Ash 20

Programação Linear - Revisitar a 1ª parte da cadeira

- Fluxo Máximo

$$\max \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$f_{uv} \leq c(u,v) \quad \text{for each } u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv} \quad \text{for each } u \in V \setminus \{s,t\}$$

$$f_{uv} \geq 0 \,, \quad \text{for each } u, v \in V$$

- Caminhos mais curtos entre s e t

$$\max d[t]$$

$$d[v] \leq d[u] + w(u,v) \quad \forall (u,v) \in E$$

$$d[s] = 0$$

$$d[v] \geq 0 \quad \forall v \in V$$

Programação Linear - Revisitar a 1ª parte da cadeira

- Fluxo Máximo

$$\max \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$f_{uv} \leq c(u,v) \quad \text{for each } u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv} \quad \text{for each } u \in V \setminus \{s,t\}$$

$$f_{uv} \geq 0 \ , \ \text{for each } u, v \in V$$

- Caminhos mais curtos entre s e t

$$\max d[t]$$

$$d[v] \leq d[u] + w(u,v) \quad \forall (u,v) \in E$$

$$d[s] = 0$$

$$d[v] \geq 0 \quad \forall v \in V$$

Dualidade

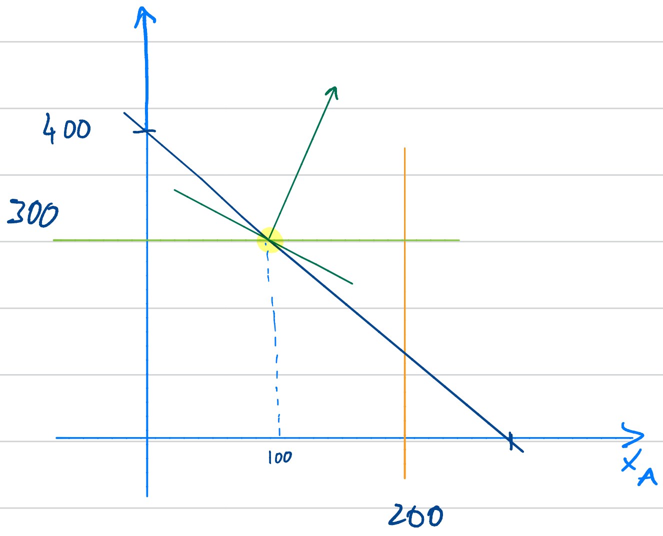
$\max \; x_A + 6x_B$

$x_A \leq 200$ (I)

$x_B \leq 300$ (II)

$x_A + x_B \leq 400$ (III)

$x_A, x_B \geq 0$



Pergunta: Conseguimos encontrar um upper bound para o valor da função objectivo olhando para as restrições (I) e (II) ?

Dualidade

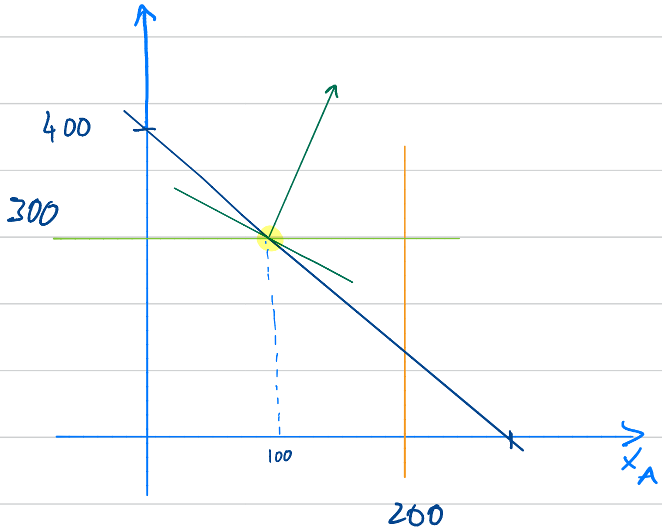
$$\max \ x_A + 6x_B$$

$x_A \leq 200$ (I)

$x_B \leq 300$ (II)

$x_A + x_B \leq 400$ (III)

$x_A, x_B \geq 0$



Pergunta: Conseguimos encontrar um upper bound para o valor da função objectivo olhando para as restrições (I) e (II)?

- $1 \times (I) + 6 \times (II) \Leftrightarrow x_A + 6x_B \leq 200 + 6 \times 300$

$\Leftrightarrow x_A + 6x_B \leq 2000$

A função objectivo nunca pode ter um valor superior a 2000.

→ Este upper bound não é apertado (o máximo é 1900).

Conseguimos uma combinação melhor?

