



1

## Tree decomposition

- Tree width
- Small tree decompositions
- Proprietăți ale tree descompunerilor
- Tree descompuneri cu rădăcină
- Aplicații

### Definiție

O tree decomposition a unui graf  $G = (V, E)$  este o pereche  $\mathcal{T} = (T, \{V_t : t \in T\})$ , unde  $T$  este un arbore iar  $\{V_t : t \in V(T)\}$  este o familie de submulțimi de noduri ale lui  $G$ ,  $V_t \subseteq V$ ,  $\forall t \in T$  astfel ca:

- (Acoperirea nodurilor)  $V = \bigcup_{t \in V(T)} V_t$ ;
- (Acoperirea muchiilor) Pentru orice muchie  $e \in E$ , ambele capete ale lui  $e$  sunt conținute într-o mulțime  $V_t$ , pentru un anumit  $t \in V(T)$ .
- (Coerența) Fie  $t_1, t_2, t_3$  trei noduri din  $T$  astfel ca  $t_2$  se află pe drumul dintre  $t_1$  și  $t_3$  în  $T$ . Atunci, dacă  $v \in V$  se află în  $V_{t_1}$  și  $V_{t_3}$ ,  $v$  se află și în  $V_{t_2}$ .

### Remarci

Proprietatea de coerență poate fi reformulată astfel:

- (Coerența') Fie  $t_1, t_2, t_3 \in V(T)$  așa încât  $t_2$  aparține drumului dintre  $t_1$  și  $t_3$  în  $T$ . Atunci  $V_{t_1} \cap V_{t_3} \subseteq V_{t_2}$ .
- (Coerența'') Pentru orice  $x \in V$ , subgraful lui  $T$  indus de  $\{t \in V(T) : x \in V_t\}$  este (un subarbore al lui  $T$ ) conex.

Mulțimile  $V_t$  ale descompunerii sunt numite **bags**.

### Definiție

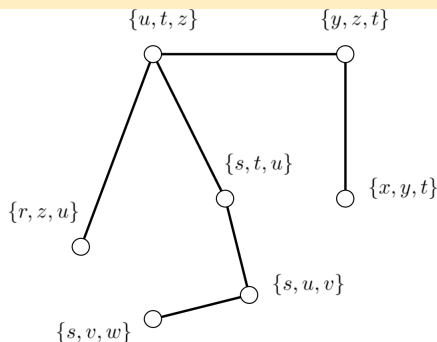
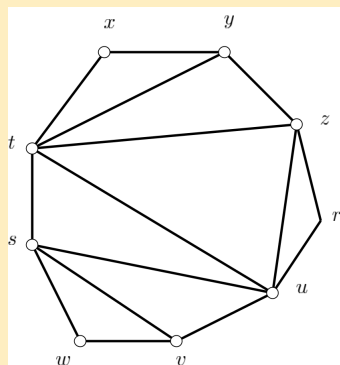
Fie  $\mathcal{T} = (T, \{V_t : t \in T\})$  o tree-descompunere a lui  $G$ , **lățimea** (**width**) lui  $\mathcal{T}$  este

$$\text{width}(\mathcal{T}) = \max_{t \in V(T)} (|V_t| - 1).$$

## Definiție

**Tree-width** a unui graf  $G$ , este cea mai mică lățime a unei *tree-descompuneri* a lui  $G$ :

$$tw(G) = \min \{ width(\mathcal{T}) : \mathcal{T} \text{ tree-descompunere a lui } G \}.$$



## Remarcă

$tw(G) = 0$  dacă și numai dacă  $E(G) = \emptyset$ .

## Propoziție

Dacă  $G$  este o pădure cu  $E(G) \neq \emptyset$ , atunci  $tw(G) = 1$ .

