Uma apresentação a Algoritmos Parametrizados

Introdução

Lehilton Pedrosa

Junho de 2017

Instituto de Computação – Unicamp

Roteiro

1. Introdução
2. Preprocessamento
3. Decomposição em Coroa
4. Ramificação

Introdução

Computação = Algoritmos e Complexidade

Computação é a ciência de resolver problemas!

> *"Beating heart of Computing is algorithms and complexity"*
>
> *Downey, Fellows*

Um problema exemplo: *race condition*

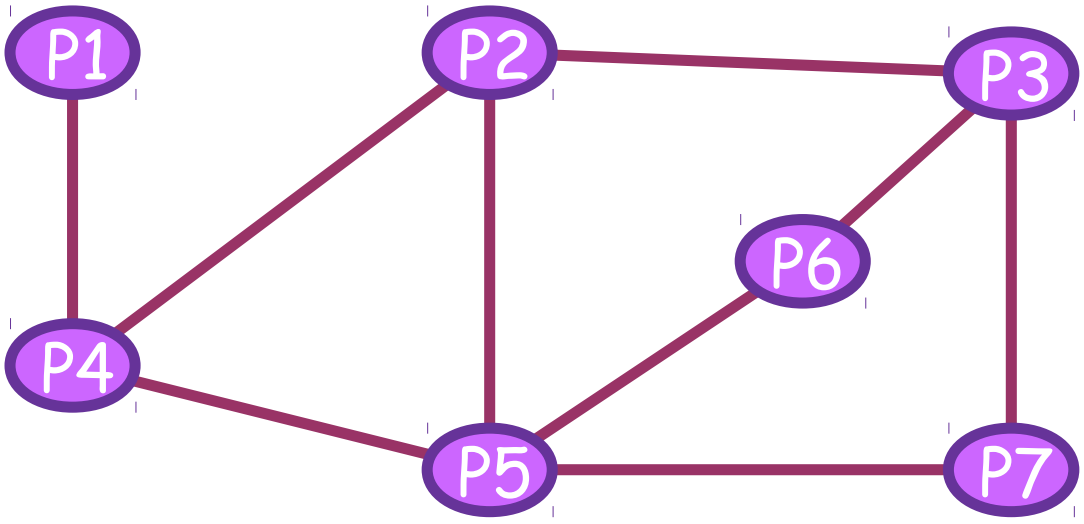
Descrição:

Um conjunto de processos precisa ser executado, mas alguns deles compartilham os mesmos arquivos. Se dois processos acessarem o mesmo arquivos simultaneamente, pode haver colisão.

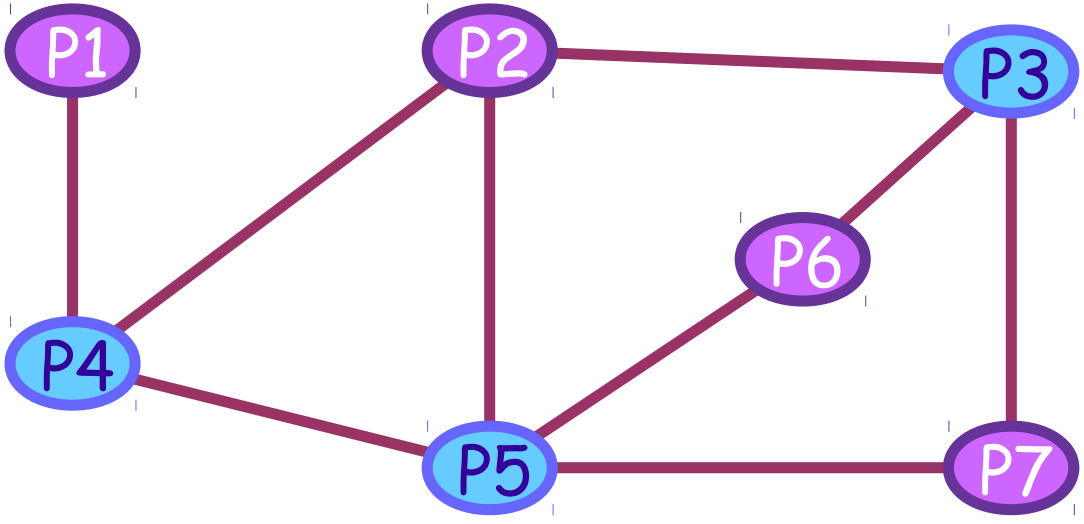
Perguntas possíveis:

1. Otimização: Qual o menor número k de processos eu preciso enfileirar?
2. Decisão: É possível enfileirar só k processos?

Melhor com um desenho



Melhor com um desenho



Problema mais conhecido como Cobertura por vértices

Algoritmo natural para Cobertura por Vértices

Problema: Cobertura por Vértices é NP-difícil

Algoritmo natural para Cobertura por Vértices

Problema: Cobertura por Vértices é NP-difícil

ExisteCobertura(G, k):

1. Para cada subconjunto $C \subseteq V(G)$:
   - Se $|C| = k$ e $G - C$ não contém arestas,
     - responda sim
2. Responda não

Complexidade:

- testamos $\binom{n}{k}$ possibilidades

Algoritmo natural para Cobertura por Vértices

Problema: Cobertura por Vértices é NP-difícil

ExisteCobertura(G, k):

1. Para cada subconjunto $C \subseteq V(G)$:
   - Se $|C| = k$ e $G - C$ não contém arestas,
     - responda sim
2. Responda não

Complexidade:

- testamos $\binom{n}{k}$ possibilidades
- e se $k = 10$?

Motivação: nem tão ruim assim

Na prática, o número de processos que precisamos enfileirar é “pequeno”

Motivação: nem tão ruim assim

Na prática, o número de processos que precisamos enfileirar é “pequeno”

Pequeno: menor que uma constante

Exemplo: $k \leq 10 \Rightarrow$ algoritmo polinomial: $\binom{n}{k} \leq \mathcal{O}(n^k)$

Motivação: nem tão ruim assim

Na prática, o número de processos que precisamos enfileirar é “pequeno”

Pequeno: menor que uma constante

Exemplo: $k \leq 10 \Rightarrow$ algoritmo polinomial: $\binom{n}{k} \leq \mathcal{O}(n^k)$

Rápido?

Testando um bilhão de possibilidades por segundo!

n	$\binom{n}{k}$	tempo gasto
100	$\binom{100}{10} \approx 1{,}73 \times 10^{13}$	mais de 4 horas

Motivação: nem tão ruim assim

Na prática, o número de processos que precisamos enfileirar é “pequeno”

Pequeno: menor que uma constante

Exemplo: $k \leq 10 \Rightarrow$ algoritmo polinomial: $\binom{n}{k} \leq \mathcal{O}(n^k)$

Rápido?

Testando um bilhão de possibilidades por segundo!

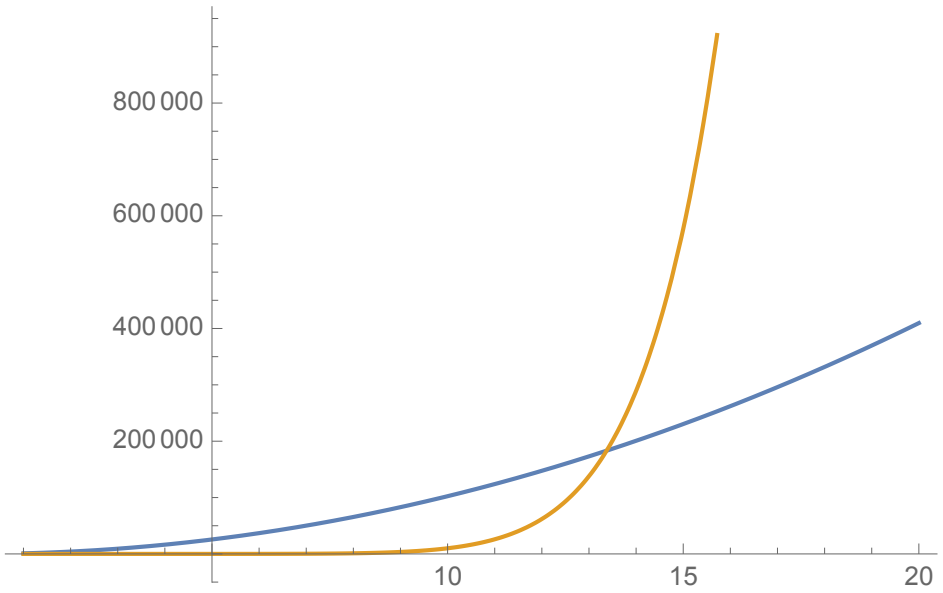
n	$\binom{n}{k}$	tempo gasto
100	$\binom{100}{10} \approx 1{,}73 \times 10^{13}$	mais de 4 horas
1000	$\binom{1000}{10} \approx 2{,}63 \times 10^{23}$	8 milhões de anos

Separando a dependência de k e n

- Suponha $k = 10$
- Comparando $2^k n^2$ versus $\frac{n^k}{10000000}$:

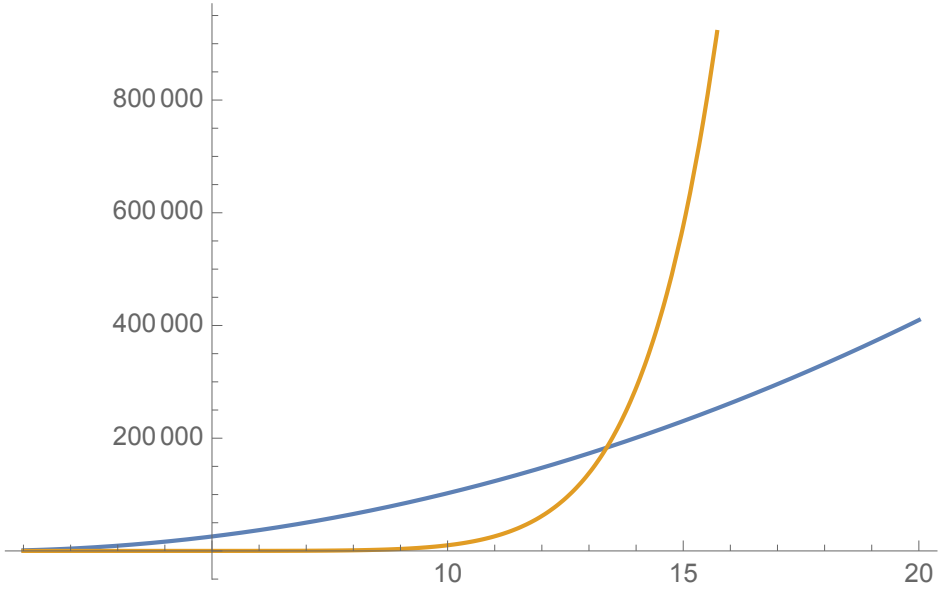
Separando a dependência de k e n

- Suponha $k = 10$
- Comparando $2^k n^2$ versus $\frac{n^k}{10000000}$:



Separando a dependência de k e n

- Suponha $k = 10$
- Comparando $2^k n^2$ versus $\frac{n^k}{10000000}$:



Queremos funções do primeiro tipo!

Problemas parametrizados

Definição (Problema parametrizado)

Um problema parametrizado $P \subseteq \Sigma^* \times \mathbb{N}$ é um problema de de decisão em que cada instância contém um parâmetro k associado, que é um número inteiro.

- Representamos uma instância como $\langle x, k \rangle \in P$
- O parâmetro k é uma medida da instância x

Problemas parametrizados

Definição (Problema parametrizado)

Um problema parametrizado $P \subseteq \Sigma^* \times \mathbb{N}$ é um problema de de decisão em que cada instância contém um parâmetro k associado, que é um número inteiro.

- Representamos uma instância como $\langle x, k \rangle \in P$
- O parâmetro k é uma medida da instância x

Exemplos de parâmetros:

- um valor explícito na instância:
  - o tamanho na Cobertura por vértices
  - a tamanho da mochila

Problemas parametrizados

Definição (Problema parametrizado)

Um problema parametrizado $P \subseteq \Sigma^* \times \mathbb{N}$ é um problema de de decisão em que cada instância contém um parâmetro k associado, que é um número inteiro.

- Representamos uma instância como $\langle x, k \rangle \in P$
- O parâmetro k é uma medida da instância x

Exemplos de parâmetros:

- um valor explícito na instância:
  - o tamanho na Cobertura por vértices
  - a tamanho da mochila
- uma medida estrutural da instância:
  - propriedades de grafo: largura arbórea, grau máximo
  - tamanho mínimo de um item

Análise multivariada

Informalmente: um problema é tratável por parâmetro fixo (**FPT**) se tiver algoritmo com tempo $O(f(k)n^{\mathcal{O}(1)})$

Análise multivariada

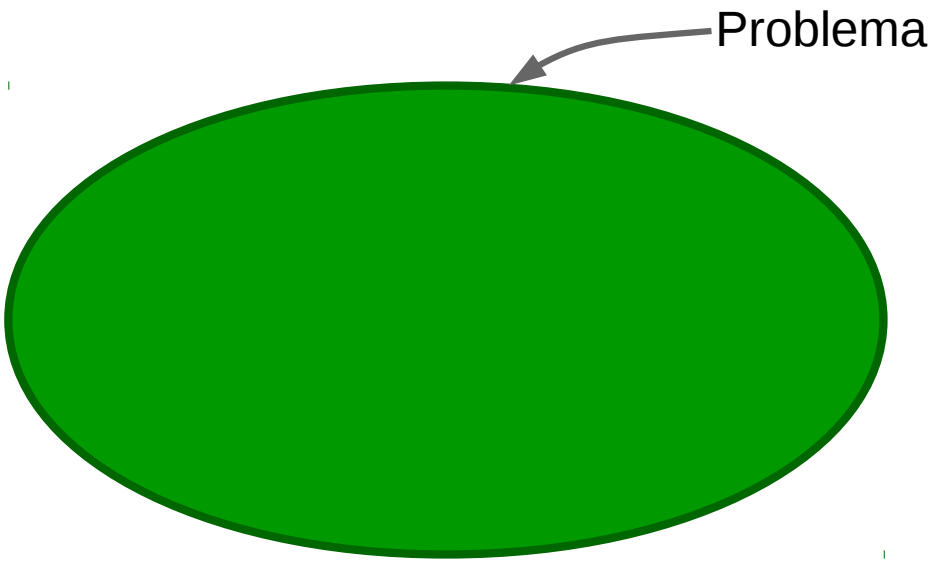
Informalmente: um problema é tratável por parâmetro fixo (**FPT**) se tiver algoritmo com tempo $O(f(k)n^{\mathcal{O}(1)})$

- consideramos subproblemas do problema original

Análise multivariada

Informalmente: um problema é tratável por parâmetro fixo (**FPT**) se tiver algoritmo com tempo $O(f(k)n^{\mathcal{O}(1)})$

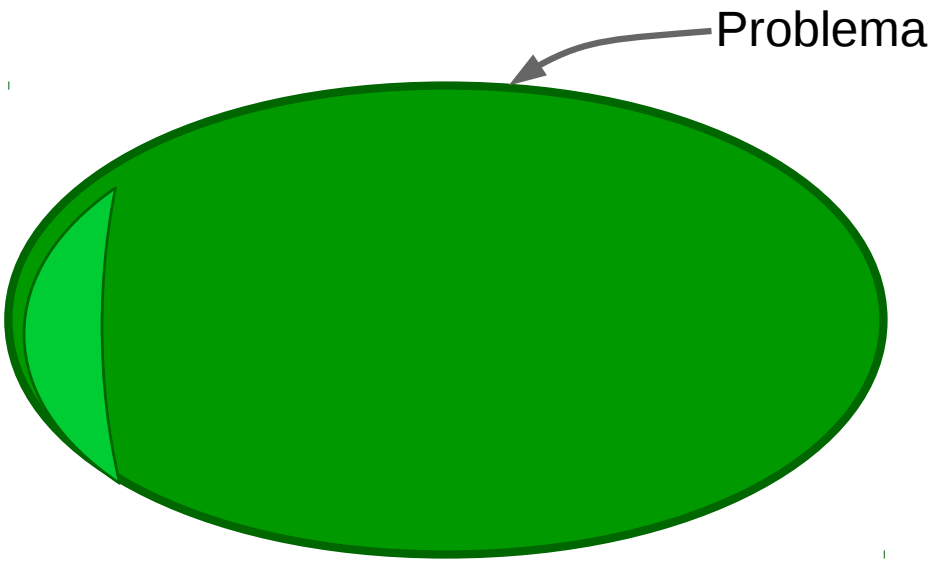
- consideramos subproblemas do problema original



