

10 Dekompozice grafů, algoritmy a minory

Další autorův výběr tématu nás zavede blíže ke strukturální teorii grafů, k různým jejich dekompozicím, grafovým minorům a navazujícím efektivním algoritmům pro jinak těžké problémy. Obecnou inspirací nám je poměrně zřejmý fakt, že většinu jinak těžkých problémů lze řešit snadno a efektivně na stromech. Proto se podíváme na grafy, které jsou svým způsobem „stromům blízké“, ve smyslu existence jejich vhodné dekompozice.

Vrcholem této lekce je formulace hlavního výsledku takzvané „Graph Minors Theory“ od Robertsona a Seymoura, který lze bez nadsázky prohlásit za asi největší výsledek, kterého dosud teorie grafů dosáhla (i v porovnání s Větou o čtyřech barvách). □

Stručný přehled lekce

- Úvodní zamyšlení nad řešením obtížných problémů.
- Stromová šířka grafu z mnoha stran. Některé jiné „šířkové“ param.
- Ukázky použití dekompozic grafů na efektivní algoritmy.
- Něco o grafových minorech a strukturální teorii.

10.1 Obtížné problémy na speciálních grafech

V Lekci 7 jsme uvedli některé „neřešitelně obtížné“, přesněji \mathcal{NP} -těžké grafové problémy, například 3-obarvení grafu nebo nezávislá množina vrcholů. \square

Oba tyto problémy snadno vyřešíme na intervalových grafech.

Algoritmus 10.1. Nalezení nezávislé množiny v intervalovém grafu

Předpokládejme, že graf G je daný svou intervalovou reprezentací (v případě potřeby je možno tuto reprezentaci efektivně sestrojít). Maximální nezávislou množinu nalezneme následovně.

- Uspořádáme intervaly reprezentující G podle jejich pravých konců. \square
- Do nezávislé množiny hladově (v tomto uspořádání) vložíme vždy první interval neprotínající se s předchozími vybranými.

Také na obecnějších chordálních grafech můžeme navrhnout rychlé a povětšinou i snadné algoritmy.

Algoritmus 10.2. Určení barevnosti chordálního grafu

- Snadno zkonstruujeme simplicialní dekompozici našeho grafu G (Věta 9.4). \square
- Graf G je k -degenerovaný, kde $k = \omega(G) - 1$ je největší stupeň simplicialního vrcholu v některém kroku simplicialní dekompozice G . Hladově tudíž můžeme G obarvit $k + 1$ barvami a to je optimální (neboť máme v G kliku velikosti $k + 1$). \square

Algoritmus 10.3. Nalezení nezávislé množiny chordálního grafu

- Opět zkonstruujeme simplicialní dekompozici našeho grafu G (Věta 9.4).
- V pořadí této dekompozice hladově přidáváme vrcholy do nezávislé množiny. (Tento postup je přímým zobecněním Algoritmu 10.1, viz také Věta 9.5.) \square

Možná zobecnění?

Zajímavou a užitečnou otázkou teď je, jak takové postupy zobecnit na širší třídy grafů, které mají nějakou specifickou omezující (ale ne příliš) vlastnost – „**parametr**“.

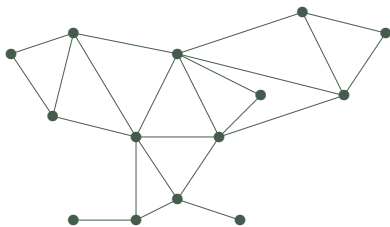
10.2 Tree-width – čtyři definice

Název „tree-width“ byl zaveden Robertsonem a Seymourem počátkem 80-tých let, ale pak se ukázalo, že ekvivalentní definice již uvažovali matematici léta před nimi, například v souvislosti s takzvanými „ k -trees“ nebo se simplicialními dekompozicemi. (Důsledkem tohoto vývoje je také bohatost různých definic stejného pojmu. . .) □

Připomeňme, že velikost největší kliky v grafu G se označuje $\omega(G)$.

Definice: *Stromovou šířkou (tree-width)* grafu G nazveme nejmenší přirozené k takové, že existuje **chordální** graf H s $\omega(H) = k + 1$ obsahující G jako podgraf ($H \supseteq G$).

Například každý podgraf následujícího chordálního grafu má tree-width ≤ 2 :



□

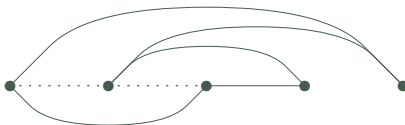
Kde je však v této definici nějaký „strom“? Je, ale skrytý – podívejte se na Větu 9.5 popisující chordální grafy jako průnikové grafy podstromů ve stromě.

