Uma introdução à complexidade parametrizada

Vinicius Fernandes dos Santos - CEFET-MG

Uéverton dos Santos Souza - UFF/CEFET-RJ

34º JAI - Jornadas de Atualização em Informática
XXXV Congresso da Sociedade Brasileira de Computação
Recife, Julho 20–23, 2015

Complexidade clássica

Problema computacional

1. Uma entrada.
2. A questão a ser respondida.

Complexidade clássica

Problema computacional

1. Uma entrada.
2. A questão a ser respondida.

Um **algoritmo** A para um problema Π é uma sequência finita de instruções para um computador, com a finalidade de solucionar Π.

Complexidade clássica

Tratabilidade de tempo polinomial

- Um *algoritmo de tempo polinomial* é definido como um algoritmo cuja sua função de complexidade de tempo é $O(p(n))$, para alguma função polinomial p, onde n é usado para denotar o tamanho da entrada.

Complexidade clássica

Tratabilidade de tempo polinomial

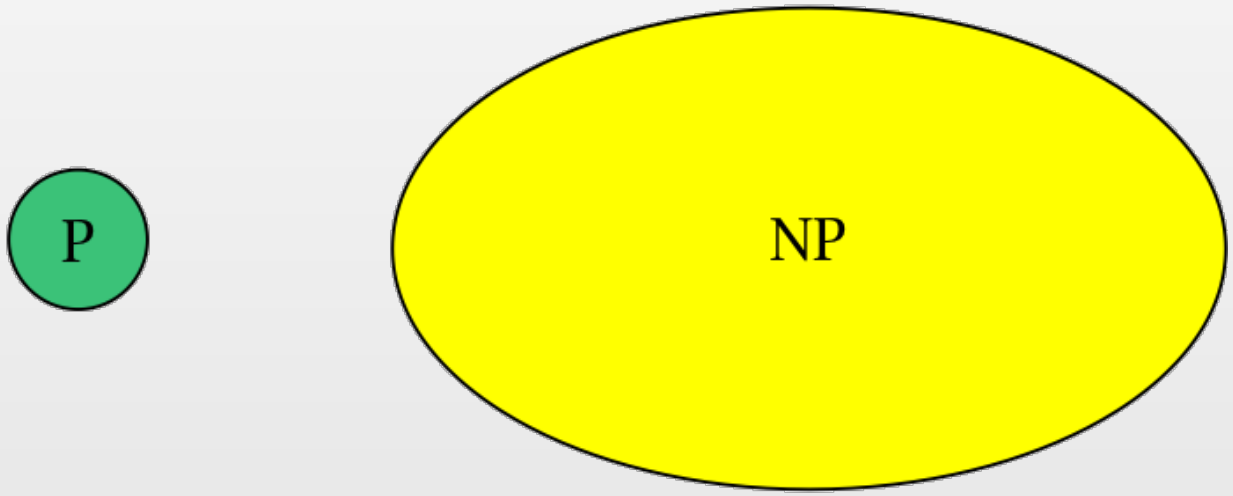
- Um *algoritmo de tempo polinomial* é definido como um algoritmo cuja sua função de complexidade de tempo é $O(p(n))$, para alguma função polinomial p, onde n é usado para denotar o tamanho da entrada.
- Um problema Π pertence à classe P se e somente se Π pode ser solucionado em tempo polinomial por um algoritmo determinístico.

Complexidade clássica

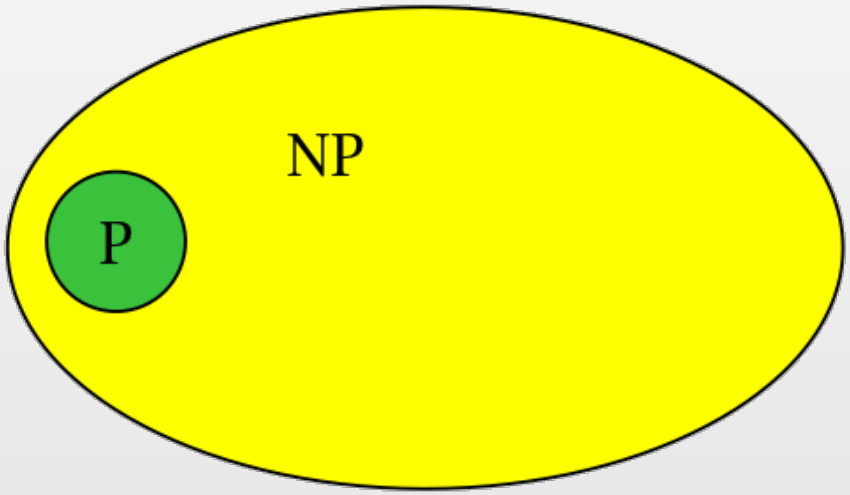
Definição

Um problema Π pertence à classe NP se e somente se para um dado certificado (uma string que certifica a resposta de uma computação), há um algoritmo determinístico que verifica sua validade em tempo polinomial.

