

# Algoritmos Parametrizados

## Decomposição em Árvore

---

Lehilton Pedrosa

Segundo Semestre de 2016

Instituto de Computação – Unicamp

1. Decomposição em Árvore
2. Programação Dinâmica com largura de árvore definidas
3. Teorema de Courcelle
4. Caracterizações Alternativas
5. Obtendo uma decomposição em árvore
6. Shifting

# Decomposição em Árvore

---

**Pergunta:** Quão parecido um grafo  $G$  é de uma árvore?

**Possibilidades:**

- Quantos círculos eu preciso quebrar para torná-lo acíclico?
- Quantos vértices eu preciso remover?

**Alternativa:** como **decompor** o grafo em nós de uma árvore?

**Motivação:** vários problemas são mais fáceis em árvores

**Ideia:**

- Tentamos obter uma decomposição “pequena” do grafo
  - (a) Se conseguirmos, então usamos um algoritmo parecido com o de árvore
  - (b) Se não conseguirmos, então descobrimos uma estrutura forte sobre o grafo

**e.g.:** instâncias sim de Cobertura por vértices e Conjuntos de retroalimentação têm decomposições pequenas

# Conjunto independente máximo

## Problema do Cobertura Independente Máximo com pesos

Dado um grafo  $G$  e uma função com pesos nas arestas  $w : V(G) \rightarrow \mathbb{R}_+$ , queremos encontrar um conjunto independente  $I \subseteq V(G)$  que maximize

$$\sum_{v \in I} w(v).$$

# Programação Dinâmica

**Ideia:** o problema é muito fácil em árvore  $T$

Programação Dinâmica:

- Enraíze  $T$  em um vértice  $r \in V(G)$  qualquer
- Seja  $T_v$  a subárvore com raiz  $v$
- Consideramos recursivamente cada subproblema para  $T_v$  de maneira *bottom-up*

**Tabela:** seja  $v$  um vértice com  $N(v) = \{v_1, \dots, v_q\}$ :

- $B[v] = \sum_{i=1}^q A[v_i]$
- $A[v] = \max\{B[v], w(v) + \sum_{i=1}^q B[v_i]\}$

Vamos considerar o problema restrito a **subgrafos** de uma grade  $k \times N$ :

Defina:

- $X_j$  os vértices que estão na coluna  $j$
- $G_j = G[X_1, X_2, \dots, X_j]$

## Programação Dinâmica na Grade

- Para cada  $j$ , enumeramos um conjunto que  $Y \subseteq X_j$  que **não** está na solução
- Consideramos o **subproblema** em  $G_j$

## Calculando caso base

**Caso:**  $c[1, Y]$

- Basta enumerar a solução  $S \subseteq X_1 \setminus Y$

## Calculando caso geral

**Caso:**  $c[j, Y]$ , com  $j \geq 2$

- Enumeramos a parte da solução  $S \subseteq X_j \setminus Y$  na coluna  $j$
- Isso induz um subproblema em  $G_{j-1}$

## Fato crucial

- $X_{j-1}$  separa  $X_j$  do grafo  $G_{j-2}$

## Generalizando

- grades de altura  $k$  são apenas caminhas mais “gordos”
- bastar obter uma sequência de separadores

# Decomposição em caminho

## Definição (Decomposição em Caminho)

Uma **decomposição em caminho** de um grafo é uma sequência  $\mathcal{P} = (X_1, X_2, \dots, X_r)$  de **bags** (sacos), onde  $X_i \subseteq V(G)$  para cada  $i \in \{1, 2, \dots, r\}$ , tal que vale:

(P1)  $\bigcup_{i=1}^r X_i = V(G)$ ;

(P2) Para cada  $uv \in E(G)$ , existe  $\ell \in \{1, 2, \dots, r\}$  tal que  $X_\ell$  contém  $u$  e  $v$ ;

(P3) Para cada  $u \in V(G)$ , se  $u \in X_i \cap X_k$  com  $i \leq k$ , então  $u \in X_j$  para cada  $j$  em  $i \leq j \leq k$ .

**Largura de caminho:** a largura de caminho é o tamanho da maior bag menos 1.

## Algumas definições

- **Separação:** dizemos que  $(A, B)$  é uma separação se  $A \cup B = V(G)$  e não há arestas entre  $A \setminus B$  e  $B \setminus A$
- **Separador:** o separador de uma separação é  $A \cap B$ , com ordem  $|A \cap B|$
- **Borda:** a borda  $\delta(A)$  de um conjunto  $A$  é o conjunto de vértices de  $A$  com vizinhos em  $N(A)$

## Decomposição em caminho e separadores

### Lema

Seja  $(X_1, \dots, X_r)$  uma decomposição em caminhos de um grafo  $G$ . Então para cada  $j \in \{1, \dots, r-1\}$ , vale

$$\delta \left( \bigcup_{i=1}^j X_i \right) \subseteq X_j \cap X_{j+1}.$$

Equivalentemente,  $(\bigcup_{i=1}^j X_i, \bigcup_{i=j+1}^r X_i)$  é uma separação de  $G$ .

