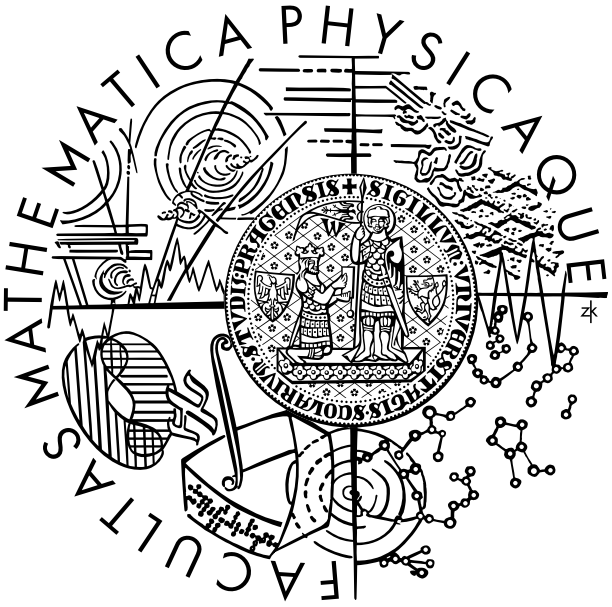
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Ondřej Suchý

PARAMETRIZOVANÁ SLOŽITOST V TEORII GRAFŮ

Katedra aplikované matematiky

Vedoucí dipl. práce: **Prof. RNDr. Jan Kratochvíl, CSc.**
Studijní program: **Matematika**
Studijní obor: **Matematické struktury**

Děkuji panu prof. Kratochvílovi za vedení, zapůjčení literatury a spoustu cenných rad a námětů k obsahu práce, k psaní matematických textů, i jiných. Také bych rád poděkoval rodičům za jejich podporu.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů.

Souhlasím se zapůjčováním práce.

V Praze, dne 7.srpna 2006 Ondřej Suchý

Obsah

1 Úvod **5**

2 Značení **7**
1 Výpočetní složitost 7
2 Teorie grafů 8

3 Parametrizovaná složitost **10**
1 Základní definice 10
2 Parametrizace 11
3 Přehled metod 13
3.1 Redukce na jádro problému 13
3.2 Omezený prohledávací strom 15
3.3 Stromový rozklad a dynamické programování 16
3.4 Iterativní komprese 19
3.5 Hladová lokalizace 21
3.6 Teorie grafových minorů 22
3.7 Barevné kódování 23
3.8 Celočíselné lineární programování 24
4 Parametrizovaná nedostupnost 25

4 Aplikace na Seidelovo přepnutí **28**
1 Definice 28
2 Známé výsledky 29
3 Přepnutí na graf se všemi vrcholy stupně $\leq k$ 34
4 Přepnutí na k-regulární graf 38
5 Přepnutí na grafy se všemi vrcholy stupně $\geq k$ 40
6 Přepnutí na graf s nejvýše k hranami 42
7 Přepnutí na graf prostý zakázaného podgrafu 44

5 Závěr **47**

Literatura **49**

Název práce: Parametrizovaná složitost v teorii grafů
Autor: Ondřej Suchý
Katedra (ústav): Katedra aplikované matematiky
Vedoucí diplomové práce: Prof. RNDr. Jan Kratochvíl, CSc.
e-mail vedoucího: honza@kam.mff.cuni.cz

Abstrakt: Seidelovo přepnutí množiny vrcholů je operace, která z grafu odebere hrany vycházející z této množiny a přidá do něj hrany tam, kde mezi množinou a zbytkem grafu nebyly. Ostatní hrany nejsou touto operací dotčeny. Parametrizovaná složitost se ptá, zda lze exponenciální část algoritmů pro těžké problémy omezit nějakou funkcí pouze zvoleného parametru, u nějž lze očekávat malé hodnoty. Tato práce zkoumá složitost otázek, zda lze zadaný graf převést na graf s nějakou vlastností P pomocí Seidelova přepnutí, z parametrizovaného hlediska. Nejdříve krátce shrneme dosud známé výsledky. Pak předvedeme parametrizovanou dostupnost přepnutí na regulární grafy, grafy s omezeným stupněm vrcholů, s omezeným počtem hran a grafy prosté zakázaného podgrafu. Krátce podáme základní definice a postupy parametrizované složitosti.

Klíčová slova: parametrizovaná složitost, graf, Seidelovo přepnutí

Title: Parameterized complexity in graph theory

Author: Ondřej Suchý
Department: Department of Applied Mathematics
Supervisor: Prof. RNDr. Jan Kratochvíl, CSc.
Supervisor's e-mail address: honza@kam.mff.cuni.cz

Abstract: Seidel's switching of a set of vertices of a graph is an operation which deletes the edges leaving this set from the graph and adds those edges between the set and the rest of graph that weren't in the original graph. Other edges remain untouched by this operation. Parameterized complexity asks if the exponential part of algorithms for hard problems can be bounded by some function only of selected parameters, which we assume small. In this thesis, we study the complexity of a question if the given graph can be turned into a graph with some property P using Seidel's switching, from the parameterized point of view. First we give an overview of known results. Then we show fixed-parameter tractability of switching to a regular graph, a graph with bounded degree of vertices, bounded number of edges and a graph without a forbidden subgraph. We briefly introduce basic definitions and techniques of parameterized complexity.

Keywords: parameterized complexity, graph, Seidel's switching

Kapitola 1

Úvod

Když se začaly rozvíjet počítače a jejich využití k řešení různých problémů, brzy bylo jasné, že ne všechny úlohy jsou stejně obtížné. Problémy se postupně začaly dělit na ty, ke kterým existuje algoritmus, který je řeší a běží v polynomiálním čase a ty, pro které takový algoritmus není znám a je tedy potřeba typicky použít algoritmy exponenciální. Průkopníkem tohoto rozdělení byl v šedesátých letech Edmonds [20, 21]. Na počátku let sedmdesátých byl tento postup formalizován a zavedeny třídy problémů P a NP, spolu s třídou NP-úplných problémů. Cook [13, 14] prokázal existenci NP-úplného problému a doložil tak opodstatněnost příslušných definic. Ještě v témže roce 1972 ukázal Karp [38] že NP-úplných problémů je celá řada. V roce 1979 pak Garey a Johnson vydali svoji knihu [27], která je dodnes považována za „bibli“ složitosti.

Ale postupem času bylo stále častěji potřeba řešit i těžké úlohy, ke kterým polynomiální algoritmy známy nejsou. Jednou z možností, jak k takovým problémům přistupovat, je i parametrizovaná složitost. Tento přístup se poprvé objevil v článku Abrahmson et al. [1], dále rozvíjeli a formalizovali ho především Downey a Fellows v několika článcích [15, 16, 17]. Jejich kniha [18], kterou vydali v roce 1999 shrnuje všechny předchozí výsledky a považuje se za základní kámen oboru, který dnes již má i své samostatné konference. Od té doby však prošel značným vývojem, včetně nových technik, která shrnuje například Niedermeier [45], nebo také Sloper [57].

Hlavní myšlenka spočívá v tom, že ne všechna zadání (všechny instance) daného problému jsou stejně těžká. Snahou je najít parametr na kterém závisí, jak obtížné bude danou úlohu řešit a omezit onu pravděpodobně nevyhnutelnou nepolynomiální část běhového času nějakou funkcí pouze tohoto parametru. Ten může být přímo součástí zadání, nebo v něm může být skryt, jako je tomu například v případě stromové šířky grafu a na ní založených algoritmech.

Přístupů k těžkým problémům je samozřejmě více, aproximace, pravděpodobnostní algoritmy, heuristiky, kvantové počítání nebo zkoumání průměrné časové náročnosti algoritmů. Zásadní výhodou parametrizované složitosti oproti jiným

postupům je získání optimálního řešení v garantovaném čase. Navíc narozdíl od některých výše uvedených technik má také svou složitostní teorii, která ukazuje, že pro řadu problémů nelze příznivý výsledek očekávat.

Tato práce se zaměřuje na parametrizovanou složitost v teorii grafů. Druhá kapitola obsahuje nejdůležitější značení a notace, které budeme dále používat. Ve třetí kapitole podáme krátce základní definice a postupy parametrizované složitosti. V kapitole čtvrté aplikujeme parametrizovanou složitost na konkrétní problémy z teorie grafů. Zaměříme se na Seidelovo přepnutí a otázky s ním spojené. Nejdříve krátce shrneme dosud známé výsledky. Pak předvedeme parametrizovanou dostupnost přepnutí na regulární grafy, grafy s omezeným stupněm vrcholů, s omezeným počtem hran a grafy prosté zakázaného podgrafu.

Kapitola 2

Značení

Jsou-li A a B dvě množiny, pak značíme $A \setminus B$ jejich rozdíl a $A \triangle B$ symetrický rozdíl. Zápis $|A|$ označuje velikost množiny. Symbol $\mathbb{N}$ značí přirozená čísla $\{1, 2, 3, ...\}$, symbol $\mathbb{N}_0$ celá nezáporná čísla $\{0, 1, 2, 3, ...\}$. Ve většině úvah však rozlišení těchto dvou množin nebude mít významnější vliv.

1 Výpočetní složitost

Je-li Σ nějaká abeceda, pak Σ^* označuje *množinu všech slov* nad touto abecedou. *Jazyk* L je libovolná podmnožina Σ^*. Pro $x \in L$ *slovo* označuje $|x|$ jeho *délku*, neboli počet znaků, ze kterých se skládá.

Naše výpočty budeme provádět na jednoprocesorovém stroji s náhodným přístupem (RAM), kde každá jednoduchá operace (aritmetická, vyhodnocení podmínky, přístup do paměti, apod.) trvá jednotkový čas. Celočíselný proměnná je u našeho stroje dost velká, aby pojala všechna čísla v našich algoritmech. Nebudeme toho zneužívat například kódováním více čísel do jedné proměnné.

Je-li A algoritmus, pak jeho *časová složitost* je funkce $f : \mathbb{N} \to \mathbb{N}$ taková, že $f(n)$ je maximální počet kroků, které A vykoná na vstupu délky n. Říkáme také, že *algoritmus běží (pracuje) v čase* $f(n)$.

Jsou-li $f, g : \mathbb{N} \to \mathbb{N}$ funkce, píšeme $f(n) = \mathcal{O}(g(n))$, pokud existují konstanty c a n_0 tak, že $\forall n \geq n_0 : f(n) \leq c \cdot g(n)$. Mluvíme-li o algoritmech, které pracují v exponenciálním čase, někdy se v novější literatuře (viz např. Woeginger [58]) používá notace $\mathcal{O}^*(g(n))$, kde vynecháváme polynomiální části funkce.

Algoritmy pracující v čase $\mathcal{O}(n^c)$ pro nějakou konstantu c nazýváme *polynomiální*, *lineární* pracují v čase $\mathcal{O}(n)$, *kvadratické* $\mathcal{O}(n^2)$, *kubické* v $\mathcal{O}(n^3)$ atd.

Jazyk L je *ve třídě P*, pokud existuje polynomiální algoritmus, který rozhodne, zda $x \in L$. Jazyk L je *ve třídě NP*, pokud existuje jazyk L' ve třídě P, takový, že $x \in L \Leftrightarrow \exists y : (x, y) \in L'$. Jazyk je *NP-těžký*, pokud na něj lze každý jazyk z NP

polynomiálně převést. Jazyk je *NP-úplný*, je-li v NP a NP-těžký. Ve složitosti se obvykle jazykům říká *problémy*.

2 Teorie grafů

Neorientovaným grafem G míníme uspořádanou dvojici (V, E), kde $E \subseteq \binom{V}{2}$ je množina dvouprvkových podmnožin V. Prvky V nazýváme *vrcholy* a prvky E *hrany*. Prvky množiny $\binom{V}{2} \setminus E$ budeme někdy nazývat *nehrany*. Nebude-li uvedeno jinak, budeme označovat n počet vrcholů grafu G, m počet jeho hran a přepokládáme, že $V = \{1, ..., n\}$. Graf $\bar{G} = (V, \binom{V}{2} \setminus E)$ nazveme *doplňkem grafu* G.

Dva vrcholy $u, v \in V$ jsou *sousední* (u *je soused* v), pokud $\{u, v\} \in E$, zkráceně $uv \in E$, tj. pokud jsou spojeny hranou. V opačném případě je nazveme *nesousední*. Symbolem $N(v) = \{u | u \text{ je soused } v\}$ označujeme *(otevřené) okolí vrcholu* v, neboli množinu všech jeho sousedů v grafu G. $N[v] = N(v) \cup \{v\}$ je *uzavřené okolí vrcholu* v. Je-li $A \subseteq V$ značíme $N(A) = \{v \notin A | uv \in E \text{ a } u \in A\}$. *Stupeň vrcholu* $\deg v = |N(v)|$ je velikost jeho okolí. Pokud nebude zřejmé, o který graf se jedná budeme výše uvedené symboly doplňovat indexem grafu např. $V_G, N_G(v), \deg_G v$ atd. Vrchol je *izolovaný*, má-li stupeň 0. Graf je *k-regulární*, pokud má všechny vrcholy stupně k.

Zobrazení $f : V_G \to V_H$ je *izomorfizmus grafů* G a H, pokud je to bijekce a $\{u, v\} \in E_G \Leftrightarrow \{f(u), f(v)\} \in E_H$. Graf G je *izomorfní* grafu H, pokud existuje nějaký izomorfizmus grafů G a H.

Řekneme, že graf H je *podgrafem* grafu G, pokud $V_H \subseteq V_G$ a $E_H \subseteq \binom{V_H}{2} \cap E_G$. H je *indukovaný podgraf*, pokud navíc $E_H = \binom{V_H}{2} \cap E_G$. Je-li $A \subseteq V$ pak $G[A] = (A, \binom{A}{2} \cap E_G)$ je *podgraf* G *indukovaný množinou* A. Budeme používat neformální zápisy $G - v$ místo $G[V \setminus \{v\}]$ pro *vymazání vrcholu*, $G \setminus U$ místo $G[V \setminus U]$ a $G - e$ místo $(V, E \setminus \{e\})$ pro *vymazání hrany*, kde v je vrchol, U množina vrcholů a e hrana. Zápis $G + H$ značí disjunktní sjednocení grafů G a H. Pokud nějaký podgraf grafu G je izomorfní grafu H, řekneme, že G obsahuje H jako podgraf (případně jako indukovaný podgraf), nebo jen, že G *obsahuje* H.

Graf $G = (V, \binom{V}{2})$ se nazývá *úplný graf* a značí se K_n. Obsahuje-li graf G podgraf K_k, říkáme tomuto podgrafu *k-klika*. Graf $I_n = \overline{K_n}$ je *diskrétní graf*. I_k jako podgraf se nazývá *k-nezávislá množina* v grafu. Graf $C_n = (\{1, 2, .., n\}, \{\{1, n\}\} \cup \{\{i, i+1\} | 1 \leq i \leq n-1\})$ se nazývá *kružnice* a graf $P_n = C_n - \{1, n\}$ *cesta* na n vrcholech. Obsahuje-li graf kružnici nebo cestu na všech vrcholech, označuje se tato jako *hamiltonovská*. Graf na n vrcholech, který neobsahuje žádnou kružnici, se nazývá *strom*, pokud má $n - 1$ hran, a *les* jinak.

Řekneme, že graf je *bipartitní*, pokud lze vrcholy rozdělit do dvou množin A a B, které obě indukují v grafu nezávislou množinu. Pokud je navíc každý

vrchol množiny A spojen hranou s každým vrcholem z B, říkáme, že je graf *úplný bipartitní* a značíme $K_{|A|,|B|}$.

Kontrakce hrany $e = \{u, v\}$ v grafu $G = (V_G, E_G), e \in E_G$ je graf $H = (V_G \setminus \{u, v\} \cup \{w\}, E_H)$ takový, že $w \notin V_G$ a

$$E_H = (E_G \cup \{\{x, w\} | \{x, u\} \in E_G \text{ nebo } \{x, v\} \in E_G\}) \cap \binom{V_H}{2}.$$

Řekneme, že graf H *je minorem grafu* G, pokud vznikne z G libovolným počtem opakování operací vymazání vrcholu, vymazání hrany a kontrakce hrany v libovolném pořadí.

Kapitola 3

Parametrizovaná složitost

V této kapitole si představíme definice základních pojmů, a také přehled nejdůležitějších technik, které se používají při hledání rychlých parametrizovaných algoritmů.

1 Základní definice

Definice 1.1. *Parametrizovaný problém* P je podmnožina kartézského součinu $\Sigma^* \times \Sigma^*$, kde Σ je nějaká pevně zvolená abeceda. Je-li P parametrizovaný problém, pak o $(I, k) \in P$ hovoříme jako o *instanci*, nebo také konkrétním zadání tohoto problému, I je jeho *hlavní část* a k se nazývá *parametr*.

Poznámka 1.2. Ve většině případů je parametr k přirozené číslo, někdy se tedy parametrizovaný problém zavádí jako $P \subseteq \Sigma^* \times \mathbb{N}$.

Následující příklad ukazuje, jak se obvykle takový parametrizovaný problém zapisuje.

Vrcholové pokrytí
Vstup: Neorientovaný graf $G = (V, E)$ a přirozené číslo k
Parametr: k
Otázka: Existuje $A \subseteq V, |A| \leq k$, která pokrývá všechny hrany, tj. každá hrana má alespoň jeden konec v A?

