NTIN099 — Algoritmické aspekty booleovských funkcí a parametrizovaná složitost

Petr Kučera

2016/17

Úvod

Sylabus

1. Exponenciální algoritmy pro k-SAT a obecný SAT
2. Úvod do parametrizované složitosti, příklady z teorie grafů.
3. Parametrizované algoritmy pro SAT založené na backdoor množinách.
4. Algoritmy pro SAT parametrizované stromovou šířkou.
5. Algoritmy pro SAT parametrizované deficiencí formule vzhledem k matched formulím
6. Kernelizace pro MaxSAT a parametrizace MaxSAT
7. Algoritmy pro MaxSAT založené na větvení a prořezávání (branch & bound)
8. Aproximační algoritmy pro MaxSAT

Obsah I

Obsah II

Parametrizované algoritmy a složitost

Parametrizovaný problém

Parametrizovaný problém

Definice 1

Parametrizovaný problém je formálně definován jako jazyk $L \subseteq \Sigma^* \times \Sigma^*$. Druhý prvek je zván parametrem problému L.

Definice 2

Parametrizovaný problém L je řešitelný s pevným parametrem (fixed-parameter tractable), pokud existuje deterministický algoritmus, který pro danou dvojici (x, k)

- rozhodne, zda $(x, k) \in L$, a
- pracuje v čase $f(k) \cdot n^{O(1)}$, kde $f(k)$ je algoritmicky vyčíslitelná funkce.

Problémy řešitelné s pevným parametrem tvoří třídu FPT.

- Parametr bývá obvykle číselný, ale není to podmínkou.

Vrcholové pokrytí

Vrcholové pokrytí (VP)	
Instance	Graf $G = (V, E)$, celé číslo $k \geq 0$.
Otázka	Existuje v G vrcholové pokrytí velikosti k? Přesněji, existuje množina $S \subseteq V$ vrcholů velikosti nejvýš k, která obsahuje alespoň jeden vrchol z každé hrany grafu G?

- Nadále označujeme $n = |V|$ a $m = |E|$.
- Je-li G graf a v jeho vrchol, pomocí $G \setminus \{v\}$ označujeme graf vzniklý z G odstraněním vrcholu v i všech hran, v nichž se vyskytuje.

Jednoduchý algoritmus pro VP

Algoritmus 1 Hledání VP grafu $G = (V, E)$ velikosti nejvýš k

```
1:  function HledejVP(G, k)
2:      if E = ∅ then
3:          return ∅
4:      end if
5:      if k = 0 then
6:          return NEEXISTUJE
7:      end if
8:      Vyber libovolnou hranu {u, v} ∈ E
9:      if HledejVP(G \ {u}, k − 1) najde množinu S′ then
10:         return S = S′ ∪ {u}
11:     else if HledejVP(G \ {v}, k − 1) najde množinu S″ then
12:         return S = S″ ∪ {v}
13:     else
14:         return NEEXISTUJE
15:     end if
16: end function
```

Vlastnosti algoritmu pro VP

- Funkce HledejVP(G, k) najde vrcholové pokrytí velikosti nejvýš k grafu G, pokud existuje.
- Pokud vrcholové pokrytí velikosti nejvýš k v grafu G neexistuje, ohlásí HledejVP(G, k) NEEXISTUJE.
- Funkce HledejVP(G, k) pracuje v čase $O(2^k \cdot n)$.
- Z toho plyne, že Vrcholové pokrytí s parametrem k patří do FPT.

Možné parametrizace VP

Obecné grafy

Parametr	Horní odhad
k	$O(1{,}28^k + kn)$
$n - k$	$\mathrm{W}[1]$-úplné (nejspíš není v FPT)
Stromová šířka w grafu G	$2^w \cdot n$

Rovinné grafy

k	$O(c^{\sqrt{k}} + kn)$, $c \leq 2^{4\sqrt{3}} \approx 121{,}8$

- Je-li G rovinný, má vrcholové pokrytí velikosti nejvýš $\frac{3}{4}n$.
- Můžeme uvážit parametr $\frac{3}{4}n - k$.
- Parametrizace nad/pod zaručenou hodnotu

Vliv funkce f na čas

k	$f(k) = 2^k$	$f(k) = 1{,}32^k$	$f(k) = 1{,}28^k$
10	$\approx 10^3$	≈ 16	≈ 12
20	$\approx 10^6$	≈ 258	≈ 140
30	$\approx 10^9$	≈ 4140	≈ 1650
40	$\approx 10^{12}$	$\approx 6{,}6 \cdot 10^4$	$\approx 2{,}0 \cdot 10^4$
50	$\approx 10^{15}$	$\approx 1{,}1 \cdot 10^6$	$\approx 2{,}3 \cdot 10^5$
75	$\approx 10^{22}$	$\approx 1{,}1 \cdot 10^9$	$\approx 1{,}1 \cdot 10^8$
100	$\approx 10^{30}$	$\approx 1{,}1 \cdot 10^{12}$	$\approx 5{,}3 \cdot 10^{10}$
500	$\approx 10^{130}$	$\approx 1{,}4 \cdot 10^{60}$	$\approx 4{,}1 \cdot 10^{53}$

Možné parametry SAT

Délka klauzule k Nedává smysl, pro $k = 2$ polynomiální, pro $k = 3$ NP-úplné.

Počet proměnných n

- Triviálně řešitelné v čase $O^*(2^n)$.
- Má větší smysl spolu s omezením délky klauzule k, pro $k = 3$ lze lépe než $O^*(1{,}49^n)$ deterministicky a $O^*(1{,}33^n)$ randomizovaně.
- Souvisí se Strong Exponential Time Hypothesis

Počet klauzulí m Lze v čase $O^*(1{,}24^m)$.

Délka formule ℓ Lze v čase $O^*(1{,}08^\ell)$.

Značení: $O^*()$ — až na polynomiální faktor.

Další možné parametry SAT

Váha ohodnocení w

- Hledáme splňující ohodnocení, které má přesně w jedniček.
- $\mathrm{W}[2]$-úplné.
- Použito k definici třídy $\mathrm{W}[2]$.

Maximální deficience formule $\delta^*(\varphi)$ Lze v čase $O(2^{\delta^*(\varphi)} \cdot n^3)$.

Stromová šířka různých grafů asociovaných s formulí Je v FPT, konkrétní složitost závisí na typu grafu, který se uvažuje.

Kernelizace

Kernel čili „těžké jádro“ problému

- Cílem je zredukovat danou instanci parametrizovaného problému na tzv. jádro (kernel), které obsahuje tu těžkou část problému.
- Redukce a úpravy by měly být proveditelné v polynomiálním čase.
- Jádro chceme co nejmenší, s velikostí závisející jen na parametru k.

Bussova redukční pravidla pro VP

1. Odstraň izolované vrcholy z grafu G.
2. Je-li vrchol u stupně 1 a v je jeho jediný soused, pak vlož v do vrcholového pokrytí S a z G odstraň u, v a s nimi incidentní hrany. Sniž hodnotu parametru k o 1.
3. Je-li u vrchol stupně alespoň $k+1$, vlož u do S, odstraň u z G spolu s incidentními hranami a sniž hodnotu parametru k o 1.

Označíme-li $(G' = (V', E'), k')$ dvojici vzniklou z $(G = (V, E), k)$ opakováním uvedených pravidel dokud je to možné, pak

- G má vrcholové pokrytí velikosti k, právě když G' má vrcholové pokrytí velikosti k'.
- Pokud G má vrcholové pokrytí velikosti k, pak $|V'| \leq k'^2 + k'$ a $|E'| \leq k'^2$.
- Graf G' lze vytvořit v čase $O(k \cdot |V|)$.
- (G', k') je jádro kvadratické velikosti.

Jádro problému — definice

Definice 3

Uvažme parametrizovaný problém $L \subseteq \Sigma^* \times \Sigma^*$, tj. L obsahuje dvojice (I, k), kde I je instance a k je parametr. Redukcí na jádro (kernel) míníme náhradu dvojice (I, k) redukovanou instancí (I', k') (zvanou jádro nebo kernel), pro kterou platí, že

$$k' \leq k \text{ a } |I'| \leq g(k),$$

kde $g(k)$ je algoritmicky vyčíslitelná funkce a

$$(I, k) \in L \Leftrightarrow (I', k') \in L.$$

Navíc redukci (I, k) na (I', k') musí být možno provést v polynomiálním čase $T_K(|I|, k)$. Funkci $g(k)$ zveme velikostí jádra (kernelu)

Jádro problému — poznámky

- Jedná se o úpravu dat před vlastním vyhledáváním řešení.
- Redukce jsou často využívány i během prohledávání.
- Parametr k je obvykle hodnotou hledaného řešení.
- Redukční pravidla mohou být

  závislá na parametru pokud je parametrem hodnota řešení, pak musíme dopředu vědět, jak velké řešení hledáme.

  nezávislá na parametru lze je použít vždy nezávisle na hodnotě parametru.

Nezávislá množina v rovinném grafu

Nezávislá množina v rovinném grafu	
Instance	Rovinný graf $G = (V, E)$ a celé číslo $k \geq 0$.
Otázka	Existuje v G nezávislá množina velikosti alespoň k?

Věta 4

Nezávislá množina v rovinném grafu má jádro (vzhledem k parametru k) velikosti $4k$.

Plyne triviálně z věty o čtyřech barvách.

Každý problém má své jádro

Věta 5

Nechť $L \subseteq \Sigma^* \times \Sigma^*$ je rozhodnutelný parametrizovaný problém. Potom L je řešitelný parametrizovaným algoritmem vzhledem k parametru k, právě když existuje redukce L na jádro vzhledem k parametru k.

$\Rightarrow$ Jde o triviální jádro, které není prakticky použitelné.

$\Leftarrow$ Na jádro lze použít „hrubou sílu“.

MaxSAT

MaxSAT	
Instance	Booleovská formule φ v KNF s m klauzulemi a celé číslo $k \geq 0$.
Otázka	Existuje ohodnocení v, které splní alespoň k klauzulí φ?

Tvrzení 6

Problém MaxSAT *má jádro (vzhledem k parametru k) velikosti $O(k^2)$, které může být nalezeno v lineárním čase.*

VP — jádro založené na párování

Věta 7 (Nemhauser, Trotter 1975)

Nechť $G = (V, E)$ je graf s $n = |V|$ vrcholy a $m = |E|$ hranami. Potom můžeme v čase $O(\sqrt{n} \cdot m)$ nalézt množiny $C_0 \subseteq V$ a $V_0 \subseteq V$, pro které platí:

1. *Je-li $D \subseteq V_0$ vrcholové pokrytí indukovaného podgrafu $G[V_0]$, pak $C := D \cup C_0$ je vrcholové pokrytí G.*
2. *G má minimální vrcholové pokrytí, které obsahuje C_0.*
3. *Indukovaný podgraf $G[V_0]$ má vrcholové pokrytí velikosti alespoň $|V_0|/2$.*

Vytvoření C_0 a V_0

Algoritmus 2 Vytvoření C_0 a V_0

Vstup: $G = (V, E)$

Výstup: C_0 a V_0

1: Vytvoř bipartitní graf $B = (V, V', E_B)$, kde
   - $V' = \{x' \mid x \in V\}$ je kopií V a
   - $E_B := \{\{x, y'\}, \{x', y\} \mid \{x, y\} \in E\}$.

2: Urči C_B jako minimální vrcholové pokrytí B nalezené pomocí algoritmu na hledání maximálního párování v bipartitním grafu v čase $O(\sqrt{n} \cdot m)$.

3: $C_0 := \{x \in V \mid \{x, x'\} \subseteq C_B\}$. ▷ Obě kopie v C_B.

4: $V_0 := \{x \in V \mid |\{x, x'\} \cap C_B| = 1\}$. ▷ Jedna z kopií v C_B.

5: $I_0 := V \setminus (C_0 \cup V_0) = \{x \in V \mid \{x, x'\} \cap C_B = \emptyset\}$. ▷ Žádná z kopií v C_B.

Jádro VP lineární velikosti

Věta 8

Nechť $(G = (V, E), k)$ je instancí VP. V čase $O(k \cdot |V| + k^3)$ lze zkonstruovat redukovanou instanci $(G' = (V', E'), k')$ s $|V'| \leq 2k$ a $k' \leq k$, pro kterou platí, že G má vrcholové pokrytí velikosti k, právě když G' má vrcholové pokrytí velikosti k'.

- Existence jádra s $(2 - \varepsilon) \cdot k$ vrcholy pro nějakou konstantu $\varepsilon > 0$ by znamenalo existenci aproximačního algoritmu pro VP s aproximačním poměrem $(2 - \varepsilon)$.

Celočíselný program pro VP

Minimální vrcholové pokrytí grafu $G = (V, E)$ můžeme najít pomocí následujícího celočíselného programu:

1. Proměnné: $x_v \in \{0, 1\}$ pro každý vrchol $v \in V$.
2. Minimalizuj $\sum_{v \in V} x_v$ za podmínek
3. $x_u + x_v \geq 1$ pro každou hranu $\{u, v\} \in E$.

Protože celočíselné programování je NP-úplné, uvážíme LP relaxaci:

- Místo $x_v \in \{0, 1\}$ použijeme podmínku $0 \leq x_v \leq 1$.

