Algoritmos de Aproximação

Segundo Semestre de 2012

Localização de facilidades

Problema: Dados um conjunto F de facilidades, um conjunto C de clientes, um custo f_i para cada i em F, um custo $c_{ij} \geq 0$ para cada i em F e cada j em C, encontrar um conjunto $F' \subseteq F$ que minimize

$$\sum_{i \in F'} f_i + \sum_{j \in C} \min\{c_{ij} \ : \ i \in F'\}.$$

Localização de facilidades

Problema: Dados um conjunto F de facilidades, um conjunto C de clientes, um custo f_i para cada i em F, um custo $c_{ij} \geq 0$ para cada i em F e cada j em C, encontrar um conjunto $F' \subseteq F$ que minimize

$$\sum_{i \in F'} f_i + \sum_{j \in C} \min\{c_{ij} \ : \ i \in F'\}.$$

Tão difícil de aproximar quanto o SET COVER.

Localização de facilidades

Problema: Dados um conjunto F de facilidades, um conjunto C de clientes, um custo f_i para cada i em F, um custo $c_{ij} \geq 0$ para cada i em F e cada j em C, encontrar um conjunto $F' \subseteq F$ que minimize

$$\sum_{i \in F'} f_i + \sum_{j \in C} \min\{c_{ij} \ : \ i \in F'\}.$$

Tão difícil de aproximar quanto o SET COVER.

Versão métrica: $c_{ij} \leq c_{i\ell} + c_{k\ell} + c_{kj}$ para todo i, j, k, ℓ.

Localização de facilidades

Problema: Dados um conjunto F de facilidades, um conjunto C de clientes, um custo f_i para cada i em F, um custo $c_{ij} \geq 0$ para cada i em F e cada j em C, encontrar um conjunto $F' \subseteq F$ que minimize

$$\sum_{i \in F'} f_i + \sum_{j \in C} \min\{c_{ij} \ : \ i \in F'\}.$$

Tão difícil de aproximar quanto o SET COVER.

Versão métrica: $c_{ij} \leq c_{i\ell} + c_{k\ell} + c_{kj}$ para todo i, j, k, ℓ.

Vimos uma 4-aproximação para a versão métrica.

Formulação inteira

Problema: Dados um conjunto F de facilidades, um conjunto C de clientes, um custo f_i para cada i em F, um custo $c_{ij} \geq 0$ para cada i em F e cada j em C, encontrar $F' \subseteq F$ que minimize $\sum_{i \in F'} f_i + \sum_{j \in C} \min\{c_{ij} \ : \ i \in F'\}$.

Formulação inteira

Problema: Dados um conjunto F de facilidades, um conjunto C de clientes, um custo f_i para cada i em F, um custo $c_{ij} \geq 0$ para cada i em F e cada j em C, encontrar $F' \subseteq F$ que minimize $\sum_{i \in F'} f_i + \sum_{j \in C} \min\{c_{ij} \ : \ i \in F'\}$.

Variáveis:

- y_i: indica se a facilidade i é aberta ou não.
- x_{ij}: indica se o cliente j se conecta à facilidade i.

Formulação inteira

Problema: Dados um conjunto F de facilidades, um conjunto C de clientes, um custo f_i para cada i em F, um custo $c_{ij} \geq 0$ para cada i em F e cada j em C, encontrar $F' \subseteq F$ que minimize $\sum_{i \in F'} f_i + \sum_{j \in C} \min\{c_{ij} \ : \ i \in F'\}$.

Variáveis:

- y_i: indica se a facilidade i é aberta ou não.
- x_{ij}: indica se o cliente j se conecta à facilidade i.

Encontrar x e y que

minimizem $\sum_{i \in F} f_i \, y_i + \sum_{i \in F, \ j \in C} c_{ij} \, x_{ij}$

sujeito a

$\sum_{i \in F} x_{ij} = 1$	para todo j em C
$x_{ij} \leq y_i$	para todo i em F e j em C
$x_{ij} \in \{0, 1\}$	para todo i em F e j em C
$y_i \in \{0, 1\}$	para todo i em F.

Relaxação linear

- y_i: indica se a facilidade i é aberta ou não.
- x_{ij}: indica se o cliente j se conecta à facilidade i.

minimizar $\sum_{i \in F} f_i \, y_i + \sum_{i \in F,\ j \in C} c_{ij} \, x_{ij}$

sujeito a $\sum_{i \in F} x_{ij} = 1$ para todo j em C

$x_{ij} \leq y_i$ para todo i em F e j em C

$x_{ij} \geq 0$ para todo i em F e j em C

$y_i \geq 0$ para todo i em F.

Relaxação linear

- y_i: indica se a facilidade i é aberta ou não.
- x_{ij}: indica se o cliente j se conecta à facilidade i.

minimizar $\sum_{i \in F} f_i \, y_i + \sum_{i \in F,\ j \in C} c_{ij} \, x_{ij}$

sujeito a $\sum_{i \in F} x_{ij} = 1$ para todo j em C

$x_{ij} \leq y_i$ para todo i em F e j em C

$x_{ij} \geq 0$ para todo i em F e j em C

$y_i \geq 0$ para todo i em F.

Dual:

maximizar $\sum_{j \in C} v_j$

sujeito a $\sum_{j \in C} w_{ij} \leq f_i$ para todo i em F

$v_j - w_{ij} \leq c_{ij}$ para todo i em F e j em C

$w_{ij} \geq 0$ para todo i em F e j em C.

