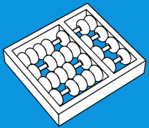
Problemas NP-completos e programação linear inteira

MC558 - Projeto e Análise de Algoritmos II

01/24

13



Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP

"The art of formulating linear and integer linear programs is, well, an art: It is hard to teach, and even harder to learn."

Gerald G. Brown and Robert F. Dell.

Reduções

Satisfatibilidade

Queremos provar que o seguinte problema é NP-completo:

Satisfatibilidade

Queremos provar que o seguinte problema é NP-completo:

Problema (Satisfatibilidade)

$$SAT = \{\langle f \rangle : f \text{ é uma fórmula booleana satisfazível}\}$$

Satisfatibilidade

Queremos provar que o seguinte problema é NP-completo:

Problema (Satisfatibilidade)

$$SAT = \{\langle f \rangle : f \text{ é uma fórmula booleana satisfazível}\}$$

Para isso, consideraremos um caso particular: quando a fórmula está escrita em formal normal conjuntiva: SAT_{FNC}:

Satisfatibilidade

Queremos provar que o seguinte problema é NP-completo:

Problema (Satisfatibilidade)

$$SAT = \{\langle f \rangle : f \text{ é uma fórmula booleana satisfazível}\}$$

Para isso, consideraremos um caso particular: quando a fórmula está escrita em formal normal conjuntiva: SAT_{FNC}:

- A fórmula é composta por conjunção de cláusulas (cláusulas relacionadas pelo operador $\wedge$).

Satisfatibilidade

Queremos provar que o seguinte problema é NP-completo:

Problema (Satisfatibilidade)

$$SAT = \{\langle f \rangle : f \text{ é uma fórmula booleana satisfazível}\}$$

Para isso, consideraremos um caso particular: quando a fórmula está escrita em formal normal conjuntiva: SAT_{FNC}:

- A fórmula é composta por conjunção de cláusulas (cláusulas relacionadas pelo operador $\wedge$).
- Cada cláusula é composta por disjunção de literais (literais relacionados pelo operador $\vee$).

Satisfatibilidade

Queremos provar que o seguinte problema é NP-completo:

Problema (Satisfatibilidade)

$$SAT = \{\langle f \rangle : f \text{ é uma fórmula booleana satisfazível}\}$$

Para isso, consideraremos um caso particular: quando a fórmula está escrita em formal normal conjuntiva: SAT_{FNC}:

- A fórmula é composta por conjunção de cláusulas (cláusulas relacionadas pelo operador $\wedge$).
- Cada cláusula é composta por disjunção de literais (literais relacionados pelo operador $\vee$).
- Cada literal é uma variável ou a negação de uma variável.

Satisfatibilidade

Queremos provar que o seguinte problema é NP-completo:

Problema (Satisfatibilidade)

$$SAT = \{\langle f \rangle : f \text{ é uma fórmula booleana satisfazível}\}$$

Para isso, consideraremos um caso particular: quando a fórmula está escrita em formal normal conjuntiva: SAT_{FNC}:

- A fórmula é composta por conjunção de cláusulas (cláusulas relacionadas pelo operador $\wedge$).
- Cada cláusula é composta por disjunção de literais (literais relacionados pelo operador $\vee$).
- Cada literal é uma variável ou a negação de uma variável.

Exemplo: $(\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z) \wedge (x \vee y \vee z)$.

Demonstração

Estratégia a seguir:

Demonstração

Estratégia a seguir:

1. Provar que $\text{SAT}_{FNC} \in \text{NP}$.

Demonstração

Estratégia a seguir:

1. Provar que $\text{SAT}_{FNC} \in \text{NP}$.
2. Encontrar um problema NP-completo Π.

Demonstração

Estratégia a seguir:

1. Provar que $\mathrm{SAT}_{FNC} \in \mathrm{NP}$.
2. Encontrar um problema NP-completo Π.
3. Provar que $\Pi \preccurlyeq_p \mathrm{SAT}_{FNC}$.

Demonstração

Estratégia a seguir:

1. Provar que $\text{SAT}_{FNC} \in \text{NP}$.
2. Encontrar um problema NP-completo Π.
3. Provar que $\Pi \preccurlyeq_p \text{SAT}_{FNC}$.

SAT_{FNC} é um caso particular de SAT, que está em NP (aula passada), portanto SAT_{FNC} também está.

Demonstração

Estratégia a seguir:

1. Provar que $\text{SAT}_{FNC} \in \text{NP}$.
2. Encontrar um problema NP-completo Π.
3. Provar que $\Pi \preccurlyeq_p \text{SAT}_{FNC}$.

SAT_{FNC} é um caso particular de SAT, que está em NP (aula passada), portanto SAT_{FNC} também está.

Sabemos que o seguinte problema é NP-completo:

Demonstração

Estratégia a seguir:

1. Provar que $\text{SAT}_{FNC} \in \text{NP}$.
2. Encontrar um problema NP-completo Π.
3. Provar que $\Pi \preccurlyeq_p \text{SAT}_{FNC}$.

SAT_{FNC} é um caso particular de SAT, que está em NP (aula passada), portanto SAT_{FNC} também está.

Sabemos que o seguinte problema é NP-completo:

Problema (Satisfatibilidade de circuito (C-SAT))

$$C\text{-}SAT = \{\langle c\rangle : c \text{ é um circuito lógico satisfazível}\}$$

Demonstração

Estratégia a seguir:

1. Provar que $\text{SAT}_{FNC} \in \text{NP}$.
2. Encontrar um problema NP-completo Π.
3. Provar que $\Pi \preccurlyeq_p \text{SAT}_{FNC}$.

SAT_{FNC} é um caso particular de SAT, que está em NP (aula passada), portanto SAT_{FNC} também está.

Sabemos que o seguinte problema é NP-completo:

Problema (Satisfatibilidade de circuito (C-SAT))

$$C\text{-}SAT = \{\langle c \rangle : c \text{ é um circuito lógico satisfazível}\}$$

Basta provar que $\text{C-SAT} \preccurlyeq_p \text{SAT}_{FNC}$.

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Dado um circuito lógico C, definimos a seguinte fórmula booleana em forma normal conjuntiva $F(C)$:

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Dado um circuito lógico C, definimos a seguinte fórmula booleana em forma normal conjuntiva $F(C)$:

1. Por cada fio **i** de C, definimos a variável $\mathbf{x_i}$.

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Dado um circuito lógico C, definimos a seguinte fórmula booleana em forma normal conjuntiva $F(C)$:

1. Por cada fio **i** de C, definimos a variável $\mathbf{x_i}$.

2. Por cada porta lógica de C, definimos cláusulas que garantem **equivalência** lógica com a relação entre as entradas da porta e suas saídas.

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Dado um circuito lógico C, definimos a seguinte fórmula booleana em forma normal conjuntiva $F(C)$:

1. Por cada fio **i** de C, definimos a variável $\mathbf{x_i}$.

2. Por cada porta lógica de C, definimos cláusulas que garantem **equivalência** lógica com a relação entre as entradas da porta e suas saídas.

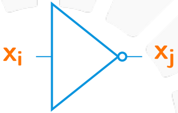
3. Finalmente, identificamos por **o** o fio de saída de C.

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Por cada porta **NOT** de C definimos duas cláusulas em $F(C)$:

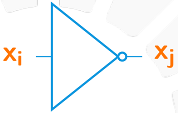
C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Por cada porta **NOT** de C definimos duas cláusulas em $F(C)$:



C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

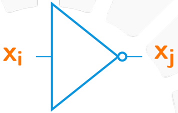
Por cada porta **NOT** de C definimos duas cláusulas em $F(C)$:



$$\neg x_i \Leftrightarrow x_j$$

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Por cada porta **NOT** de C definimos duas cláusulas em $F(C)$:



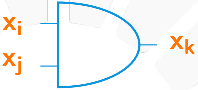
$$
\begin{aligned}
& \neg \mathbf{x_i} \Leftrightarrow \mathbf{x_j} \\
\equiv \quad & (\neg \mathbf{x_i} \lor \neg \mathbf{x_j}) \land (\mathbf{x_i} \lor \mathbf{x_j})
\end{aligned}
$$

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Por cada porta **AND** de C definimos três cláusulas em $F(C)$:

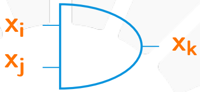
C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Por cada porta **AND** de C definimos três cláusulas em $F(C)$:



C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

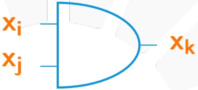
Por cada porta **AND** de C definimos três cláusulas em $F(C)$:



$\mathbf{x_i} \wedge \mathbf{x_j} \Leftrightarrow \mathbf{x_k}$

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

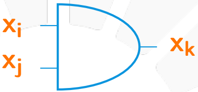
Por cada porta **AND** de C definimos três cláusulas em $F(C)$:



$$
\begin{aligned}
& \mathbf{x_i} \land \mathbf{x_j} \Leftrightarrow \mathbf{x_k} \\
\equiv \quad & (\neg \mathbf{x_i} \lor \neg \mathbf{x_j} \lor \mathbf{x_k}) \land (\mathbf{x_i} \lor \neg \mathbf{x_k}) \land (\mathbf{x_j} \lor \neg \mathbf{x_k})
\end{aligned}
$$

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Por cada porta **AND** de C definimos três cláusulas em $F(C)$:



$$
\begin{aligned}
& x_i \wedge x_j \Leftrightarrow x_k \\
\equiv \quad & (\neg x_i \vee \neg x_j \vee x_k) \wedge (x_i \vee \neg x_k) \wedge (x_j \vee \neg x_k)
\end{aligned}
$$

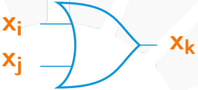
Se uma porta **AND** tiver m fios de entrada, então definimos $m + 1$ cláusulas.

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Por cada porta **OR** de C definimos três cláusulas em $F(C)$:

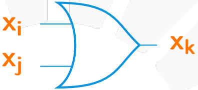
C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Por cada porta **OR** de C definimos três cláusulas em $F(C)$:



C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

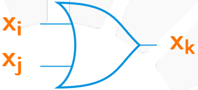
Por cada porta **OR** de C definimos três cláusulas em $F(C)$:



$\mathbf{x_i} \vee \mathbf{x_j} \Leftrightarrow \mathbf{x_k}$

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

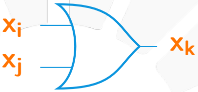
Por cada porta **OR** de C definimos três cláusulas em $F(C)$:



$$
\begin{aligned}
& x_i \lor x_j \Leftrightarrow x_k \\
\equiv \quad & (x_i \lor x_j \lor \neg x_k) \land (\neg x_i \lor x_k) \land (\neg x_j \lor x_k)
\end{aligned}
$$

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Por cada porta **OR** de C definimos três cláusulas em $F(C)$:



$$
\begin{aligned}
& x_i \vee x_j \Leftrightarrow x_k \\
\equiv \quad & (x_i \vee x_j \vee \neg x_k) \wedge (\neg x_i \vee x_k) \wedge (\neg x_j \vee x_k)
\end{aligned}
$$

Se uma porta **OR** tiver m fios de entrada, então definimos $m + 1$ cláusulas.

C-SAT $\preccurlyeq_p$ SAT$_{FNC}$

Dado um circuito lógico C, $F(C)$ será da forma:

$$\mathbf{x_o} \wedge \psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_m$$

Onde $\mathbf{x_o}$ corresponde ao fio de saída o de C e cada fórmula ψ_k corresponde às cláusulas em forma normal conjuntiva geradas pela porta k de C. Portanto, C é satisfazível se e somente se $F(C)$ for satisfazível.

O número de variáveis em $F(C)$ é igual ao número de fios em C e o número de cláusulas não é maior que o número de fios multiplicado pelo número de portas em C. Portanto, $F(C)$ pode ser construída em tempo polinomial a partir de C.

Teoremas

Teorema

SAT_{FNC} *é* NP-*completo.*

3-SAT

Outro caso especial de SAT é quando a fórmula está em forma normal conjuntiva e cada cláusula tem exatamente três literais.

3-SAT

Outro caso especial de SAT é quando a fórmula está em forma normal conjuntiva e cada cláusula tem exatamente três literais.

Teorema

3-SAT é NP*-completo.*

3-SAT

Outro caso especial de SAT é quando a fórmula está em forma normal conjuntiva e cada cláusula tem exatamente três literais.

Teorema

3-SAT é NP*-completo.*

Por ser um caso particular de SAT, sabemos que 3-SAT está em NP.

3-SAT

Outro caso especial de SAT é quando a fórmula está em forma normal conjuntiva e cada cláusula tem exatamente três literais.

Teorema

3-SAT é NP*-completo.*

Por ser um caso particular de SAT, sabemos que 3-SAT está em NP.

Basta provar que é NP-difícil.