Dolní odhady na velikost jádra

Věta 9

Pokud $\mathrm{P} \neq \mathrm{NP}$*, nemá* VRCHOLOVÉ POKRYTÍ *jádro s* $1{,}36k$ *vrcholy.*

Věta 10

Pokud $\mathrm{P} \neq \mathrm{NP}$*, nemá* VRCHOLOVÉ POKRYTÍ V ROVINNÉM GRAFU *jádro v podobě rovinného grafu s nejvýše* $(\frac{4}{3} - \varepsilon) \cdot k$ *vrcholy pro libovolné* $\varepsilon > 0$*.*

Prohledávání s omezenou hloubkou

Prohledávání s omezenou hloubkou

- Idea: Pro danou instanci (I, k) parametrizovaného problému L se snažíme najít „malou množinu“ případů, na které můžeme instanci rozložit.
- Větvíme podle těchto podpřípadů, hledání podpřípadů a jejich konstrukce by měla být polynomiální.
- V principu rekurzivní.
- Hloubka rekurze je omezena na základě parametru k.
- Rekurzivní volání tvoří prohledávací strom (search tree).
- Zajímá nás velikost prohledávacího stromu — počet rekurzivních volání.

Jednoduché příklady

- Algoritmus 1 pro Vrcholové pokrytí, který konstruuje prohledávací strom velikosti $O(2^k)$.
- Prohledávací strom velikosti $O(6^k)$ pro Nezávislou množina v rovinném grafu:
  1. Najdi v $G = (V, E)$ vrchol v stupně nejvýš 5.
  2. Jeden z vrcholů v $v \cup N(v)$ je v maximální nezávislé množině.
  3. Větvíme na 6 případů, můžeme skončit v hloubce k.

Jednoduchý větvící algoritmus pro 3-SAT

Algoritmus 3 Algoritmus pro 3-SAT

Vstup: Formule v 3KNF $\varphi = C_1 \wedge \cdots \wedge C_m$ na n proměnných.
Výstup: Splňující ohodnocení v nebo NESPLNITELNÁ.

1: **if** $n = 0$ nebo $m = 0$ **then**
2: **return** Prázdné ohodnocení v
3: **end if**
4: Ať $C_1 = (e_1 \vee e_2 \vee e_3)$, kde e_1, e_2, e_3 jsou literály.
5: $\varphi_1 := \varphi[e_1 := 1]$.
6: $\varphi_2 := \varphi[e_1 := 0, e_2 := 1]$.
7: $\varphi_3 := \varphi[e_1 := 0, e_2 := 0, e_3 := 1]$.
8: **if** Jedna z formulí φ_1, φ_2 a φ_3 je splnitelná **then**
9: **return** Vrať odpovídající splňující ohodnocení
10: **else**
11: **return** NESPLNITELNÁ
12: **end if**

Složitost jednoduchého algoritmu pro 3-SAT

Věta 11

Velikost stromu prohledávání (=počet rekurzivních volání) algoritmu 3 je $O(1{,}83929^n)$*. Celkový čas algoritmu 3 je tedy* $O(1{,}83929^n \cdot n)$*.*

- Jak ukázat velikost stromu prohledávání?

Charakteristický polynom

Uvažme algoritmus, který řeší problém velikosti n a provádí rekurzivní volání na instance velikosti $n - d_1, n - d_2, \ldots, n - d_i$

- $(d_1, \ldots, d_i)$ nazveme větvícím vektorem (branching vector) této rekurze.
- Počet listů T_n stromu rekurzivních volání je dána řešením rekurentní rovnice

$$T_n = T_{n-d_1} + T_{n-d_2} + \cdots + T_{n-d_i} .$$

- Této rovnici odpovídá charakteristický polynom

$$z^d - z^{d-d_1} - z^{d-d_2} - \ldots - z^{d-d_i} .$$

Velikost stromu prohledávání

Tvrzení 12

Prohledávací strom s větvícím vektorem $(d_1, \ldots, d_i)$ má velikost $n^{O(1)} \cdot |\alpha|^n$, kde α je kořenem charakteristického polynomu pro tento větvící vektor, kde $d = \max\{d_1, \ldots, d_i\}$.

- Stupeň u polynomu $n^{O(1)}$ je daný násobností kořene α.
- Při prohledávání se může větvící vektor pro jednotlivá rekurzivní volání lišit, uvažujeme nejhorší případ.
- Stačí počítat počet listů stromu, počet vnitřních vrcholů je nejvýš počet listů (větvení je vždy alespoň 2).

Cluster Editing

Definice 13

Graf $G = (V, E)$ jehož každá komponenta souvislosti je úplný podgraf G, nazveme cluster grafem.

Cluster Editing	
Instance	Graf $G = (V, E)$, celé číslo $k \geq 0$.
Otázka	Je možné odebráním nebo přidáním nejvýše k hran upravit graf G na cluster graf G'?

Lemma 14

Graf $G = (V, E)$ je cluster graf, právě když v G není cesta se dvěma hranami jako indukovaný podgraf.

Jednoduchý algoritmus pro Cluster Editing

Algoritmus 4 Jednoduchý algoritmus pro Cluster Editing

Vstup: Graf $G = (V, E)$, $k \geq 0$.

1: **if** G je cluster graf **then**
2: **return** Řešení nalezeno.
3: **end if**
4: **if** $k \leq 0$ **then**
5: **return** Řešení v této větvi prohledávání neexistuje.
6: **end if**
7: Najdi vrcholy $u, v, w \in V$, které indukují cestu délky 2 v G, tj. $\{u, v\}, \{u, w\} \in E$ a $\{v, w\} \notin E$.
8: Volej rekurzivně tento algoritmus pro případy:
9: (B1) $E' := E \setminus \{\{u, v\}\}$ a $k' := k - 1$.
10: (B2) $E' := E \setminus \{\{u, w\}\}$ a $k' := k - 1$.
11: (B3) $E' := E \cup \{\{v, w\}\}$ a $k' := k - 1$.

- Algoritmus 4 vytváří strom prohledávání velikosti $O(3^k)$.

Vylepšený rozbor případů

Při větvení rozlišíme tři případy pro trojici $u, v, w \in V$ z algoritmu 4, tj. $\{u, v\}, \{u, w\} \in E$ a $\{v, w\} \notin E$:

(C1) Jediným společným sousedem vrcholů v a w je u.

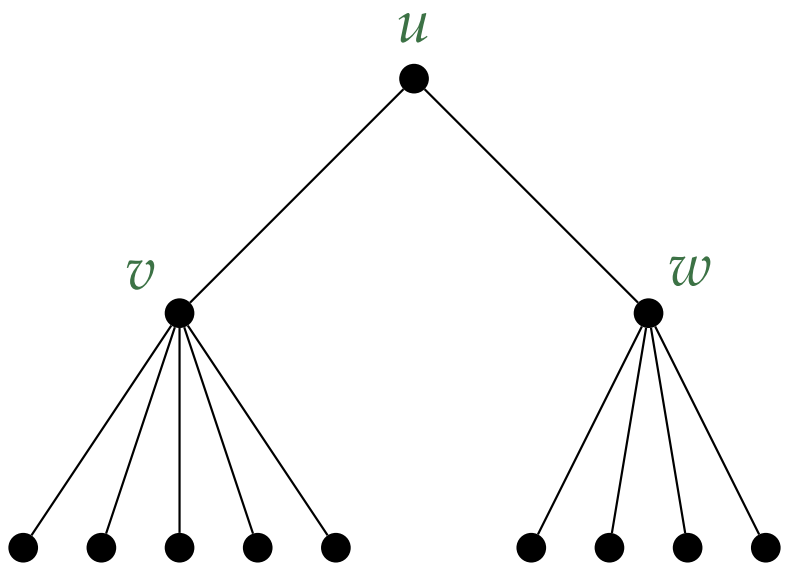
(C2) Vrcholy v a w mají společného souseda $x \neq u$ a $\{u, x\} \in E$.

(C3) Vrcholy v a w mají společného souseda $x \neq u$ a $\{u, x\} \notin E$.

Permanentní hrana Hrana grafu G, která nesmí být odstraněna.

Zakázaná hrana Dvojice vrcholů $\{u, v\} \notin E$, která nesmí být přidána jako hrana.

Případ C1



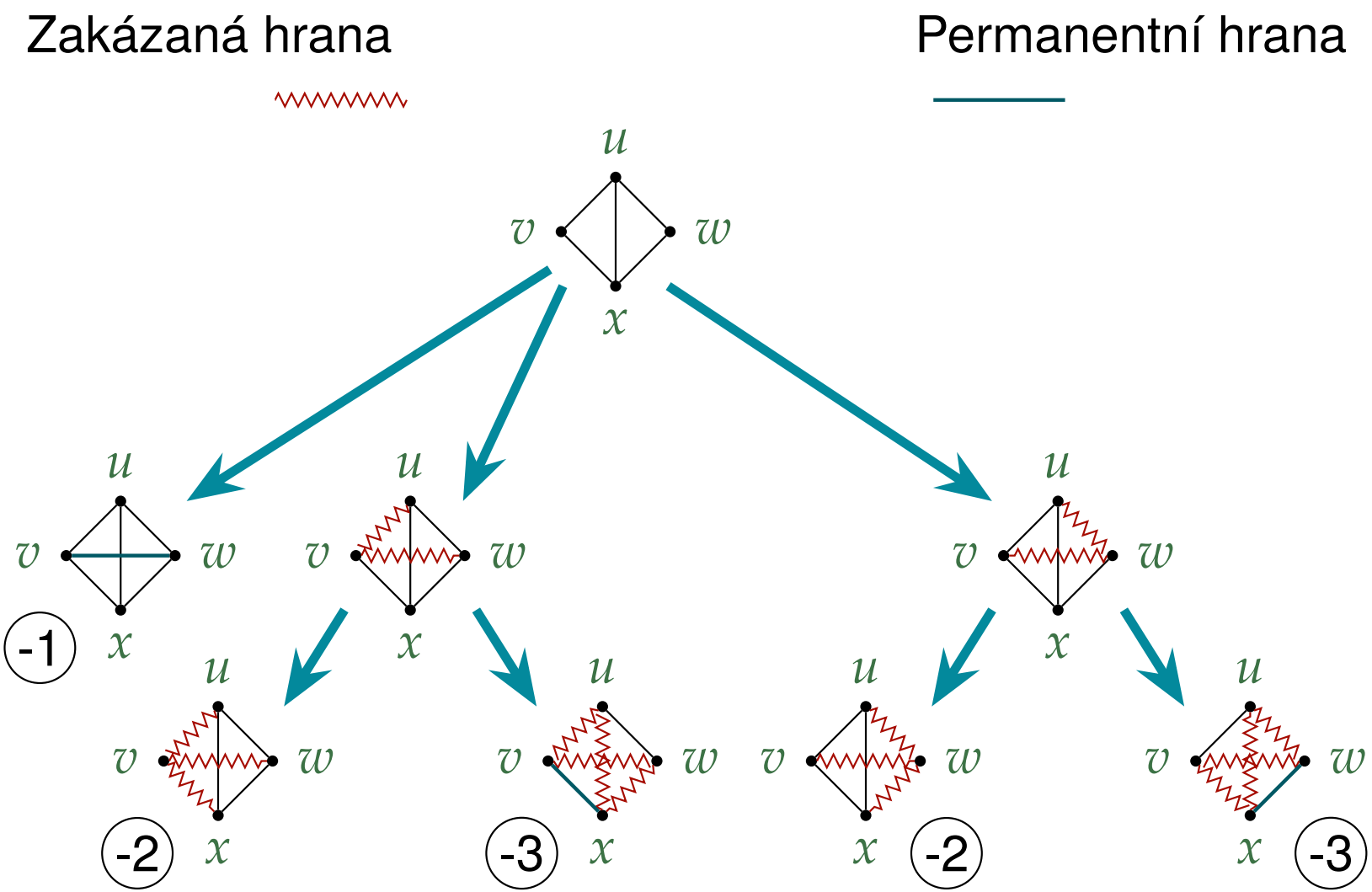
Větvíme do dvou podpřípadů:

(B1) odebrání hrany $\{u, v\}$ a

(B2) odebrání hrany $\{u, w\}$ a

Přidání hrany $\{v, w\}$ (případ (B3)) nemůže dát lepší řešení.