Relaxação linear

Primal:

minimizar $\sum_{i \in F} f_i \, y_i + \sum_{i \in F,\ j \in C} c_{ij} \, x_{ij}$

sujeito a $\sum_{i \in F} x_{ij} = 1$ para todo j em C

$x_{ij} \leq y_i$ para todo i em F e j em C

$x_{ij} \geq 0$ para todo i em F e j em C

$y_i \geq 0$ para todo i em F.

Relaxação linear

Primal:

minimizar $\sum_{i \in F} f_i \, y_i + \sum_{i \in F,\ j \in C} c_{ij} \, x_{ij}$

sujeito a $\sum_{i \in F} x_{ij} = 1$ para todo j em C

$x_{ij} \leq y_i$ para todo i em F e j em C

$x_{ij} \geq 0$ para todo i em F e j em C

$y_i \geq 0$ para todo i em F.

Dual:

maximizar $\sum_{j \in C} v_j$

sujeito a $\sum_{j \in C} w_{ij} \leq f_i$ para todo i em F

$v_j - w_{ij} \leq c_{ij}$ para todo i em F e j em C

$w_{ij} \geq 0$ para todo i em F e j em C.

Relaxação linear

Primal:

minimizar $\sum_{i \in F} f_i \, y_i + \sum_{i \in F,\ j \in C} c_{ij} \, x_{ij}$

sujeito a $\sum_{i \in F} x_{ij} = 1$ para todo j em C

$x_{ij} \leq y_i$ para todo i em F e j em C

$x_{ij} \geq 0$ para todo i em F e j em C

$y_i \geq 0$ para todo i em F.

Dual:

maximizar $\sum_{j \in C} v_j$

sujeito a $\sum_{j \in C} w_{ij} \leq f_i$ para todo i em F

$v_j - w_{ij} \leq c_{ij}$ para todo i em F e j em C

$w_{ij} \geq 0$ para todo i em F e j em C.

- v_j: indica quanto o cliente j paga para se conectar.
- w_{ij}: indica quanto j pagaria para a facilidade i abrir.

Relaxação linear

(P) minimizar $\sum_{i \in F} f_i \, y_i + \sum_{i \in F,\, j \in C} c_{ij} \, x_{ij}$

sujeito a $\sum_{i \in F} x_{ij} = 1$ para todo j em C

$x_{ij} \leq y_i$ para todo i em F e j em C

$x_{ij} \geq 0$ para todo i em F e j em C

$y_i \geq 0$ para todo i em F.

(D) maximizar $\sum_{j \in C} v_j$

sujeito a $\sum_{j \in C} w_{ij} \leq f_i$ para todo i em F

$v_j - w_{ij} \leq c_{ij}$ para todo i em F e j em C

$w_{ij} \geq 0$ para todo i em F e j em C.

Lema: Sejam (x^*, y^*) e e (v^*, w^*) soluções ótimas do primal e do dual. Se $x^*_{ij} > 0$ então $c_{ij} \leq v^*_j$.

Como usar as soluções de (P) e (D)

Facilidade i é vizinha do cliente j se $x^*_{ij} > 0$.

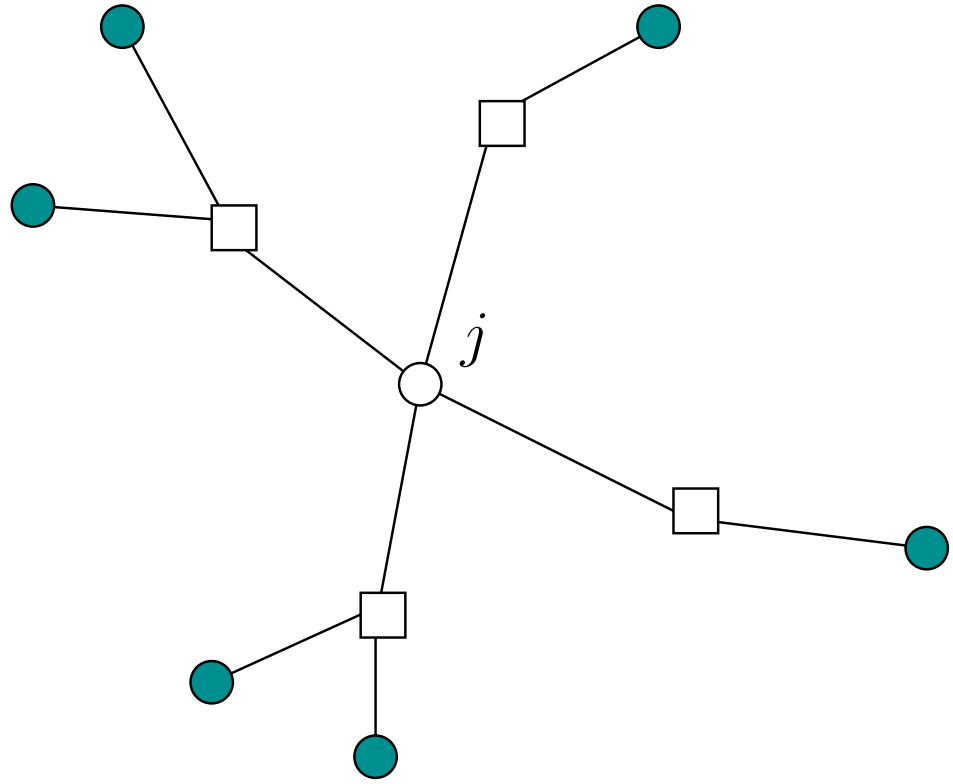
Seja $N(j) = \{i \in F \ : \ i$ é vizinha de $j\}$.

Como usar as soluções de (P) e (D)

Facilidade i é vizinha do cliente j se $x^*_{ij} > 0$.

Seja $N(j) = \{i \in F \ : \ i \text{ é vizinha de } j\}$.

