Aproximační Algoritmy

poznámky z přednášky

Tomáš Sláma
10. 1. 2022

Toto PDF bylo automaticky vygenerováno z webové stránky https://slama.dev/aproximacni-algoritmy, která je preferovaný způsob jak dokument číst. Za případné chyby způsobené převodem se omlouvám.

Obsah

Úvodní informace **2**

Základní definice **2**
Pravděpodobnost v algoritmech . . . 2

Metrický TSP **2**
Kostrový algoritmus . . . 3
Christofidesův algoritmus . . . 3

Quicksort **4**

Konflikty v distribuovaných systémech **4**

Globální minimální řez **5**
Přímočarý algoritmus . . . 5
Spojující algoritmus . . . 5

Rozvrhování **6**
Hladový algoritmus . . . 6
Algoritmus lokálního prohledávání . . . 7
Largest Processing Time (LPT) . . . 7
Odbočka: online algoritmy . . . 8
Bin packing . . . 8

Hledání disjunktních cest **9**
Jednotkové kapacity . . . 9
Nejednotkové kapacity . . . 10

Splnitelnost (MAX-SAT) **11**
RAND-SAT . . . 11
LP-SAT . . . 12
BEST-SAT . . . 13
Derandomizace metodou podmíněných pravděpodobností . . . 14

Pokrývací problémy **14**
Vrcholové pokrytí . . . 14
Množinové pokrytí . . . 14
g-aproximační algoritmy . . . 15
Maximální nezávislá množina . . . 16

Hashovací funkce **18**
Dynamický slovník . . . 18
Statický slovník . . . 19

Testování **20**
Násobení matic . . . 20
Nulovost polynomů (Polynomial Identity Testing) . . . 20

Perfektní párování **21**
Izolující lemma . . . 21

Odkazy **22**

Úvodní informace

Tato stránka obsahuje moje poznámky z přednášky Jiřího Sgalla z akademického roku 2021/2022. Pokud by byla někde chyba/nejasnost, nebo byste rádi někam přidali obrázek/text, tak stránku můžete upravit pull requestem (případně mi dejte vědět na mail.

Základní definice

Definice (Optimalizační problém) je $\mathcal{I}, \mathcal{F}, f, g$

- $\mathcal{I}$ ... množina všech vstupů/instancí
  - *množina všech ohodnocených grafů*
- $\forall I \in \mathcal{I} : \mathcal{F}(I)$ ... množina přípustných řešení
  - *pro daný ohodnocený graf všechny kostry*
- $\forall I \in \mathcal{I}, A \in \mathcal{F}(I) : f(I, A)$ ... účelová funkce
  - *součet hran na kostře*
- g ... bit (zda chceme maximalizovat nebo minimalizovat)
  - *maximalizujeme*

Definice (NP-Optimalizační problém) je $\mathcal{I}, \mathcal{F}, f, g$, pro které platí stejné věci jako pro normální optimalizační problémy, ale navíc

- délka přípustných řešení $\leq \text{poly}(|I|)$.
- jazyk dvojic $(I, A), I \in \mathcal{I}, A \in \mathcal{F}(I)$ je v P (rychle umíme ověřit, zda je řešení přípustné)
- f počitatelná v polynomiálním čase

Definice: algoritmus A je R-aproximační alg., pokud:

- v polynomiálním čase v $|I|$ na vstupu I najde $A \in \mathcal{F}(I)$
- pro minimalizační problém: $\forall I : f(A) \leq R \cdot \text{OPT}(I)$
- pro maximalizační problém: $\forall I : f(A) \geq \text{OPT}(I)/R$

[1]

Pravděpodobnost v algoritmech

1. algoritmy s chybou: někdy dělají chybu, ale většinou ji neudělají
2. bez chyb, běží v průměrném čase polynomiálním

- třída NP: jazyky, pro které existuje polynomiální algoritmus A, který ověří správnost:
  - $\forall a \in L \; \exists b : A(a, b) = 1$
  - $\forall a \notin L \; \forall b : A(a, b) = 0$
- třída RP: jazyky, pro které existuje polynomiální algoritmus A, který ověří správnost:
  - $\forall a \in L \; \Pr_b [A(a, b)] \geq \frac{1}{2}$
  - $\forall a \notin L \; \Pr_b [A(a, b)] = 0$

Metrický TSP

- *Vstup:* metrika V, d na úplném ohodnoceném grafu
  - metrika $\equiv$ vrcholy splňují následující:
    1. trojúhelníkova nerovnost
    2. symetrie
    3. $d(x, y) = 0 \iff x = y$
  - pokud by to nebyla metrika, tak poly algoritmus neexistuje (jde převést na normální TSP)
- *Výstup:* cyklus C na všech vrcholech V
- *Cíl:* minimalizovat $d(C)$

[1] Pro minimalizační zajišťujeme, že naše je vždy dostatečně malé. Pro maximalizační zajišťujeme, že je vždy dostatečně velké.