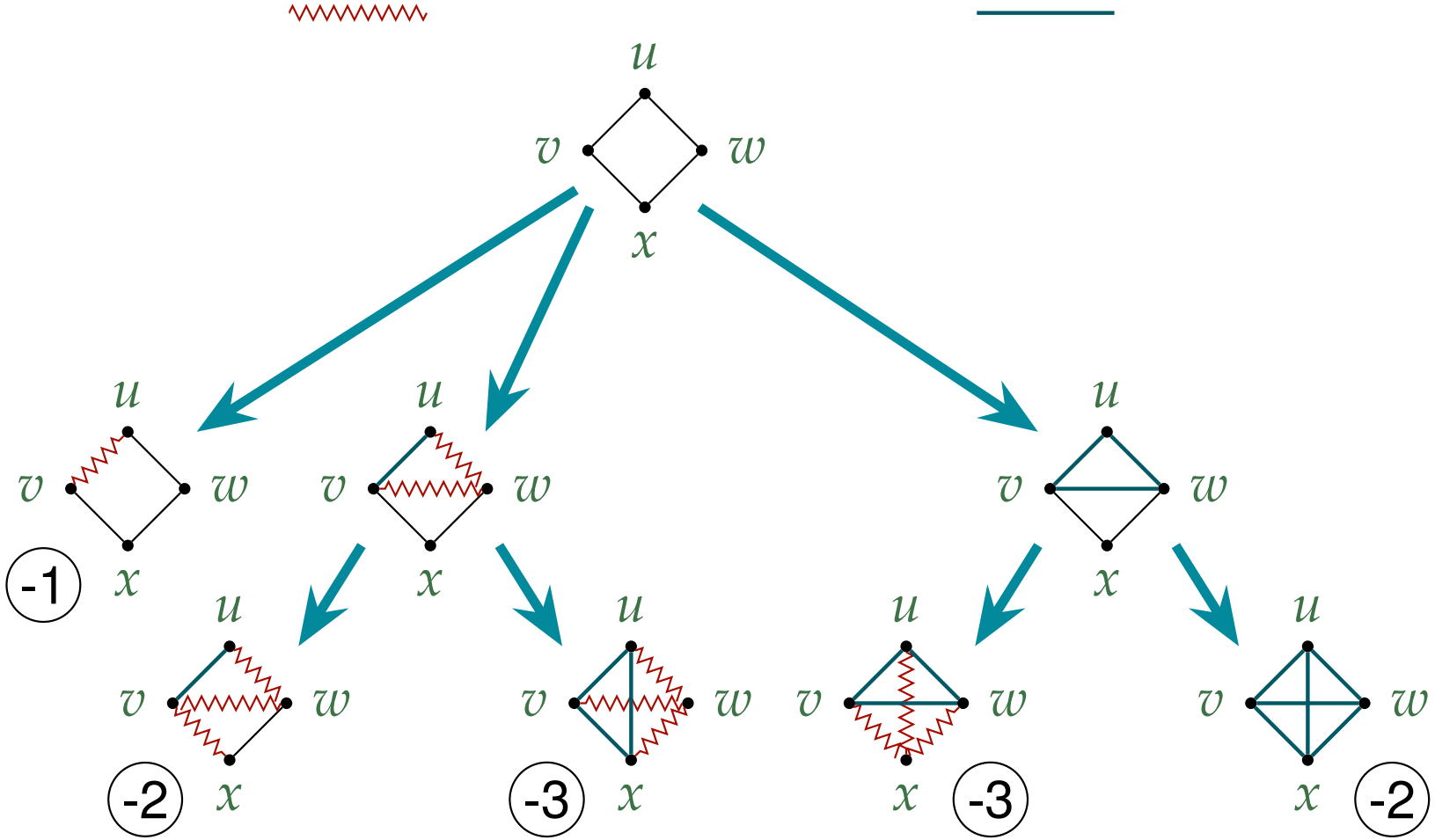
Případ C2



Případ C3

Zakázaná hrana

Permanentní hrana



Shrnutí

Nejhorší větvící vektor je $(1, 2, 2, 3, 3)$, z toho plyne:

Věta 15

Pro Cluster Editing existuje strom prohledávání velikosti $O(2{,}27^k)$.

Nejbližší řetězec

Definice 16

Nechť s, s' jsou dva řetězce délky L nad abecedou Σ. Jejich hammingovská vzdálenost $d_H(s, s')$ je rovna počtu pozic, v nichž se tyto dva řetězce liší.

Nejbližší řetězec (Closest String)	
Instance	Řetězce $s_1, \ldots, s_k$ nad abecedou Σ. Všechny řetězce mají touž délku L. Přirozené číslo d.
Cíl	Najít řetězec s, jehož hammingovská vzdálenost od ostatních řetězců je nejvýš d (jinými slovy pro každé $i = 1, \ldots, k$ platí, že $d_H(s, s_i) \leq d$).

Nejbližší řetězec — vlastnosti

- Řetězce $s_1, \ldots, s_k$ v instanci si uspořádáme do tabulky s k řádky a L sloupci.
- Sloupec nazveme špinavý, pokud obsahuje alespoň dva různé znaky.
- Pokud je špinavých sloupcí více než kd, instance nemá řešení.
- Dále stačí uvažovat špinavé sloupce — máme jádro velikosti kd.
- Pokud existuje dvojice řetězců s_i a s_j s $d_H(s_i, s_j) > 2d$, pak instance nemá řešení.

Algoritmus pro hledání nejbližšího řetězce

Algoritmus 5 Procedura CSD$(s, \Delta d)$

Vstup: Kandidát na řešení s a celé číslo Δd.
Výstup: Řetězec $\hat{s}$ splňující $\max_{i=1,\ldots,k} d_H(\hat{s}, s_i) \leq d$ a $d_H(\hat{s}, s) \leq \Delta d$. Nebo „nenalezeno“.

1: **if** $\Delta d < 0$ **then** ▷ $D0$
2: **return** „nenalezeno“.
3: **end if**
4: **if** $d_H(s, s_i) > d + \Delta d$ pro nějaké $i \in \{1, \ldots, k\}$ **then** ▷ $D1$
5: **return** „nenalezeno“.
6: **end if**
7: **if** $d_H(s, s_i) \leq d$ pro každé $i = 1, \ldots, k$ **then** ▷ $D2$
8: **return** „nenalezeno“.
9: **end if**

▷ Pokračuje na další straně

Procedura CSd (dokončení)

10: Zvol libovolný index i splňující $d_H(s, s_i) > d$ ▷ D3
11: $P \leftarrow \{p \mid s[p] \neq s_i[p]\}$
12: Zvol libovolnou podmnožinu indexů $P' \subseteq P$, $|P'| = d + 1$
13: **for all** $p \in P'$ **do**
14: $s' \leftarrow s$
15: $s'[p] \leftarrow s_i[p]$
16: $s_{ret} \leftarrow \mathrm{CSd}(s', \Delta d - 1)$
17: **if** $s_{ret} \neq$ „nenalezeno“ **then**
18: **return** s_{ret}
19: **end if**
20: **end for**
21: **return** „nenalezeno“ ▷ D4

Algoritmus CSd — vlastnosti

- Voláme $\mathrm{CSd}(s_i, d)$ s libovolným řetězcem $s_i, i \in \{1, \ldots, k\}$.
- Prohledávací strom má velikost $O(d^d)$.
- Dostaneme tak algoritmus pro hledání nejbližšího řetězce, který pracuje v čase $O(kL + kd \cdot d^d)$.

Parametrizovaná složitost

Parametrizovaný problém — přísnější definice

- Nově budeme požadovat, aby hodnota parametru byla funkcí dané instance.
- Parametr je celé číslo.

Definice 17 (Parametrizovaný problém)

Parametrizovaný problém L nad abecedou Σ je množina dvojic (x, k), kde $x \in \Sigma^*$ a k je celé číslo, pro které platí, že

$$(\forall x \in \Sigma^*)(\forall k, k' \in \mathbb{N})\,[((x,k) \in L \wedge (x,k') \in L) \Rightarrow k = k']\ .$$

VRCHOLOVÉ POKRYTÍ parametrem je velikost minimálního vrcholového pokrytí daného grafu.

NEZÁVISLÁ MNOŽINA parametrem je velikost maximální nezávislé množiny daného grafu.

Vrcholové pokrytí

Vrcholové pokrytí (VP)	
Instance	Graf $G = (V, E)$, celé číslo $k \geq 0$.
Otázka	Existuje v G vrcholové pokrytí velikosti přesně k?

Vážené Vrcholové pokrytí (VVP)	
Instance	Graf $G = (V, E)$, ke každému vrcholu $v \in V$ přirozené číslo $w(v)$, celé číslo $k \geq 0$.
Otázka	Existuje v G vrcholové pokrytí se součtem vah rovným k?

Nezávislá množina a klika

Nezávislá množina (NM)	
Instance	Graf $G = (V, E)$, celé číslo $k \geq 0$.
Otázka	Existuje v G nezávislá množina velikosti přesně k?

Klika	
Instance	Graf $G = (V, E)$, celé číslo $k \geq 0$.
Otázka	Existuje v G klika velikosti přesně k?

Dominující množina

Dominující množina	
Instance	Graf $G = (V, E)$, celé číslo $k \geq 0$.
Otázka	Existuje v G dominující množina velikosti přesně k? Přesněji, existuje množina vrcholů $S \subseteq V, \lvert S \rvert = k$, která splňuje, že každý vrchol $v \in V \setminus S$ je spojen hranou s nějakým vrcholem v S?

Vážená splnitelnost

- Třídy parametrizované složitosti jsou definované s pomocí parametrizované převoditelnosti a vážené splnitelnosti.
- Je-li v ohodnocení proměnných, pak jeho váhou je počet proměnných, kterým v přiřazuje hodnotu 1.

VÁŽENÁ (2-)KNF SPLNITELNOST (WEIGHTED (2-)SAT)	
Instance	Formule φ v (2-)KNF, celé číslo $k \geq 0$.
Otázka	Existuje splňující ohodnocení v pro φ s váhou přesně k?

- Nejspíš nepatří do FPT.
- Pokud bychom hledali ohodnocení s váhou nejvýš k, tak tato verze s 2-KNF do FPT patří (tj. je zde rozdíl mezi „přesně“ a „nejvýš“).

Parametrizovaná převoditelnost

Definice 18

Nechť $L, L' \in \Sigma^* \times \mathbb{N}$ jsou dva parametrizované problémy. Řekneme, že L lze převést na L' standardní parametrizovanou převoditelností, pokud existují algoritmicky vyčíslitelné funkce $k \mapsto k'$, $k \mapsto k''$ a $(x,k) \mapsto x'$, pro které platí:

1. $(x,k) \mapsto x'$ lze spočítat v čase $k'' \cdot |(x,k)|^c$ a
2. $(x,k) \in L$ právě když $(x',k') \in L'$.

- Budeme též říkat, že problém L je fpt-převoditelný na problém L', budeme používat značení $L \leq_{\text{fpt}} L'$.
- Parametrizovaná převoditelnost je reflexivní a tranzitivní.
- Pokud L' patří do FPT, pak i L patří do FPT.

Příklady

- Nezávislá množina $\leq_{\text{fpt}}$ Klika.
- Nezávislá množina $\leq_{\text{fpt}}$ Vážená 2-KNF splnitelnost.
- Dominující množina $\leq_{\text{fpt}}$ Vážená KNF splnitelnost.
- Vážené vrcholové pokrytí $\leq_{\text{fpt}}$ Vrcholové pokrytí.
- Vrcholového pokrytí $\leq_{\text{fpt}}$ Dominující množina.
- Řada klasických polynomiálních převodů není fpt, např.
  - Převod Vrcholového pokrytí na Nezávislou množinu.
  - Převod SAT na 3-SAT není parametrizovaným převodem Vážené KNF splnitelnosti na Váženou 3-KNF splnitelnosti.

Třídy $\mathrm{W}[1]$ a $\mathrm{W}[2]$

Definice 19

1. Třída $\mathrm{W}[1]$ obsahuje parametrizované problémy, které jsou fpt-převoditelné na Váženou 2-KNF splnitelnost.
2. Parametrizovaný problém L je $\mathrm{W}[1]$-těžký, pokud je na něj fpt-převoditelná Vážená 2-KNF splnitelnost.
3. Pokud parametrizovaný problém L patří do $\mathrm{W}[1]$ a navíc je $\mathrm{W}[1]$-těžký, pak je $\mathrm{W}[1]$-úplný.
4. Třída $\mathrm{W}[2]$ je definována analogicky s pomocí Vážené KNF splnitelnosti.

- Platí, že $\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2]$.
- Pokud by platilo $\mathrm{W}[1] = \mathrm{FPT}$, pak by bylo možné vyřešit 3-SAT v čase $2^{o(n)} \cdot |\varphi|^{O(1)}$. ($\sim$ Exponential Time Hypothesis)

$\mathrm{W}[1]$-úplné problémy

- Vážená antimonotónní 2-KNF splnitelnost se od Vážené 2-KNF splnitelnosti liší tím, že se v ní vyskytují pouze negativní literály.

Věta 20

Vážená 2-KNF splnitelnost je fpt převoditelná na Váženou antimonotónní 2-KNF splnitelnost. Vážená antimonotónní 2-KNF splnitelnost je tedy $\mathrm{W}[1]$-úplný problém.

Další $\mathrm{W}[1]$-úplné problémy:

- Nezávislá množina
- Klika
- Vážená q-KNF splnitelnost pro libovolné $q \geq 2$.

$\mathrm{W}[2]$-úplné problémy

Příklady $\mathrm{W}[2]$-úplných problémů:

- Dominující množina.
- Hitting set.
- Set Cover.

W = weft obvodu

Definice 21

- Řekneme, že logické hradlo (AND nebo OR) je velké, pokud má více než dva vstupy, pokud má dva vstupy, pak je malé.
- Je-li C obvod, pak pojmem weft obvodu C označujeme maximální počet velkých hradel na jakékoli cestě ze vstupu k výstupu.
- Uvažme formuli φ a odpovídající obvod C_φ, řekneme, že φ je t-normalizovaná, pokud
  1. výstupním hradlem C_φ je velké hradlo AND a
  2. C_φ má weft nejvýš t,
  3. na žádné cestě ze vstupu k výstupu po sobě nenásledují dvě malá hradla téhož typu.