Definice 1.3. Řekneme, že parametrizovaný problém P je *parametrizovaně dostupný*, pokud existuje algoritmus, který rozhodne o vstupu (I, k) zda patří do P v čase $f(k)|I|^c$, kde c je pevná konstanta a $f(k)$ je funkce nezávislá na celkové délce vstupu $|I|$. Třídu všech parametrizovaně dostupných problémů označujeme $\mathcal{FPT}$.

Toto je standardní způsob, kterým se zavádí parametrizovaná dostupnost. Obě definice jsou převzaty z Downey a Fellows [18], ale v podobné formě se objevují již v jejich předchozích článcích [15, 16, 17]. Definice jsou v této podobě ustálené, novější autoři [45] je přejímají bez větších změn, nebo pouze s přirozeným parametrem [57], tak jako v poznámce 1.2.

Příklad 1.4. VRCHOLOVÉ POKRYTÍ $\in \mathcal{FPT}$, jak bude uvedeno dále.

Problém vrcholového pokrytí je velmi názorný a proto na něm budeme v dalších sekcích (3.1, 3.2, 3.3 a 3.6) ukazovat mnoho z technik, které se používají k důkazům parametrizované dostupnosti. S řešením této úlohy se však váže i zajímavá historie.

Klasický problém vrcholového pokrytí, je uváděn jako NP-úplný v knize Garey a Johnson [27]. V roce 1986 Fellows a Langston [25] použitím teorie grafových minorů (viz sekce 3.6) ukázali, že VRCHOLOVÉ POKRYTÍ lze řešit v čase $\mathcal{O}\left(f(k) \cdot n^3\right)$. O rok později zlepšil jejich výsledek na $\mathcal{O}\left(f(k) \cdot n^2\right)$ použitím stromového rozkladu (sekce 3.3) Johnson [36]. Následoval výsledek Fellowse [24] z roku 1988, $\mathcal{O}\left(2^k \cdot n\right)$ pomocí prohledávacích stromů (sekce 3.2). V roce 1989 S. Buss [9] popsal algoritmus běžící v čase $\mathcal{O}\left(kn + 2^k k^{2k+2}\right)$ založený na redukci na jádro problému (sekce 3.1). Kombinací obou předchozích technik dosáhli Balasubramanian, Downey, Fellows a Raman v roce 1992 (publikováno až v [6]) času $\mathcal{O}\left(kn + 2^k k^2\right)$. V roce 1993 Papadimitriou a Yannakakis [48] nalezli algoritmus pracující v čase $\mathcal{O}\left(3^k \cdot n\right)$ s maximálním párováním jako podprogramem. VRCHOLOVÉ POKRYTÍ je typické tím, že na něj lze použít celou řadu technik a tak šel vývoj rychle kupředu a například Chandran a Grandoni [11] omezili v roce 2005 exponenciální část pouze na 1.28^k.

2 Parametrizace

Klíčovým předpokladem pro skutečnou použitelnost navržených algoritmů je to, že hodnoty parametru budou relativně malé. K tomu je potřeba především správně zvolit, co bude parametrem.

Vezměme třeba VRCHOLOVÉ POKRYTÍ. Zvolili jsme nejčastější parametrizaci, obvykle se jí říká *parametrizace velikostí řešení*. Mohli bychom ale zvolit jako parametr i nějakou jinou informaci, která s řešením tolik nesouvisí, například maximální stupeň grafu, nebo jako v části 3.3 stromovou šířku. Takové parametry bývají označovány jako *strukturní*.

Pokud ponecháme parametrem k, ale změníme otázku na: „Existuje vrcholové pokrytí velikosti nejvýše $n - k$?“ získáme tzv. *parametrizovaný duál* problému. Ten se ovšem v tomto případě formuluje častěji takto:

NEZÁVISLÁ MNOŽINA
Vstup: Neorientovaný graf $G = (V, E)$ a přirozené číslo k
Parametr: k
Otázka: Existuje v G nezávislá množina, tj. taková, mezi jejímiž vrcholy nevedou žádné hrany, $A \subseteq V$ velikosti alespoň $|A| \geq k$?

Z klasického hlediska jsou oba problémy NP-úplné a tedy stejně těžké. Z hlediska parametrizované složitosti ovšem ne, neboť narozdíl od VRCHOLOVÉHO POKRYTÍ je NEZÁVISLÁ MNOŽINA W[1]-úplná (viz sekci 4) a tedy pravděpodobně není parametrizovaně dostupná.

Zůstaňme ještě u tohoto problému, ale omezme se na rovinné grafy. U nich máme zaručenu nezávislou množinu velikosti alespoň $\frac{n}{4}$, protože jsou 4-barevné [4, 5] a stačí vybrat největší jednobarevnou množinu. Odpovíme-li tedy ANO pro $k \leq \frac{n}{4}$, a pro ostatní vstupy ozkoušíme všech $\binom{n}{k} \leq \binom{4k}{k}$ kandidátů, získáme algoritmus, který ukazuje triviální dostupnost toho problému. Nabízí se tedy možnost *parametrizace nad zaručené hodnoty*, neboli zkoumat níže uvedenou verzi. Bohužel nejsou známy žádné výsledky ohledně podobných parametrizací.

NEZÁVISLÁ MNOŽINA V ROVINNÉM GRAFU
Vstup: Neorientovaný rovinný graf $G = (V, E)$ a přirozené číslo k
Parametr: k
Otázka: Existuje v G nezávislá množina, $A \subseteq V$ velikosti alespoň $|A| \geq \lceil \frac{n}{4} \rceil + k$?

Pro ilustraci širokých možností parametrizace ještě uveďme jeden příklad NP-úplného (jak ukázali v roce 1997 Frances a Litman [26]) problému, který má hned několik přirozených parametrizací. Je tedy možnost ho zkoumat z řady různých pohledů a až při použití rozhodnout, který z případných algoritmů použít.

NEJBLIŽŠÍ ŘETĚZEC
Vstup: k řetězců $s_1, s_2, ..., s_k$ nad abecedou Σ, každý délky L, a nezáporné celé číslo d.
Úkol : Najít řetězec s tak, aby $d_H(s, s_i) \leq d$ pro všechna $i = 1, ..., k$.
Zde $d_H(s, s_i)$ označuje Hammingovu vzdálenost obou řetězců, tedy počet pozic, na kterých se navzájem liší.

Přirozenými parametry jsou například k, d, méně obvyklými třeba $|\Sigma|$ nebo L. V tomto případě nelze opomenout možnost označit více charakteristik, například dvojici k a d, zároveň jako parametr.

Gramm, Niedermeier a Rossmanith [28] ukázali algoritmus založený na prohledávacích stromech (viz sekce 3.2), který řeší NEJBLIŽŠÍ ŘETĚZEC v čase $\mathcal{O}\left(d^d \cdot kd + kL\right)$. To znamená, že je tento problém parametrizovaně dostupný s parametrem d (a všemi kombinacemi parametrů, které obsahují d) a pro pevné d máme lineární algoritmus. Ve stejném článku ukazují i parametrizovanou dostupnost s parametrem k, tentokrát pomocí celočíselného lineárního programování (sekce 3.8).

3 Přehled metod

Následuje přehled postupů, které se obvykle používají, chceme-li o nějakém problému dokázat, že je parametrizovaně dostupný. Jde vlastně o jakési šablony jednotlivých algoritmů. Protože je ovšem velmi obtížné popisovat tato schémata obecně, a i z důvodu lepší srozumitelnosti, budeme raději ukazovat jejich užití na konkrétních problémech.

V klasifikaci budeme především sledovat Niedermeiera [45], protože v Downey a Fellows [18] chybí řada novějších technik, zatímco Sloper [57] uvádí řadu dílčích technik a některým postupům nepřikládá takový význam.

3.1 Redukce na jádro problému

Když zkoumáme nějaký problém, často zjistíme, že pro některé jeho části lze snadno určit optimální řešení a tak je vyřadit z dalších úvah, nebo je nějakým způsobem zjednodušit případně poupravit tak, aby se následnému algoritmu lépe zpracovávala. Mnohdy pak zjistíme, že objem těch složitých částí závisí pouze na zvoleném parametru.

Hlavní myšlenkou redukce na jádro problému je převést instanci problému I na instanci I', která je s ní „ekvivalentní“ a jejíž velikost je omezena nějakou funkcí parametru k. Na instanci I' můžeme pak použít v podstatě jakýkoliv algoritmus, například otestovat všechny možnosti. Získané řešení, pokud existuje, je pak rozšířeno na řešení pro celou instanci I.

Postup je obvykle takový, že nejdříve navrhneme zjednodušující pravidla, podle kterých budeme instanci modifikovat a ukážeme o nich, že zachovávají řešení. Pak zadefinujeme redukovanou instanci a následně dokážeme lemma o hranici, které říká, že buď kladné, nebo záporné redukované instance nemohou být větší, než stanovená mez.

Poprvé tuto techniku použil Buss [9] na uváděném příkladu. Downey a Fellows [18] ji formalizovali a uvádějí jako základní techniku. V poslední době se objevují pravidla založená na speciálních grafových strukturách, zejména tzv. korunkách, viz třeba Chor et al. [12], nebo Sloper [57].

Příklad

Předvedeme si to na problému Vrcholového pokrytí. Myšlenka řešení pochází od Busse [9], takto zjednodušeně se příklad s drobnými odchylkami objevuje v Downey a Fellows [18] a Sloper [57], Niedermeier [45] na stejném problému demonstruje rafinovanější pravidla, využívající třeba již zmiňované korunky.

Pravidlo 1 : Nechť (G, k) je nějaká instance a v vrchol stupně $deg_G v > k$. Pak sestrojíme novou instanci $(G', k') = (G - v, k - 1)$.

Lemma 3.1 (Pravidlo 1 je korektní). *Má-li graf G vrcholové pokrytí velikosti nejvýše k, má i $G - v$ vrcholové pokrytí velikosti nejvýše $k - 1$ a naopak.*

Důkaz. Pokud by v nebylo ve vrcholovém pokrytí, muselo by v něm být celé okolí $N(v)$ bodu v. To ale není možné, protože $|N(v)| > k$ a tedy v je v každém vrcholovém pokrytí G velikosti nejvýše k. Odtud $A \subset V(G')$ je vrcholové pokrytí G', právě tehdy, když $A \cup \{v\}$ je vrcholové pokrytí G. Pravidlo 1 je korektní.

Pravidlo 2 : Nechť (G, k) je nějaká instance, v vrchol stupně 1 a u jeho soused. Pak sestrojíme novou instanci $(G', k') = (G \setminus \{u, v\}, k - 1)$.

Lemma 3.2 (Pravidlo 2 je korektní). *Má-li graf G vrcholové pokrytí velikosti nejvýše k, má i $G \setminus \{u, v\}$ vrcholové pokrytí velikosti nejvýše $k - 1$ a naopak.*

Důkaz. Je-li $A \subset V(G')$ vrcholové pokrytí G', pak $A \cup \{u\}$ je vrcholové pokrytí G. Je-li naopak $A \subset V(G)$ vrcholové pokrytí G, pak musí být $u \in A$ nebo $v \in A$, pak tedy $A \setminus \{u, v\}$ je vrcholové pokrytí G' a $A \setminus \{u, v\} < A \leq k$.

Narozdíl od pravidla 1 pravidlo 2 nepoužívá hodnotu parametru, můžeme ho tedy použít i k redukci původního optimalizačního problému, se kterým je náš rozhodovací spojen a až následně nasadit pravidlo 1 pro konkrétní testovanou hodnotu parametru.

Následující pravidlo je natolik zjevné, že jeho korektnost nebudeme dokazovat.

Pravidlo 0 : Nechť (G, k) je nějaká instance a v izolovaný vrchol. Pak sestrojíme novou instanci $(G', k') = (G - v, k)$.

Definice 3.3. Instanci nazveme *redukovanou*, nelze-li na ni uplatnit pravidlo 0, 1 ani 2.

Lemma 3.4 (O hranici). *Pro kladnou redukovanou instanci (G, k) platí $|V(G)| \leq \frac{k^2}{2} + k$.*

Důkaz. Nechť (G, k) je taková instance a A příslušné vrcholové pokrytí. Všechny hrany z vrcholů ve $V \setminus A$ musí pak vést do A. Dvojic $\{(u, v) | \{u, v\} \in E, u \in (V \setminus A), v \in A\}$ je tedy aspoň $2|V \setminus A|$, protože vrcholy $(V \setminus A)$ mají stupeň aspoň 2 podle pravidla 0 a 2, a nanejvýš $k|A|$, neboť stupeň každého vrcholu z A je menší roven k podle pravidla 1. Je tedy $|V \setminus A| \leq \frac{k^2}{2}$ a odtud plyne tvrzení.

Poznámka 3.5. Bez použití pravidla 2 bychom získali jádro velikosti $k^2 + k$.

Věta 3.6 (Buss [9]). VRCHOLOVÉ POKRYTÍ *lze řešit v čase* $\mathcal{O}\left(\binom{\frac{k^2}{2}+k}{k} k^4 + kn\right)$ *a tedy* $\in \mathcal{FPT}$.

Důkaz. Nejdříve použijeme pravidlo 1 tolikrát, kolikrát to bude možné. Na získanou instanci aplikujeme co nejvíce krát pravidlo 2 a nakonec pravidlo 0. Má-li zbylý graf více než $\frac{k^2}{2}+k$ vrcholů odpovíme NE, jinak otestujeme všechny správně velké kandidáty na pokrytí.

Hlavní výhodou redukce na jádro problému je, že zachovává řešení a tedy příslušná pravidla mohou být použita jako příprava dat pro jakýkoliv algoritmus, nikoliv pouze parametrizovaný.

3.2 Omezený prohledávací strom

Další technikou založenou na velmi jednoduché myšlence je metoda omezeného prohledávacího stromu. Mnoho kombinatorických algoritmů probíhá tak, že nejdříve počítáme všechny možné kandidáty, což často odpovídá vytváření exponenciálně velkého prohledávacího stromu, a u těch pak následně efektivně ověřujeme, zda jsou skutečně řešeními. Z parametrizovaného pohledu je důležité, že mnohdy tento strom lze omezit pouze nějakou funkcí parametru.

Klíčové je, aby prohledávaný prostor nebyl příliš velký, nezkoušíme proto všechna řešení, ale jen ta v jistém smyslu rozumná. Je potřeba, aby počet možných pokračování byl v každém vrcholu stromu malý, nejlépe omezen konstantou, hlouba stromu byla omezena nějakou funkcí parametru a v každém vrcholu stromu (tedy především v listech) se prováděla pouze polynomiální práce.

V literatuře se objevuje řada podobných technik s různými názvy. V našem smyslu je poprvé technika použita u Mehlhorna [44], postupně byla vylepšována především pro Vrcholové pokrytí v článku Balasubramarian et al. [6], knize Downey, Fellows [18] článku Chandran a Grandoni [11] a dalších. Sloper [57] uvádí přesné podmínky pro prohledávací strom, který ukazuje parametrizovanou dostupnost.

Příklad

Použití si ukážeme na příkladu Nezávislé množiny v rovinném grafu. Uváděný příklad pochází od Slopera [57]. Podobně jednoduchý příklad s Vrcholovým pokrytím uvádí Niedermeier [45] a Downey a Fellows [18]. Dobrým příkladem je i podrobně vysvětlený algoritmus, který uvádíme na straně 45.

Věta 3.7 (Sloper [57]). Nezávislou množinu v rovinném grafu *lze řešit v čase* $\mathcal{O}\left(6^k \cdot n\right)$.

Důkaz. Pro maximální nezávislou množinu $I \subset V(G)$ a všechna $v \in V(G)$ platí $N[v] \cap I \neq \emptyset$. Pokud by tomu tak nebylo, můžeme v přidat do I, což je spor s její maximalitou. Spolu s dobře známým faktem, že v rovinném grafu existuje

vrchol stupně nejvýše 5, nám to umožňuje vybrat vždy vrchol v malého stupně a rozhodnout se, zda do vytvářené nezávislé množiny zařadíme vrchol v, nebo některého z jeho nejvýše pěti sousedů. Zvolíme-li vrchol u, odstraníme jeho uzavřené okolí $N[u]$ z grafu. Získáme tak nejvýše 6 menších instancí, každou s parametrem $k' = k - 1$. To nám dává strom velikosti maximálně 6^k, kde v každém uzlu provádíme práci nejhůře $\mathcal{O}(n)$, odtud plyne požadovaný výsledek.

Tato technika se v mnoha případech kombinuje s redukcí na jádro problému (viz 3.1) tak, že se prohledávací strom aplikuje až na jádro, mezi hladinami se instance opět redukují a používá se i lemma o hranici. To se někdy označuje jako *prokládání* obou technik. O této možnosti se zmiňuje Niedermeier a Rosmanith [46] a je dále rozvedena v Niedermeierovi [45].

3.3 Stromový rozklad a dynamické programování

Myšlenka sestavit řešení větších problémů z řešení pro menší části tohoto problému není v programování rozhodně nová. Mnohdy je počet kombinací řešení menších částí, které je třeba zvážit, abychom získali řešení pro naše zadání, omezen jen nějakou funkcí parametru. Výsledkem pak bývá algoritmus, který není polynomiální, ale ukazuje na parametrizovanou dostupnost problému.

První parametrizovaný algoritmus založený na dynamickém programovaní se objevuje v práci Dreyfuse a Wagnera [19], hlubší přehled o technice nabízí například Niedermeier [45].

