Teorema PCP - Probabilistically Checkable Proof

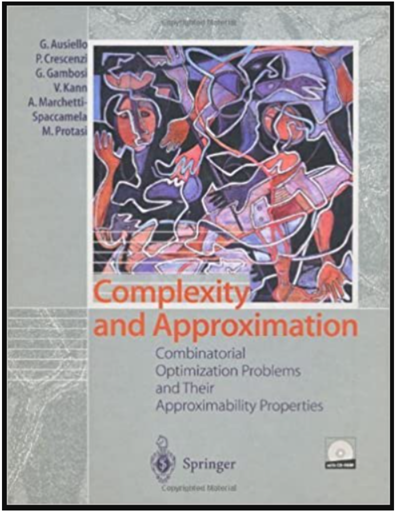


Figure: Livro "Complexity and Approximation" de Ausiello et al., 2003

Teorema PCP - Probabilistically Checkable Proof

A Short Guide to Approximation Preserving Reductions

Pierluigi Crescenzi*
Università di Roma "La Sapienza"

Abstract

Comparing the complexity of different combinatorial optimization problems has been an extremely active research area during the last 23 years. This has led to the definition of several approximation preserving reducibilities and to the development of powerful reduction techniques. We first review the main approximation preserving reducibilities that have appeared in the literature and suggest which one of them should be used. Successively, we give some hints on how to prove new non-approximability results by emphasizing the most interesting techniques among the new ones that have been developed in the last few years.

1. Introduction

What is it that makes algorithms for different problems behave in the same way? Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results, or are they due to some structural similarity between the problems as we define them?

David S. Johnson [26]

classes	inapproximability	completeness
NPO	$\leq_L$	$\leq_{AP}$
poly-APX		
log-APX		
APX		$\leq_{PTAS}$

Table 2. The three candidates

We want to conclude by recalling that, whenever no problem has been found that is reducible to the target problem A, there is still another possibility: the non-approximability result for A may be derived directly from the PCP theorem (or from one of its variations [9]) by making use, for instance, of the gadget method introduced in [9] (this method has the advantage that the gadgets can be constructed automatically [43]). This alternative may be summarized in the following statement (which is a slight modification of a motto that has been used in the 80s):

real men reduce from PCP!

1997

"real programmers use Assembly"

Teorema PCP - Probabilistically Checkable Proof

DEFINIÇÃO: Um problema de decisão $L \in PCP_{c,s}[r(n), q(n)]$ se tem um verificador $\mathcal{V}$ polinomial (probabilístico) que, para toda instância I de L:

a. dada uma prova π: $\mathcal{V}$ lê a instância I, recebe uma palavra aleatória R com $r(n)$ bits e, de forma não-adaptativa, lê $q(n)$ bits de π e decide se aceita ou rejeita.
b. **Se I é SIM em L:** existe π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \geq c$ (*completeness*)
c. **Se I é NÃO em L:** para todo π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \leq s$ (*soundness* $s < c$)

- Em português, é comum usar "certificado", ao invés de "prova", que usaremos mais.
- Padrão $\mathbf{c = 1}$: Probabilidade de aceitar uma prova correta.
- Padrão $\mathbf{s = 1/2}$: Probabilidade de aceitar uma prova errada.
- *Randomness complexity* $r(n)$, *Query complexity* $q(n)$
- *Proof complexity* $q(n) \cdot 2^{r(n)}$. Podemos assumir que as provas tem no máximo esse tamanho
- A probabilidade é sobre a palavra binária aleatória R (de tam. $r(n)$)