Vizinhança aumentada:
$N^2(j)$: clientes vizinhos de alguma facilidade de $N(j)$



Algoritmo

Arred-Det (F, C, f, c)

1 sejam (x^*, y^*) e (v^*, w^*) soluções ótimas de (P) e (D)
2 $D \leftarrow C$ $k \leftarrow 0$
3 **enquanto** $D \neq \emptyset$ **faça**
4 $k \leftarrow k + 1$
5 $j_k \leftarrow \arg\min\{v_j^* : j \in D\}$
6 $i_k \leftarrow \arg\min\{f_i : i \in N(j_k)\}$
7 abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k
8 $D \leftarrow D \setminus (\{j_k\} \cup N^2(j_k))$

Algoritmo

Arred-Det (F, C, f, c)

1 sejam (x^*, y^*) e (v^*, w^*) soluções ótimas de (P) e (D)
2 $D \leftarrow C$ $k \leftarrow 0$
3 **enquanto** $D \neq \emptyset$ **faça**
4 $k \leftarrow k + 1$
5 $j_k \leftarrow \arg\min\{v_j^* : j \in D\}$
6 $i_k \leftarrow \arg\min\{f_i : i \in N(j_k)\}$
7 abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k
8 $D \leftarrow D \setminus (\{j_k\} \cup N^2(j_k))$

Teorema: Arred-Det é uma 4-aproximação.

Algoritmo

Arred-Det (F, C, f, c)

1 sejam (x^*, y^*) e (v^*, w^*) soluções ótimas de (P) e (D)
2 $D \leftarrow C$ $\quad$ $k \leftarrow 0$
3 **enquanto** $D \neq \emptyset$ **faça**
4 $\quad$ $k \leftarrow k + 1$
5 $\quad$ $j_k \leftarrow \arg\min\{v_j^* : j \in D\}$
6 $\quad$ $i_k \leftarrow \arg\min\{f_i : i \in N(j_k)\}$
7 $\quad$ abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k
8 $\quad$ $D \leftarrow D \setminus (\{j_k\} \cup N^2(j_k))$

Teorema: Arred-Det é uma 4-aproximação.

Prova: Como os $N(j_k)$ são 2-a-2 disjuntos, segue que $\sum_k f_{i_k} \leq \sum_{i \in F} f_i \, y_i^* \leq \mathrm{OPT}$.

Algoritmo

Arred-Det (F, C, f, c)

1 sejam (x^*, y^*) e (v^*, w^*) soluções ótimas de (P) e (D)
2 $D \leftarrow C$ $\quad k \leftarrow 0$
3 **enquanto** $D \neq \emptyset$ **faça**
4 $\quad k \leftarrow k + 1$
5 $\quad j_k \leftarrow \arg\min\{v_j^* : j \in D\}$
6 $\quad i_k \leftarrow \arg\min\{f_i : i \in N(j_k)\}$
7 $\quad$ abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k
8 $\quad D \leftarrow D \setminus (\{j_k\} \cup N^2(j_k))$

Teorema: Arred-Det é uma 4-aproximação.

Prova: Como os $N(j_k)$ são 2-a-2 disjuntos, segue que $\sum_k f_{i_k} \leq \sum_{i \in F} f_i \, y_i^* \leq \mathrm{OPT}$. E os custos de conexão?

E os custos de conexão?

5 $j_k \leftarrow \arg\min\{v_j^* : j \in D\}$

6 $i_k \leftarrow \arg\min\{f_i : i \in N(j_k)\}$

7 abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k

Lembre-se que $c_{ij} \leq v_j^*$ sempre que $i \in N(j)$.

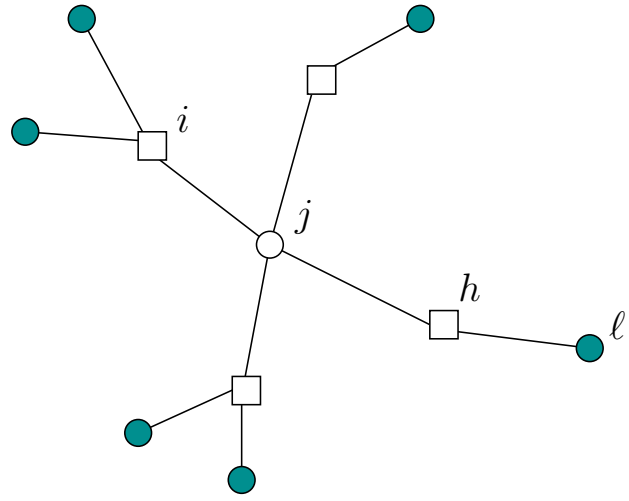
E os custos de conexão?

5 $j_k \leftarrow \arg\min\{v_j^* : j \in D\}$

6 $i_k \leftarrow \arg\min\{f_i : i \in N(j_k)\}$

7 abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k

Lembre-se que $c_{ij} \leq v_j^*$ sempre que $i \in N(j)$.



$$c_{i\ell} \leq c_{ij} + c_{hj} + c_{h\ell} \leq v_j^* + v_j^* + v_\ell^* \leq 3\, v_\ell^*.$$

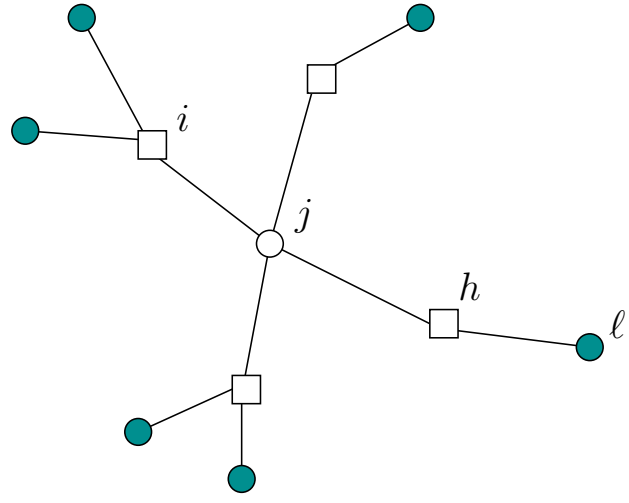
E os custos de conexão?