**Demonstrație.** Din observația de mai sus  $tw(G) \geq 1$ . Dacă  $G$  este un arbore, atunci

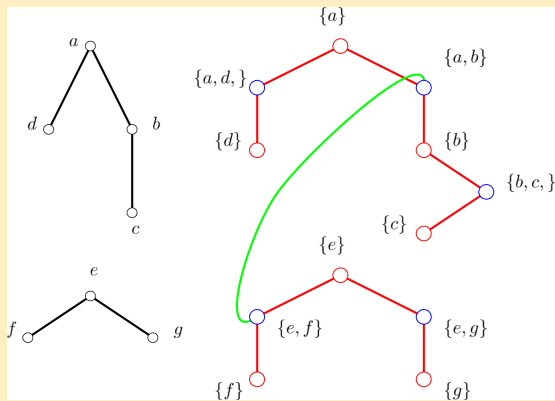
- fie  $T$  obținut din  $G$  prin redenumirea fiecărui nod  $v \in V(G)$  prin  $t_v$ ,
- inserează pe fiecare muchie  $t_u t_v$  ( $uv \in E(G)$ ) un vârf nou  $t_{uv}$ ,
- setează  $V_{t_u} = \{u\}$  pentru orice  $t_u$  asociat lui  $u \in V(G)$  și  $V_{t_{uv}} = \{u, v\}$  pentru orice  $t_{uv} \in V(T)$  asociat lui  $uv \in E(G)$ .
- $(T, \{V_t : t \in V(T)\})$  este o tree-descompunere a lui  $G$  cu lățimea 1.

## Tree-width

**Demonstrație (continuare).** O tree-descompunere a unei păduri cu  $p$  componente se poate obține adăugând arbitrar  $p - 1$  muchii la tree-descompunerile componentelor (fără a crea circuite).



\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph



### Definiție

O tree-descompunere,  $\mathcal{T} = (T, \{V_t : t \in V(T)\})$ , este mică (small) dacă nu există noduri distincte  $t_1, t_2 \in V(T)$  astfel ca  $V_{t_1} \subseteq V_{t_2}$ .

Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

### Propoziție

Dată o tree-descompunere a lui  $G$ , o tree-descompunere small a lui  $G$  de aceeași lățime poate fi construită în timp polinomial.

Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

**Demonstrație.** Fie  $\mathcal{T} = (T, \{V_t : t \in V(T)\})$  o tree-descompunere a lui  $G$  cu  $V_{t_1} \subseteq V_{t_2}$  pentru  $t_1, t_2 \in V(T)$ ,  $t_1 \neq t_2$ . Putem presupune că  $t_1 t_2 \in E(T)$  (altfel, putem determina două noduri adiacente cu această proprietate considerând drumul de la  $t_1$  la  $t_2$ ).

Contractând  $t_1 t_2$  într-un nod nou  $t_{12}$  cu  $V_{t_{12}} = V_{t_2}$ , se obține o tree-descompunere a lui  $G$  mai mică (conține mai puține perechi de noduri  $(t'_1, t'_2)$  cu  $V_{t'_1} \subseteq V_{t'_2}$ ).

**Demonstrație (continuare).** Repetăm această reducere până se obține o tree-descompunere small.



### Propoziție

Dacă  $\mathcal{T} = (T, \{V_t : t \in V(T)\})$  este o tree-descompunere small a lui  $G$ , atunci  $|T| \leq |G|$ .

**Demonstrație.** Prin inducție după  $n = |G|$ . Dacă  $n = 1$ , atunci  $|T| = 1$ . În pasul inductiv, pentru  $n \geq 2$ , considerăm o frunză  $t_1$  a lui  $T$  cu vecinul  $t_2$ .  $(T - t_1, \{V_t : t \in V(T - t_1)\})$  este o tree-descompunere small a lui  $G' = G \setminus (V_{t_1} \setminus V_{t_2})$ . Din ipoteza inductivă  $|T - t_1| \leq |G'|$ , astfel

$$|T| = |T - t_1| + 1 \leq |G'| + 1 \leq |G|.$$



## Remarci

- dacă graful  $H$  se obține din  $G$  prin contractarea unei muchii  $uv$  în  $z$ , atunci  $tw(H) \leq tw(G)$ : într-o tree-descompunere a lui  $G$ , inserăm  $z$  în fiecare bag care conține pe  $u$  sau  $v$  și ștergem apoi pe  $u$  și  $v$  din orice bag pentru a obține o tree-descompunere a lui  $H$ .
- dacă  $H$  este un subgraf al lui  $G$ , atunci  $tw(H) \leq tw(G)$ .