V teorii grafů se dynamické programování používá nejčastěji v souvislosti se stromovým rozkladem grafu. Začneme jeho definicí:

Definice 3.8. *Stromový rozklad* grafu $G = (V, E)$ je dvojice $(\{X_i | i \in I\}, T)$, kde $X_i \subseteq V$ nazýváme *pytle*, T je strom na vrcholech I a platí:

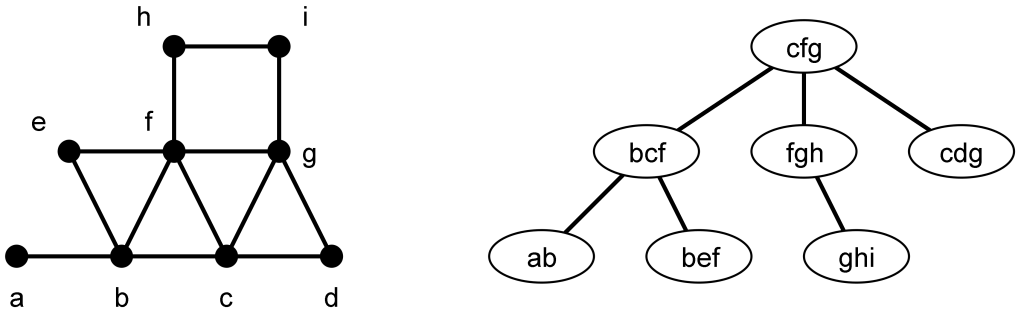
1. $\bigcup_{i \in I} X_i = V$
2. Každá hrana je v nějakém pytli, tj. pro každou hranu $\{u, v\} \in E$ existuje $i \in I$ tak, že $\{u, v\} \subseteq X_i$.
3. Pro každá tři $i, j, k \in I$, pokud j leží na cestě mezi i a k v T pak $X_i \cap X_k \subseteq X_j$, neboli pytle obsahující jeden vrchol grafu G vytváří podstrom T.

Šířka stromového rozkladu je $\max\{|X_i| | i \in I\} - 1$.
Stromová šířka grafu G je nejmenší k takové, že graf G má stromový rozklad šířky k.

Příklad grafu a jeho stromového rozkladu ukazuje obrázek 3.1.

Stromový rozklad zavedli Robertson a Seymour [51], mnoho informací a algoritmů lze nalézt v řadě prací Bodlaendera ([7, 8] a dalších).

Problém určení stromové šířky grafu je NP-úplný, ale je parametrizovaně dostupný. Existuje algoritmus, který rozhodne o existenci stromového rozkladu šířky



Obr. 3.1: Graf a jeho stromový rozklad

k a případně i nějaký takový nalezne v čase $f(k)\cdot n$ (viz Bodlaender [7]), kde však velikost funkce $f(k)$ vylučuje jeho praktickou použitelnost. To lze obvykle obejít tak, že použijeme nějaký aproximační postup pro stromový rozklad, protože navazující řešení nevyužívá optimality rozkladu. Pro účely dynamického programování se od stromového rozkladu často požaduje jeho jednoduchá struktura, tak jak jí zavádí následující definice.

Definice 3.9. Stromový rozklad $(\{X_i | i \in I\}, T)$ se nazývá *pěkný*, pokud splňuje tyto podmínky:

1. Každý uzel stromu T má nejvýše dva syny.
2. Pokud má uzel i dva syny j a k, pak $X_i = X_j = X_k$ a nazývá se *spojovací uzel*.
3. Pokud má uzel i jednoho syna j, pak nastává jedna ze situací
   - $|X_i| = |X_i| + 1$ a $X_j \subset X_i$, pak se i nazývá *zaváděcí uzel*.
   - $|X_i| = |X_i| - 1$ a $X_i \subset X_j$, pak se i nazývá *zapomínací uzel*.

Lemma 3.10. *K zadanému stromovému rozkladu grafu G šířky k o n uzlech lze nalézt pěkný stromový rozklad grafu G o šířce k a $\mathcal{O}(n)$ uzlech v čase $\mathcal{O}(n)$.*

Poznámka 3.11 (Alber, Bodlaender, Fernau, Kloks a Niedermeier [2]). Rovinný graf na n vrcholech má stromovou šířku $\mathcal{O}(\sqrt{n})$.

Užití stromového rozkladu pro parametrizované algoritmy je velmi rozsáhle popsáno v Downey a Fellows [18].

Příklad

Autorem následujícího postupu pro VRCHOLOVÉ POKRYTÍ je Niedermeier [45], význam má především vzhledem k důsledku pro rovinné grafy.

Věta 3.12 (Niedermeier [45]). *Pro graf G a jeho stromový rozklad $(\{X_i | i \in I\}, T)$ lze nalézt optimální vrcholové pokrytí v čase $\mathcal{O}(2^w \cdot w \cdot |I|)$, kde w je šířka stromového rozkladu.*

Důkaz. Ke každému z $|I|$ mnoha pytlů si vytvoříme tabulku velikosti $2^{|X_i|}$, kde řádky odpovídají jednotlivým možnostem, jak zajistit pokrytí podgrafu $G[X_i]$. Budeme je tedy označovat jednotlivými obarveními $C_i : X_i \to \{0, 1\}$ vrcholů X_i barvami 0 a 1, kde 0 znamená, že vrchol do pokrytí nezařadíme a 1 že ano. V příslušném políčku pak budeme ukládat hodnotu $m_i(C_j)$ nejmenšího pokrytí, které obsahuje vrcholy X_i vybrané obarvením C_j. Přesněji tedy

$$\begin{aligned} m_i(C_j) := \min\{|V'| : V' \subseteq V \text{ je vrcholové pokrytí } G \text{ takové,} \\ \text{že } v \in V' \text{ pro všechna } v \in (C_j)^{-1}(1) \text{ a} \\ v \notin V' \text{ pro všechna } v \in (C_j)^{-1}(0)\}. \end{aligned}$$

Samozřejmě ne každé obarvení vede k vrcholovému pokrytí. Ta která k němu nevedou budeme nazývat *neplatná* a jsou to taková C_j, pro která existuje hrana (u, v) v grafu $G[X_i]$, že $C_j(u) = C_j(v) = 0$, ostatní nazýváme *platná*.

Inicializace tabulky Pro každý pytel X_i a každé obarvení C_j nastavíme

$$m_i(C_j) := \begin{cases} |(C_j)^{-1}(1)| & \text{pro } C_i \text{ platné} \\ +\infty & \text{jinak.} \end{cases}$$

Dynamické programování Procházíme rozkladový strom T od listů ke kořeni a porovnáváme příslušné tabulky navzájem. Mějme tedy $i \in I$ uzel stromu T a j jeho syna. Předpokládejme, že $X_i = \{z_1, ..., z_r, v_1, ..., v_s\}$ a $X_j = \{z_1, ..., z_r, u_1, ..., u_t\}$.
Rozšířením obarvení $C : U \to \{0, 1\}, U \subseteq V$ myslíme obarvení $C' : W \to \{0, 1\}$ takové, že $U \subseteq W \subseteq V$ a $C'|_U = C$.
Vylepšíme tabulku pro m_i pomocí tabulky pro m_j. Pro každé obarvení $C : \{z_1, ..., z_r\} \to \{0, 1\}$ a každé jeho rozšíření $C_k : X_i \to \{0, 1\}$ nastavíme

$$\begin{aligned} m_i(C_k) := m_i(C_k) \\ + \min\{m_j(C_l) | C_l : X_j \to \{0, 1\} \text{ je rozšíření } C\} \\ - |(C)^{-1}(1)|. \end{aligned}$$

Jinými slovy $m_i(C_j)$ vzroste o minimální velikost pokrytí podgrafu indukovaného všemi vrcholy obsaženými v podstromu zakořeněném v j, zmenšeném o vrcholy pokrytí obsažené také v X_i, které bychom jinak počítali dvakrát. Pro účely nalezení pokrytí se ještě hodí uložit si, z kterého obarvení syna minimum vyšlo.
Má-li uzel více synů, provedeme postupně vylepšení tabulky pomocí všech synů. To vše opakujeme, dokud nakonec úplně nevylepšíme kořen.

Nalezení vrcholového pokrytí Velikost nejmenšího vrcholového pokrytí je prostě nejmenší záznam v tabulce m_r kořene r. Pokud bychom chtěli i příslušnou množinu, snadno ji zrekonstruujeme použitím záznamů z druhého kroku.

Korektnost algoritmu

- První podmínka Definice 3.8 zajišťuje, že každý vrchol je brán v potaz.
- Druhá že po inicializaci v prvním kroku jsou uvažována pouze skutečná vrcholová pokrytí.
- Třetí podmínka Definice 3.8 zajišťuje konzistenci dynamického programování. Pokud se vrchol v objeví ve dvou pytlích X_i a X_j, je pro spočítané minimální pokrytí zaručeno, že nedostane v odpovídajících obarveních X_i a X_j různou barvu. To je zajištěno nejméně ve společném předkovi k uzlů i a j ve stromě T.

Náročnost algoritmu Porovnání tabulek lze provést v čase $\mathcal{O}(2^w \cdot w)$. Je nutné ho provést pro každou hranu stromu T, což dává požadovanou časovou náročnost. Zmiňme, že ani paměťové nároky nejsou u takovýchto algoritmů zanedbatelné, v tomto případě například $\mathcal{O}(2^w \cdot |I|)$.

Věta 3.13 (Niedermeier [45]). *Má-li graf G vrcholové pokrytí velikosti k, je jeho stromová šířka shora omezena $\mathcal{O}\left(\sqrt{k}\right)$.*

Důsledek 3.14. VRCHOLOVÉ POKRYTÍ ROVINNÉHO GRAFU *lze řešit v čase $2^{\mathcal{O}(\sqrt{k})} \cdot n$, kde k označuje velikost vrcholového pokrytí a n počet vrcholů grafu.*

3.4 Iterativní komprese

Iterativní komprese je poměrně nová technika, která se používá k řešení minimalizačních problémů parametrizovaných velikostí řešení. Základem algoritmu je *kompresní krok*, kdy v $\mathcal{FPT}$ čase k danému řešení velikosti $k+1$ buď nalezneme řešení velikosti k, nebo ukážeme, že žádné takové není. Tento krok postupně aplikujeme na počáteční část grafu, kterou mezi kroky vždy zvětšíme o další vrchol, který přidáme i do řešení.

Jedná se o poměrně novou techniku, kterou představili Reed et al. [49]. S její pomocí se podařilo ukázat parametrizovanou dostupnost několika problémů, jejichž parametrizovaná složitost byla dlouho otevřená.

Příklad

Celý postup si opět ukážeme na VRCHOLOVÉM POKRYTÍ. Ukázku přebíráme z Niedermiera [45], pochází z osobní komunikace autora knihy, autory řešení jsou Jiong Guo a Sebastian Wernicke. Příklad, který uvádí Sloper [57] je o něco málo složitější.

Kompresní krok Mějme graf $G = (V, E)$ a jeho vrcholové pokrytí C velikosti $k + 1$. Uvažme rozdělení C na dvě části C_0 a C_1, kde vrcholy v C_0 jsou ty, které narozdíl od těch v C_1 nechceme do nového pokrytí zařadit. Pro každé takové rozdělení provedeme následující kroky.

1. Nastavíme $D := \emptyset$.
2. Pro každou hranu $\{u, v\} \in E$, pro kterou $\{u, v\} \cap C_1 = \emptyset$ rozlišíme případy:
   (a) Pokud $\{u, v\} \subseteq C_0$, pak neexistuje pokrytí bez vrcholu z C_0 a zvolené rozdělení nevede k řešení.
   (b) Pokud $v \in (V \setminus C)$ přidáme v do D.
   (c) Pokud $u \in (V \setminus C)$ přidáme u do D.

   Případy (b) a (c) nemohou nastat současně, protože C bylo vrcholové pokrytí.
3. Pokud $|D \cup C_1| \leq k$ je $D \cup C_1$ výsledek tohoto kompresního kroku.

Pokud žádné rozdělení nevedlo k výsledku, pak graf G nemá žádné k-vrcholové pokrytí. Kompresní krok trvá $\mathcal{O}\left(2^k \cdot |E|\right)$.

Celý algoritmus Mějme dán graf $G = (V, E), V = \{v_1, ..., v_n\}$ a k. Algoritmus bude postupovat takto:

1. Označme $C := \{v_1, ..., v_k\}$.
2. Pro každé $k + 1 \leq i \leq n$ provedeme kompresní krok na graf $G_i = G[\{v_1, ..., v_i\}]$ s pokrytím $C \cup \{v_i\}$ (velikosti nejvýše $k+1$) a výsledek uložíme zpět do C. Pokud kompresní krok výsledek nevydá, nemá graf G pokrytí velikosti k.
3. Po poslední iteraci kompresního kroku, pokud se podařila, je množina C vrcholové pokrytí grafu $G_n = G$ velikosti nejvýše k a tedy hledané řešení.

Celý algoritmus neprovádí nic jiného, než nejvýše n kompresních kroků a celkem tedy trvá $\mathcal{O}\left(2^k \cdot |E| \cdot n\right)$.

3.5 Hladová lokalizace

Hladová lokalizace je další z poměrně nových metod. Používá se u maximalizačních úloh parametrizovaných velikostí řešení. Nejdříve hladově získáme nějaké (v inkluzi) maximální řešení. To je buď dostatečně velké, nebo jej můžeme použít k omezení prostoru, který je třeba prohledat.

Jia et al. [35] poprvé používal popisovanou techniku v roce 2004. Uvedený příklad přebíráme od Slopera [57], který je jeho autorem. Podle tohoto příkladu lze zařadit tuto techniku jako speciální případ omezeného prohledávacího stromu (viz 3.2), jak to také autor dělá. Příklad uváděný v Niedermeierovi [45] je ale spíše bližší redukci na jádro problému (sekce 3.1), proto jsme tuto techniku stejně jako druhý z autorů zařadili samostatně.

Příklad

Použití techniky si předvedeme na následujícím problému, jehož NP-úplnost ukázali Hell a Kirkpatrick [39]:

Pakování trojúhelníků
Vstup: Graf $G = (V, E)$
Parametr: Přirozené číslo k
Otázka: Obsahuje G sadu k vrcholově disjunktních podgrafů izomorfních K_3?

Věta 3.15 (Sloper [57]). Pakování trojúhelníků *lze řešit užitím hladové lokalizace v čase* $\mathcal{O}\left(\binom{3k}{k} \cdot (2k)! \cdot n^3\right)$.

Důkaz. Nejdříve získáme nějaké maximální pakování trojúhelníků C pomocí jednoduchého hladového algoritmu. Pokud obsahuje alespoň k trojúhelníků, jsme hotovi, v opačném případě se každý trojúhelník z optimálního pakování C_{Opt} musí protínat s C alespoň v jednom vrcholu, jinak bychom ho mohli přidat k C v hladové fázi.

Vyzkoušíme tedy všechny výběry k vrcholů z množiny V_C. Její velikost je nejvýše $3k$. Každý vybraný vrchol zařadíme do jedné z množin $S_1, ..., S_k$, které se pak pokusíme doplnit vždy o dva vrcholy na trojúhelník. To budeme zkoušet opět hladově. Pokud se nám to podaří, máme řešení, pravděpodobně se nám však nepodaří doplnit nějakou množinu S_j. Předpokládáme-li, že množiny lze doplnit, pak jediný důvod, proč se nám to mohlo nepodařit je, že nějaký vrchol přidaný do $S_1, ..., S_{j-1}, S_{j+1}, ..., S_k$ jsme měli přidat do S_j. Máme tedy nejvýše $2k$ možností, který vrchol přidat do S_j. Takto jsme tedy přidali jeden vrchol do našich částečných množin a můžeme pokračovat dále.

Takto nám vznikne $\binom{3k}{k}$ prohledávacích stromů, které mají $2k$ hladin a na i-té hladině od kořene mají nejvýše $2k-i$ možností pokračování. To dává požadovanou složitost.

3.6 Teorie grafových minorů

Následující tři sekce ukazují techniky, kterými sice lze ukázat o nějakém problému, že je parametrizovaně dostupný, ale příslušné algoritmy nelze v praxi použít. I přesto mohou velmi dobře sloužit k určení složitosti mnoha problémů, u kterých lze pak očekávat nalezení nějakého praktického algoritmu.

Teorii grafových minorů vybudovali Robertson a Seymour v celé řadě prací ([50, 51, 52, 53] a mnoho dalších). Použití v prostředí parametrizované složitosti je rozsáhle popsáno v Downey a Fellows [18].

Abychom mohli použít teorii grafových minorů k prokazování parametrické dostupnosti, potřebujeme tyto dva zásadní výsledky:

Věta 3.16 (Robertson a Seymour [52]). *Je-li $\mathcal{F}$ množina konečných grafů uzavřených na minory, pak k ní existuje konečná obstrukční množina grafů $O_{\mathcal{F}} = \{H_1, ..., H_t\}$ tak, že*

$$(G \notin \mathcal{F}) \Leftrightarrow (\exists 1 \leq i \leq t : H_i \text{ je minorem } G).$$

Věta 3.17 (Robertson a Seymour [53]). *Problém*

Test minoru
Vstup: *Graf G na n vrcholech*
Parametr: *Graf H na k vrcholech*
Otázka: *Je H minorem G?*

je řešitelný v čase $f(H) \cdot n^3$ a je tedy parametrizovaně dostupný.

Tato věta má pro parametrizovanou složitost mnoha problémů zásadní význam, předvedeme si její použití. Příklady užití těchto vět jsou si velmi podobné, ten námi uváděný sleduje Slopera [57].

