vertex Cover - Método Dual

Problema Primal

$$\text{Max} \quad Z = 3x_1 + 5x_2$$

Sujeito a

$$\begin{cases} x_1 \leq 4 \\ 2x_2 \leq 12 \\ 3x_1 + 2x_2 \leq 18 \end{cases}$$

$$x_1 \text{ e } x_2 \geq 0$$

Notação Matricial

$$\text{Max} \quad Z = \begin{bmatrix} 3 & 5 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Sujeito a

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

$$\text{e} \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Problema Dual

$$\text{Min} \quad W = 4y_1 + 12y_2 + 18y_3$$

Sujeito a

$$\begin{cases} y_1 + 3y_3 \geq 3 \\ 2y_2 + 2y_3 \geq 5 \end{cases}$$

$$y_1, \; y_2 \text{ e } y_3 \geq 0$$

Notação Matricial

$$\text{Min} \quad W = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

Sujeito a

$$\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix} \geq \begin{bmatrix} 3 & 5 \end{bmatrix}$$

$$\text{e} \quad \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \geq \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Teorema Forte da Dualidade: opt do primal = opt do dual

Weighted Vertex Cover - Método Dual

Método Primal - Relaxação (Programação Linear)

- Vetores c e x, indexados por $V(G)$. **Ex:** c_v e x_v para cada $v \in V(G)$
- **Minimizar** $c \cdot x = \sum_{v \in V(G)} c_v x_v$, **restrito a**:
- **Para cada vértice** $v \in V(G)$: $x_v \in [0, 1]$
- **Para cada aresta** $uv \in E(G)$: $x_u + x_v \geq 1$

Método Dual - Relaxação (Programação Linear)

- Vetor y, indexado por $E(G)$. **Ex:** y_{uv} para cada $uv \in E(G)$
- **Maximizar** $y \cdot \mathbf{1} = \sum_{uv \in E(G)} y_{uv}$, **restrito a**:
- **Para cada aresta** $uv \in E(G)$: $y_{uv} \geq 0$
- **Para cada vértice** $v \in V(G)$: $\sum_{uv \in E(G)} y_{uv} \leq c_v$ $(R1)$

- É viável; basta tomar $y_{uv} = 0$ para toda aresta $uv \in E(G)$
- $opt(G, c) \geq c \cdot x_{opt} = y_{opt} \cdot \mathbf{1}$, pelo Teorema Forte da Dualidade.

Weighted Vertex Cover - Método Dual

Interpretação do Dual

- O custo de cada vértice é repartido entre ele e suas arestas
- Se uma aresta uv não tem u nem v com igualdade na restrição R1, podemos aumentar o valor de y_{uv}. Ou seja, no ótimo, toda aresta tem uma extremidade com igualdade na restrição R1.
- Colocar na cobertura os vértices com igualdade na restrição R1.

Método Dual - Relaxação (Programação Linear)

- Vetor y, indexado por $E(G)$. **Ex:** y_{uv} para cada $uv \in E(G)$
- **Maximizar** $y \cdot \mathbf{1} = \sum_{uv \in E(G)} y_{uv}$, **restrito a**:
- **Para cada aresta** $uv \in E(G)$: $y_{uv} \geq 0$
- **Para cada vértice** $v \in V(G)$: $\sum_{uv \in E(G)} y_{uv} \leq c_v$ $(R1)$

- É viável; basta tomar $y_{uv} = 0$ para toda aresta $uv \in E(G)$
- $opt(G, c) \geq c \cdot x_{opt} = y_{opt} \cdot \mathbf{1}$, pelo Teorema Forte da Dualidade.

Weighted Vertex Cover - Método Dual

Algoritmo MinCV-Hochbaum-dual (G, c)

1. **seja** y uma solução ótima racional da PL (dual, relax. PI)
2. **seja** $\mathcal{C} = \{v \in V(G) : \sum_{uv \in E(G)} y_{uv} = c_v\}$ (igualdade em R1)
3. **retorne** $\mathcal{C}$

LEMA: $\mathcal{C}$ é uma cobertura de $E(G)$ por vértices de G
Prova: Já visto.