- Například 2-KNF je 1-normalizovaná, obecná KNF je 2-normalizovaná.

Varianty vážené splnitelnosti

Vážená t-normalizovaná splnitelnost vážená splnitelnost t-normalizovaných formulí.

Vážená splnitelnost vstupem je formule, na niž neklademe žádná omezení.

Vážená obvodová splnitelnost vstupem je obvod, na nějž neklademe žádná omezení.

Hierarchie W

Definice 22

1. $\mathrm{W}[t]$, $t \geq 1$ je třídou parametrizovaných problémů, které jsou fpt-převoditelné na Váženou t-normalizovanou splnitelnost.
2. $\mathrm{W}[Sat]$ je třídou parametrizovaných problémů, které jsou fpt-převoditelné na Váženou splnitelnost.
3. $\mathrm{W}[P]$ je třídou parametrizovaných problémů, které jsou fpt-převoditelné na Váženou obvodovou splnitelnost
4. XP obsahuje parametrizovaný problém L, pokud lze v čase $f(k) \cdot |x|^{g(k)}$ rozhodnout, zda $(x, k) \in L$, kde f a g jsou algoritmicky vyčíslitelné funkce.

Věta 23

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

Booleovské formule — definice a značení

Booleovské formule

- Literál je proměnná x (pozitivní literál), nebo negace proměnné $\overline{x}$ (negativní literál).
- Klauzule je disjunkcí literálů, např. $C = (x \vee y \vee \overline{z})$.
- Klauzuli považujeme za množinu literálů, např. $C = \{x, y, \overline{z}\}$.
- Konjunktivně normální forma (KNF) je konjunkcí klauzulí, např. $\varphi = (x \vee y \vee \overline{z})(\overline{x} \vee \overline{y})(y)$.
- KNF považujeme za množinu klauzulí, např. $\varphi = \{(x \vee y \vee \overline{z}), (\overline{x} \vee \overline{y}), (y)\}$.
- $\mathrm{var}(\varphi)$ označuje množinu proměnných formule φ.
- $\overline{C}$ označuje klauzuli s literály opačné polarity ke klauzuli C, tj. $\overline{C} = \{\overline{x} \mid x \in C\} \cup \{x \mid \overline{x} \in C\}$.
- Klauzule C_1 a C_2 mají konflikt v proměnné x pokud $x \in \overline{C_1} \cap C_2$ nebo $x \in C_1 \cap \overline{C_2}$.

Ohodnocení proměnných

- Ohodnocení $v : \mathrm{var}(\varphi) \mapsto \{0, 1\}$ je splňující ohodnocení čili model formule φ, pokud $\varphi(v) = 1$.
- Množinu modelů formule φ označíme $T(\varphi)$.
- Ohodnocení $v : \mathrm{var}(\varphi) \mapsto \{0, 1\}$ je nesplňující ohodnocení čili falsepoint formule φ, pokud $\varphi(v) = 0$.
- Množinu falsepointů formule φ pomocí $F(\varphi)$.
- Je-li $B \subseteq \mathrm{var}(\varphi)$, pak $v : B \mapsto \{0, 1\}$ je částečné ohodnocení.
- Pomocí $\varphi[v]$ označíme formuli vzniklou z φ aplikací částečného ohodnocení v — odstraníme splněné klauzule a nesplněné literály.
- Prázdná klauzule $\bot$ není splněna žádným ohodnocením.
- Prázdná KNF $\top$ je splněna každým ohodnocením.

Operace s KNF

- Je-li φ KNF a $B \subseteq \mathrm{var}(\varphi)$, pak $\varphi - B$ označuje formuli vzniklou odstraněním všech literálů s proměnnými z B, tj.:

$$\varphi - B = \{C \setminus (B \cup \overline{B}) \mid C \in \varphi\}$$

Parametrizace splnitelnosti

Parametrizace splnitelnosti

- Parametrem splnitelnosti rozumíme nějakou algoritmicky vyčíslitelnou funkci π, která každé formuli φ přiřazuje nezáporné číslo $\pi(\varphi)$.
- Předpokládáme, že $\pi(\varphi) = \pi(\varphi')$, pokud jsou formule φ a φ' izomorfní (tj. liší se jen přejmenováním proměnných).

SAT(π)	
Instance	Formule φ a číslo $k \geq 0$, pro které platí $\pi(\varphi) \leq k$.
Otázka	Je φ splnitelná?

Ověření parametru

- SAT(π) je tzv. promise problém — slibujeme řešiteli, že daná formule má hodnotu parametru omezenou k.
- Ověření toho, zda je tomu opravdu tak, je samostatným problémem.

VER(π)	
Instance	Formule φ a číslo $k \geq 0$
Otázka	Je $\pi(\varphi) \leq k$?

Backdoor množiny

Backdoor množina

Definice 24

Nechť C je třída KNF, φ je KNF a $B \subseteq \mathrm{var}(\varphi)$. Potom B je

silná C-backdoor množina, pokud pro každé ohodnocení $v : B \mapsto \{0,1\}$ platí, že $\varphi[v] \in C$. Velikost nejmenší silné C-backdoor množiny označíme $\mathbf{b}_C(\varphi)$.

slabá C-backdoor množina, pokud existuje ohodnocení $v : B \mapsto \{0,1\}$ takové, že $\varphi[v] \in C$ a navíc je splnitelná. Velikost nejmenší slabé C-backdoor množiny označíme $\mathbf{wb}_C(\varphi)$.

výmazová C-backdoor množina (deletion C-backdoor set), pokud po odstranění všech literálů s proměnnými z B nová formule $\varphi - B$ patří do C. Velikost nejmenší výmazové C-backdoor množiny označíme $\mathbf{db}_C(\varphi)$.

Základní třída a splnitelnost

Obvyklé požadavky na základní třídu $\mathcal{C}$:

1. $\mathcal{C}$ je uzavřená na izomorfismus (=přejmenování proměnných).
2. Rozhodnutí, zda KNF $\varphi \in \mathcal{C}$ lze učinit v polynomiálním čase.
3. Splnitelnost formule KNF $\varphi \in \mathcal{C}$ lze rozhodnout v polynomiálním čase.
4. Někdy navíc: *$\mathcal{C}$ je seberedukovatelná, tj. pokud $\varphi \in \mathcal{C}$ a x je proměnná φ, pak $\varphi[x := 0] \in \mathcal{C}$ i $\varphi[x := 1] \in \mathcal{C}$.*

Pozorování 25

Pokud $\mathcal{C}$ splňuje požadavky 1-3, potom jsou problémy SAT($\mathbf{b}_\mathcal{C}$) a SAT($\mathbf{wb}_\mathcal{C}$) v FPT.

Hornovské a 2-KNF formule

Definice 26

- Klauzule je hornovská, obsahuje-li nejvýš jeden pozitivní literál.
- KNF φ je hornovská, skládá-li se z hornovských klauzulí.
- Booleovská funkce f je hornovská, pokud ji lze reprezentovat nějakou hornovskou KNF.
- Třídu hornovských KNF označíme Horn.
- KNF φ je 2-KNF pokud se skládá z klauzulí, které obsahují nejvýš dva literály.
- Třídu 2-KNF označíme pomocí Krom.

Věta 27

Splnitelnost hornovských a 2-KNF formulí je možné rozhodnout v lineárním čase vzhledem k délce formule.

Slabé Horn a Krom-backdoor množiny

Věta 28 (Nishimura et al. (2004))

Problémy VER($\mathbf{wb}_{\mathrm{Horn}}$) a VER($\mathbf{wb}_{\mathrm{Krom}}$) jsou $\mathrm{W}[2]$*-těžké.*

Výmazové a silné backdoor množiny

Definice 29

Řekneme, že třída KNF $\mathcal{C}$ je uzavřená na odebírání klauzulí (*clause induced*), pokud pro každou KNF $\varphi \in \mathcal{C}$ a její pod-KNF $\varphi' \subseteq \varphi$ platí, že $\varphi' \in \mathcal{C}$.

Třídy Horn i Krom jsou uzavřené na odebírání klauzulí.

Lemma 30

Nechť $\mathcal{C}$ je třída KNF uzavřená na odebírání klauzulí a nechť $\varphi \in \mathcal{C}$ je KNF z třídy $\mathcal{C}$, potom každá výmazová $\mathcal{C}$-backdoor množina B pro φ je současně silná $\mathcal{C}$-backdoor množina pro φ.

Hornovské a 2-KNF výmazové backdoor množiny

Lemma 31 (Crama et al. (1997); Nishimura et al. (2004))

Je-li $\mathcal{C}$ třída Horn nebo Krom, pak pro libovolnou formuli $\varphi \in \mathcal{C}$ a množinu proměnných B platí, že B je výmazová $\mathcal{C}$-backdoor množina pro φ, právě když B je silná $\mathcal{C}$-backdoor množina pro φ.

Silné Hornovské-backdoor množiny

S KNF φ asociujeme graf $G_\varphi(V, E)$, kde

$$\begin{aligned} V &= \mathrm{var}(\varphi) \\ E &= \{\{x, y\} \mid (\exists C \in \varphi)\,[\{x, y\} \subseteq C]\} \end{aligned}$$

Nechť $B \subseteq \mathrm{var}(\varphi)$, potom následující tvrzení jsou ekvivalentní:

1. B je silná Horn-backdoor množina pro φ.
2. B je výmazová Horn-backdoor množina pro φ.
3. B je vrcholové pokrytí G_φ.

Věta 32 (Nishimura et al. (2004))

Problémy VER($\mathbf{b}_{\mathrm{Horn}}$) a VER($\mathbf{db}_{\mathrm{Horn}}$) jsou v FPT. *Lze je vyřešit v čase* $O^*(1{,}273^k)$.

Silné 2-KNF-backdoor množiny

S KNF φ asociujeme množinu trojic proměnných $\mathcal{U}_\varphi$:

$$\mathcal{U}_\varphi = \{\{x, y, z\} \mid (\exists C \in \varphi)\ [\{x, y, z\} \subseteq \mathrm{var}(C)]\}$$

Nechť $B \subseteq \mathrm{var}(\varphi)$, potom následující tvrzení jsou ekvivalentní:

1. B je silná Krom-backdoor množina pro φ.
2. B je výmazová Krom-backdoor množina pro φ.
3. B je hitting set $\mathcal{U}_\varphi$.

Věta 33 (Nishimura et al. (2004))

Problémy VER($\mathbf{b}_{\mathrm{Krom}}$) a VER$\mathbf{db}_{\mathrm{Krom}}$ jsou v FPT*. Lze je vyřešit v čase* $O^*(2{,}270^k)$.

Backdoor množiny a #SAT

Úlohu #SAT lze parametrizovat parametrem π:

#SAT(π)	
Instance	Formule φ a číslo $k \geq 0$, pro které platí $\pi(\varphi) \leq k$.
Cíl	Určit $\lvert T(\varphi)\rvert$ (tj. počet modelů φ).

Je-li φ KNF a B je silná backdoor množina, pak

$$\lvert T(\varphi)\rvert = \sum_{\tau : B \mapsto \{0,1\}} \lvert T(\varphi[\tau])\rvert.$$

Hitting a cluster formule

Definice 34

- Řekneme, že KNF φ je hitting formule, pokud každé dvě klauzule $C_1, C_2 \in \varphi$ mají konflikt v nějaké proměnné.
- Třídu hitting formulí označíme HIT.
- Řekneme, že KNF φ je cluster formule, pokud jde o konjunkci několika hitting formulí, které mají po dvou disjunktní množiny proměnných.
- Třídu cluster formulí označíme CLU.

#SAT hitting a cluster formulí

Lemma 35

Je-li φ hitting formule na n proměnných, pak

$$|T(\varphi)| = 2^n - \sum_{C \in \varphi} 2^{n-|C|}.$$

Lemma 36

#SAT je pro cluster formule řešitelný v polynomiálním čase.

CLU-backdoor množiny

- Třída CLU je uzavřená na odebírání klauzulí, proto pro každou formuli $\varphi \in \mathrm{CLU}$ platí $\mathbf{b}_{\mathrm{CLU}}(\varphi) \leq \mathbf{db}_{\mathrm{CLU}}(\varphi)$.
- Na rozdíl od hornovských a 2KNF formulí existují formule $\varphi \in \mathrm{CLU}$, pro které platí $\mathbf{b}_{\mathrm{CLU}}(\varphi) < \mathbf{db}_{\mathrm{CLU}}$.
- Silné CLU-backdoor množiny (nejspíš) nelze hledat parametrizovaným algoritmem přímo, protože

Věta 37 (Nishimura et al. (2007))

Problém VER($\mathbf{b}_{\mathrm{CLU}}$) je $\mathrm{W}[2]$-těžký.