## Definiție

$H$  este un **minor** al unui graf  $G$  dacă se poate obține din  $G$  prin ștergerea și contractarea succesivă a unor muchii din  $G$ .

## Corolar

Dacă  $H$  este un minor al grafului  $G$ , atunci  $tw(H) \leq tw(G)$ .

**Demonstrație.** Folosind observațiile de mai sus.





## Tree-width este FPT (fixed-parameter tractable)

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

### Lemă

Pentru orice  $k \in \mathbb{N}^*$ ,  $TW_k$  este o familie închisă la minori.

### Teoremă

(Bodlaender) Pentru orice  $k$  fixat, problema determinării dacă  $G$  este în  $TW_k$  sau nu poate fi rezolvată în timpul  $\mathcal{O}(f(k) \cdot n)$ .

**Proofs.** Omisă. ( $f(k)$  este o funcție exponențială în  $k$ .)

Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru

Notăție: Fie  $\mathcal{T} = (T, \{V_t : t \in V(T)\})$  o tree-descompunere of  $G$ . Dacă  $T'$  este un subgraf al lui  $T$ ,  $G_{T'}$  este subgraful lui  $G$  indus de mulțimea de noduri  $\bigcup_{t \in V(T')} V_t$ .

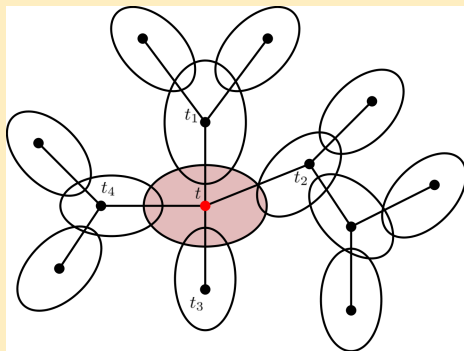
## Proprietăți ale tree descompunerilor

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

### Teoremă

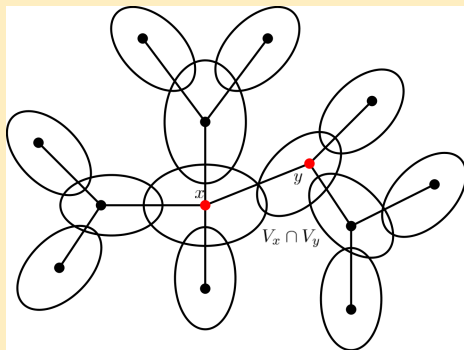
(**Proprietatea separării nodurilor.**) Presupunem că  $T - t$  are componentele conexe  $T_1, T_2, \dots, T_p$ . Atunci subgrafurile  $G_{T_1} - V_t, G_{T_2} - V_t, \dots, G_{T_p} - V_t$  nu au noduri în comun și nici muchii între ele.

Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -



### Teoremă

(Proprietatea separării muchiilor.) Fie  $X$  și  $Y$  componentele conexe ale lui  $T$  după ștergerea unei muchii  $xy \in E(T)$ . Atunci, ștergerea nodurilor din  $V_x \cap V_y$  deconectează  $G$  în două subgrafuri  $H_X = G_X - V_x \cap V_y$  și  $H_Y = G_Y - V_x \cap V_y$ . Cu alte cuvinte  $H_X$  și  $H_Y$  nu au noduri în comun și nici muchii între ele.





### Definiție

O **tree-descompunere cu rădăcină** a lui  $G$  este o tree-descompunere  $\mathcal{T} = (T, \{V_t : t \in V(T)\})$  a lui  $G$ , în care un anumit nod  $r$  al lui  $T$  este declarat rădăcină.

Notății: Fie  $t$  un nod dintr-o tree-descompunere cu rădăcină  $\mathcal{T} = (T, \{V_t : t \in V(T)\})$ .