Teorema PCP - Probabilistically Checkable Proof

DEFINIÇÃO: Um problema de decisão $L \in PCP_{c,s}[r(n), q(n)]$ se tem um verificador $\mathcal{V}$ polinomial (probabilístico) que, para toda instância I de L:

a. dada uma prova π: $\mathcal{V}$ lê a instância I, recebe uma palavra aleatória R com $r(n)$ bits e, de forma não-adaptativa, lê $q(n)$ bits de π e decide se aceita ou rejeita.
b. **Se I é SIM em L:** existe π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \geq c$ (*completeness*)
c. **Se I é NÃO em L:** para todo π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \leq s$ (*soundness* $s < c$)

- Padrão $\mathbf{c} = \mathbf{1}$: Probabilidade de aceitar uma prova correta. Padrão $\mathbf{s} = \mathbf{1/2}$: Probabilidade de aceitar uma prova errada.
- *Proof complexity* $q(n) \cdot 2^{r(n)}$. Podemos assumir que as provas tem no máximo esse tamanho

Exemplos de classes PCP

- $\mathbf{P} = \mathbf{PCP_{1,0}}\,[\mathbf{0}, \mathbf{0}]$. Tempo polinomial sem aleatoriedade e sem prova.
- $\mathbf{P} = \mathbf{PCP}\,[\mathbf{0}, \mathbf{0}]$. Idem: sem aleatoriedade, é certeza de sim (1) ou de não (0)
- $\mathbf{P} = \mathbf{PCP}[\log \mathbf{n}, \mathbf{0}]$. Podemos gerar todos os R_i's e simular o verificador
- $\mathbf{P} = \mathbf{PCP}[\mathbf{0}, \log \mathbf{n}]$. Podemos gerar todas as provas e testar no verificador
- $\mathbf{NP} = \mathbf{PCP_{1,0}}[\mathbf{0}, \mathbf{poly(n)}] = \mathbf{PCP}[\log \mathbf{n}, \mathbf{poly(n)}]$, onde
  $$poly(n) = \cup_{k=1}^{\infty} O(n^k)$$
- $\mathbf{NP} \supseteq \mathbf{PCP}[\log \mathbf{n}, \mathbf{O(1)}]$

Teorema PCP - Probabilistically Checkable Proof

DEFINIÇÃO: Um problema de decisão $L \in PCP_{c,s}[r(n), q(n)]$ se tem um verificador $\mathcal{V}$ polinomial (probabilístico) que, para toda instância I de L:

a. dada uma prova π: $\mathcal{V}$ lê a instância I, recebe uma palavra aleatória R com $r(n)$ bits e, de forma não-adaptativa, lê $q(n)$ bits de π e decide se aceita ou rejeita.

b. **Se I é SIM em L:** existe π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \geq c$ (*completeness*)

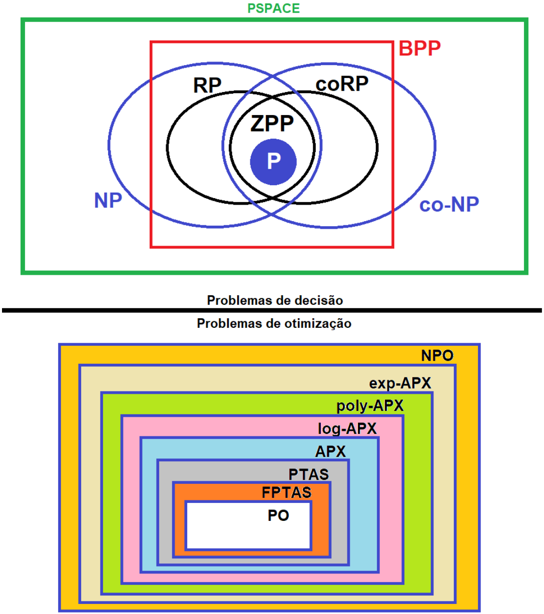
c. **Se I é NÃO em L:** para todo π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \leq s$ (*soundness* $s < c$)

- Padrão $\mathbf{c} = \mathbf{1}$: Probabilidade de aceitar uma prova correta. Padrão $\mathbf{s} = \mathbf{1/2}$: Probabilidade de aceitar uma prova errada.
- *Proof complexity* $q(n) \cdot 2^{r(n)}$. Podemos assumir que as provas tem no máximo esse tamanho