TEOREMA: MinCV-Hochbaum-dual é 2-**aproximativo**
Prova: Seja $V = V(G)$ e $E = E(G)$.

$$c(\mathcal{C}) = \sum_{v \in \mathcal{C}} c_v = \sum_{v \in \mathcal{C}} \sum_{uv \in E} y_{uv} \leq \sum_{v \in V} \sum_{uv \in E} y_{uv} = 2 \sum_{uv \in E} y_{uv} \leq 2 \cdot opt(G, c)$$

Weighted Set Cover - Método Dual

Método Primal - Relaxação (Programação Linear)

- Vetores c e x, indexados por $\mathcal{S}$. **Ex:** c_S e x_S para cada $S \in \mathcal{S}$
- **Minimizar** $c \cdot x = \sum_{S \in \mathcal{S}} c_S x_S$, **restrito a**:
- **Para cada conjunto** $S \in \mathcal{S}$: $x_S \in [0,1]$
- **Para cada elemento** $u \in U$: $\sum_{S \in \mathcal{S}:\ u \in S} x_S \geq 1$

Método Dual - Relaxação (Programação Linear)

- Vetor y, indexado por U. **Ex:** y_u para cada $u \in U$
- **Maximizar** $y \cdot \mathbf{1} = \sum_{u \in U} y_u$, **restrito a**:
- **Para cada elemento** $u \in U$: $y_u \geq 0$
- **Para cada conjunto** $S \in \mathcal{S}$: $\sum_{u \in S} y_u \leq c_S$ $\quad (R1)$

  - É viável; basta tomar $y_u = 0$ para todo $u \in U$
  - $opt(U, \mathcal{S}, c) \geq c \cdot x_{opt} = y_{opt} \cdot \mathbf{1}$, pelo Teo Forte da Dualidade.

Weighted Set Cover - Método Dual

Interpretação do Dual

- O custo de cada conjunto S é repartido entre ele e seus elementos
- Se um elemento u não está em um conjunto S com igualdade na restrição R1, podemos aumentar o valor y_u. Ou seja, no ótimo, todo elemento pertence a um conjunto com igualdade na restrição R1.
- Colocar na cobertura os conjuntos S com igualdade na restrição R1.

Método Dual - Relaxação (Programação Linear)

- Vetor y, indexado por U. **Ex:** y_u para cada $u \in U$
- **Maximizar** $y \cdot \mathbf{1} = \sum_{u \in U} y_u$, **restrito a**:
- **Para cada elemento** $u \in U$: $y_u \geq 0$
- **Para cada conjunto** $S \in \mathcal{S}$: $\sum_{u \in S} y_u \leq c_S$ $\quad (R1)$

- É viável; basta tomar $y_u = 0$ para todo $u \in U$
- $opt(U, \mathcal{S}, c) \geq c \cdot x_{opt} = y_{opt} \cdot \mathbf{1}$, pelo Teo Forte da Dualidade.

Weighted Set Cover - Método Dual

Algoritmo MinCC-Hochbaum-dual (U, $\mathcal{S}$, c)

1. **seja** y uma solução ótima racional da PL (dual, relax. PI)
2. **seja** $\mathcal{C} = \{S \in \mathcal{S} : \sum_{u \in S} y_u = c_S\}$ (igualdade em R1)
3. **retorne** $\mathcal{C}$

LEMA: $\mathcal{C}$ é uma cobertura de U por conjuntos de $\mathcal{S}$
Prova: Já visto.

TEOREMA: MinCC-Hochbaum-dual é f-**aproximativo**
Prova: Para cada $u \in U$, seja $f_u = |\{S \in \mathcal{S} : u \in S\}|$.
Seja $f = \max\{f_u : u \in U\}$.

$$c(\mathcal{C}) = \sum_{S \in \mathcal{C}} c_S = \sum_{S \in \mathcal{C}} \sum_{u \in S} y_u \leq \sum_{S \in \mathcal{S}} \sum_{u \in S} y_u = \sum_{u \in U} f_u y_u \leq f \sum_{u \in U} y_u \leq f \cdot opt(U, \mathcal{S}, c)$$

Conclusão: Set Cover e Vertex Cover

Set Cover sem custos

- f-aproximação: Algoritmo MinCC-Guloso

Set Cover com custos

- $(1 + \ln n)$-aproximação: Algoritmo MinCC-Chvátal.
- f-aproximação: Algoritmo MinCC-Hochbaum (método primal)
- f-aproximação: Algoritmo MinCC-Hochbaum-dual (método dual)

Vertex Cover sem custos

- 2-aproximação: Algoritmo MinCV-Guloso

Vertex Cover com custos

- 2-aproximação: Algoritmo MinCV-Hochbaum (método primal)
- 2-aproximação: Algoritmo MinCV-Hochbaum-dual (método dual)

O Esquema Primal-Dual Clássico

Programa Primal Clássico: n variáveis x_j com m restrições b_i

- **Minimizar** $c \cdot x = \sum_{j=1}^{n} c_j x_j$, **onde** $x_j \geq 0, \forall j = 1, \ldots, n$
- Restrito à $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$ para todo $i = 1, \ldots, m$