Análise multivariada

Informalmente: um problema é tratável por parâmetro fixo (**FPT**) se tiver algoritmo com tempo $O(f(k)n^{\mathcal{O}(1)})$

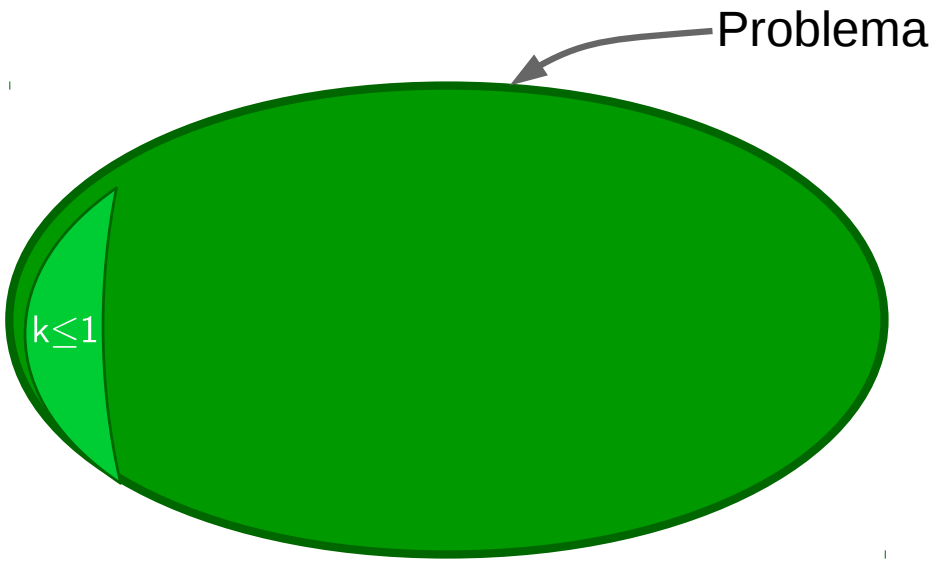
- consideramos subproblemas do problema original



Análise multivariada

Informalmente: um problema é tratável por parâmetro fixo (**FPT**) se tiver algoritmo com tempo $O(f(k)n^{\mathcal{O}(1)})$

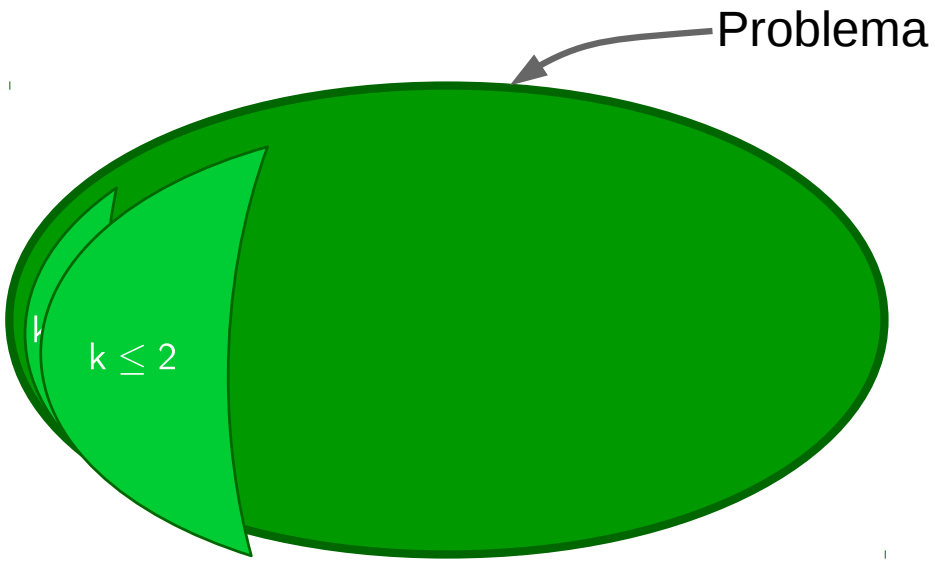
- consideramos subproblemas do problema original



Análise multivariada

Informalmente: um problema é tratável por parâmetro fixo (**FPT**) se tiver algoritmo com tempo $O(f(k)n^{\mathcal{O}(1)})$

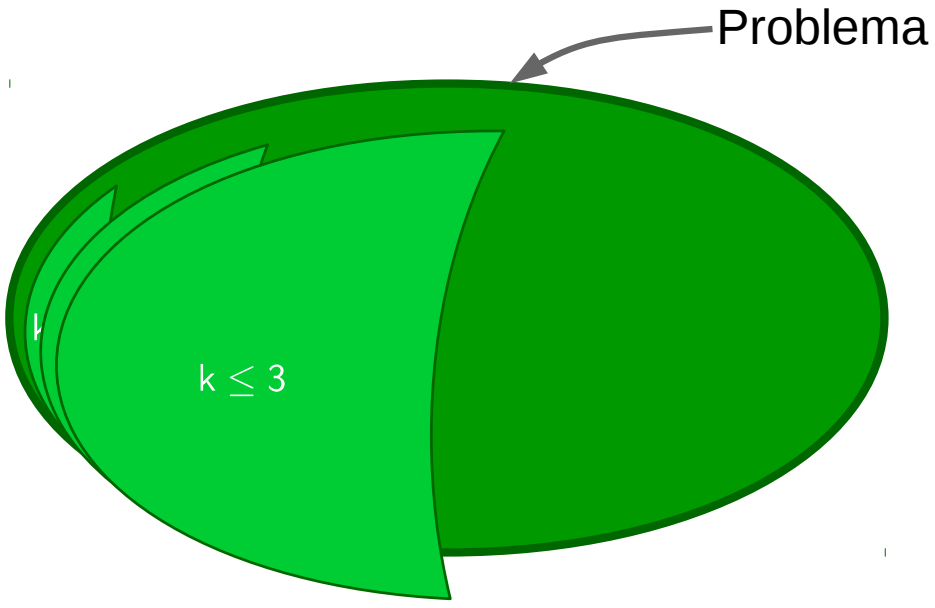
- consideramos subproblemas do problema original



Análise multivariada

Informalmente: um problema é tratável por parâmetro fixo (**FPT**) se tiver algoritmo com tempo $O(f(k)n^{\mathcal{O}(1)})$

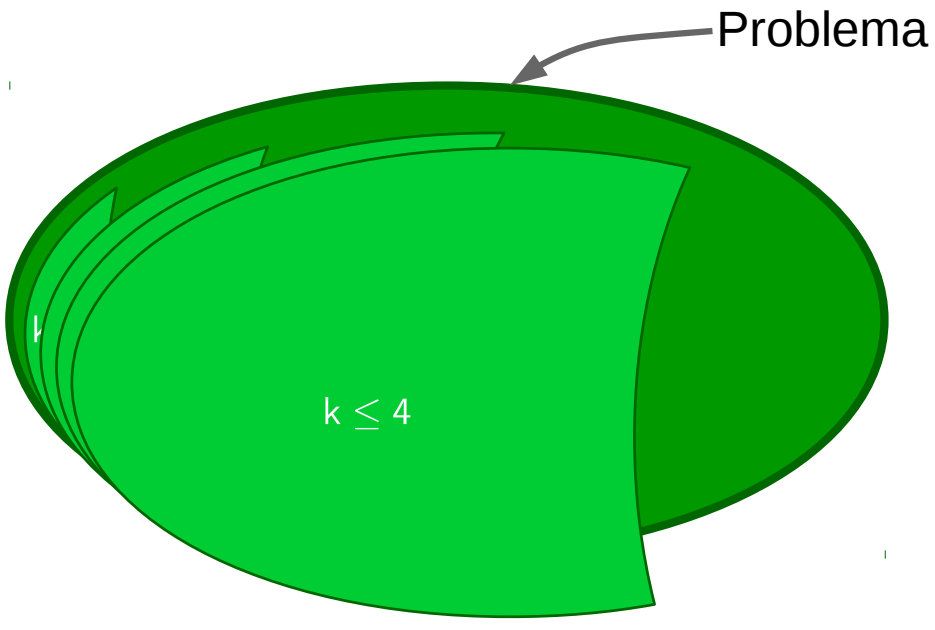
- consideramos subproblemas do problema original



Análise multivariada

Informalmente: um problema é tratável por parâmetro fixo (**FPT**) se tiver algoritmo com tempo $O(f(k)n^{\mathcal{O}(1)})$

- consideramos subproblemas do problema original



Isolando a dificuldade

Ideia principal: isolamos o parâmetro que contribui fundamentalmente para a dificuldade:

Isolando a dificuldade

Ideia principal: isolamos o parâmetro que contribui fundamentalmente para a dificuldade:

Entrada: x

```
11101100101011000111111000011
01011011100000100011100100001
00001110100000111011111101000
10000110011110001100100101000
00110001110101011100110111111
01010110001111111011111000011
01010101110111001100001101010
11100000101001100110001011010
10100111001100000110001001111
```

Isolando a dificuldade

Ideia principal: isolamos o parâmetro que contribui fundamentalmente para a dificuldade:

Entrada: x, $n := |x|$ bits

```
11101100101011000111111000011
01011011100000100011100100001
00001110100000111011111101000
10000110011110001100100101000
00110001110101011100110111111
01010110001111111011111000011
01010101110111001100001101010
11100000101001100110001011010
10100111001100000110001001111
```

Isolando a dificuldade

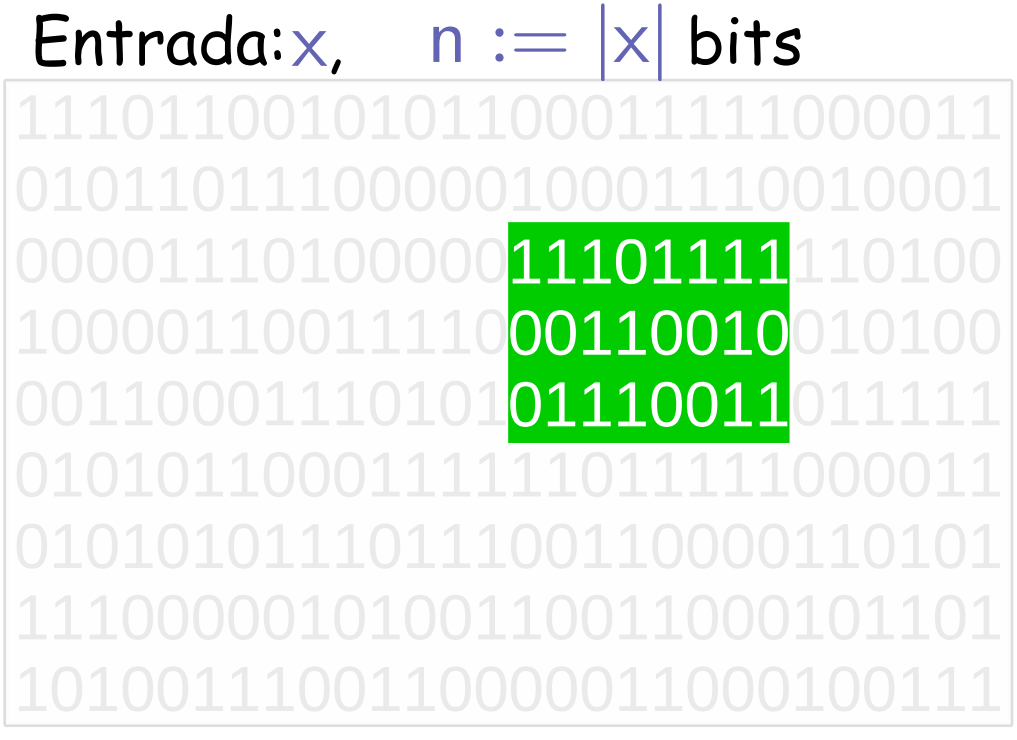
Ideia principal: isolamos o parâmetro que contribui fundamentalmente para a dificuldade:

Entrada: x, $n := |x|$ bits

```
11101100101011000111111000011
01011011100000100011100100001
00001110100000111011111110100
10000110011111000110010010100
00110001110101011100111011111
01010110001111111011111000011
01010101110111001100001101011
11100000101001100110001011011
10100111001100000110001001111
```

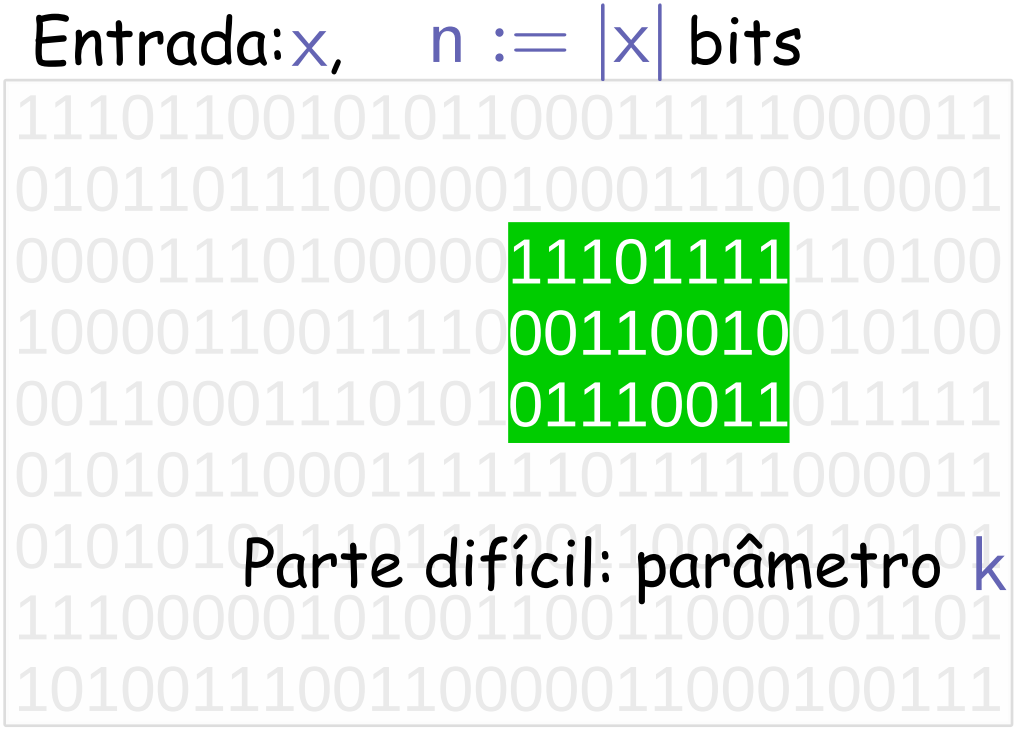
Isolando a dificuldade

Ideia principal: isolamos o parâmetro que contribui fundamentalmente para a dificuldade:



Isolando a dificuldade

Ideia principal: isolamos o parâmetro que contribui fundamentalmente para a dificuldade:



Uma mudança de paradigma

- 1992: trabalhos de (in)tratabilidade com parâmetros fixos
- 1999: livro de Downey e Fellows