Jinou možnost popisu ukazuje:

Definice: Vrcholy $V(G)$ grafu G uspořádáme do posloupnosti (permutace) (v_1, v_2, \dots, v_n) . Pro $i = 1, 2, \dots, n$ definujeme $\ell(v_i)$ jako počet všech indexů $j \in \{1, \dots, i-1\}$ takových, že vrcholy v_i a v_j jsou v G spojeny cestou používající pouze vrcholy z množiny $\{v_j, v_i, v_{i+1}, \dots, v_n\}$. \square

Druhou *stromovou šířkou* grafu G nazveme nejmenší hodnotu výrazu $\max_v \ell(v)$ přes všechny permutace vrcholů $V(G)$.

Všimněte si, že uspořádání vrcholů z této definice je vlastně zpětnou simplicíální dekompozicí chordálního grafu $H \supseteq G$ z předchozí definice (viz také Věta 9.4).



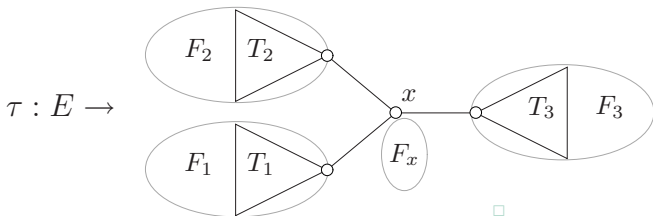
V nakresleném příkladě grafu C_5 vidíme uspořádání vrcholů se šířkou 2. (Tečkované hrany ukazují tzv. *chordální doplnění* grafu, relevantní k první definici tree-width.)

Ještě další, značně odlišný, přístup má tato definice:

Definice: Pro libovolný strom T uvažujeme (libovolné) zobrazení $\tau : E(G) \rightarrow V(T)$. Pro vrchol $t \in V(T)$ označíme T_1, \dots, T_d jednotlivé komponenty lesa $T - t$ a $F_i = \tau^{-1}(V(T_i))$. Označme

$$\ell_\tau(t) = |V(G)| + (d - 1) \cdot c(G) - \sum_{i=1}^d c(G - F_i),$$

kde $c(H)$ značí počet souvislých komponent grafu H .



Třetí *stromovou šířkou* grafu G pak definujeme jako nejmenší možnou, přes všechny dvojice T, τ , hodnotu výrazu $\max_{t \in V(T)} \ell_\tau(t)$.

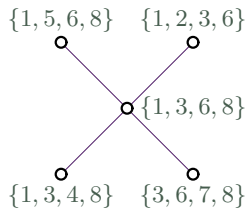
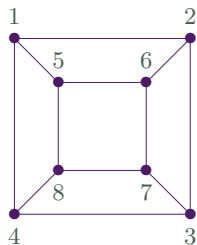
Nakonec si uvedeme ještě původní **definici Robertsona a Seymoura**, která se většinou uvádí jako ta první a hlavní (a na ostatní možné definice málokdy přijde řeč).

Definice 10.4. Stromová dekompozice grafu G .

Stromovou dekompozicí grafu G nazveme strom T spolu se systémem množin \mathcal{X}_t (zvaných „balíky“) pro $t \in V(T)$, kde

- $\mathcal{X}_t \subseteq V(G)$ a $\bigcup_{t \in V(T)} \mathcal{X}_t = V(G)$, \square
- pro každou hranu $e = uv \in E(G)$ je $u, v \in \mathcal{X}_t$ pro nějaké $t \in V(T)$, \square
- (*interpolační* vlastnost) pro každý vrchol $v \in V(G)$ tvoří podmnožina všech $t \in V(T)$ s $v \in \mathcal{X}_t$ podstrom v T . \square

Šířkou dekompozice T, \mathcal{X} rozumíme největší hodnotu $|\mathcal{X}_t| - 1$ pro $t \in V(T)$ a čtvrtou *stromovou šířkou* grafu G nazveme nejmenší možnou šířku stromové dekompozice G .



Přes veškerou zdánlivou odlišnost uvedených definic platí následovně.

Věta 10.5. *Všechny čtyři výše uvedené definice stromové šířky definují přesně tutéž hodnotu, pokud graf G má neprázdnou množinu hran.*

Důkaz plného znění této věty je však nad rámec našeho textu.

O četnících a zloději

Na závěr si představme *hru na četníky a zloděje* s těmito pravidly: Zloděj se bleskovou rychlostí pohybuje po hranách grafu přes vrcholy neobsazené četníky (jeho pohyb vždy skončí ve vrcholu, ne „uprostřed“ hrany). Naopak četníci se grafem vůbec nepohybují, jen přilétají do a odlétají z vrcholů helikoptérou. Zloděj je chycen ve svém vrcholu z přiletivším četníkem, pokud jsou i všichni sousedé z zrovna obsazeni četníky.