Dualidade

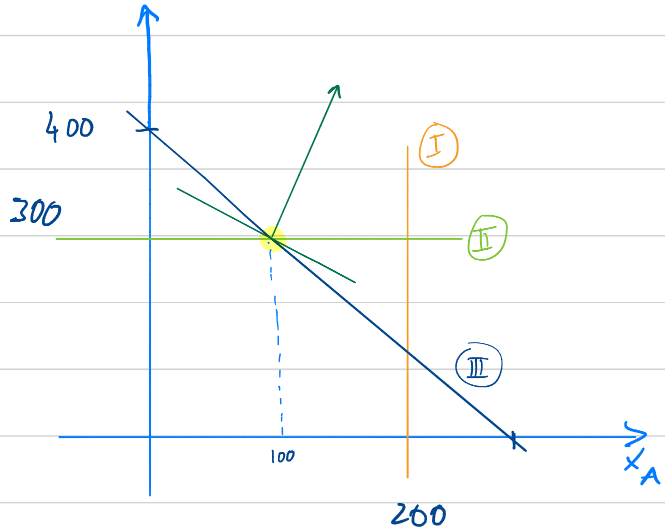
$\max \; x_A + 6x_B$

$x_A \le 200$ I

$x_B \le 300$ II

$x_A + x_B \le 400$ III

$x_A, x_B \ge 0$



Pergunta: Conseguimos encontrar um upper bound mais apertado usando as restrições II e III ?

Dualidade

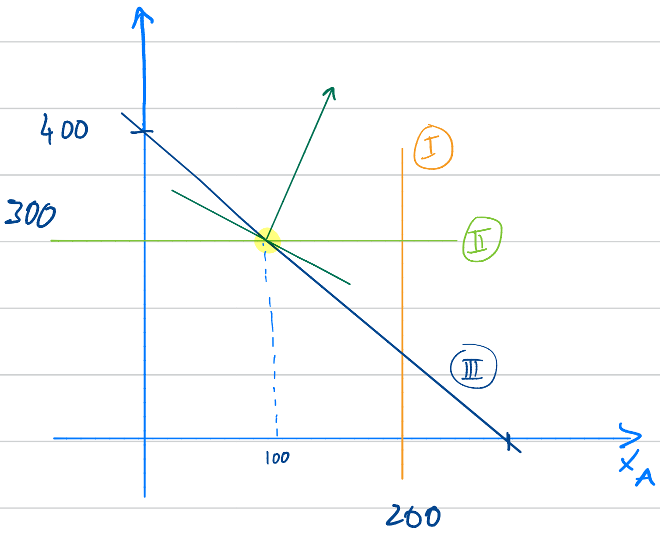
$\max \; x_A + 6x_B$

$x_A \le 200$ (I)

$x_B \le 300$ (II)

$x_A + x_B \le 400$ (III)

$x_A, x_B \ge 0$



- $1 \times \text{(I)} + 6 \times \text{(II)} \Leftrightarrow x_A + 6x_B \le 200 + 6 \times 300$

  $\Leftrightarrow x_A + 6x_B \le 2000$ A função objectivo nunca pode ter um valor superior a 2000.

- $5 \times \text{(II)} + 1 \times \text{(III)} \Leftrightarrow 5 \times x_B + x_A + x_B \le 5 \times 300 + 400$

  $\Leftrightarrow x_A + 6x_B \le 1900$ Este upper bound é apertado!

Dualidade - Sistematização

- Queremos encontrar o <u>mais pequeno</u> upper bound para a função objectivo

$$\begin{array}{lrl} & \lambda_I \times & x_A \leq 200 \\ & \lambda_{II} \times & x_B \leq 300 \\ + & \lambda_{III} \times & x_A + x_B \leq 400 \\ \hline \end{array}$$

Função objectivo : $x_A + 6x_B$

$$(\lambda_I + \lambda_{III})\, x_A + (\lambda_{II} + \lambda_{III}) \times x_B \leq \lambda_I \times 200 + \lambda_{II} \times 300 + \lambda_{III} \times 400$$

Dualidade - Sistematização

- Queremos encontrar o <u>mais pequeno</u> upper bound para a função objectivo

$$\begin{aligned} \lambda_I \times \quad & x_A \leq 200 \\ \lambda_{II} \times \quad & x_B \leq 300 \\ \lambda_{III} \times \quad & x_A + x_B \leq 400 \\ + \quad & \end{aligned}$$

Função objectivo: $x_A + 6x_B$

$$(\lambda_I + \lambda_{III})\, x_A + (\lambda_{II} + \lambda_{III}) \times x_B \leq \lambda_I \times 200 + \lambda_{II} \times 300 + \lambda_{III} \times 400$$

$$\left.\begin{aligned} \min \quad & 200 \times \lambda_I + 300 \times \lambda_{II} + 400 \times \lambda_{III} \\ & \lambda_I + \lambda_{III} \geq 1 \\ & \lambda_{II} + \lambda_{III} \geq 6 \\ & \lambda_I, \lambda_{II}, \lambda_{III} \geq 0 \end{aligned}\right\} \text{Programa Dual}$$

$$(\lambda_I^*, \lambda_{II}^*, \lambda_{III}^*) = (0, 5, 1)$$

Dualidade - Sistematização

- Queremos encontrar o <u>mais pequeno</u> upper bound para a função objectivo

$$\lambda_I \times \quad x_A \leq 200$$
$$\lambda_{II} \times \quad x_B \leq 300$$
$$\lambda_{III} \times \quad x_A + x_B \leq 400$$
$$+$$

Função objectivo: $x_A + 6x_B$

$$(\lambda_I + \lambda_{III})\, x_A + (\lambda_{II} + \lambda_{III}) \times x_B \leq \lambda_I \times 200 + \lambda_{II} \times 300 + \lambda_{III} \times 400$$

$$\underbrace{\begin{array}{l} \min \quad 200 \times \lambda_I + 300 \times \lambda_{II} + 400 \times \lambda_{III} \\ \lambda_I + \lambda_{III} \geq 1 \\ \lambda_{II} + \lambda_{III} \geq 6 \\ \lambda_I, \lambda_{II}, \lambda_{III} \geq 0 \end{array}}_{\text{Programa Dual}}$$

$$\max \; c^T x \longleftrightarrow \min \; y^T b$$

$$Ax \leq b \qquad\qquad y^T A \geq c^T$$

$$x \geq 0 \qquad\qquad y \geq 0$$

Teorema da Dualidade Forte

Se qualquer um dos problemas tiver solução então o outro também tem e as soluções coincidem.