$\mathrm{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Considere uma fórmula ψ em forma normal conjuntiva, construímos a fórmula $F(\psi)$ em forma normal conjuntiva com exatamente três literais por cláusula, tal que $\psi \in \mathrm{SAT}_{FNC}$ se e somente se $F(\psi) \in$ 3-SAT.

$\mathrm{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Considere uma fórmula ψ em forma normal conjuntiva, construímos a fórmula $F(\psi)$ em forma normal conjuntiva com exatamente três literais por cláusula, tal que $\psi \in \mathrm{SAT}_{FNC}$ se e somente se $F(\psi) \in$ 3-SAT.

Para construir $F(\psi)$, cada cláusula de ψ com número de literais diferentes de três será substituída por novas cláusulas, podendo adicionar novas variáveis.

$\text{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Cada cláusula c ψ com somente um literal ($c = \ell$), é substituída por quatro novas cláusulas que adicionam duas novas variáveis ($x_{c,1}$ e $x_{c,2}$):

$\text{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Cada cláusula c ψ com somente um literal ($c = \ell$), é substituída por quatro novas cláusulas que adicionam duas novas variáveis ($x_{c,1}$ e $x_{c,2}$):

$$(\ell \vee x_{c,1} \vee x_{c,2}) \wedge (\ell \vee x_{c,1} \vee \neg x_{c,2}) \wedge (\ell \vee \neg x_{c,1} \vee x_{c,2}) \wedge (\ell \vee \neg x_{c,1} \vee \neg x_{c,2})$$

$\text{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Cada cláusula c de ψ com somente dois literais ($c = \ell_1 \vee \ell_2$), é substituída por dois novas cláusulas que adicionam uma nova variável (x_c):

$\text{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Cada cláusula c de ψ com somente dois literais ($c = \ell_1 \vee \ell_2$), é substituída por dois novas cláusulas que adicionam uma nova variável (x_c):

$$(\ell_1 \vee \ell_2 \vee x_c) \wedge (\ell_1 \vee \ell_2 \vee \neg x_c)$$

$\mathrm{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Cada cláusula c de ψ com $k > 3$ literais ($c = \ell_1 \vee \ell_2 \vee \ell_3 \vee \ldots \vee \ell_k$) é substituída por $k-2$ novas cláusulas que adicionam $k-3$ novas variáveis ($x_{c,1}, x_{c,2}, \ldots, x_{c,k-3}$):

$\text{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Cada cláusula c de ψ com $k > 3$ literais ($c = \ell_1 \vee \ell_2 \vee \ell_3 \vee \ldots \vee \ell_k$) é substituída por $k-2$ novas cláusulas que adicionam $k-3$ novas variáveis ($x_{c,1}, x_{c,2}, \ldots, x_{c,k-3}$):

$$
\begin{aligned}
&(\ell_1 \vee \ell_2 \vee x_{c,1}) \wedge (\neg x_{c,1} \vee \ell_3 \vee x_{c,2}) \wedge (\neg x_{c,2} \vee \ell_4 \vee x_{c,3}) \\
&\wedge \ldots \wedge (\neg x_{c,i-2} \vee \ell_i \vee x_{c,i-1}) \wedge \ldots \\
&\wedge (\neg x_{c,k-4} \vee \ell_{k-2} \vee x_{c,k-3}) \wedge (\neg x_{c,k-3} \vee \ell_{k-1} \vee \ell_k)
\end{aligned}
$$

$\mathrm{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Dada uma fórmula booleana ψ em forma normal conjuntiva, a fórmula booleana $F(\psi)$ em forma normal conjuntiva com exatamente três literais por cláusula garante que $\psi \in \mathrm{SAT}_{FNC}$ se e somente se $F(\psi) \in$ 3-SAT.

$\mathrm{SAT}_{FNC} \preccurlyeq_p$ 3-SAT

Dada uma fórmula booleana ψ em forma normal conjuntiva, a fórmula booleana $F(\psi)$ em forma normal conjuntiva com exatamente três literais por cláusula garante que $\psi \in \mathrm{SAT}_{FNC}$ se e somente se $F(\psi) \in$ 3-SAT.

Por cada cláusula c de ψ o número de novas cláusulas e variáveis em $F(\psi)$ é no máximo 4 ou o número de literais em c. Portanto, a construção de $F(\psi)$ pode ser feita em tempo polinomial no tamanho de ψ.

Um problema em grafos

Uma **clique** de um grafo G é um subgrafo completo de G. Ou seja, um subgrafo de G com uma aresta entre cada par de vértices.

Um problema em grafos

Uma **clique** de um grafo G é um subgrafo completo de G. Ou seja, um subgrafo de G com uma aresta entre cada par de vértices.

Problema (Clique)

$$CLIQUE = \{\langle G, k\rangle : G \text{ é um grafo com uma clique de } k \text{ vértices}\}$$

Um problema em grafos

Uma **clique** de um grafo G é um subgrafo completo de G. Ou seja, um subgrafo de G com uma aresta entre cada par de vértices.

Problema (Clique)

$$CLIQUE = \{\langle G, k\rangle : G \textit{ é um grafo com uma clique de } k \textit{ vértices}\}$$

Teorema

CLIQUE é NP*-completo.*

CLIQUE $\in$ NP

Algoritmo 1: $\textsc{Verifica-Clique}(\langle G, k\rangle, \langle C\rangle)$

se *C não for subgrafo de G*
  devolva NAO
se *C não for uma clique*
  devolva NAO
se *C não tiver k vértices*
  devolva NAO
devolva SIM

3-SAT $\preccurlyeq_p$ CLIQUE

Dada uma fórmula booleana ψ em forma normal conjuntiva com m cláusulas e exatamente três literais por cláusula, construímos uma instância $\langle G, k \rangle = F(\psi)$ da CLIQUE como segue:

3-SAT $\preccurlyeq_p$ CLIQUE

Dada uma fórmula booleana ψ em forma normal conjuntiva com m cláusulas e exatamente três literais por cláusula, construímos uma instância $\langle G, k\rangle = F(\psi)$ da CLIQUE como segue:

- Por cada cláusula $c = \ell_1 \vee \ell_2 \vee \ell_3$ de ψ, definimos um **cluster** em G que consiste em três vértices (um por cada literal de c): v_{c,ℓ_1}, v_{c,ℓ_2} e v_{c,ℓ_3}.

3-SAT $\preccurlyeq_p$ CLIQUE

Dada uma fórmula booleana ψ em forma normal conjuntiva com m cláusulas e exatamente três literais por cláusula, construímos uma instância $\langle G, k\rangle = F(\psi)$ da CLIQUE como segue:

- Por cada cláusula $c = \ell_1 \vee \ell_2 \vee \ell_3$ de ψ, definimos um **cluster** em G que consiste em três vértices (um por cada literal de c): v_{c,ℓ_1}, v_{c,ℓ_2} e v_{c,ℓ_3}.
- Adicionamos uma aresta $(v_{c,\ell}, v_{c',\neg\ell'})$ entre cada par de vértices $v_{c,\ell}$ e $v_{c',\ell'}$ em clusters diferentes ($c \neq c'$), a menos que um dos literais associados seja a negação do outro ($\ell = \neg\ell'$).

3-SAT $\preccurlyeq_p$ CLIQUE

Dada uma fórmula booleana ψ em forma normal conjuntiva com m cláusulas e exatamente três literais por cláusula, construímos uma instância $\langle G, k\rangle = F(\psi)$ da CLIQUE como segue:

- Por cada cláusula $c = \ell_1 \vee \ell_2 \vee \ell_3$ de ψ, definimos um **cluster** em G que consiste em três vértices (um por cada literal de c): v_{c,ℓ_1}, v_{c,ℓ_2} e v_{c,ℓ_3}.
- Adicionamos uma aresta $(v_{c,\ell}, v_{c',\neg\ell'})$ entre cada par de vértices $v_{c,\ell}$ e $v_{c',\ell'}$ em clusters diferentes ($c \neq c'$), a menos que um dos literais associados seja a negação do outro ($\ell = \neg\ell'$).
- Não adicionamos arestas entre vértices do mesmo cluster.

3-SAT $\preccurlyeq_p$ CLIQUE

Dada uma fórmula booleana ψ em forma normal conjuntiva com m cláusulas e exatamente três literais por cláusula, construímos uma instância $\langle G, k\rangle = F(\psi)$ da CLIQUE como segue:

- Por cada cláusula $c = \ell_1 \vee \ell_2 \vee \ell_3$ de ψ, definimos um **cluster** em G que consiste em três vértices (um por cada literal de c): v_{c,ℓ_1}, v_{c,ℓ_2} e v_{c,ℓ_3}.
- Adicionamos uma aresta $(v_{c,\ell}, v_{c',\neg\ell'})$ entre cada par de vértices $v_{c,\ell}$ e $v_{c',\ell'}$ em clusters diferentes ($c \neq c'$), a menos que um dos literais associados seja a negação do outro ($\ell = \neg\ell'$).
- Não adicionamos arestas entre vértices do mesmo cluster.
- Finalmente, definimos o parâmetro k como o número de cláusulas ($k = m$).

3-SAT $\preccurlyeq_p$ CLIQUE

Dada uma fórmula booleana ψ em forma normal conjuntiva com m cláusulas e exatamente três literais por cláusula, construímos uma instância $\langle G, k\rangle = F(\psi)$ da CLIQUE como segue:

- Por cada cláusula $c = \ell_1 \vee \ell_2 \vee \ell_3$ de ψ, definimos um **cluster** em G que consiste em três vértices (um por cada literal de c): v_{c,ℓ_1}, v_{c,ℓ_2} e v_{c,ℓ_3}.
- Adicionamos uma aresta $(v_{c,\ell}, v_{c',\neg\ell'})$ entre cada par de vértices $v_{c,\ell}$ e $v_{c',\ell'}$ em clusters diferentes ($c \neq c'$), a menos que um dos literais associados seja a negação do outro ($\ell = \neg\ell'$).
- Não adicionamos arestas entre vértices do mesmo cluster.
- Finalmente, definimos o parâmetro k como o número de cláusulas ($k = m$).

Note que $\langle G, k\rangle$ pode ser construída em tempo polinomial no tamanho de ψ.

3-SAT $\preccurlyeq_p$ CLIQUE

Antes de provar que $\langle \psi \rangle \in$ 3-SAT se e somente se $\langle G, m \rangle \in$ CLIQUE, consideremos as seguintes propriedade:

3-SAT $\preccurlyeq_p$ CLIQUE

Antes de provar que $\langle\psi\rangle \in$ 3-SAT se e somente se $\langle G, m\rangle \in$ CLIQUE, consideremos as seguintes propriedade:

- Se dois vértices $v_{c,\ell}$ e $v_{c',\ell'}$ em G são adjacentes, então os literais associados podem ser simultaneamente verdadeiros ($\ell = \ell' = 1$).

3-SAT $\preccurlyeq_p$ CLIQUE

Antes de provar que $\langle \psi \rangle \in$ 3-SAT se e somente se $\langle G, m \rangle \in$ CLIQUE, consideremos as seguintes propriedade:

- Se dois vértices $v_{c,\ell}$ e $v_{c',\ell'}$ em G são adjacentes, então os literais associados podem ser simultaneamente verdadeiros ($\ell = \ell' = 1$).
- Se dois litareis ℓ e ℓ' de cláusulas diferentes c e c' ($c \neq c'$) podem ser simultaneamente verdadeiros, então os vértices associados $v_{c,\ell}$ e $v_{c',\ell'}$ são adjacentes.

3-SAT $\preccurlyeq_p$ CLIQUE

$\langle \psi \rangle \in$ 3-SAT se e somente se $\langle G, m \rangle \in$ CLIQUE:

3-SAT $\preccurlyeq_p$ CLIQUE

$\langle \psi \rangle \in$ 3-SAT se e somente se $\langle G, m \rangle \in$ CLIQUE:

$\Rightarrow$ Se ψ é satisfazível, então denote por α uma atribuição de valores que a satisfaz. Por cada cláusula, selecione um literal que faz essa cláusula verdadeira com a atribuição α. Os m vértices associados aos literais selecionados são adjacentes em G, portanto formam uma clique de m vértices.

3-SAT $\preccurlyeq_p$ CLIQUE

$\langle \psi \rangle \in$ 3-SAT se e somente se $\langle G, m \rangle \in$ CLIQUE:

$\Rightarrow$ Se ψ é satisfazível, então denote por α uma atribuição de valores que a satisfaz. Por cada cláusula, selecione um literal que faz essa cláusula verdadeira com a atribuição α. Os m vértices associados aos literais selecionados são adjacentes em G, portanto formam uma clique de m vértices.