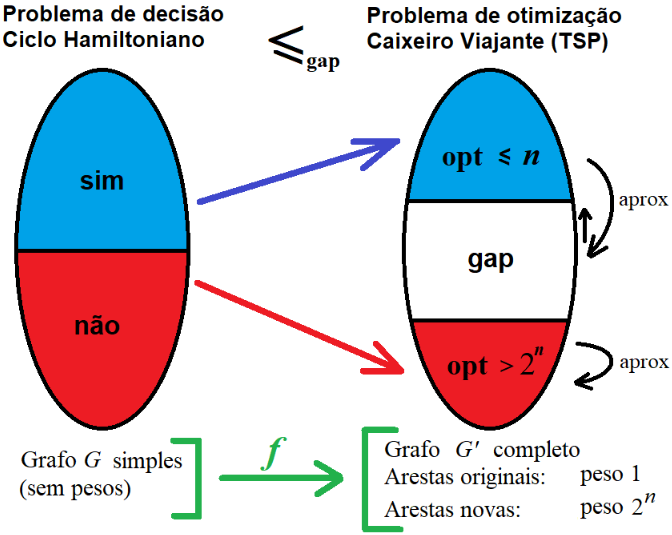
Exemplos de classes PCP

- $\mathbf{ZPP} = \mathbf{PCP_{1,0}}\,[\mathbf{poly(n), 0}]$ $\qquad$ $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$
- $\mathbf{RP} = \mathbf{PCP_{1/2,\ 0}}\,[\mathbf{poly(n), 0}]$
- $\mathbf{coRP} = \mathbf{PCP_{1,\ 1/2}}\,[\mathbf{poly(n), 0}]$ $\qquad$ $\mathbf{RP = co\text{-}RP\ ?}$
- $\mathbf{BPP} = \mathbf{PCP_{2/3,\ 1/3}}\,[\mathbf{poly(n), 0}]$. $\qquad$ $\mathbf{BPP = co\text{-}BPP}$.
- $\mathbf{BPP} = \mathbf{PCP_{1/2+\varepsilon,\ 1/2-\varepsilon}}\,[\mathbf{poly(n), 0}]$. $\qquad$ $\mathbf{BPP} \supseteq \mathbf{RP} \cup \mathbf{coRP}$
- $\qquad$ $\mathbf{BPP = P\ ?}$

Teorema PCP - Hierarquia de classes (se P≠NP)