Dualidade - Sistematização

$$\max\ c^T x \longleftrightarrow \min\ y^T b$$

$$Ax \leq b \qquad y^T A \geq c^T$$

$$x \geq 0 \qquad y \geq 0$$

Lema [Dualidade Fraca]

Seja y^* a solução do problema dual e x^* a solução do problema primal, concluímos que:

$$c^T x^* \leq (y^*)^T b$$

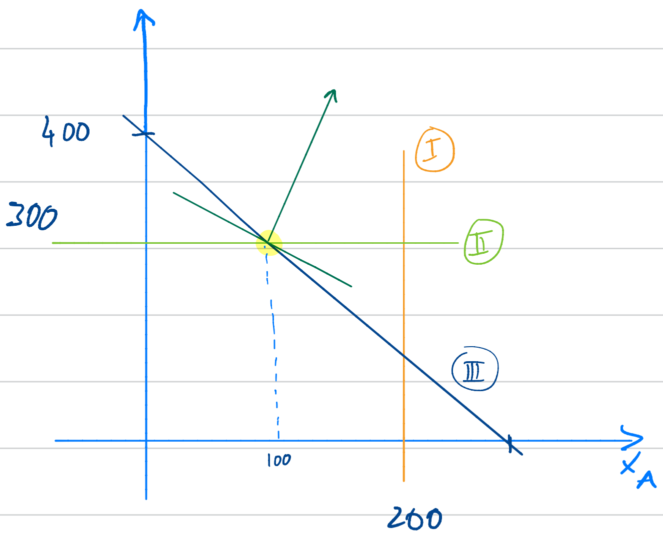
<u>Prova</u>

$$c^T x^* \leq (y^*)^T A x^*$$
$$\leq (y^*)^T b$$

Teorema da Dualidade Forte

Se qualquer um dos problemas tiver solução então o outro também tem e as soluções coincidem.

Dualidade - Sistematização



Primal

$$\max\ x_A + 6x_B$$

$x_A \leq 200$ — I

$x_B \leq 300$ — II

$x_A + x_B \leq 400$ — III

$x_A, x_B \geq 0$

Sol: 1900

Dual

$$\min\ 200y_1 + 300y_2 + 400y_3$$

$y_1 + y_3 \geq 1$

$y_2 + y_3 \geq 6$

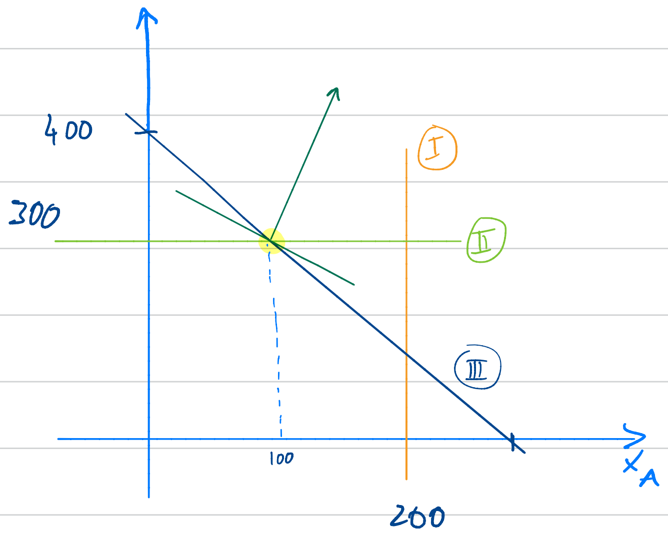
$y_1, y_2, y_3 \geq 0$

Sol*: 1900

Teorema da Dualidade Forte

Se qualquer um dos problemas tiver solução então o outro também tem e as soluções coincidem.

Dualidade - Sistematização



Primal

$\max \; x_A + 6x_B$

$x_A \leq 200$ (I)

$x_B \leq 300$ (II)

$x_A + x_B \leq 400$ (III)

$x_A, x_B \geq 0$

Sol: 1900

• $(100, 300)$

Dual

$\min \; 200y_1 + 300y_2 + 400y_3$

$y_1 + y_3 \geq 1$

$y_2 + y_3 \geq 6$

$y_1, y_2, y_3 \geq 0$

Sol*: 1900

$(0, 5, 1)$

• Determinar a solução do problema dual a partir do primal:

• $y_1 = 0 \hookrightarrow \begin{cases} y_3 = 1 \\ y_2 + y_3 = 6 \end{cases} \Leftrightarrow \begin{cases} y_3 = 1 \\ y_2 = 5 \end{cases}$

Colocar a 0 as variáveis q correspondem a restrições não activas do problema primal.