5 $j_k \leftarrow \arg\min\{v_j^* : j \in D\}$

6 $i_k \leftarrow \arg\min\{f_i : i \in N(j_k)\}$

7 abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k

Lembre-se que $c_{ij} \leq v_j^*$ sempre que $i \in N(j)$.



$$c_{i\ell} \leq c_{ij} + c_{hj} + c_{h\ell} \leq v_j^* + v_j^* + v_\ell^* \leq 3\, v_\ell^*.$$

Portanto o custo de conexão é $\leq 3 \sum_{\ell \in C} v_\ell^* \leq 3\,\mathrm{OPT}$.

Algoritmo

Arred-Det (F, C, f, c)

1 sejam (x^*, y^*) e (v^*, w^*) soluções ótimas de (P) e (D)
2 $D \leftarrow C$ $\quad k \leftarrow 0$
3 **enquanto** $D \neq \emptyset$ **faça**
4 $\quad k \leftarrow k + 1$
5 $\quad j_k \leftarrow \arg\min\{v_j^* : j \in D\}$
6 $\quad i_k \leftarrow \arg\min\{f_i : i \in N(j_k)\}$
7 $\quad$ abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k
8 $\quad D \leftarrow D \setminus (\{j_k\} \cup N^2(j_k))$

Teorema: Arred-Det é uma 4-aproximação.

Prova: Primeiro note que os $N(j_k)$ são 2-a-2 disjuntos.
Logo $\sum_k f_{i_k} \leq \sum_{i \in F} f_i \, y_i^* \leq \mathrm{OPT}$.
E os custos de conexão somados são $\leq 3\,\mathrm{OPT}$. ∎

Aleatoriedade

Arred-Aleat (F, C, f, c)

1 sejam (x^*, y^*) e (v^*, w^*) soluções ótimas de (P) e (D)
2 **para cada** $j \in C$ **faça** $C_j^* \leftarrow \sum_{i \in F} c_{ij} x_{ij}^*$
3 $D \leftarrow C \qquad k \leftarrow 0$
4 **enquanto** $D \neq \emptyset$ **faça**
5 $k \leftarrow k + 1$
6 $j_k \leftarrow \arg\min\{v_j^* + C_j^* : j \in D\}$
7 escolha $i_k \in N(j_k)$ com probabilidade x_{ij_k}
8 abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k
9 $D \leftarrow D \setminus (\{j_k\} \cup N^2(j_k))$

Aleatoriedade

Arred-Aleat (F, C, f, c)

1 sejam (x^*, y^*) e (v^*, w^*) soluções ótimas de (P) e (D)
2 **para cada** $j \in C$ **faça** $C_j^* \leftarrow \sum_{i \in F} c_{ij} x_{ij}^*$
3 $D \leftarrow C \qquad k \leftarrow 0$
4 **enquanto** $D \neq \emptyset$ **faça**
5 $k \leftarrow k + 1$
6 $j_k \leftarrow \arg\min\{v_j^* + C_j^* : j \in D\}$
7 escolha $i_k \in N(j_k)$ com probabilidade x_{ij_k}
8 abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k
9 $D \leftarrow D \setminus (\{j_k\} \cup N^2(j_k))$

Note que $\sum_{i \in N(j)} x_{ij}^* = 1$.

Aleatoriedade

Arred-Aleat (F, C, f, c)

1 sejam (x^*, y^*) e (v^*, w^*) soluções ótimas de (P) e (D)
2 **para cada** $j \in C$ **faça** $C_j^* \leftarrow \sum_{i \in F} c_{ij} x_{ij}^*$
3 $D \leftarrow C \qquad k \leftarrow 0$
4 **enquanto** $D \neq \emptyset$ **faça**
5 $k \leftarrow k + 1$
6 $j_k \leftarrow \arg\min\{v_j^* + C_j^* : j \in D\}$
7 escolha $i_k \in N(j_k)$ com probabilidade x_{ij_k}
8 abra i_k e conecte $\{j_k\} \cup (N^2(j_k) \cap D)$ a i_k
9 $D \leftarrow D \setminus (\{j_k\} \cup N^2(j_k))$

Note que $\sum_{i \in N(j)} x_{ij}^* = 1$.

Vamos mostrar que este algoritmo é uma 3-aproximação.

Análise do algoritmo probabilístico

Custo das facilidades abertas

O custo esperado da facilidade aberta na iteração k é

$$\sum_{i \in N(j_k)} f_i \, x^*_{ij_k} \leq \sum_{i \in N(j_k)} f_i \, y^*_i,$$

porque $x^*_{ij} \leq y^*_i$ para todo ij.

Análise do algoritmo probabilístico

Custo das facilidades abertas

O custo esperado da facilidade aberta na iteração k é

$$\sum_{i \in N(j_k)} f_i \, x^*_{ij_k} \leq \sum_{i \in N(j_k)} f_i \, y^*_i,$$

porque $x^*_{ij} \leq y^*_i$ para todo ij.