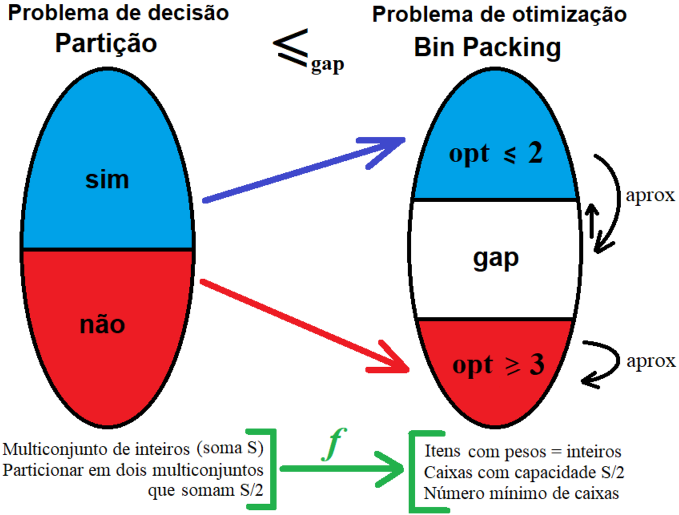


Decisão × Otimização: Redução Gap



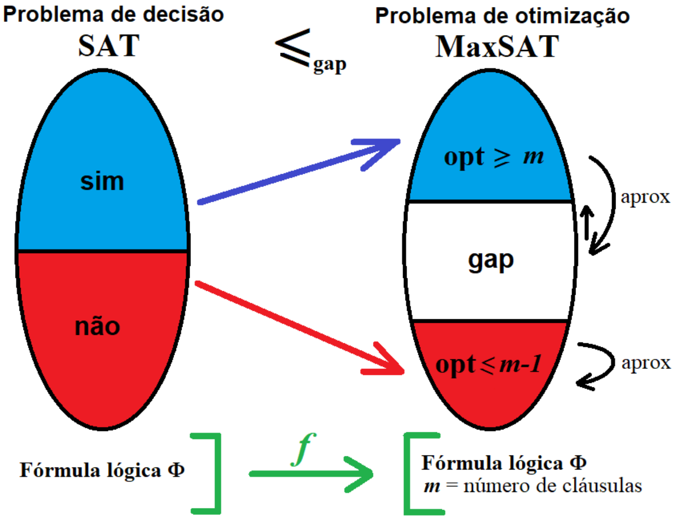
Conclusão: Não existe algoritmo poli. $(2^n/n)$-aproximativo para TSP, a menos que P=NP, onde n é o número de vértices de G.

Decisão × Otimização: Redução Gap



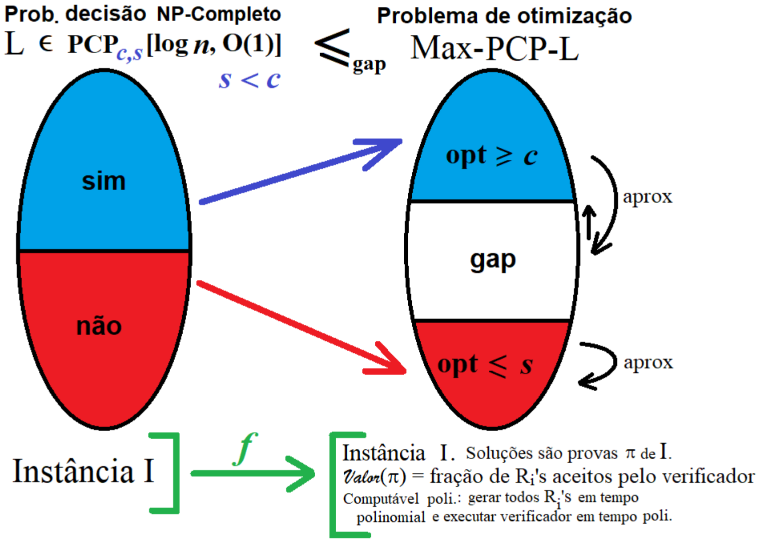
Conclusão: Não existe algoritmo poli. (3/2 - ε) - aproximativo para BP a menos que P=NP.

Decisão × Otimização: Redução Gap



Conclusão: Não existe algoritmo poli. $(1 - 1/m + \varepsilon)$ - aproximativo para **MaxSAT** a menos que P=NP.

Decisão × Otimização: Redução Gap



Conclusão: Não existe algoritmo poli. $(s/c + \varepsilon)$ - aproximativo para **Max-PCP-L** a menos que P=NP.

Teorema PCP - Probabilistically Checkable Proof

DEFINIÇÃO: Um problema de decisão $L \in PCP_{c,s}[r(n), q(n)]$ se tem um verificador $\mathcal{V}$ polinomial (probabilístico) que, para toda instância I de L:

a. dada uma prova π: $\mathcal{V}$ lê a instância I, recebe uma palavra aleatória R com $r(n)$ bits e, de forma não-adaptativa, lê $q(n)$ bits de π e decide se aceita ou rejeita.

b. **Se I é SIM em L:** existe π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \geq c$ (*completeness*)

c. **Se I é NÃO em L:** para todo π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \leq s$ (*soundness*)