Lema 2 [Dualidade Fraca]

Seja x^* uma solução* do programa linear:

$$\max \; c^T x$$
$$Ax \leq b$$
$$x \geq 0$$

e y^* uma solução* do programa linear dual.

$$\min \; b^T y$$
$$y^T A \geq c^T$$
$$y \geq 0$$

Então: $c^T x^* \leq (y^*)^T A x^* \leq (y^*)^T b$

(*) Uma solução não necessariamente óptima.

Prova:

1. $A x^* \leq b$ (porque x^* é solução)
2. $(y^*)^T A x^* \leq (y^*)^T b$ (porque y^* é ≥ 0)
3. $(y^*)^T A \geq c^T$ (porque y^* é solução)
4. $(y^*)^T A x^* \geq c^T x^*$ (porque x^* é ≥ 0)
5. $c^T x^* \leq (y^*)^T A x^* \leq (y^*)^T b$

Lema 1 [Dual- dual = Primal]

O problema dual do problema dual é o problema primal.

Prova

$$\begin{array}{l} \max\ c^T x \\ Ax \le b \\ x \ge 0 \end{array} \quad \underset{(\text{dual})}{\Longrightarrow} \quad \begin{array}{l} \min\ y^T b \\ y^T A \ge c^T \\ y \ge 0 \end{array} \quad \underset{(\text{trans})}{\Longrightarrow} \quad \begin{array}{l} \max\ -y^T b \\ -y^T A \le -c^T \\ y \ge 0 \end{array}$$

$$\Downarrow \ (\text{trans})$$

$$\begin{array}{l} \min\ -(-c)^T x \\ -(-A^T)^T x \le -(-b^T)^T \\ y \ge 0 \end{array} \quad \underset{\text{dual}}{\Longleftarrow} \quad \begin{array}{l} \max\ -b^T \cdot y \\ -A^T y \le -c^T \\ y \ge 0 \end{array}$$

$\Uparrow$ (trans)

Caminhos mais curtos - Duas formulações alternativas

- Problema de Maximização:

$$\max \; d[t]$$

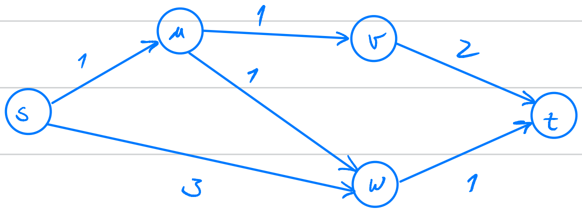
$$d[v] \leq d[u] + w(u,v) \quad \forall (u,v) \in E$$

$$d[s] = 0$$

$$d[v] \geq 0 \quad \forall v \in V$$

Conseguimos escrever o dual?

Caminhos mais curtos - Duas formulações alternativas



$$
\begin{array}{lrl}
 & \max & d_t \\
su & & d_u \le 1 \\
uv & & d_v \le d_u + 1 \\
uw & & d_w \le d_u + 1 \\
sw & & d_w \le 3 \\
vt & & d_t \le d_v + 2 \\
wt & & d_t \le d_w + 1 \\
 & & d_u, d_s, d_v, d_w, d_t \ge 0
\end{array}
$$

(reformulação) $\longrightarrow$

$$
\begin{array}{ll}
\max & d_t \\
 & d_u \le 1 \\
 & d_v - d_u \le 1 \\
 & d_w - d_u \le 1 \\
 & d_w \le 3 \\
 & d_t - d_v \le 2 \\
 & d_t - d_w \le 1 \\
 & d_u, d_s, d_v, d_w, d_t \ge 0
\end{array}
$$

dual

$$
\begin{array}{lll}
\min & x_{su} + x_{uv} + x_{uw} + 3x_{sw} + 2x_{vt} + x_{wt} & \\
 & x_{su} - x_{uv} - x_{uw} \ge 0 & (d_u) \\
 & x_{uv} - x_{vt} \ge 0 & (d_v) \\
 & x_{uw} + x_{sw} - x_{wt} \ge 0 & (d_w) \\
 & x_{vt} + x_{wt} \ge 1 & \\
 & x_{su}, x_{uv}, x_{uw}, x_{sw}, x_{vt}, &
\end{array}
$$

Caminhos mais curtos - Duas formulações alternativas

- Problema de Maximização:

$$\max \ d[t]$$

$$d[v] \leq d[u] + w(u,v) \quad \forall (u,v) \in E$$

$$d[s] = 0$$

$$d[v] \geq 0 \quad \forall v \in V$$

- <u>Problema dual</u>:

$$\min \sum_{(u,v)\in E} x_{uv} \cdot w(u,v)$$

$$\sum_{u \in V} x_{uv} - \sum_{w \in V} x_{vw} \geq 0 \ , \quad \text{para todo } v \notin \{s,t\}$$

$$\sum_{u \in V} x_{ut} - \sum_{u \in V} x_{tu} \geq 1$$

Algoritmo Simplex - Unboundedness

Exemplo: $\max \; 2x_1 + x_2$

$$\left.\begin{array}{ll} x_1 - x_2 \leq 10 & (1) \\ 2x_1 - x_2 \leq 40 & (2) \\ x_1, x_2 \geq 0 & (3), (4) \end{array}\right\} \text{Unbounded}$$