Kostrový algoritmus

Algoritmus (kostrový)

1. najdeme minimální kostru
2. navštívíme všechny vrcholy (například přes DFS), čímž dostaneme tah přes všechny vrcholy
3. zkrátíme ji na cyklus tak, že vynecháme opakující-se vrcholy

Věta: algoritmus je 2-aproximační.

Důkaz: kostra je nejvýše tak velká, jako optimální řešení a tenhle algoritmus je lepší než 2 kostry (díky trojúhelníkové nerovnosti a symetrii – procházíme i tam i zpět)

Christofidesův algoritmus

(👀): brát hrany dvakrát je plýtvání – pospojujeme liché vrcholy přes minimální párování, abychom nemuseli chodit tam a zpět

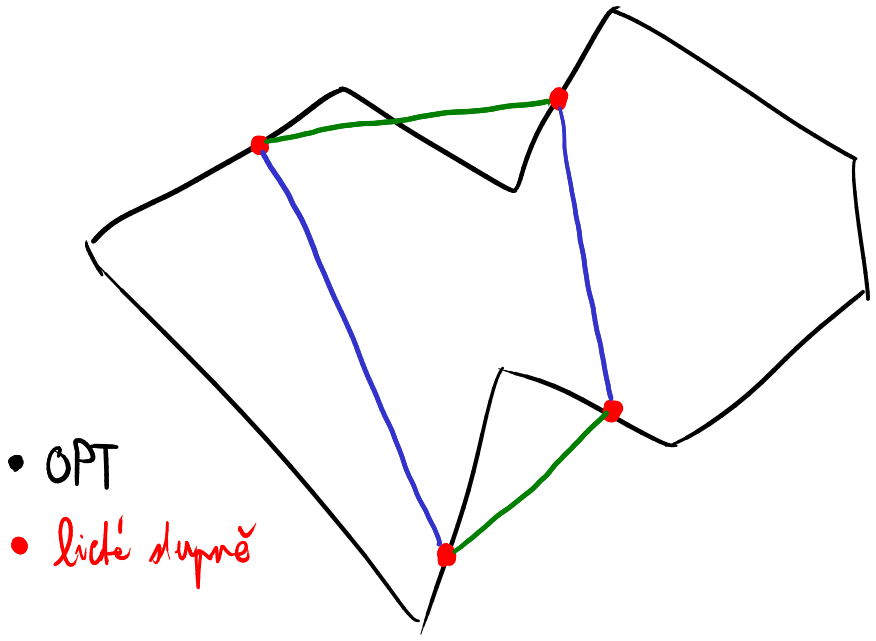
1. najdeme minimální kostru T
2. najdeme minimální perfektní párování M na vrcholech s lichými stupni v T
   - vždy existuje, jelikož máme úplný graf a vrcholů s lichým stupněm je sudý počet (všech je sudý)
3. zkrátíme na cyklus $T \cup M$

Věta: algoritmus je 3/2-aproximační.

Důkaz:

$$\text{ALG} \leq d(T) + d(M) \leq \text{OPT} + \frac{1}{2}\text{OPT}$$

Důkaz $d(M) \leq \frac{1}{2}\text{OPT}$ uděláme obrázkem:



Alespoň jeden z párování v cylku bude $\leq \frac{1}{2}\text{OPT}$, jelikož celý cyklus je jen zkrácené optimální řešení.

Poznámka:

- dnes umíme $(\frac{3}{2} - \varepsilon)$-aproximaci
- TSP v rovině: existuje $(1 + \varepsilon)$-aproximační schéma (ale stále je NP těžký)

Quicksort

Algoritmus (quicksort)

1. $|S| \leq 1 \ldots$ konec, vystoupíme S (base case)
2. jinak vybereme uniformě náhodně $p \in S$
3. $A = \{x \in S \mid x < p\}, B = \{x \in S \mid x > p\}$
   - posloupnost má všechny prvky rozdílné, takže chceme ostrá nerovnítka
4. rekurzivně se zavoláme na A, B
5. vystoupíme A, p, B

Věta: quicksort má průměrnou časovou složitost $n \cdot \log n$.