Complexidade clássica

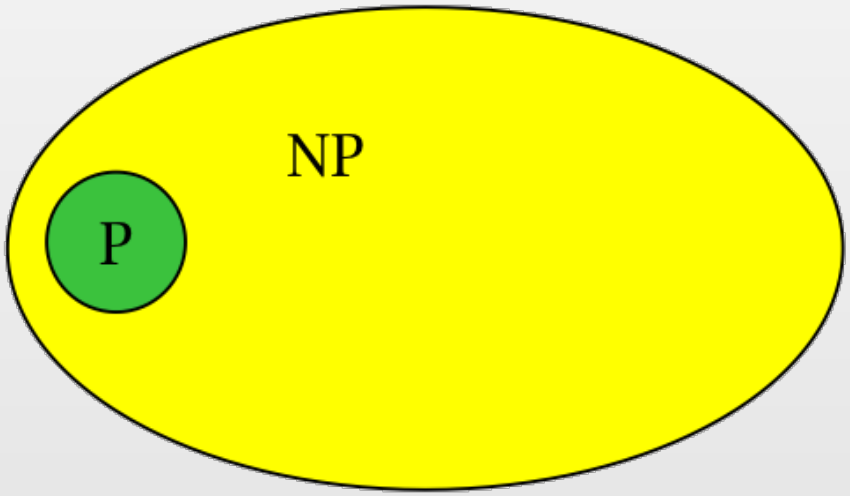


Complexidade clássica



A principal questão em aberto da computação

$$\mathcal{P} = \mathcal{NP}?$$



Reduções

Definição

Dados dois problemas Π *e* Π', $\Pi \propto \Pi'$ *(*Π *se reduz a* Π' *em tempo polinomial) se existe um algoritmo que, dado uma instância* I *de* Π, *constrói uma instância* I' *de* Π' *em tempo polinomial em* $|I|$ *tal que a partir de uma solução para* I', *uma resposta correta para* I *pode ser emitida em tempo polinomial, e vice-versa.*

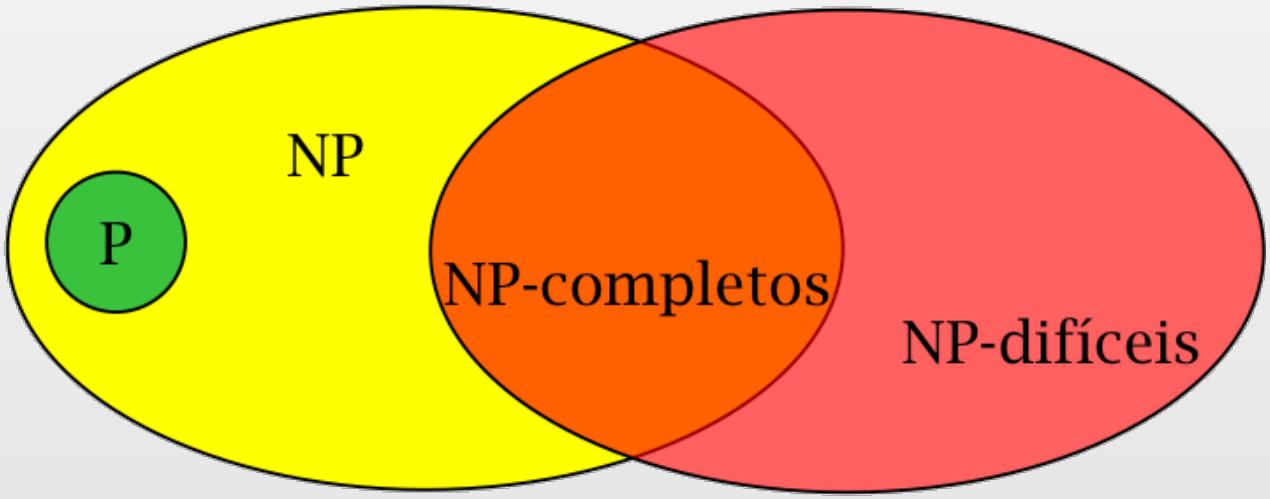
NP-completude

Definição

Um problema Π' *é* NP-difícil *se para todo problema* $\Pi \in NP$, $\Pi \propto \Pi'$; *se* Π' *também está em NP, então* Π' *é NP-completo.*