- Valores padrão: $\mathbf{c = 1}$ e $\mathbf{s = 1/2}$
- Aqui assumimos que a prova (ou certificado) π tem tam. polin. no tam. n da instância I
- A probabilidade é sobre a palavra binária aleatória R (de tamanho $r(n)$)
- $\mathbf{P} = \mathbf{PCP_{1,0}\,[0,0]} = \mathbf{PCP[0,0]} = \mathbf{PCP[\log n, 0]} = \mathbf{PCP[0, \log n]}$
- $\mathbf{NP} = \mathbf{PCP_{1,0}\,[0, poly(n)]} = \mathbf{PCP[\log n, poly(n)]}$, onde $poly(n) = \cup_{k=1}^{\infty} O(n^k)$

Teorema PCP (Arora et al, 1998): $\boxed{\mathbf{NP} = \mathbf{PCP\,[\,O(\log n),\ O(1)\,]}}$

Teorema PCP - Melhorias

DEFINIÇÃO: Um problema de decisão $L \in PCP_{c,s}[r(n), q(n)]$ se tem um verificador $\mathcal{V}$ polinomial (probabilístico) que, para toda instância I de L:

a. dada uma prova π: $\mathcal{V}$ lê a instância I, recebe uma palavra aleatória R com $r(n)$ bits e, de forma não-adaptativa, lê $q(n)$ bits de π e decide se aceita ou rejeita.

b. **Se** I **é SIM em** L**:** existe π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \geq c$ (padrão $c = 1$)

c. **Se** I **é NÃO em** L**:** para todo π, $\mathbb{P}(\mathcal{V}(\mathrm{I}, \pi) = 1) \leq s$ (padrão $s = 1/2$)

Teo. PCP (Arora et al, 1998): $\boxed{\mathbf{NP} = \mathbf{PCP}\ [\,\mathbf{O}(\log \mathbf{n}),\ \mathbf{O}(\mathbf{1})\,]}$

Teo. PCP: $\mathbf{NP} = \mathbf{PCP}_{1,\ 0.32}\ [\,\mathbf{O}(\log \mathbf{n}),\ \mathbf{9}\,]$

Teo. PCP: $\mathbf{NP} = \mathbf{PCP}_{1,\ 0.76}\ [\,\mathbf{O}(\log \mathbf{n}),\ \mathbf{3}\,]$

Teo. PCP: $\mathbf{NP} = \mathbf{PCP}_{0.99,\ 0.51}\ [\,\mathbf{O}(\log \mathbf{n}),\ \mathbf{3}\,]$

Teo. PCP (Hastad, 2001): $\mathbf{NP} = \mathbf{PCP}_{1-\varepsilon,\ 1/2+\varepsilon}\ [\mathbf{O}(\log \mathbf{n}), \mathbf{3}], \forall \varepsilon > 0.$

Além disso, após ler 3 bits i_1, i_2, i_3 da prova π, o verificador aceita se e só se $\pi_{i_1} \oplus \pi_{i_2} \oplus \pi_{i_3} = b_i$, para um certo bit de teste $b_i \in \{0, 1\}$ que depende apenas de I e de R_i.

(Ou exclusivo) $\pi_i \oplus \pi_j \oplus \pi_k = \pi_i + \pi_j + \pi_k \pmod 2$

(Ou exclusivo) $\pi_i \oplus \pi_j \oplus \pi_k = (\pi_i \vee \pi_j \vee \pi_k) \wedge (\pi_i \vee \overline{\pi_j} \vee \overline{\pi_k}) \wedge (\overline{\pi_i} \vee \pi_j \vee \overline{\pi_k}) \wedge (\overline{\pi_i} \vee \overline{\pi_j} \vee \pi_k)$