- Zavedeme jiný parametr $\mathbf{clu}(\varphi)$ (clusterová šířka formule), pro který bude platit

$$\mathbf{b}_{\mathrm{CLU}}(\varphi) \leq \mathbf{clu}(\varphi) \leq \mathbf{db}_{\mathrm{CLU}}(\varphi).$$

Obstrukce

Definice 38

Překryvová obstrukce dvojice klauzulí $\{C_1, C_2\}$, které mají neprázdný průnik, ale nemají konflikt. S touto obstrukcí asociujeme výmazovou dvojici

$$\{\mathrm{var}(C_1 \cap C_2), \mathrm{var}(C_1 \triangle C_2)\}.$$

Konfliktová obstrukce trojice klauzulí $\{C_1, C_2, C_3\}$, kde C_2 má konflikt s C_1 i C_3, ale C_1 a C_3 konflikt nemají. S touto obstrukcí asociujeme výmazovou dvojici

$$\{\mathrm{var}((C_1 \setminus C_3) \cap \overline{C_2}), \mathrm{var}((C_3 \setminus C_1) \cap \overline{C_2})\}.$$

Vlastnosti obstrukcí

Lemma 39

1. *KNF φ je cluster formule, právě když neobsahuje překryvovou ani konfliktovou obstrukci.*
2. *Nechť φ je KNF a $B \subseteq \mathrm{var}(\varphi)$. Pokud $\varphi - B$ je cluster formule, pak pro každou výmazovou dvojici $\{X, Y\}$ asociovanou s nějakou obstrukcí v φ platí, že $X \subseteq B$ nebo $Y \subseteq B$.*

Obstrukční graf

Definice 40

S KNF φ asociujeme obstrukční graf $G_\varphi = (V, E)$, kde

- $V = \mathrm{var}(\varphi)$ a
- $\{x, y\} \in E$ pokud ve φ je výmazová dvojice $\{X, Y\}$, kde $x \in X$ a $y \in Y$.

$\mathbf{clu}(\varphi) =$ velikost minimálního vrcholového pokrytí G_φ.

Věta 41

- *Pro každou KNF φ platí, že*

$$\mathbf{b}_{\mathrm{CLU}}(\varphi) \leq \mathbf{clu}(\varphi) \leq \mathbf{db}_{\mathrm{CLU}}(\varphi).$$

- *Problém #SAT($\mathbf{clu}$) je v* FPT.

Další třídy formulí

UP	obsahuje formule, které je možné rozhodnout s pomocí jednotkové propagace.
PL	obsahuje formule, které lze rozhodnout s pomocí eliminace čistých literálů („pure literal“).
UP + PL	obsahuje formule, které je možné rozhodnout s kombinací jednotkové propagace a eliminace čistých literálů. *Základní třída pro DPLL.*
RHorn	skrytě hornovské formule.
q-Horn	q-Hornovské formule.
Matched	matched formule.
$\mathcal{C}^{\{\}}$	třída $\mathcal{C}$ s přidanou detekcí prázdné klauzule, tj. $\mathcal{C}^{\{\}} = \mathcal{C} \cup \mathcal{E}$, kde $\mathcal{E}$ je třída formulí obsahujících prázdnou klauzuli.

Backdoor množiny k dalším třídám formulí

Třída	VER($\mathbf{b}_C$)	VER($\mathbf{db}_C$)
UP	W$[P]$-úplné	nemá význam
PL	W$[P]$-úplné	nemá význam
UP + PL	W$[P]$-úplné	nemá význam
RHorn	W$[1]$-těžké	FPT
q-Horn	? (nejspíš těžké)	FPT
Matched	W$[2]$-těžké	W$[2]$-těžké
$\mathcal{C}^{\{\}}$	W$[1]$-těžké pro zmíněné třídy	nemá význam

Stromová šířka

Grafy asociované s KNF

S KNF φ můžeme asociovat např.

Primární graf $G(\varphi) = (V, E)$, kde $V = \mathrm{var}(\varphi)$ a

$$E = \{\{x, y\} \mid (\exists C \in \varphi)\,[\{x, y\} \subseteq \mathrm{var}(C)]\}$$

Duální graf $G^d(\varphi) = (V, E)$, kde $V = \varphi$ (tj. klauzule) a

$$E = \{\{C_1, C_2\} \mid (\exists x \in \mathrm{var}(\phi))\,[x \in \mathrm{var}(C_1) \cap \mathrm{var}(C_2)]\}$$

Incidenční graf $I(\varphi) = (V_1, V_2, E)$, kde $V_1 = \mathrm{var}(\varphi)$, $V_2 = \varphi$ a

$$E = \{\{x, C\} \mid x \in \mathrm{var}(C)\}$$

Stromová šířka

Definice 42

Nechť $G = (V, E)$ je graf, stromovou dekompozicí grafu G nazveme dvojici (T, χ), kde $T = (V', E')$ je strom a $\chi : V' \mapsto \mathcal{P}(V)$ je zobrazení, pro které platí:

1. Pro každý vrchol $v \in V$ existuje vrchol stromu $t \in V'$, pro nějž $v \in \chi(t)$,
2. pro každou hranu $\{u, v\} \in E$ existuje vrchol stromu $t \in V'$, pro nějž $\{u, v\} \subseteq \chi(t)$, a
3. pro každé tři vrcholy stromu $t_1, t_2, t_3 \in V'$ platí, že pokud t_1 leží na cestě mezi t_2 a t_3, pak $\chi(t_2) \cap \chi(t_3) \subseteq \chi(t_1)$.

Šířka stromové dekompozice $\max_{t \in V'} |\chi(t)| - 1$.

Stromová šířka $tw(G)$ grafu G minimální šířka stromové dekompozice G.

Stromová šířka jako parametr SAT

Definice 43

Pro formuli φ označíme pomocí

$\mathbf{tw}(\varphi)$ stromovou šířku primárního grafu $G(\varphi)$.

$\mathbf{tw}^d(\varphi)$ stromovou šířku duálního grafu $G(\varphi)$.

$\mathbf{tw}^*(\varphi)$ stromovou šířku incidenčního grafu $I(G)$.

Platí:

- Pro každou formuli φ je $\mathbf{tw}^*(\varphi) \leq \mathbf{tw}(\varphi) - 1$
- Pro každou formuli φ je $\mathbf{tw}^*(\varphi) \leq \mathbf{tw^d}(\varphi) - 1$
- Existují třídy formulí, které mají stromovou šířku incidenčního grafu 1 a libovolně velkou stromovou šířku primárního nebo duálního grafu.

Parametrizace SAT stromovou šířkou

Věta 44 (Samer and Szeider (2010))

Splnitelnost parametrizovanou stromovou šířkou lze vyřešit v časech daných tabulkou:

#SAT(**tw**)	*#SAT*(**tw**$^{\mathbf{d}}$)	*#SAT*(**tw***)
$O(2^k kdN)$	$O(2^k klN)$	$O(2^k k(l+2^k)N)$

- n *počet proměnných KNF* φ,
- m *počet klauzulí,*
- d *maximální počet výskytů nějaké proměnné v* φ,
- l *počet literálů v nejdelší klauzuli,*
- k *hodnota parametru (stromové šířky),*
- N *počet vrcholů odpovídající stromové dekompozice.*

Hezká stromová dekompozice

Definice 45

Trojice (T, χ, r) je hezkou stromovou dekompozicí grafu $G = (V, E)$, pokud:

1. (T, χ) je stromovou dekompozicí G.
2. T je zakořeněný strom s kořenem r.
3. Každý vnitřní vrchol $t \in V'$ má nejvýš dva syny a má jeden z následujících typů:

   Spojovací vrchol (join node) t má dva syny t_1 a t_2, kde $\chi(t) = \chi(t_1) = \chi(t_2)$.

   Uváděcí vrchol (introduce node) t má jednoho syna t' a $\chi(t) = \chi(t') \cup \{x\}$ pro nějaký vrchol $x \in V$.

   Zapomínací vrchol (forget node) t má jednoho syna t' a $\chi(t) = \chi(t') \setminus \{x\}$ pro nějaký vrchol $x \in V$.

Konstrukce stromové dekompozice

- Stromovou dekompozici s minimální šířkou lze určit v čase $O^*(1{,}8899^n)$ — Fomin et al. (2008); Villanger (2006).
- Určení stromové šířky je v FPT — Bodlaender (1996) (použitelný jen pro velmi malá k).
- Aproximační algoritmy (nejlepší známý poměr $O(\log k)$).
- FPT aproximační algoritmus s poměrem 4.
- Heuristiky a speciální třídy grafů.
- Každou stromovou dekompozici lze přetvořit do hezké stromové dekompozice bez nárůstu šířky.

Algoritmus pro #SAT($\mathbf{tw}$)

- Vycházíme z hezké stromové dekompozice $(T = (V', E'), \chi, r)$ primárního grafu $G(\varphi)$ formule φ.
- $V_t = \bigcup_{t' \in V(T_t)} \chi(t')$, kde T_t je podstrom T s kořenem t.
- Pro vrchol $t \in V'$ a ohodnocení $\alpha : \chi(t) \mapsto \{0, 1\}$ definujeme $N(t, \alpha)$ jako množinu ohodnocení $\tau : V_t \mapsto \{0, 1\}$, pro která platí:
  1. $\tau(x) = \alpha(x)$ pro všechny proměnné $x \in \chi(t)$, a
  2. τ nefalzifikuje žádnou klauzuli z φ.
- $n(t, \alpha) = |N(t, \alpha)|$.
- $|T(\varphi)| = \sum_{\alpha : \chi(r) \mapsto \{0,1\}} n(r, \alpha)$.
- Hodnoty $n(t, \alpha)$ reprezentujeme tabulkou M_t s $|\chi(t)| + 1$ sloupci (proměnné a hodnota) a $2^{|\chi(t)|}$ řádky (pro každé α).
- Řádky s $n(t, \alpha) = 0$ můžeme vynechávat.

Určení tabulky M_t (pro případ #SAT(tw))

1. t je list:

$$n(t,\alpha) = \begin{cases} 0 & \text{pokud } C(\alpha) = 0 \text{ pro nějakou klauzuli } C \in \varphi \\ 1 & \text{jinak} \end{cases}$$

2. t je spojovací uzel se dvěma syny t_1 a t_2:

$$n(t,\alpha) = n(t_1,\alpha) \cdot n(t_2,\alpha)$$

3. t je uváděcí uzel se synem t':

$$n(t,\alpha) = \begin{cases} 0 & C(\alpha) = 0 \text{ pro nějakou } C \in \varphi \\ n(t', \alpha \restriction_{\chi(t')}) & \text{jinak} \end{cases}$$

4. t je zapomínací uzel se synem t' a $\chi(t) = \chi(t') \setminus \{x\}$:

$$n(t,\alpha) = n(t', \alpha \cup \{(x,0)\}) + n(t', \alpha \cup \{(x,1)\})$$

Složitost #SAT(tw)

Věta 46 (Samer and Szeider (2010))

Nechť φ je KNF formule, $T = (V', E')$ je stromová dekompozice primárního grafu $G(\varphi)$ šířky k s N vrcholy, potom

1. *Je-li $t \in V'$ uzel stromové dekompozice T formule φ, a máme-li již určeny tabulky $M_{t'}$ pro všechny syny t, pak tabulku M_t lze určit v čase $O(2^k kd)$.*
2. *$|T(\varphi)|$ lze určit v čase $O(2^k kdN)$.*

Algoritmus pro #SAT($\mathbf{tw}^*$)

- Vycházíme z hezké stromové dekompozice $(T = (V', E'), \chi, r)$ primárního grafu $G(\varphi)$ formule φ.
- $V_t = \bigcup_{t' \in V(T_t)} \chi(t')$, kde T_t je podstrom T s kořenem t.
- $X_t = V_t \cap \mathrm{var}(\varphi)$ (tj. proměnné ve V_t)
- $\varphi_t = V_t \cap \varphi$ (tj. klauzule ve V_t)
- $\chi_v(t) = \chi(t) \cap \mathrm{var}(\varphi)$ (tj. proměnné v $\chi(t)$)
- $\chi_c(t) = \chi(t) \cap \varphi$ (tj. klauzule v $\chi(t)$)
- Pro uzel stromu $t \in V'$, ohodnocení $\alpha : \chi_v(t) \mapsto \{0, 1\}$ a množinu klauzulí $A \subseteq \chi_c(t)$ definujeme množinu $N(t, \alpha, A)$ ohodnocení $\tau : X_t \mapsto \{0, 1\}$, která splňují
  1. $\tau(x) = \alpha(x)$ pro každé $x \in \chi_v(t)$ a
  2. $A = \{C \in \varphi_t \mid C(\tau) \neq 1\}$.
- $n(t, \alpha, A) = |N(t, \alpha, A)|$
- $|T(\varphi)| = \sum_{\alpha : \chi_v(r) \mapsto \{0,1\}} n(r, \alpha, \emptyset)$

Tabulka M_t

Hodnoty $n(t, \alpha, A)$ reprezentujeme tabulkou M_t s $|\chi(t)| + 1$ sloupci a $2^{|\chi(t)|+1}$ řádky, kde

- $|\chi_v(t)|$ sloupců odpovídá proměnným a ohodnocení α,
- $|\chi_c(t)|$ sloupců odpovídá klauzulím a množině A,
- poslední sloupec obsahuje hodnotu $n(t, \alpha, A)$.