A principal questão em aberto da computação

$$\mathcal{P} = \mathcal{NP}?$$



Questões

- Como **resolver** um problema **NP-completo** de forma eficiente **na prática**?

Questões

- Como **resolver** um problema **NP-completo** de forma eficiente **na prática**?
- **O que torna** um problema NP-completo, de fato, **difícil** de ser solucionado **na prática**?

Questões

- Como **resolver** um problema **NP-completo** de forma eficiente **na prática**?
- **O que torna** um problema NP-completo, de fato, **difícil** de ser solucionado **na prática**?
- Quais aspectos de um problema NP-completo são suas **fontes de intratabilidade**?

Questões

- Como **resolver** um problema **NP-completo** de forma eficiente **na prática**?
- **O que torna** um problema NP-completo, de fato, **difícil** de ser solucionado **na prática**?
- Quais aspectos de um problema NP-completo são suas **fontes de intratabilidade**?
- Como **explorar** algumas **fontes** de intratabilidade de um problema para **auxiliar** a sua **resolução**?

Questões

- Como **resolver** um problema **NP-completo** de forma eficiente **na prática**?
- **O que torna** um problema NP-completo, de fato, **difícil** de ser solucionado **na prática**?
- Quais aspectos de um problema NP-completo são suas **fontes de intratabilidade**?
- Como **explorar** algumas **fontes** de intratabilidade de um problema para **auxiliar** a sua **resolução**?
- Todo problema NP-completo possui o mesmo **nível de dificuldade**?

Questões

- Como **resolver** um problema **NP-completo** de forma eficiente **na prática**?
- **O que torna** um problema NP-completo, de fato, **difícil** de ser solucionado **na prática**?
- Quais aspectos de um problema NP-completo são suas **fontes de intratabilidade**?
- Como **explorar** algumas **fontes** de intratabilidade de um problema para **auxiliar** a sua **resolução**?
- Todo problema NP-completo possui o mesmo **nível de dificuldade**?
- Como **distinguir diferentes níveis** de dificuldade entre problemas NP-completos?

Questões

- Como **resolver** um problema **NP-completo** de forma eficiente **na prática**?
- **O que torna** um problema NP-completo, de fato, **difícil** de ser solucionado **na prática**?
- Quais aspectos de um problema NP-completo são suas **fontes de intratabilidade**?
- Como **explorar** algumas **fontes** de intratabilidade de um problema para **auxiliar** a sua **resolução**?
- Todo problema NP-completo possui o mesmo **nível de dificuldade**?
- Como **distinguir diferentes níveis** de dificuldade entre problemas NP-completos?
- Quão bom pode ser um algoritmo de **pré-processamento** para um problema NP-completo?

Questões

- Como **resolver** um problema **NP-completo** de forma eficiente **na prática**?
- **O que torna** um problema NP-completo, de fato, **difícil** de ser solucionado **na prática**?
- Quais aspectos de um problema NP-completo são suas **fontes de intratabilidade**?
- Como **explorar** algumas **fontes** de intratabilidade de um problema para **auxiliar** a sua **resolução**?
- Todo problema NP-completo possui o mesmo **nível de dificuldade**?
- Como **distinguir diferentes níveis** de dificuldade entre problemas NP-completos?
- Quão bom pode ser um algoritmo de **pré-processamento** para um problema NP-completo?
- Existe alguma **teoria** para avaliar a **eficiência** de algoritmos de **pré-processamento**?

Complexidade parametrizada

Complexidade parametrizada

Os pais da teoria

Rod Downey & Michael Fellows

Complexidade parametrizada

Os pais da teoria

Rod Downey & Michael Fellows

Uma ideia simples, mas brilhante

Talvez boa parte ou toda a explosão combinatória que precisamos enfrentar para resolver de forma exata uma instância qualquer de um problema intratável, esteja confinada em uma "característica estrutural pequena" dessa instância.

Questões

"Como solucionar um problema NP-difícil da melhor forma possível na prática?"