Věta 10.6. *Nejmenší počet četníků potřebných k zaručenému chycení zloděje v grafu G je roven stromové šířce G plus 1.*

10.3 Některé další parametry

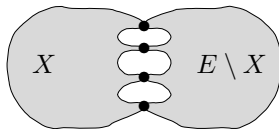
Definice: *Cestní* dekompozici a šířku (path-width) grafu G definujeme stejně jako v Definici 10.4, jen požadujeme navíc, aby T byla **cesta**. ◻

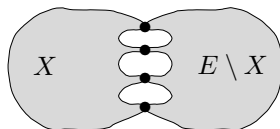
Definice: Vrcholy $V(G)$ grafu G uspořádáme do posloupnosti (permutace) (v_1, v_2, \dots, v_n) . *Bandwidth* grafu G definujeme jako nejmenší hodnotu výrazu $\max_{v_i v_j \in E(G)} |i - j|$ přes všechny permutace vrcholů $V(G)$. ◻

Větvené dekompozice

Graf je *kubický*, pokud má všechny vrcholy stupně 3. Strom je *podkubický*, pokud má všechny vrcholy stupně ≤ 3 .

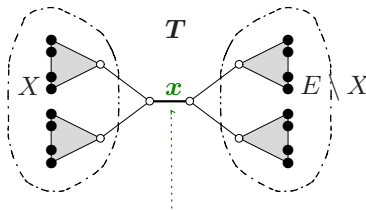
Definice: Pro libovolný graf G a podmnožinu $X \subseteq E(G)$ definujeme *funkci souvislosti* $\lambda_G(X)$ jako počet vrcholů G , které jsou konci některých hran z X i hran z $E(G) \setminus X$ (*separace* množiny X).





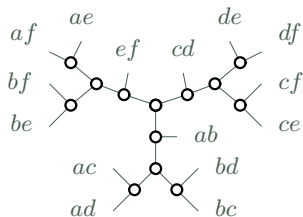
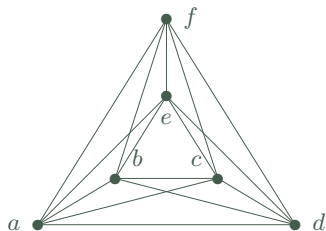
Definice 10.7. Větvená dekompozice grafu G

Nechť T je podkubický strom a $\tau : E(G) \rightarrow L(T)$ je bijekce hran grafu G do listů $L(T)$ stromu T . Pro každou hranu x stromu T definujeme šířku x jako $\lambda_G(x)$, kde $X = \tau^{-1}(V(T_1))$ pro jednu z komponent T_1, T_2 lesa $T - x$.



$$\text{šířka}(x) := \lambda_G(x) = \lambda_G(X) = \lambda_G(E \setminus X) \quad \square$$

Pak šířkou dekompozice T, τ je maximální šířka ze všech hran T a **větvenou šířkou** grafu G je nejmenší možná šířka větvené dekompozice G . \square



Věta 10.8. Pokud graf G má stromovou šířku t a větvenou šířku $b > 1$, tak

$$b \leq t + 1 \leq \left\lfloor \frac{3}{2} b \right\rfloor. \square$$

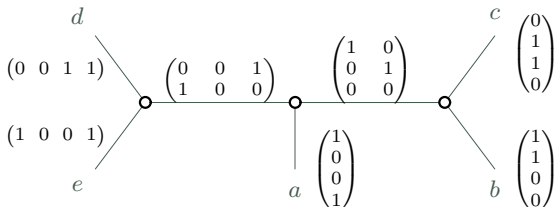
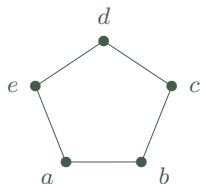
Jiný přístup

Nechť G je graf a $X \subseteq V(G)$. Jako funkci **hodnosti řezu** $\gamma_G(F)$ na množině F označíme hodnotu $X \times (V \setminus X)$ -matice $A = (a_{i,j})$ nad binárním tělesem $GF(2)$, kde $a_{u,v} = 1$ pro $u \in X$ a $v \in V \setminus X$, právě když uv je hranou v G . \square

Definice 10.9. Ranková dekompozice grafu G

Nechť T je podkubický strom a $\tau : V(G) \rightarrow L(T)$ je bijekce **vrcholů** grafu G do listů $L(T)$ stromu T . Pro každou hranu x stromu T definujeme šířku x jako $\gamma_G(X)$, kde $X = \tau^{-1}(V(T_1))$ pro jednu z komponent T_1, T_2 lesa $T - x$. \square

Pak šířkou dekompozice T, τ je maximální šířka ze všech hran T a **rankovou šířkou** grafu G je nejmenší možná šířka rankové dekompozice G .