$\Leftarrow$ Se G tem uma clique com m vértices, então há um vértice de cada cluster nela (vértices do mesmo cluster não podem estar na mesma clique). Como todos os vértices da clique são adjacentes, significa que os literais associados podem ser simultaneamente verdadeiros e como cada literal associado corresponde exatamente a uma das m cláusulas de ψ, temos que a fórmula é satisfazível.

Programação linear inteira

Considere uma versão de decisão para programação linear binária:

Programação linear inteira

Considere uma versão de decisão para programação linear binária:

Problema (Programação linear inteira (PLI))

$$PLI = \begin{array}{l} \{\langle c, A, b, k\rangle : c \in \mathbb{Z}^n, A \in \mathbb{Z}^{n\times m}, b \in \mathbb{Z}^m, k \in \mathbb{Z}, \\ \quad existe\ x \in \{0,1\}^n\ tal\ que\ Ax \leq b\ e\ c^t x \geq k\} \end{array}$$

Programação linear inteira

Considere uma versão de decisão para programação linear binária:

Problema (Programação linear inteira (PLI))

$$PLI = \begin{array}{l} \{\langle c, A, b, k\rangle : c \in \mathbb{Z}^n, A \in \mathbb{Z}^{n\times m}, b \in \mathbb{Z}^m, k \in \mathbb{Z}, \\ \quad \textit{existe } x \in \{0,1\}^n \textit{ tal que } Ax \leq b \textit{ e } c^t x \geq k\} \end{array}$$

Teorema

PLI é NP*-completo.*

3-SAT $\preccurlyeq_p$ PLI

Dada ψ uma fórmula booleana em forma normal conjuntiva com três literais por cláusula, definimos o seguinte problema de programação linear inteira:

3-SAT $\preccurlyeq_p$ PLI

Dada ψ uma fórmula booleana em forma normal conjuntiva com três literais por cláusula, definimos o seguinte problema de programação linear inteira:

Variáveis:

3-SAT $\preccurlyeq_p$ PLI

Dada ψ uma fórmula booleana em forma normal conjuntiva com três literais por cláusula, definimos o seguinte problema de programação linear inteira:

Variáveis:

- Por cada variável v de ψ definimos a variável $x_v \in \{0, 1\}$.

3-SAT $\preccurlyeq_p$ PLI

Dada ψ uma fórmula booleana em forma normal conjuntiva com três literais por cláusula, definimos o seguinte problema de programação linear inteira:

Variáveis:

- Por cada variável v de ψ definimos a variável $x_v \in \{0, 1\}$.
- Por cada cláusula c de ψ definimos a variável $y_c \in \{0, 1\}$.

3-SAT $\preccurlyeq_p$ PLI

Dada ψ uma fórmula booleana em forma normal conjuntiva com três literais por cláusula, definimos o seguinte problema de programação linear inteira:

Variáveis:

- Por cada variável v de ψ definimos a variável $x_v \in \{0, 1\}$.
- Por cada cláusula c de ψ definimos a variável $y_c \in \{0, 1\}$.

Função objetivo:

3-SAT $\preccurlyeq_p$ PLI

Dada ψ uma fórmula booleana em forma normal conjuntiva com três literais por cláusula, definimos o seguinte problema de programação linear inteira:

Variáveis:

- Por cada variável v de ψ definimos a variável $x_v \in \{0, 1\}$.
- Por cada cláusula c de ψ definimos a variável $y_c \in \{0, 1\}$.

Função objetivo:

- max $\sum_{j=0}^{m} z_j$.

3-SAT $\preccurlyeq_p$ PLI. Restrições

- Por cada cláusula $c = (u \vee v \vee w)$, definimos a restrição: $y_c - x_u - x_v - x_w \leq 0$.

3-SAT $\preccurlyeq_p$ PLI. Restrições

- Por cada cláusula $c = (u \vee v \vee w)$, definimos a restrição: $y_c - x_u - x_v - x_w \leq 0$.

- Por cada cláusula $c = (u \vee v \vee \neg w)$, definimos a restrição: $y_c - x_u - x_v + x_w \leq 1$.

3-SAT $\preccurlyeq_p$ PLI. Restrições

- Por cada cláusula $c = (u \vee v \vee w)$, definimos a restrição: $y_c - x_u - x_v - x_w \leq 0$.
- Por cada cláusula $c = (u \vee v \vee \neg w)$, definimos a restrição: $y_c - x_u - x_v + x_w \leq 1$.
- Por cada cláusula $c = (u \vee \neg v \vee \neg w)$, definimos a restrição: $y_c - x_u + x_v + x_w \leq 2$.

3-SAT $\preccurlyeq_p$ PLI. Restrições

- Por cada cláusula $c = (u \vee v \vee w)$, definimos a restrição: $y_c - x_u - x_v - x_w \leq 0$.
- Por cada cláusula $c = (u \vee v \vee \neg w)$, definimos a restrição: $y_c - x_u - x_v + x_w \leq 1$.
- Por cada cláusula $c = (u \vee \neg v \vee \neg w)$, definimos a restrição: $y_c - x_u + x_v + x_w \leq 2$.
- Por cada cláusula $c = (\neg u \vee \neg v \vee \neg w)$, definimos a restrição: $y_c + x_u + x_v + x_w \leq 3$.

3-SAT $\preccurlyeq_p$ PLI

$\psi \in$ 3-SAT se e somente se o seguinte programa linear binário tiver solução com valor objetivo de pelo menos $k = m$.

3-SAT $\preccurlyeq_p$ PLI

$\psi \in$ 3-SAT se e somente se o seguinte programa linear binário tiver solução com valor objetivo de pelo menos $k = m$.

Provamos que uma atribuição de valores faz ψ satisfazível se e somente se a mesma atribuição for viável para o programa binário e o valor objetivo for exatamente $k = m$:

3-SAT $\preccurlyeq_p$ PLI

$\psi \in$ 3-SAT se e somente se o seguinte programa linear binário tiver solução com valor objetivo de pelo menos $k = m$.

Provamos que uma atribuição de valores faz ψ satisfazível se e somente se a mesma atribuição for viável para o programa binário e o valor objetivo for exatamente $k = m$:

Para toda variável v de ψ, $x_v = v$.

3-SAT $\preccurlyeq_p$ PLI

$\psi \in$ 3-SAT se e somente se o seguinte programa linear binário tiver solução com valor objetivo de pelo menos $k = m$.

Provamos que uma atribuição de valores faz ψ satisfazível se e somente se a mesma atribuição for viável para o programa binário e o valor objetivo for exatamente $k = m$:

Para toda variável v de ψ, $x_v = v$.

A cláusula c é verdadeira se e somente se $y_c = 1$:

3-SAT $\preccurlyeq_p$ PLI

$\psi \in$ 3-SAT se e somente se o seguinte programa linear binário tiver solução com valor objetivo de pelo menos $k = m$.

Provamos que uma atribuição de valores faz ψ satisfazível se e somente se a mesma atribuição for viável para o programa binário e o valor objetivo for exatamente $k = m$:

Para toda variável v de ψ, $x_v = v$.

A cláusula c é verdadeira se e somente se $y_c = 1$:

- $c = (u \vee v \vee w) \Leftrightarrow y_c - x_u - x_v - x_w \leq 0$.

3-SAT $\preccurlyeq_p$ PLI

$\psi \in$ 3-SAT se e somente se o seguinte programa linear binário tiver solução com valor objetivo de pelo menos $k = m$.

Provamos que uma atribuição de valores faz ψ satisfazível se e somente se a mesma atribuição for viável para o programa binário e o valor objetivo for exatamente $k = m$:

Para toda variável v de ψ, $x_v = v$.

A cláusula c é verdadeira se e somente se $y_c = 1$:

- $c = (u \vee v \vee w) \Leftrightarrow y_c - x_u - x_v - x_w \leq 0.$
- $c = (u \vee v \vee \neg w) \Leftrightarrow y_c - x_u - x_v + x_w \leq 1.$

3-SAT $\preccurlyeq_p$ PLI

$\psi \in$ 3-SAT se e somente se o seguinte programa linear binário tiver solução com valor objetivo de pelo menos $k = m$.

Provamos que uma atribuição de valores faz ψ satisfazível se e somente se a mesma atribuição for viável para o programa binário e o valor objetivo for exatamente $k = m$:

Para toda variável v de ψ, $x_v = v$.

A cláusula c é verdadeira se e somente se $y_c = 1$:

- $c = (u \vee v \vee w) \Leftrightarrow y_c - x_u - x_v - x_w \leq 0$.
- $c = (u \vee v \vee \neg w) \Leftrightarrow y_c - x_u - x_v + x_w \leq 1$.
- $c = (u \vee \neg v \vee \neg w) \Leftrightarrow y_c - x_u + x_v + x_w \leq 2$.

3-SAT $\preccurlyeq_p$ PLI

$\psi \in$ 3-SAT se e somente se o seguinte programa linear binário tiver solução com valor objetivo de pelo menos $k = m$.

Provamos que uma atribuição de valores faz ψ satisfazível se e somente se a mesma atribuição for viável para o programa binário e o valor objetivo for exatamente $k = m$:

Para toda variável v de ψ, $x_v = v$.

A cláusula c é verdadeira se e somente se $y_c = 1$:

- $c = (u \vee v \vee w) \Leftrightarrow y_c - x_u - x_v - x_w \leq 0$.
- $c = (u \vee v \vee \neg w) \Leftrightarrow y_c - x_u - x_v + x_w \leq 1$.
- $c = (u \vee \neg v \vee \neg w) \Leftrightarrow y_c - x_u + x_v + x_w \leq 2$.
- $c = (\neg u \vee \neg v \vee \neg w) \Leftrightarrow y_c + x_u + x_v + x_w \leq 3$.

Truques para formular problemas como PLI

Variáveis binárias

Só tem valores no conjunto $\{0, 1\}$ e permitem modelar diferentes cenários:

Variáveis binárias

Só tem valores no conjunto $\{0, 1\}$ e permitem modelar diferentes cenários:

- Se a solução pode satisfazer ou não alguma propriedade ou condição p, então podemos definir uma variável binária para p:

$$\mathbf{x_p} = \begin{cases} 1, & \text{se a solução satisfaz } p \\ 0, & \text{em outro caso} \end{cases}$$

Variáveis binárias

Só tem valores no conjunto $\{0, 1\}$ e permitem modelar diferentes cenários:

- Se a solução pode satisfazer ou não alguma propriedade ou condição p, então podemos definir uma variável binária para p:
  $$\mathbf{x_p} = \begin{cases} 1, & \text{se a solução satisfaz } p \\ 0, & \text{em outro caso} \end{cases}$$
- Se a solução requer a seleção de elementos de alguma coleção, podemos definir uma variável binária para elemento e:
  $$\mathbf{x_e} = \begin{cases} 1, & \text{se o elemento } e \text{ for selecionado} \\ 0, & \text{em outro caso} \end{cases}$$

Variáveis binárias

Só tem valores no conjunto $\{0, 1\}$ e permitem modelar diferentes cenários:

- Se a solução pode satisfazer ou não alguma propriedade ou condição *p*, então podemos definir uma variável binária para *p*:

  $$\mathbf{x_p} = \begin{cases} 1, & \text{se a solução satisfaz } p \\ 0, & \text{em outro caso} \end{cases}$$

- Se a solução requer a seleção de elementos de alguma coleção, podemos definir uma variável binária para elemento *e*:

  $$\mathbf{x_e} = \begin{cases} 1, & \text{se o elemento } e \text{ for selecionado} \\ 0, & \text{em outro caso} \end{cases}$$

- Se a solução precisa dar ordem aos elementos de um conjunto, podemos definir uma variável binária entre cada par de elementos i, j:

  $$\mathbf{x_{ij}} = \begin{cases} 1, & \text{se o elemento } i \text{ for antes do } j \\ 0, & \text{em outro caso} \end{cases}$$

Variáveis binárias

Só tem valores no conjunto $\{0, 1\}$ e permitem modelar diferentes cenários:

- Se a solução pode satisfazer ou não alguma propriedade ou condição p, então podemos definir uma variável binária para p:
  $$\mathbf{x_p} = \begin{cases} 1, & \text{se a solução satisfaz } p \\ 0, & \text{em outro caso} \end{cases}$$
- Se a solução requer a seleção de elementos de alguma coleção, podemos definir uma variável binária para elemento e:
  $$\mathbf{x_e} = \begin{cases} 1, & \text{se o elemento } e \text{ for selecionado} \\ 0, & \text{em outro caso} \end{cases}$$
- Se a solução precisa dar ordem aos elementos de um conjunto, podemos definir uma variável binária entre cada par de elementos i, j:
  $$\mathbf{x_{ij}} = \begin{cases} 1, & \text{se o elemento } i \text{ for antes do } j \\ 0, & \text{em outro caso} \end{cases}$$
- Se a solução precisa associar ou atribuir pares de elementos, podemos definir uma variável binária para cada par i, j:
  $$\mathbf{x_{ij}} = \begin{cases} 1, & \text{se o elemento } i \text{ for associado com o } j \\ 0, & \text{em outro caso} \end{cases}$$