# Decomposição em caminho padrão

## Decomposição boa

Dizemos que uma decomposição em caminhos  $(X_1, X_2, \dots, X_r)$  é **boa** se:

- $X_1 = X_r = \emptyset$  e
  - para cada  $i \in \{1, 2, \dots, r-1\}$ :
    1. ou existe um vértice  $v \in X_i$  tal que  $X_{i+1} = X_i \cup \{v\}$ ,
    2. ou existe um vértice  $w \in X_i$  tal que  $X_{i+1} = X_i \setminus \{w\}$
- 
- No caso 1, dizemos que **introduzimos** o vértice  $v$
  - No caso 2, dizemos que **esquecemos** o vértice  $w$

# Decomposição em árvore

## Definição (Decomposição em Árvore)

Uma **decomposição em árvore** de um grafo  $G$  é um par  $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$  onde  $T$  é uma árvore em que cada nó  $t$  é atribuído a um conjunto de vértices  $X_t \subseteq V(G)$ , chamado de **bag**, tal que vale:

- (T1)  $\bigcup_{i=1}^r X_i = V(G)$ ;
- (T2) Para cada  $uv \in E(G)$ , existe  $\ell \in \{1, 2, \dots, r\}$  tal que  $X_\ell$  contém  $u$  e  $v$ ;
- (T3) Para cada  $u \in V(G)$ , o conjunto  $T_u = \{t \in V(T) : u \in X_t\}$  induz uma subárvore conexa de  $T$ .

**Largura de árvore:** a largura de árvore (ou largura arbórea) é o tamanho da maior bag menos 1.

### Lema

Seja  $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$  uma decomposição em árvore de um grafo  $G$  e seja  $ab$  uma aresta de  $T$ . A floresta  $T - ab$  obtida de  $T$  deletando-se a aresta  $ab$  consiste de duas componentes conexas  $T_a$  (com  $a$ ) e  $T_b$  (com  $b$ ).

Seja  $A = \cup_{t \in V(T_a)} X_t$  e  $B = \cup_{t \in V(T_b)} X_t$ . Então

$$\delta(A), \delta(B) \subseteq X_a \cap X_b.$$

Equivalentemente,  $(A, B)$  é uma separação de  $G$ .

# Decomposição em árvore padrão

## Decomposição boa

Dizemos que uma decomposição em árvore associada com um  $r$  é **boa** se:

- $X_r = \emptyset$  e  $X_t = \emptyset$  para toda folha  $t$
- cada nó interno de  $T$  é de um dos tipos:
  1. **Introdução:** um nó  $t$  com exatamente **um** filho  $t'$ , tal que  $X_t = X_{t'} \cup \{v\}$  para algum  $v \in X_t$ ;
  2. **Esquecimento:** um nó  $t$  com exatamente **um** filho  $t'$ , tal que  $X_t = X_{t'} \setminus \{w\}$  para algum  $w \in X_{t+1}$ ;
  3. **Junção:** um nó  $t$  com exatamente **dois** filhos  $t_1$  e  $t_2$ , tais que  $X_t = X_{t_1} = X_{t_2}$ .

Dada uma decomposição em árvore com largura  $k$ :

- encontramos uma decomposição boa com  $\mathcal{O}(k|V(G)|)$  nós;
- o tempo de execução é  $\mathcal{O}(k^2 \cdot \max\{|V(T)|, |V(G)|\})$ .

# Programação Dinâmica com largura de árvore definidas

---

# Programação dinâmica: comparando

Estender a programação dinâmica da árvore:

- **Na árvore:** uma aresta  $(u, v)$  divide o grafo em dois:
  - seja  $T_v$  a subárvore enraizada em  $v$
  - não existe aresta de  $V(T_v)$  a antecessor de  $v$ , ou  
→ a borda de  $V_t$  é  $v$
  - uma decisão na subárvore  $T_v$  só depende de  $v$
  - enumeramos o destino de  $v$
  - criamos um **subproblema**
- **Na decomposição:** um nó  $t$  divide o grafo em dois:
  - seja  $T_t$  a subárvore enraizada em  $t$  (na decomposição)
  - seja  $V_t$  todos os vértices das bags de  $V[T_t]$ :  $V_t = \cup_{r \in T_t} X_r$
  - não existe aresta de  $V(T_v)$  a  $X_s \setminus X_t$ , para antecessor  $s$  de  $v$   
→ a borda de  $V_t \subseteq X_t$
  - uma decisão na subárvore  $T_v$  só depende de  $v$
  - enumeramos o destino de  $v$
  - criamos um **subproblema**

# Programação dinâmica: subproblema

Precisamos listar e enumerar:

- cada nó  $t$  da árvore
- o destino de cada vértices  $v$  de  $X_t$ 
  - $v$  está em uma solução de maior custo
  - $v$  não está em nenhuma solução de maior custo

## Subproblema

Consideramos o seguinte subproblema definido por:

$c[t, S]$  = máximo peso de um conjunto independente  $\hat{S}$  tal que  $S \subseteq \hat{S} \subseteq V_t$ , e  $\hat{S}$  é independente.