10.4 Efektivní algoritmy na dekompozicích

Již víme z 7.17, že určení velikosti největší nezávislé množiny v grafu je \mathcal{NP} -úplný problém. Stromová dekompozice fixní šířky však tento problém umožňuje řešit velice snadno.

Algoritmus 10.10. Nezávislá množina na stromové dekompozici

Danou stromovou dekompozici vstupního grafu si libovolně „zakořeníme“.

- V každém listě dekompozice vyřešíme problém hrubou silou v konstantním čase. \square
- Ve směru od listů ke kořeni sbíráme následující **informaci**:
V každém balíku B dekompozice, pro každou $X \subseteq B$, velikost největší nezávislé množiny I v grafu indukovaném na podstromu pod B takové, že $I \cap B = X$. \square
- Vzhledem k interpolační vlastnosti naší dekompozice lze výše popsanou informaci v každém balíku dekompozice zjistit v **konstantním čase** pouze ze znalosti stejné informace ze všech jeho potomků v dekompozici.

Výsledný algoritmus pracuje v čase úměrném počtu uzlů dekompozice, tedy v lineárním čase! \square

Další problém hledání párování v grafu sice je polynomiálně řešitelný, ale zjišťování počtu všech párování už je **#P-úplné**, neboli stejně těžké (beznadějně) jako výpočet permanentu matice nebo spočítání všech řešení SAT problému.

Algoritmus 10.11. Počet párování na větvené dekompozici

Opět si větvenou dekompozici vstupního grafu libovolně „zakořeníme“.

- V každém listě dekompozice je řešení triviální. □
- Ve směru od listů ke kořeni sbíráme následující **informaci**:
V každé hraně dekompozice, pro každou podmnožinu $X \subseteq S$ vrcholů separace S indukované touto hranou v dekompozici, počet všech párování, která ze separace S „obsazují“ právě vrcholy X . □
- Opět lze výše popsanou informaci na každé hraně dekompozice zjistit v **konstantním čase** pouze ze znalosti stejné informace z obou jejich podstromů v dekompozici (vzájemným vynásobením a sečtením počtů).

Výsledný algoritmus pracuje v čase úměrném počtu hran dekompozice, tedy opět v lineárním čase. □

Jeden všeobecný výsledek

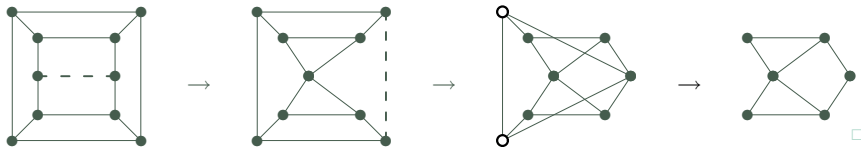
Nápadná vzájemná podobnost předchozích algoritmů určitě není náhodná, chtěli jsme jimi ilustrovat jeden důležitý obecný princip, který objevili postupně [Courcelle / Arnborg, Lagergren, Seese / Borie, Parker, Tovey].

Věta 10.12. *Každá vlastnost grafů, která je vyjádřitelná v tzv. (E)MSO jazyce, se dá vypočítat v lineárním čase pro všechny grafy omezené stromové šířky.*

Pro zjednodušení nebudeme přesně definovat, co (E)MSO jazyk znamená, ale zhruba jde o jazyk, který má dovoleno kvantifikovat přes podmnožiny vrcholů a hran grafu a enumerovat počty prvků množin. Většina grafových vlastností, které jsme zatím probírali, spadá do této kategorie.

10.5 Minory v grafech

Definice: Říkáme, že graf G je *minorem* grafu H , pokud lze G získat z H kontrakcemi hran a vypouštěním vrcholů a hran.



Robertson–Seymourova věta

Věta 10.13. Mějme libovolnou grafovou vlastnost ϕ , která je *uzavřená na minory* (tj. pokud G má ϕ , pak každý minor G má také ϕ).

Pak existuje *konečně* mnoho grafů F_1, \dots, F_ℓ (*zakázané minory*) takových, že G má ϕ právě když G neobsahuje minor isomorfní žádnému z F_1, \dots, F_ℓ .

Mimo jiné lze tudíž vlastnost ϕ *rozhodnout v čase $O(n^3)$* pro každý n -vrcholový graf G .

□

Poznámka: Tato věta je zcela *nekonstruktivní*, neboli nepodává žádný návod, jak zmíněný algoritmus sestavit!