Restrições lógicas com variáveis binárias

NOT: A restrição seguinte garante que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = 0$:

Restrições lógicas com variáveis binárias

NOT: A restrição seguinte garante que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = 0$:

$$\mathbf{z} = 1 - \mathbf{x}$$

Restrições lógicas com variáveis binárias

NOT: A restrição seguinte garante que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = 0$:

$$\mathbf{z} = 1 - \mathbf{x}$$

$z = \neg x$	x
0	1
1	0

Restrições lógicas com variáveis binárias

AND: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = 1$ e $\mathbf{y} = 1$:

Restrições lógicas com variáveis binárias

AND: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = 1$ e $\mathbf{y} = 1$:

$$
\begin{aligned}
2\mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\
\mathbf{z} + 1 &\geq \mathbf{x} + \mathbf{y}
\end{aligned}
$$

Restrições lógicas com variáveis binárias

AND: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = 1$ e $\mathbf{y} = 1$:

$$\begin{aligned} 2\mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\ \mathbf{z} + 1 &\geq \mathbf{x} + \mathbf{y} \end{aligned}$$

$z = x \wedge y$	x	y
1	1	1
0	1	0
0	0	1
0	0	0

Restrições lógicas com variáveis binárias

OR: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = 1$ ou $\mathbf{y} = 1$:

Restrições lógicas com variáveis binárias

OR: As seguintes restrições garantem que $z = 1$ se e somente se $x = 1$ ou $y = 1$:

$$\begin{aligned} z &\leq x + y \\ 2z &\geq x + y \end{aligned}$$

Restrições lógicas com variáveis binárias

OR: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = 1$ ou $\mathbf{y} = 1$:

$$\begin{aligned} \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\ 2\mathbf{z} &\geq \mathbf{x} + \mathbf{y} \end{aligned}$$

$z = x \vee y$	x	y
1	1	1
1	1	0
1	0	1
0	0	0

Restrições lógicas com variáveis binárias

OR: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = 1$ ou $\mathbf{y} = 1$:

$$\begin{aligned} \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\ 2\mathbf{z} &\geq \mathbf{x} + \mathbf{y} \end{aligned}$$

$z = x \vee y$	x	y
1	1	1
1	1	0
1	0	1
0	0	0

Para **obrigar** $\mathbf{x} \vee \mathbf{y}$ podemos usar a restrição $\mathbf{x} + \mathbf{y} \geq 1$.

Restrições lógicas com variáveis binárias

XOR: As seguintes restrições garantem que $z = 1$ se e somente se exatamente um entre x ou y for 1:

.

Restrições lógicas com variáveis binárias

XOR: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se exatamente um entre $\mathbf{x}$ ou $\mathbf{y}$ for 1:

$$\begin{aligned} \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\ 2 - \mathbf{z} &\geq \mathbf{x} + \mathbf{y} \\ \mathbf{z} &\geq \mathbf{x} - \mathbf{y} \\ \mathbf{z} &\geq \mathbf{y} - \mathbf{x} \end{aligned}$$

Restrições lógicas com variáveis binárias

XOR: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se exatamente um entre $\mathbf{x}$ ou $\mathbf{y}$ for 1:

$$\begin{aligned} \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\ 2 - \mathbf{z} &\geq \mathbf{x} + \mathbf{y} \\ \mathbf{z} &\geq \mathbf{x} - \mathbf{y} \\ \mathbf{z} &\geq \mathbf{y} - \mathbf{x} \end{aligned}$$

$z = x \oplus y$	x	y
0	1	1
1	1	0
1	0	1
0	0	0

.

Restrições lógicas com variáveis binárias

XOR: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se exatamente um entre $\mathbf{x}$ ou $\mathbf{y}$ for 1:

$$\begin{aligned} \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\ 2 - \mathbf{z} &\geq \mathbf{x} + \mathbf{y} \\ \mathbf{z} &\geq \mathbf{x} - \mathbf{y} \\ \mathbf{z} &\geq \mathbf{y} - \mathbf{x} \end{aligned}$$

$z = x \oplus y$	x	y
0	1	1
1	1	0
1	0	1
0	0	0

Para **obrigar** $\mathbf{x} \oplus \mathbf{y}$ podemos usar a restrição $\mathbf{x} + \mathbf{y} = 1$.

Restrições lógicas com variáveis binárias

EQUIVALÊNCIA: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = \mathbf{y}$:

Restrições lógicas com variáveis binárias

EQUIVALÊNCIA: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = \mathbf{y}$:

$$
\begin{aligned}
1 - \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\
1 + \mathbf{z} &\geq \mathbf{x} + \mathbf{y} \\
1 - \mathbf{z} &\geq \mathbf{x} - \mathbf{y} \\
1 - \mathbf{z} &\geq \mathbf{y} - \mathbf{x}
\end{aligned}
$$

Restrições lógicas com variáveis binárias

EQUIVALÊNCIA: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = \mathbf{y}$:

$$
\begin{aligned}
1 - \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\
1 + \mathbf{z} &\geq \mathbf{x} + \mathbf{y} \\
1 - \mathbf{z} &\geq \mathbf{x} - \mathbf{y} \\
1 - \mathbf{z} &\geq \mathbf{y} - \mathbf{x}
\end{aligned}
$$

$z = x \Leftrightarrow y$	x	y
1	1	1
0	1	0
0	0	1
1	0	0

Restrições lógicas com variáveis binárias

EQUIVALÊNCIA: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{x} = \mathbf{y}$:

$$\begin{aligned} 1 - \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\ 1 + \mathbf{z} &\geq \mathbf{x} + \mathbf{y} \\ 1 - \mathbf{z} &\geq \mathbf{x} - \mathbf{y} \\ 1 - \mathbf{z} &\geq \mathbf{y} - \mathbf{x} \end{aligned}$$

$z = x \Leftrightarrow y$	x	y
1	1	1
0	1	0
0	0	1
1	0	0

Para **obrigar** $\mathbf{x} \Leftrightarrow \mathbf{y}$ podemos usar a restrição $\mathbf{x} - \mathbf{y} = 0$.

Restrições lógicas com variáveis binárias

IMPLICAÇÃO: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{y} = 1$ sempre que $\mathbf{X} = 1$:

.

Restrições lógicas com variáveis binárias

IMPLICAÇÃO: As seguintes restrições garantem que $z = 1$ se e somente se $y = 1$ sempre que $x = 1$:

$$\begin{aligned} z - 1 &\leq y - x \\ 2z - 1 &\geq y - x \end{aligned}$$

.

Restrições lógicas com variáveis binárias

IMPLICAÇÃO: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{y} = 1$ sempre que $\mathbf{X} = 1$:

$$\begin{aligned} \mathbf{z} - 1 &\leq \mathbf{y} - \mathbf{x} \\ 2\mathbf{z} - 1 &\geq \mathbf{y} - \mathbf{x} \end{aligned}$$

$z = x \Rightarrow y$	x	y
1	1	1
0	1	0
1	0	1
1	0	0

.

Restrições lógicas com variáveis binárias

IMPLICAÇÃO: As seguintes restrições garantem que $\mathbf{z} = 1$ se e somente se $\mathbf{y} = 1$ sempre que $\mathbf{X} = 1$:

$$\begin{aligned} \mathbf{z} - 1 &\leq \mathbf{y} - \mathbf{x} \\ 2\mathbf{z} - 1 &\geq \mathbf{y} - \mathbf{x} \end{aligned}$$

$z = x \Rightarrow y$	x	y
1	1	1
0	1	0
1	0	1
1	0	0

Para **obrigar** que $\mathbf{x} \Rightarrow \mathbf{y}$ podemos usar a restrição $\mathbf{y} \geq \mathbf{x}$.

Vinculando condições com variáveis binárias

Uma propriedade ou condição pode ser escrita de forma linear como:

.

Vinculando condições com variáveis binárias

Uma propriedade ou condição pode ser escrita de forma linear como:

$$a^t \mathbf{x} \leq b$$

.

Vinculando condições com variáveis binárias

Uma propriedade ou condição pode ser escrita de forma linear como:

$$a^t\mathbf{x} \leq b$$

Para vincular se é satisfeita com uma uma variável binária, devemos encontrar um valor de M grande o suficiente, tal que $a^t\mathbf{x} \leq b + M$ para qualquer solução viável $\mathbf{x}$.

Vinculando condições com variáveis binárias

Uma propriedade ou condição pode ser escrita de forma linear como:

$$a^t \mathbf{x} \leq b$$

Para vincular se é satisfeita com uma uma variável binária, devemos encontrar um valor de M grande o suficiente, tal que $a^t \mathbf{x} \leq b + M$ para qualquer solução viável $\mathbf{x}$.

Então, definimos a variável binária $\mathbf{y}$ e escrevemos a restrição:

Vinculando condições com variáveis binárias

Uma propriedade ou condição pode ser escrita de forma linear como:

$$a^t\mathbf{x} \leq b$$

Para vincular se é satisfeita com uma uma variável binária, devemos encontrar um valor de M grande o suficiente, tal que $a^t\mathbf{x} \leq b + M$ para qualquer solução viável $\mathbf{x}$.

Então, definimos a variável binária $\mathbf{y}$ e escrevemos a restrição:

$$a^t\mathbf{x} \leq b + M(1 - \mathbf{y})$$

Vinculando um conjunto de condições com variáveis

Considere um conjunto $\{c_i\}_{i=1}^{n}$ de propriedades ou condições cada uma já associada com uma variável binária $\{\mathbf{x_i}\}_{i=1}^{n}$:

Vinculando um conjunto de condições com variáveis

Considere um conjunto $\{c_i\}_{i=1}^{n}$ de propriedades ou condições cada uma já associada com uma variável binária $\{\mathbf{x_i}\}_{i=1}^{n}$:

- Para indicar se há pelo menos k condições satisfeitas, podemos usar uma variável binária $\mathbf{z}$, onde:

$$k - 1 + n\mathbf{z} \geq \sum_{i=1}^{n} \mathbf{x_i} \quad \text{e} \quad k - n(1 - \mathbf{z}) \leq \sum_{i=1}^{n} \mathbf{x_i}$$

Vinculando um conjunto de condições com variáveis

Considere um conjunto $\{c_i\}_{i=1}^{n}$ de propriedades ou condições cada uma já associada com uma variável binária $\{\mathbf{x_i}\}_{i=1}^{n}$:

- Para indicar se há pelo menos k condições satisfeitas, podemos usar uma variável binária $\mathbf{z}$, onde:

$$k - 1 + n\mathbf{z} \geq \sum_{i=1}^{n} \mathbf{x_i} \quad \text{e} \quad k - n(1 - \mathbf{z}) \leq \sum_{i=1}^{n} \mathbf{x_i}$$

- Para indicar se há no máximo k condições satisfeitas, podemos usar uma variável binária $\mathbf{z}$, onde:

$$k + 1 - n\mathbf{z} \leq \sum_{i=1}^{n} \mathbf{x_i} \quad \text{e} \quad k + n(1 - \mathbf{z}) \geq \sum_{i=1}^{n} \mathbf{x_i}$$

Restrições para selecionar elementos

Considere um conjunto de n elementos a serem selecionados (ou condições a satisfazer) cada um associado com uma variável binária $\{\mathbf{x_i}\}_{i=1}^{n}$:

Restrições para selecionar elementos

Considere um conjunto de n elementos a serem selecionados (ou condições a satisfazer) cada um associado com uma variável binária $\{\mathbf{x_i}\}_{i=1}^{n}$:

- Para selecionar (ou satisfazer) pelo menos k elementos (ou condições), podemos usar:

$$\sum_{i=1}^{n} \mathbf{x_i} \geq k$$

Restrições para selecionar elementos

Considere um conjunto de n elementos a serem selecionados (ou condições a satisfazer) cada um associado com uma variável binária $\{x_i\}_{i=1}^{n}$:

- Para selecionar (ou satisfazer) pelo menos k elementos (ou condições), podemos usar:

$$\sum_{i=1}^{n} x_i \geq k$$

- Para selecionar (ou satisfazer) exatamente k elementos (ou condições), podemos usar:

$$\sum_{i=1}^{n} x_i = k$$

Restrições para selecionar elementos

Considere um conjunto de n elementos a serem selecionados (ou condições a satisfazer) cada um associado com uma variável binária $\{\mathsf{x_i}\}_{i=1}^{n}$:

- Para selecionar (ou satisfazer) pelo menos k elementos (ou condições), podemos usar:

$$\sum_{i=1}^{n} \mathsf{x_i} \geq k$$

- Para selecionar (ou satisfazer) exatamente k elementos (ou condições), podemos usar:

$$\sum_{i=1}^{n} \mathsf{x_i} = k$$

- Para selecionar (ou satisfazer) no máximo k elementos (ou condições), podemos usar:

$$\sum_{i=1}^{n} \mathsf{x_i} \leq k$$

Variáveis e restrições para seleção de intervalos

Considere uma expressão por intervalos formada por funções lineares sem termos independentes h_i e constantes c_i:

Variáveis e restrições para seleção de intervalos

Considere uma expressão por intervalos formada por funções lineares sem termos independentes h_i e constantes c_i:

$$f(\mathbf{x}) = \begin{cases} c_1 + h_1(\mathbf{x}), & \text{se } \ell_1 \leq \mathbf{x} \leq \nu_1 \\ c_2 + h_2(\mathbf{x}), & \text{se } \ell_2 \leq \mathbf{x} \leq \nu_2 \\ \dots \\ c_n + h_n(\mathbf{x}), & \text{se } \ell_n \leq \mathbf{x} \leq \nu_n \end{cases}$$

Variáveis e restrições para seleção de intervalos

Considere uma expressão por intervalos formada por funções lineares sem termos independentes h_i e constantes c_i:

$$f(\mathbf{x}) = \begin{cases} c_1 + h_1(\mathbf{x}), & \text{se } \ell_1 \leq \mathbf{x} \leq \nu_1 \\ c_2 + h_2(\mathbf{x}), & \text{se } \ell_2 \leq \mathbf{x} \leq \nu_2 \\ \dots & \\ c_n + h_n(\mathbf{x}), & \text{se } \ell_n \leq \mathbf{x} \leq \nu_n \end{cases}$$

Para formular $f(\mathbf{x})$, podemos definir as variáveis binárias

$\mathbf{y_i} = \begin{cases} 1, & \text{se } \ell_i \leq \mathbf{x} \leq \nu_i \\ 0, & \text{em outro caso} \end{cases}$ e as variáveis $\mathbf{x_i} = \begin{cases} x, & \text{se } \ell_i \leq \mathbf{x} \leq \nu_i \\ 0, & \text{em outro caso} \end{cases}$,

re-escrevendo a expressão como:

Variáveis e restrições para seleção de intervalos

Considere uma expressão por intervalos formada por funções lineares sem termos independentes h_i e constantes c_i:

$$f(\mathbf{x}) = \begin{cases} c_1 + h_1(\mathbf{x}), & \text{se } \ell_1 \leq \mathbf{x} \leq \nu_1 \\ c_2 + h_2(\mathbf{x}), & \text{se } \ell_2 \leq \mathbf{x} \leq \nu_2 \\ \dots & \\ c_n + h_n(\mathbf{x}), & \text{se } \ell_n \leq \mathbf{x} \leq \nu_n \end{cases}$$

Para formular $f(\mathbf{x})$, podemos definir as variáveis binárias $\mathbf{y_i} = \begin{cases} 1, & \text{se } \ell_i \leq \mathbf{x} \leq \nu_i \\ 0, & \text{em outro caso} \end{cases}$ e as variáveis $\mathbf{x_i} = \begin{cases} x, & \text{se } \ell_i \leq \mathbf{x} \leq \nu_i \\ 0, & \text{em outro caso} \end{cases}$, re-escrevendo a expressão como:

$$f(\mathbf{x}) = \sum_{i=1}^{n} c_i \mathbf{y_i} + h_i(\mathbf{x_i})$$

Variáveis e restrições para seleção de intervalos

Considere uma expressão por intervalos formada por funções lineares sem termos independentes h_i e constantes c_i:

$$f(\mathbf{x}) = \begin{cases} c_1 + h_1(\mathbf{x}), & \text{se } \ell_1 \leq \mathbf{x} \leq \nu_1 \\ c_2 + h_2(\mathbf{x}), & \text{se } \ell_2 \leq \mathbf{x} \leq \nu_2 \\ \dots & \\ c_n + h_n(\mathbf{x}), & \text{se } \ell_n \leq \mathbf{x} \leq \nu_n \end{cases}$$

Para formular $f(\mathbf{x})$, podemos definir as variáveis binárias

$\mathbf{y_i} = \begin{cases} 1, & \text{se } \ell_i \leq \mathbf{x} \leq \nu_i \\ 0, & \text{em outro caso} \end{cases}$ e as variáveis $\mathbf{x_i} = \begin{cases} x, & \text{se } \ell_i \leq \mathbf{x} \leq \nu_i \\ 0, & \text{em outro caso} \end{cases}$,

re-escrevendo a expressão como:

$$f(\mathbf{x}) = \sum_{i=1}^{n} c_i \mathbf{y_i} + h_i(\mathbf{x_i})$$

Ademais, para cada i, adicionamos a restrição $\ell_i \mathbf{y_i} \leq \mathbf{x_i} \leq \nu_i \mathbf{y_i}$. Também adicionamos $\sum_{i=1}^{n} \mathbf{y_i} = 1$.

Restrições para seleção condicional

Considere três condições (ou elementos), cada uma associada com uma das variáveis binárias $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{z}$. Para garantir que $\mathbf{z} = 1$ sempre que $\mathbf{x} = 1$ e $\mathbf{y} = 1$, podemos adicionar a restrição:

Restrições para seleção condicional

Considere três condições (ou elementos), cada uma associada com uma das variáveis binárias $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{z}$. Para garantir que $\mathbf{z} = 1$ sempre que $\mathbf{x} = 1$ e $\mathbf{y} = 1$, podemos adicionar a restrição:

$$\mathbf{x} + \mathbf{y} \leq 1 + \mathbf{z}$$

Restrições para seleção condicional

Considere três condições (ou elementos), cada uma associada com uma das variáveis binárias $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{z}$. Para garantir que $\mathbf{z} = 1$ sempre que $\mathbf{x} = 1$ e $\mathbf{y} = 1$, podemos adicionar a restrição:

$$\mathbf{x} + \mathbf{y} \leq 1 + \mathbf{z}$$

Generalizando para um conjunto de $n + 1$ variáveis binárias, a condição seria: se $\mathbf{x_1} = \mathbf{x_2} = \ldots = \mathbf{x_n} = 1$ então $\mathbf{x_{n+1} = 1}$, que pode ser formulada como:

Restrições para seleção condicional

Considere três condições (ou elementos), cada uma associada com uma das variáveis binárias $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{z}$. Para garantir que $\mathbf{z} = 1$ sempre que $\mathbf{x} = 1$ e $\mathbf{y} = 1$, podemos adicionar a restrição:

$$\mathbf{x} + \mathbf{y} \leq 1 + \mathbf{z}$$

Generalizando para um conjunto de $n + 1$ variáveis binárias, a condição seria: se $\mathbf{x_1} = \mathbf{x_2} = \ldots = \mathbf{x_n} = 1$ então $\mathbf{x_{n+1} = 1}$, que pode ser formulada como:

$$\sum_{i=1}^{n} \mathbf{x_i} \leq n - 1 + \mathbf{x_{n+1}}$$

Restrições para ordem

Dados n elementos, se desejamos expressar uma ordem entre os elementos na solução, podemos usar, para cada par de elementos (i, j) uma variável binária $\mathbf{x_{ij}}$ que expresse se o elemento i está na frente do j e adicionar as seguintes restrições de ordem:

Restrições para ordem

Dados n elementos, se desejamos expressar uma ordem entre os elementos na solução, podemos usar, para cada par de elementos (i, j) uma variável binária $\mathbf{x_{ij}}$ que expresse se o elemento i está na frente do j e adicionar as seguintes restrições de ordem:

$$\mathbf{x_{ij}} + \mathbf{x_{ji}} = 1 \qquad \text{para cada par de elementos } (i, j) \text{ (antisimetria)}$$

Restrições para ordem

Dados n elementos, se desejamos expressar uma ordem entre os elementos na solução, podemos usar, para cada par de elementos (i, j) uma variável binária $\mathbf{x_{ij}}$ que expresse se o elemento i está na frente do j e adicionar as seguintes restrições de ordem:

$$\mathbf{x_{ij}} + \mathbf{x_{ji}} = 1 \qquad \text{para cada par de elementos } (i, j) \text{ (antisimetria)}$$

$$\mathbf{x_{ij}} + \mathbf{x_{jk}} \leq 1 + \mathbf{x_{ik}} \qquad \text{para cada três elementos } (i, j, k) \text{ (transitividade)}$$

Restrições para ordem

Outra opção seria usar as variáveis binárias x_{ij} para cada possível elemento i e posição j, indicando se o elemento está naquela posição ou não. Nesse caso, podemos adicionar as seguintes restrições:

Restrições para ordem

Outra opção seria usar as variáveis binárias x_{ij} para cada possível elemento i e posição j, indicando se o elemento está naquela posição ou não. Nesse caso, podemos adicionar as seguintes restrições:

$$\sum_{i=1}^{n} \mathbf{x_{ij}} = 1 \qquad \text{para cada posição } j \text{ (um elemento por posição)}$$

Restrições para ordem

Outra opção seria usar as variáveis binárias x_{ij} para cada possível elemento i e posição j, indicando se o elemento está naquela posição ou não. Nesse caso, podemos adicionar as seguintes restrições:

$$\sum_{i=1}^{n} \mathbf{x_{ij}} = 1 \qquad \text{para cada posição } j \text{ (um elemento por posição)}$$

$$\sum_{j=1}^{n} \mathbf{x_{ij}} = 1 \qquad \text{para cada elemento } i \text{ (uma posição por elemento)}$$

Exemplos de formulações

Partição balanceada

Dado um conjunto finito V de inteiros positivos, procurar uma partição de V em dois conjuntos U e $V \setminus U$ que minimize $\left|\sum_{v \in U} v - \sum_{v \in V \setminus U} v\right|$.

Partição balanceada

Por cada elemento $v \in V$ precisamos decidir se o elemento estará em U ou fora de U, para isso podemos usar uma variável binária:

Partição balanceada

Por cada elemento $v \in V$ precisamos decidir se o elemento estará em U ou fora de U, para isso podemos usar uma variável binária:

$$\mathbf{x_v} = \begin{cases} 1, & \text{se o elemento } v \text{ está em} \\ 0, & \text{em outro caso} \end{cases}$$

Partição balanceada

Função objetivo não linear:

Partição balanceada

Função objetivo não linear:

$$\min \quad \left| \sum_{v \in V} v \mathbf{x_v} - \sum_{v \in V} v(1 - \mathbf{x_v}) \right|$$

Partição balanceada

Função objetivo não linear:

$$\min \quad \left| \sum_{v \in V} v\mathbf{x_v} - \sum_{v \in V} v(1 - \mathbf{x_v}) \right|$$

Podemos linearizar essa função, minimizando uma variável inteira $\mathbf{z} \in \mathbb{Z}_+$, tal que:

Partição balanceada

Função objetivo não linear:

$$\min \quad \left| \sum_{v \in V} v \mathbf{x_v} - \sum_{v \in V} v(1 - \mathbf{x_v}) \right|$$

Podemos linearizar essa função, minimizando uma variável inteira $\mathbf{z} \in \mathbb{Z}_+$, tal que:

$$\sum_{v \in V} v \mathbf{x_v} - \sum_{v \in V} v(1 - \mathbf{x_v}) \leq \mathbf{z}$$

Partição balanceada

Função objetivo não linear:

$$\min \quad \left| \sum_{v \in V} v\mathbf{x_v} - \sum_{v \in V} v(1 - \mathbf{x_v}) \right|$$

Podemos linearizar essa função, minimizando uma variável inteira $\mathbf{z} \in \mathbb{Z}_+$, tal que:

$$\sum_{v \in V} v\mathbf{x_v} - \sum_{v \in V} v(1 - \mathbf{x_v}) \leq \mathbf{z}$$

$$\sum_{v \in V} v(1 - \mathbf{x_v}) - \sum_{v \in V} v\mathbf{x_v} \leq \mathbf{z}$$

Partição balanceada

Instância: V

Partição balanceada

Instância: V

$$\min \quad \mathbf{z}$$

Partição balanceada

Instância: V

$$\begin{aligned} \min \quad & z \\ s.a.: \quad & \end{aligned}$$

Partição balanceada

Instância: V

$$\begin{aligned} \min \quad & z \\ s.a.: \quad & \sum_{v \in V} v x_v - \sum_{v \in V} v(1 - x_v) \leq z \end{aligned}$$

Partição balanceada

Instância: V

$$
\begin{aligned}
\min \quad & \mathbf{z} \\
s.a.: \quad & \\
& \sum_{v \in V} v\mathbf{x_v} - \sum_{v \in V} v(1 - \mathbf{x_v}) \leq \mathbf{z} \\
& \sum_{v \in V} v(1 - \mathbf{x_v}) - \sum_{v \in V} v\mathbf{x_v} \leq \mathbf{z}
\end{aligned}
$$

Partição balanceada

Instância: V

$$
\begin{aligned}
\min \quad & \mathbf{z} \\
s.a.: \quad & \\
& \sum_{v \in V} v\mathbf{x_v} - \sum_{v \in V} v(1 - \mathbf{x_v}) \leq \mathbf{z} \\
& \sum_{v \in V} v(1 - \mathbf{x_v}) - \sum_{v \in V} v\mathbf{x_v} \leq \mathbf{z} \\
& \mathbf{z} \in \mathbb{Z}_+, \mathbf{x_v} \in \{0, 1\} \quad \forall v \in V
\end{aligned}
$$

Clique e conjunto independente

Uma **CLIQUE** de um grafo G é um subgrafo que completo de G.