$(1/2+\varepsilon)$-inaproximabilidade de MaxE3Lin-2

Teo. PCP (Hastad, 2001): $\mathbf{NP} = \mathbf{PCP}_{\mathbf{1-\varepsilon},\ \mathbf{1/2+\varepsilon}}\ [\mathbf{O}(\log \mathbf{n}), \mathbf{3}], \forall \varepsilon > 0.$
Além disso, após ler 3 bits i_1, i_2, i_3 da prova π, o verificador aceita se e só se $\pi_{i_1} \oplus \pi_{i_2} \oplus \pi_{i_3} = b_i$, para um certo bit de teste $b_i \in \{0, 1\}$ que depende apenas de I e de R_i.

(Ou exclusivo) $\pi_{i_1} \oplus \pi_{i_2} \oplus \pi_{i_3} \;=\; \pi_{i_1} + \pi_{i_2} + \pi_{i_3} \pmod 2 \;=$
$= \;\pi_{i_1} \oplus \pi_{i_2} \oplus \pi_{i_3} \;=\; (\pi_{i_1} \vee \pi_{i_2} \vee \pi_{i_3}) \wedge (\pi_{i_1} \vee \overline{\pi_{i_2}} \vee \overline{\pi_{i_2}}) \wedge (\overline{\pi_{i_1}} \vee \pi_{i_2} \vee \overline{\pi_{i_3}}) \wedge (\overline{\pi_{i_1}} \vee \overline{\pi_{i_2}} \vee \pi_{i_3})$

Seja L um problema qualquer NP-Completo.

MaxE3Lin-2

- Instância: Conj. equações lineares mod 2 com ≤ 3 variáveis
- Redução Gap de um problema L qualquer NP-Completo.
- Construção: Dada I em L: para cada R_i, sejam i_1, i_2, i_3 as posições dos bits a serem lidos do certificado e seja b_i o bit de teste. Crie 3 variáveis $x_{i_1}, x_{i_2}, x_{i_3}$ e a equação $x_{i_1} + x_{i_2} + x_{i_3} = b_i$.
- Cada prova π de I leva a uma valoração das var. x's (e vice-versa).
- I é SIM $\Rightarrow \exists \pi : \geq (1-\varepsilon)$ dos R_i's são aceitos $\Rightarrow \geq (1-\varepsilon)$ das equações são satisfeitas.
- I é NÃO $\Rightarrow \forall \pi : \leq (\frac{1}{2}+\varepsilon)$ dos R_i's são aceitos $\Rightarrow \leq (\frac{1}{2}+\varepsilon)$ das equações são satisfeitas.
- $\Rightarrow$ MaxE3Lin-2 é $\left(\frac{1/2+\varepsilon}{1-\varepsilon}\right) = (\frac{1}{2}+\varepsilon')$-inaprox poli., se P$\neq$NP
- MaxE3Lin-2 tem algoritmo $\frac{1}{2}$-aprox. poli. (chuta valores para as variáveis)

$(7/8 + \varepsilon)$-inaproximabilidade de Max3SAT

Teo. PCP (Hastad, 2001): NP $= \mathbf{PCP}_{1-\varepsilon,\ 1/2+\varepsilon}\,[\mathbf{O}(\log \mathbf{n}), \mathbf{3}], \forall \varepsilon > 0.$
Além disso, após ler 3 bits i_1, i_2, i_3 da prova π, o verificador aceita se e só se $\pi_{i_1} \oplus \pi_{i_2} \oplus \pi_{i_3} = b_i$, para um certo bit de teste $b_i \in \{0,1\}$ que depende apenas de I e de R_i.