As vizinhanças $N(j_k)$ são 2-a-2 disjuntas, logo

$$\sum_k \sum_{i \in N(j_k)} f_i \, x^*_{ij_k} \leq \sum_{i \in F} f_i \, y^*_i.$$

Análise do algoritmo probabilístico

Custos de conexão

Considere a iteração k e sejam $j = j_k$ e $i = i_k$.

O custo esperado de conectar j a i é $\sum_{i \in N(j_k)} c_{ij}\, x^*_{ij} \;=\; C^*_j$.

Análise do algoritmo probabilístico

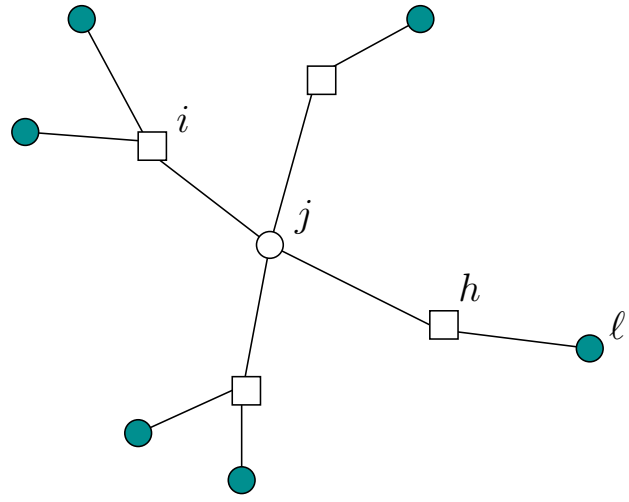
Custos de conexão

Considere a iteração k e sejam $j = j_k$ e $i = i_k$.

O custo esperado de conectar j a i é $\sum_{i \in N(j_k)} c_{ij}\, x^*_{ij} \;=\; C^*_j$.

O custo esperado de conectar um cliente $\ell \in N^2(j)$ a i é

$$c_{h\ell} + c_{hj} + \sum_{i \in N(j)} c_{ij}\, x^*_{ij} \;=\; c_{h\ell} + c_{hj} + C^*_j.$$



Análise do algoritmo probabilístico

Custos de conexão

Considere a iteração k e sejam $j = j_k$ e $i = i_k$.

O custo esperado de conectar j a i é $\sum_{i \in N(j_k)} c_{ij}\, x^*_{ij} \;=\; C^*_j$.

O custo esperado de conectar um cliente $\ell \in N^2(j)$ a i é

$$c_{h\ell} + c_{hj} + \sum_{i \in N(j)} c_{ij}\, x^*_{ij} \;=\; c_{h\ell} + c_{hj} + C^*_j.$$

Pelo Lema, $c_{h\ell} \leq v^*_\ell$ and $c_{hj} \leq v^*_j$, logo

$$c_{h\ell} + c_{hj} + C^*_j \;\leq\; v^*_\ell + v^*_j + C^*_j$$

Análise do algoritmo probabilístico

Custos de conexão

Considere a iteração k e sejam $j = j_k$ e $i = i_k$.

O custo esperado de conectar j a i é $\sum_{i \in N(j_k)} c_{ij}\, x^*_{ij} \;=\; C^*_j$.

O custo esperado de conectar um cliente $\ell \in N^2(j)$ a i é

$$c_{h\ell} + c_{hj} + \sum_{i \in N(j)} c_{ij}\, x^*_{ij} \;=\; c_{h\ell} + c_{hj} + C^*_j.$$

Pelo Lema, $c_{h\ell} \leq v^*_\ell$ and $c_{hj} \leq v^*_j$, logo

$$c_{h\ell} + c_{hj} + C^*_j \;\leq\; v^*_\ell + v^*_j + C^*_j \;\leq\; v^*_\ell + v^*_\ell + C^*_\ell.$$

Análise do algoritmo probabilístico

Custos de conexão

Considere a iteração k e sejam $j = j_k$ e $i = i_k$.

O custo esperado de conectar j a i é $\sum_{i \in N(j_k)} c_{ij}\, x^*_{ij} \;=\; C^*_j$.

O custo esperado de conectar um cliente $\ell \in N^2(j)$ a i é

$$c_{h\ell} + c_{hj} + \sum_{i \in N(j)} c_{ij}\, x^*_{ij} \;=\; c_{h\ell} + c_{hj} + C^*_j.$$

Pelo Lema, $c_{h\ell} \leq v^*_\ell$ and $c_{hj} \leq v^*_j$, logo

$$c_{h\ell} + c_{hj} + C^*_j \;\leq\; v^*_\ell + v^*_j + C^*_j \;\leq\; 2\, v^*_\ell + C^*_\ell.$$

Análise do algoritmo probabilístico

Custo das facilidades abertas:

$$\sum_{i \in F} f_i \, y_i^*$$

Custos de conexão

$$\sum_{\ell \in C} (2 \, v_\ell^* + C_\ell^*)$$

Análise do algoritmo probabilístico

Custo das facilidades abertas:

$$\sum_{i \in F} f_i \, y_i^*$$

Custos de conexão

$$\sum_{\ell \in C} (2 \, v_\ell^* + C_\ell^*)$$

Soma dos custos:

$$\sum_{i \in F} f_i \, y_i^* + \sum_{\ell \in C} (2 \, v_\ell^* + C_\ell^*) \quad = \quad \sum_{i \in F} f_i \, y_i^* + \sum_{j \in C, i \in F} c_{ij} x_{ij}^* + 2 \sum_{\ell \in C} v_\ell^*$$