Programa Dual Clássico: m variáveis y_i com n restrições c_j

- **Maximizar** $b \cdot y = \sum_{i=1}^{m} b_i y_i$, onde $y_i \geq 0, \forall i = 1, \ldots, m$
- **Restrito à** $\sum_{i=1}^{m} a_{ij} y_i \leq c_j$ para todo $j = 1, \ldots, n$

Teoremas da Dualidade e das Folgas Complementares

- **Dualidade Forte:** Se viáveis, $c \cdot x \geq b \cdot y$ e $c \cdot x_{opt} = b \cdot y_{opt}$
- **Folgas Complementares:** $c \cdot x = b \cdot y$ se e só se valem as folgas completares primais e duais:
  - Primais: $x_j = 0$ ou $\sum_{i=1}^{m} a_{ij} y_i = c_j$, para todo $j = 1, \ldots, n$
  - Duais: $y_i = 0$ ou $\sum_{j=1}^{n} a_{ij} x_j = b_i$, para todo $i = 1, \ldots, m$
- **Esquema PL:** Começa com solução dual y viável e solução primal x inviável. Observando as folgas complementares, vai melhorando a otimalidade da dual y e a viabilidade da primal x **até** $c \cdot x = b \cdot y$.

O Esquema Primal-Dual de Aproximação

Programa Primal Clássico: n variáveis x_j com m restrições b_i

- **Minimizar** $c \cdot x = \sum_{j=1}^{n} c_j x_j$, **onde** $x_j \geq 0, \forall j = 1, \ldots, n$
- Restrito à $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$ para todo $i = 1, \ldots, m$

Programa Dual Clássico: m variáveis y_i com n restrições c_j

- **Maximizar** $b \cdot y = \sum_{i=1}^{m} b_i y_i$, onde $y_i \geq 0, \forall i = 1, \ldots, m$
- **Restrito à** $\sum_{i=1}^{m} a_{ij} y_i \leq c_j$ para todo $j = 1, \ldots, n$

Folgas Complementares (α, β)-aproximadas

- Primais: $x_j = 0$ ou $c_j/\alpha \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$, para todo $j = 1, \ldots, n$
- Duais: $y_i = 0$ ou $b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta b_i$, para todo $i = 1, \ldots, m$
- **Lema:** Se x e y satisfazem folgas complementares (α, β)-aproximadas, então $c \cdot x \leq (\alpha\beta)\ b \cdot y$. **Prova:**

$$\sum_{j=1}^{n} c_j x_j \leq \alpha \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \alpha \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \leq \alpha\beta \sum_{j=1}^{n} b_i y_i$$

O Esquema Primal-Dual de Aproximação

Programa Primal Clássico: n variáveis x_j com m restrições b_i

- **Minimizar** $c \cdot x = \sum_{j=1}^{n} c_j x_j$, **onde** $x_j \geq 0, \forall j = 1, \ldots, n$
- Restrito à $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$ para todo $i = 1, \ldots, m$

Programa Dual Clássico: m variáveis y_i com n restrições c_j

- **Maximizar** $b \cdot y = \sum_{i=1}^{m} b_i y_i$, onde $y_i \geq 0, \forall i = 1, \ldots, m$
- **Restrito à** $\sum_{i=1}^{m} a_{ij} y_i \leq c_j$ para todo $j = 1, \ldots, n$

Folgas Complementares (α, β)-aproximadas

- Primais: $x_j = 0$ ou $c_j/\alpha \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$, para todo $j = 1, \ldots, n$
- Duais: $y_i = 0$ ou $b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta b_i$, para todo $i = 1, \ldots, m$
- **Aprox Primal-Dual:** Começa com soluções y dual viável e x primal inviável (mas sempre inteira). Observ folgas compl (α, β)-aprox, vai melhorando otimal. dual y e viab. primal x. No fim, a solução dual vezes $\alpha\beta$ serve como limite superior para o do primal inteiro.