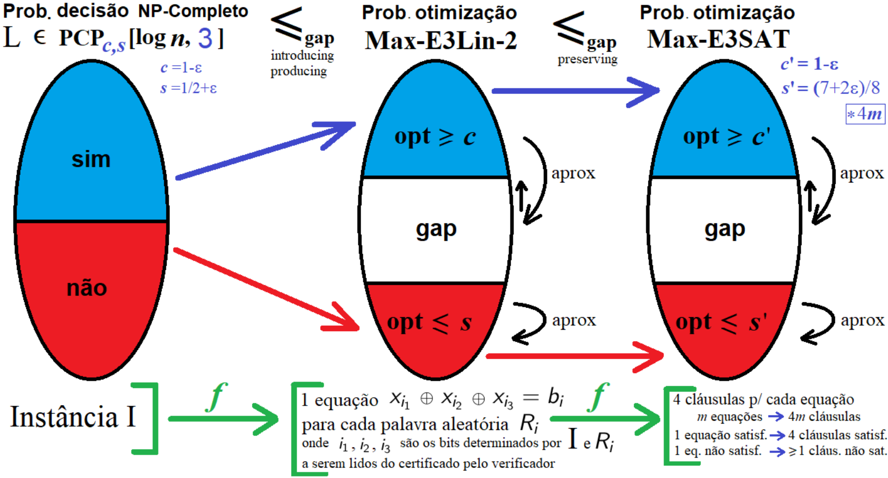
(Ou exclusivo) $\pi_{i_1} \oplus \pi_{i_2} \oplus \pi_{i_3} \;=\; \pi_{i_1} + \pi_{i_2} + \pi_{i_3} \pmod 2 \;=$
$= \pi_{i_1} \oplus \pi_{i_2} \oplus \pi_{i_3} \;=\; (\pi_{i_1} \vee \pi_{i_2} \vee \pi_{i_3}) \wedge (\pi_{i_1} \vee \overline{\pi_{i_2}} \vee \overline{\pi_{i_2}}) \wedge (\overline{\pi_{i_1}} \vee \pi_{i_2} \vee \overline{\pi_{i_3}}) \wedge (\overline{\pi_{i_1}} \vee \overline{\pi_{i_2}} \vee \pi_{i_3})$

Seja L um problema qualquer NP-Completo.

Max-3SAT

- Redução Gap de um problema L qualquer NP-Completo.
- Construção: Dada I em L: para cada R_i, sejam i_1, i_2, i_3 as posições dos bits a serem lidos do certificado e seja b_i o bit de teste. Crie 3 variáveis $x_{i_1}, x_{i_2}, x_{i_3}$ e 4 claúsulas.
- Cada prova π de I leva a uma atribuição das var. x's (e vice-versa).
- Dada prova π de I: se R_i é aceito, então 4 cláusulas são satisfeitas.
- I é SIM $\Rightarrow \exists \pi : \geq (1-\varepsilon)$ dos R_i's são aceitos $\Rightarrow\ \geq (1-\varepsilon)$ das cláusulas são satisfeitas.
- I é NÃO $\Rightarrow \forall \pi : \geq (\frac{1}{2} - \varepsilon)$ dos R_i's não são aceitos $\Rightarrow\ \geq (\frac{1}{2} - \varepsilon) \cdot \frac{1}{4}$ das cláusulas não são satisfeitas $\Rightarrow\ \leq 1 - (\frac{1}{2} - \varepsilon) \cdot \frac{1}{4} = \frac{7+2\varepsilon}{8}$ das cláusulas são satisfeitas.
- $\Rightarrow$ Max-E3SAT é $\left(\frac{(7+2\varepsilon)/8}{1-\varepsilon}\right) = (\frac{7}{8} + \varepsilon')$-inaprox poli., se P$\neq$NP
- Max-E3SAT tem algoritmo $\frac{7}{8}$-aprox. poli. (MaxSAT-Johnson desaleat.)