Příklad

Ukážeme, že problém

Regulační vrcholová množina
Vstup: Graf $G = (V, E)$
Parametr: Přirozené číslo k
Otázka: Existuje množina $V' \subseteq V$ vrcholů grafu G, velikosti $|V'| \leq k$, tak, že po jejím odstranění z grafu zbyde les, tj. $G[V \setminus V']$ nebude mít žádné kružnice?

je parametrizovaně dostupný.

Pozorování 3.18. *Je-li* (G, k) *kladná instance* Regulační vrcholové množiny *a* H *je minorem* G*, pak* (H, k) *je také kladná instance toho problému.*

Důkaz. Vlastnost se snadno ověří pro jednotlivé minorové operace.

Množina $\mathcal{F}_k$ všech grafů s regulační množinou velikosti nejvýše k je uzavřená na minory. Existuje k ní proto podle věty 3.16 konečná obstrukční množina $O_{\mathcal{F}_k} = \{H_1, ..., H_t\}$. K zjištění, zda má zadaný graf G regulační množinu velikosti nejvýše k stačí ověřit, že žádný z grafů $H_1, ..., H_t$ není jeho minorem. To lze podle věty 3.17 v čase $b(k) \cdot n^3$, kde $b(k)$ nezávisí na G, neboť vlastnosti množiny $O_{\mathcal{F}_k}$ závisí také pouze na k.

3.7 Barevné kódování

Další ze spíše teoretických metod se využívá při hledání podgrafu, který je izomorfní nějakému grafu. Původnímu grafu přiřadíme barvy pomocí hashovacích funkcí tak, že námi hledaný podgraf nějak s barvami reaguje, nejčastěji obsahuje každou z nich právě jednou. Abychom mohli metodu použít potřebujeme následující vlastnost hashovacích funkcí.

Definice 3.19. Systém $\mathcal{H}$ funkcí z $\{1, ..., n\}$ na $\{1, ..., k\}$ se nazývá *k-perfektní systém hashovacích funkcí*, pokud pro každou $S \subset \{1, ..., n\}, |S| = k$ existuje $h \in H$ tak, že $h|_S$ je bijekce.

Věta 3.20 (Alon,Yuster a Zwick [3]). *Je možné sestrojit* k*-perfektní systém* $\mathcal{H}$ *hashovacích funkcí z* $\{1, ..., n\}$ *na* $\{1, ..., k\}$ *o velikosti* $|\mathcal{H}| = 2^{\mathcal{O}(k)} \cdot \log n$*. Hodnotu takové funkce lze spočítat v lineárním čase.*

Bohužel skrytá konstanta dělá tento výsledek poněkud méně praktickým. Techniku barevného kódování vyvinuli Alon et al. [3] v roce 1995.

Příklad

Použití si předvedeme na příkladu z práce Slopera [57]:

Kružnice
Vstup: Graf G
Parametr: Přirozené číslo k
Otázka: Obsahuje G kružnici délky k?

Problém je NP-úplný, neboť jeho speciálním případem pro $k = n$ je dobře známý NP-úplný problém (viz Garey a Johnson [27]) Hamiltonovská kružnice.

Mějme graf $G = (V, E)$ a k-perfektní systém $\mathcal{F}$ hashovacích funkcí, který máme zaručen větou 3.20. Budeme postupovat tak, že zvolíme nějakou hashovací funkci $f \in \mathcal{F}$ a nějaké pořadí $c_1, c_2, ..., c_k$ barev $\{1, ..., k\}$. Z grafu G uděláme orientovaný tak, že hrany $\{u, v\} \in E$, které spojují sousední barvy, tj. takové, že $\exists 1 \leq i \leq k : f(u) = c_i \wedge f(v) = c_{(i+1) \mod k}$ nahradíme orientovanou hranou z u do v. Ostatní hrany odstraníme. Z každého vrcholu v pak provedeme prohledávání do šířky. Pokud nalezneme kružnici C z v do v, délky nejvýše k pak má jistě délku právě k, neboť v našem orientovaném grafu kratší nejsou, a tedy máme řešení. Tento postup opakujeme pro každou funkci f a každé pořadí barev. Obsahuje-li G cyklus, jistě ho musíme vzhledem k vlastnostem k-perfektního systému hashovacích funkcí v některé iteraci nalézt.

Popsaný algoritmus korektně rozhodne problém kružnice v čase $2^{\mathcal{O}(k)} \cdot k! \cdot \mathcal{O}((n + m) \cdot n \cdot \log n)$.

3.8 Celočíselné lineární programování

Jako poslední techniku, která se používá k prokázání parametrické dostupnosti, si ukážeme jeden výsledek z celočíselného lineárního programování. Klíčové je řešení tohoto problému:

ŘEŠITELNOST CELOČÍSELNÉHO LINEÁRNÍHO PROGRAMOVÁNÍ
Vstup: Systém lineárních nerovnic s celočíselnými koeficienty v proměnných $x_1, ...x_p$
Parametr: Přirozené číslo p udávající počet proměnných
Otázka: Existuje takové přiřazení celých čísel proměnným, že jsou všechny nerovnice splněny?

Věta 3.21 (Lenstra [43], Kannan [37]). ŘEŠITELNOST CELOČÍSELNÉHO LINEÁRNÍHO PROGRAMOVÁNÍ *lze rozhodnout použitím* $\mathcal{O}\left(p^{9p/2}L\right)$ *aritmetických operací s celými čísly o* $\mathcal{O}\left(p^{2p}L\right)$ *bitech, kde* L *je počet bitů celého vstupu.*

Použití této techniky je natolik přímočaré, že ji uvedeme bez příkladu. Je pouze potřeba zajistit, aby počet proměnných byl omezen nějakou funkcí parametru. Je-li tato funkce rozumná, může být dokonce technika pro velmi malé hodnoty parametru prakticky použitelná.

Uvedená věta je výsledkem Hendrika W. Lenstry [43], dále vylepšená Ravi Kannanem [37]. Použití lze nalézt v Niedermeierovi [45].

4 Parametrizovaná nedostupnost

Parametrizovaná složitost obsahuje také teorii parametrizované nedostupnosti. Je to způsob jak o nějakém problému ukázat, že pravděpodobně není parametrizovaně dostupný. Postup je obdobný jako u NP-úplných problémů. Definujeme redukci mezi problémy, pak ukážeme nějakou velkou třídu problémů, které jsou všechny stejně obtížné, neboť je lze mezi sebou převést pomocí této redukce. Ukážeme-li pak, že je lze převést i na nějaký náš problém, znamená to, že pokud bychom nalezli efektivní řešení našeho problému, umíme rychle řešit všechny problémy v dané třídě.

Teorii parametrizované nedostupnosti rozvinuli a ve svých článcích popsali Downey a Fellows [15, 16, 17]. Jejich kniha [18] je v tomto směru dosud nejrozsáhlejší. Náš krátký úvod částečně čerpá také z Niedermeiera [45], charakterizace pomocí Turingových strojů je podle Cesatiho [10].

Začneme s definicí redukce mezi dvěma problémy.

Definice 4.1. Nechť $P, P' \subseteq \Sigma^* \times \mathbb{N}$ jsou dva parametrizované jazyky. Řekneme, že se P *redukuje* na P' *pomocí standardní parametrizované redukce*, pokud existují funkce $f, g : \mathbb{N} \to \mathbb{N}$, funkce $r : \Sigma^* \times \mathbb{N} \to \Sigma^*$ a konstanta c takové, že pro každé $(x, k) \in \Sigma^* \times \mathbb{N}$

1. funkce $r : \Sigma^* \times \mathbb{N} \to \Sigma^*$ lze vypočíst v čase $f(k) \cdot |(x, k)|^c$ a
2. $(x, k) \in P \Leftrightarrow (r(x, k), g(k)) \in P'$.

Poznámka 4.2. Pokud bychom chtěli definici zobecnit na problémy v $\Sigma^* \times \Sigma^*$, je klíčové, aby se každá vrstva (množina všech instancí se stejným parametrem) problému P převedla pouze na konečně mnoho vrstev problému P'.

Tříd parametrizované nedostupnosti je více, jejich základem bude splnitelnost formulí ve speciálním tvaru, který zavádí následující definice.

Definice 4.3. Logická formule je 1-*normalizovaná*, pokud je v konjunktivní normální formě a každá klausule má velikost nejvýše 2, neboli pokud je to tzv. 2-CNF formule.
Nechť $t \geq 2$, pak formule je t-*normalizovaná*, pokud je ve tvaru „konjunkce disjunkcí konjunkcí…“ literálů, kde se konjunkce s disjunkcemi střídají $(t-1)$-krát. Literál je buď proměnná, nebo její negace.
Formule je *antimonotónní*, pokud všechny její literály jsou tvaru negace proměnné a *monotónní*, pokud naopak neobsahuje žádnou negaci.

VÁŽENÁ SPLNITELNOST t-NORMALIZOVANÝCH FORMULÍ
Vstup: t-normalizovaná formule $\mathcal{F}$ s proměnnými $x_1, ..., x_n$
Parametr: Přirozené číslo k
Otázka: Existuje ohodnocení proměnných $x_1, ..., x_n$ tak, že formule $\mathcal{F}$ je tímto ohodnocením splněna a právě k proměnných je nastaveno na „pravda“?

VÁŽENÁ SPLNITELNOST je zobecněním předchozího problému na libovolné formule a VÁŽENÁ SPLNITELNOST OBVODŮ dokonce na libovolné logické obvody.

Definice 4.4. Nechť P je parametrizovaný problém. Řekneme, že problém P je

1. *ve třídě W[t]*, pokud lze P redukovat na VÁŽENOU SPLNITELNOST t-NORMALIZOVANÝCH FORMULÍ (pomocí parametrizované redukce).
2. *W[t]-těžký* a tedy *parametrizovaně nedostupný*, pokud lze VÁŽENOU SPLNITELNOST t-NORMALIZOVANÝCH FORMULÍ redukovat na P.
3. *W[t]-úplný*, pokud je zároveň ve W[t] a W[t]-těžký.

Definice 4.5. *W[Sat]* je třída všech problémů, které lze převést na VÁŽENOU SPLNITELNOST a *W[P]* všech problémů redukovatelných na VÁŽENOU SPLNITELNOST OBVODŮ.
Parametrizovaný problém P patří do *třídy XP*, pokud pro všechna $(x, k) \in \Sigma^* \times \Sigma^*$ lze rozhodnout zda $(x, k) \in P$ v čase $f(k) \cdot |x|^{g(k)}$, kde f a g jsou vyčíslitelné funkce závislé pouze na k.

Nyní si můžeme shrnout celou hierarchii parametrizované složitosti:

$$\mathcal{FPT} \subseteq W[1] \subseteq W[2] \subseteq ... \subseteq W[Sat] \subseteq W[P] \subseteq XP$$

Stejně jako v klasické složitosti se věří, že všechny inkluze jsou ostré. Přesto není známo, že by bylo možné řešit úlohy například z W[1] nějak rychleji než ostatní úlohy z W[2]. Přestože tříd parametrizované nedostupnosti je celá řada, většina běžných problémů je ve třídách W[1] nebo W[2]. Pro lepší představu o těchto třídách si ukážeme několik základních výsledků.

Věta 4.6 (Downey a Fellows [18]). *Nechť t je přirozené číslo, pak* VÁŽENÁ SPLNITELNOST ANTIMONOTÓNNÍCH t-NORMALIZOVANÝCH FORMULÍ *je W[t]-úplná pro t liché a ve W[$t-1$] pro t sudé. Naopak* VÁŽENÁ SPLNITELNOST MONOTÓNNÍCH t-NORMALIZOVANÝCH FORMULÍ *je ve W[$t-1$] pro t liché a W[t]-úplná pro t sudé.*

Důsledek 4.7. NEZÁVISLÁ MNOŽINA *je W[1]-úplná.*

Důkaz. Nezávislé množiny v grafu $G = (V, E)$ si jednoznačně odpovídají s množinami proměnných nastavených na „pravda“ ve splňujícím ohodnocení formule

$$\bigwedge_{\{i,j\} \in E} (\neg x_i \vee \neg x_j).$$

Naopak k jakékoliv antimonotónní formuli není problém nalézt příslušný graf.

Následující problém je základem pro celou řadu důkazů W[2]-úplnosti především v teorii grafů.

Dominující množina
Vstup: Graf $G = (V, E)$
Parametr: Přirozené číslo k
Otázka: Nalézt množinu vrcholů $S \subseteq V$ tak, že každý vrchol $v \in V$ je v S, nebo tam má nějakého souseda.

Věta 4.8 (Downey a Fellows [18]). Dominující množina *je W[2]-úplná.*

Pro lepší orientaci v obou složitostních třídách se hodí ještě následující charakterizace pomocí Turingových strojů, více se o tomto přístupu rozepisuje Cesati [10].

Krátký výpočet nedeterministického Turingova stroje
Vstup: (Jednopáskový) nedeterministický Turingův stroj M
Parametr: Přirozené číslo k
Otázka: Přijme M prázdný vstup po nejvýše k krocích?

Věta 4.9 (Downey a Fellows [18]). Krátký výpočet nedeterministického Turingova stroje *je W[1]-úplný.*

Povolíme-li, aby Turingův stroj přistupoval v každém kroku na více pásek, stane se problém ještě těžší:

Krátký výpočet vícepáskového vícehlavého nedeterministického Turingova stroje
Vstup: Vícepáskový vícehlavý nedeterministický Turingův stroj M a vstup x
Parametr: Přirozené číslo k
Otázka: Přijme M vstup x po nejvýše k krocích?

Věta 4.10 (Cesati [10]). Krátký výpočet vícepáskového vícehlavého nedeterministického Turingova stroje *je W[2]-úplný.*

Kapitola 4

Aplikace na Seidelovo přepnutí

Seidelovo přepnutí je jedna z nejjednodušších přepínacích operací na grafech. Přesto dosud není mnoho známo o složitosti otázek typu: „Je možné přepnout graf tak, aby měl vlastnost P?“. V této kapitole nejdříve podáme přehled takových výsledků. Ukazuje se, že spolu složitost rozpoznání P a příslušné úlohy pro přepnutí nemusí souviset. Dále v této kapitole ukážeme parametrizovanou dostupnost několika problémů, pro které úloha přepnutí na graf s vlastností P je NP-úplná, nebo její složitost není známa, ačkoli rozpoznání vlastnosti P je polynomiální.

1 Definice

Definice 1.1. Mějme graf $G = (V, E)$ a nějakou podmnožinu $A \subseteq V$ jeho vrcholů. *Seidelovým přepnutím množiny* A v grafu G je graf $S(G, A) = (V, E')$, kde $E' = E \triangle \{\{u, v\} | u \in A, v \in V \setminus A\}$. Jedná se o graf na stejné množině vrcholů jako G, kde hrany vedoucí z množiny A ven se změní na nehrany a naopak.
Je-li speciálně $A = \{v\}$, mluvíme o *Seidelově přepnutí vrcholu* v.

Protože se v této práci hovoří jedině o přepnutí Seidelově, budeme tento přídomek někdy vynechávat.

V souvislosti se symetrickými strukturami jako přímkami o stejném úhlu, silně regulárními grafy a tzv. dvojgrafy zavedl myšlenku Seidelova přepnutí dánský matematik J. J. Seidel [54, 55, 56]. Následující základní poznatky lze spolu se spoustou dalších nalézt v práci Hage [29].

Pozorování 1.2. *Nechť G je graf a $A, B \subseteq V$, pak*
(i) $S(G, A)[A] = G[A]$,
(ii) $S(G, A) = S(G, V \setminus A)$,
(iii) $S(G, \emptyset) = S(G, V) = G$,

(iv) $S(S(G,A),B) = S(S(G,B),A) = S(G, A \triangle B)$,
(v) $S(S(G,A),A) = G$,
(vi) $S(\overline{G},A) = \overline{S(G,A)}$.

Definice 1.3. Řekneme, že grafy G a H jsou *ekvivalentní v přepnutí* (značíme $G \sim H$, pokud existuje $A \subseteq V_G$ taková, že $S(G,A)$ je izomorfní H.
Množinu $[G] = \{S(G,A) : A \subseteq V_G\}$ nazveme třídou přepnutí grafu G.

Lemma 1.4. *Nechť G je graf, v jeho vrchol a $A \subseteq V \setminus \{v\}$, pak existuje právě jeden graf $H \in [G]$ takový, že $N_H(v) = A$.*

Důkaz. Graf $S(G, N_G(v) \triangle A)$ má požadovanou vlastnost.

Naopak předpokládejme, že vrchol v má v grafu $S(G,B)$ okolí A. Protože $S(G,B) = S(G, V \setminus B)$ můžeme předpokládat $v \notin B$. Pak $u \in B$ právě pro ta u, pro která $u \in N_G(v) \triangle A$ a tedy nutně $B = N_G(v) \triangle A$.

Důsledek 1.5. *Nechť G je graf na n vrcholech, pak $|[G]| = 2^{n-1}$.*

Důkaz. Existuje 2^n podmnožin V, ale vždy množina A a $V \setminus A$ dávají při přepnutí stejný graf. Třída tedy obsahuje nejvýše 2^{n-1} grafů. Naopak zvolíme $v \in V$ libovolně, pak pro každou podmnožinu $A \subseteq V \setminus \{v\}$ z 2^{n-1} možných existuje graf v $[G]$ takový, že A je okolí v. Tyto musí být jistě různé.