1. Je-li t list, pak

$$n(t, \alpha, A) = \begin{cases} 1 & A = \{C \in \chi_c(t) \mid C(\alpha) \neq 1\} \\ 0 & \text{jinak} \end{cases}$$

2. Je-li t spojovací uzel se syny t_1, t_2, pak

$$n(t, \alpha, A) = \sum_{\substack{A_1, A_2 \subseteq \chi_c(t) \\ A_1 \cap A_2 = A}} n(t_1, \alpha, A_1) \cdot n(t_2, \alpha, A_2)$$

Určení M_t pro zaváděcí uzel (proměnná)

3. Je-li t uváděcí uzel se synem t' a $\chi(t) = \chi(t') \cup \{x\}$ pro proměnnou $x \in \mathrm{var}(\varphi)$, pak

$$n(t, \alpha \cup \{(x,0)\}, A) = \begin{cases} 0 & (\exists C \in A)[\overline{x} \in C] \\ \sum_{B' \subseteq B} n(t', \alpha, A \cup B') & \text{jinak, kde } B = \{C \in \chi_c(t) \mid \overline{x} \in C\} \end{cases}$$

$$n(t, \alpha \cup \{(x,1)\}, A) = \begin{cases} 0 & (\exists C \in A)[x \in C] \\ \sum_{B' \subseteq B} n(t', \alpha, A \cup B') & \text{jinak, kde } B = \{C \in \chi_c(t) \mid x \in C\} \end{cases}$$

Určení M_t pro zaváděcí uzel (klauzule)

4. Je-li t uváděcí uzel se synem t' a $\chi(t) = \chi(t') \cup \{C\}$ pro klauzuli $C \in \varphi$, pak

$$n(t, \alpha, A) = \begin{cases} n(t', \alpha, A) & C \notin A \text{ a } C(\alpha) = 1 \\ n(t', \alpha, A \setminus \{C\}) & C \in A \text{ a } C(\alpha) \neq 1 \\ 0 & \text{jinak} \end{cases}$$

Určení M_t pro zapomínací uzel

5. Je-li t zapomínací uzel se synem t' a $\chi(t) = \chi(t') \setminus \{x\}$ pro nějakou proměnnou $x \in \mathrm{var}(\varphi)$, pak

$$n(t, \alpha, A) = n(t', \alpha \cup \{(x, 0)\}, A) + n(t', \alpha \cup \{(x, 1)\}, A)$$

6. Je-li t zapomínací uzel se synem t' a $\chi(t) = \chi(t') \setminus \{C\}$ pro nějakou klauzuli $C \in \varphi$, pak

$$n(t, \alpha, A) = n(t', \alpha, A)$$

Složitost #SAT($\mathbf{tw}^*$)

Věta 47 (Samer and Szeider (2010))

Nechť φ je KNF formule, $T = (V', E')$ je stromová dekompozice incidenčního grafu $I(\varphi)$ šířky k s N vrcholy, potom

1. *Je-li $t \in V'$ uzel stromové dekompozice T formule φ, a máme-li již určeny tabulky $M_{t'}$ pro všechny syny t, pak tabulku M_t lze určit v čase $O(2^k(l + 2^k)k)$.*
2. *$|T(\varphi)|$ lze určit v čase $O(2^k(l + 2^k)kN)$.*

Hypotéza o exponenciálním čase

Hypotéza o exponenciálním čase

Následující hypotézy zavedli Impagliazzo and Paturi (2001):

Exponential Time Hypothesis (ETH)

Existuje konstanta $\varepsilon > 0$, pro kterou platí, že 3 nelze vyřešit v čase $O^*(2^{\varepsilon n})$.

Strong Exponential Time Hypothesis (SETH)

Neexistuje algoritmus pro k-SAT, který pracuje v čase $O^*(2^{\delta n})$, kde $\delta < 1$ je konstanta, jež nezávisí na k.

Algoritmy pro 3-SAT

Randomizované algoritmy	
$1{,}3633^n$	Paturi et al. (1998)
$1{,}33334^n$	Schoning (1999)
$1{,}32373^n$	Iwama and Tamaki (2004)
$1{,}30704^n$	Hertli (2014)
Deterministické algoritmy	
$1{,}5^n$	Dantsin et al. (2002)
$1{,}481^n$	Dantsin et al. (2002)
$1{,}3334^n$	Moser and Scheder (2011)
1.3303^n	Makino et al. (2011)
$1{,}30704^n$	Rolf (2005) (Unique k-SAT)

Schöningův algoritmus pro SAT

Schöningův algoritmus

Náhodná procházka po hyperkrychli

Algoritmus 6 Hledání splňujícího ohodnocení

Vstup: Formule φ v k-KNF s n proměnnými.

```
1:  function HLEDEJ(φ)
2:      Vyber náhodně počáteční ohodnocení α ∈ {0,1}^n.
3:      for i := 1 to 3n do
4:          if φ(α) = 1 then
5:              return α
6:          end if
7:          Vyber klauzuli C ∈ φ, pro kterou C(α) = 0.
8:          Vyber nějakou proměnnou x ∈ var(C).
9:          Polož α(x) := 1 − α(x).
10:     end for
11: end function
```

Vlastnosti náhodné procházky

Věta 48 (Schoning (1999))

Uvažme k-KNF φ na n proměnných a předpokládejme, že je φ splnitelná, pak HLEDEJ(φ) *najde splňující ohodnocení s pravděpodobností $p \geq \left(\frac{k}{2(k-1)}\right)^n$ (až na polynomiální faktor).*

1. Opakujeme-li volání funkce HLEDEJ t-krát, pak pravděpodobnost, že nenajdeme splňující ohodnocení je nejvýš e^{-pt}.
2. Uvážíme-li $t = \frac{20}{p} = O^*\left(\frac{2(k-1)}{k}\right)^n$, pak pravděpodobnost neúspěchu je nejvýš e^{-20}.
3. Pro $k = 3$ dostáváme s tímto počtem opakování čas $O^*(t) = O^*((2 * \frac{2}{3})^n) = O^*(1{,}334^n)$

Derandomizace Schöningova algoritmu

Hammingovská vzdálenost a koule

Definice 49

- Jsou-li $\alpha, \beta \in \{0,1\}^n$ dva booleovské vektory, pak jejich hammingovskou vzdálenost $d_H(\alpha, \beta)$ definujeme jako počet pozic, ve kterých se liší, tedy

$$d_H(\alpha, \beta) = |\{i \in \{1, \ldots, n\} \mid \alpha[i] \neq \beta[i]\}| \ .$$

- Hammingovskou kouli $B_r(\alpha)$ o poloměru r kolem vektoru $\alpha \in \{0,1\}$ definujeme jako množinu vektorů v hammingovské vzdálenosti nejvýš r od α, tedy

$$B_r(\alpha) = \{\beta \in \{0,1\}^n \mid d_H(\alpha, \beta) \leq r\} \ .$$

Lokální prohledávání

Promise-Ball-k-SAT	
Instance	k-KNF φ na n proměnných, ohodnocení proměnných $\alpha \in \{0,1\}^n$ a poloměr $r \in \mathbb{N}$.
Cíl	Pokud existuje splňující ohodnocení $\beta \in B_r(\alpha)$, najdi nějaké splňující ohodnocení formule φ. (Ne nutně v $B_r(\alpha)$.)

Promise-Ball-k-SAT lze vyřešit deterministicky v čase

- $O^*(k^r)$ (Dantsin et al. (2002)),
- $O^*\left((k-1+\varepsilon)^r\right)$ pro každé $\varepsilon > 0$ (Moser and Scheder (2011)).

Objem hamingovské koule

Definice 50

Pomocí $V(n,r)$ označíme objem hammingovské koule na n proměnných s poloměrem r.

- Platí

$$V(n,r) = \sum_{i=0}^{r} \binom{n}{i} .$$

- Pro $\rho = \frac{r}{n}$ platí

$$\frac{1}{\sqrt{8n\rho(1-\rho)}} \cdot 2^{h(\rho)n} \leq V(n,r) \leq 2^{h(\rho)n} ,$$

kde $h(\rho) = -\rho \log_2 \rho - (1-\rho)\log_2(1-\rho)$ je entropická funkce.

Pokrývací kódy

Definice 51

Řekneme, že $C \subseteq \{0,1\}^n$ je pokrývací kód (*covering code*) C délky n s poloměrem r, pokud platí, že

$$(\forall \alpha \in \{0,1\}^n)\,(\exists \beta \in C)\,[d_H(\alpha,\beta) \leq r]\ .$$

$\rho = \frac{r}{n}$ je potom normalizovaný pokrývací poloměr C.

Lemma 52

Pro každé $n \geq 1$ a $0 \leq r \leq n$ existuje pokrývací kód délky n s poloměrem nejvýš r a velikostí

$$|C| \leq \left\lceil n \cdot \frac{2^n}{V(n,r)} \right\rceil ,$$

Hledání pokrývacího kódu

Lemma 53

Nechť $0 < \rho < \frac{1}{2}$ *a nechť* $\beta(n) = \sqrt{n\rho(1-\rho)}$ *a nechť* $n \geq 1$, *potom:*

1. *Existuje pokrývací kód* C *délky* n *s poloměrem nejvýš* ρn *a velikostí nejvýš* $n\beta(n) \cdot 2^{(1-h(\rho))n}$.
2. *Pokrývací kód s poloměrem nejvýš* ρn *a velikostí nejvýš* $n^2\beta(n) \cdot 2^{(1-h(\rho))n}$ *lze najít v čase* $O^*(2^{3n})$.
3. *Je-li* $d \geq 2$ *dělitel čísla* n, *pak existuje polynom* $q_d(n)$ *takový, že pokrývací kód s poloměrem nejvýš* ρn *a velikostí nejvýš* $q_d(n) \cdot 2^{(1-h(\rho))n}$ *lze zkonstruovat v čase nejvýš* $q_d(n) \cdot (2^{3n/d} + 2^{(1-h(\rho))n})$.

Derandomizovaný algoritmus

Lemma 54 (Dantsin et al. (2002))

Pokud máme deterministický algoritmus A, který řeší PROMISE-BALL-k-SAT v čase $O^*(a^r)$, pak existuje deterministický algoritmus B, který řeší k-SAT v čase $O^*\left(\left(\frac{2a}{a+1}\right)^n\right)$.

- Pro k^r dostaneme čas $O^*\left(\left(\frac{2k}{k+1}\right)^n\right)$, pro $k = 3$ jde o $O^*(1{,}5^n)$.
- Pro $(k-1+\varepsilon)^r$ dostaneme čas $O^*\left(\left(\frac{2(k-1)}{k}+\varepsilon\right)^n\right)$, pro $k = 3$ jde o $O^*(1{,}334^n)$.

Algoritmy pro obecnou splnitelnost

Pravděpodobnostní algoritmus pro SAT (1. část)

Algoritmus 7 Algoritmus pro SAT (1. část)

Vstup: Formule $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ v KNF na n proměnných. Počet klauzulí původní formule m_0.

Výstup: Splňující ohodnocení φ, nebo „nevím“.

1: **for** $i := 1$ **to** m **do**
2: **if** $|C_i| > 1 + \log m_0$ **then**
3: $D_i :=$ klauzule na prvních $1 + \log m_0$ literálech z C_i.
4: **else**
5: $D_i := C_i$
6: **end if**
7: **end for**
8: Vyřeš k-SAT s $k = 1 + \log m_0$ a formulí $\psi := \bigwedge_{i=1}^{m} D_i$.

▹ Pokračuje na další straně

Pravděpodobnostní algoritmus pro SAT (2. část)

Algoritmus 8 Algoritmus pro SAT (2. část)

9: **if** bylo nalezeno splňující ohodnocení ψ **then**
10: **return** nalezené ohodnocení.
11: **else if** všechny klauzule φ mají délku nejvýš $1 + \log m_0$ **then**
12: **return** „nevím“.
13: **else**
14: Vyber uniformně náhodně klauzuli D_i velikosti $1 + \log m_0$.
15: $\beta :=$ částečné ohodnocení přiřazující literálům v D_i nulu.
16: Zavolej rekurzivně algoritmus na $\varphi[\beta]$.
17: **end if**

Vlastnosti pravděpodobnostního algoritmu

Věta 55 (Schuler (2005))

Algoritmus 7 pracuje v polynomiálním čase a pokud je formule φ splnitelná KNF, pak algoritmus najde splňující ohodnocení s pravděpodobností alespoň $2^{-n(1-1/(1+\log_2 m))}$.

- Je-li formule φ s n proměnnými a m klauzulemi splnitelná, pak lze splňující ohodnocení φ najít v očekávaném čase $O^*(2^{n(1-1/(1+\log m))})$.
- Pokud platí, že $m \leq n^c$ pro nějakou konstantu $c > 1$, pak lze splňující ohodnocení najít v očekávaném čase $O^*(2^{n(1-1/(1+c\log n))})$.

Algoritmy exponenciální v počtu klauzulí a délce formule

Autarky

Definice 56

Jsou-li φ a ψ formule, pak pomocí $\varphi \equiv_{SAT} \psi$ označíme fakt, že φ je splnitelná, právě když je splnitelná ψ.

Definice 57

Nechť φ je KNF, $B \subseteq \mathrm{var}(\varphi)$ množina proměnných. Částečné ohodnocení $v : B \mapsto \{0, 1\}$ je autarky, pokud splňuje každou klauzuli $C \in \varphi$, pro kterou platí, že $B \cap \mathrm{var}(C) \neq \emptyset$.

Lemma 58

Je-li φ KNF a v autarky, pak

- $\varphi[v] \subseteq \varphi$ *a*
- $\varphi \equiv_{SAT} \varphi[v]$.