Análise do algoritmo probabilístico

Custo das facilidades abertas:

$$\sum_{i \in F} f_i \, y_i^*$$

Custos de conexão

$$\sum_{\ell \in C} (2 \, v_\ell^* + C_\ell^*)$$

Soma dos custos:

$$\begin{aligned} \sum_{i \in F} f_i \, y_i^* + \sum_{\ell \in C} (2 \, v_\ell^* + C_\ell^*) &= \sum_{i \in F} f_i \, y_i^* + \sum_{j \in C, i \in F} c_{ij} x_{ij}^* + 2 \sum_{\ell \in C} v_\ell^* \\ &\leq z_P^* + 2 \, z_D^* \leq 3 \, \mathrm{OPT}. \end{aligned}$$

Escalonamento de uma máquina

Dados: n tarefas $([n] = \{1, \ldots, n\})$
duração $d[i]$ da tarefa i $(i = 1, \ldots, n)$
momento de chegada $r[i]$ da tarefa i $(i = 1, \ldots, n)$

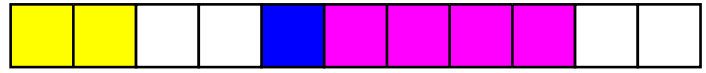
um escalonamento é uma ordenação de $[n]$.

Escalonamento de uma máquina

Dados: n tarefas $([n] = \{1, \ldots, n\})$
duração $d[i]$ da tarefa i $(i = 1, \ldots, n)$
momento de chegada $r[i]$ da tarefa i $(i = 1, \ldots, n)$

um escalonamento é uma ordenação de $[n]$.

Exemplo: $d_1 = 2$, $r_1 = 0$, $d_2 = 1$, $r_2 = 4$, $d_3 = 4$, $r_3 = 1$.

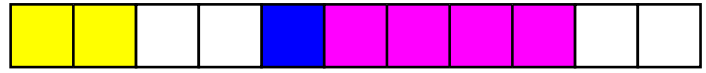


Escalonamento de uma máquina

Dados: n tarefas $([n] = \{1, \ldots, n\})$
duração $d[i]$ da tarefa i $(i = 1, \ldots, n)$
momento de chegada $r[i]$ da tarefa i $(i = 1, \ldots, n)$

um escalonamento é uma ordenação de $[n]$.

Exemplo: $d_1 = 2$, $r_1 = 0$, $d_2 = 1$, $r_2 = 4$, $d_3 = 4$, $r_3 = 1$.



$C_1 = 2$, $C_2 = 5$, $C_3 = 9$.

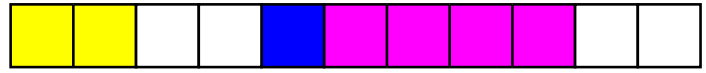
Encontrar escalonamento com
soma dos tempos de conclusão mínima.

Escalonamento de uma máquina

Dados: n tarefas $([n] = \{1, \ldots, n\})$
duração $d[i]$ da tarefa i $(i = 1, \ldots, n)$
momento de chegada $r[i]$ da tarefa i $(i = 1, \ldots, n)$

um escalonamento é uma ordenação de $[n]$.

Exemplo: $d_1 = 2$, $r_1 = 0$, $d_2 = 1$, $r_2 = 4$, $d_3 = 4$, $r_3 = 1$.



$C_1 = 2$, $C_2 = 5$, $C_3 = 9$.

Encontrar escalonamento com
soma dos tempos de conclusão mínima.

Mesmo que minimizar a média dos tempos de conclusão.

Versão com pesos

Dados: n tarefas $([n] = \{1, \ldots, n\})$
duração $d[i]$ da tarefa i $(i = 1, \ldots, n)$
momento de chegada $r[i]$ da tarefa i $(i = 1, \ldots, n)$
peso $w[i]$ da tarefa i $(i = 1, \ldots, n)$

Versão com pesos

Dados: n tarefas $([n] = \{1, \ldots, n\})$
duração $d[i]$ da tarefa i $(i = 1, \ldots, n)$
momento de chegada $r[i]$ da tarefa i $(i = 1, \ldots, n)$
peso $w[i]$ da tarefa i $(i = 1, \ldots, n)$

Um escalonamento é uma ordenação de $[n]$.

Encontrar escalonamento com
soma ponderada dos tempos de conclusão mínima.

Versão com pesos

Dados: n tarefas $([n] = \{1, \ldots, n\})$
duração $d[i]$ da tarefa i $(i = 1, \ldots, n)$
momento de chegada $r[i]$ da tarefa i $(i = 1, \ldots, n)$
peso $w[i]$ da tarefa i $(i = 1, \ldots, n)$

Um escalonamento é uma ordenação de $[n]$.

Encontrar escalonamento com
soma ponderada dos tempos de conclusão mínima.

Versão com interrupções com pesos é NP-difícil...

Versão com pesos

Dados: n tarefas $([n] = \{1, \ldots, n\})$
duração $d[i]$ da tarefa i $(i = 1, \ldots, n)$
momento de chegada $r[i]$ da tarefa i $(i = 1, \ldots, n)$
peso $w[i]$ da tarefa i $(i = 1, \ldots, n)$

Um escalonamento é uma ordenação de $[n]$.

Encontrar escalonamento com
soma ponderada dos tempos de conclusão mínima.

Versão com interrupções com pesos é NP-difícil...

Vimos uma 3-aproximação para esse problema.

Relaxação linear

Encontrar $C_1, \ldots, C_n$ que minimizem $\sum_{j=1}^{n} w_j C_j$, sujeitos a

Relaxação linear

Encontrar $C_1, \ldots, C_n$ que minimizem $\sum_{j=1}^{n} w_j C_j$, sujeitos a

$$C_j \geq r_j + d_j \qquad \text{para todo } j$$

Relaxação linear

Encontrar $C_1, \ldots, C_n$ que minimizem $\sum_{j=1}^{n} w_j C_j$, sujeitos a

$$C_j \geq r_j + d_j \qquad \text{para todo } j$$

e... vamos deduzir uma segunda restrição, menos óbvia.