Muitos recursos disponíveis

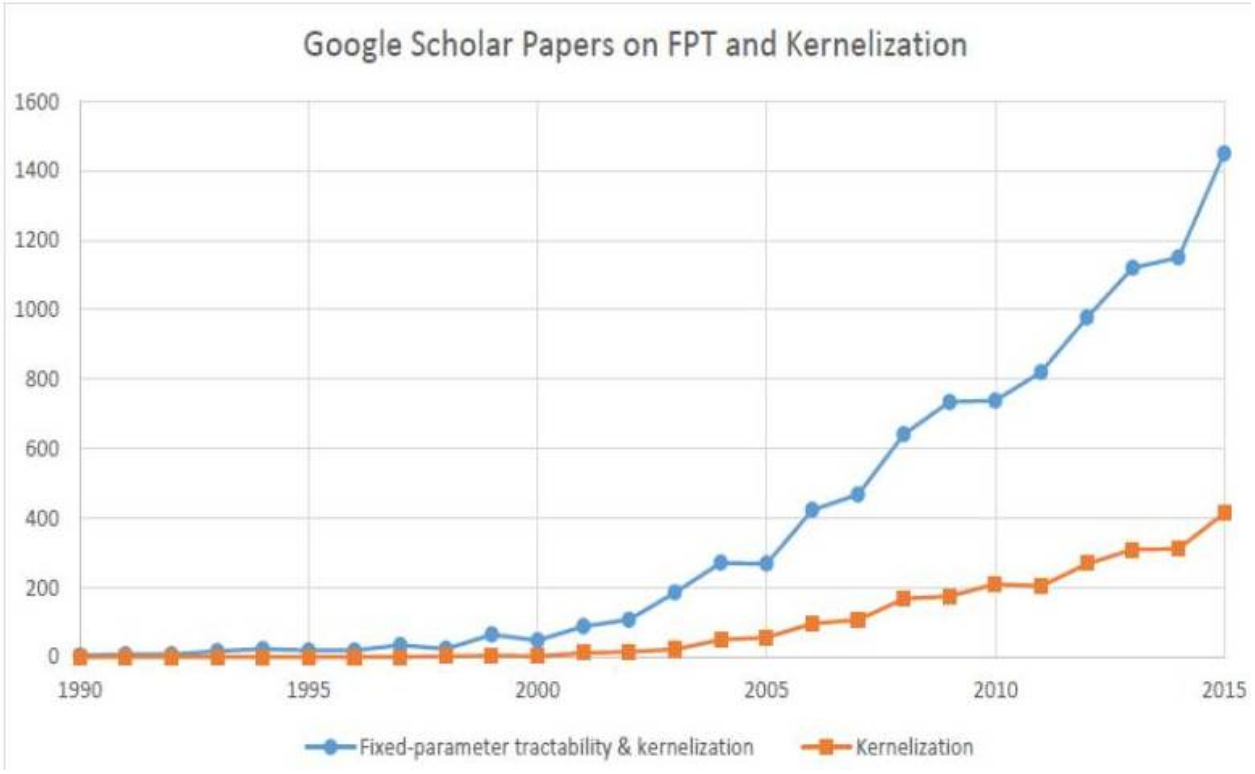
- Diversos livros
- Wiki de parametrizados: `http://fpt.wikidot.com/`
- Escolas: `http://fptschool.mimuw.edu.pl/`

O futuro?

"The future of algorithmics is multivariate"

Downey, Fellows

- Conferências especializadas: IPEC
- Diversos artigos: ICALP, STOC, FOCS, ESA, LATIN...



(Fonte: Bart Jansen, http://fpt.wikidot.com/)

Formalizando FPT

Definição (*Fixed Parameter Tractable*)

Dizemos que um problema parametrizado $P \subseteq \Sigma^* \times \mathrm{N}$ é tratável por parâmetro fixo (**FPT**) se, e somente se, existe um algoritmo A, uma constante c e uma função computável f tais que, para todo par $\langle x, k \rangle \in \Sigma^* \times \mathrm{N}$,

1. $A(\langle x, k \rangle)$ executa em tempo $f(k)|x|^c$;
2. $\langle x, k \rangle \in P$ sss $A(\langle x, k \rangle) = 1$.

Formalizando FPT

Definição (*Fixed Parameter Tractable*)

Dizemos que um problema parametrizado $P \subseteq \Sigma^* \times \mathrm{N}$ é tratável por parâmetro fixo (**FPT**) se, e somente se, existe um algoritmo A, uma constante c e uma função computável f tais que, para todo par $\langle x, k \rangle \in \Sigma^* \times \mathrm{N}$,

1. $A(\langle x, k \rangle)$ executa em tempo $f(k)|x|^c$;
2. $\langle x, k \rangle \in P$ sss $A(\langle x, k \rangle) = 1$.

Notação: Para funções $g : \mathrm{N} \times \mathrm{N} \to \mathrm{N}$ e $f : \mathrm{N} \to \mathrm{N}$, escrevemos:

$g(k, n) = \mathcal{O}^*(f(k))$ sss existe constante c tal que $g(k, n) = \mathcal{O}(f(k)\, n^c)$.

Preprocessamento

Separando a dificuldade

Considere um problema $\mathcal{NP}$-difícil

- Nem toda instância é “difícil”
- Parte de uma instância pode ser “fácil”

Separando a dificuldade

Considere um problema $\mathcal{NP}$-difícil

- Nem toda instância é "difícil"
- Parte de uma instância pode ser "fácil"

Ideia: se concentrar na dificuldade

Separando a dificuldade

Considere um problema $\mathcal{NP}$-difícil

- Nem toda instância é "difícil"
- Parte de uma instância pode ser "fácil"

Ideia: se concentrar na dificuldade

Pré-processamento

- resolver instâncias fáceis
- revolver a parte fácil

Separando a dificuldade

Considere um problema $\mathcal{NP}$-difícil

- Nem toda instância é "difícil"
- Parte de uma instância pode ser "fácil"

Ideia: se concentrar na dificuldade

Pré-processamento

- resolver instâncias fáceis
- revolver a parte fácil
- diminuir o tamanho da instância

Pré-processamento

Visão clássica

- rápido: polinomial

Pré-processamento

Visão clássica

- rápido: polinomial
- efetivo: diminui o tamanho da instância

Pré-processamento

Visão clássica

- rápido: polinomial
- efetivo: diminui o tamanho da instância

Não é muito útil:

- se P tem pre-processamento $\Rightarrow$ então P é polinomial
- se P é $\mathcal{NP}$-difícil $\Rightarrow \mathcal{P} = \mathcal{NP}$!

Pré-processamento

Visão clássica

- rápido: polinomial
- efetivo: diminui o tamanho da instância

Não é muito útil:

- se P tem pre-processamento $\Rightarrow$ então P é polinomial
- se P é $\mathcal{NP}$-difícil $\Rightarrow \mathcal{P} = \mathcal{NP}$!

Visão **parametrizada**

- rápido: polinomial
- efetivo: diminui o tamanho da instância

Pré-processamento

Visão clássica

- rápido: polinomial
- efetivo: diminui o tamanho da instância

Não é muito útil:

- se P tem pre-processamento $\Rightarrow$ então P é polinomial
- se P é $\mathcal{NP}$-difícil $\Rightarrow \mathcal{P} = \mathcal{NP}$!

Visão **parametrizada**

- rápido: polinomial
- efetivo: diminui o tamanho da instância:
  se a instância já não for muito pequena

Pré-processamento

Visão clássica

- rápido: polinomial
- efetivo: diminui o tamanho da instância

Não é muito útil:

- se P tem pre-processamento $\Rightarrow$ então P é polinomial
- se P é $\mathcal{NP}$-difícil $\Rightarrow \mathcal{P} = \mathcal{NP}$!

Visão **parametrizada**

- rápido: polinomial
- efetivo: diminui o tamanho da instância:
  se a instância já não for muito pequena

Recuperamos o *lost continent* do pré-processamento!

Formalizando: redução

- Seja $Q \subseteq \Sigma^* \times \mathbb{N}$ um problema parametrizado

Definição (Regra de redução de dados)

Uma regra de redução é uma transformação $\mathcal{A} : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ tal que, dada uma instância $(x, k) \in \Sigma^* \times \mathbb{N}$,

1. $\mathcal{A}$ executa em tempo polinomial em $|x|$ e k;
2. $(x, k) \in Q$ sss $\mathcal{A}(x, k) \in Q$.

Formalizando: redução

- Seja $Q \subseteq \Sigma^* \times \mathrm{N}$ um problema parametrizado

Definição (Regra de redução de dados)

Uma regra de redução é uma transformação $\mathcal{A} : \Sigma^* \times \mathrm{N} \to \Sigma^* \times \mathrm{N}$ tal que, dada uma instância $(x, k) \in \Sigma^* \times \mathrm{N}$,

1. $\mathcal{A}$ executa em tempo polinomial em $|x|$ e k;
2. $(x, k) \in Q$ sss $\mathcal{A}(x, k) \in Q$.

Uma transformação que satisfaz a segunda condição é dita segura.

Formalizando: Kernelização (núcleo)

Dado um algoritmo de pré-processamento A, definimos o tamanho de saída

$$\text{size}_{\mathcal{A}}(k) = \sup\{\, |x'| + k' \;:\; (x', k') = \mathcal{A}(x, k),\, x \in \Sigma^* \,\}.$$

Formalizando: Kernelização (núcleo)

Dado um algoritmo de pré-processamento A, definimos o tamanho de saída

$$\mathrm{size}_{\mathcal{A}}(k) = \sup\{\, |x'| + k' \;:\; (x', k') = \mathcal{A}(x, k),\, x \in \Sigma^* \,\}.$$

Note que $\mathrm{size}_{\mathcal{A}}(k) = \infty$ quando $|x'|$ não depende de k.

Formalizando: Kernelização (núcleo)

Dado um algoritmo de pré-processamento A, definimos o tamanho de saída

$$\text{size}_{\mathcal{A}}(k) = \sup\{\, |x'| + k' \;:\; (x', k') = \mathcal{A}(x, k),\, x \in \Sigma^* \,\}.$$

Note que $\text{size}_{\mathcal{A}}(k) = \infty$ quando $|x'|$ não depende de k.

Definição (Kernelização)

Um algoritmo de kernelização ou núcleo para um problema parametrizado Q é uma transformação $\mathcal{A}$ que mapeia $(x, k) \in \Sigma^* \times \mathrm{N}$ em $(x', k') \in \Sigma^* \times \mathrm{N}$ tal que

Formalizando: Kernelização (núcleo)

Dado um algoritmo de pré-processamento A, definimos o tamanho de saída

$$\text{size}_{\mathcal{A}}(k) = \sup\{\, |x'| + k' \;:\; (x', k') = \mathcal{A}(x, k),\, x \in \Sigma^* \,\}.$$

Note que $\text{size}_{\mathcal{A}}(k) = \infty$ quando $|x'|$ não depende de k.

Definição (Kernelização)

Um algoritmo de kernelização ou núcleo para um problema parametrizado Q é uma transformação $\mathcal{A}$ que mapeia $(x, k) \in \Sigma^* \times \mathrm{N}$ em $(x', k') \in \Sigma^* \times \mathrm{N}$ tal que

1. $\mathcal{A}$ executa em tempo polinomial;
2. $(x, k) \in Q$ sss $\mathcal{A}(x, k) \in Q$;
3. $\text{size}_{\mathcal{A}}(k) \leq g(k)$ para alguma função computável $g(k)$.

Observações

- Queremos núcleos com $g(k)$ menor possível:
  - se $g(k) \leq k^c$, o problema admite um núcleo polinomial

Observações

- Queremos núcleos com $g(k)$ menor possível:
  - se $g(k) \leq k^c$, o problema admite um núcleo polinomial
- O algoritmo de kernelização pode devolver sim ou não:
  - quando o pré-processamento já decide a instância
  - formalmente, substituímos por uma instância trivial

Observações

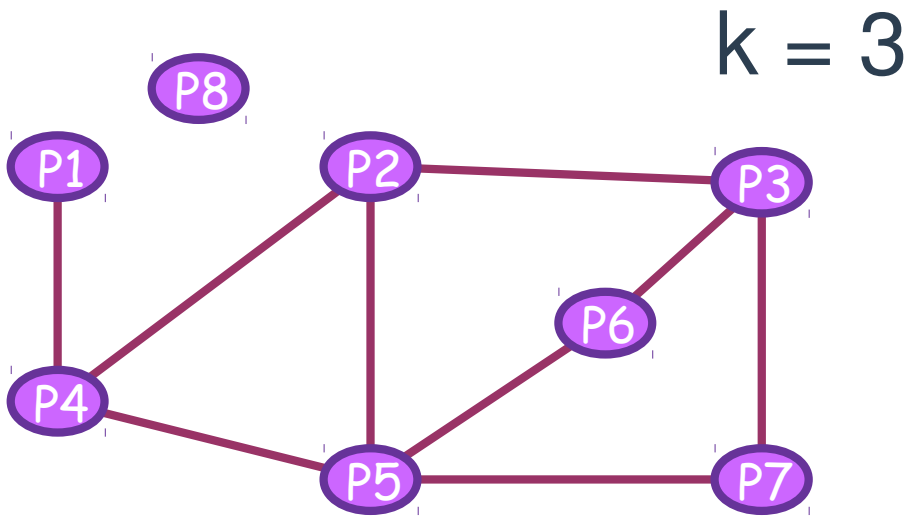
- Queremos núcleos com $g(k)$ menor possível:
  - se $g(k) \leq k^c$, o problema admite um núcleo polinomial
- O algoritmo de kernelização pode devolver sim ou não:
  - quando o pré-processamento já decide a instância
  - formalmente, substituímos por uma instância trivial
- Podemos medir uma instância reduzida pelo parâmetro:
  - ex: $\mathcal{O}(k^3)$ vértices, ou $\mathcal{O}(k^5)$ arestas

Observações

- Queremos núcleos com $g(k)$ menor possível:
  - se $g(k) \leq k^c$, o problema admite um núcleo polinomial
- O algoritmo de kernelização pode devolver sim ou não:
  - quando o pré-processamento já decide a instância
  - formalmente, substituímos por uma instância trivial
- Podemos medir uma instância reduzida pelo parâmetro:
  - ex: $\mathcal{O}(k^3)$ vértices, ou $\mathcal{O}(k^5)$ arestas
- O parâmetro k' da instância reduzida pode mudar:
  - normalmente temos $k' \leq k$

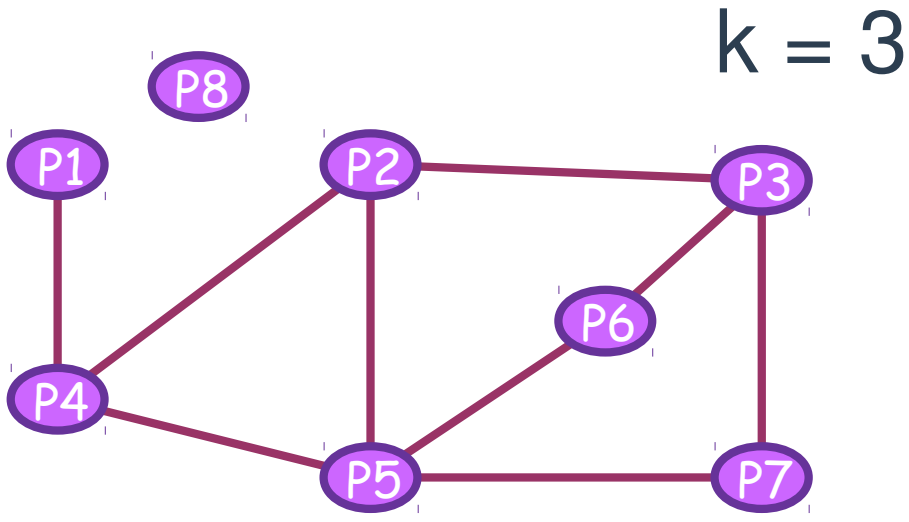
Voltando ao nosso problema

Voltando ao nosso problema



Perguntas:

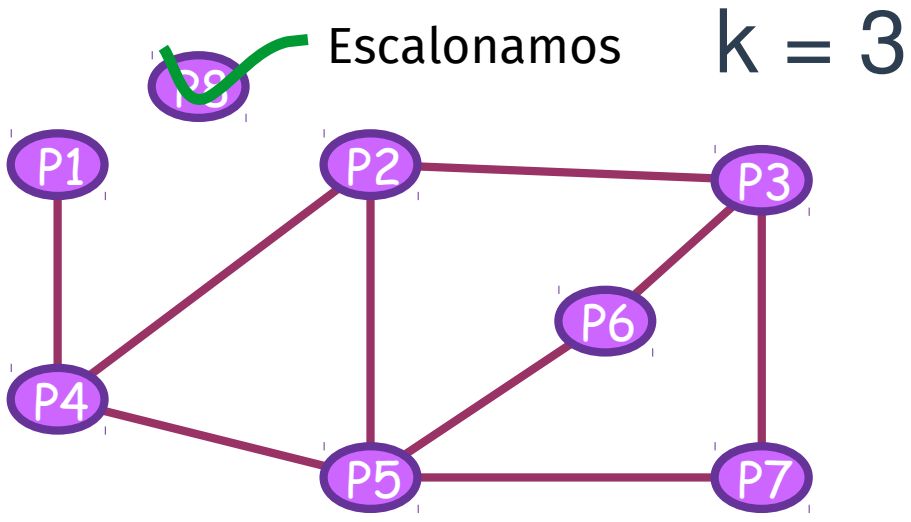
Voltando ao nosso problema



Perguntas:

1. O processo P8 será enfileirado?

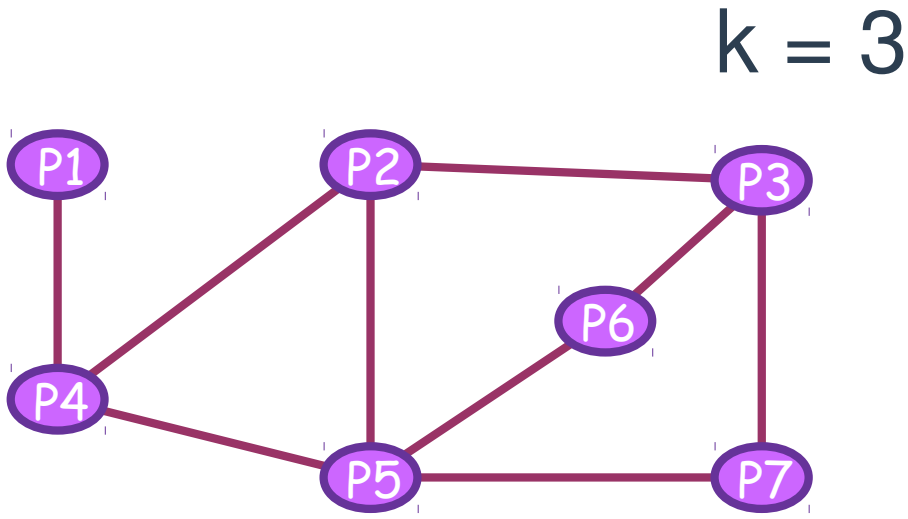
Voltando ao nosso problema



Perguntas:

1. O processo P8 será enfileirado?
    - não, um vértice isolado não tem conflito k^2 arestas

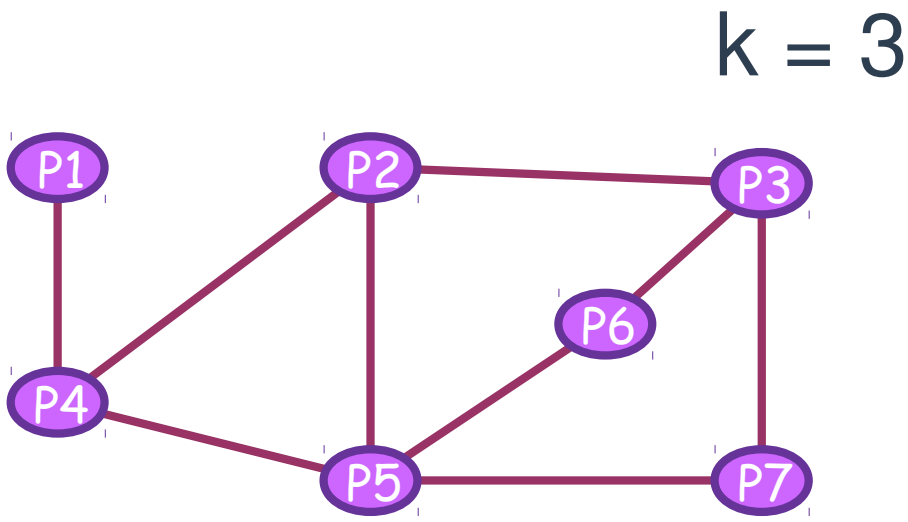
Voltando ao nosso problema



Perguntas:

1. O processo P8 será enfileirado?
    - não, um vértice isolado não tem conflito k^2 arestas

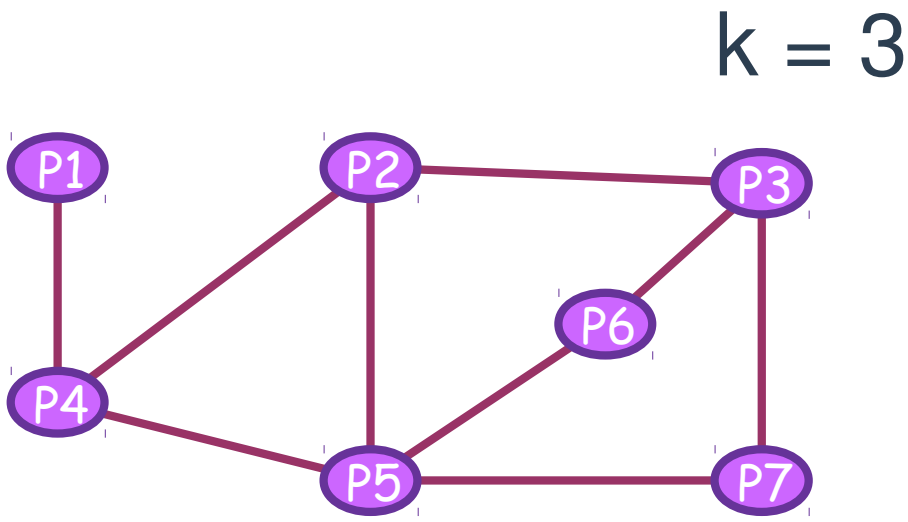
Voltando ao nosso problema



Perguntas:

1. O processo P8 será enfileirado?
   - não, um vértice isolado não tem conflito k^2 arestas
2. Existe solução com valor $k = 3$

Voltando ao nosso problema

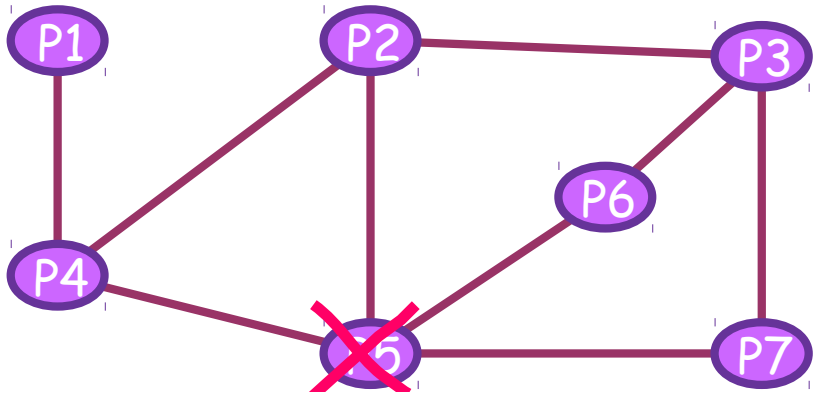


Perguntas:

1. O processo P8 será enfileirado?
   - não, um vértice isolado não tem conflito k^2 arestas
2. Existe solução com valor $k = 3$, mas que não enfileira P5?

Voltando ao nosso problema

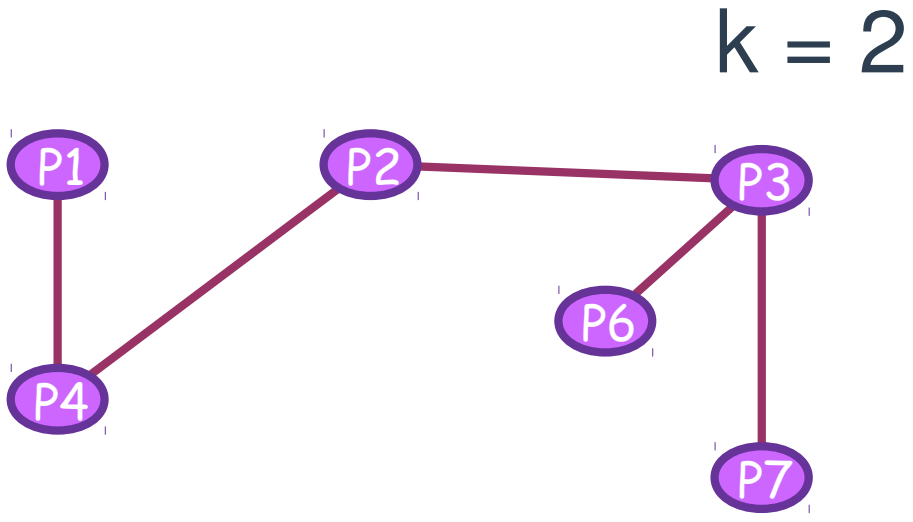
Enfileiramos e diminuímos k $k = 3$



Perguntas:

1. O processo P8 será enfileirado?
   - não, um vértice isolado não tem conflito k^2 arestas
2. Existe solução com valor $k = 3$, mas que não enfileira P5?
   - não, P5 tem 4 conflitos!

Voltando ao nosso problema



Perguntas:

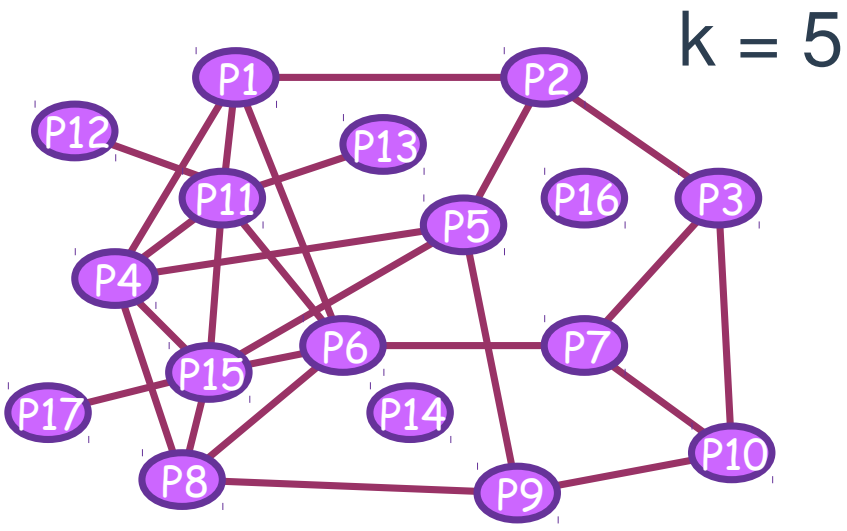
1. O processo P8 será enfileirado?
   - não, um vértice isolado não tem conflito k^2 arestas
2. Existe solução com valor $k = 3$, mas que não enfileira P5?
   - não, P5 tem 4 conflitos!

Voltando ao nosso problema

Agora podemos simplificar uma instância dada.

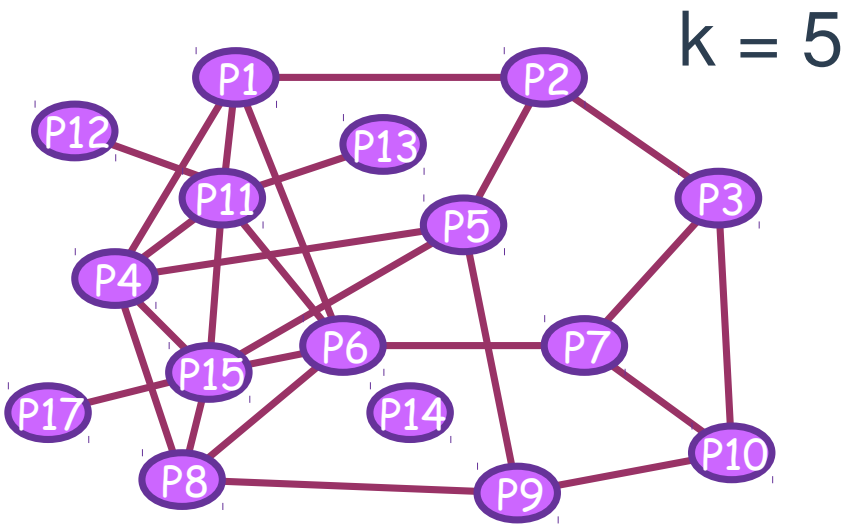
Voltando ao nosso problema

Agora podemos simplificar uma instância dada.



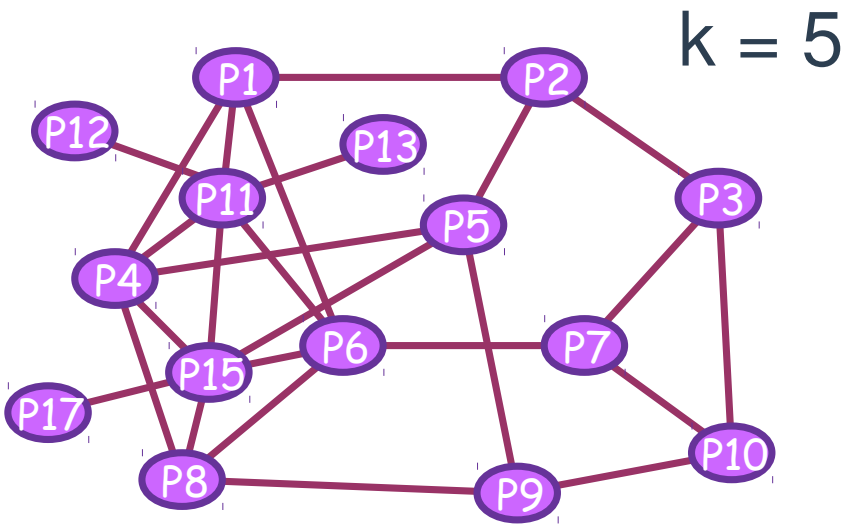
Voltando ao nosso problema

Agora podemos simplificar uma instância dada.



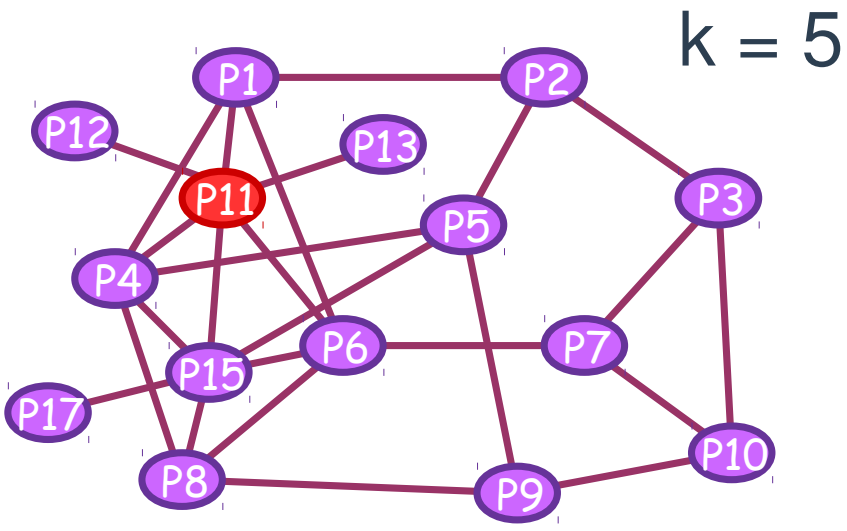
Voltando ao nosso problema

Agora podemos simplificar uma instância dada.