- se não há solução, marcamos  $c[t, S] = -\infty$
- o problema original é  $c[r, \emptyset]$

## Algoritmo

1. Obter uma decomposição **boa**  $\mathcal{T}$
2. Preencher tabela de maneira *bottom-up*, dependendo do tipo do nó: **folha**, **esquecimento**, **introdução**

### Folha $t$ :

- Definimos  $c[t, \emptyset] = 0$ .

## Nó de introdução $t$ :

- Seja  $t'$  o único filho de  $t$
- Temos  $X_t = X_{t'} \cup \{v\}$  para algum  $v \in X_t$
- Seja  $S \subseteq X_t$ :
  - (a) Se  $S$  não é independente,  $c[t, S] = -\infty$
  - (b) Senão,

$$c[t, S] = \begin{cases} c[t', S] & \text{se } v \notin S, \\ c[t', S \setminus \{v\}] + w(v) // & \text{se } v \in S. \end{cases}$$

**Afirmção:** o passo está bem definido

### Nó de esquecimento $t$ :

- Seja  $t'$  o único filho de  $t$
- Temos  $X_t = X_{t'} \setminus \{w\}$  para algum  $w \in X_{t'}$
- Seja  $S \subseteq X_t$ :
  - (a) Se  $S$  não é independente,  $c[t, S] = -\infty$
  - (b) Senão,

$$c[t, S] = \max \left\{ c[t', S], c[t', S \cup \{w\}] \right\}.$$

**Afirmção:** o passo está bem definido

### Teorema

Seja  $G$  um grafo com  $n$  vértices e pesos nos vértices. Dados  $G$  e uma decomposição em árvore de  $G$  com largura no máximo  $k$ , então o Problema do Conjunto Independente de Peso Máximo pode ser resolvido em tempo  $2^k \cdot k^{O(1)} \cdot n$ , i.e., em *tempo linear* para  $k$  fixo.

# Decomposição boa estendida

Mais um tipo de nó:

- **Introdução de aresta:** um nó  $t$ , rotulado com uma aresta  $E(G)$  tal que:
  - $u, v \in X_t$
  - tem exatamente um filho  $t'$  tal  $X_t = X_{t'}$ .

## Decomposição boa estendida

Uma decomposição em árvore boa que além dos tipos tradicionais, pode ter nós de **introdução de aresta** e:

- para cada  $uv \in E(G)$ , existe um único nó que introduz  $uv$

Uma subárvore  $T_t$  da decomposição induz um subgrafo:

$$G_t = (V_t, E_t = \{e : e \text{ é introduzida em um nó de } T_t\})$$

## Problema de Steiner

**Árvore de Steiner mínima:** Dados grafo  $G$ , conjunto de terminais  $K \subseteq V(G)$ , queremos encontrar uma árvore de Steiner  $H$  cujo número  $|E(H)|$  seja mínimo.

# Quebrando o problema

**Simplificando:** evitando subproblemas vazios:

- escolha terminal  $u^* \in K$  e adicione a cada bag
- largura da decomposição aumenta somente em uma unidade

**Dividindo:** Como um separador  $X_t$  divide uma árvore de Steiner  $H$ ?

- somente um subconjunto  $X \subseteq X_t$  está em  $H$
- $G[X]$  contém componentes conexas (que particionam  $X$ )
- $K \cap V_t \subseteq H[V_t]$

## Subproblema

Seja  $t$  um nó,  $X \subseteq X_t$  e  $\mathcal{P} = \{P_1, P_2, \dots, P_q\}$  uma partição de  $X$ . Então  $c[t, X, \mathcal{P}]$  é o menor número de arestas de uma floresta  $F$  que:

1. a floresta  $F$  contém exatamente componentes  $C_1, \dots, C_q$  com  $V(C_s) \cap X_t = P_s$ , para cada  $s \in \{1, \dots, q\}$ ;
2.  $X_t \cap V(F) = X$ ;
3.  $K \cap V_t \subseteq V(F)$ .

- se não há solução, marcamos  $c[t, S, \mathcal{P}] = +\infty$
- o problema original é  $c[r, \{u^*\}, \{\{u^*\}\}]$

### Folha $t$ :

- Temos  $X_t = \{u^*\}$ 
  - (a) Se  $X = \emptyset$ , então  $\mathcal{P} = \emptyset$  e  $c[t, X, \mathcal{P}] = +\infty$
  - (b) Se  $S = \{u^*\}$ , então  $\mathcal{P} = \{\{u^*\}\}$  e  $c[t, X, \mathcal{P}] = 0$