Seja $S \subseteq [n]$ e considere a soma $\sum_{j \in S} d_j C_j$.

Relaxação linear

Encontrar $C_1, \ldots, C_n$ que minimizem $\sum_{j=1}^{n} w_j C_j$, sujeitos a

$$C_j \geq r_j + d_j \qquad \text{para todo } j$$

e... vamos deduzir uma segunda restrição, menos óbvia.

Seja $S \subseteq [n]$ e considere a soma $\sum_{j \in S} d_j C_j$.

Essa soma é o menor possível se $r_j = 0$ para todo $j \in S$ e as tarefas de S vem antes de todas as demais.

Relaxação linear

Encontrar $C_1, \ldots, C_n$ que minimizem $\sum_{j=1}^{n} w_j\, C_j$, sujeitos a

$$C_j \geq r_j + d_j \qquad \text{para todo } j$$

e... vamos deduzir uma segunda restrição, menos óbvia.

Seja $S \subseteq [n]$ e considere a soma $\sum_{j \in S} d_j\, C_j$.

Essa soma é o menor possível se $r_j = 0$ para todo $j \in S$ e as tarefas de S vem antes de todas as demais.

Neste caso,

$$\sum_{j \in S} d_j\, C_j = \sum_{j,k \in S, j \leq k} d_j\, d_k = \frac{1}{2} \left(\sum_{j \in S} d_j\right)^2 + \frac{1}{2} \sum_{j \in S} d_j^2 \geq \frac{1}{2} \left(\sum_{j \in S} d_j\right)^2.$$

Relaxação linear

Seja $d(S) = \sum_{j \in S} d_j$.

O PL fica: encontrar $C_1, \ldots, C_n$ que

minimizem $\sum_{j=1}^{n} w_j C_j$,
sujeitos a

$$C_j \geq r_j + d_j \qquad \text{para todo } j$$

$$\sum_{j \in S} d_j C_j \geq \tfrac{1}{2} d(S)^2 \qquad \text{para todo } S \subseteq [n].$$

Algoritmo

Seja $C_1^*, \ldots, C_n^*$ uma solução ótima do PL.

Reordene as tarefas de modo que $C_1^* \leq \cdots \leq C_n^*$.

Algoritmo

Seja $C_1^*, \ldots, C_n^*$ uma solução ótima do PL.

Reordene as tarefas de modo que $C_1^* \leq \cdots \leq C_n^*$.

Obtenha um escalonamento, escalonando as tarefas na ordem $1, \ldots, n$.

Algoritmo

Seja $C_1^*, \ldots, C_n^*$ uma solução ótima do PL.

Reordene as tarefas de modo que $C_1^* \leq \cdots \leq C_n^*$.

Obtenha um escalonamento, escalonando as tarefas na ordem $1, \ldots, n$.

Chame de $C_1^N, \ldots, C_n^N$ os tempos de conclusão resultantes.

Algoritmo

Seja $C_1^*, \ldots, C_n^*$ uma solução ótima do PL.

Reordene as tarefas de modo que $C_1^* \leq \cdots \leq C_n^*$.

Obtenha um escalonamento, escalonando as tarefas na ordem $1, \ldots, n$.

Chame de $C_1^N, \ldots, C_n^N$ os tempos de conclusão resultantes.

Teorema: O algoritmo acima é uma 3-aproximação.

Prova: Mostramos que $C_j^N \leq 3\, C_j^*$ para todo j, logo

$$\sum_{j=1}^{n} w_j\, C_j^N \leq 3 \sum_{j=1}^{n} w_j\, C_j^* \leq 3\, \mathrm{OPT}.$$

Uma outra relaxação linear

Seja $T = \max_j r[j] + \sum_{j=1}^{n} d[j]$.

Nenhuma tarefa pode terminar depois do instante T.

Uma outra relaxação linear

Seja $T = \max_j r[j] + \sum_{j=1}^{n} d[j]$.

Nenhuma tarefa pode terminar depois do instante T.

Variáveis:

$y_{jt} = 1$ sse a tarefa j está executando no instante t.

Uma outra relaxação linear

Seja $T = \max_j r[j] + \sum_{j=1}^{n} d[j]$.

Nenhuma tarefa pode terminar depois do instante T.

Variáveis:
$y_{jt} = 1$ sse a tarefa j está executando no instante t.

No máximo uma tarefa executa a cada instante t:

$$\sum_{j=1}^{n} y_{jt} \leq 1.$$

Uma outra relaxação linear

Seja $T = \max_j r[j] + \sum_{j=1}^{n} d[j]$.

Nenhuma tarefa pode terminar depois do instante T.

Variáveis:
$y_{jt} = 1$ sse a tarefa j está executando no instante t.

No máximo uma tarefa executa a cada instante t:

$$\sum_{j=1}^{n} y_{jt} \leq 1.$$

A tarefa j executa por $d[j]$ unidades de tempo:

$$\sum_{t=1}^{T} y_{jt} = d[j].$$

Uma outra relaxação linear

Seja $T = \max_j r[j] + \sum_{j=1}^{n} d[j]$.

$y_{jt} = 1$ sse a tarefa j está executando no instante t.

No máximo uma tarefa executa a cada instante:

$$\sum_{j=1}^{n} y_{jt} \leq 1.$$

A tarefa j executa por $d[j]$ unidades de tempo:

$$\sum_{t=1}^{T} y_{jt} = d[j].$$

Uma outra relaxação linear

Seja $T = \max_j r[j] + \sum_{j=1}^{n} d[j]$.