Questões

"Como solucionar um problema NP-difícil da melhor forma possível na prática?"

Primeiramente, temos duas possibilidades:

Questões

"Como solucionar um problema NP-difícil da melhor forma possível na prática?"

Primeiramente, temos duas possibilidades:

- × – Tentar construir um algoritmo de tempo polinomial (implica $P = NP$).

Questões

"Como solucionar um problema NP-difícil da melhor forma possível na prática?"

Primeiramente, temos duas possibilidades:

- $\times$ – Tentar construir um algoritmo de tempo polinomial (implica $P = NP$).
- $\surd$ – Invocar algum tipo de algoritmo heurístico ou aproximativo, de tempo polinomial.

Questões

"Como solucionar um problema NP-difícil da melhor forma possível na prática?"

Primeiramente, temos duas possibilidades:

- × – Tentar construir um algoritmo de tempo polinomial (implica $P = NP$).
- √ – Invocar algum tipo de algoritmo heurístico ou aproximativo, de tempo polinomial.

- Entretanto, caso estejamos restritos a soluções exatas e ótimas, heurísticas e aproximações podem não ser suficientes.

Questões

"Como solucionar um problema NP-difícil da melhor forma possível na prática?"

Observação

Frequentemente, problemas práticos possuem diversos aspectos adicionais que podem ser considerados como, por exemplo, o grafo de entrada ser planar, as instâncias possuírem grau máximo ou diâmetro constantes, a estrutura buscada possuir tamanho limitado, e assim por diante.

Questões

"Como solucionar um problema NP-difícil da melhor forma possível na prática?"

Observação

Frequentemente, problemas práticos possuem diversos aspectos adicionais que podem ser considerados como, por exemplo, o grafo de entrada ser planar, as instâncias possuírem grau máximo ou diâmetro constantes, a estrutura buscada possuir tamanho limitado, e assim por diante.

Sendo assim, dado que na prática alguns aspectos do problema possuem tamanho ou valor delimitado, existem outras abordagens para a sua resolução:

Questões

"Como solucionar um problema NP-difícil da melhor forma possível na prática?"

× – Invocar algum tipo de técnica de **"força bruta"**, que pode ser executada em tempo polinomial com complexidade $O(n^{f(k)})$, onde n é usado para denotar o tamanho da entrada e k é algum aspecto com tamanho ou valor delimitado. Neste caso, quando as instâncias a serem resolvidas são grandes, esta abordagem pode não ser muito viável.

Questões

"Como solucionar um problema NP-difícil da melhor forma possível na prática?"

√ – Invocar um algoritmo de tempo não polinomial tal que sua complexidade de tempo não polinomial é *apenas* uma função de algum subconjunto de aspectos do problema que possuem tamanho ou valor delimitado na prática.

Questões

Como podemos observar, um problema genérico possui inúmeros aspectos que poderiam ser considerados para análise. No entanto, para cada aplicação existe um subconjunto particular de aspectos do problema que são de fato interessantes.

Questões

Como podemos observar, um problema genérico possui inúmeros aspectos que poderiam ser considerados para análise. No entanto, para cada aplicação existe um subconjunto particular de aspectos do problema que são de fato interessantes.

Parâmetro

Um **parâmetro**, por sua vez, é uma função que extrai um aspecto ou um conjunto de aspectos particulares de um problema; isto é, um parâmetro é um mecânismo para isolar aspectos de um problema e um "receptáculo" no qual aspectos são encapsulados para manipulação e análise subsequente.

Questões

Como podemos observar, um problema genérico possui inúmeros aspectos que poderiam ser considerados para análise. No entanto, para cada aplicação existe um subconjunto particular de aspectos do problema que são de fato interessantes.

Parâmetro

Um **parâmetro**, por sua vez, é uma função que extrai um aspecto ou um conjunto de aspectos particulares de um problema; isto é, um parâmetro é um mecânismo para isolar aspectos de um problema e um "receptáculo" no qual aspectos são encapsulados para manipulação e análise subsequente.

Neste ponto, outras questoes emergem:

Questões

1. Dado um problema e um parâmetro do problema, existe um algoritmo para o problema cuja complexidade de tempo não polinomial é puramente uma função deste parâmetro?
2. Relativo a quais parâmetros do problema tal algoritmo existe?

Complexidade parametrizada

Um parâmetro é aspecto do problema encapsulado para posterior análise.