# Preenchendo a tabela: nó de introdução de aresta

## Nó $t$ introduz vértice $uv$ :

- (a) Se  $u \notin X$  ou  $v \notin X$ , daí  $c[t, X, \mathcal{P}] = +\infty$
- (b) Se  $u, v \in X$  e não existe  $P \in \mathcal{P}$  com  $u, v \in P$ , daí  $c[t, X, \mathcal{P}] = +\infty$
- (c) Senão, existe  $\hat{P}$  com  $u, v \in \hat{P}$ :
  - $uv$  não faz parte da solução:  
 $\Rightarrow$  existe uma solução com a mesma partição em  $X_{t'}$
  - $uv$  faz parte de uma solução:  
 $\Rightarrow u$  e  $v$  estavam em componentes distintas em  $X_{t'}$

Definimos:

$$c[t, X, \mathcal{P}] = \min \left\{ c[t', X, \mathcal{P}], \min_{\mathcal{P}'} c[t', X, \mathcal{P}'] \right\},$$

onde  $\mathcal{P}'$  contém as mesmas partes  $\mathcal{P}$ , exceto que  $\hat{P}$  foi dividido em  $P_u, P_v$ , com  $u \in P_u$  e  $v \in P_v$

## Preenchendo a tabela: nó de esquecimento e junção

**Nó  $t$  esquece  $w$ :**

$$c[t, X, \mathcal{P}] = \min \left\{ c[t', X, \mathcal{P}], \min_{\mathcal{P}'} c[t', X \cup \{w\}, \mathcal{P}'] \right\},$$

onde  $\mathcal{P}'$  contém as mesmas partes  $\mathcal{P}$ , exceto que foi adicionado  $w$  a alguma parte  $P$

**Nó  $t$  junta  $t_1$  e  $t_2$ :**

$$c[t, X, \mathcal{P}] = \min_{\mathcal{P}_1, \mathcal{P}_2} c[t_1, X, \mathcal{P}_1] + c[t_2, X, \mathcal{P}_2],$$

onde  $\mathcal{P}_1$  e  $\mathcal{P}_2$  são todas as partições que, quando juntadas, não induzem componentes cíclicas

### Teorema

Dado um grafo  $G$ , terminais  $K \subseteq V(G)$  e uma decomposição em árvore de  $G$  com largura  $k$ , então podemos encontrar uma Árvore de Steiner Mínima que conecta  $K$  com tempo de execução  $k^{O(k)} \cdot n$ .

## Diversos problemas

Diversos problemas admitem **FPT** quando a largura de árvore é pequena.

Os algoritmos de programação dinâmica podem ser usados para obter:

- **Algoritmos exponenciais simples:**
  - Cobertura por vértice, conjunto independente máximo, Conjunto dominante, Corte máximo, Transversal de Ciclo Ímpar,  $q$ -Coloração.
- **Algoritmos super-exponenciais (com dependência linear):**
  - Árvore de Steiner, Conjunto de retroalimentação de vértices, Caminho Hamiltoniano, Número cromático, Empacotamento de ciclos, Cobertura por Vértices conexa, Conjunto dominante conexo, conjunto de retroalimentação conexo.

# Teorema de Courcelle

---

**Exemplo** de fórmula **MSO<sub>2</sub>** (*Monadic Second Order*):

$$\begin{aligned}\text{conn}(X) &= \forall_{Y \subseteq V} [(\exists_{u \in V} u \in Y \wedge \exists_{v \in X} v \notin Y) \\ &\Rightarrow (\exists_{e \in E} \exists_{u \in X} \exists_{v \in X} \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y)]\end{aligned}$$

**Em português:** Para todo subconjunto e vértices  $Y$ , se  $X$  contém tanto um vértice de  $Y$  como um vértice fora de  $Y$ , então existe uma aresta  $e$  cujos extremos  $u, v$  ambos pertencem a  $X$ , mas um deles está em  $Y$  e o outro está fora de  $Y$ .

# Um “programa”

## Fórmula:

- uma sequência de **símbolos**, **variáveis** e **predicados**.
- pode ser interpretada como um “programa” para verificar uma propriedade: **o subgrafo induzido por  $X$  é conexo?**

**Variáveis:** podem representar

- vértices e arestas de um grafo
- **conjuntos** de vértices ou de arestas (variáveis monádicas)

**parâmetros:** os parâmetros ou **variáveis livres** são as variáveis que são externas à fórmula, e.g.:  $X$

**predicados:** os predicados são afirmações sobre determinado número de variáveis, e.g.: **inc**

**observação:** se não permitirmos conjuntos de arestas, obteremos uma fórmula **MSO<sub>1</sub>**

## Operadores lógicos:

- operadores tradicionais:  $\wedge, \vee, \neg, \Rightarrow$
- representam testes e condições

## Quantificadores:

- **universal:**
  - símbolo  $\forall$
  - testa uma condição sobre uma variável em **todas** avaliações do domínio, i.e.: falso se em alguma for falsa
- **existencial:**
  - símbolo  $\exists$
  - testa uma condição sobre uma variável em **alguma** avaliações do domínio, i.e.: verdadeiro se em alguma for verdadeira