2 Známé výsledky

V této sekci si shrneme některé známé výsledky především klasické složitosti úloh, které se ptají, zda lze zadaný graf přepnout na graf s nějakou pevně zvolenou vlastností. Je zajímavé, že složitost takovéto úlohy nemusí vůbec souviset se složitostí rozpoznání zvolené vlastnosti. Jsou známy vlastnosti, jejichž rozpoznání je v P a otázka, zda zadaný graf lze přepnout na graf s touto vlastností, je NP-úplná (viz věta 2.3) i naopak (věta 2.9).

Začněme s výsledky, které jsou nejbližší většině námi zkoumaných problémů, tedy otázkami spojenými se stupni vrcholů.

Věta 2.1 (Kratochvíl [40]). *Nechť k je pevné a $\mathcal{A}$ je třída grafů uzavřená na izomorfizmy taková, že v každém grafu z $\mathcal{A}$ je nějaký vrchol stupně nejvýše k. Pokud lze třídu $\mathcal{A}$ rozpoznat v polynomiálním čase, pak je také možné v polynomiálním čase rozhodnout, zda graf na vstupu je v přepnutí ekvivalentní nějakému grafu z $\mathcal{A}$. Přesněji tuto otázku lze rozhodnout v čase $\mathcal{O}\left(n^{k+3} \cdot p(n)\right)$, kde $p(n)$ je časová složitost rozpoznání třídy $\mathcal{A}$ v nejhorším případě.*

Důkaz věty ukazuje přímo postup, jak takové přepnutí najít. Vyzkoušíme všech n kandidátů na vrchol malého stupně a pro takto zvolený vrchol vždy všech $\mathcal{O}\left(n^k\right)$ možností jak vytvořit jeho okolí. Takové okolí podle lemmatu 1.4 jednoznačně určuje graf $H \in [G]$, pro který pak stačí zjistit, zda $H \in \mathcal{A}$.

Důsledek 2.2 (Kratochvíl [40]). *Lze rozhodnout v polynomiálním čase, zda je graf v přepnutí ekvivalentní stromu, lesu, rovinnému grafu, vnějškově rovinnému grafu, grafu s omezeným rodem, omezenou stromovou šířkou, nebo k regulárnímu pro pevné k.*

Pokud je pro přepnutí na k-regulární graf k součástí vstupu, je otázka výrazně složitější:

Věta 2.3 (Kratochvíl [40]). *Rozhodnout, zda graf na vstupu je v přepnutí ekvivalentní regulárnímu grafu, je NP-úplný problém.*

Parametrizovanou dostupnost tohoto problému ukážeme v sekci 4.

Dalším výsledkem, který se zabývá stupni vrcholů se týká eulerovských grafů. Řekneme, že graf je *sudý* (*lichý*), jsou-li stupně všech jeho vrcholů sudé (liché). Graf nazveme *eulerovský*, pokud obsahuje uzavřený tah procházející právě jednou každou hranu, což je ekvivalentní s tím, že je sudý a souvislý.

Věta 2.4 (Seidel [54]). *Nechť G je graf na lichém počtu vrcholů, pak $[G]$ obsahuje jediný sudý graf.*

Tento jediný graf lze získat tak, že přepneme množinu všech vrcholů, které mají lichý stupeň.

Věta 2.5 (Ehrenfeucht, Hage, Harju, Rozenberg [23]). *Nechť G je graf na sudém počtu vrcholů. Pak buď $[G]$ neobsahuje žádný sudý ani lichý graf, nebo je přesně polovina grafů z $[G]$ sudá a přesně polovina lichá.*

Věta 2.6 (Hage, Harju, Welzl [31]). *Nechť G je graf na sudém počtu vrcholů a nechť $[G]$ obsahuje sudý graf. Pak $[G]$ obsahuje eulerovský graf, pokud neobsahuje úplný graf.*

Důsledek 2.7 (Hage, Harju, Welzl [31]). *V polynomiálním čase lze rozhodnout, jestli je možné zadaný graf přepnout na sudý, lichý, nebo eulerovský graf.*

Důkaz. Pokud je n liché, je odpověď na první otázku vždy ANO, na druhou vždy NE a pro třetí stačí otestovat zda je jediný sudý graf souvislý.

Pro n sudé stačí ověřit zda je zadaný graf sudý nebo lichý. Pokud ano, je odpověď na první dvě otázky kladná a je třeba užitím lemmatu 1.4 ověřit, zda třída přepnutí obsahuje úplný graf, abychom získali třetí odpověď. Pokud graf není ani sudý ani lichý, jsou všechny tři odpovědi záporné.

Velmi přípuznou je otázka přepnutí na hamiltonovský a pancyklický graf. Graf je *hamiltonovský*, pokud obsahuje kružnici na všech vrcholech a *pancyklický*, pokud obsahuje kružnice délky i pro všechna $3 \leq i \leq n$. Pro obě vlastnosti existuje jednoduchá charakterizace, ze které přímo plyne polynomiální algoritmus.

Věta 2.8 (Kratochvíl, Nešetřil, Zýka [42]). *Každý graf lze přepnout na graf obsahující hamiltonovskou cestu, tj. cestu délky n.*

Věta 2.9 (Kratochvíl, Nešetřil, Zýka [42]). *Graf na aspoň třech vrcholech lze přepnout na hamiltonovský právě tehdy, když není úplný bipartitní na lichém počtu vrcholů.*

Věta 2.10 (Ehrenfeucht, Hage, Harju, Rozenberg [22]). *Graf lze přepnout na pancyklický právě tehdy, když není úplný bipartitní.*

Další vlastností, která se zabývá grafem jako celkem, je barevnost. Máme-li dán graf $G = (V, E)$, nazveme funkci $c : V \to \{1, 2, ..., k\}$ *řádné k-obarvení grafu*, pokud $\forall\{u, v\} \in E : c(u) \neq c(v)$. Graf je *k-barevný*, pokud má nějaké řádné k-obarvení. Nejmenší k takové, že G je k-barevný nazveme *barevnost* grafu G a píšeme $\chi(G)$

Trojbarevnost grafu je dobře známý NP-úplný problém a to i pro grafy maximálního stupně čtyři [27].

Věta 2.11 (Ehrenfeucht, Hage, Harju, Rozenberg [23]). *Pro zadaný graf G, je rozhodnout, zda* $[G]$ *obsahuje tříbarevný graf, NP-úplný problém.*

Věta 2.12 (Kratochvíl [40]). *Rozhodnout, zda graf na vstupu je v přepnutí ekvivalentní nějakému tříbarevnému s maximálním stupněm 4, je NP-úplný problém.*

Naopak to, zda je graf bipartitní (což je totéž jako dvoubarevný), lze rozlišit v polynomiálním čase.

Věta 2.13 (Hage, Harju, Welzl [31]). *Rozhodnout, zda třída přepnutí obsahuje bipartitní graf nebo ne, lze v čase kubickém k počtu vrcholů zadaného grafu.*

Algoritmus pracuje převodem na 2-SAT. Hage v práci [29] ukazuje, že vedlejším produktem tohoto algoritmu je 4-obarvení vstupního grafu, takže každý graf, který má bipartitní přepnutí, je čtyřbarevný.

Je snadné nahlédnout, že třída přepnutí úplného bipartitního grafu obsahuje pouze úplné bipartitní grafy a jako speciální případ diskrétní graf. To podle Hage et al. [31] nastane, pokud graf neobsahuje indukovaný K_3 ani $K_2 + K_1$.

Následující výsledek ukazuje celou řadu NP-úplných problémů. Budeme potřebovat jednoduchou definici:

Definice 2.14. Nechť P je vlastnost uzavřená na izomorfizmy. Řekneme, že P je

- *netriviální*, pokud existuje graf, který vlastnost P nemá, a libovolně velké grafy, které ji mají;
- *netriviální v přepnutí*, pokud je netriviální a existuje třída přepnutí, která neobsahuje graf s vlastností P;
- *dědičná*, jestliže všechny indukované podgrafy G mají P, kdykoliv ji má G.

Věta 2.15 (Ehrenfeucht, Hage, Harju, Rozenberg [23]). *Nechť P je dědičná vlastnost netriviální v přepnutí. Pak následující problém pro instance $(G, k), k \leq |V_G|$ je NP-těžký: obsahuje třída $[G]$ graf H s indukovaným podgrafem $H[A]$ o vlastnosti P a velikosti $|A| \geq k$? Je-li rozpoznání vlastnosti P ve třídě NP, pak je tento problém NP-úplný.*

Příklad 2.16. „Být úplný graf“ je dědičná vlastnost netriviální v přepnutí.

Důsledek 2.17. *Rozhodnout, zda graf je v přepnutí ekvivalentní nějakému grafu obsahujícímu k-kliku je NP-úplný problém, je-li k součástí vstupu.*

K tomuto důsledku se přímo váže další skupina výsledků, která se zabývá problémy, souhrnně označovanými jako vnořování. Řekneme, že graf H *lze vnořit do třídy* $[G]$, pokud existuje $A \subseteq V_G$, že $S(G, A)$ obsahuje H.

Věta 2.18 (Ehrenfeucht, Hage, Harju, Rozenberg [23]). *Rozhodnout, zda lze H vnořit do třídy $[G]$ je NP-úplný problém, jsou-li G i H vstupem.*

Věta 2.19 (Ehrenfeucht, Hage, Harju, Rozenberg [23]). *Pro instance (G, H, k), kde $V_G = V_H$ a $3 \leq k \leq n - 1$, je problém, zda existuje $X \subseteq V_H, |X| \geq k$ taková, že $H[X] \in [G[X]]$, NP-úplný .*

Pro srovnání je třeba uvést, že:

Lemma 2.20 (Ondráčková [47]). *Pro pevně zvolené H lze v polynomiálním čase rozhodnout, zda k zadanému grafu G existuje graf s ním ekvivalentní v přepnutí, který obsahuje H.*

Důkaz. Stačí otestovat všechny indukované podgrafy grafu G na $|V_H|$ vrcholech, zda je lze přepnout na graf H. To lze udělat v polynomiálním čase pomocí lemmatu 1.4.

Další související výsledek uvádí Ondráčková [47], když zkoumá zda lze v třídě přepnutí najít graf obsahující kliku velikosti asymptoticky srovnatelné s velikostí grafu.

Věta 2.21 (Ondráčková [47]). *Pro každé pevné reálné číslo $c \in (0, 1)$ je NP-úplné rozhodnout, zda lze zadaný graf přepnout na graf obsahující kliku velikosti alespoň cn, kde n je velikost grafu na vstupu.*

Předcházející problémy se ptaly, zda existuje graf ekvivalentní v přepnutí ke vstupnímu grafu takový, že obsahuje hledaný podgraf. Přirozeně se nabízí zcela opačný problém a sice, zda lze zadaný graf přepnout na graf, který nějaký pevně zvolený graf neobsahuje jako indukovaný podgraf. Říkáme, že je prostý zakázaného (indukovaného) podgrafu. Takovými úlohami se zabývá řada článků, ukážeme si příslušné výsledky.

Začněme jednoduchým pozorováním, které celou klasifikaci výrazně zjednoduší.

Pozorování 2.22. *Pokud algoritmus $\mathcal{A}$ rozhodne pro vstup G, zda je graf G ekvivalentní v přepnutí nějakému grafu, který je H-prostý, pak pro vstup $\overline{G}$ algoritmus $\mathcal{A}$ rozhodne, zda je graf G ekvivalentní v přepnutí nějakému grafu, který je $\overline{H}$-prostý.*

Je známa kompletní klasifikace pro H na nejvýše třech vrcholech a některé další grafy:

Pro K_2 je odpověď jednoduchá, G musí být úplný bipartitní.

Kratochvíl et al. [42] uvádějí algoritmus pro přepnutí na $K_{1,2}$-prostý graf (tedy i pro $K_2 + K_1$-prostý), kdy nejdříve zadaný graf přepne tak, aby se některý z vrcholů stal izolovaný, ten pak z grafu odstraní a spolu s ním každý vrchol, pro který existuje sousední vrchol se stejným uzavřeným okolím. Graf je ekvivalentní $K_{1,2}$-prostému právě tehdy, když upravený graf je hvězda $K_{1,m}$ nebo diskrétní graf. Tento algoritmus pracuje v čase $\mathcal{O}(n^3)$.

Hayward [32, 33] a Hage et al. [31] nezávisle objevili dva různé algoritmy na rozpoznávání grafů, které lze přepnout na graf bez trojúhelníků. Oba pracují v čase $\mathcal{O}(n^3)$ a používají převod na 2-SAT.

Tím jsou vyřešeny všechny zakázané grafy H na nejvýše třech vrcholech.

Věta 2.23 (Ondráčková [47]). *K zadanému grafu G lze v polynomiálním čase nalézt množinu $A \subset V$ takovou, že $S(G, A)$ je prostý $K_{1,3}$, nebo ukázat, že taková množina neexistuje.*

Algortimus pracuje v čase $\mathcal{O}(n^6)$ a používá redukci na soustavu lineárních rovnic nad GF(2).

Je-li vlastnost P dědičná, a H je indukovaný podgraf grafu G takový, že nejde přepnout na graf s vlastností P, pak ani G nelze přepnout na graf s vlastností P. Můžeme tedy pro danou vlastnost vytvořit množinu minimálních zakázaných indukovaných podgrafů, která nám umožní snadněji rozpoznat grafy, které lze přepnout na grafy s vlastností P. V ideálním případě bude tato množina konečná.

Ukázkou takového případu je charakterizace grafů, které lze přepnout na diskrétní graf, ze strany 31. Ondráčková ve své práci [47] uvádí charakterizaci pro třídu přepnutí obsahující graf bez $K_{1,2}$ pomocí deseti zakázaných indukovaných podgrafů z množiny $[C_5] \cup [C_4 + K_1]$. Předvedeme si ještě některé další.

Maximální k takové, že graf G obsahuje k-kliku nazveme *klikovost* grafu G a píšeme $\omega(G)$. Graf nazveme *perfektní*, pokud každý jeho indukovaný podgraf má stejnou klikovost a barevnost. Hertz [34] poskytuje charakterizaci hned několika vlastností zároveň, která dává polynomiální algoritmus pro otázku přepnutí na graf bez P_4:

Věta 2.24 (Hertz [34]). *Pro třídu přepnutí $\mathcal{S}$ jsou následující tvrzení ekvivalentní:*

(i) $\mathcal{S}$ obsahuje pouze perfektní grafy.

(ii) Každý graf z $\mathcal{S}$ je H-prostý pro všechna $H \in [C_5]$.

(iii) Nějaký graf z $\mathcal{S}$ je P_4-prostý.

Hage a Harju [30] podávají charakterizaci grafů, které lze přepnout na les. Takové grafy nesmí kromě C_n pro $n \geq 7$ jako indukovaný podgraf obsahovat dalších 905 grafů z 27 tříd přepnutí.

Definice 2.25. Nechť máme dán graf G, pak množina $C \subseteq V$ je *t-perfektní kód* v G, pokud pro každé $v \in V$ existuje právě jedno $u \in C$ takové, že $d(u, v) \leq t$.

Věta 2.26 (Kratochvíl [41]). *Graf G neobsahuje žádný z grafů $[C_5] \cup [I_4]$ jako indukovaný podgraf, právě tehdy, když všechny grafy z $[G]$ obsahují 1-perfektní kód.*

3 Přepnutí na graf se všemi vrcholy stupně $\leq k$

Pro mnoho problémů spojených se Seidelovým přepnutím nejsou známy polynomiální algoritmy. Nabízí se tedy možnost nahradit exponenciální programy, které jsou k dispozici, řešeními na bázi parametrizované dostupnosti, které mohou potřebný čas pro výpočet výrazně omezit.

Prvním problémem, na který se zaměříme je, zda lze přepnutím zmenšit stupeň všech vrcholů pod stanovenou mez.

Přepnutí na grafy se všemi vrcholy stupně $\leq k$
Vstup: Graf $G = (V, E)$
Parametr: Přirozené číslo k
Otázka: Existuje množina vrcholů $A \subseteq V$ tak, že stupeň každého vrcholu v přepnutí množiny A v grafu G je nejvýše k, tj. $\forall v \in V : \deg_{S(G,A)} v \leq k$?

Abychom mohli s grafem lépe pracovat, potřebujeme mít graf v nějaké předem dané formě. Vybereme si ke každé třídě $[G]$ nějaké reprezentanta, kterého bude snadné pro všechny grafy nalézt. Tím pro nás bude graf s alespoň jedním izolovaným vrcholem. Jeho existence je zaručena lemmatem 1.4, které zároveň ukazuje postup, jak takového reprezentanta najít. Takováto příprava vstupní instance nám zabere pouze polynomiální čas a tedy si ji můžeme dovolit. Vrchol, který bude v upravené instanci izolovaný, si můžeme zvolit libovolně, např. vrchol nejmenšího stupně, a označíme ho v_0. Pro další úvahy je tedy naším zadáním graf G s izolovaným vrcholem v_0.

Je-li A množina vrcholů, která určuje hledané přepnutí, určuje i $V \setminus A$ toto přepnutí vzhledem k pozorování 1.2. Tato vlastnost by nám poněkud ztěžovala zápis řešení, proto budeme sledovat pouze taková A, že $v_0 \notin A$. Tím pouze vybíráme jednu z množin, kterou bude hledané přepnutí určeno a nemůžeme tedy žádné řešení ztratit. V hledaném přepnutí tak bude platit $N_{S(G,A)}(v_0) = A$. To ovšem znamená, že $|A| \leq k$. Odtud plyne jednoduchý algoritmus pracující v čase $\mathcal{O}\left(n^{k+2}\right)$.