$y_{jt} = 1$ sse a tarefa j está executando no instante t.

No máximo uma tarefa executa a cada instante:

$$\sum_{j=1}^{n} y_{jt} \;\leq\; 1.$$

A tarefa j executa por $d[j]$ unidades de tempo:

$$\sum_{t=1}^{T} y_{jt} \;=\; d[j].$$

A tarefa j só pode executar depois que estiver disponível:

$$y_{jt} \;=\; 0 \quad \text{para } t = 1, \ldots, r[j].$$

Tempo de conclusão

Num escalonamento sem interrupção,
se a tarefa j termina no instante D, então

Tempo de conclusão

Num escalonamento sem interrupção,
se a tarefa j termina no instante D, então

$$y_{jt} = 1 \quad \text{para } t = D - d[j] + 1, \ldots, D,$$

caso contrário, $y_{jt} = 0$.

Tempo de conclusão

Num escalonamento sem interrupção,
se a tarefa j termina no instante D, então

$$y_{jt} = 1 \quad \text{para } t = D - d[j] + 1, \ldots, D,$$

caso contrário, $y_{jt} = 0$.

A média da metade $t - \frac{1}{2}$ de cada um destes instantes satisfaz

$$\frac{1}{d[j]} \sum_{t=D-d[j]+1}^{D} \left(t - \frac{1}{2}\right) = D - \frac{d[j]}{2}.$$

Tempo de conclusão

Num escalonamento sem interrupção,
se a tarefa j termina no instante D, então

$$y_{jt} = 1 \quad \text{para } t = D - d[j] + 1, \ldots, D,$$

caso contrário, $y_{jt} = 0$.

A média da metade $t - \frac{1}{2}$ de cada um destes instantes satisfaz

$$\frac{1}{d[j]} \sum_{t=D-d[j]+1}^{D} \left(t - \frac{1}{2}\right) = D - \frac{d[j]}{2}.$$

Disso deduzimos que

$$C_j = \frac{1}{d[j]} \sum_{t=1}^{T} y_{jt}\left(t - \frac{1}{2}\right) + \frac{d[j]}{2}.$$

Relaxação linear inteira

Encontrar matriz y e vetor C que

minimizem $\sum_{j=1}^{n} w_j C_j$

sujeito a

$$\sum_{j=1}^{n} y_{jt} \leq 1 \qquad \text{para } t = 1, \ldots, T$$

$$\sum_{t=1}^{T} y_{jt} = d[j] \qquad \text{para } j = 1, \ldots, n$$

$$C_j = \frac{1}{d[j]} \sum_{t=1}^{T} y_{jt}\left(t - \frac{1}{2}\right) + \frac{d[j]}{2} \qquad \text{para } j = 1, \ldots, n$$

$$y_{jt} \in \{0, 1\} \qquad \text{para } j = 1, \ldots, n, \; t = 1, \ldots, T.$$

Relaxação linear inteira

Encontrar matriz y e vetor C que

minimizem $\sum_{j=1}^{n} w_j C_j$

sujeito a

$$\sum_{j=1}^{n} y_{jt} \leq 1 \qquad \text{para } t = 1, \ldots, T$$

$$\sum_{t=1}^{T} y_{jt} = d[j] \qquad \text{para } j = 1, \ldots, n$$

$$C_j = \frac{1}{d[j]} \sum_{t=1}^{T} y_{jt}\left(t - \frac{1}{2}\right) + \frac{d[j]}{2} \qquad \text{para } j = 1, \ldots, n$$

$$y_{jt} \in \{0, 1\} \qquad \text{para } j = 1, \ldots, n, \; t = 1, \ldots, T.$$

Todo y viável para esse programa corresponde a um escalonamento com interrupções.

Relaxação linear inteira

Encontrar matriz y e vetor C que

minimizem $\sum_{j=1}^{n} w_j C_j$

sujeito a

$$\sum_{j=1}^{n} y_{jt} \leq 1 \qquad \text{para } t = 1, \ldots, T$$

$$\sum_{t=1}^{T} y_{jt} = d[j] \qquad \text{para } j = 1, \ldots, n$$

$$C_j = \frac{1}{d[j]} \sum_{t=1}^{T} y_{jt}\left(t - \frac{1}{2}\right) + \frac{d[j]}{2} \qquad \text{para } j = 1, \ldots, n$$

$$y_{jt} \in \{0, 1\} \qquad \text{para } j = 1, \ldots, n, \; t = 1, \ldots, T.$$

Todo y viável para esse programa corresponde a um escalonamento com interrupções.

É uma relaxação pois...

Todo escalonamento sem interrupções corresponde a uma solução viável e...

Relaxação linear inteira

Encontrar matriz y e vetor C que

minimizem $\sum_{j=1}^{n} w_j C_j$

sujeito a

$$\sum_{j=1}^{n} y_{jt} \leq 1 \qquad \text{para } t = 1, \ldots, T$$

$$\sum_{t=1}^{T} y_{jt} = d[j] \qquad \text{para } j = 1, \ldots, n$$

$$C_j = \frac{1}{d[j]} \sum_{t=1}^{T} y_{jt}\left(t - \frac{1}{2}\right) + \frac{d[j]}{2} \qquad \text{para } j = 1, \ldots, n$$

$$y_{jt} \in \{0, 1\} \qquad \text{para } j = 1, \ldots, n, \; t = 1, \ldots, T.$$

Todo y viável para esse programa corresponde a um escalonamento com interrupções.

É uma relaxação pois...

Todo escalonamento sem interrupções corresponde a uma solução viável e...

o valor desta solução no programa é no problema é exatamente o mesmo.