# Definição formal

Distinguimos:

- **sintaxe:** regras para uma expressão estar bem formada
- **semântica:** significado dado a uma fórmula e elementos
  - depende do contexto
- **assinatura:** conjunto  $\Sigma$  de variáveis livres
  - $x \in \Sigma$  contém um tipo associado (“parâmetro formal”)
  - $x^G$  é o elemento correspondente em  $G$  (“parâmetro real”)
- **estrutura:** par  $\langle G, \Sigma^G \rangle$  que representa um contexto
  - $\Sigma^G$  contém todas avaliações de  $\Sigma$
  - se  $\phi$  é verdadeira com  $\Sigma^G$ , escrevemos

$$\langle G, \Sigma^G \rangle \models \phi$$

i.e.,  $\langle G, \Sigma^G \rangle$  modela  $\phi$ .

Uma fórmula **MSO<sub>2</sub>** é definida recursivamente.

**Base:** Uma fórmula atômica é:

- $u \in X$ , para cada  $u, X \in \Sigma$ , que representam vértice (aresta) e conjunto de vértices (arestas)
- $\text{inc}(u, e)$ , para cada  $u, e \in \Sigma$ , que representam vértice e aresta
- $x = y$ , para cada  $x, y \in \Sigma$

# Compondo fórmulas

Uma fórmula **MSO<sub>2</sub>** é definida recursivamente.

**Operações:** dadas fórmulas  $\phi_1, \phi_2$ , também são fórmulas:

- $\neg\phi_1, \phi_1 \wedge \phi_2, \phi_1 \vee \phi_2$
- $\phi_1 \Rightarrow \phi_2$

**Quantificação:** dada fórmula  $\psi$  com assinatura  $\Sigma'$  com  $v \in \Sigma'$ , então também é fórmula com assinatura  $\Sigma = \Sigma' \setminus \{v\}$ :

- $\phi_{\forall} = \forall_{v \in V} \psi$
- $\phi_{\exists} = \exists_{v \in V} \psi$

Podemos escreve notação equivalente quando for conveniente, e.g.:

- $\exists_{v \in X} \psi$  é o mesmo que  $\exists_{v \in V} (v \in X) \wedge \psi$
- $\text{adj}(u, v)$  é  $(u \neq v) \wedge (\exists_{e \in E} \text{inc}(u, e) \wedge \text{inc}(v, e))$
- $X \subseteq Y$  é  $\forall_{v \in V} [(v \in X) \Rightarrow (v \in Y)]$

$G$  é 3-colorível:

## Teorema

Seja  $\phi$  uma fórmula **MSO<sub>2</sub>** com assinatura  $\Sigma$  e  $G$  um grafo com uma avaliação  $\Sigma^G$ . Suponha que seja dada uma decomposição em árvore de  $G$  com largura  $t$ . Então existe um algoritmo que verifica se  $\phi$  é satisfeita para  $G$  em tempo  $f(\|\phi\|, t) \cdot n$ , para alguma função  $f$  computável.

# Teorema de Courcelle em problemas de otimização

O teorema anterior não fornece um **FPT** para o problema de Cobertura por Vértices no parâmetro  $t$  diretamente!

- Seja  $\alpha(x_1, x_2, \dots, x_p) = a_0 + a_1x_1 + \dots + a_px_p$
- Considere  $\phi$  que contenha variáveis livres sobre conjuntos (monádicas)  $X_1, \dots, X_p$  de vértices ou arestas.
- Considere o problema: minimizar  $\alpha(|X_1|, \dots, |X_p|)$  tal que  $\phi$  vale para  $G$

## Teorema

*Existe um algoritmo que resolve o problema acima em tempo  $f(\|\phi\|, t) \cdot n$ , para alguma função  $f$  computável.*

# Caracterizações Alternativas

---

## Graph searching

- Um **detetive** (buscador) segue um **programa**, que é sequência de dois movimentos:
  - **posicionar**: um detetive livre posiciona-se em um determinado vértice
  - **remove**: um detetive posicionado se retira e fica livre
- O **fugitivo** está em um vértice e se move:
  - por arestas em velocidade **ilimitada**
  - não pode se **chocar** com um detetive posicionado

**Detetives ganha:** o fugitivo é encontrado

**Fugitivo ganha:** consegue fugir indefinidamente

**Número de busca:** menor número de detetives necessários

# Contaminação

Se o grafo não tem vértices isolado, podemos imaginar que todos todas as arestas pelas quais o fugitivo pode andar estão **contaminadas com gás** que flui pelas arestas;

- uma aresta é descontaminada se dois vértices forem ocupados pelos **buscadores**
- uma aresta pode ser recontaminada se houver um caminho livre de uma aresta contaminada

## Teorema

*Para todo grafo  $G$ , se  $k$  buscadores podem limpar  $G$ , então  $k$  buscadores podem limpar  $G$  de tal maneira que nenhuma aresta volte a se recontaminar.*

## Definição (Grafo de intervalos)

Um grafo  $G$  é de intervalo se:

- para cada  $v \in V$ , existe intervalo real  $I_v = [l_v, r_v]$
  - para cada  $u, v \in V, u \neq v$ , temos que  $uv \in E$  sss  $I_v \cap I_u \neq \emptyset$ .
- 
- os intervalos associados são a **representação** de  $G$
  - a representação é **canônica** se  $\{l_v : v \in V\} = \{1, 2, \dots, n\}$

# Largura de intervalos

Lembrando:  $G'$  é **supergrafo** de  $G$ , ou  $G \subseteq G'$ , se:

- $V(G) \subseteq V(G')$ , e
- $E(G) \subseteq E(G')$ .