My ovšem hledáme lepší, parametrizovaný, algoritmus. Základem bude v podstatě použití redukce na jádro problému, které je popsáno v sekci 3.1 kapitoly o parametrizované složitosti. Naše úloha je specifická tím, že přepnutí každého vrcholu ovlivní celý graf. Proto nemůžeme vyřešené části z grafu prostě mazat, neboť další kroky by mohli získané řešení opět narušit.

Naše pravidla tak pouze o některých vrcholech rozhodnou, že musí, nebo naopak nesmí být zařazeny do množiny A (tj. být přepnuty), máme-li získat řešení. Pro velké instance bude možné ukázat, že každý vrchol (kromě v_0) musí nebo nesmí být přepnut. Pokud se vyskytnou vrcholy, o nichž naše pravidla rozhodnou, že musí a zároveň nesmí být přepnuty, znamená to, že úloha nemá řešení. Jinak stačí pouze přepnout právě vrcholy, které musí být přepnuty a zkontrolovat, zda výsledný graf je řešením. Nelze proto mluvit ani o klasickém lemmatu o hranici.

Začneme tedy zavedením pravidel:

Lemma 3.1 (Pravidlo 1). *Vrchol v stupně většího než $2k$ musí být přepnut, abychom získali řešení.*

Důkaz. Budeme postupovat sporem. Nechť A je množina, která určuje přepnutí řešící zadání a $v \notin A$. Vrchol v má v G alespoň $2k+1$ sousedů. Při přepnutí mohl ztratit nejvýše $|A|$ z nich a tedy nejvýše k. Zůstalo mu tedy alespoň $k+1$ sousedů a přepnutí neřeší zadání, což je spor.

Lemma 3.2 (Pravidlo 2). *Vrchol v stupně menšího než $n-2k$ nesmí být přepnut, chceme-li získat řešení.*

Důkaz. Opět použijeme důkaz sporem. Nechť A je množina, která určuje přepnutí řešící zadání a $v \in A$. Vrchol v má v G stupeň d. Protože podle pozorování 1.2 je $S(G, A) = S(S(G, \{v\}), A \setminus \{v\})$ můžeme nejdříve přepnout v a pak zbylou nejvýše $(k-1)$-prvkovou množinu. Vrchol v bude mít po svém přepnutí stupeň $n-d-1$ a přepnutí $A \setminus \{v\}$ to může změnit nanejvýš o $k-1$. Protože $d < n-2k$, byl po přepnutí stupeň $n-d-1 > n-(n-2k)-1 = 2k-1$ a výsledný stupeň $> k$. Dostáváme opět spor s tím, že přepnutí podle A řeší zadání.

Důsledek 3.3 (Obdoba lemmatu o hranici). *Je-li $n \geq 4k+1$, pak na každý vrchol lze aplikovat alespoň jedno z pravidel 1 a 2.*

Důkaz. Nelze-li použít pravidlo 1 na vrchol stupně d, pak $d \leq 2k < (4k+1)-2k \leq n-2k$ a tedy lze použít pravidlo 2.

Na základě právě dokázaného lemmatu není těžké ukázat parametrizovanou dostupnost problému, pro lepší odhad časové složitosti budeme ještě potřebovat následující kombinatorické lemma.

Lemma 3.4. *Nechť $j \leq (n/3)$, pak $\sum_{i=0}^{j} \binom{n}{i} \leq 2 \cdot \binom{n}{j}$.*

Důkaz. Budeme postupovat indukcí podle j. Pro $j = 0$ je platnost zřejmá. Ukažme tedy, že výsledek platí pro $1 \leq j \leq (n/3)$, za předpokladu, že platí pro $j-1$. $\sum_{i=0}^{j} \binom{n}{i} = \binom{n}{j} + \sum_{i=0}^{j-1} \binom{n}{i}$, což je dle indukčního předpokladu menší než $\binom{n}{j} + 2\binom{n}{j-1} = \binom{n}{j} \cdot (1 + 2 \cdot \frac{j}{n-j+1})$. Ale $j \leq (n/3)$, proto $\frac{j}{n-j+1} \leq \frac{1}{2}$. Odtud $\binom{n}{j} \cdot (1 + 2 \cdot \frac{j}{n-j+1}) \leq 2\binom{n}{j}$ a tvrzení platí i pro j.

Věta 3.5. PŘEPNUTÍ NA GRAFY SE VŠEMI VRCHOLY STUPNĚ $\leq k$ *lze řešit v čase* $\mathcal{O}\left(\binom{4k-1}{k} \cdot k\right) \cdot \mathcal{O}(n) + \mathcal{O}(m)$, *je tedy parametrizovaně dostupné a pro každé pevné* k *máme lineární algoritmus.*

Důkaz. Algoritmus nejdříve vytvoří ze vstupní instance graf G s izolovaným vrcholem v_0. Pokud je $n > 4k$, zjistí, na které vrcholy se vztahuje pravidlo 1 a 2. Vztahují-li se na nějaký vrchol obě pravidla, odpoví NE a skončí. Stejně se zachová, pokud se pravidlo 2 vztahuje na více než k vrcholů. V ostatních případech přepne vrcholy, u kterých je to nutné a ověří, zda v tomto přepnutí mají všechny vrcholy stupeň $\leq k$.

Pro $n \leq 4k$ nejdříve aplikujeme obě pravidla a vrcholy, které musí být přepnuty přepneme. Pokud je takových více než k, odpovíme NE. Ze zbylých vrcholů, které přepnuty být mohou, ale nemusí, vyzkoušíme všechny množiny A, které počet přepnutých vrcholů doplní nejvýše na k, a pro každou z nich ověříme, zda přepnutí podle množiny A není hledaným řešením.

Korektnost algoritmu pro $n > 4k$ plyne z korektnosti obou pravidel. Pro ostatní n zkoušíme všechny možné kandidáty, správné řešení tedy jistě nemůžeme přeskočit.

Pokud jde o časovou náročnost algoritmu, graf G s izolovaným vrcholem lze získat v čase $\mathcal{O}(m)$. Zvolíme-li za v_0 vrchol nejmenšího stupně δ, pro každého jeho souseda je potřeba provést práci $\mathcal{O}(n)$, dohromady $\delta \cdot \mathcal{O}(n) = \mathcal{O}(m)$. V dalších fázích provádíme testy kandidátských množin velikosti nejvýše k. Přepnout graf podle kandidáta trvá $\mathcal{O}(k \cdot n)$ a ověřit $\mathcal{O}(n)$. Pro $n > 4k$ máme nejvýše jediného kandidáta, jeho vytvoření trvá $\mathcal{O}(n)$. Pro $n \leq 4k$ na vytvoření každého kandidáta postačí amortizovaně konstantní čas, v nejhorším případě $\mathcal{O}(n)$. Celkem je jich však až $\sum_{i=0}^{k} \binom{n-1}{i}$, což lze zhora odhadnout $2\binom{4k-1}{k}$ podle lemmatu 3.4, odkud plyne udávaný čas.

Algoritmus se dá ještě výrazně zrychlit. Než se do toho pustíme, je třeba upozornit na podstatný důsledek tohoto algoritmu:

Lemma 3.6. *Je-li* G *graf na* n *vrcholech, pak v* $[G]$ *existuje nejvýše jeden graf se všemi stupni vrcholů menšími než* $\lceil \frac{n}{4} \rceil$.

Důkaz. Položíme-li $k = \lceil \frac{n}{4} \rceil - 1$, bude $n > 4k$. V takovém případě lze podle lematu 3.3 o každém vrcholu rozhodnout, zda má být přepnut či ne, máme-li najít řešení. Existuje tedy nejvýše jeden kandidát na graf se všemi stupni $\leq k$.

Náš algoritmus můžeme ještě vylepšit. Vrcholy, které musíme přepnout budeme přepínat hned a průběžně si budeme udržovat počet vrcholů l, které ještě můžeme přepnout, abychom u vrcholu v_0 nepřekročili stupeň k. Na začátku nastavíme $l := k$. Naše pravidla upravíme následovně:
*Pravidlo 1':*Vrchol v stupně většího než $k + l$ přepneme, a snížíme l o 1. Tento vrchol již samozřejmě nesmí být znovu přepínán.
*Pravidlo 2':*Vrchol v stupně menšího než $n - k - l$ nesmí být přepnut, chceme-li získat řešení.

Důkazy korektnosti obou pravidel jsou zcela obdobné důkazům lemmat 3.1 a 3.2.

Úprava pravidel urychlí částečně zpracování některých velkých instancí, pro nás je ale podstatné zrychlení pro malé instance. Změny v lemmatu o hranici nám umožní jeho použití i v průběhu algoritmu.

Důsledek 3.7 (Vylepšené lemma o hranici). *Je-li $n \geq 2k + 2l + 1$, pak na každý vrchol lze aplikovat alespoň jedno z pravidel 1 a 2.*

Důkaz. Nelze-li použít pravidlo 1 na vrchol stupně d, pak $d \leq k + l < (2k + 2l + 1) - k - l \leq n - k - l$ a tedy lze použít pravidlo 2.

Je-li $0 \leq p \leq k - 1$ a $n = 2k + 2p + 2$ nebo $n = 2k + 2p + 1$, pak po přepnutí $k - p$ vrcholů lze o každém z vrcholů rozhodnout, zda má být přepnut, nebo ne. Kandidáty na prvních $k-p$ vrcholů ozkoušíme všechny, kromě nich také všechny množiny menší než $k-p$. V podstatě tak používáme prohledávací strom, který prokládáme s použitím redukce na jádro problému (viz sekce 3.2 kapitoly o parametrizované složitosti). Počet kandidátů, které je třeba ověřit, tedy pro takováto n odpovídá $\sum_{i=0}^{k-p} \binom{n-1}{i}$. To zhora odhadneme $\binom{2k+2p+1}{k-p} \cdot k$.

Abychom zjistili, pro které n je při daném k instance nejsložitější, musíme najít největší z čísel

$$\binom{2k+1}{k}, \binom{2k+3}{k-1}, \binom{2k+5}{k-2}, ..., \binom{2k+2p+1}{k-p}, ..., \binom{4k-3}{2}, \binom{4k-1}{1}.$$

Porovnáme proto dvě sousední hodnoty:

$$\binom{2k+2p+1}{k-p} \begin{matrix} \leqq \\ > \end{matrix} \binom{2k+2p+3}{k-p-1} \tag{3.1}$$

První člen lze rozepsat jako:

$$\frac{(2k+2p+1) \cdot (2k+2p) \cdot ... \cdot (k+3p+4) \cdot (k+3p+3) \cdot (k+3p+2)}{(k-p) \cdot (k-p-1)!}$$

a druhý jako

$$\frac{(2k+2p+3) \cdot (2k+2p+2) \cdot ... \cdot (k+3p+7) \cdot (k+3p+6) \cdot (k+3p+5)}{(k-p-1)!}$$

po zkrácení shodných členů zbyde pro $k - p \geq 3$:

$$(k + 3p + 4) \cdot (k + 3p + 3) \cdot (k + 3p + 2) \lesseqgtr (2k + 2p + 3) \cdot (2k + 2p + 2) \cdot (k - p).$$

Vztah pro přesné řešení tohoto porovnání je poměrně složitý, i když příslušná kubická rovnice má pouze jediné reálné řešení. Pro $k \to \infty$ nastává rovnost přibližně pro $p = 0.223k$, pro menší p je levá strana menší, pro vetší p větší. Totéž platí i v porovnání (3.1). Navíc se snadno ukáže, že pro $k \geq 2$ je

$$\binom{4k-3}{2} > \binom{4k-1}{1},$$

takže největší hodnota je pro velká k ta, který má $p = \lceil 0.223k \rceil$ tedy

$$\binom{2\lceil 1.223k \rceil + 1}{\lfloor 0.777k \rfloor}.$$

Pro $n \leq 2k$ nám nezbyde nic jiného, než vyzkoušet všechny správně velké možnosti pro množinu $V \setminus A$, ale pro $n \leq 2k$ je $\binom{n}{n-k} < \binom{2k+1}{k}$, takže se nejedná o nejtěžší instance.

Výsledek můžeme shrnout takto:

Věta 3.8. *Popsaný algoritmus řeší* Přepnutí na grafy se všemi vrcholy stupně $\leq k$ *v čase* $\mathcal{O}\left(\binom{2\lceil 1.223k \rceil + 1}{\lfloor 0.777k \rfloor} \cdot k^2\right) \cdot \mathcal{O}(n) + \mathcal{O}(m)$.

4 Přepnutí na k-regulární graf

Podle věty 2.3 je otázka, zda je graf ekvivalentní v přepnutí s nějakým regulárním grafem NP-úplná. Proto je zajímavé, zda následující parametrizovaný problém je parametrizovaně dostupný či nikoliv:

Přepnutí na k-regulární graf
Vstup: Graf $G = (V, E)$
Parametr: Přirozené číslo k
Otázka: Je G ekvivalentní v přepnutí s k-regulárním grafem?

Podle věty 2.1 máme pro tento problém samozřejmě algoritmus pracující v čase $\mathcal{O}\left(n^{k+4}\right)$. Ukážeme, že je tento problém parametrizovaně dostupný.

Nejdříve zvažme, že k-regulární graf má také všechny stupně $\leq k$. To ale podle lemmatu 3.6 znamená, že pro $n > 4k$ můžeme nalézt pomocí algoritmu z předchozí sekce 3 jediného kandidáta na takový graf a toho prověřit. Obdobně lze použít i druhou část algoritmu, jen je samozřejmě potřeba upravit ověření. To ukazuje parametrizovanou dostupnost tohoto problému.

Algoritmus však můžeme značně zrychlit, pokud využijeme toho, že hledáme přepnutí na graf s velmi zvláštní strukturou. Opět předpokládáme, že máme graf G s izolovaným vrcholem v_0. Můžeme přidat další pravidlo k pravidlům 1' a 2' ze sekce 3:

Lemma 4.1. *Je-li $n \geq 2k$ a vrchol v má stupeň $< k - l$ nebo $> n - k + l$, pak instance nemá řešení.*

Důkaz. Nechť v má stupeň $d < k-l$. Pokud jej nepřepneme, bude mít po přepnutí zbývajících l vrcholů stupeň nejvýše $d + l < k$. Pokud ho přepneme, bude mít stupeň $n - d - 1$, takže po přepnutí zbylých $l - 1$ vrcholů bude mít stupeň aspoň $n - d - 1 - l + 1 > 2k - (k - l) - l = k$, takže ani tato varianta k řešení nevede. Obdobně vrchol stupně $d > n - k + l$ může mít bez přepnutí stupeň $d - l > n - k + l - l \geq 2k - k = k$, po přepnutí $n - d - 1 + (l - 1) < n - (n - k + l) - 1 + (l - 1) = k - 2$.

Toto pravidlo sice urychlí zamítnutí některých zadání, nejdéle však trvá zpracování instancí o velikosti $n \leq 4k$, proto se zaměříme právě na ně. Každá kandidátská množina A, která má šanci na úspěch, musí mít velikost právě k (kvůli vrcholu v_0). Pro $n \leq 2k$ nám nezbyde nic jiného, než vyzkoušet všechny správně velké možnosti pro množinu $V \setminus A$. Zbývá $2k < n \leq 4k$.

Máme-li tedy $0 \leq p \leq k - 1$, instanci velikosti $n = 2k + 2p + 2$ nebo $n = 2k+2p+1$ a vrcholy si označíme $V = \{v_0, v_1, v_2, ..., v_{n-1}\}$, víme, že každé přepnutí, které vede k řešení, musí přepnout alespoň $k - p$ vrcholů z prvních $n - p - 1$, tedy z vrcholů $\{v_1, v_2, ..., v_{n-p-1}\}$. Pokud nalezneme těchto $k - p$ vrcholů, máme lemmatem 3.7 zaručeno, že o všech ostatních vrcholech již lze rozhodnout, zda mají nebo nemají být přepnuty, chceme-li dosáhnout řešení. V některých případech nám s hledáním pomůže pravidlo 1', mohou se ale vyskytnout i zadání, na kterých toto pravidlo nebude účinné.