Clique e conjunto independente

Uma **CLIQUE** de um grafo G é um subgrafo que completo de G.

Um **CONJUNTO INDEPENDENTE** de um grafo G é um conjunto de vértices que não contém adjacentes.

Clique e conjunto independente

Uma **CLIQUE** de um grafo G é um subgrafo que completo de G.

Um **CONJUNTO INDEPENDENTE** de um grafo G é um conjunto de vértices que não contém adjacentes.

Formulações para encontrar a máxima clique e o máximo conjunto independente podem ser muito parecidas.

Clique e conjunto independente

Para cada vértice do grafo, podemos criar uma variável binária para decidir se ele pertence ou não à solução (seja uma clique ou um conjunto independente):

Clique e conjunto independente

Para cada vértice do grafo, podemos criar uma variável binária para decidir se ele pertence ou não à solução (seja uma clique ou um conjunto independente):

$$x_v = \begin{cases} 1, & \text{se o vértice } v \text{ está na solução} \\ 0, & \text{em outro caso} \end{cases}$$

Clique e conjunto independente

O objetivo pode ser visto como maximizar o número de vértices selecionados:

Clique e conjunto independente

O objetivo pode ser visto como maximizar o número de vértices selecionados:

$$\max \quad \sum_{v \in V} x_v$$

Clique e conjunto independente

Em uma clique, cada par de vértices selecionados precisam ser adjacentes, portanto se dois vértices u e v não forem adjacentes, os dois não podem ser selecionados para uma solução:

Clique e conjunto independente

Em uma clique, cada par de vértices selecionados precisam ser adjacentes, portanto se dois vértices u e v não forem adjacentes, os dois não podem ser selecionados para uma solução:

$$\mathbf{x_u} + \mathbf{x_v} \leq 1 \qquad \forall (u, v) \notin E$$

Clique e conjunto independente

Em uma clique, cada par de vértices selecionados precisam ser adjacentes, portanto se dois vértices u e v não forem adjacentes, os dois não podem ser selecionados para uma solução:

$$\mathbf{x_u} + \mathbf{x_v} \leq 1 \qquad \forall (u, v) \notin E$$

Em um conjunto independente, não podem ser selecionados dois vértices adjacentes, portanto se dois vértices u e v forem adjacentes, os dois não podem ser selecionados para uma solução:

Clique e conjunto independente

Em uma clique, cada par de vértices selecionados precisam ser adjacentes, portanto se dois vértices u e v não forem adjacentes, os dois não podem ser selecionados para uma solução:

$$\mathbf{x_u} + \mathbf{x_v} \leq 1 \qquad \forall (u, v) \notin E$$

Em um conjunto independente, não podem ser selecionados dois vértices adjacentes, portanto se dois vértices u e v forem adjacentes, os dois não podem ser selecionados para uma solução:

$$\mathbf{x_u} + \mathbf{x_v} \leq 1 \qquad \forall (u, v) \in E$$

Clique e conjunto independente

Clique máxima

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} x_v$$

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} \mathbf{x_v}$$
$$s.a:$$

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} \mathbf{x_v}$$

$$s.a: \quad \mathbf{x_u} + \mathbf{x_v} \leq 1 \quad \forall (u, v) \notin E$$

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} x_v$$

$s.a:$

$$x_u + x_v \leq 1 \quad \forall (u, v) \notin E$$
$$x_v \in \{0, 1\} \quad \forall v \in V$$

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} x_v$$

$$s.a:$$

$$x_u + x_v \leq 1 \quad \forall (u, v) \notin E$$

$$x_v \in \{0, 1\} \quad \forall v \in V$$

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\begin{aligned} \max \quad & \sum_{v \in V} \mathbf{x_v} \\ s.a: \quad & \\ & \mathbf{x_u} + \mathbf{x_v} \leq 1 \quad \forall (u, v) \notin E \\ & \mathbf{x_v} \in \{0, 1\} \quad \forall v \in V \end{aligned}$$

Conjunto independente máximo

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\begin{aligned} \max \quad & \sum_{v \in V} \mathbf{x_v} \\ s.a: \quad & \\ & \mathbf{x_u} + \mathbf{x_v} \leq 1 \quad \forall (u, v) \notin E \\ & \mathbf{x_v} \in \{0, 1\} \quad \forall v \in V \end{aligned}$$

Conjunto independente máximo
Instância: $G = (V, E)$

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} \mathbf{x_v}$$

$$s.a: \quad \mathbf{x_u} + \mathbf{x_v} \leq 1 \quad \forall (u,v) \notin E$$

$$\mathbf{x_v} \in \{0,1\} \quad \forall v \in V$$

Conjunto independente máximo
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} \mathbf{x_v}$$

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} x_v$$

$$s.a:$$

$$x_u + x_v \leq 1 \quad \forall (u, v) \notin E$$

$$x_v \in \{0, 1\} \quad \forall v \in V$$

Conjunto independente máximo
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} x_v$$

$$s.a:$$

Clique e conjunto independente

Clique máxima

Instância: $G = (V, E)$

$$
\begin{aligned}
\max \quad & \sum_{v \in V} x_v \\
s.a: \quad & \\
& x_u + x_v \leq 1 \quad \forall (u, v) \notin E \\
& x_v \in \{0, 1\} \quad \forall v \in V
\end{aligned}
$$

Conjunto independente máximo

Instância: $G = (V, E)$

$$
\begin{aligned}
\max \quad & \sum_{v \in V} x_v \\
s.a: \quad & \\
& x_u + x_v \leq 1 \quad \forall (u, v) \in E
\end{aligned}
$$

Clique e conjunto independente

Clique máxima
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} x_v$$

$s.a:$

$$x_u + x_v \leq 1 \quad \forall (u, v) \notin E$$

$$x_v \in \{0, 1\} \quad \forall v \in V$$

Conjunto independente máximo
Instância: $G = (V, E)$

$$\max \quad \sum_{v \in V} x_v$$

$s.a:$

$$x_u + x_v \leq 1 \quad \forall (u, v) \in E$$

$$x_v \in \{0, 1\} \quad \forall v \in V$$

Coloração mínima

Uma **COLORAÇÃO** de um grafo é uma atribuição de cores aos vértices de forma tal que não há dois vértices adjacentes com a mesma cor.

Coloração mínima

Uma **COLORAÇÃO** de um grafo é uma atribuição de cores aos vértices de forma tal que não há dois vértices adjacentes com a mesma cor.

A coloração mínima procura por uma coloração que minimize o número de cores.

Coloração mínima

Como não há mais de uma cor por vértice, no pior caso precisaremos considerar o mesmo número de cores que de vértices. Logo, podemos definir uma variável binária para cada cor possível, indicando se ela é usada ou não na solução:

Coloração mínima

Como não há mais de uma cor por vértice, no pior caso precisaremos considerar o mesmo número de cores que de vértices. Logo, podemos definir uma variável binária para cada cor possível, indicando se ela é usada ou não na solução:

$$\mathbf{x_c} = \begin{cases} 1, & \text{se a cor } c \text{ é usada} \\ 0, & \text{caso contrário} \end{cases}$$

Coloração mínima

Como não há mais de uma cor por vértice, no pior caso precisaremos considerar o mesmo número de cores que de vértices. Logo, podemos definir uma variável binária para cada cor possível, indicando se ela é usada ou não na solução:

$$\mathbf{x_c} = \begin{cases} 1, & \text{se a cor } c \text{ é usada} \\ 0, & \text{caso contrário} \end{cases}$$

Ademais podemos identificar qual cor foi usada por cada vértice com as seguintes variáveis binárias:

Coloração mínima

Como não há mais de uma cor por vértice, no pior caso precisaremos considerar o mesmo número de cores que de vértices. Logo, podemos definir uma variável binária para cada cor possível, indicando se ela é usada ou não na solução:

$$\mathbf{x_c} = \begin{cases} 1, & \text{se a cor } c \text{ é usada} \\ 0, & \text{caso contrário} \end{cases}$$

Ademais podemos identificar qual cor foi usada por cada vértice com as seguintes variáveis binárias:

$$\mathbf{y_{vc}} = \begin{cases} 1, & \text{se o vértice } v \text{ usa a cor } c \\ 0, & \text{em outro caso} \end{cases}$$

Coloração mínima

O objetivo é minimizar o número de cores:

Coloração mínima

O objetivo é minimizar o número de cores:

$$\min \quad \sum_{c \in C} \mathbf{x_c}$$

Coloração mínima

Cada vértice precisa ter uma cor:

Coloração mínima

Cada vértice precisa ter uma cor:

$$\sum_{c \in C} \mathbf{y_{vc}} = 1 \quad \forall v \in V$$

Coloração mínima

Cada vértice precisa ter uma cor:

$$\sum_{c \in C} \mathbf{y_{vc}} = 1 \quad \forall v \in V$$

Dois vértices adjacentes não podem ter a mesma cor:

Coloração mínima

Cada vértice precisa ter uma cor:

$$\sum_{c \in C} \mathbf{y_{vc}} = 1 \quad \forall v \in V$$

Dois vértices adjacentes não podem ter a mesma cor:

$$\mathbf{y_{uc}} + \mathbf{y_{vc}} \leq 1 \quad \forall c \in C, (u, v) \in E$$

Coloração mínima

Cada vértice precisa ter uma cor:

$$\sum_{c \in C} \mathbf{y_{vc}} = 1 \quad \forall v \in V$$

Dois vértices adjacentes não podem ter a mesma cor:

$$\mathbf{y_{uc}} + \mathbf{y_{vc}} \leq 1 \quad \forall c \in C, (u, v) \in E$$

Se uma cor foi usada em um vértice, então a variável dela deve ser 1:

Coloração mínima

Cada vértice precisa ter uma cor:

$$\sum_{c \in C} \mathbf{y_{vc}} = 1 \quad \forall v \in V$$

Dois vértices adjacentes não podem ter a mesma cor:

$$\mathbf{y_{uc}} + \mathbf{y_{vc}} \leq 1 \quad \forall c \in C, (u, v) \in E$$

Se uma cor foi usada em um vértice, então a variável dela deve ser 1:

$$\mathbf{x_c} \geq \mathbf{y_{vc}} \qquad \forall c \in C, v \in V$$

Coloração mínima

Cada vértice precisa ter uma cor:

$$\sum_{c \in C} \mathbf{y_{vc}} = 1 \quad \forall v \in V$$

Dois vértices adjacentes não podem ter a mesma cor:

$$\mathbf{y_{uc}} + \mathbf{y_{vc}} \leq 1 \quad \forall c \in C, (u, v) \in E$$

Se uma cor foi usada em um vértice, então a variável dela deve ser 1:

$$\mathbf{x_c} \geq \mathbf{y_{vc}} \qquad \forall c \in C, v \in V$$

$$\mathbf{x_c} \leq \sum_{v \in V} \mathbf{y_{vc}} \qquad \forall c \in C$$

Coloração mínima

Instância: $G = (V, E)$

Coloração mínima

Instância: $G = (V, E)$
Defina $C = \{1, 2, \ldots, |V|\}$

Coloração mínima

Instância: $G = (V, E)$
Defina $C = \{1, 2, \ldots, |V|\}$

$$\min \quad \sum_{c \in C} \mathbf{x_c}$$

Coloração mínima

Instância: $G = (V, E)$
Defina $C = \{1, 2, \ldots, |V|\}$

$$\begin{array}{ll} \min & \sum_{c \in C} \mathbf{x_c} \\ s.a: & \end{array}$$

Coloração mínima

Instância: $G = (V, E)$
Defina $C = \{1, 2, \ldots, |V|\}$

$$
\begin{array}{ll}
\min & \sum_{c \in C} \mathbf{x_c} \\
s.a: & \\
& \sum_{c \in C} \mathbf{y_{vc}} = 1 \quad \forall v \in V
\end{array}
$$

Coloração mínima

Instância: $G = (V, E)$
Defina $C = \{1, 2, \dots, |V|\}$

$$
\begin{array}{ll}
\min & \sum_{c \in C} \mathbf{x_c} \\
s.a: & \\
& \sum_{c \in C} \mathbf{y_{vc}} = 1 \quad \forall v \in V \\
& \mathbf{y_{uc}} + \mathbf{y_{vc}} \leq 1 \quad \forall c \in C, (u, v) \in E
\end{array}
$$