## Definição

A largura de intervalo de um grafo  $G$  é definida como:

$$\text{largura-intervalo}(G) = \min\{\omega(G') : G \subseteq G' \wedge G' \text{ é de intervalo}\}$$

isso é, menor valor **número de clique** de um supergrafo  $G'$  de  $G$  que é de intervalo.

# Equivalências de largura de caminho

## Teorema

*Para todo grafo  $G$  e  $k \geq 0$ , são equivalentes:*

- (i) O número de busca de  $G$  é no máximo  $k + 1$ ;*
- (ii) A largura de intervalo de  $G$  é no máximo  $k + 1$ ;*
- (iii) A largura de caminho de  $G$  é no máximo  $k$ .*

# Grafos cordais

Um grafo  $G$  é **cordal** se não contém um ciclo induzido de tamanho maior que 3.

⇒ todo ciclo com tamanho 4 ou mais tem uma **corda**.

## Definição

A largura cordal de um grafo  $G$  é definida como:

$$\text{largura-cordal}(G) = \min\{\omega(G') : G \subseteq G' \wedge G' \text{ é cordal}\}$$

isso é, menor valor **número de clique** de um supergrafo  $G'$  de  $G$  que é cordal.

# Outra busca em grafos

## *Graph searching 2*

Considere a seguinte busca:

- Um conjunto de **policiais** em um conjuntos de vértices
- Existe um **ladrão** cuja posição é conhecida em um vértice
- Em um movimento, acontece o seguinte:
  - um subconjunto de policiais pode tomar um helicóptero e declara seus destinos
  - enquanto isso o ladrão pode mover-se sem passar por policiais, conhecendo a localização de todos policiais
  - os policiais posicionam-se nos destinos declarados

**Número de busca:** menor número de policiais necessários para capturar ladrão.

Dizemos que conjuntos  $A, B \subseteq V(G)$  tocam se:

- $A \cap B \neq \emptyset$ , ou
- existe  $u \in A$  e  $v \in B$  e  $uv \in E(G)$ .

## Definição (Bramble)

Uma *bramble* (espinheiro)  $\mathcal{B}$  é uma família de conjuntos de vértices tais que:

- conjuntos tocam-se dois a dois;
- cada conjunto induz uma componente conexa.

- $C \subseteq V(G)$  cobre uma  $\mathcal{B}$  se cada conjunto intersepta  $C$
- a ordem de  $\mathcal{B}$  é o menor número de vértices que cobrem  $\mathcal{B}$

# Equivalências de largura de árvore

## Teorema

Para todo  $k \geq 0$  e grafo  $G$ :

- a largura de árvore de  $G$  é pelo menos  $k$  SSS
- $G$  contém uma bramble de ordem pelo menos  $k + 1$ .

## Teorema

Para todo grafo  $G$  e  $k \geq 0$ , são equivalentes:

- (i) A largura de árvore de  $G$  é no máximo  $k$ .
- (ii) A largura cordal de  $G$  é no máximo  $k + 1$ .
- (iii)  $k + 1$  policiais podem capturar um ladrão visível em  $G$ .
- (iv) Não existe bramble com ordem maior que  $k + 1$  em  $G$ .

## Obtendo uma decomposição em árvore

---

# Separadores

Relembrando:

- **Separação:**  $(A, B)$  é uma separação de  $A \cup B = V(G)$  se não há arestas entre  $A \setminus B$  e  $B \setminus A$
- **Separador:**
  - dizemos que  $C \subseteq V(G)$  **separa**  $X$  e  $Y$  se todo caminho  $X$ – $Y$  passa por vértice de  $C$
  - se  $C$  separa  $X$  e  $Y$ , então podemos encontrar uma separação  $(A, B)$  que separa  $X$  e  $Y$  com  $C$  como separador

## Teorema (Teorema de Menger)

- Seja  $\mu(X, Y)$  o tamanho do menor separador de  $X$  e  $Y$
- Então  $\mu(X, Y)$  é igual ao maior número de caminhos disjuntos nos vértices entre  $X$  e  $Y$ .