Budeme tedy zkoušet všech nejvýše $\binom{n-p-1}{k-p} \leq \binom{2k+p+1}{k-p}$ možností, jak vybrat $k-p$ vrcholů z nejvýše $n-p$ možných, kde $n = 2k+2p+2$ nebo $n = 2k+2p+1$. Zjistíme, kolik jich pro pevné k může nejvíce být. Hledáme tedy největší z hodnot

$$\binom{2k+1}{k}, \binom{2k+2}{k-1}, ..., \binom{2k+p+1}{k-p}, ..., \binom{3k-1}{2}, \binom{3k}{1}$$

pro dané k. Porovnáme dvě sousední hodnoty

$$\binom{2k+p+1}{k-p} \begin{matrix} \leqq \\ > \end{matrix} \binom{2k+p+2}{k-p-1} \tag{4.1}$$

obě strany opět rozepíšeme:

$$\frac{(2k+p+1)\cdot(2k+p)\cdot ... \cdot (k+2p+4)\cdot(k+2p+3)\cdot(k+2p+2)}{(k-p)\cdot(k-p-1)!}$$

$$\frac{(2k+p+2)\cdot(2k+p+1)\cdot\ldots\cdot(k+2p+6)\cdot(k+2p+5)\cdot(k+2p+4)}{(k-p-1)!}$$

po zkrácení shodných členů zbyde pro všechna $0 \leq p \leq k-2$:

$$(k+2p+3)\cdot(k+2p+2) \lesseqgtr (2k+p+2)\cdot(k-p).$$

Toto porovnání můžeme řešit jako kvadratickou nerovnici s parametrem k. Vztah pro rovnost je tedy podstatně jednodušší než v sekci 3 a můžeme si ho uvést. V porovnání 4.1 tedy platí stejná nerovnost jako ve vztahu

$$p \lesseqgtr \frac{1}{10}(-5k+\sqrt{45k^2+60k+24}-12).$$

Druhé řešení této nerovnice nesplňuje $0 \leq p \leq k-2$. Pro $k \to \infty$ nastává rovnost přibližně pro $p = 0.171k$, takže největší hodnota je pro velká k ta, který má $p = \lceil 0.171k \rceil$, tedy

$$\binom{\lceil 2.171k \rceil + 1}{\lfloor 0.829k \rfloor}.$$

Vzhledem k tomu, že pro $n \leq 2k$ je $\binom{n}{n-k} < \binom{2k+1}{k}$, právě jsme ukázali následující:

Věta 4.2. *Popsaný algoritmus řeší* Přepnutí na k-regulární graf *v čase* $\mathcal{O}\left(\binom{\lceil 2.171k \rceil+1}{\lfloor 0.829k \rfloor}\cdot k\right)\cdot\mathcal{O}(n)+\mathcal{O}(m)$*, je tedy parametrizovaně dostupné a pro pevné* k *je náš algoritmus lineární.*

5 Přepnutí na grafy se všemi vrcholy stupně $\geq k$

Poslední ze tří úloh, které se ptají po přepnutí na grafy s nějak omezenými stupni vrcholů, je:

Přepnutí na grafy se všemi vrcholy stupně $\geq k$
Vstup: Graf $G = (V, E)$
Parametr: Přirozené číslo k
Otázka: Existuje množina vrcholů $A \subseteq V$ tak, že stupeň každého vrcholu v přepnutí množiny A v grafu G je aspoň k?

Podle pozorování 1.2 přepnutí grafu G na graf se všemi stupni $\geq k$ existuje, právě když existuje přepnutí doplňkového grafu $\overline{G}$ na graf se všemi stupni nejvýše $n-k-1$. Z hlediska klasické složitosti se tedy jedná téměř o stejnou úlohu. Z našeho pohledu jde však o parametrizovaný duál, takže parametrizovaná složitost obou problémů spolu nemusí vůbec souviset.

Ukážeme, že i tato úloha je parametrizovaně dostupná. Dokonce pro tento problém existuje tzv. garantovaná hodnota, neboli pro malý parametr vzhledem k velikosti instance máme zaručeno, že instance je kladná. To dokazuje následující lemma:

Lemma 5.1. *Nechť $G = (V, E)$ je graf na n vrcholech. Pak existuje graf $H \in [G]$, takový, že $\forall v \in V : \deg_H v \geq \lfloor n/2 \rfloor$.*

Důkaz. Budeme postupovat sporem. Nechť $G = (V, E)$ je graf na n vrcholech takový, že $\forall H \in [G] \quad \exists v \in V : \deg_H v < \lfloor n/2 \rfloor$. Označme H_0 graf s největším počtem hran v $[G]$. Nechť v_0 je takový, že $d = \deg_{H_0} v_0 < \lfloor n/2 \rfloor$. Přepneme vrchol v_0 v H_0. To neovlivní hrany, které nemají žádný konec ve v_0. Protože hran, které končily ve v_0, bylo před přepnutím d a po přepnutí jich je $n-d-1 > n-1-\lfloor n/2 \rfloor \geq \lfloor n/2 \rfloor - 1 \geq d$, má graf po přepnutí více hran, což je spor s maximalitou grafu H_0.

Poznámka 5.2. Z lemmatu 5.1 a pozorování 1.2 použitím na doplněk grafu plyne, že taková instance (G, k) problému Přepnutí na grafy se všemi vrcholy stupně $\leq k$, pro kterou platí $n \leq 2k$, je vždy kladná. Vzhledem k tomu, že se však nejedná o nejtěžší zadání tohoto problému, nemůže to ovlivnit asymptotickou časovou náročnost uváděného algoritmu.

Máme-li zajištěnu garantovanou hodnotu, snadno se dokáže parametrizovaná dostupnost.

Věta 5.3. Přepnutí na grafy se všemi vrcholy stupně $\geq k$ *lze rozhodnout v čase $\mathcal{O}\left(2^{2k} \cdot k^2\right)$, tedy pro pevné k v konstantním čase. Pokud požadujeme, aby algoritmus vydal příklad nějakého řešení, pak lze navrhnout takový, který pracuje v čase $\mathcal{O}\left(2^{2k} \cdot k^2\right) + \mathcal{O}\left(n^3\right)$.*

Důkaz. Je-li $n > 2k$, pak $k \leq \lfloor n/2 \rfloor$ a tedy podle lemmatu 5.1 můžeme ihned odpovědět ANO. Zároveň lemma dává návod, jak takové řešení hledat. Začneme s grafem G. V každém kroku přepneme nějaký vrchol příliš malého stupně. Jeho nalezení trvá $\mathcal{O}(n)$, stejně jako jeho přepnutí. V každém kroku se zvýší počet hran, takže kroků může být nejvýše n^2. Pokud nenalezneme vrchol malého stupně, nalezli jsme řešení.

Pro $n \leq 2k$ vyzkoušíme všech $2^{n-1} = 2^{2k-1}$ přepnutí a pro každé ověříme, zda je řešením. Každé přepnutí trvá nejvýše $\mathcal{O}(n^2) = \mathcal{O}(k^2)$ a ověření $\mathcal{O}(k)$. Odtud plyne požadovaný čas.

Korektnost algoritmu plyne především z lemmatu 5.1.

Vzhledem k tomu, že máme jistou hodnotu zaručenu, nabízí se parametrizace nad zaručené hodnoty. Tedy taková verze, kde otázku změníme na „Existuje množina vrcholů $A \subseteq V$ tak, že stupeň každého vrcholu v přepnutí množiny A v grafu G je aspoň $\lfloor n/2 \rfloor + k$?“. Ovšem stejně jako nejsou známy k většině podobných parametrizací, ani k této neznáme žádné výsledky.

6 Přepnutí na graf s nejvýše k hranami

V důkazu lemmatu 5.1 jsme se odkazovali na takové přepnutí našeho grafu, které má největší počet hran. Jak ale takové přepnutí najít? Není znám žádný polynomiální algoritmus, který by takové přepnutí našel, nebo jen ukázal, kolik má hran. Přejdeme-li k doplňku, aby se problém lépe parametrizoval, jedná se o následující úlohu:

Přepnutí na graf s nejvýše k hranami
Vstup: Graf $G = (V, E)$
Parametr: Přirozené číslo k
Otázka: Existuje $A \subseteq V$ tak, že $|E_{S(G,A)}| \leq k$ tedy, že přepnutí grafu G podle A má nejvýše k hran?

Má-li graf nejvýše k hran, je i stupeň každého vrcholu nejvýše k. Všechna přepnutí, která by mohla mít stupně všech vrcholů $\leq k$, nám generuje algoritmus ze sekce 3. Stačí tedy zaměnit původní ověření za nové, které v čase $\mathcal{O}(n)$ zjistí zda graf nemá více hran než k, a získáme algoritmus, který řeší náš problém v čase $\mathcal{O}\left(\binom{2\lceil 1.223k\rceil+1}{\lfloor 0.777k\rfloor} \cdot k^2\right) \cdot \mathcal{O}(n) + \mathcal{O}(m)$.

Ovšem náš graf je omezen mnohem více. Toho můžeme využít k sestavení algoritmu, který by tento problém řešil rychleji, než ten ze sekce 3. Základní příprava bude stejná, předpokládáme, že náš graf G má izolovaný vrchol v_0 (viz lemma 1.4). Opět použijeme redukci na jádro problému a předpokládáme, že vrchol v_0 nebude přepnut, můžeme tedy přepnout maximálně k vrcholů. V průběhu algoritmu si udržujeme množinu B vrcholů, které jsme dosud přepnuli a počet $l = k - |B|$ vrcholů, které můžeme ještě přepnout. Na začátku je $B = \emptyset$ a $l = k$. Zavedeme tedy dvě pravidla:

Lemma 6.1 (Pravidlo 1). *Vrchol* $v \in V \setminus B \setminus \{v_0\}$ *stupně* $\deg_{S(G,B)} v > l$ *musí být přepnut, máme-li najít řešení.*

Důkaz. Předpokládejme, že $v \in V \setminus B \setminus \{v_0\}$ je vrchol stupně $d = \deg_{S(G,B)} v > l$ a $A \subset V \setminus B \setminus \{v_0\}$ je množina, jejíž přepnutí v $S(G, B)$ dává graf H. Ukážeme, že pokud $v \notin A$, pak graf H má příliš mnoho hran. Označme $D = A \cap N_{S(G,B)}(v)$. Pak $\{\{v_0, x\} | x \in D \cup B\}$ a $\{\{v, x\} | x \in N_{S(G,B)}(v) \setminus D\}$ jsou dvě disjunktní množiny hran grafu H jejichž velikost je dohromady rovna $|B| + |D| + |N_{S(G,B)}(v) \setminus D| = |B| + |N_{S(G,B)}(v)| = d + k - l > k$. Graf H má tedy více než k hran a přepnutí množiny A nevede k řešení.

Takový vrchol ihned přepneme, přidáme do B a l snížíme o jedna.

Lemma 6.2 (Pravidlo 2). *Vrchol* $v \in V \setminus B \setminus \{v_0\}$ *stupně* $\deg_{S(G,B)} v < n - l - 1$ *nesmí být přepnut, máme-li najít řešení.*

Důkaz. Mějme opět $d = \deg_{S(G,B)} v < n - l - 1$, $A \subset V \setminus B \setminus \{v_0\}$ a $H = S(G, A \cup B)$. Předpokládejme, že $v \in A$. Potom $\{\{v_0, x\} | x \in A \cup B \setminus \{v\}\}$ a $\{\{v, x\} | x \in V \setminus N_{S(G,B)}(v) \setminus A\}$ jsou dvě disjunktní množiny hran grafu H. Dohromady je jejich velikost aspoň $(|B|+|A|-1)+(n-d-|A|) = k-l-1+n-d > k - l + n - (n - l - 1) - 1 = k$. Graf H má tedy opět příliš mnoho hran a tedy ani přepnutí této množiny A nevede k řešení.

Lemma 6.3 (Lemma o hranici). *Je-li $n > 2l+1$, pak na každý vrchol lze aplikovat pravidlo 1 nebo 2.*

Důkaz. Pokud pravidlo jedna nelze aplikovat na vrchol v stupně d, je $d \leq l = (2l + 1) - l - 1 < n - l - 1$ a tedy lze použít pravidlo 2.

Jsme tedy ve velmi podobné situaci jako v předchozích případech. Máme-li $n > 2k + 1$, pak postupně rozhodneme o všech vrcholech, jestli mají, nebo nemají být přepnuty. Pokud některý zároveň musí a nesmí být přepnuty, pak pro toto zadání neexistuje řešení, odpovíme NE. Dosáhneme-li hodnoty $l = 0$ stačí ověřit, že všechny vrcholy o kterých jsme ještě nerozhodovali jsou izolované. Pokud tomu tak není, pak takovou instanci zamítneme, neboť nemá řešení. Jinak ověříme, kolik má naše přepnutí hran.

Je-li $0 \leq p \leq k-1$ a $n = 2p+2$ nebo $n = 2p+1$, pak po přepnutí $k-p$ vrcholů lze podle lemmatu 6.3 o každém z vrcholů rozhodnout, zda má být přepnut, nebo ne. Potřebujeme tedy získat nanejvýš prvních $k-p$ vrcholů. Možná některé získáme použitím pravidla 1, jinak ozkoušíme všechny správně velké množiny, kterými je doplníme. Kromě nich také všechny množiny menší než $k - p - l$. Opět tedy používáme prohledávací strom prokládaný s použitím redukce na jádro problému. Přepnutí trvá $\mathcal{O}(k^2)$ a testování $\mathcal{O}(k)$.

Počet kandidátů, které může být potřeba ověřit, tedy pro takováto n odpovídá $\sum_{i=0}^{k-p} \binom{n-1}{i}$. To shora odhadneme $\binom{2p+1}{k-p} \cdot k$ pro $p \geq k/2$ a 2^k pro $p < k/2$. Abychom našli nejtěžší instanci pro pevné k, stačí porovnat hodnoty

$$\binom{2\lceil k/2 \rceil + 1}{\lfloor k/2 \rfloor}, \binom{2\lceil k/2 \rceil + 3}{\lfloor k/2 \rfloor - 1}, ..., \binom{2p+1}{k-p}, ..., \binom{2k-3}{2}, \binom{2k-1}{1},$$

protože pro $k \geq 2$ je

$$2^k = \frac{1}{2} \sum_{i=0}^{k+1} \binom{k+1}{i} \leq \frac{k+2}{2} \cdot \binom{2\lceil k/2 \rceil + 1}{\lfloor k/2 \rfloor} \leq k \cdot \binom{2\lceil k/2 \rceil + 1}{\lfloor k/2 \rfloor}.$$

Stejně jako v předchozích sekcích zjistíme největší hodnotu tak, že porovnáme dvě sousední:

$$\binom{2p+1}{k-p} \lesseqgtr \binom{2p+3}{k-p-1}$$

Levá strana se rozepíše na:

$$\frac{(2p+1) \cdot (2p) \cdot (2p-1) \cdot ... \cdot (3p-k+4) \cdot (3p-k+3) \cdot (3p-k+2)}{(k-p) \cdot (k-p-1)!}$$

a pravá jako:

$$\frac{(2p+3)\cdot(2p+2)\cdot(2p+1)\cdot\ldots\cdot(3p-k+7)\cdot(3p-k+6)\cdot(3p-k+5)}{(k-p-1)!}$$

Když zkrátíme činitele, které se objevují na obou stranách, vyjde pro $k/2 \leq p \leq k-3$:

$$(3p-k+4)\cdot(3p-k+3)\cdot(3p-k+2) \lesseqgtr (2p+3)\cdot(2p+2)\cdot(k-p)$$

Řešení kubické rovnice s parametrem k, která vznikne, pokud požadujeme rovnost obou stran, je opět značně komplikované, ikdyž reálné řešení je opět pouze jediné. Pro nás je důležité, že pro $k \to \infty$ nastává rovnost přibližně pro $p = 0,611k$. Snadno se ověří, že pro $k \geq 4$ je

$$\binom{2k-3}{2} > \binom{2k-1}{1}.$$

Nejvíc kandidátů je tak pro velká k v instanci, které odpovídá $p = \lceil 0,611k \rceil$. Tam jich je

$$\binom{2\lceil 0,611k \rceil + 1}{\lfloor 0,389k \rfloor}.$$

To dává časovou složitost právě popsaného algoritmu tak, jak ji shrnuje následující věta:

Věta 6.4. PŘEPNUTÍ NA GRAF S NEJVÝŠE k HRANAMI *lze řešit v čase* $\mathcal{O}\left(\binom{2\lceil 0,611k \rceil+1}{\lfloor 0,389k \rfloor} \cdot k^2\right) \cdot \mathcal{O}(n) + \mathcal{O}(m)$, *je tedy parametrizovaně dostupné a pro pevné* k *máme lineární algoritmus.*

7 Přepnutí na graf prostý zakázaného podgrafu

Je známa řada polynomiálních algoritmů, které rozhodnou, zda lze graf G přepnout na graf G', který neobsahuje pevně zvolený graf H jako podgraf (viz sekce 2). Většinou používají převod na 2-SAT, na soustavu lineárních rovnic nad GF(2), nebo charakterizaci pomocí zakázaných podgrafů.

Můžeme se však ptát, nakolik musíme náš graf změnit, tj. kolik vrcholů je třeba přepnout, pokud se chceme podgrafu H zbavit. Pak ale zjistíme, že VÁŽENÝ 2-SAT je NP-úplný [18], pro problém ŘEŠENÍ LINEÁRNÍHO PROGRAMOVÁNÍ NAD GF(2) S NEJMENŠÍM POČTEM JEDNIČEK také není znám polynomiální algoritmus a charakterizace pomocí zakázaných podgrafů vůbec není konstruktivní. Proto je zajímavé zkoumat tyto problémy z parametrizovaného hlediska. Ukážeme, že jsou parametrizovaně dostupné.

Jedním z nejvýznamnějších takovýchto problémů je:

Přepnutí na graf bez trojúhelníků
Vstup: Graf G
Parametr: Přirozené číslo k
Otázka: Lze graf G změnit na graf, který neobsahuje trojúhelníky, pomocí nejvýše k operací přepnutí vrcholu?

Jak jsme již uvedli na straně 33, Hayward [32, 33] a Hage et al. [31] nezávisle představili dva různé algoritmy na rozpoznávání grafů, které lze přepnout na graf bez trojúhelníků. Oba pracují v čase $\mathcal{O}(n^3)$, ale používají převod na 2-SAT. Ukážeme, že stejné složitosti lze dosáhnout i pro pevně omezený počet přepnutí.

Věta 7.1. Přepnutí na graf bez trojúhelníků *lze řešit v čase* $\mathcal{O}\left(3^k\right)\cdot\mathcal{O}(n^3)$.