Čisté a singulární literály

Definice 59

- Literál x je čistý literál (*pure literal*), pokud se v φ nevyskytuje $\overline{x}$ (zatímco x se v φ vyskytuje).
- Řekneme, že literál x je singulární v KNF φ, pokud se x vyskytuje v právě jedné klauzuli φ, ale není to čistý literál.

- Je-li x čistý literál, pak částečné ohodnocení splňující právě jen x je autarky.

Rezoluce

Definice 60

- Řekneme, že klauzule C_1 a C_2 jsou rezolvovatelné, pokud mají konflikt v jediné proměnné x.
- Jejich rezolventou je klauzule $R(C_1, C_2) = (C_1 \cup C_2) \setminus \{x, \overline{x}\}$.
- Je-li φ formule, pak posloupnost klauzulí $C_1, \ldots, C_p$ je rezolučním odvozením klauzule C_p z φ, pokud pro každé $1 \leq i \leq p$ buď
  - $C_i \in \varphi$, nebo
  - $C_i = R(C_{j_1}, C_{j_2})$ pro nějaké indexy $j_1, j_2 < i$.
- Rezolučnímu odvození prázdné klauzule z φ, říkáme také rezoluční zamítnutí φ (*resolution refutation*).

- KNF φ je nesplnitelná, právě když má rezoluční zamítnutí.

DP rezoluce (Davisova-Putnamova)

Definice 61

Nechť φ je KNF a x je literál φ. Označme

$$\begin{aligned} \varphi_0 &= \{C \in \varphi \mid \{x, \overline{x}\} \cap C = \emptyset\} \\ \varphi_x &= \{C \in \varphi \mid x \in C\} \\ \varphi_{\overline{x}} &= \{C \in \varphi \mid \overline{x} \in C)\} \end{aligned}$$

Pak DP rezolucí φ nazveme KNF

$$\begin{aligned} DP_x(\varphi) &= \varphi_0 \cup \\ &\cup \{R(C_1, C_2) \mid C_1 \in \varphi_x \wedge C_2 \in \varphi_{\overline{x}} \wedge C_1 \cap \overline{C_2} = \{x\}\} \end{aligned}$$

- Platí, že $\varphi \equiv_{SAT} DP_x(\varphi)$.
- Je-li x singulární literál, pak $|DP_x(\varphi)| < |\varphi|$.

Redukční pravidla

Eliminace jednotkových klauzulí Obsahuje-li φ jednotkovou klauzuli a (kde a je literál), pak $\varphi \equiv_{SAT} \varphi[a := 1]$.

Eliminace čistých literálů Je-li x čistý literál ve formuli φ, pak $\varphi \equiv_{SAT} \varphi[x := 1]$

Subsumpce Pokud φ obsahuje klauzule C a D, kde $C \subseteq D$, pak $\varphi \equiv_{SAT} \varphi \setminus \{D\}$.

Rezoluce se subsumpcí Pokud φ obsahuje klauzule C a D, kde $R(C, D) \subseteq D$, potom $\varphi \equiv_{SAT} (\varphi \setminus \{D\}) \cup \{R(C, D)\}$.

Eliminace proměnné rezolucí Je-li a proměnná φ, pak $\varphi \equiv_{SAT} DP_a(\varphi)$.

Blokované klauzule

Definice 62

- Klauzule C je blokovaná vzhledem k literálu a a formuli φ, pokud $a \in C$ a pro každou klauzuli D, kde $\overline{a} \in D$, platí, že $(D \cap \overline{C}) \setminus \{\overline{a}\} \neq \emptyset$.
- Pomocí γ_a označíme částečné ohodnocení, pro které platí, že $\gamma_a(x) = 1$ pro literál x, právě když $x = a$ nebo $(x \vee \overline{a})$ je blokovaná v φ vzhledem k $\overline{a}$.

Lemma 63

- *Je-li C blokovaná v φ vzhledem k literálu a, pak* $\varphi \equiv_{SAT} \varphi \setminus \{C\} \equiv_{SAT} \varphi \cup \{C\}$.
- *Je-li a literál φ, pak φ je splnitelná, právě když jedna z* $\varphi[a := 0]$ *a* $\varphi[\gamma_a]$ *je splnitelná.*

Princip černobílých literálů

- Nechť P je binární relace mezi literály a formulemi taková, že pro každý literál v platí $\neg P(v, \varphi) \vee \neg P(\overline{v}, \varphi)$.
- Předpokládejme, že pro každou klauzuli $C \in \varphi$ platí, že obsahuje-li literál w splňující $P(w, \varphi)$, pak také obsahuje literál b splňující $P(\overline{b}, \varphi)$.
- Označme α částečné ohodnocení splňující právě literály z množiny $\{v \mid P(\overline{v}, \varphi)\}$.

Lemma 64

Platí $\varphi \equiv_{SAT} \varphi[\alpha]$.

Značení velikosti klauzulí a počtu výskytů literálů

i-klauzule obsahuje právě i literálů.

(i, j)-literál se ve formuli vyskytuje i-krát pozitivně a j-krát negativně.

(i^+, j^-)-literál se ve formuli vyskytuje alespoň i-krát pozitivně a nejvýš j-krát negativně.

Další kombinace jsou analogické.

- Pomocí $\tau(d_1, d_2, \ldots, d_i)$ označíme řešení rekurentní rovnice s větvícím vektorem $(d_1, d_2, \ldots, d_i)$. Jde tedy o kořen odpovídajícího charakteristického polynomu.

Algoritmus exponenciální v počtu klauzulí

Algoritmus 9 REDUCE-K

Vstup: Formule φ v KNF.
Výstup: Zredukovaná formule v KNF.

1: **while** Některé z následujících redukcí lze provést **do**
2: Eliminace jednotkových klauzulí. Je-li a jednotková klauzule φ, polož $\varphi := \varphi[a := 1]$.
3: Princip černobílých literálů. Pokud každá klauzule $C \in \varphi$, která obsahuje $(2, 3^+)$-literál, obsahuje též $(3^+, 2)$-literál, pak $\varphi := \varphi[\alpha]$, kde α je částečné ohodnocení splňující právě $(3^+, 2)$-literály.
4: Eliminace proměnné rezolucí. Vyber literál a takový, že a nebo $\overline{a}$ se vyskytuje v φ a $\Delta = |\varphi| - |DP_a(\varphi)|$ je maximální. Pokud je takových literálů víc, vyber takový, jehož počet výskytů je minimální. Pokud $\Delta \geq 0$, pak polož $\varphi := DP_a(\varphi)$.
5: **end while**

Rozdělení na podpřípady (1. část)

Algoritmus 10 SPLIT-K (1. část)

Vstup: Formule φ v KNF.
Výstup: Pokud je φ splnitelná, tak TRUE, jinak FALSE.

1: Prázdná formule. Pokud $\varphi = \emptyset$, vrať TRUE.
2: Prázdná klauzule. Pokud $\emptyset \in \varphi$, vrať FALSE.
3: Dva podpřípady. Pro každý literál a z φ zkonstruuj formule

$$\begin{aligned} \varphi_1 &= \text{REDUCE-K}(\varphi[a := 1]) \\ \varphi_2 &= \text{REDUCE-K}(\varphi[a := 0]) \ . \end{aligned}$$

Pokud pro nějaký literál splňuje, že $\tau(|\varphi| - |\varphi_1|, |\varphi| - |\varphi_2|) \leq \tau(6, 7, 6, 7)$, zavolej rekurzivně funkci SPLIT-K na φ_1 a φ_2.

▹ Pokračuje na další straně

Rozdělení na podpřípady (2. část)

Algoritmus 11 SPLIT-K (2. část)

4: Vyber literál a, který se vyskytuje v φ. Pro každé dva literály b a c, kde b se vyskytuje v $\varphi[a := 1]$ a c se vyskytuje v $\varphi[a := 0]$, zkonstruuj formule

$$\begin{aligned}
\varphi_{11} &= \text{REDUCE-K}(\varphi[a := 1, b := 1]) \\
\varphi_{12} &= \text{REDUCE-K}(\varphi[a := 1, b := 0]) \\
\varphi_{21} &= \text{REDUCE-K}(\varphi[a := 0, c := 1]) \\
\varphi_{22} &= \text{REDUCE-K}(\varphi[a := 0, c := 0]) \ .
\end{aligned}$$

Pokud pro nějaké literály b a c platí, že $\tau(|\varphi| - |\varphi_{11}|, |\varphi| - |\varphi_{12}|, |\varphi| - |\varphi_{21}|, |\varphi| - |\varphi_{22}|) \leq \tau(6, 7, 6, 7)$, rozvětvi se na případy $\varphi_{11}, \varphi_{12}, \varphi_{21}, \varphi_{22}$.

Odhad složitosti SPLIT-K

Věta 65 (Hirsch (2000))

Algoritmus daný voláním SPLIT-K(REDUCE-K(φ)) pracuje korektně a vyžaduje čas $O^*(\tau(6{,}7{,}6{,}7)^{|\varphi|} = O^*(2^{0{,}30897\cdot|\varphi|})$.

Algoritmus exponenciální v délce formule

- Pomocí $||\varphi||$ označíme celkovou délku formule (součet délek klauzulí).
- Pomocí REDUCE-L(φ) označíme redukční proceduru, která provádí:
  1. Eliminaci jednotkových klauzulí.
  2. Eliminaci subsumpce.
  3. Eliminaci blokovaných klauzulí.
  4. Rezoluci se subsumpcí.
  5. Eliminaci proměnné rezolucí (jako v REDUCE-K).

Algoritmus 12 SPLIT-L

Vstup: Formule φ v KNF.
Výstup: Pokud je φ splnitelná, tak TRUE, jinak FALSE.

1: Prázdná formule. Pokud $\varphi = \emptyset$, vrať TRUE.
2: Prázdná klauzule. Pokud $\emptyset \in \varphi$, vrať FALSE.
3: Formule bez binárních klauzulí. Pokud φ neobsahuje klauzule délky 2, použij SPLIT-K(REDUCE-K(φ)).
4: Dva podpřípady. Pro každý literál a z φ zkonstruuj formule

$$\begin{aligned} \varphi_1 &= \text{REDUCE-L}(\varphi[\gamma_a]) \\ \varphi_2 &= \text{REDUCE-L}(\varphi[a := 0]) \ . \end{aligned}$$

Pokud pro nějaký literál splňuje, že $\tau(||\varphi|| - ||\varphi_1||, ||\varphi|| - ||\varphi_2||) \leq \tau(5, 17)$, zavolej rekurzivně funkci SPLIT-L na φ_1 a φ_2.

Odhad složitosti SPLIT-L

Věta 66 (Hirsch (2000))

Algoritmus daný voláním SPLIT-L(REDUCE-L(φ)) pracuje korektně a vyžaduje čas $O^*(\tau(6,7,6,7)^{||\varphi||/3}) = O^*(2^{0,10299 \cdot ||\varphi||})$.

MaxSAT

MaxSAT

MaxSAT	
Instance	Booleovská formule φ v KNF s n proměnnými a m klauzulemi.
Cíl	Najít ohodnocení proměnných α, které maximalizuje počet splněných klauzulí ve formuli φ.

Max2SAT vstupem je 2-KNF.

Weighted MaxSAT každá klauzule má přiřazenou váhu, maximalizujeme součet vah splněných klauzulí.

Aproximace

Aproximační algoritmy

- Pro MaxSAT s faktorem $0{,}7584$ (Goemans and Williamson, 1995).
- Pro Max3SAT s faktorem $\frac{7}{8}$ (Karloff and Zwick, 1997) ($\frac{7}{8} = 0{,}875$).
- Pro Max2SAT s faktorem $0{,}878$ (Goemans and Williamson, 1994).

Neaproximovatelnost

- Max3SAT nelze aproximovat s faktorem $\left(\frac{7}{8} + \varepsilon\right)$ pro jakékoli $\varepsilon > 0$ (Håstad, 2001).
- Max2SAT nelze aproximovat s faktorem lepším než $\frac{21}{22} + \varepsilon$ pro jakékoli $\varepsilon > 0$ (Håstad, 2001).

Možné parametrizace MaxSAT

n	počet proměnných.
m	počet klauzulí.
ℓ	délka formule.
k	počet splněných klauzulí.
$k - \frac{m}{2}$	počet splněných klauzulí nad $m/2$.
$m - k$	počet nesplněných klauzulí.

Varianty a parametry MaxSAT

Parametr	Složitost	Zdroj
k	$O^*(1{,}3695^k)$	Chen and Kanj (2004)
$k' = m - k$	$O(15^{k'} k' m^3)$	Razgon and O'Sullivan (2009)
$k' = k - \frac{m}{2}$	$O(\ell + k'^2 1{,}618^{6k'})$	Mahajan and Raman (1999)
m	$O^*(1{,}3247^m)$	Chen and Kanj (2004)
ℓ	$O^*(1{,}105729^\ell)$	Bansal and Raman (1999)
Max2SAT		
m	$O^*(1{,}1487^m)$	Gramm et al. (2003)
ℓ	$O^*(1{,}0718^\ell)$	Gramm et al. (2003)

Algoritmus pro WEIGHTED MAX2SAT

- Uvažujeme kladné celočíselné váhy klauzulí.
- Cílem je najít takové ohodnocení, které maximalizuje součet vah splněných klauzulí.
- Algoritmus konstruuje strom prohledávání, kde v listech jsou formule jen s jednotkovými klauzulemi.
- Jednotkové klauzule jsou proto nepodstatné.