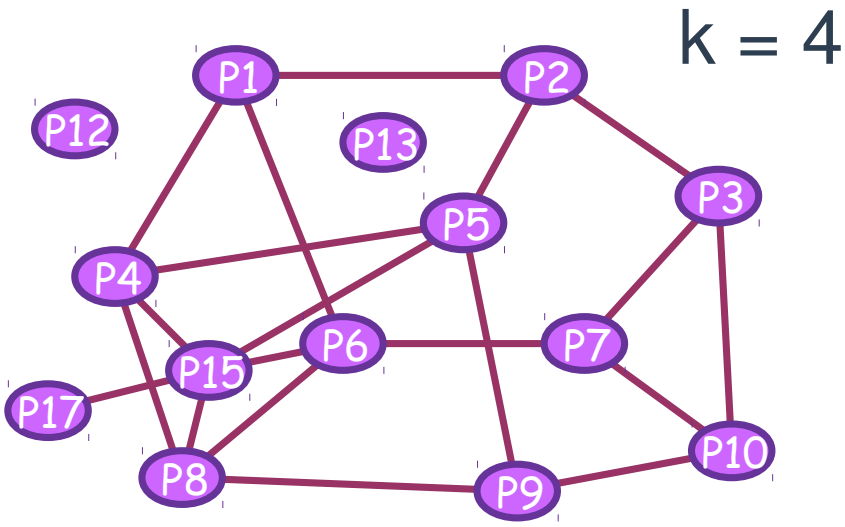
Voltando ao nosso problema

Agora podemos simplificar uma instância dada.



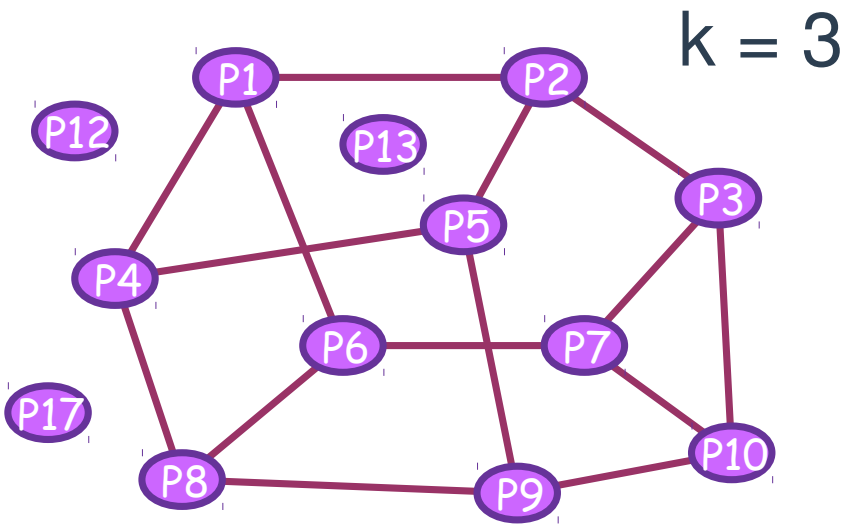
Voltando ao nosso problema

Agora podemos simplificar uma instância dada.



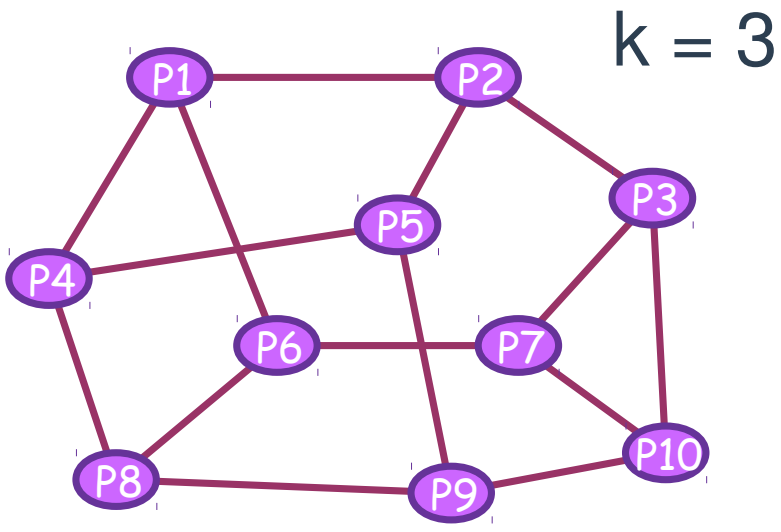
Voltando ao nosso problema

Agora podemos simplificar uma instância dada.



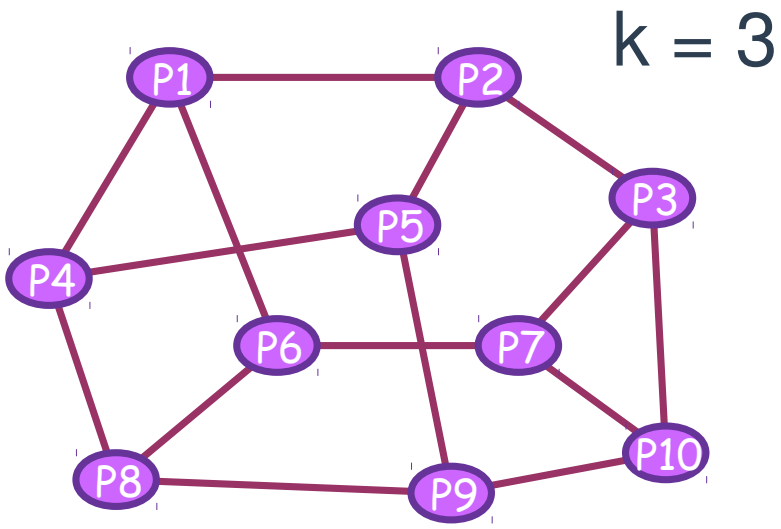
Voltando ao nosso problema

Agora podemos simplificar uma instância dada.



Voltando ao nosso problema

Agora podemos simplificar uma instância dada.

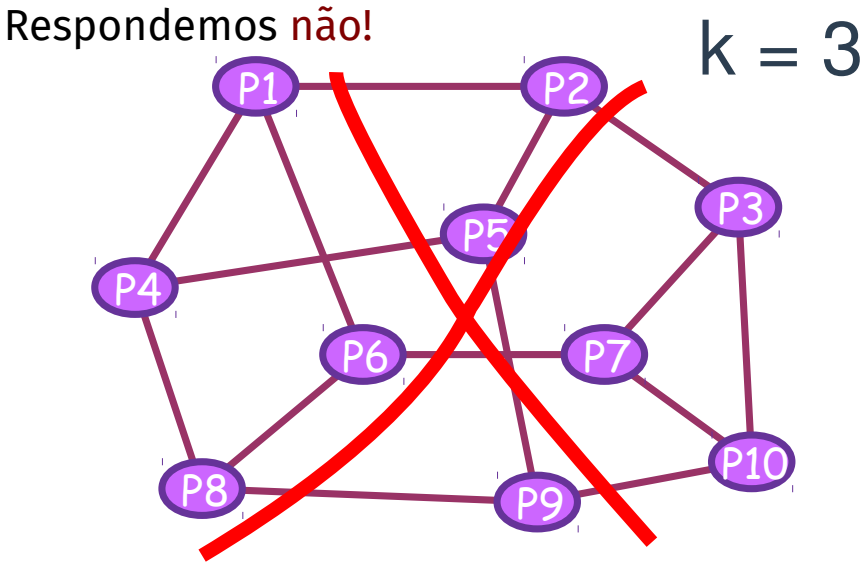


Mais uma pergunta:

4. Existe solução para essa instância com valor $k = 3$?

Voltando ao nosso problema

Agora podemos simplificar uma instância dada.



Mais uma pergunta:

4. Existe solução para essa instância com valor $k = 3$?
    - não, uma instância sim tem no máximo k^2 arestas

FPT e núcleos

Lema

*Se um problema parametrizado Q é **FPT**, então ele admite um algoritmo de kernelização.*

Conjunto de retroalimentação em torneios

Um torneio T é um digrafo tal que para todo $u, v \in V(T)$:

- ou $(u, v) \in E(T)$, ou $(v, u) \in E(T)$.

Conjunto de retroalimentação em torneios

Um torneio T é um digrafo tal que para todo $u, v \in V(T)$:

- ou $(u, v) \in E(T)$, ou $(v, u) \in E(T)$.

Um conjunto de arcos de retroalimentação de um digrafo G é um conjunto de arcos A tal que $G - A$ é acíclico.

Conjunto de retroalimentação em torneios

Um torneio T é um digrafo tal que para todo $u, v \in V(T)$:

- ou $(u, v) \in E(T)$, ou $(v, u) \in E(T)$.

Um conjunto de arcos de retroalimentação de um digrafo G é um conjunto de arcos A tal que $G - A$ é acíclico.

Problema dos arcos de retroalimentação em torneios (FAST)

Dado um torneio T, existe um conjunto de arcos de retroalimentação de tamanho no máximo k?

Conjunto de retroalimentação em torneios

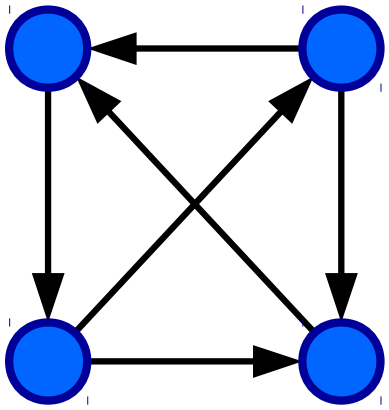
Um torneio T é um digrafo tal que para todo $u, v \in V(T)$:

- ou $(u, v) \in E(T)$, ou $(v, u) \in E(T)$.

Um conjunto de arcos de retroalimentação de um digrafo G é um conjunto de arcos A tal que $G - A$ é acíclico.

Problema dos arcos de retroalimentação em torneios (FAST)

Dado um torneio T, existe um conjunto de arcos de retroalimentação de tamanho no máximo k?



Conjunto de retroalimentação em torneios

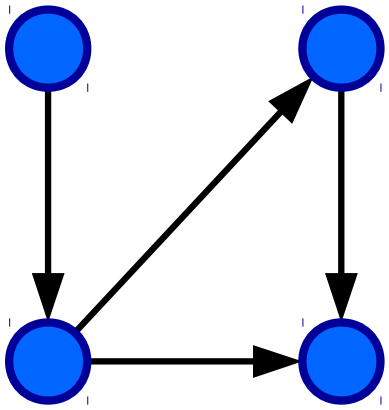
Um torneio T é um digrafo tal que para todo $u, v \in V(T)$:

- ou $(u, v) \in E(T)$, ou $(v, u) \in E(T)$.

Um conjunto de arcos de retroalimentação de um digrafo G é um conjunto de arcos A tal que $G - A$ é acíclico.

Problema dos arcos de retroalimentação em torneios (FAST)

Dado um torneio T, existe um conjunto de arcos de retroalimentação de tamanho no máximo k?



Revertendo arestas

- Dado $F \subseteq E(G)$, $G \otimes F$ é o digrafo obtido de G invertendo F.
- Vamos relacionar retroalimentação com reversão de arcos.

Revertendo arestas

- Dado $F \subseteq E(G)$, $G \otimes F$ é o digrafo obtido de G invertendo F.
- Vamos relacionar retroalimentação com reversão de arcos.

Lema

Um digrafo G é acíclico sss existe uma ordenação dos vértices tal que, se $(u, v) \in E(G)$, então $u < v$.

Revertendo arestas

- Dado $F \subseteq E(G)$, $G \otimes F$ é o digrafo obtido de G invertendo F.
- Vamos relacionar retroalimentação com reversão de arcos.

Lema

Um digrafo G é acíclico sss existe uma ordenação dos vértices tal que, se $(u, v) \in E(G)$, então $u < v$.

Lema

Se $G \otimes F$ é acíclico, então F é um conjunto de retroalimentação.

Revertendo arestas

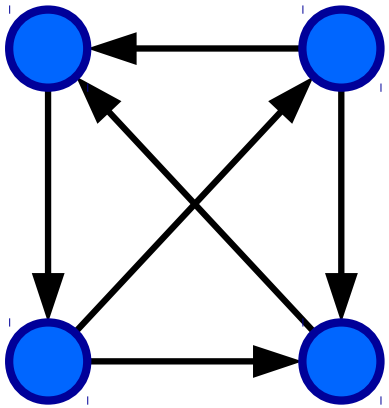
- Dado $F \subseteq E(G)$, $G \otimes F$ é o digrafo obtido de G invertendo F.
- Vamos relacionar retroalimentação com reversão de arcos.

Lema

Um digrafo G é acíclico sss existe uma ordenação dos vértices tal que, se $(u, v) \in E(G)$, então $u < v$.

Lema

Se $G \otimes F$ é acíclico, então F é um conjunto de retroalimentação.



Revertendo arestas

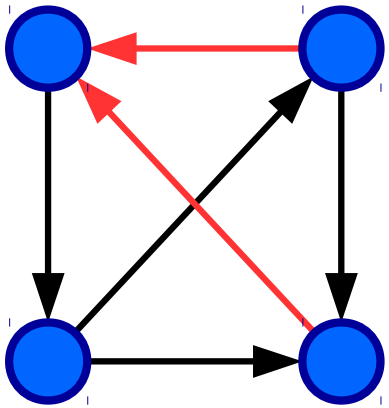
- Dado $F \subseteq E(G)$, $G \otimes F$ é o digrafo obtido de G invertendo F.
- Vamos relacionar retroalimentação com reversão de arcos.

Lema

Um digrafo G é acíclico sss existe uma ordenação dos vértices tal que, se $(u, v) \in E(G)$, então $u < v$.

Lema

Se $G \otimes F$ é acíclico, então F é um conjunto de retroalimentação.



Revertendo arestas

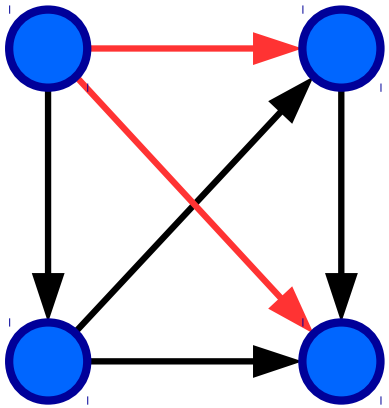
- Dado $F \subseteq E(G)$, $G \otimes F$ é o digrafo obtido de G invertendo F.
- Vamos relacionar retroalimentação com reversão de arcos.

Lema

Um digrafo G é acíclico sss existe uma ordenação dos vértices tal que, se $(u, v) \in E(G)$, então $u < v$.

Lema

Se $G \otimes F$ é acíclico, então F é um conjunto de retroalimentação.



Problema equivalente

Lema

Seja G um digrafo e $F \subseteq E(G)$. São equivalentes:

Problema equivalente

Lema

Seja G um digrafo e $F \subseteq E(G)$. São equivalentes:

- *F é um conjunto de retroalimentação minimal de G;*

Problema equivalente

Lema

Seja G um digrafo e $F \subseteq E(G)$. São equivalentes:

- *F é um conjunto de retroalimentação minimal de G;*
- *F é conjunto de arcos minimal tal que $G \otimes F$ é acíclico.*

Problema equivalente

Lema

Seja G um digrafo e $F \subseteq E(G)$. São equivalentes:

- *F é um conjunto de retroalimentação minimal de G;*
- *F é conjunto de arcos minimal tal que $G \otimes F$ é acíclico.*

Arestas de reversão em torneios

Dado um torneio T, existe um conjunto de arcos F, $|F| \leq k$ tal que $G \otimes F$ é acíclico?

Reduções

Reduções

Redução FAST.1: Se uma arco e é contido em pelo menos $k+1$ triângulos, devolva $(G \otimes \{e\}, k-1)$.

Reduções

Redução FAST.1: Se uma arco e é contido em pelo menos $k+1$ triângulos, devolva $(G \otimes \{e\}, k-1)$.

Redução FAST.2: Se vértice v não é contido em nenhum triângulo, devolva $(G - v, k)$.

Reduções

Redução FAST.1: Se uma arco e é contido em pelo menos $k+1$ triângulos, devolva $(G \otimes \{e\}, k-1)$.

Redução FAST.2: Se vértice v não é contido em nenhum triângulo, devolva $(G - v, k)$.