## Obtendo uma decomposição

Existe um algoritmo **FPT** para obter uma decomposição em árvore:

### Teorema (Boadlander)

Existe algoritmo que, dado um grafo  $G$  com  $n$  vértices e um inteiro  $k$ , executa em tempo  $k^{\mathcal{O}(k^3)} \cdot n$  e:

- *ou* constroi uma decomposição em árvore de largura  $k$ ;
- *ou* conclui que a largura de árvore de  $G$  é maior que  $k$ .

### Observações:

- o algoritmo é linear em  $n$ ;
- mas **não** é de exponencial simples em  $k$ ;
- o fator com  $k$  pode ser dominante

**Alternativa:** obter decomposições “quase” ótimas

# Obtendo uma decomposição aproximada

Como obter uma decomposição “pequena” de  $G$ ?

## Ideia:

1. Suponha que  $H \subseteq G$  é tal que  $\delta(H) \subseteq S$
2. Quebramos  $H$  em  $H_1, H_2$  com algum separador  $\hat{S} \supseteq S$
3. **Obtemos:**
  - $V(H_1) \cap \hat{S}$  separa  $H_1$  do resto do grafo  $G$
  - $V(H_2) \cap \hat{S}$  separa  $H_2$  do resto do grafo  $G$
4. **Gostaríamos:**
  - $|V(H_1) \cap \hat{S}|$  e  $|V(H_2) \cap \hat{S}|$  sejam “pequenos”
  - i.e., com tamanho menor que  $ck$  para  $c$  constante
5. **Novos objetivos:**
  - dividir  $S$  entre  $H_1$  e  $H_2$
  - encontrar  $\hat{S}$  não muito grande

# Separadores balanceados

- seja  $w : V(G) \rightarrow \mathbb{R}_+$  e  $w(X) = \sum_{u \in X} w(u)$  para  $X \subseteq V(G)$
- $X \subseteq V(G)$  é um  $\alpha$ -separador balanceado em  $G$  se:
  - $G - X$  contém componentes conexas  $D_1, \dots, D_p$
  - para cada  $i$ ,  $w(D_i) \leq \alpha \cdot w(V(G))$ .

## Lema

Seja  $G$  com largura de árvore  $k$  e  $w : V(G) \rightarrow \mathbb{R}_+$ . Então existe um  $\frac{1}{2}$ -separador balanceado  $X$  com  $|X| \leq k + 1$ .

# Separações balanceadas

Uma separação  $(A, B)$  é um  $\alpha$ -balanceada se:

- $w(A \setminus B) \leq \alpha \cdot w(V(G))$  e
- $w(B \setminus A) \leq \alpha \cdot w(V(G))$ .

## Lema

Seja  $G$  com largura de árvore  $k$  e  $w : V(G) \rightarrow \mathbb{R}_+$ . Então existe uma  $\frac{2}{3}$ -separação balanceada  $(A, B)$  com ordem no máximo  $k + 1$ .

## Corolário

Seja  $G$  com largura de árvore  $k$  e  $S \subseteq V(G)$ . Então existe uma partição  $S_A, S_B$  de  $S$  tal que:

- $k + 2 \leq |S_A|, |S_B| \leq 2k + 2$  e
- $\mu_G(S_A, S_B) \leq k + 1$ .

## Teorema

Existe algoritmo que, dado um grafo  $G$  com  $n$  vértices e um inteiro  $k$ , executa em tempo  $\mathcal{O}(8^k k^2 \cdot n^2)$  e:

- *ou* constroi uma decomposição em árvore de largura  $k$ ;
- *ou* conclui que a largura de árvore de  $G$  é maior que  $k$ .

# Algoritmo

**ENTRADA:**  $W \subseteq V(G), S \subseteq W$

- INVARIANTES:**
- (i)  $|S| \leq 3k + 4$  e  $W \setminus S \neq \emptyset$ ;
  - (ii)  $G[W]$  e  $G[S]$  são conexos;
  - (iii)  $S = N_G(W \setminus S)$

## **decompose**( $W, S$ ):

1. Construa um conjunto  $\hat{S}$  com as **propriedades**:
  - (a)  $S \subsetneq \hat{S} \subseteq W$ ;
  - (b)  $|\hat{S}| \leq 4k + 5$ ;
  - (c) cada componente de  $G[W \setminus \hat{S}]$  tem até  $3k + 4$  vizinho em  $\hat{S}$
2. Sejam  $D_1, \dots, D_p$  componentes de  $G[W \setminus \hat{S}]$ 
  - 2.1 Para cada  $i$ ,  $\mathcal{T}_i \leftarrow \text{decompose}(N_G[D_i], N_G(D_i))$
  - 2.2 Crie uma nó  $r$  com conjunto  $X_r = \hat{S}$
  - 2.3 Devolva  $\mathcal{T}_{W,S}$  com raiz  $r$  e filhos  $\mathcal{T}_i$ , para cada  $i$

# Shifting

---

Seja  $r \geq 0$  e  $v$  um vértice de  $G$ .