Dual:

$$\left.\begin{array}{l} \min \; 10y_1 + 40y_2 \\ y_1 + 2y_2 \geq 2 \\ -y_1 - y_2 \geq 1 \\ y_1, y_2 \geq 0 \end{array}\right\} \text{No Solution}$$

Algoritmo Simplex - Unboundedness

Exemplo: $\max \ 2x_1 + x_2$

$x_1 - x_2 \leq 10$ ①

$2x_1 - x_2 \leq 40$ ②

$x_1, x_2 \geq 0$ ③, ④

Dual:

$\min \ 10y_1 + 40y_2$

$y_1 + 2y_2 \geq 2$

$-y_1 - y_2 \geq 1$

$y_1, y_2 \geq 0$

Primal	Dual
• variáveis	• restrições
• restrições	• variáveis
• max	• min
• unbounded	• no solution
• No solution	• Unbounded

Complétude NP

Problemas de Decisão vs Problemas de Optimização

- Problemas de Decisão
  - Problemas cuja solução é sim/não
  - Exemplos: SAT, Primes, Euler Tour, Hamiltonian etc
  - Formalmente, um problema de decisão X corresponde ao conjunto das instâncias q satisfazem a condição do problema.

    Exemplo: $Primes = \{ n \in \mathbb{N} \mid \neg \exists m.\ m \neq 1 \wedge m \neq n \wedge m \mid n \}$

    $SAT = \{ \psi \mid$ existe valoração ρ tal que : $\rho(\psi) = 1 \}$

- Problemas de Optimização
  - Exemplos: Caminhos mais curtos, caminhos mais longos, fluxo máximo etc
  - Podem ser reformulados como problemas de decisão

    Exemplo: Caminhos mais curtos
    - $SPath = \{ <G, s, t, k> \mid s, t \in G.V$ e $G \vdash s \overset{k}{\leadsto} t \}$
    - $(G, s, t, k) \in SPath$ sse existe um caminho entre s e t em G com no máximo k arcos

Algoritmos de Decisão versus Algoritmos de Verificação

- Algoritmos de Decisão

O algoritmo A decide o problema $\mathcal{X}$ sse:

$$\forall x \in \Sigma.\ x \in \mathcal{X} \Leftrightarrow A(x) = 1$$

Exemplo:

$A_{SAT}(\varphi)$: decide se φ é satisfazível

- Algoritmos de Verificação

O algoritmo A certifica o problema $\mathcal{X}$ sse:

$$\forall x \in \Sigma.\ x \in \mathcal{X} \Leftrightarrow \exists y.\ A(x, \underbrace{y}_{\text{certificado}}) = 1$$

$A_{SAT}(\varphi, \rho)$: verifica se φ é satisfazível

Certificado: valoração ρ q̄ satisfaz φ $(\rho(\varphi) = 1)$

$A_{Composite}(n, (m_1, \dots, m_k))$: verifica se n é um nº composto

Certificado: lista de nºs cujo produto é n

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial

- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)

- Exp - conjunto dos problemas decidíveis em tempo exponencial

- co-NP - conjunto dos problemas cujo complemento está em NP

  Exemplo:

  UNSAT = $\{ \Phi \mid \Phi$ não é satisfazível $\}$

  - $X \in NP$ - conseguimos certificar a "pertença" a X em tempo polinomial

    Exemplo: SAT

  - $X \in$ co-NP - conseguimos certificar a "não-pertença" a X em tempo polinomial

    Exemplo: UNSAT

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)

Proposição 1: $P \subseteq NP$

- Suponhamos q $X \in P$, temos de provar que $X \in NP$.
- Como $X \in P$, concluímos que existe um algoritmo A que decide X em tempo polinomial:

$$x \in X \iff A(x) = 1$$

- Temos de mostrar que existe um algoritmo A' q verifica X em tempo polinomial. Definimos A' como se segue:

$$A'(x, y) = A(x)$$

↳ o certificado pode ser a string vazia

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- Exp - conjunto dos problemas decidíveis em tempo exponencial

Proposição 2: $NP \subseteq Exp$

- Suponhamos q $X \in NP$, temos de provar que $X \in Exp$.
- Como $X \in NP$, concluímos que existe um algoritmo A q verifica X em tempo polinomial:

$$x \in X \iff \exists y.\ |y| < p(|x|) \wedge A(x,y) = 1$$

- Temos de mostrar que existe um algoritmo A' q decide X em tempo exponencial.

- Definimos A' como se segue:

```
A'(x) =
  for each y of size ≤ |p(x)|
    if A(x,y)
      then return 1
  return 0
```

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial

- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)

- Exp - conjunto dos problemas decidíveis em tempo exponencial

- co-NP - conjunto dos problemas cujo complemento está em NP

  Exemplo:

  UNSAT = $\{ \Phi \mid \Phi$ não é satisfazível $\}$

Proposição 3: $P \subseteq NP \cap co\text{-}NP$

- $P \subseteq NP$ (proposição 1)
- $P \subseteq co\text{-}NP$ (falta provar)

  $\Leftrightarrow X \in P \Rightarrow X \in co\text{-}NP \Rightarrow \bar{X} \in NP$

  mas $\underline{X \in P \Rightarrow \bar{X} \in P}$

- Temos q provar q existe um algoritmo A' q decide $\bar{X}$.