Důkaz: počítáme $A_{i,j} = \Pr[\text{porovnáme } i\text{-tý a } j\text{-tý prvek}]$

- zavedeme indikátorové veličiny $X_{i,j} = \begin{cases} 1 & A_{i,j} \text{ nastane} \\ 0 & \text{jinak} \end{cases}$ [2]

Lemma: nechť $i < j$. Pak $\Pr[A_{i,j}] = \frac{2}{j-i+1}$

Důkaz: to, že se dva prvky porovnají musí znamentat, že jeden z jich byl pivot, ale žádný mezi nimi pivot nebyl (jelikož by je to rozdělilo). Musíme tedy vybrat právě jeden z těchto dvou z intervalu $[i, j]$, kde je celkově $j - i + 1$ čísel.

Sečtením přes všechny dvojice $i < j$ dostaváme následující:

$$\begin{aligned} \mathbb{E}[\#\text{ porovnání}] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j} \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\ &= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i} \frac{2}{k} \\ &\cong n \cdot H_n \qquad H_n \ldots \text{harmonická posloupnost} \\ &\cong n \cdot \log n \end{aligned}$$

Konflikty v distribuovaných systémech

- n procesorů se snaží o přístup k jednomu zdroji
- přímá komunikace není možná
- v každém cylku může každý procesor požadovat přístup
  - povede se pouze, když o něho žádá jeden

Algoritmus:

1. v každém cylku zkus s pravděpodobností p přistoupit ke zdroji
   - p si nastavíme tak, aby to vyšlo hezky
2. opakuj, dokud se ti to nepovede

Věta: algoritmus s $p = \frac{1}{n}$ s pravděpodobností alespoň $1 - \frac{1}{n}$ uspěje po $t = 2en \ln n$ cyklech.

Důkaz: modifikujme algoritmus, aby zkoušel přistupovat i po té, co ho získal (lehčí počítání, které pouze zhorší pravděpodobnost úspěchu).

Nechť $A_{i,r}$ je jev, že i-tý proces upěl v r-tém cyklu. Pak

$$\Pr[A_{i,r}] = p \cdot (1-p)^{n-1} = \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{en}$$

[2] Pro připomenutí: - $\mathbb{E}[X_{i,j}] = \Pr[A_{i,j}]$ (indikátorová veličina) - $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$

Nyní počítáme $F_{i,t}$ které říká, že i-tý proces neuspěje v žádném z $t = 2en \ln n$ cyklů:

$$\Pr[F_{i,t}] = \prod_{r=1}^{t} (1 - A_{i,r}) \leq \left(1 - \frac{1}{en}\right)^t = \left(\left(1 - \frac{1}{en}\right)^{en}\right)^{\frac{t}{en}} \leq n^{-2}$$

To, že neexistuje proces, který neuspěje, odhadneme jako

$$\Pr\left[\bigcup_{i=1}^{n} F_{i,t}\right] \leq \sum_{i=1}^{n} \Pr[F_{i,t}] \leq n \cdot n^{-2} = n^{-1} = \frac{1}{n}$$

Globální minimální řez

- *Vstup:* neorientovaný graf (V, E)
- *Výstup:* $F \subseteq E$ tak, že $V, E \setminus F$ není souvislý
- *Cíl:* minimalizovat $|F|$

Přímočarý algoritmus

Algoritmus:

1. převedeme graf na ohodnocený s jednotkovými cenami
2. zafixujeme s
3. pro všechna t najdeme minimalní řez
4. vrátíme minimální, který jsme našli

- umíme vyřešit řádové v $\mathcal{O}(n^3)$

Spojující algoritmus

Algoritmus:

1. vybereme náhodnou hranu a její vrcholy spojíme do jednoho
2. opakujeme, dokud nemáme pouze dva vrcholy
3. zbylé hrany na konci jsou náš řez

- idea je to, že hran v minimálním řezu je málo a nejspíš se do nich netrefíme
- pracujeme s multigrafy – při kontrakci **zachováváme hrany**
- umíme ho implementovat rychle (řádově $\mathcal{O}(n^2 \cdot \log n)$)
- opravdu produkuje řez, protože vrcholy mezi výslednými komponentami danými vrcholy nemizí

Lemma: multigraf s n vrcholy a min. řezem velikosti k má alespoň $kn/2$ hran.