- denotamos por  $G_v^r$  o subgrafo induzido por vértices com distância até  $r$  de  $v$

## Teorema

Seja  $G$  um grafo planar e  $v \in V(G)$ . Então

- $tw(G_v^r) \leq 3r + 1$
- *uma decomposição em árvore com largura  $3r + 1$  pode ser encontrada em tempo polinomial.*

# Fatiamento do grafo

Consideramos uma fatia de um grafo planar  $G$  a partir de um vértice  $v$ .

Seja  $i \geq 0$  e  $j \geq 1$ :

- $L_i$  são os vértices  $u$  tais que  $d(v, u) = i$ ;
- $G_{i,i+j-1} = G[L_i \cup L_{i+1} \cup \dots \cup L_{i+j-1}]$ .

$G_{i,i+j-1}$  é uma fatia com  $j$  camadas a partir de  $i$

## Corolário

Para cada  $i \geq 0$  e  $j \geq 1$ :

- $tw(G_{i,i+j-1}) \leq 3j + 1$ ;
- *uma tal decomposição pode ser obtida em tempo polinomial.*

# Método Shifting

Suponha que temos um problema de encontrar um subconjunto de até  $k$  vértices.

## Ideia:

1. Primeiro cortamos uma fatia com  $q$  camadas,  $0 \leq q \leq k$
2. Fatiamos o resto de  $G$  fazendo:
  - 2.1 removemos uma camada;
  - 2.2 cortamos uma fatia com  $k$  camadas.
3. Repetimos o mesmo procedimento para cada  $q = 0, \dots, k$  (i.e., deslocamos as camadas)

## Pontos-chaves:

- em algum deslocamento, nenhum dos  $k$  vértices da solução foi removido;
- o grafo resultante de cada deslocamento tem largura de árvore pequena

### Lema

Seja  $G$  planar e  $k \geq 0$ . Então o conjunto de vértices de  $G$  pode ser particionado em  $k + 1$  subconjuntos (possivelmente vazios) tais que a **remoção** de qualquer um deles induz um grafo com largura de árvore no máximo  $3k + 1$ . Ainda, a partição bem como uma tal decomposição pode ser obtida em tempo polinomial.

## Problema do isomorfismo de subgrafo

**Relembrando:** No problema do isomorfismo, dado um grafo  $H$  com tamanho fixo  $k$  e um grafo  $G$ , existe um subgrafo de  $G$  isomorfo a  $H$ ?

**Primeiro-passo:** encontrar um algoritmo **FPT** para o caso particular do problema com largura de árvore fixa.

### Lema

*Existe um algoritmo que, dado um grafo  $G$  e um grafo  $H$ , decide se existe um subgrafo de  $G$  isomorfo a  $H$ . O tempo de execução é  $f(|V(H)|, tw(G)) \cdot |V(G)|^{\mathcal{O}(1)}$ , onde  $f$  é uma função computável que depende de  $|V(H)|$ .*

## Obtendo um FPT com parâmetro $k$ em grafos planares

### Teorema

Existe um algoritmo que, dado um grafo *planar*  $G$  e um grafo  $H$ , decide se existe um subgrafo de  $G$  isomorfo a  $H$ . O tempo de execução é  $f(|V(H)|) \cdot |V(G)|^{\mathcal{O}(1)}$ , onde  $f$  é uma função computável que depende de  $|V(H)|$ .

## Problema da Bissecção Mínima

Dado um grafo  $G$  com  $n$  vértices e um inteiro  $k$ .

**Pergunta:** Existe uma partição de  $V(G)$  em  $A$  e  $B$  tal que :

- $\lfloor n/2 \rfloor \leq |A|, |B| \leq \lceil n/2 \rceil$  e
- o número de arestas entre  $A$  e  $B$  é no máximo  $k$ ?

Consideraremos uma versão mais geral:

- Pode haver arestas múltiplas;
- Cada vértice  $v$  tem um peso inteiro  $w(v) \geq 0$ ;
- $A$  e  $B$  são tais que  $\lfloor w(V(G))/2 \rfloor \leq s(A), (B) \leq \lceil w(V(G))/2 \rceil$

## Um lema de shifting com contração de arestas

### Lema

Seja  $G$  planar e  $k \geq 0$ . Então o conjunto de arestas de  $G$  pode ser particionado em  $k + 1$  subconjuntos tais que a **contração** de qualquer um deles induz um grafo com largura de árvore no máximo  $ck$ , para alguma constante  $c > 0$ . Ainda, a partição bem como uma tal decomposição pode ser obtida em tempo polinomial.

Combinamos com um algoritmo para largura de árvore fixa.

### Lema

O Problema da Bisseção Mínima pode ser resolvido em tempo  $2^t \cdot W \cdot n^{O(1)}$  em multigrafo com  $n$  vértices com largura de árvore  $t$  e cujo maior peso de um vértice é  $W$ .

### Teorema

*Bisseção Mínima em grafos planares pode ser resolvida em tempo  $2^{\mathcal{O}(k)} \cdot W \cdot n^{\mathcal{O}(1)}$ , onde  $W$  é o maior peso de um vértice.*