Značení

(ω, C) klauzule C s vahou $\omega > 0$.

$(\omega, \top)$ speciální vážená klauzule, která je vždy splněná.

$\mathrm{OptVal}(\varphi)$ optimální hodnota pro formuli φ.

$K_2(\varphi)$ součet vah klauzulí délky 2 ve formuli φ.

$\#_l^{(k)}$ součet vah klauzulí délky k, v nichž se vyskytuje literál l.

$\#_l$ součet vah všech klauzulí, v nichž se vyskytuje literál l.

Váha proměnné x součet vah klauzulí délky 2, v nichž se vyskytuje proměnná x.

Operace s váženými formulemi

- Nechť φ, ψ jsou dvě vážené formule, l je literál.
- Pro účely následujících definic uvažujeme $(0, C) \in \varphi$ pro každou klauzuli C.

$$\varphi + \psi = \{(\omega_1 + \omega_2, C) \mid (\omega_1, C) \in \varphi \,\&\, (\omega_2, C) \in \psi \,\&\, \omega_1 + \omega_2 > 0\}$$

$$\varphi - \psi = \{(\omega_1 - \omega_2, C) \mid (\omega_1, C) \in \varphi \,\&\, (\omega_2, C) \in \psi \,\&\, \omega_1 - \omega_2 > 0\}$$

$$\varphi[l] = \left\{(\omega, C) \mid (\omega, C) \in \varphi \,\&\, l, \bar{l} \notin C\right\} + \left\{\left(\omega, C \setminus \{\bar{l}\}\right) \mid \bar{l} \in C \,\&\, |C| > 1\right\} + \left\{(\omega, \top) \mid \omega = \sum_{\substack{(\omega', C) \in \varphi \\ l \in C}} \omega'\right\}$$

Eliminace čistých literálů

- Pokud je a čistý literál v KNF φ, pak

$$\mathrm{OptVal}(\varphi) = \mathrm{OptVal}(\varphi[a]).$$

- Pravidlo $\mathbf{T}_{\mathrm{pure}}$ provede náhradu

$$\varphi \leftarrow \varphi[a].$$

Eliminace jednotkových klauzulí

- Předpokládejme, že ve formuli φ jsou dvě jednotkové klauzule (ω_1, a) a $(\omega_2, \overline{a})$ (kde a je literál).
- Pravidlo $\mathbf{T}_{\text{ann}}$ provede „anihilaci“ těchto klauzulí náhradou

$$\varphi \leftarrow (\varphi - \{(\omega, a), (\omega, \overline{a})\}) + (\omega, \top),$$

kde $\omega = \min(\omega_1, \omega_2)$.

Rezoluce vážených klauzulí

Jsou-li $C = (\omega_1, (l_1 \vee l_2)$ a $D = (\omega_2, (\bar{l}_1 \vee l_3))$ dvě vážené klauzule, pak jejich rezolventu definujeme následujícím způsobem.

- Pokud $l_2 \neq \overline{l_3}$, pak

$$\mathcal{R}(C, D) = (\max(\omega_1, \omega_2), \top), (\min(\omega_1, \omega_2), (l_2 \vee l_3)).$$

- Pokud $l_2 = \overline{l_3}$, pak

$$\mathcal{R}(C, D) = (\omega_1 + \omega_2, \top).$$

Eliminace pomocí rezoluce

Pokud φ obsahuje dvě klauzule $C = (\omega_1, (v \wedge l_1))$ a $D = (\omega_2, (\overline{v} \wedge l_2))$, kde v je proměnná, jež se nevyskytuje v jiných klauzulích φ, pak

$$\mathrm{OptVal}(\varphi) = \mathrm{OptVal}\left((\varphi - \{C, D\}) + \mathcal{R}(C, D)\right).$$

Pravidlo $\mathbf{T}_{\mathrm{DP}}$ provede náhradu

$$\varphi \leftarrow (\varphi - \{C, D\}) + \mathcal{R}(C, D)$$

Dominující jednotkové klauzule

Je-li l literál a φ formule, kde $\#_l^{(1)} \geq \#_{\bar{l}}$, pak

$$\mathrm{OptVal}(\varphi) = \mathrm{OptVal}(\varphi[l]).$$

Pravidlo $\mathbf{T}_{\mathrm{dom}}$ provede náhradu

$$\varphi \leftarrow \varphi[l].$$

Eliminace malých uzavřených podformulí

Uzavřená podformule formule φ je podformule $\psi \subseteq \varphi$, jejíž proměnné se ve φ mimo ψ nevyskytují.

Malá uzavřená podformule má málo proměnných, například 12.

Je-li $\psi \subseteq \varphi$ malá uzavřená podformule formule φ, pak hodnotu $\mathrm{OptVal}(\psi)$ lze najít hrubou silou. Platí

$$\mathrm{OptValue}(\varphi) = \mathrm{OptValue}(\varphi - \psi) + \mathrm{OptValue}(\psi).$$

Pravidlo $\mathbf{T}_{\mathrm{small}}$ provede náhradu

$$\phi \leftarrow (\varphi - \psi) + (\mathrm{OptValue}(\psi), \top).$$

Eliminace vzácných proměnných

Nechť φ je 2-KNF a a je literál, který splňuje

$$\#_a^{(2)} = 2,\ \#_{\overline{a}}^{(2)} = \#_a^{(1)} = 0, \text{ a } \#_{\overline{a}}^{(1)} = 1.$$

Pravidlo $\mathbf{T}_{\mathrm{rare}}$ provede náhradu $\varphi \leftarrow \varphi'$, kde φ' vznikne z φ následujícím způsobem:

- Nechť $(\omega, (a \vee b))$ je binární klauzule obsahující a.
- Ve φ' nahradíme tuto klauzuli klauzulí $(\omega, \top)$, navíc
- nahradíme všechny výskyty literálu a literálem $\overline{b}$ a
- všechny výskyty literálu $\overline{a}$ nahradíme literálem b.

Pak $\mathrm{OptValue}(\varphi) = \mathrm{OptValue}(\varphi')$.

Algoritmus pro MAX2SAT

Algoritmus 13 Algoritmus pro MAX2SAT

Vstup: Vážená 2-KNF φ.
Výstup: $\mathrm{OptVal}(\varphi)$.

1: Opakuj použití pravidel $\mathbf{T}_{\mathrm{pure}}$, $\mathbf{T}_{\mathrm{ann}}$, $\mathbf{T}_{\mathrm{DP}}$, $\mathbf{T}_{\mathrm{dom}}$, $\mathbf{T}_{\mathrm{small}}$, $\mathbf{T}_{\mathrm{rare}}$ na formuli φ dokud je to možné.
2: **if** $\varphi = (\omega, \top)$ **then**
3: **return** ω
4: **else if** $\varphi = \psi_1 \wedge \psi_2$ pro disjunktní uzavřené formule ψ_1, ψ_2 **then**
5: Přidej dvě nové proměnné u, v
6: **return** $\mathrm{OptVal}(\psi_1 + (1, (u \vee v))) + \mathrm{OptVal}(\psi_2 + (1, (u \vee v))) - 2$
7: **else if** φ obsahuje proměnnou v s vahou alespoň 5 **then**
8: **return** $\max(\mathrm{OptVal}(\varphi[v]), \mathrm{OptVal}(\varphi[\overline{v}]))$
9: **else if** všechny proměnné v φ mají váhu 4 **then**
10: Vyber libovolnou proměnnou v.
11: **return** $\max(\mathrm{OptVal}(\varphi[v]), \mathrm{OptVal}(\varphi[\overline{v}]))$

Algoritmus pro MAX2SAT (2. část)

12: **else if** φ obsahuje proměnné s vahami 3 nebo 4 **then**
13: Vyber proměnnou v, pro kterou platí, že po vhodné transformaci $\varphi[v]$ a $\varphi[\overline{v}]$ vzniknou formule ψ_1 a ψ_2, kde $K_2(\varphi) - K_2(\psi_i) \geq 5$ pro $i = 1, 2$ a navíc obě obsahují proměnnou s vahou nejvýš 3.
14: **return** $\max(\mathrm{OptVal}(\psi_1), \mathrm{OptVal}(\psi_2))$
15: **else**
16: Vyber proměnnou v, pro kterou platí, že po vhodné transformaci $\varphi[v]$ a $\varphi[\overline{v}]$ vzniknou formule ψ_1 a ψ_2, kde $K_2(\varphi) - K_2(\psi_i) \geq 5$ pro $i = 1, 2$.
17: **return** $\max(\mathrm{OptVal}(\psi_1), \mathrm{OptVal}(\psi_2))$
18: **end if**

Vlastnosti algoritmu

Věta 67 (Gramm et al. (2003))

Nechť φ je vážená 2-KNF. Potom algoritmus 13 určí hodnotu $\mathrm{OptVal}(\varphi)$ v čase $\mathrm{poly}(|\varphi|) \cdot 2^{K_2(\varphi)/5}$.

Důsledek 68 (Gramm et al. (2003))

Nechť φ je nevážená 2-KNF. Potom algoritmus 13 určí hodnotu $\mathrm{OptVal}(\varphi)$ v čase $\mathrm{poly}(|\varphi|) \cdot 2^{||\varphi||/10}$.

Literatura

Literatura I

Bansal, N. and Raman, V. (1999). Upper bounds for maxsat: Further improved. *Lecture Notes in Computer Science*, pages 247–258.

Bodlaender, H. L. (1996). A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317.

Chen, J. and Kanj, I. A. (2004). Improved exact algorithms for max-sat. *Discrete Applied Mathematics*, 142(1-3):17–27.

Crama, Y., Ekin, O., and Hammer, P. L. (1997). Variable and term removal from boolean formulae. *Discrete Applied Mathematics*, 75(3):217–230.

Dantsin, E., Goerdt, A., Hirsch, E. A., Kannan, R., Kleinberg, J., Papadimitriou, C., Raghavan, P., and Schöning, U. (2002). A deterministic (2- 2/(k+ 1)) n algorithm for k-sat based on local search. *Theoretical Computer Science*, 289(1):69–83.

Literatura II

Fomin, F. V., Kratsch, D., Todinca, I., and Villanger, Y. (2008). Exact algorithms for treewidth and minimum fill-in. *SIAM Journal on Computing*, 38(3):1058–1079.

Goemans, M. X. and Williamson, D. P. (1994). New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(4):656–666.

Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145.

Gramm, J., Hirsch, E. A., Niedermeier, R., and Rossmanith, P. (2003). Worst-case upper bounds for max-2-sat with an application to max-cut. *Discrete Applied Mathematics*, 130(2):139–155.

Håstad, J. (2001). Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859.

Literatura III

Hertli, T. (2014). 3-sat faster and simpler—unique-sat bounds for ppsz hold in general. *SIAM Journal on Computing*, 43(2):718–729.

Hirsch, E. (2000). New worst-case upper bounds for sat. *Journal of Automated Reasoning*, 24(4):397–420.

Impagliazzo, R. and Paturi, R. (2001). On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367 – 375.

Iwama, K. and Tamaki, S. (2004). Improved upper bounds for 3-sat. In *SODA*, volume 4, pages 328–328.

Karloff, H. and Zwick, U. (1997). A 7/8-approximation algorithm for MAX 3SAT? In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 406–415. IEEE.

Mahajan, M. and Raman, V. (1999). Parameterizing above guaranteed values: Maxsat and maxcut. *Journal of Algorithms*, 31(2):335–354.

Literatura IV

Makino, K., Tamaki, S., and Yamamoto, M. (2011). *Derandomizing HSSW Algorithm for 3-SAT*, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg.

Moser, R. A. and Scheder, D. (2011). A full derandomization of Schöning's k-SAT algorithm. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 245–252. ACM.

Nishimura, N., Ragde, P., and Szeider, S. (2004). Detecting backdoor sets with respect to horn and binary clauses. *SAT*, 4:96–103.

Nishimura, N., Ragde, P., and Szeider, S. (2007). Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8):509–523.

Paturi, R., Pudlik, P., Saks, M. E., and Zane, F. (1998). An improved exponential-time algorithm for k-sat. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 628–637. IEEE.

Literatura V

Razgon, I. and O'Sullivan, B. (2009). Almost 2-sat is fixed-parameter tractable. *Journal of Computer and System Sciences*, 75(8):435 – 450.

Rolf, D. (2005). Derandomization of ppsz for unique-k-sat. *Lecture Notes in Computer Science*, pages 216–225.

Samer, M. and Szeider, S. (2010). Algorithms for propositional model counting. *Journal of Discrete Algorithms*, 8(1):50–64.

Schoning, T. (1999). A probabilistic algorithm for k-sat and constraint satisfaction problems. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 410–414. IEEE.

Schuler, R. (2005). An algorithm for the satisfiability problem of formulas in conjunctive normal form. *Journal of Algorithms*, 54(1):40–44.

Villanger, Y. (2006). Improved exponential-time algorithms for treewidth and minimum fill-in. In *LATIN 2006: Theoretical Informatics*, pages 800–811. Springer.