  Seja A o algoritmo q decide X.

A'(x) :
  if (A(x))
    then return 0
    else return 1

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- Exp - conjunto dos problemas decidíveis em tempo exponencial
- co-NP - conjunto dos problemas cujo complemento está em NP

  Exemplo:

  UNSAT = $\{ \Phi \mid \Phi$ não é satisfazível $\}$

- NP = co-NP ?

· Suponhamos que $X \in NP$, temos de provar q $\overline{X} \in NP$.

· De $X \in NP$, concluímos q existe um algoritmo A tal que:

$$x \in X \text{ sse } \exists y.\ A(x,y) = 1$$

$$\downarrow$$

$$x \notin X \text{ sse } \underbrace{\neg \exists y.\ A(x,y) = 1}$$

Temos de experimentar todos os certificados!

Não conseguimos usar o verificador de X para construir o verificador de $\overline{X}$.

Fail!

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial

- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)

Exp - conjunto dos problemas decidíveis em tempo exponencial

co-NP - conjunto dos problemas cujo complemento está em NP

Exemplo:

$UNSAT = \{ \Phi \mid \Phi$ não é satisfazível $\}$

Caracterização Alternativa: $\mathcal{X} \in$ co-NP sse existe um algoritmo de verificação polinomial A tal que:

$$x \in \mathcal{X} \Leftrightarrow \forall y \in Cert(\mathcal{X}).\ A(x, y) = 0$$

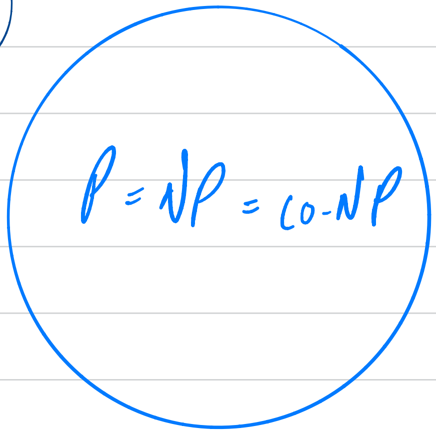
(*) Em geral, não conseguimos provar em tempo polinomial que uma dada fórmula Ψ pertence a UNSAT, mas conseguimos provar que não pertence.

$$x \notin \mathcal{X} \Leftrightarrow \exists y \in Cent(\mathcal{X}).\ A(x, y) = 1$$

Classes de Complexidade - Conjecturas

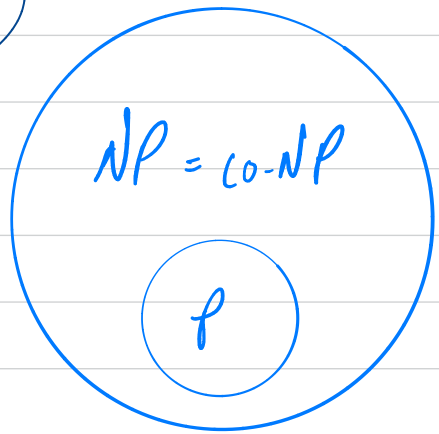
* 4 possibilidades

I



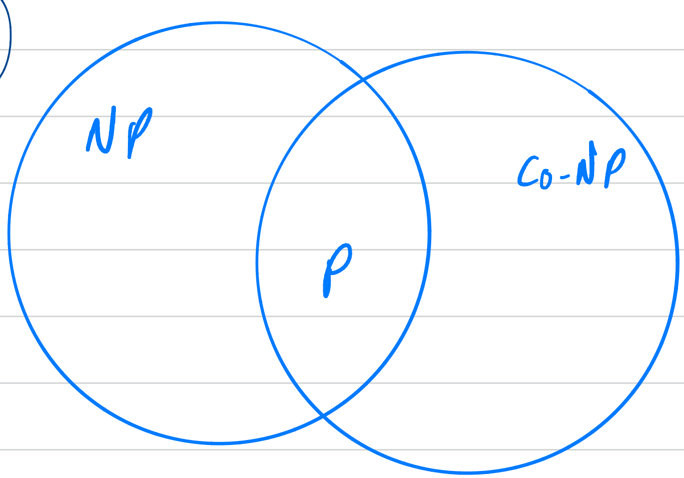
P = NP?

II



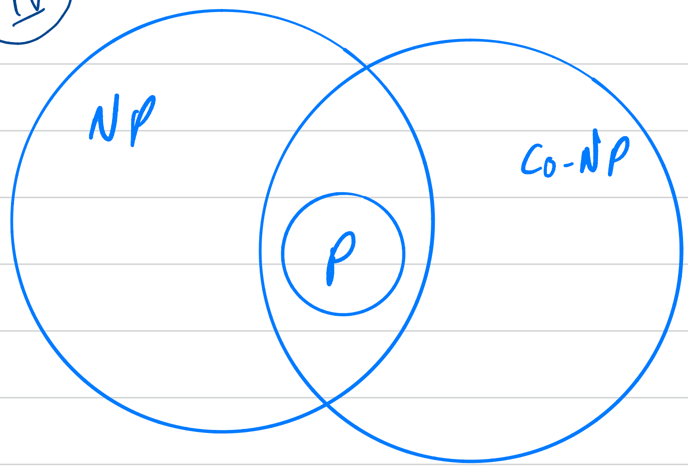
NP = co-NP?