- $T_t$  este subarborele lui  $T$  cu rădăcina în  $t$ .
- $G[t]$  este subgraful lui  $G$  indus de nodurile din  $\bigcup_{x \in V(T_t)} V_x$  (i. e.,

$$G[t] = G_{T_t}).$$

- Reamintim: o  $p$ -colorare a nodurilor unui graf  $G = (V, E)$  este o funcție  $c : V \rightarrow \{1, 2, \dots, p\}$  astfel ca  $c(u) \neq c(v)$  pentru orice  $uv \in E$ .
- Fie  $H'$  și  $H''$  două subgrafuri ale lui  $G$ , cu  $p$ -colorările  $c'$  și  $c''$ , respectiv.  $c''$  este  $c'$ -compatibilă dacă pentru orice  $v \in V(H') \cap V(H'')$  avem  $c'(v) = c''(v)$ .
- Fie  $\mathcal{T} = (T, \{V_t : t \in V(T)\})$  tree-descompunere cu rădăcină a lui  $G$ . Pentru orice  $t \in T$  și orice  $p$ -colorare  $c$  a lui  $G_t$ , definim

$$Prev_t(c) = \begin{cases} 1, & \text{dacă } G[t] \text{ are o } p\text{-colorare } \bar{c}, c\text{-compatibilă} \\ 0, & \text{altfel.} \end{cases}$$

### Propoziție

$Prev_u(c) = 1$  dacă și numai dacă pentru orice copil  $v$  al lui  $u$ , există o colorare  $\bar{c}$  a lui  $G_v$ ,  $c$ -compatibilă cu  $Prev_v(\bar{c}) = 1$

**Demonstrație.** " $\implies$ " dacă  $\gamma$  este colorare a lui  $G[u]$ ,  $c$ -compatibilă, cum  $G_v$  este un subgraf al lui  $G[u]$ , atunci restricția lui  $\gamma$  la  $G_v$  ne oferă colorarea dorită,  $\bar{c}$ .

" $\impliedby$ " Presupunem că  $u$  are exact doi copii  $v$  și  $w$ , și că avem două colorări  $\bar{c}'$  și  $\bar{c}''$ ,  $c$ -compatibile, respectiv (demonstrația este similară pentru mai mulți copii).

Deoarece  $(T, \{V_t : t \in V(T)\})$  este o tree-descompunere,  $V(G[v]) \cap V(G[w]) \subseteq V_u$ , urmează că  $\bar{c}'$  este  $\bar{c}''$ -compatibilă.

Combinând  $\bar{c}'$  și  $\bar{c}''$  obținem  $\bar{c} : V(G[u]) \rightarrow \{1, 2, \dots, p\}$ . Cum  $(T, \{V_t : t \in V(T)\})$  este o tree-descompunere, nu există muchii  $xy \in E(G)$  cu  $x \in V(G[v]) - V_u$  și  $y \in V(G[w]) - V_u$ , deci  $\bar{c}$  este o  $p$ -colorare a lui  $G[u]$ .



## Teoremă

Dacă  $G$ , un graf de ordin  $n$ , are o tree-descompunere small  $(T, \{V_t : t \in V(T)\})$  cu lățimea  $w$ , atunci putem decide dacă  $G$  este  $p$ -colorabil în timpul  $\mathcal{O}(p^{w+1} \cdot n^{\mathcal{O}(1)})$ .

\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

**Demonstrație.** Transformăm  $(T, \{V_t : t \in V(T)\})$  într-o a tree-descompunere cu rădăcină  $(r)$ . Pentru orice  $v \in V(T)$  și orice  $p$ -colorare  $c$  a lui  $G_v$ , determinăm  $Prev_v(c)$ : plecând din frunzele lui  $T$ , și apoi folosind proprietatea de mai sus pentru celelalte noduri, într-o ordine corespunzătoare.

$G = G[r]$  este  $p$ -colorabil dacă și numai dacă  $Prev_r(c) = 1$  pentru o colorare  $c$ . Testarea dacă  $c$  este  $G_v$  colorare și determinarea lui  $Prev_v(c)$  pot fi făcute în timpul  $\mathcal{O}(n^{\mathcal{O}(1)})$ , astfel complexitatea timp totală este dată în principal de numărul de candidați pentru  $c$ , adică  $p^{|V_v|}$ .

Complexitate:  $|V(T)| \cdot p^{w+1} \cdot n^{\mathcal{O}(1)} = \mathcal{O}(p^{w+1} \cdot n^{\mathcal{O}(1)})$ .