Důkaz: $\forall v$, hrany incidentní s v tvoří řez, proto musí platit $\forall v : d_v \geq k$. Dosazením dostáváme

$$|E| = \frac{1}{2} \sum_{v} d_v \geq \frac{1}{2} nk$$

Věta: pravděpodobnost, že najdeme daný minimální řez C je alespoň $\binom{n}{2}^{-1} = \frac{2}{n \cdot (n-1)}$.

Důkaz: nechť A_i jev, že v prvních i iteracích jsme nevybrali hranu z C.

- $\Pr[A_0] = 1$ (žádnou jsme ještě nevybrali)
- $\Pr[A_1] \geq 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}$
- $\Pr[A_2 \mid A_1] \geq 1 - \frac{2}{n-1}$
- $\Pr[A_2] = \Pr[A_2 \mid A_1] \cdot \Pr[A_1]$, z čehož vyplývá:

$$\begin{aligned}\Pr[A_{n-2}] &\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\cdots\left(1-\frac{2}{3}\right)\\ &= \frac{n-2}{n}\frac{n-3}{n-1}\frac{n-4}{n-2}\cdots\frac{2}{4}\frac{1}{3}\\ &= \frac{2}{n\cdot(n-1)} = \frac{1}{\binom{n}{2}} = \binom{n}{2}^{-1}\end{aligned}$$

Důsledek: každý graf G má $\leq \binom{n}{2}$ globálních minimálních řezů.

- jeden takový je například cyklus $k = 2$ – ten má opravdu řádově tolik řezů

Důkaz: každý běh algoritmu vystoupí právě jeden řez. Kdyby jich bylo více, tak nám pravděpodobnost nevychází (jevy jsou disjunktní).

(👀): Pro n^2 opakování algoritmu výše dostáváme nejmenší řez s pravděpodobností $\geq \frac{1}{2}$

Poznámka: algoritmus můžeme vylepšit tak, že části, ve kterých se nejvíce dělají chyby (konkrétně ty pozdější) opakujeme vícekrát (a vezmeme minimum).

Rozvrhování

- *Vstup:* m strojů, n úloh, každá o délce p_i
- *Výstup:* rozklad $\{1,\ldots,n\} = I_1 \cup I_2 \cup \ldots \cup I_m$ (rozvržení úloh na stroje)
- *Cíl:* minimalizovat $\max_{i=1}^{m} \sum_{j\in I_i}^{p_j}$ (délka nejdelšího stroje)

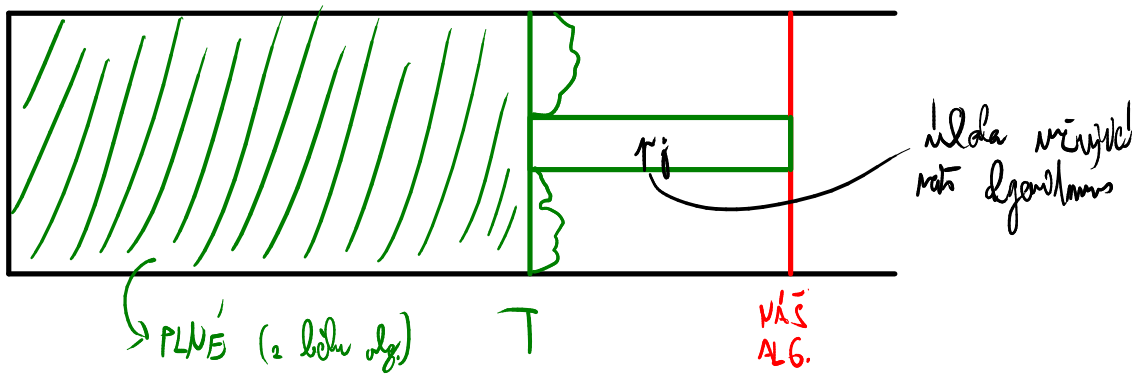
Hladový algoritmus

Algoritmus (hladový)

1. úlohu přidej vždy tam, kde jich je zatím nejméně
2. profit?

Věta (slabší odhad) hladový rozvrhovací algoritmus je 2-aproximační.