Coloração mínima

Instância: $G = (V, E)$
Defina $C = \{1, 2, \ldots, |V|\}$

$$
\begin{array}{lll}
\min & \sum_{c \in C} \mathbf{x_c} & \\
s.a: & & \\
& \sum_{c \in C} \mathbf{y_{vc}} = 1 & \forall v \in V \\
& \mathbf{y_{uc}} + \mathbf{y_{vc}} \leq 1 & \forall c \in C, (u, v) \in E \\
& \mathbf{x_c} \geq \mathbf{y_{vc}} & \forall c \in C, v \in V
\end{array}
$$

Coloração mínima

Instância: $G = (V, E)$
Defina $C = \{1, 2, \ldots, |V|\}$

$$
\begin{array}{lll}
\min & \sum_{c \in C} \mathbf{x_c} & \\
s.a: & & \\
 & \sum_{c \in C} \mathbf{y_{vc}} = 1 & \forall v \in V \\
 & \mathbf{y_{uc}} + \mathbf{y_{vc}} \leq 1 & \forall c \in C, (u, v) \in E \\
 & \mathbf{x_c} \geq \mathbf{y_{vc}} & \forall c \in C, v \in V \\
 & \mathbf{x_c} \leq \sum_{v \in V} \mathbf{y_{vc}} & \forall c \in C
\end{array}
$$

Coloração mínima

Instância: $G = (V, E)$
Defina $C = \{1, 2, \ldots, |V|\}$

$$
\begin{array}{lll}
\min & \sum_{c \in C} \mathbf{x_c} & \\
s.a: & & \\
& \sum_{c \in C} \mathbf{y_{vc}} = 1 & \forall v \in V \\
& \mathbf{y_{uc}} + \mathbf{y_{vc}} \leq 1 & \forall c \in C, (u, v) \in E \\
& \mathbf{x_c} \geq \mathbf{y_{vc}} & \forall c \in C, v \in V \\
& \mathbf{x_c} \leq \sum_{v \in V} \mathbf{y_{vc}} & \forall c \in C \\
& \mathbf{x_c}, \mathbf{y_{vc}} \in \{0, 1\} & \forall c \in C, v \in V
\end{array}
$$

Caminho mais curto

Dados um digrafo ponderado G e dois vértices s e t, encontrar um caminho de peso mínimo entre s e t

Caminho mais curto

Dados um digrafo ponderado G e dois vértices s e t, encontrar um caminho de peso mínimo entre s e t

Um caminho pode ser visto como uma sequência de arcos. Assim a solução pode ter variáveis binárias associados aos arcos que indiquem se este foi selecionado ou não:

Caminho mais curto

Dados um digrafo ponderado G e dois vértices s e t, encontrar um caminho de peso mínimo entre s e t

Um caminho pode ser visto como uma sequência de arcos. Assim a solução pode ter variáveis binárias associados aos arcos que indiquem se este foi selecionado ou não:

$$\mathbf{x_a} = \begin{cases} 1, & \text{se o arco } a \text{ pertence à solução} \\ 0, & \text{em outro caso} \end{cases}$$

Caminho mais curto

O objetivo é minimizar a soma dos pesos dos arcos:

Caminho mais curto

O objetivo é minimizar a soma dos pesos dos arcos:

$$\min \quad \sum_{a \in A} \ell(a) \mathbf{x_a}$$

Caminho mais curto

Com exceção de s e t, qualquer vértice tem que ter o mesmo número de arcos e saindo:

Caminho mais curto

Com exceção de s e t, qualquer vértice tem que ter o mesmo número de arcos e saindo:

$$\sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0$$

Caminho mais curto

Com exceção de s e t, qualquer vértice tem que ter o mesmo número de arcos e saindo:

$$\sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0$$

No caso de s a diferença de arcos entrando e saindo deve ser 1, enquanto para t o valor deve ser -1:

Caminho mais curto

Com exceção de s e t, qualquer vértice tem que ter o mesmo número de arcos e saindo:

$$\sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0$$

No caso de s a diferença de arcos entrando e saindo deve ser 1, enquanto para t o valor deve ser -1:

$$\sum_{a \in \delta^+(s)} \mathbf{x_a} - \sum_{a \in \delta^-(s)} \mathbf{x_a} = 1$$

Caminho mais curto

Com exceção de s e t, qualquer vértice tem que ter o mesmo número de arcos e saindo:

$$\sum_{a\in\delta^+(v)} \mathbf{x_a} - \sum_{a\in\delta^-(v)} \mathbf{x_a} = 0$$

No caso de s a diferença de arcos entrando e saindo deve ser 1, enquanto para t o valor deve ser -1:

$$\sum_{a\in\delta^+(s)} \mathbf{x_a} - \sum_{a\in\delta^-(s)} \mathbf{x_a} = 1 \qquad \sum_{a\in\delta^+(t)} \mathbf{x_a} - \sum_{a\in\delta^-(t)} \mathbf{x_a} = -1$$

Caminho mais curto

Com exceção de s e t, qualquer vértice tem que ter o mesmo número de arcos e saindo:

$$\sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0$$

No caso de s a diferença de arcos entrando e saindo deve ser 1, enquanto para t o valor deve ser -1:

$$\sum_{a \in \delta^+(s)} \mathbf{x_a} - \sum_{a \in \delta^-(s)} \mathbf{x_a} = 1 \qquad \sum_{a \in \delta^+(t)} \mathbf{x_a} - \sum_{a \in \delta^-(t)} \mathbf{x_a} = -1$$

Finalmente, para cada vértice não deve haver mais do que um arco saindo:

Caminho mais curto

Com exceção de s e t, qualquer vértice tem que ter o mesmo número de arcos e saindo:

$$\sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0$$

No caso de s a diferença de arcos entrando e saindo deve ser 1, enquanto para t o valor deve ser -1:

$$\sum_{a \in \delta^+(s)} \mathbf{x_a} - \sum_{a \in \delta^-(s)} \mathbf{x_a} = 1 \qquad \sum_{a \in \delta^+(t)} \mathbf{x_a} - \sum_{a \in \delta^-(t)} \mathbf{x_a} = -1$$

Finalmente, para cada vértice não deve haver mais do que um arco saindo:

$$\sum_{a \in \delta^+(v)} \mathbf{x_a} \leq 1$$

Caminho mais curto

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

Caminho mais curto

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$\min \quad \sum_{a \in A} \ell(a) \mathbf{x_a}$$

Caminho mais curto

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{ll}
\min & \sum_{a \in A} \ell(a) \mathbf{x_a} \\
s.a: &
\end{array}
$$

Caminho mais curto

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{ll}
\min & \sum_{a \in A} \ell(a) \mathbf{x_a} \\
s.a: & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0 \quad \forall v \in V \setminus \{s, t\}
\end{array}
$$

Caminho mais curto

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lll}
\min & \sum_{a \in A} \ell(a) \mathbf{x_a} & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} \mathbf{x_a} - \sum_{a \in \delta^-(s)} \mathbf{x_a} = 1 &
\end{array}
$$

Caminho mais curto

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lll}
\min & \sum_{a \in A} \ell(a) x_a & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a = 1 & \\
& \sum_{a \in \delta^+(t)} x_a - \sum_{a \in \delta^-(t)} x_a = -1 &
\end{array}
$$

Caminho mais curto

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lrl}
\min & \sum_{a \in A} \ell(a) \mathbf{x_a} & \\
s.a: & & \\
 & \sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0 & \forall v \in V \setminus \{s, t\} \\
 & \sum_{a \in \delta^+(s)} \mathbf{x_a} - \sum_{a \in \delta^-(s)} \mathbf{x_a} = 1 & \\
 & \sum_{a \in \delta^+(t)} \mathbf{x_a} - \sum_{a \in \delta^-(t)} \mathbf{x_a} = -1 & \\
 & \sum_{a \in \delta^+(v)} \mathbf{x_a} \leq 1 & \forall v \in V
\end{array}
$$

Caminho mais curto

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lrl}
\min & \sum_{a \in A} \ell(a) x_a & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a = 1 & \\
& \sum_{a \in \delta^+(t)} x_a - \sum_{a \in \delta^-(t)} x_a = -1 & \\
& \sum_{a \in \delta^+(v)} x_a \leq 1 & \forall v \in V \\
& x_a \in \{0, 1\} & \forall a \in A
\end{array}
$$

Caminho mais curtos e ciclos negativos

Uma estratégia para evitar ciclos negativos em uma solução é a **PREVENÇÃO DE SUB-ROTAS**, que tem como base a propriedade de que para qualquer caminho o número de arcos com os dois extremos em um mesmo conjunto de vértices S é no máximo $|S| - 1$

Caminho mais curtos e ciclos negativos

Uma estratégia para evitar ciclos negativos em uma solução é a **PREVENÇÃO DE SUB-ROTAS**, que tem como base a propriedade de que para qualquer caminho o número de arcos com os dois extremos em um mesmo conjunto de vértices S é no máximo $|S| - 1$

Se $A(S)$ denota o número com dois extremos em S, podemos definir as seguintes restrições:

Caminho mais curtos e ciclos negativos

Uma estratégia para evitar ciclos negativos em uma solução é a **PREVENÇÃO DE SUB-ROTAS**, que tem como base a propriedade de que para qualquer caminho o número de arcos com os dois extremos em um mesmo conjunto de vértices S é no máximo $|S| - 1$

Se $A(S)$ denota o número com dois extremos em S, podemos definir as seguintes restrições:

$$\sum_{a \in A(S)} \mathbf{x_a} \leq |S| - 1 \qquad \forall S \subseteq V \setminus \{s, t\} \text{ e } |S| \geq 2$$

Caminho mais curto e ciclos negativos

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

Caminho mais curto e ciclos negativos

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

min $\qquad \sum_{a \in A} \ell(a)\mathbf{x_a}$

Caminho mais curto e ciclos negativos

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{ll}
\min & \sum_{a \in A} \ell(a) \mathbf{x_a} \\
s.a: &
\end{array}
$$

Caminho mais curto e ciclos negativos

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lr}
\min & \sum_{a \in A} \ell(a) \mathbf{x_a} \\
s.a: & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0 \quad \forall v \in V \setminus \{s, t\}
\end{array}
$$

Caminho mais curto e ciclos negativos

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lll}
\min & \sum_{a \in A} \ell(a) \mathbf{x_a} & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} \mathbf{x_a} - \sum_{a \in \delta^-(s)} \mathbf{x_a} = 1 &
\end{array}
$$

Caminho mais curto e ciclos negativos

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lrl}
\min & & \sum_{a \in A} \ell(a) x_a \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a = 1 & \\
& \sum_{a \in \delta^+(t)} x_a - \sum_{a \in \delta^-(t)} x_a = -1 &
\end{array}
$$

Caminho mais curto e ciclos negativos

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lrl}
\min & \sum_{a \in A} \ell(a) \mathbf{x_a} & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} \mathbf{x_a} - \sum_{a \in \delta^-(s)} \mathbf{x_a} = 1 & \\
& \sum_{a \in \delta^+(t)} \mathbf{x_a} - \sum_{a \in \delta^-(t)} \mathbf{x_a} = -1 & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} \leq 1 & \forall v \in V
\end{array}
$$

Caminho mais curto e ciclos negativos

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{llr}
\min & \sum_{a\in A} \ell(a)\mathbf{x_a} & \\
s.a: & & \\
& \sum_{a\in\delta^+(v)} \mathbf{x_a} - \sum_{a\in\delta^-(v)} \mathbf{x_a} = 0 & \forall v \in V \setminus \{s,t\} \\
& \sum_{a\in\delta^+(s)} \mathbf{x_a} - \sum_{a\in\delta^-(s)} \mathbf{x_a} = 1 & \\
& \sum_{a\in\delta^+(t)} \mathbf{x_a} - \sum_{a\in\delta^-(t)} \mathbf{x_a} = -1 & \\
& \sum_{a\in\delta^+(v)} \mathbf{x_a} \le 1 & \forall v \in V \\
& \sum_{a\in A(S)} \mathbf{x_a} \le |S| - 1 & \forall S \subseteq V \setminus \{s,t\}, |S| \ge 2
\end{array}
$$

Caminho mais curto e ciclos negativos

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lrl}
\min & \sum_{a \in A} \ell(a) \mathbf{x_a} & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} \mathbf{x_a} - \sum_{a \in \delta^-(s)} \mathbf{x_a} = 1 & \\
& \sum_{a \in \delta^+(t)} \mathbf{x_a} - \sum_{a \in \delta^-(t)} \mathbf{x_a} = -1 & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} \leq 1 & \forall v \in V \\
& \sum_{a \in A(S)} \mathbf{x_a} \leq |S| - 1 & \forall S \subseteq V \setminus \{s, t\}, |S| \geq 2 \\
& \mathbf{x_a} \in \{0, 1\} & \forall a \in A
\end{array}
$$