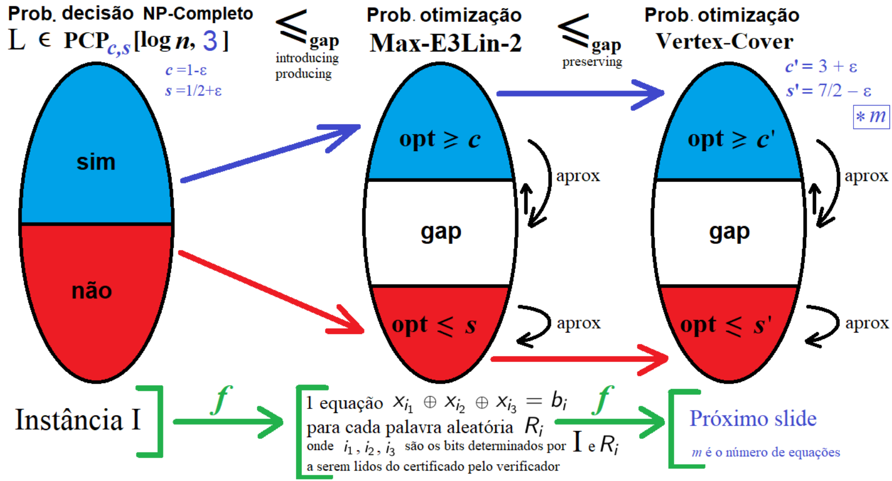
$(7/8+\varepsilon)$-inaproximabilidade de Max3SAT



Conclusão: Não existe algoritmo poli. $(1/2+\varepsilon)$ - aproximativo p/ **Max-E3Lin-2** a menos que P=NP.

Não existe algoritmo poli. $(7/8+\varepsilon)$ - aproximativo p/ **Max-E3SAT** a menos que P=NP.

$(7/6 - \varepsilon)$-inaproximabilidade de Vertex-Cover



Conclusão: Não existe algoritmo poli. $(1/2 + \varepsilon)$ - aproximativo p/ **Max-E3Lin-2** a menos que P=NP.
Não existe algoritmo poli. $(7/6 - \varepsilon)$ - aproximativo p/ **Vertex-Cover** a menos que P=NP.

$(7/6 - \varepsilon)$-inaproximabilidade de Vertex-Cover

Redução de Max-E3Lin-2

- Redução: Para cada equação $x_{i_1} \oplus x_{i_2} \oplus x_{i_3} = b_i$, crie uma clique em G com 4 vértices, onde cada vértice representa uma atribuição de $(x_{i_1}, x_{i_2}, x_{i_3})$ que satisfaz a equação. Crie ainda uma aresta entre dois vértices se são conflitantes.

- Dada uma valoração em Max-E3Lin-2, selecione o vértice de G referente a essa valoração na clique de cada equação satisfeita. Esses vértices formam um conjunto independente.
- $\geq (1-\varepsilon)m$ equações satisf. em Max-E3Lin-2 $\Rightarrow \alpha(G) \geq (1-\varepsilon)m \Leftrightarrow vc(G) \leq (3+\varepsilon)m$

- Dada um conjunto independente S em G, seus vértices estão associados a valorações não conflitantes em diferentes equações, que serão satisfeitas com essa valoração.

- $\leq (\frac{1}{2}+\varepsilon)m$ equações satisf. em Max-E3Lin-2 $\Rightarrow \alpha(G) \leq (\frac{1}{2}+\varepsilon)m \Leftrightarrow vc(G) \leq (\frac{7}{2}-\varepsilon)m$

- Vertex-Cover é $(\frac{(7/2-\varepsilon)m}{(3+\varepsilon)m}) = (\frac{7}{6} - \varepsilon')$-inaprox poli., se P$\neq$NP
- Vertex-Cover possui algoritmo 2-aprox. poli.

- Resultado similar para MaxCut pode ser obtido assim (um pouco mais complicado): MaxCut é $(\frac{16}{17}+\varepsilon')$=$(0.941+\varepsilon')$-inaproximável poli., se P$\neq$NP

- Max-Cut possui algoritmo 0.878-aprox. poli.