III



$P = NP \cap co\text{-}NP$?

IV



Exemplo: Incompatibilidades

II.d) Uma matriz de incompatibilidades é uma matriz quadrada cujas células guardam valores decimais entre 0 e 1. Intuitivamente, dada uma matriz de incompatibilidades M, $n \times n$, a célula M_{ij} guarda a incompatibilidade entre os índices i e j; $M_{ij} = 0$ se i e j são completamente compatíveis e $M_{ij} = 1$ se i e j são completamente incompatíveis. Dado um sub-conjunto de índices $I \subseteq \{1, ..., n\}$, o nível de incompatibilidade do conjunto é dado por: $\sum_{i,j \in I} M_{ij}$. O problema das incompatibilidades define-se formalmente da seguinte maneira:

$$\textbf{Incompat} = \{\langle M, k, v\rangle \mid M \text{ contém um sub-conjunto de índices de tamanho } k \text{ e incompatibilidade igual ou inferior a } v\}$$

Algoritmo de Verificação:

Input: • $<M, k, v>$

• Certificado: I - conjunto de índices com incompatibilidade k

• Verificar q:

- $\sum_{i,j \in I} M_{ij} \leq v$
- $|I| = k$

• Complexidade: $O(n^2)$

Exemplo: Cromos Partilhados

II.d) Seja $C = \{c_1, ..., c_n\}$ uma colecção de cromos e $\mathcal{A} = \{a_1, ..., a_m\}$ um grupo de amigos que coleccionam cromos. Cada membro do grupo detém um subconjunto de C; seja C_i o conjunto de cromos detido por a_i e $\mathcal{C} = \{C_i \mid 1 \leq i \leq m\}$ o conjunto dos conjuntos de cromos de todos os membros do grupo. Os membros do grupo pretendem determinar o mais pequeno conjunto de cromos que contém pelo menos um cromo detido por cada membro do grupo. Formalmente, este problema pode ser modelado através do seguinte problema de decisão:

$$\textbf{SharedStickers} = \{\langle C, \mathcal{C}, k\rangle \mid \exists X \subseteq C.\, |X| = k \wedge \forall_{1 \leq i \leq m}.\, C_i \cap X \neq \emptyset\}$$

Input:

- $\langle C, \mathcal{C}, k \rangle$
- certificado: $X \subseteq C$

- Algoritmo de Verificação:
  - Para todo C_i, $1 \leq i \leq m$, verificar q: $C_i \cap X \neq \emptyset$
  - Verificar q $|X| = k$

Complexidade: $O(m \cdot n^2)$

Classes de Complexidade - TPC

- $NP \neq co\text{-}NP \Rightarrow P \neq NP$

- P é fechada para a intersecção, união, complemento, concatenação e fecho de Kleene.
  - $X_1, X_2 \in P \Rightarrow X_1 \cap X_2 \in P$
  - $X_1, X_2 \in P \Rightarrow X_1 \cup X_2 \in P$
  - $X_1 \in P \Rightarrow \overline{X_1} \in P$
  - $X_1, X_2 \in P \Rightarrow X_1 . X_2 \in P$
  - $X_1 \in P \Rightarrow X_1^* \in P$

- NP é fechada para a intersecção, união, concatenação e fecho de Kleene.

Redutibilidade NP

- O problema X é redutível em tempo polinomial (polynomial-time reducible) ao problema Y sse existe uma função f calculável em tempo polinomial tal que:

$$x \in X \iff f(x) \in Y$$

Escrevemos: $X \leq_P Y$

Proposição 4: $Y \in P \wedge X \leq_P Y \Rightarrow X \in P$

Prova:

- Suponhamos que $Y \in P$ e $X \leq_P Y$, há que provar que $X \in P$.
- Seja A o algoritmo q decide Y e f a função q reduz X a Y.

- Temos q construir um algoritmo A' q decide X em tempo polinomial

$A'(x)$:

retorna $A(f(x))$

Completude NP

- Um problema X diz-se NP-difícil sse:
  $\forall Y \in NP.\ Y \leq_p X$

- Um problema X diz-se NP-completo sse:
  - $X \in NP$
  - X é NP-difícil

Proposição 5: Se um problema NP-completo for resolúvel em tempo polinomial então $P = NP$.

Prova:

- Assumindo q existe $X \in P$ tal q X é NP-difícil, temos de provar q $\forall Y \in NP.\ Y \in P$.
- Tomemos um qualquer $Y \in NP$; como X é NP-difícil, concluímos q existe h, calculável em tempo polinomial, tal q:

  $$y \in Y \iff h(y) \in X$$

- Seja A o algoritmo polinomial q decide X, definimos o algoritmo A' que decide Y em tempo polinomial como se segue:

  $A'(y)$:
  return $A(f(y))$

Completude NP

- Um problema X diz-se NP-difícil sse:
  $\forall Y \in NP.\ Y \leq_p X$

- Um problema X diz-se NP-completo sse:
  - $X \in NP$
  - X é NP-difícil

- Seja $\mathcal{C}$ uma classe de problemas, dizemos que $\mathcal{X}$ é completo para $\mathcal{C}$ sse:
  - $\mathcal{X} \in \mathcal{C}$
  - $\forall \mathcal{Y} \in \mathcal{C}.\ \mathcal{Y} \leq_p \mathcal{X}$

Generalização do conceito de completude

Proposição 6: $\mathcal{X}$ é completo para NP sse $\bar{\mathcal{X}}$ é completo para co-NP.