Caminho mais longo

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

Caminho mais longo

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

max $\sum_{a \in A} \ell(a) \mathbf{x_a}$

Caminho mais longo

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$\begin{array}{ll} \max & \sum_{a \in A} \ell(a) \mathbf{x_a} \\ s.a: & \end{array}$$

Caminho mais longo

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{ll}
\max & \sum_{a \in A} \ell(a) \mathbf{x_a} \\
s.a: & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0 \quad \forall v \in V \setminus \{s, t\}
\end{array}
$$

Caminho mais longo

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lll}
\max & \sum_{a \in A} \ell(a) \mathbf{x_a} & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} \mathbf{x_a} - \sum_{a \in \delta^-(v)} \mathbf{x_a} = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} \mathbf{x_a} - \sum_{a \in \delta^-(s)} \mathbf{x_a} = 1 &
\end{array}
$$

Caminho mais longo

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lll}
\max & \sum_{a \in A} \ell(a) x_a & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a = 1 & \\
& \sum_{a \in \delta^+(t)} x_a - \sum_{a \in \delta^-(t)} x_a = -1 &
\end{array}
$$

Caminho mais longo

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lll}
\max & \sum_{a \in A} \ell(a) x_a & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a = 1 & \\
& \sum_{a \in \delta^+(t)} x_a - \sum_{a \in \delta^-(t)} x_a = -1 & \\
& \sum_{a \in \delta^+(v)} x_a \leq 1 & \forall v \in V
\end{array}
$$

Caminho mais longo

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lll}
\max & \sum_{a\in A} \ell(a) \mathbf{x_a} & \\
s.a: & & \\
& \sum_{a\in\delta^+(v)} \mathbf{x_a} - \sum_{a\in\delta^-(v)} \mathbf{x_a} = 0 & \forall v \in V \setminus \{s,t\} \\
& \sum_{a\in\delta^+(s)} \mathbf{x_a} - \sum_{a\in\delta^-(s)} \mathbf{x_a} = 1 & \\
& \sum_{a\in\delta^+(t)} \mathbf{x_a} - \sum_{a\in\delta^-(t)} \mathbf{x_a} = -1 & \\
& \sum_{a\in\delta^+(v)} \mathbf{x_a} \leq 1 & \forall v \in V \\
& \sum_{a\in A(S)} \mathbf{x_a} \leq |S| - 1 & \forall S \subseteq V \setminus \{s,t\}, |S| \geq 2
\end{array}
$$

Caminho mais longo

Instância: $G = (V, A), \ell : A \to \mathbb{Q}_+$ e $s, t \in V$.

$$
\begin{array}{lll}
\max & \sum_{a \in A} \ell(a) x_a & \\
s.a: & & \\
& \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 & \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a = 1 & \\
& \sum_{a \in \delta^+(t)} x_a - \sum_{a \in \delta^-(t)} x_a = -1 & \\
& \sum_{a \in \delta^+(v)} x_a \leq 1 & \forall v \in V \\
& \sum_{a \in A(S)} x_a \leq |S| - 1 & \forall S \subseteq V \setminus \{s, t\}, |S| \geq 2 \\
& x_a \in \{0, 1\} & \forall a \in A
\end{array}
$$

Árvore geradora mínima

Semelhante aos caminhos, uma árvore pode ser vista como um conjunto de arestas, assim podemos definir variáveis binárias associadas às arestas que determinam se a aresta pertence ou não à árvore:

Árvore geradora mínima

Semelhante aos caminhos, uma árvore pode ser vista como um conjunto de arestas, assim podemos definir variáveis binárias associadas às arestas que determinam se a aresta pertence ou não à árvore:

$$\mathbf{x_e} = \begin{cases} 1, & \text{se a aresta } e \text{ está na árvore} \\ 0, & \text{em outro caso} \end{cases}$$

Árvore geradora mínima

O objetivo é minimizar o custo das arestas selecionadas:

Árvore geradora mínima

O objetivo é minimizar o custo das arestas selecionadas:

$$\min \quad \sum_{e \in E} \omega(e) \mathbf{x_e}$$

Árvore geradora mínima

Árvores devem ser conexas e não ter ciclos:

Árvore geradora mínima

Árvores devem ser conexas e não ter ciclos:

- A conexidade pode ser garantida se para cada subconjunto de vértices $S \subset V$ há pelo menos uma aresta em S e outro fora:

Árvore geradora mínima

Árvores devem ser conexas e não ter ciclos:

- A conexidade pode ser garantida se para cada subconjunto de vértices $S \subset V$ há pelo menos uma aresta em S e outro fora:

$$\sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 \qquad \forall \emptyset \subset S \subset V$$

Árvore geradora mínima

Árvores devem ser conexas e não ter ciclos:

- A conexidade pode ser garantida se para cada subconjunto de vértices $S \subset V$ há pelo menos uma aresta em S e outro fora:

$$\sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 \qquad \forall \emptyset \subset S \subset V$$

- Se $E(S)$ é o conjunto de arestas com os dois extremos em S, então para garanti que a árvore seja acíclica, não deve haver menos arestas em $E(S)$ do que vértices em S:

Árvore geradora mínima

Árvores devem ser conexas e não ter ciclos:

- A conexidade pode ser garantida se para cada subconjunto de vértices $S \subset V$ há pelo menos uma aresta em S e outro fora:

$$\sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 \qquad \forall \emptyset \subset S \subset V$$

- Se $E(S)$ é o conjunto de arestas com os dois extremos em S, então para garanti que a árvore seja acíclica, não deve haver menos arestas em $E(S)$ do que vértices em S:

$$\sum_{e \in E(S)} \mathbf{x_e} \leq |S| - 1 \qquad \forall S \subseteq V, |S| \geq 2$$

Árvore geradora mínima

Definições equivalentes de árvores:

Árvore geradora mínima

Definições equivalentes de árvores:

- Um grafo conexo com $|V| - 1$ arestas.

Árvore geradora mínima

Definições equivalentes de árvores:

- Um grafo conexo com $|V| - 1$ arestas.
- Um grafo acíclico com $|V| - 1$ arestas.

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

min $\qquad \sum_{e \in E} \omega(e) \mathbf{x_e}$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$\begin{array}{ll} \text{min} & \sum_{e \in E} \omega(e) \mathbf{x_e} \\ s.a: & \end{array}$$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$
\begin{array}{ll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} \\
s.a: & \\
& \sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 \qquad \forall \emptyset \subset S \subset V
\end{array}
$$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$
\begin{array}{ll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} \\
s.a: & \\
& \sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 \qquad \forall \emptyset \subset S \subset V \\
& \sum_{e \in E} \mathbf{x_e} = |V| - 1
\end{array}
$$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$
\begin{array}{lll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} & \\
s.a: & & \\
& \sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 & \forall \emptyset \subset S \subset V \\
& \sum_{e \in E} \mathbf{x_e} = |V| - 1 & \\
& \mathbf{x_v} \in \{0, 1\} & \forall v \in V
\end{array}
$$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$
\begin{array}{lll|}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} & \\
s.a: & & \\
& \sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 & \forall \emptyset \subset S \subset V \\
& \sum_{e \in E} \mathbf{x_e} = |V| - 1 & \\
& \mathbf{x_v} \in \{0, 1\} & \forall v \in V
\end{array}
$$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$
\begin{array}{lll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} & \\
s.a: & & \\
& \sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 & \forall \emptyset \subset S \subset V \\
& \sum_{e \in E} \mathbf{x_e} = |V| - 1 & \\
& \mathbf{x_v} \in \{0, 1\} & \forall v \in V
\end{array}
$$

$$
\begin{array}{ll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e}
\end{array}
$$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$
\begin{array}{lll}
\min & \sum_{e \in E} \omega(e) x_e & \\
s.a: & & \\
& \sum_{e \in \delta(S)} x_e \geq 1 & \forall \emptyset \subset S \subset V \\
& \sum_{e \in E} x_e = |V| - 1 & \\
& x_v \in \{0, 1\} & \forall v \in V
\end{array}
$$

$$
\begin{array}{ll}
\min & \sum_{e \in E} \omega(e) x_e \\
s.a: &
\end{array}
$$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$
\begin{array}{lll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} & \\
s.a: & & \\
& \sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 & \forall \emptyset \subset S \subset V \\
& \sum_{e \in E} \mathbf{x_e} = |V| - 1 & \\
& \mathbf{x_v} \in \{0, 1\} & \forall v \in V
\end{array}
$$

$$
\begin{array}{lll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} & \\
s.a: & & \\
& \sum_{e \in E(S)} \mathbf{x_e} \leq |S| - 1 & \forall S \subseteq V, |S| \geq 2
\end{array}
$$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$
\begin{array}{lll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} & \\
s.a: & & \\
& \sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 & \forall \emptyset \subset S \subset V \\
& \sum_{e \in E} \mathbf{x_e} = |V| - 1 & \\
& \mathbf{x_v} \in \{0, 1\} & \forall v \in V
\end{array}
$$

$$
\begin{array}{lll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} & \\
s.a: & & \\
& \sum_{e \in E(S)} \mathbf{x_e} \leq |S| - 1 & \forall S \subseteq V, |S| \geq 2 \\
& \sum_{e \in E} \mathbf{x_e} = |V| - 1 &
\end{array}
$$

Árvore geradora mínima

Instância: $G = (V, E)$, $\omega : E \to \mathbb{Q}$

$$
\begin{array}{lll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} & \\
s.a: & & \\
& \sum_{e \in \delta(S)} \mathbf{x_e} \geq 1 & \forall \emptyset \subset S \subset V \\
& \sum_{e \in E} \mathbf{x_e} = |V| - 1 & \\
& \mathbf{x_v} \in \{0, 1\} & \forall v \in V
\end{array}
$$

$$
\begin{array}{lll}
\min & \sum_{e \in E} \omega(e) \mathbf{x_e} & \\
s.a: & & \\
& \sum_{e \in E(S)} \mathbf{x_e} \leq |S| - 1 & \forall S \subseteq V, |S| \geq 2 \\
& \sum_{e \in E} \mathbf{x_e} = |V| - 1 & \\
& \mathbf{x_v} \in \{0, 1\} & \forall v \in V
\end{array}
$$

Comentários finais sobre formulações

Passos para formular

1. Selecionar um conjunto de variáveis de decisão

Passos para formular

1. Selecionar um conjunto de variáveis de decisão

2. Escrever a função objetivo.

Passos para formular

1. Selecionar um conjunto de variáveis de decisão
2. Escrever a função objetivo.
3. Escrever as restrições.

Passos para formular

1. Selecionar um conjunto de variáveis de decisão
2. Escrever a função objetivo.
3. Escrever as restrições.

Observação. Se for mais simples escrever a função objetivo ou as restrições como funções não-lineares ou expressões lógicas, então faça isso primeiro, depois tente linearizá-las. Também, caso veja a necessidade de adicionar novas variáveis, adicione-as.

Considerações

- Existem muitas formas de formular um problema.

Considerações

- Existem muitas formas de formular um problema.
- As estratégias que existem na atualidade para solucionar problemas escritos em programação linear inteiras, são muito sensíveis à formulação proposta::

Considerações

- Existem muitas formas de formular um problema.

- As estratégias que existem na atualidade para solucionar problemas escritos em programação linear inteiras, são muito sensíveis à formulação proposta::
  - Alguns programas lineares inteiros são mais fáceis de solucionar mesmo tendo milhões de variáveis ou restrições.

Considerações

- Existem muitas formas de formular um problema.
- As estratégias que existem na atualidade para solucionar problemas escritos em programação linear inteiras, são muito sensíveis à formulação proposta::
  - Alguns programas lineares inteiros são mais fáceis de solucionar mesmo tendo milhões de variáveis ou restrições.
  - Outros são muito difíceis, inclusive quando o número de variáveis e restrições não ultrapassa as centenas.

Considerações

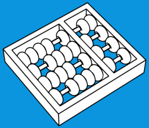
- Existem muitas formas de formular um problema.
- As estratégias que existem na atualidade para solucionar problemas escritos em programação linear inteiras, são muito sensíveis à formulação proposta::
  - Alguns programas lineares inteiros são mais fáceis de solucionar mesmo tendo milhões de variáveis ou restrições.
  - Outros são muito difíceis, inclusive quando o número de variáveis e restrições não ultrapassa as centenas.
  - Requer experiência para identificar qual é qual e usualmente isso não é uma tarefa simples.

Problemas NP-completos e programação linear inteira

MC558 - Projeto e Análise de Algoritmos II

13

01/24



Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br