Complexidade parametrizada

Um parâmetro é aspecto do problema encapsulado para posterior análise.

Problema parametrizado

1. A parte principal da entrada do problema.
2. Alguns aspectos do problema, que constituem o(s) parâmetro(s).
3. A questão a ser respondida.

Complexidade parametrizada

Um parâmetro é aspecto do problema encapsulado para posterior análise.

Problema parametrizado

1. A parte principal da entrada do problema.
2. Alguns aspectos do problema, que constituem o(s) parâmetro(s).
3. A questão a ser respondida.

Definição

Sejam

- Π *um problema NP-difícil;*
- $S = \{a_1, a_2, \ldots, a_\ell\}$ *um subconjunto de aspectos de* Π

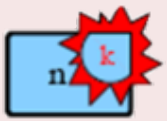
Denotamos por:

- $\Pi(a_1, a_2, \ldots, a_\ell)$*, ou* $\Pi(S)$*, a versão parametrizada de* Π *onde os aspectos em* S *são parâmetros fixados.*

Complexidade parametrizada

Tratabilidade por parâmetro fixo

- Um problema parametrizado $\Pi(k)$ pertence à classe FPT se e somente se existe um algoritmo determinístico que o resolva em tempo $f(k).n^{O(1)}$.

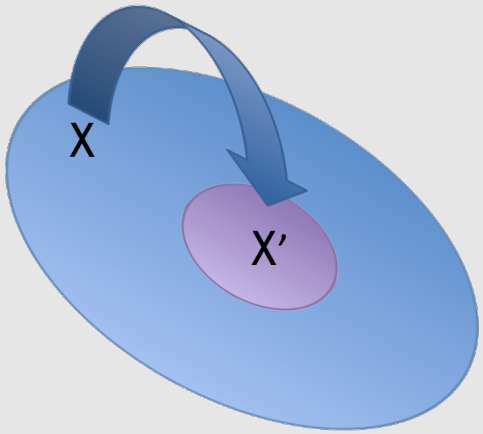


Complexidade parametrizada na prática

Caracterização da classe FPT

Redução a um núcleo (kernelization):

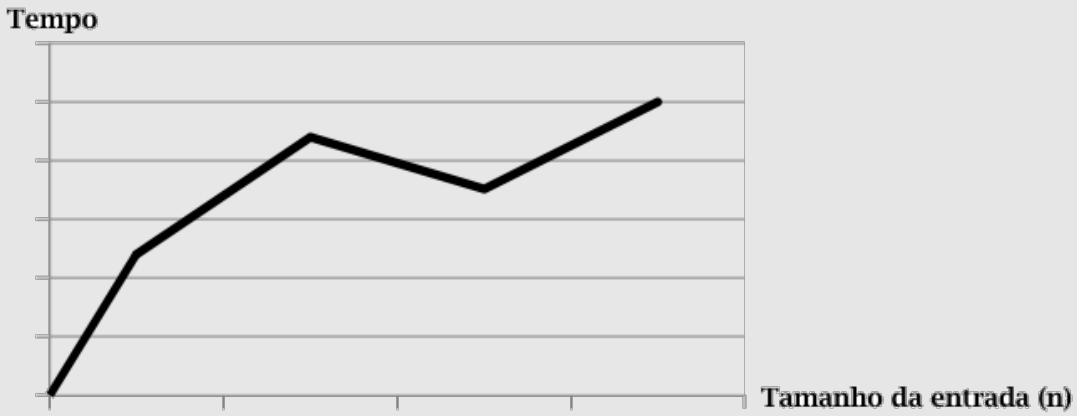
- Um problema parametrizado $\Pi(k)$ pertence à classe FPT se e somente se existe um algoritmo de **pré-processamento** que em tempo **polinomial** reduz a instância original do problema a uma outra instância cujo **tamanho é uma função de k**.