Weighted Vertex Cover - Método Primal-Dual

Programa Primal - Programação Inteira

- Vetor x, indexado por $V(G)$. **Ex:** x_v para cada $v \in V(G)$
- **Minimizar** $c \cdot x = \sum_{v \in V(G)} c_v x_v$, **restrito a**:
- **Para cada vértice** $v \in V(G)$: $x_v \in \{0, 1\}$
- **Para cada aresta** $uv \in E(G)$: $x_u + x_v \geq 1$

▶ É viável; basta tomar $x_v = 1$ para toda aresta $uv \in E(G)$

Programa Dual - Relaxação (Programação Linear)

- Vetor y, indexado por $E(G)$. **Ex:** y_{uv} para cada $uv \in E(G)$
- **Maximizar** $y \cdot \mathbf{1} = \sum_{uv \in E(G)} y_{uv}$, **restrito a**:
- **Para cada aresta** $uv \in E(G)$: $y_{uv} \geq 0$
- **Para cada vértice** $v \in V(G)$: $\sum_{uv \in E(G)} y_{uv} \leq c_v$ $(R1)$

▶ É viável; basta tomar $y_{uv} = 0$ para toda aresta $uv \in E(G)$

▶ Primal-Dual início: $x_v = 0 \ \forall v \in V(G)$ e $y_{uv} = 0 \ \forall uv \in E(G)$.

Weighted Vertex Cover - Método Primal-Dual

Algoritmo MinCV-primal-dual (G, c)

1. **seja** y o vetor com $y_{uv} = 0$ para toda aresta uv de G
2. **enquanto** existe aresta ab t.q. a e b não estão com (=R1)
3. **faça** $y_{ab} \leftarrow y_{ab} + \min\left\{c_v - \sum_{uv \in E(G)} y_{uv} : v \in \{a, b\}\right\}$
4. **seja** $\mathcal{C} = \{v \in V(G) : \sum_{uv \in E(G)} y_{uv} = c_v\}$ (igualdade em R1)
5. **retorne** $\mathcal{C}$

- Vetor x não está explícito, mas $x_v = 1$ se v com (=R1) e $x_v = 0$ cc.

Programa Dual - Relaxação (Programação Linear)

- Vetor y, indexado por $E(G)$. **Ex:** y_{uv} para cada $uv \in E(G)$
- **Maximizar** $y \cdot \mathbf{1} = \sum_{uv \in E(G)} y_{uv}$, **restrito a**:
- **Para cada aresta** $uv \in E(G)$: $y_{uv} \geq 0$
- **Para cada vértice** $v \in V(G)$: $\sum_{uv \in E(G)} y_{uv} \leq c_v$ (R1)

Weighted Vertex Cover - Método Primal-Dual

Algoritmo MinCV-primal-dual (G, c)

1. **seja** y o vetor com $y_{uv} = 0$ para toda aresta uv de G
2. **enquanto** existe aresta ab t.q. a e b não estão com (=R1)
3. **faça** $y_{ab} \leftarrow y_{ab} + \min\left\{c_v - \sum_{uv \in E(G)} y_{uv} : v \in \{a, b\}\right\}$
4. **seja** $\mathcal{C} = \{v \in V(G) : \sum_{uv \in E(G)} y_{uv} = c_v\}$ (igualdade em R1)
5. **retorne** $\mathcal{C}$

LEMA: $\mathcal{C}$ é uma cobertura de $E(G)$ por vértices de G

TEOREMA: MinCV-primal-dual é 2-**aproximativo**
Prova: Bastaria $\alpha = 1$ e $\beta = 2$. **Outra prova:** p/ cobert opt $\mathcal{C}^*$:

$$\sum_{uv \in E} y_{uv} \leq \sum_{v \in \mathcal{C}^*} \sum_{uv \in E} y_{uv} \leq \sum_{v \in \mathcal{C}^*} c_v = opt.$$ Logo, p/ cobert $\mathcal{C}$ retorn p/ algoritmo

$$c(\mathcal{C}) = \sum_{v \in \mathcal{C}} c_v = \sum_{v \in \mathcal{C}} \sum_{uv \in E} y_{uv} \leq \sum_{v \in V} \sum_{uv \in E} y_{uv} = 2 \sum_{uv \in E} y_{uv} \leq 2 \cdot opt(G, c)$$

Weighted Set Cover - Método Primal-Dual

Programa Primal - Programação Inteira

- Vetor x, indexado por $\mathcal{S}$. **Ex:** x_S para cada $S \in \mathcal{S}$
- **Minimizar** $c \cdot x = \sum_{S \in \mathcal{S}} x_S$, **restrito a**:
- **Para cada conjunto** $S \in \mathcal{S}$: $x_S \in \{0, 1\}$
- **Para cada elemento** $u \in U$: $\sum_{S \in \mathcal{S}:\ u \in S} x_S \geq 1$

▶ É viável; basta tomar $x_S = 1$ para todo $S \in \mathcal{S}$

Programa Dual - Relaxação (Programação Linear)

- Vetor y, indexado por U. **Ex:** y_u para cada $u \in U$
- **Maximizar** $y \cdot \mathbf{1} = \sum_{u \in U} y_u$, **restrito a**:
- **Para cada elemento** $u \in U$: $y_u \geq 0$
- **Para cada conjunto** $S \in \mathcal{S}$: $\sum_{u \in S} y_u \leq c_S$ $(R1)$

▶ É viável; basta tomar $y_u = 0$ para toda aresta $u \in U$

▶ Primal-Dual início: $x_S = 0\ \forall S \in \mathcal{S}$ e $y_u = 0\ \forall u \in U$.