Teorema

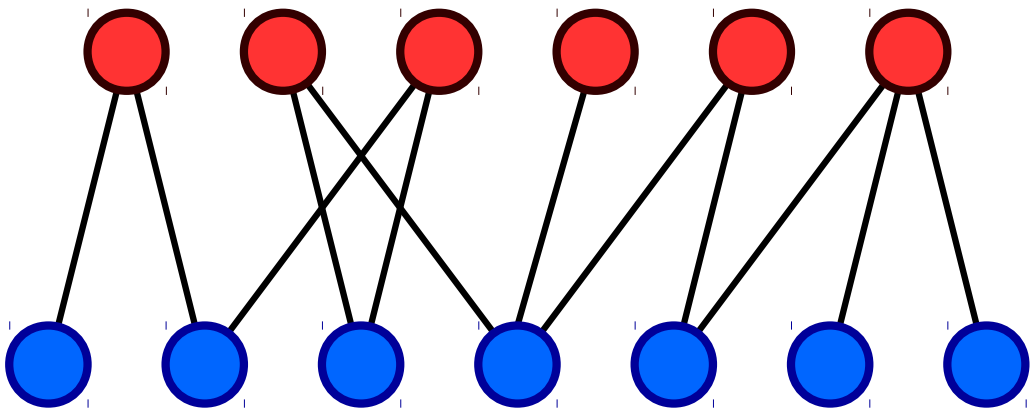
O problema de conjunto de retroalimentação em torneios admite um núcleo com no máximo $k^2 + 2k$ vértices.

Decomposição em Coroa

Emparelhamento

Dados conjuntos de vértices disjuntos U e W, um conjunto de arestas M é um emparelhamento de U em W se:

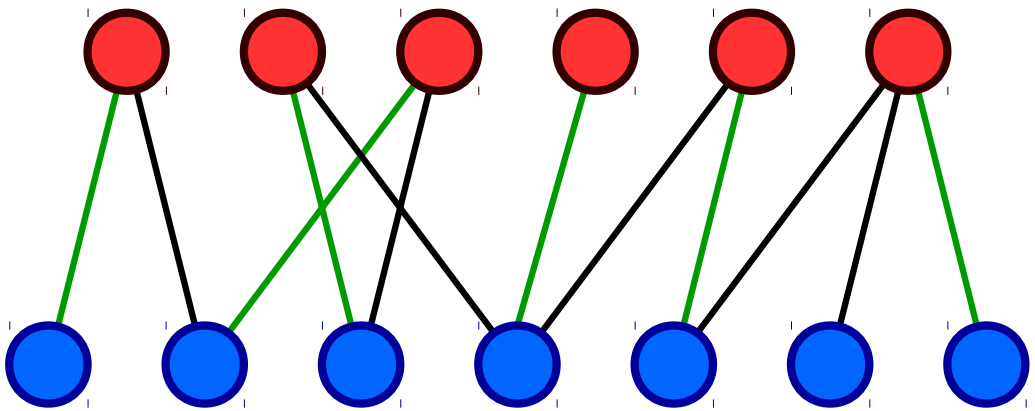
1. M é um emparelhamento
2. cada aresta tem exatamente um extremo em U e um em W
3. M satura U (i.e., todo vértice de U é um extremo de alguma aresta de M)



Emparelhamento

Dados conjuntos de vértices disjuntos U e W, um conjunto de arestas M é um emparelhamento de U em W se:

1. M é um emparelhamento
2. cada aresta tem exatamente um extremo em U e um em W
3. M satura U (i.e., todo vértice de U é um extremo de alguma aresta de M)



Decomposição em coroa

Definição (Decomposição em coroa)

Uma decomposição em coroa de um grafo G é uma partição de $V(G)$ em três partes C, H e R tal que:

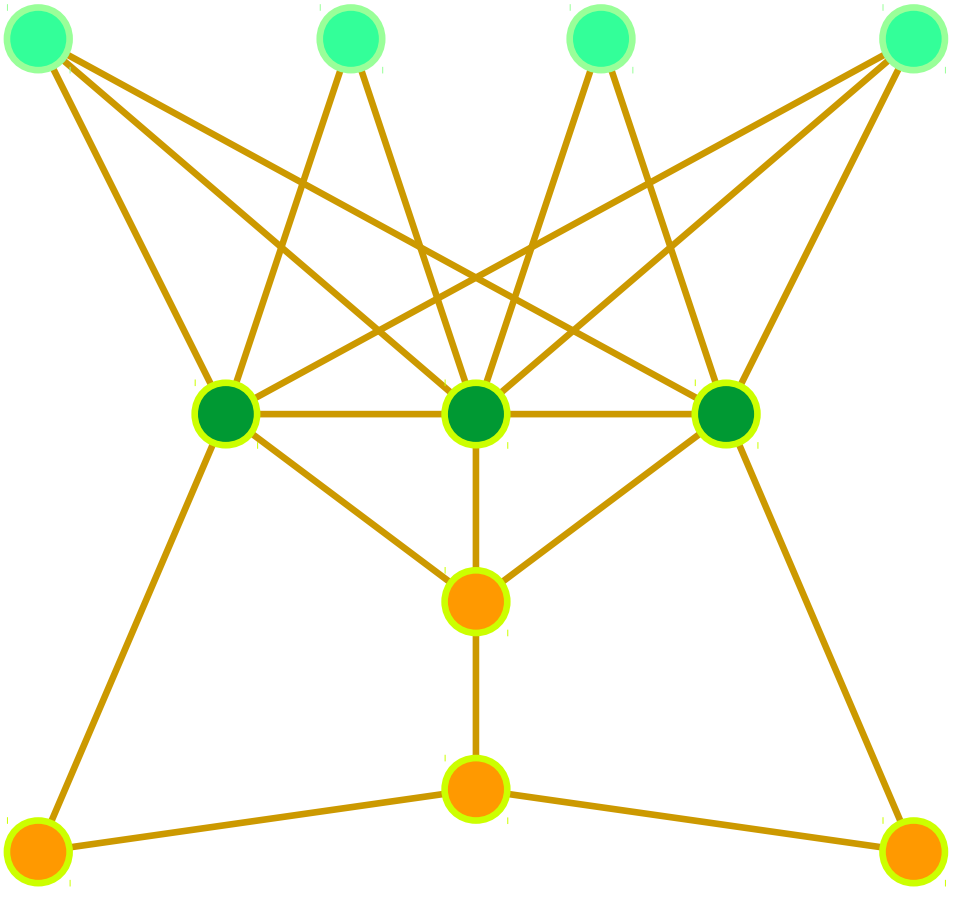
Decomposição em coroa

Definição (Decomposição em coroa)

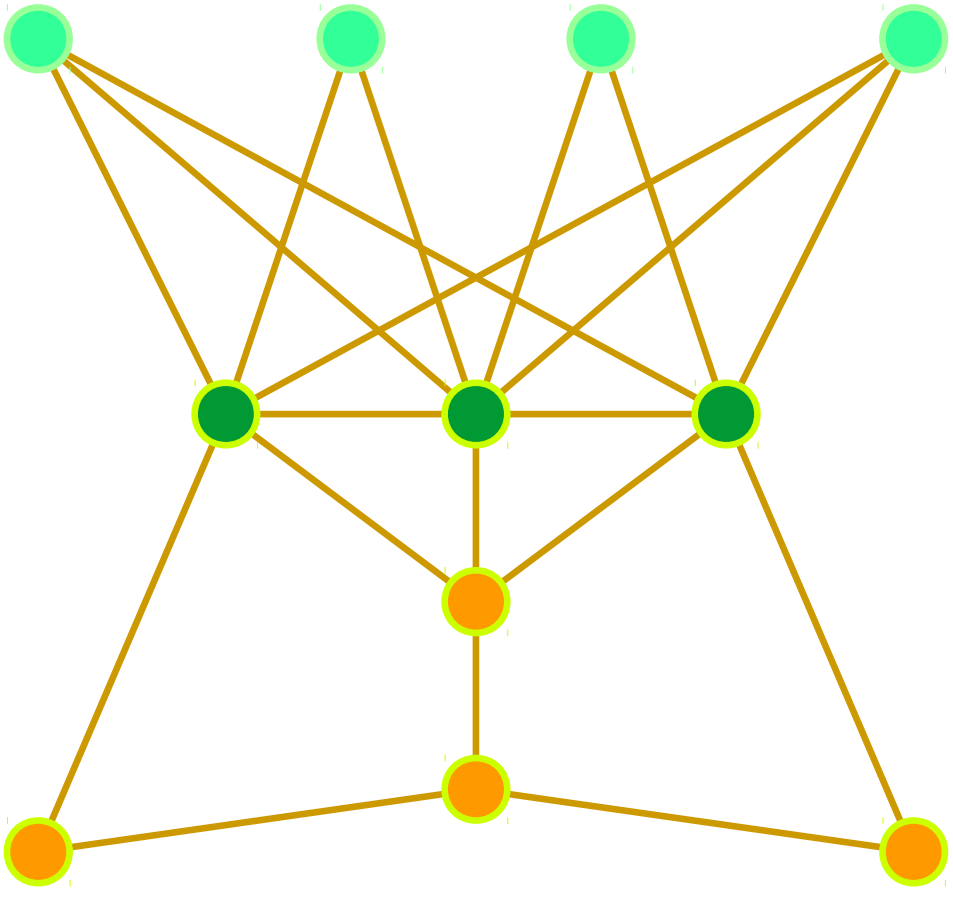
Uma decomposição em coroa de um grafo G é uma partição de $V(G)$ em três partes C, H e R tal que:

1. C é não vazio;
2. C é um conjunto independente;
3. H é um separador de (C, R);
4. G contém um emparelhamento de H em C