Důkaz. Použijeme metodu omezeného prohledávacího stromu, tak je popsána v sekci 3.2 kapitoly o parametrizované složitosti. Náš algoritmus tedy bude rekurzivní, každé volání dostane na vstupu graf G, číslo k a množinu A již přepnutých vrcholů.

Na začátku každého volání prohledáme graf $S(G, A)$ a nalezneme v něm všechny trojúhelníky. Nenalezneme-li žádný, je výsledek tohoto volání ANO, s tím, že množina, která odpovídá řešení je A. Je-li $|A| = k$ a graf nějaký trojúhelník obsahuje bude výsledek NE, stejně jako když je $|A|$ jakákoliv a graf $G[A] = S(G, A)[A]$ obsahuje trojúhelník.

Jinak vybereme takový trojúhelník, který má nejvíce vrcholů v A. Z těch, které mají stejně, vybereme libovolně. Pak pro každý vrchol v tohoto trojúhelníka, který neleží v A, zavoláme rekurzivně algoritmus na graf G, číslo k a množinu $A \cup \{v\}$. Pokud některé z těchto volání odpoví ANO, i my odpovíme ANO, se stejnou množinou jako řešením. Pokud všechny odpoví NE, budeme tuto odpověď kopírovat.

Výsledkem celého algoritmu je pak výsledek volání na G, k a $\emptyset$.

Korektnost algoritmu: Pokud algoritmus odpoví ANO, nalezl řešení a tedy je tato odpověď správná. Naopak je-li A nějaká množina a H trojúhelník v grafu $S(G, A)$, pak každé řešení, jehož je A podmnožina, musí přepnout alespoň jeden vrchol z trojúhelníka H, který není v A. Každé volání tedy korektně rozhodne o existenci řešení velikosti nejvýše k, jehož je A podmnožina. Kořen tak korektně rozhodne o existenci řešení.

Časová složitost: V každém volání provádíme hledání trojúhelníků, to lze provést v čase $\mathcal{O}(n^3)$. Všechna volání tvoří zakořeněný strom, který se v každém vrcholu větví na nejvýše tři syny a jehož hloubka nepřesahuje k. Proto může mít nejvýše $\sum_{i=0}^{k} 3^i = \frac{1}{2}(3^{k+1} - 1) = \mathcal{O}\left(3^k\right)$ vrcholů, což dává požadovanou složitost.

Tento výsledek není příliš závislý na tom, že se jedná právě o trojúhelníky. Můžeme ho zobecnit na celé množiny zakázaných podgrafů. Mějme tedy dánu konečnou množinu $\mathcal{S} = \{H_1, H_2, ..., H_p\}$ zakázaných podgrafů. Definujeme problém

Přepnutí na graf prostý indukovaného podgrafu z $\mathcal{S}$
Vstup: Graf G
Parametr: Přirozené číslo k
Otázka: Lze graf G změnit na graf, který neobsahuje žádný z grafů v množině $\mathcal{S}$ jako indukovaný podgraf, pomocí nejvýše k operací přepnutí vrcholu?

Ještě než vyslovíme větu, označme $l(\mathcal{S}) = \max\{|V_H||H \in \mathcal{S}\}$, tedy velikost největšího grafu z množiny $\mathcal{S}$.

Věta 7.2. *Problém* Přepnutí na graf prostý indukovaného podgrafu z $\mathcal{S}$ *je pro každou konečnou množinu grafů* $\mathcal{S}$ *parametrizovaně dostupný*[1] *a řešitelný v čase* $\mathcal{O}\left(l(S)^k\right) \cdot \mathcal{O}\left(n^{l(S)}\right)$.

Důkaz. Algoritmus bude velmi podobný tomu pro trojúhelníky. Bude opět rekurzivní, volání si mezi sebou budou předávat stejné údaje, tedy graf G, k a množinu A již přepnutých vrcholů.

Graf $S(G, A)$ nyní na začátku každého volání prohledáme na všechny grafy z $\mathcal{S}$. Pokud žádný nenalezneme, odpovíme ANO, je-li $|A| = k$ a nějaký nalezneme, odpovíme NE, totéž obsahuje-li $G[A]$ graf z $\mathcal{S}$ ať už je A jakkoliv velká.

V ostatních případech vybereme takový nalezený graf, který má nejméně vrcholů mimo A. Pro každý takový vrchol v zavoláme algoritmus rekurzivně se vstupy G, k a $A \cup \{v\}$. Pokud některé z těchto volání odpoví ANO, odpovíme i my, jinak odpovíme NE. Celkový výsledek dává opět volání G, k a $\emptyset$.

Zdůvodnění korektnosti je zcela analogické tomu z věty 7.1, podobné je i zdůvodnění časové složitosti.

Nalezení zakázaného pografu H na r vrcholech trvá $\mathcal{O}(n^r)$. Vyzkoušíme všechny r-tice (ve všech pořadích) a porovnáme s grafem H. Porovnání sice trvá kvadratický čas v r, ale z hlediska k i n jde o konstantu. Nalezení všech zakázaných podgrafů tak trvá nejvýše $\mathcal{O}\left(|\mathcal{S}| \cdot n^{l(S)} \cdot l(S)^2\right) = \mathcal{O}\left(n^{l(S)}\right)$.

Prohledávací strom má opět hloubku k, synové jednotlivých vrcholů vždy odpovídají vrcholům nějakého zakázaného podgrafu, jejich počet tedy nemůže překročit $l(S)$. To dává omezení na celkovou velikost stromu ve tvaru $\sum_{i=0}^{k} l(S)^i = \frac{1}{l(S)-1}(l(S)^{k+1} - 1) = \mathcal{O}\left(l(S)^k\right)$. Odtud plyne celková složitost.

Poznámka 7.3. Omezení na indukované podgrafy není nijak zásadní, spíše usnadňuje zápis. Chceme-li zakázat graf $H = (V_H, E_H)$ jako podgraf, ne pouze jako indukovaný podgraf, stačí pouze použít množinu $\mathcal{S}_H = \{G|V_G = V_H$ a H je podgrafem $G\}$.

[1] Věta 7.2 ukazuje parametrizovanou dostupnost problému s parametrem k pro každé pevné $\mathcal{S}$. Neukazuje parametrizovanou dostupnost problému s parametrem k a $\mathcal{S}$ dohromady a je velmi nepravděpodobné, že by tento problém byl parametrizovaně dostupný.

Kapitola 5

Závěr

Výsledky ukazují, že teorie parametrizované složitosti má svoje uplatnění i v úlohách spojených se Seidlovým přepnutím. Dokázali jsme, že problémy PŘEPNUTÍ NA GRAF SE VŠEMI VRCHOLY STUPNĚ NEJVÝŠE k, PRÁVĚ k, ASPOŇ k, PŘEPNUTÍ NA GRAF S NEJVÝŠE k HRANAMI a PŘEPNUTÍ NA GRAF PROSTÝ ZAKÁZANÉHO PODGRAFU jsou parametrizovaně dostupné problémy. Věříme, že časová náročnosti algoritmů udávaných větami 3.8 (strana 38) a 6.4 (strana 44) lze zlepšit o faktor k pouze zpřesněním použitých odhadů, což však nezlepší jejich použitelnost. Ta je ovšem z hlediska parametrizované složitosti poměrně dobrá.

U problémů, které jsme zkoumali, kromě PŘEPNUTÍ NA k-REGULÁRNÍ GRAF, není jejich klasická složitost známá. Je pravděpodobné, že jsou NP-úplné, přesto tato otázka zůstává otevřená. Každopádně naše algoritmy jsou pro tyto problémy dosud nejrychlejší.

Existuje však celá řada dalších problémů spojených se Seidlovým přepnutím, jejichž parametrizovaná, někdy dokonce ani klasická, složitost není známa. Uveďme alespoň některé zajímavé příklady.

Přepnutí na k degenerovaný graf Graf je k-degenerovaný, pokud každý jeho podgraf má alespoň jeden vrchol stupně nejvýše k. Pro tento problém nám poskytuje věta 2.1 ze strany 29 algoritmus pracující v čase $\mathcal{O}\left(n^{k+5}\right)$. Je-li k součástí vstupu, není polynomiální algoritmus znám. Proto by mohla být parametrizovaná dostupnost zajímavou alternativou.

Přepnutí na bipartitní graf Tuto úlohu lze chápat jako přepnutí na graf bez lichých kružnic. Věta 2.13 ze strany 31 popisuje polynomiální řešení převodem na 2-SAT, není-li počet přepnutí vrcholů omezen. To ukazuje, že omezíme-li maximální počet přepnutí v zadání, může se jednat o výrazně těžší problém. Lichých kružnic je nekonečně mnoho, proto nelze použít větu 7.2 (strana 46). Pokud by se podařilo prokázat parametrizovanou dostupnost, bylo by to jistě přínosem.

Tyto dva problémy nejvíce souvisí s výsledky, kterých jsme dosáhli. Ale samozřejmě existuje mnohem více zajímavých parametrizovaných problémů spojených se Seidelovým přepnutím. Naše výsledky mohou být dobrou startovní pozicí pro útok na tyto úlohy.

Literatura

[1] Karl R. Abrahamson, John A. Ellis, Michael R. Fellows, and Manuel E. Mata. On the complexity of fixed parameter problems (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 210–215, 1989.

[2] Jochen Alber, Hans L. Bodlaender, Henning Fernau, Ton Kloks, and Rolf Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002.

[3] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[4] K. Appel and W. Haken. Every planar map is 4-colorable - 1: Discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977.

[5] K. Appel and W. Haken. Every planar map is 4-colorable - 2: Reducibility. *Illinois Journal of Mathematics*, 21:491–567, 1977.

[6] R. Balasubramanian, Michael R. Fellows, and Venkatesh Raman. An improved fixed-parameter algorithm for vertex cover. *Information Processing Letters*, 65(3):163–168, 1998.

[7] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *Society for Industrial and Applied Mathematics Journal on Computing*, 25(6):1305–1317, 1996.

[8] Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In Igor Prívara and Peter Ruzicka, editors, *Mathematical Foundations of Computer Science 1997, 22nd International Symposium, MFCS'97, Bratislava, Slovakia, August 25-29, 1997, Proceedings*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 1997.

[9] S. Buss. osobní korespondence s R. Downeym a M. Fellowsem. nepublikováno, 1989.

[10] Marco Cesati. The turing way to parameterized complexity. *Journal of Computer and System Sciences*, 67(4):654–685, 2003.

[11] L. Sunil Chandran and Fabrizio Grandoni. Refined memorization for vertex cover. *Information Processing Letters*, 93(3):123–131, 2005.

[12] Benny Chor, Michael R. Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $\mathcal{O}(n^2)$ steps. In *Graph-Theoretic Concepts in Computer Science, International Workshop WG*, pages 257–269, 2004.

[13] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[14] Stephen A. Cook. A hierarchy for nondeterministic time complexity. In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing*, pages 187–192, 1972.

[15] Rodney G. Downey and Michael R. Fellows. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research*, pages 191–225, 1992.

[16] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *Society for Industrial and Applied Mathematics Jornal on Computing*, 24(4):873–921, 1995.

[17] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1&2):109–131, 1995.

[18] Rodney G. Downey and Michael R. Fellows. *Parametrized Complexity*. Monographs in Computer Science. Springer, 1999.

[19] S.E. Dreyfus and R.A.Wagner. The steiner problem in graphs. *Networks*, 1:195–207, 1972.

[20] Jack Edmonds. Minimum partition of a matroid into independent subsets. *Journal of Research National Bureau of Standards*, 69 B:67–72, 1965.

[21] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972.

[22] Andrzej Ehrenfeucht, Jurriaan Hage, Tero Harju, and Grzegorz Rozenberg. Complexity issues in switching of graphs. In Hartmut Ehrig, Gregor Engels, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, *Theory and Application of Graph Transformation*, volume 1764 of *Lecture Notes in Computer Science*, pages 59–70. Springer, 1998.

[23] Andrzej Ehrenfeucht, Jurriaan Hage, Tero Harju, and Grzegorz Rozenberg. Pancyclicity in switching classes. *Information Processing Letters*, 73(5–6):153–156, 2000.

[24] Michael R. Fellows. On the complexity of vertex set problems. Technical report, Computer Science Department, University of New Mexiko, 1988.

[25] Michael R. Fellows and Michael A. Langston. Nonconstructive advances in polynomial-time complexity. *Information Processing Letters*, 26(3):155–162, 1987.

[26] Moti Frances and Ami Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.

[27] M. Garey and D. Johnson. *Computers and Intractability, A guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[28] Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37(1):25–42, 2003.

[29] Jurriaan Hage. *Structural Aspects Of Switching Classes*. PhD thesis, Leiden Institute of Advanced Computer Science, 2001.

[30] Jurriaan Hage and Tero Harju. A characterization of acyclic switching classes of graphs using forbidden subgraphs. *SIAM J. Discrete Math.*, 18(1):159–176, 2004.

[31] Jurriaan Hage, Tero Harju, and Emo Welzl. Euler graphs, triangle-free graphs and bipartite graphs in switching classes. In Andrea Corradini, Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, *Graph Transformation, First International Conference, ICGT 2002, Barcelona, Spain, October 7-12, 2002, Proceedings*, volume 2505 of *Lecture Notes in Computer Science*, pages 148–160. Springer, 2002.

[32] Ryan B. Hayward. Recognizing P_3-structure: A switching approach. *J. Comb. Theory, Ser. B*, 66(2):247–262, 1996.

[33] Ryan B. Hayward, Stefan Hougardy, and Bruce A. Reed. Polynomial time recognition of P_4-structure. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 382–389, 2002.

[34] Alain Hertz. On perfect switching classes. *Discrete Applied Mathematics*, 94(1-3):3–7, 1999.

[35] Weijia Jia, Chuanlin Zhang, and Jianer Chen. An efficient parameterized algorithm for m-set packing. *Journal of Algorithms*, 50(1):106–117, 2004.

[36] David S. Johnson. The NP-completeness column. *ACM Transactions on Algorithms*, 1(1):160–176, 2005.

[37] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 193–206, New York, NY, USA, 1983. ACM Press.

[38] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Plenum Press, 1972.

[39] David G. Kirkpatrick and Pavol Hell. On the completeness of a generalized matching problem. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing, 1-3 May 1978, San Diego, California, USA*, pages 240–245. ACM, 1978.

[40] Jan Kratochvíl. Complexity of hypergraph coloring and seidel's switching. In Hans L. Bodlaender, editor, *WG*, volume 2880 of *Lecture Notes in Computer Science*, pages 297–308. Springer, 2003.

[41] Jan Kratochvíl. Perfect codes and two-graphs. *Commentationes Mathematicae Universitatis Carolinae. Charles Univ., Prague*, 30:755–760, 1989.

[42] Jan Kratochvíl, Jaroslav Nešetřil, and Ondřej Zýka. On the computational complexity of seidel's switching. In *Proceedings 4th Czechoslovak Symposium on Combinatorics, Prachatice 1990*, volume 51 of *Annals of Discrete Math.*, pages 161–166. North Holland, Amsterdam, 1992.

[43] H.W. Lenstra Jr. Integer programming with a fixed number of variables. Technical Report 81-03, Mathematisch Instituut, Universiteit ban Amsterdam, 1981.

[44] Kurt Mehlhorn. *Data Structures and Algorithms*, volume 1 of *Monographs in Theoretical Computer Science. An European Association for Theoretical Computer Science Monographs Series*. Springer, 1984.

[45] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and Its Aplications*. Oxford University Press, 2006.

[46] Rolf Niedermeier and Peter Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73(3-4):125–129, 2000.

[47] Eva Ondráčková. Computational complexity in graph theory. Master's thesis, Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University in Prague, 2006.

[48] Christos H. Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the V-C dimension. In *Proc. of the 8th Annual Conference on Structure in Complexity Theory, CSCT'93 (San Diego, California, May 1993)*, pages 12–18. IEEE Computer Society Press, 1993.

[49] Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[50] Neil Robertson and Paul D. Seymour. Graph minors I. Excluding a forest. *J. of Combinatorial Theory, Series B*, 35(1):39–61, 1983.

[51] Neil Robertson and Paul D. Seymour. Graph minors II. Algorithmic aspects of tree-width. *Journal on Algorithms*, 7(3):309–322, 1986.

[52] Neil Robertson and Paul D. Seymour. Graph minors IV. Tree-width and well-quasi-ordering. *J. of Combinatorial Theory, Series B*, 48(2):227–254, 1990.

[53] Neil Robertson and Paul D. Seymour. Graph minors XIII. The disjoint paths problem. *J. of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[54] J.J. Seidel. Graphs and two-graphs. In *Proc. 5th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, Winnipeg, Canada, 1974. Utilitas Mathematica Publishing Inc.

[55] J.J. Seidel. A survey of two-graphs. In *Proc. Colloquio Internazionale sulle Teorie Combinatorie (Rome,1973)*, volume 17, pages 481–511, Rome, 1976. Accademia Nazionale dei Lincei.

[56] J.J. Seidel and D.E. Taylor. Two-graphs, a second survey. In *Algebraic Methods in Graph Theory (Proc. Internat. Colloq., Szeged,1978)*, volume II, pages 689–711, Amsterdam, 1981. North-Holland.

[57] Christian Sloper. *Techniques in parametrized algorithm design.* PhD thesis, University of Bergen, Norway, 2006.

[58] Gerhard J. Woeginger. Exact algorithms for NP-hard problems: A survey. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–208. Springer, 2001.