Weighted Set Cover - Método Primal-Dual

Algoritmo MinCC-primal-dual (U, $\mathcal{S}$, c)

1. **seja** y o vetor com $y_u = 0$ para todo elemento $u \in U$
2. **enquanto** $\exists$ elem $u \in U$ sem conj. que o contém com (=R1)
3. **faça** $y_u \leftarrow y_u + \min\left\{c_S - \sum_{u \in S} y_u : \ S \in \mathcal{S}, u \in S\right\}$
4. **seja** $\mathcal{C} = \{S \in \mathcal{S} : \ \sum_{u \in S} y_u = c_S\}$ (igualdade em R1)
5. **retorne** $\mathcal{C}$

- Vetor x não está explícito, mas $x_S = 1$ se S com (=R1) e $x_S = 0$ cc.

Programa Dual - Relaxação (Programação Linear)

- Vetor y, indexado por U. **Ex:** y_u para cada $u \in U$
- **Maximizar** $y \cdot \mathbf{1} = \sum_{u \in U} y_u$, **restrito a**:
- **Para cada elemento** $u \in U$: $y_u \geq 0$
- **Para cada conjunto** $S \in \mathcal{S}$: $\sum_{u \in S} y_u \leq c_S$ $(R1)$

Weighted Set Cover - Método Primal-Dual

Algoritmo MinCC-primal-dual (U, $\mathcal{S}$, c)

1. **seja** y o vetor com $y_u = 0$ para todo elemento $u \in U$
2. **enquanto** $\exists$ elem $u \in U$ sem conj. que o contém com (=R1)
3. **faça** $y_u \leftarrow y_u + \min\left\{c_S - \sum_{u \in S} y_u : \ S \in \mathcal{S}, u \in S\right\}$
4. **seja** $\mathcal{C} = \{S \in \mathcal{S} : \ \sum_{u \in S} y_u = c_S\}$ (igualdade em R1)
5. **retorne** $\mathcal{C}$

LEMA: $\mathcal{C}$ é uma cobertura de U por conjuntos de $\mathcal{S}$

TEOREMA: MinCC-primal-dual é f-**aproximativo**
Prova: $\forall u \in U$, seja $f_u = |\{S \in \mathcal{S} : \ u \in S\}|$. Seja $f = \max\{f_u : u \in U\}$.
Bastaria $\alpha = 1$ e $\beta = f$. **Outra prova:** Para cobertura opt $\mathcal{C}^*$:

$$\sum_{u \in U} y_u \leq \sum_{S \in \mathcal{C}^*} \sum_{u \in S} y_u \leq \sum_{S \in \mathcal{C}^*} c_S = opt.$$ Logo, p/ cobert $\mathcal{C}$ retorn p/ algoritmo:

$$c(\mathcal{C}) = \sum_{S \in \mathcal{C}} c_S = \sum_{S \in \mathcal{C}} \sum_{u \in S} y_u \leq \sum_{S \in \mathcal{S}} \sum_{u \in S} y_u = \sum_{u \in U} f_u y_u \leq f \sum_{u \in U} y_u \leq f \cdot opt(U, \mathcal{S}, c)$$

Conclusão: Set Cover e Vertex Cover

Set Cover sem custos

- f-aproximação: Algoritmo MinCC-Guloso

Set Cover com custos

- $(1 + \ln n)$-aproximação: Algoritmo MinCC-Chvátal.
- f-aproximação: Algoritmo MinCC-Hochbaum (método primal)
- f-aproximação: Algoritmo MinCC-Hochbaum-dual (método dual)
- f-aproximação: Algoritmo MinCC-primal-dual (método primal-dual)

Vertex Cover sem custos

- 2-aproximação: Algoritmo MinCV-Guloso

Vertex Cover com custos

- 2-aproximação: Algoritmo MinCV-Hochbaum (método primal)
- 2-aproximação: Algoritmo MinCV-Hochbaum-dual (método dual)
- 2-aproximação: Algoritmo MinCV-primal-dual (método primal-dual)