($\Rightarrow$) Suponhamos q $\mathcal{X}$ é completo para NP, queremos provar que $\bar{\mathcal{X}}$ é completo para co-NP.

- Para todo o $\mathcal{Y} \in$ co-NP, há q mostrar q $\mathcal{Y} \leq_p \bar{\mathcal{X}}$.

$\mathcal{Y} \in$ co-NP $\Rightarrow \bar{\mathcal{Y}} \in NP \Rightarrow \bar{\mathcal{Y}} \leq_p \mathcal{X}$

$\forall y \in dom(\bar{\mathcal{Y}}).\ y \in \bar{\mathcal{Y}} \Leftrightarrow f(y) \in \mathcal{X}$

$\forall y \in dom(\mathcal{Y}).\ y \notin \mathcal{Y} \Leftrightarrow f(y) \in \mathcal{X}$

$\forall y \in dom(\mathcal{Y}).\ y \in \mathcal{Y} \Leftrightarrow f(y) \notin \mathcal{X}$

$$\underbrace{\forall y \in dom(\mathcal{Y}).\ y \in \mathcal{Y} \Leftrightarrow f(y) \in \bar{\mathcal{X}}}_{\mathcal{Y} \leq_p \bar{\mathcal{X}}}$$

Completude NP

- Um problema X diz-se NP-difícil sse:
  $\forall Y \in NP.\ Y \leq_p X$

- Um problema X diz-se NP-completo sse:
  - $X \in NP$
  - X é NP-difícil

Proposição 7: $X \in NP \land Y \leq_p X \land Y \in NPC \Rightarrow X \in NPC$

Prova: Há que provar que $\forall Z \in NP.\ Z \leq_p X$

- Tomemos $Z \in NP$.
- Como $Y \in NPC$, temos que $Z \leq_p Y$.
- De $Y \leq_p X$ e $Z \leq_p Y$ segue que $Z \leq_p X$ (pela transitividade de $\leq_p$).

■

Completude NP

- Um problema X diz-se NP-difícil sse:
  $\forall\, Y \in NP.\ Y \leq_p X$

- Um problema X diz-se NP-completo sse:
  - $X \in NP$
  - X é NP-difícil

Proposição 7: $X \in NP \wedge Y \leq_p X \wedge Y \in NPC \Rightarrow X \in NPC$

* Como provar que um problema X é NP-completo?

① Provamos que X está em NP (fácil)

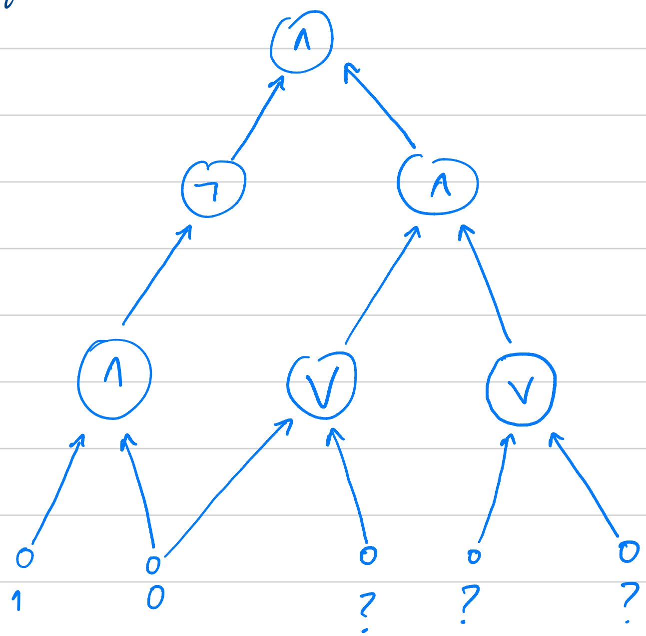
↳ Descobrir o certificado q nos permite verificar X em tempo polinomial

② Seleccionar um problema NP-completo Y e construir uma redução $Y \leq_p X$. (difícil)

O 1º Problema NP-Completo

Circuit-SAT: Dado um circuito combinatório construído com portas And, Or e Not, existem inputs q̄ tornam o output do circuito $\underline{1}$.

Exemplo:



Sim $\Rightarrow$ $\underline{101}$

O 1º Problema NP-Completo

Circuit-SAT: Dado um circuito combinatório construído com portas And, Or e Not, existem inputs q̄ tornam o output do circuito 1.

Teorema [Cook-Levin] Circuit-SAT é NP-completo.

Esboço de Prova:

- Tome-se $X \in NP$. Da definição de NP segue que existe um algoritmo de verificação A tal que:
$$x \in X \iff \exists y.\ A(x,y)=1$$
- Um algoritmo polinomial pode ser implementado por um circuito combinatório de tamanho polinomial. Seja R esse circuito.
- Fixamos as 1ªs $|x|$ entradas de R com os bits de x. As restantes $|y|$ entradas ficam com ?.
- O circuito R é satisfazível sse $\exists y.\ A(x,y)=1$ (sse $x \in X$).