$$|x'| = f(k)$$

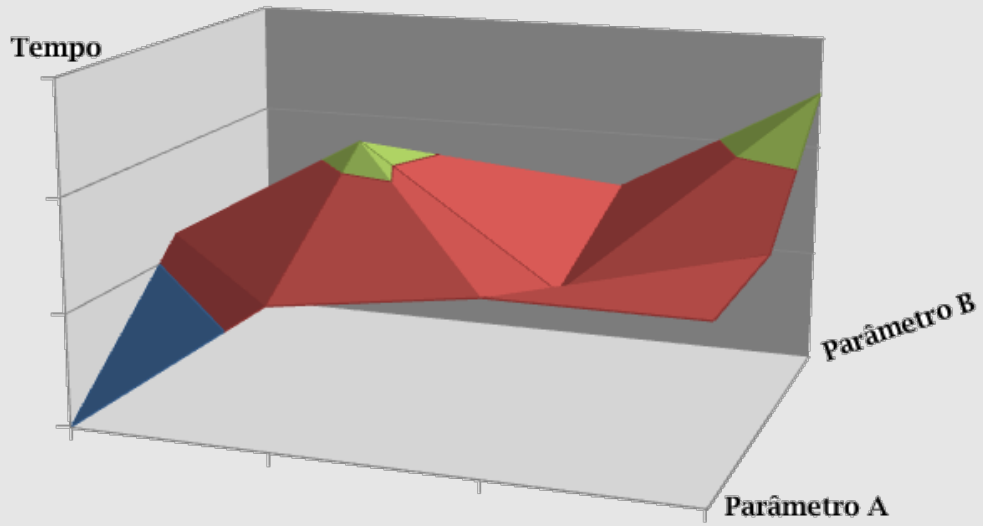
Complexidade clássica × complexidade parametrizada

Análise clássica



Complexidade clássica × complexidade parametrizada

Análise multivariada



Complexidade parametrizada

Tratabilidade por parâmetro fixo

- Um problema parametrizado $\Pi(k)$ pertence à classe FPT se e somente se existe um algoritmo determinístico que o resolva em tempo $f(k).n^{O(1)}$.

Dado um problema NP-difícil Π e algum subconjunto de aspectos S de Π, dizemos que S é uma **uma fonte de intratabilidade de tempo polinomial** para Π, se $\Pi(S)$ pertence a FPT.

Complexidade parametrizada

Tratabilidade por parâmetro fixo

- Um problema parametrizado $\Pi(k)$ pertence à classe FPT se e somente se existe um algoritmo determinístico que o resolva em tempo $f(k).n^{O(1)}$.

Dado um problema NP-difícil Π e algum subconjunto de aspectos S de Π, dizemos que S é uma **uma fonte de intratabilidade de tempo polinomial** para Π, se $\Pi(S)$ pertence a FPT.

> *" Na teoria da complexidade parametrizada, o foco não está em verificar* se *um problema é difícil, a teoria parte da suposição que os problemas mais interessantes são intratáveis quando considerados classicamente. O foco desta teoria está na seguinte questão:* O que faz o problema computacionalmente difícil? *".*

Como provar que um problema pertence à classe FPT?

Primeiro passo:

Como provar que um problema pertence à classe FPT?

Primeiro passo:

$$FPT \subseteq XP$$

Como provar que um problema pertence à classe FPT?

Primeiro passo:

$$FPT \subseteq XP$$

verificar se o problema parametrizado em questão, $\Pi(k)$, pertence à classe XP.

A classe XP

Definição

Um problema parametrizado $\Pi(S)$ pertence à classe XP se existe um algoritmo para solucionar $\Pi(S)$ em tempo $f(S).n^{g(S)}$, onde n é usado para denotar o tamanho da entrada e f e g são funções arbitrárias.

A classe XP

Lema

Dado um problema NP-difícil Π *e um subconjunto* S *de seus aspectos, se* Π *permanece NP-difícil mesmo quando os aspectos em* S *são delimitados por constantes, então um problema parametrizado* $\Pi(S)$ *não está em XP, a menos que* $P = NP$.

A classe XP

Lema

Dado um problema NP-difícil Π *e um subconjunto* S *de seus aspectos, se* Π *permanece NP-difícil mesmo quando os aspectos em* S *são delimitados por constantes, então um problema parametrizado* $\Pi(S)$ *não está em XP, a menos que* $P = NP$.

Prova. Se Π está em XP então por definição este problema é solucionado por um algoritmo que pode ser executado em tempo $f(S)n^{g(S)}$ para algumas funções f e g. Quando o valor de todos os aspectos em S são delimitados por uma constate, os valores de $f(S)$ e $g(S)$ são constantes e este tempo de execução torna-se polinomial em n. Como Π é NP-difícil então $P = NP$.

A classe XP

Corolário

Se $P \neq NP$, então $\Pi(S)$ pertence a XP se e somente se Π é solucionável em tempo polinomial quando os aspectos em S são delimitados por constantes.