Coroa

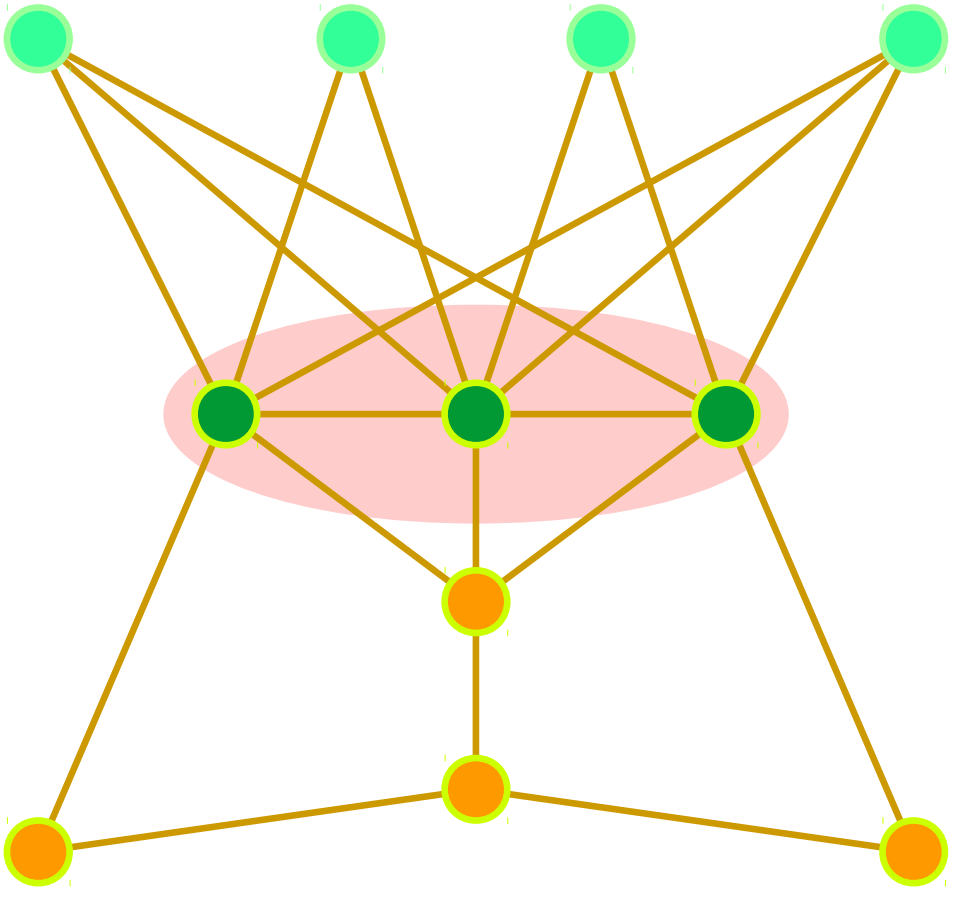


Coroa

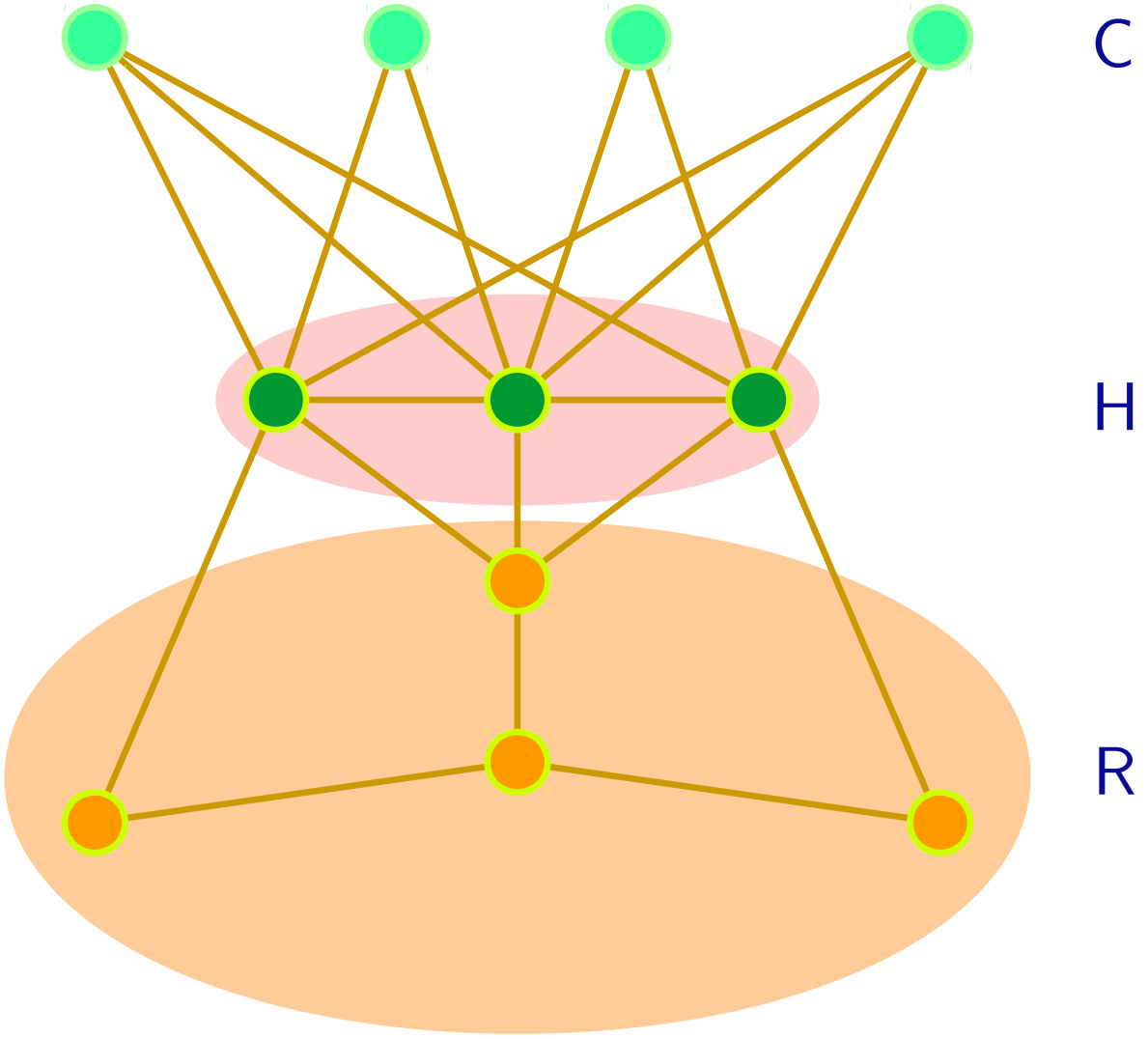


C

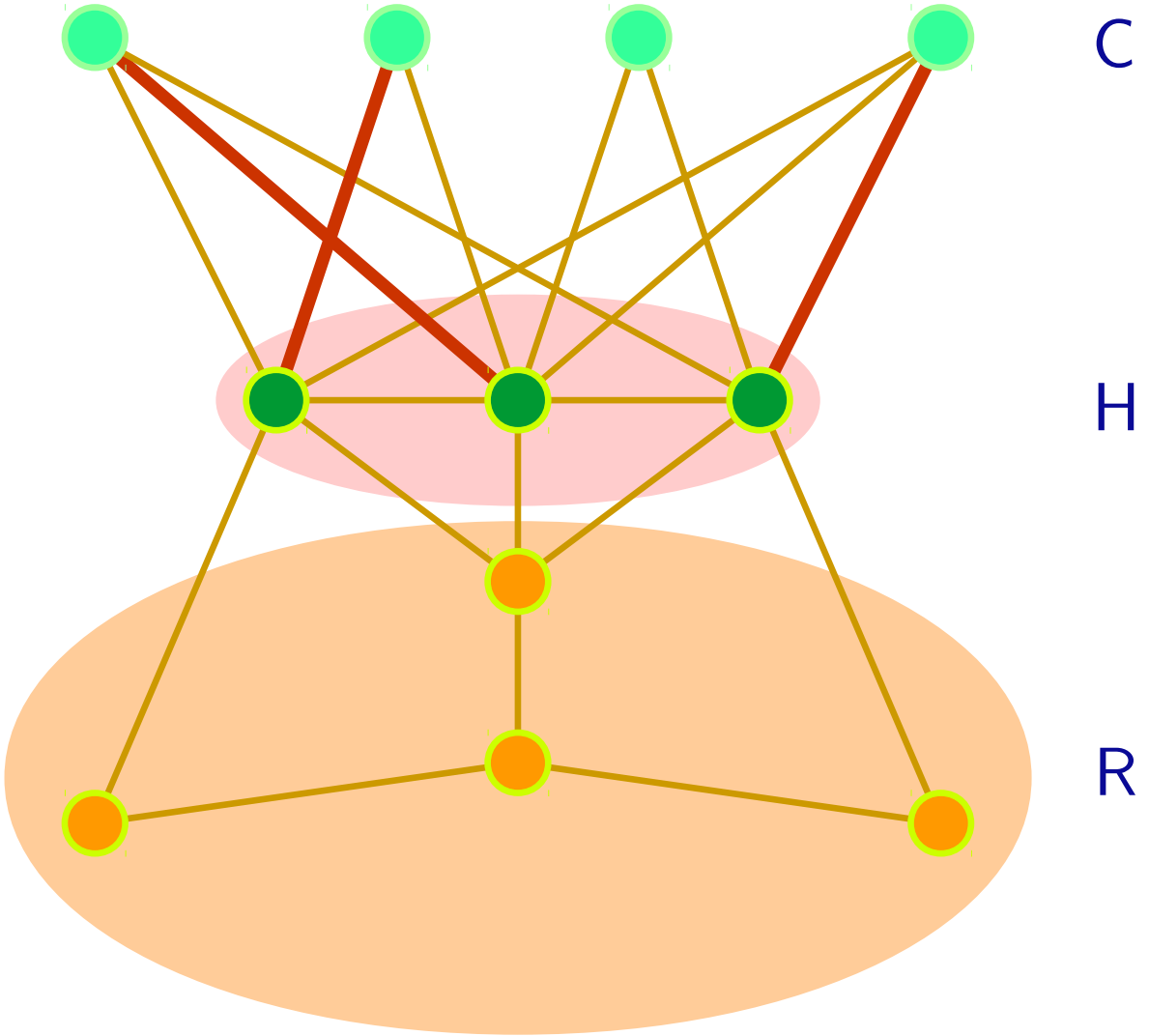
Coroa



Coroa



Coroa



Teoremas estruturais

Teorema (Kőnig)

Se G é um grafo bipartido, então o tamanho do emparelhamento máximo é igual ao tamanho da cobertura por vértices mínima.

Teoremas estruturais

Teorema (Kőnig)

Se G é um grafo bipartido, então o tamanho do emparelhamento máximo é igual ao tamanho da cobertura por vértices mínima.

Teorema (Hall)

Seja G um grafo bipartido com partição V_1, V_2. Existe um emparelhamento que satura V_1 sss para todo $X \subseteq V_1$ vale $|N(X)| \geq |X|$.

Encontrando um emparelhamento

Teorema (Hopcroft-Karp)

Seja G um grafo bipartido com n vértices e m arestas e partição V_1, V_2.

Encontrando um emparelhamento

Teorema (Hopcroft-Karp)

Seja G um grafo bipartido com n vértices e m arestas e partição V_1, V_2.

- *Existe um algoritmo que encontra um emparelhamento máximo e uma cobertura por vértices mínima em tempo $\mathcal{O}(m\sqrt{n})$.*

Encontrando um emparelhamento

Teorema (Hopcroft-Karp)

Seja G um grafo bipartido com n vértices e m arestas e partição V_1, V_2.

- *Existe um algoritmo que encontra um emparelhamento máximo e uma cobertura por vértices mínima em tempo $\mathcal{O}(m\sqrt{n})$.*
- *Além disso, o algoritmo encontra:*
  - *ou um emparelhamento que satura V_1,*
  - *ou um conjunto minimal $X \subseteq V_1$ tal que $|N(X)| < |X|$.*

Lema da coroa

Lema (Lema da coroa)

Seja G um grafo sem vértices isolados e com pelo menos $3k + 1$ vértices. Existe um algoritmo polinomial que:

Lema da coroa

Lema (Lema da coroa)

Seja G um grafo sem vértices isolados e com pelo menos $3k + 1$ vértices. Existe um algoritmo polinomial que:

- *encontra um emparelhamento de tamanho $k + 1$ em G; ou*
- *encontra uma decomposição em coroa de G.*

Lema da coroa

Lema (Lema da coroa)

Seja G um grafo sem vértices isolados e com pelo menos $3k + 1$ vértices. Existe um algoritmo polinomial que:

- *encontra um emparelhamento de tamanho $k + 1$ em G; ou*
- *encontra uma decomposição em coroa de G.*

Aplicação: Se um parâmetro k for limitante superior para a cobertura por vértices, então pode ser uma ferramenta para encontrar um núcleo pequeno

Aplicação: Cobertura por vértices

NÚCLEO-VC(G, k):

1. Remova vértices isolados ;

Aplicação: Cobertura por vértices

NÚCLEO-VC(G, k):

1. Remova vértices isolados ;
2. Se $|V| \geq 3k + 1$, obtenha uma coroa (C, H, R)

Aplicação: Cobertura por vértices

NÚCLEO-VC(G, k):

1. Remova vértices isolados ;
2. Se $|V| \geq 3k + 1$, obtenha uma coroa (C, H, R)
   - 2.1 Se há um emparelhamento M, com $|M| = k + 1$:
     - responda não;

Aplicação: Cobertura por vértices

NÚCLEO-VC(G, k):

1. Remova vértices isolados ;
2. Se $|V| \geq 3k + 1$, obtenha uma coroa (C, H, R)
   - 2.1 Se há um emparelhamento M, com $|M| = k + 1$:
     - responda não;
   - 2.2 Senão:
     - Faça $(G, k) \leftarrow (G - H, k - |H|)$;
     - Volte ao passo 1

Aplicação: Cobertura por vértices

NÚCLEO-VC(G, k):

1. Remova vértices isolados ;
2. Se $|V| \geq 3k + 1$, obtenha uma coroa (C, H, R)
   - 2.1 Se há um emparelhamento M, com $|M| = k + 1$:
     - responda não;
   - 2.2 Senão:
     - Faça $(G, k) \leftarrow (G - H, k - |H|)$;
     - Volte ao passo 1
3. Devolva (G, k)

Aplicação: Cobertura por vértices

NÚCLEO-VC(G, k):

1. Remova vértices isolados ;
2. Se $|V| \geq 3k + 1$, obtenha uma coroa (C, H, R)
   - 2.1 Se há um emparelhamento M, com $|M| = k + 1$:
     - responda não;
   - 2.2 Senão:
     - Faça $(G, k) \leftarrow (G - H, k - |H|)$;
     - Volte ao passo 1
3. Devolva (G, k)

Teorema

Cobertura por vértices admite um núcleo com $3k$ vértices.

Problema de satisfatibilidade máxima

Problema de satisfatibilidade máxima

Dada uma fórmula normal conjuntiva (CNF) F, e um inteiro k, existe uma atribuição de verdade nas variáveis que satisfaz pelo menos k cláusulas?

Exemplo:

$$F = (x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1) \wedge (\neg x_0 \vee \neg x_1)$$

$$k = 3$$

Aplicação: Satisfatibilidade máxima

Teorema

Satisfatibilidade máxima admite um núcleo com no máximo k variáveis e $2k$ cláusulas.

Ramificação

Árvore de busca

Ideia: *backtracking*

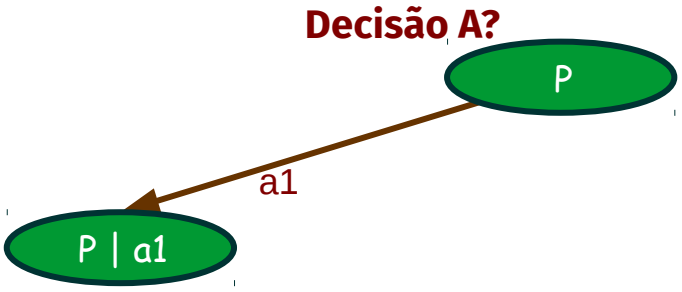
- Fazer uma sequência de decisões
- Em cada decisão, obter um subproblema
- Uma folha representa:
  - uma solução; ou
  - um certificado de que não há solução

Árvore de busca: genérica

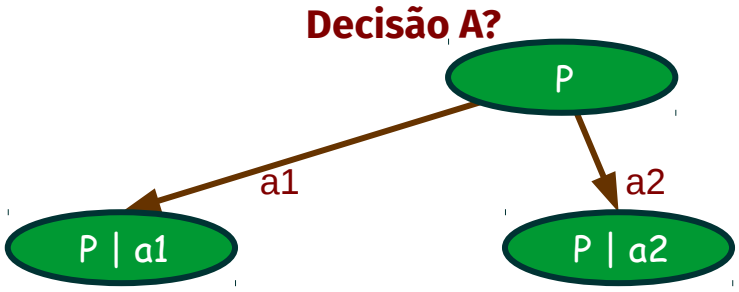
Árvore de busca: genérica

Árvore de busca: genérica

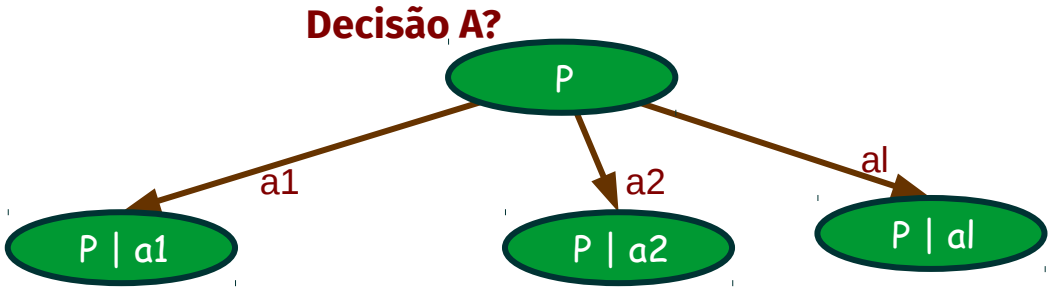
Árvore de busca: genérica



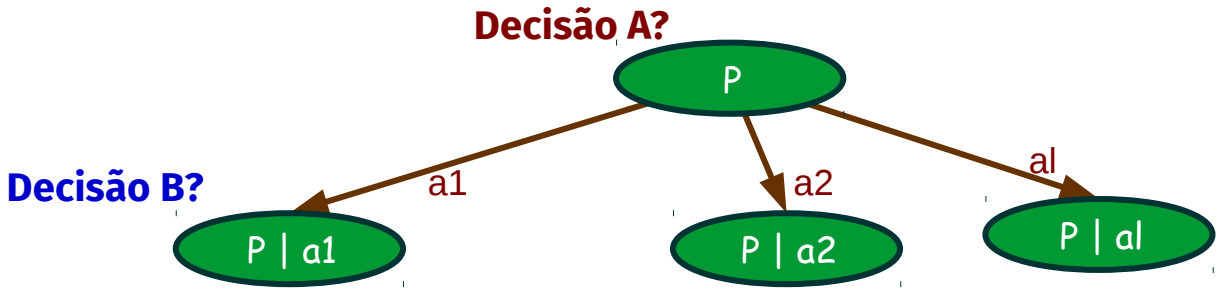
Árvore de busca: genérica



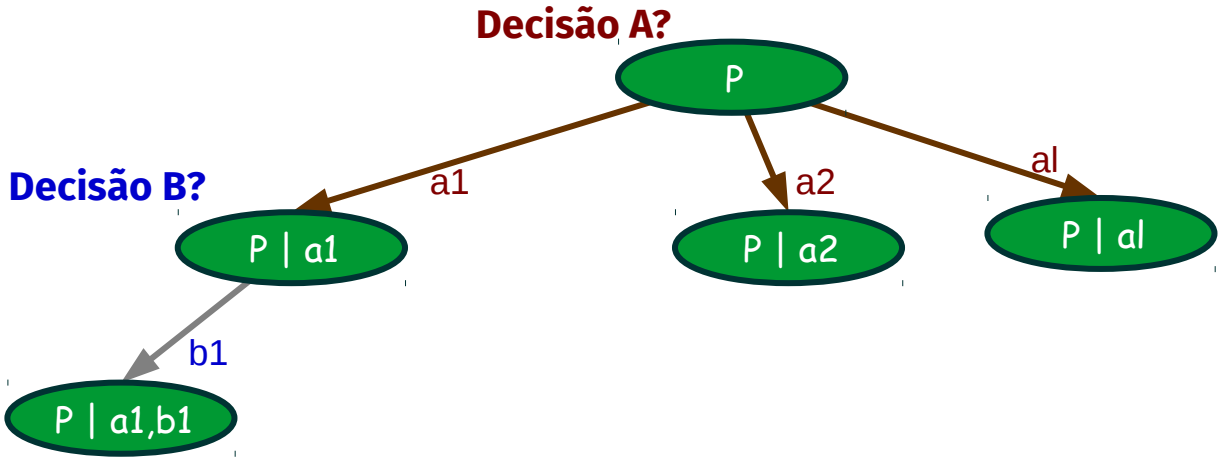
Árvore de busca: genérica



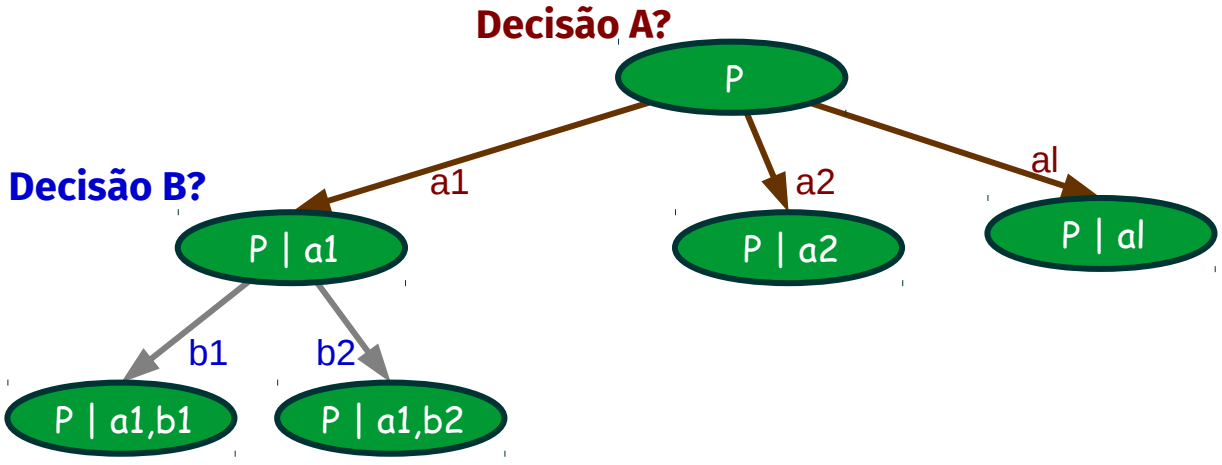
Árvore de busca: genérica



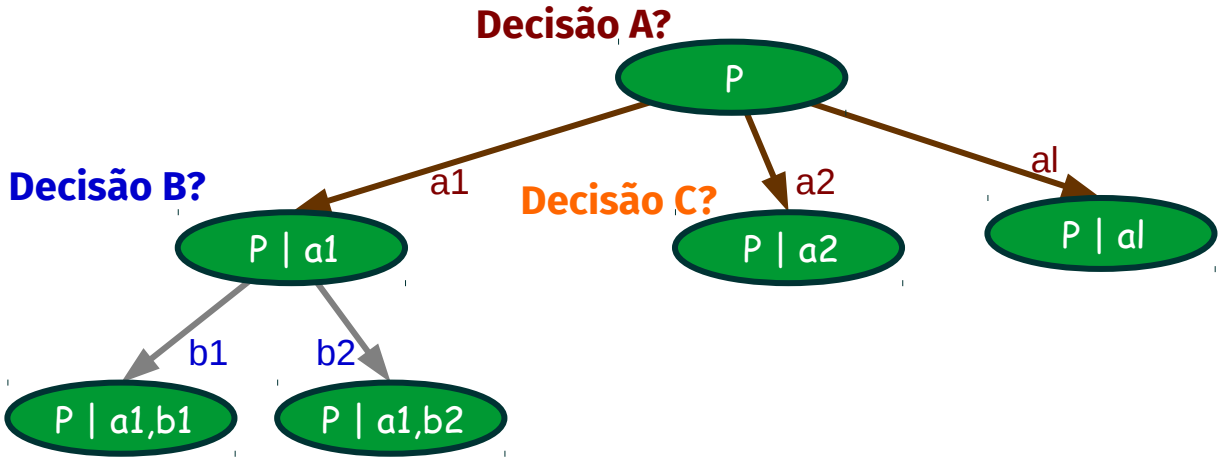
Árvore de busca: genérica



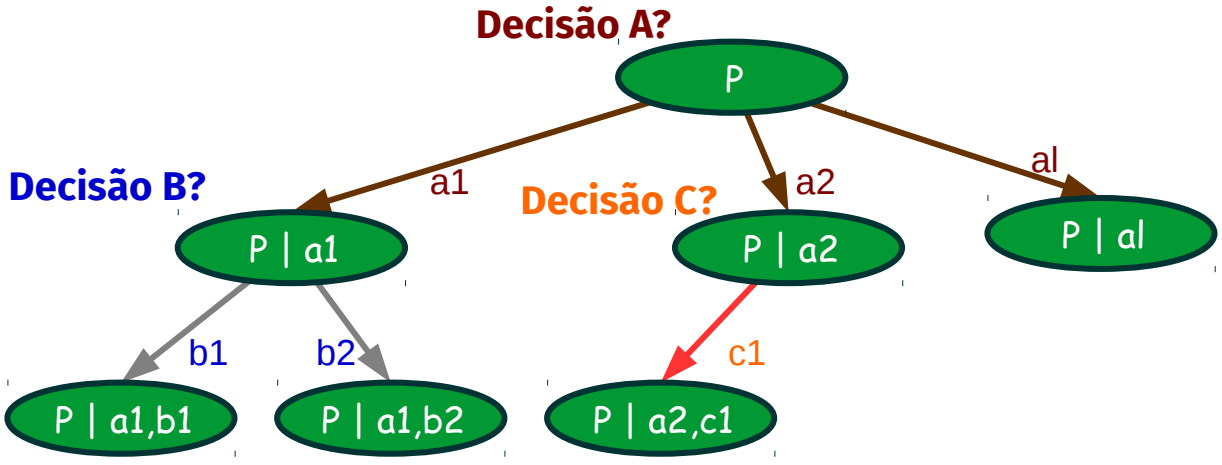
Árvore de busca: genérica



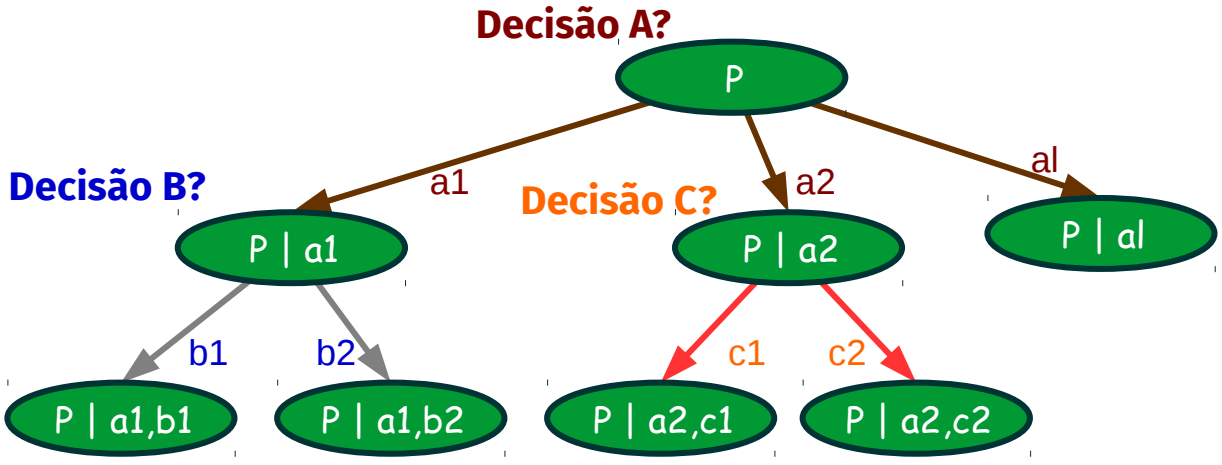
Árvore de busca: genérica



Árvore de busca: genérica



Árvore de busca: genérica



Árvores limitadas

Pergunta: Como usar árvores de busca para obter **FPT**?

- Tamanho da árvore é pequeno (função de k)
- Tempo gasto em cada nó é polinomial

Árvores limitadas

Pergunta: Como usar árvores de busca para obter **FPT**?

- Tamanho da árvore é pequeno (função de k)
- Tempo gasto em cada nó é polinomial

Vamos considerar:

- Uma instância I de um problema de minimização
- Uma medida $\mu(I)$:

Árvores limitadas

Pergunta: Como usar árvores de busca para obter **FPT**?

- Tamanho da árvore é pequeno (função de k)
- Tempo gasto em cada nó é polinomial

Vamos considerar:

- Uma instância I de um problema de minimização
- Uma medida $\mu(I)$:
  - queremos que $\mu(I)$ dependa só do parâmetro k

Ramificação

Em cada nó da árvores executamos uma ramificação

Ramificação

Em cada nó da árvores executamos uma ramificação

Ramificação de I

Crie instâncias I_1, ..., I_ℓ tais que:

Ramificação

Em cada nó da árvores executamos uma ramificação

Ramificação de I

Crie instâncias I_1, ..., I_ℓ tais que:

1. dada solução S_i de I_i:
   - existe solução $h_i(S_i)$ de I e
   - $\{h_i(S)\}_{1 \leq i \leq \ell}$ contém solução ótima;

Ramificação

Em cada nó da árvores executamos uma ramificação

Ramificação de I

Crie instâncias I_1, ..., I_ℓ tais que:

1. dada solução S_i de I_i:
   - existe solução $h_i(S_i)$ de I e
   - $\{h_i(S)\}_{1 \leq i \leq \ell}$ contém solução ótima;
2. ℓ é uma função do parâmetro $\mu(I)$ somente;

Ramificação

Em cada nó da árvores executamos uma ramificação

Ramificação de I

Crie instâncias $I_1, ..., I_\ell$ tais que:

1. dada solução S_i de I_i:
   - existe solução $h_i(S_i)$ de I e
   - $\{h_i(S)\}_{1 \leq i \leq \ell}$ contém solução ótima;
2. ℓ é uma função do parâmetro $\mu(I)$ somente;
3. existe $c > 0$ tal que $\mu(I_i) \leq \mu(I) - c$ para cada i.

Limitando a altura

Um algoritmo de ramificação tem altura limitada:

- se $\mu(I) < 0$, então instância é fácil;
- grau de ramificação ℓ de um nó é limitado;
- altura é limitada

Framework para FPT

Framework para FPT

Solução é um subconjunto de algum universo U:

Framework para FPT

Solução é um subconjunto de algum universo U:

1. Identificamos $S \subseteq U$ pequeno (em tempo polinomial)
2. Adivinhamos que elemento de S está em uma solução
3. Garantimos que a medida $\mu(I)$ diminui
   (i.e., o "número" de decisões que faltam diminui)

Cobertura por vértices

Recapitulando:

Cobertura por vértices

Recapitulando:

- Queremos $X \subseteq V(G)$ tal que $E[G - X] = \emptyset$

Cobertura por vértices

Recapitulando:

- Queremos $X \subseteq V(G)$ tal que $E[G - X] = \emptyset$
- Núcleo: $3k$ vértices usando o Lema da coroa

Cobertura por vértices

Recapitulando:

- Queremos $X \subseteq V(G)$ tal que $E[G - X] = \emptyset$
- Núcleo: $3k$ vértices usando o Lema da coroa
- Melhor atual: $\mathcal{O}(m\sqrt{n} + 4^k k^{\mathcal{O}(1)})$

Cobertura por vértices

Recapitulando:

- Queremos $X \subseteq V(G)$ tal que $E[G - X] = \emptyset$
- Núcleo: $3k$ vértices usando o Lema da coroa
- Melhor atual: $\mathcal{O}(m\sqrt{n} + 4^k k^{\mathcal{O}(1)})$

Observação

Para $v \in V(G)$:

- *ou v está na solução;*
- *ou $N(v)$ está na solução.*

Cobertura por vértices

Recapitulando:

- Queremos $X \subseteq V(G)$ tal que $E[G - X] = \emptyset$
- Núcleo: $3k$ vértices usando o Lema da coroa
- Melhor atual: $\mathcal{O}(m\sqrt{n} + 4^k k^{\mathcal{O}(1)})$

Observação

Para $v \in V(G)$:

- *ou v está na solução;*
- *ou $N(v)$ está na solução.*

Observação

Se, para todo $v \in V(G)$, $d(v) \leq 1$, então Cobertura por vértices é polinomial.

Ramificando

Ramificação

- Escolha $v \in V(G)$ com $d(v) \geq 2$;

Ramificando

Ramificação

- Escolha $v \in V(G)$ com $d(v) \geq 2$;
- Temos $S = \{v\} \cup N(v)$ contém elemento de uma solução;

Ramificando

Ramificação

- Escolha $v \in V(G)$ com $d(v) \geq 2$;
- Temos $S = \{v\} \cup N(v)$ contém elemento de uma solução;
- Ramificamos em:

Ramificando

Ramificação

- Escolha $v \in V(G)$ com $d(v) \geq 2$;
- Temos $S = \{v\} \cup N(v)$ contém elemento de uma solução;
- Ramificamos em:
  1. Se v está na solução:
     1.1 remova v de G;
     1.2 faça $k \leftarrow k - 1$

Ramificando

Ramificação

- Escolha $v \in V(G)$ com $d(v) \geq 2$;
- Temos $S = \{v\} \cup N(v)$ contém elemento de uma solução;
- Ramificamos em:
  1. Se v está na solução:
     - 1.1 remova v de G;
     - 1.2 faça $k \leftarrow k - 1$
  2. Se $N(v)$ está na solução:
     - 2.1 remova $N[v]$ de G;
     - 2.2 faça $k \leftarrow k - |N(v)|$.

Tempo de execução

Tempo de execução

- cada nó da árvore gasta temo $n^{\mathcal{O}(1)}$
- seja $\tau(k)$ o número de nós
- TOTAL: $\tau(k)n^{\mathcal{O}(1)}$.

Tempo de execução

- cada nó da árvore gasta temo $n^{\mathcal{O}(1)}$
- seja $\tau(k)$ o número de nós
- TOTAL: $\tau(k)n^{\mathcal{O}(1)}$.

Observação

Dada uma árvore de ramificação $\mathcal{T}$ com ℓ folhas, então o número de nós de $\mathcal{T}$ é no máximo $2\ell - 1$.

Tempo de execução

- cada nó da árvore gasta temo $n^{\mathcal{O}(1)}$
- seja $\tau(k)$ o número de nós
- TOTAL: $\tau(k)n^{\mathcal{O}(1)}$.

Observação

Dada uma árvore de ramificação $\mathcal{T}$ com ℓ folhas, então o número de nós de $\mathcal{T}$ é no máximo $2\ell - 1$.

- Seja $T(k)$ o número de folhas na árvore de ramificação da Cobertura por vértices

Tempo de execução

- cada nó da árvore gasta temo $n^{\mathcal{O}(1)}$
- seja $\tau(k)$ o número de nós
- TOTAL: $\tau(k)n^{\mathcal{O}(1)}$.

Observação

Dada uma árvore de ramificação $\mathcal{T}$ com ℓ folhas, então o número de nós de $\mathcal{T}$ é no máximo $2\ell - 1$.

- Seja $T(k)$ o número de folhas na árvore de ramificação da Cobertura por vértices

$$T(i) = \begin{cases} T(i-1) + T(i-2) & \text{se } i \geq 2; \\ 1 & \text{caso contrário.} \end{cases}$$

Calculando

Lema

Para todo $k \geq 0$, $T(k) \leq 1{,}6181^k$.

Olhando

Observação: a base impacta muito no tempo de execução

Olhando

Observação: a base impacta muito no tempo de execução

Conclusão: tentar melhorar o passo de ramificação

Olhando

Observação: a base impacta muito no tempo de execução

Conclusão: tentar melhorar o passo de ramificação

Quando ramificamos:

- se v não é parte da solução, diminuímos k em $|N(v)| \geq 2$

Olhando

Observação: a base impacta muito no tempo de execução

Conclusão: tentar melhorar o passo de ramificação

Quando ramificamos:

- se v não é parte da solução, diminuímos k em $|N(v)| \geq 2$;
- mas se $|N(v)| > 2$ a árvore é menor!

Olhando

Observação: a base impacta muito no tempo de execução

Conclusão: tentar melhorar o passo de ramificação

Quando ramificamos:

- se v não é parte da solução, diminuímos k em $|N(v)| \geq 2$;
- mas se $|N(v)| > 2$ a árvore é menor!

Observação

Cobertura por vértices é polinomial quando $d(v) \leq 2$ para todo $v \in V(G)$.

Melhorando

Agora resolvemos um nó em tempo polinomial sempre que $\Delta(G) \leq 2$:

Melhorando

Agora resolvemos um nó em tempo polinomial sempre que $\Delta(G) \leq 2$:

$$T(i) = \begin{cases} T(i-1) + T(i-3) & \text{se } i \geq 3; \\ 1 & \text{caso contrário.} \end{cases}$$

Melhorando

Agora resolvemos um nó em tempo polinomial sempre que $\Delta(G) \leq 2$:

$$T(i) = \begin{cases} T(i-1) + T(i-3) & \text{se } i \geq 3; \\ 1 & \text{caso contrário.} \end{cases}$$

Teorema

Cobertura por vértices pode ser resolvida em tempo $\mathcal{O}(n\sqrt{m} + 1{,}4656\, k^{\mathcal{O}(1)})$.