Důkaz:



Z obrázku vyplývá:

- $T \leq \text{OPT}$ (optimum muselo úlohy také někam dát)
- $p_j \leq \text{OPT}$ (optimum p_j muselo použít)

Spojením dostáváme $\text{ALG} = T + p_j \leq 2 \cdot \text{OPT}$

Věta (silnější odhad) hladový rozvrhovací algoritmus je $\left(2-\frac{1}{m}\right)$-aproximační.

Důkaz:

- $\frac{1}{m}\sum_{k=1}^{n} p_k \leq \mathrm{OPT}$
  - lépe, než rovnoměrně všechny úlohy rozvrhnout nemůžeme
- $\sum_{k=1}^{n} p_k \geq m \cdot T + p_j$
  - součet všech úloh je alespoň součet před T+ posledí úloha (vynechám „ocásky“)

Kombinací nerovností dostávám $T + \frac{p_j}{m} \leq \mathrm{OPT}$.

Nyní místo odhadu $T \leq \mathrm{OPT}$ použijeme tyto dva odhady:

$$T + \frac{p_j}{m} \leq \mathrm{OPT}$$
$$(p_j \leq \mathrm{OPT}) \cdot \left(1 - \frac{1}{m}\right)$$

Součtem dostáváme

$$T + \frac{p_j}{m} + \left(1 - \frac{1}{m}\right) p_j \leq \mathrm{OPT} + \left(1 - \frac{1}{m}\right) \mathrm{OPT}$$
$$\mathrm{ALG} \leq \left(2 - \frac{1}{m}\right) \mathrm{OPT}$$

Algoritmus lokálního prohledávání

Pro lokální algoritmus potřebujeme s rozvrhem pracovat formálněji:

- *Vstup:* m strojů, n úloh, každá o délce p_i
- *Výstup:* funkce přiřazující každé úloze startovní čas s_i, koncový čas c_i a stroj i
  - musí platit že $c_i = s_i + p_i$ a že se úlohy nepřekrývají
- *Cíl:* minimalizovat $\max_{i=1}^{m} \sum_{j \in I_i}^{p_j}$ (délka nejdelšího stroje)

Algoritmus (lokální prohledávání)

1. najdi libovolný rozvrh **bez mezer** (i na začátku) [3]
2. vezmi libovolné j s maximálním c_j
   - pokud existuje stroj i s délkou rozvrhu $< s_j$, tak přesuň j na i **s minimální délkou**
   - jinak vystup aktuální rozvrh

(👀): $c_{\min} nekles$

(👀): Každou úlohu přesuneme nejvýše jednou.

Důkaz: jelikož ji přesouváme na stroj s minimální délkou, tak by musel existovat nějaký s ještě menší, což by byl spor s tím, jak algoritmus funguje (dáváme na nejmenší).

Důsledek: algoritmus je polynomiální.

Věta (silnější odhad pro lokální algoritmus) algoritmus je $\left(2 - \frac{1}{m}\right)$-aproximační.

Largest Processing Time (LPT)

Algoritmus (LPT)

1. úlohy uspořádáme tak, že $p_1 \geq p_2 \geq \ldots \geq p_n$
2. použijeme hladový algoritmus

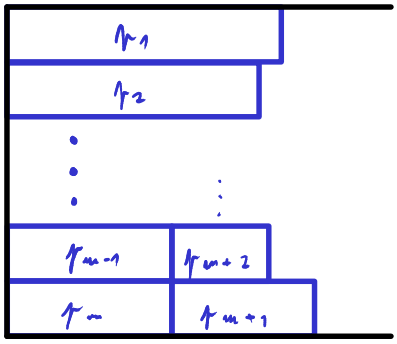
Věta: LPT je $\left(\frac{4}{3} - \frac{1}{3m}\right)$-aproximační algoritmus.

Důkaz: BUNO předpokládejme, že p_n určuje délku rozvrhu (kdyby ne tak na další úlohy zapomenu a řešení se tím nezmění). Rozlišíme 2 případy:

- $p_n \leq \frac{1}{3}\mathrm{OPT}$ – stejný výpočet jako předtím, jen silnější nerovnost:

[3] Prostě přesouváme stroje, které končí nejpozději někam, aby začínaly dříve a zlepšujeme tím maximum.

  - ALG $= T + p_n$, $T + \frac{p_n}{m} \leq$ OPT
  - stejným výpočtem jako předtím máme ALG $\leq$ OPT $+ \left(1 - \frac{1}{m}\right) \frac{1}{3}$OPT
- $p_n > \frac{1}{3}$OPT – LPT vygeneruje optimální rozvrh
  - víme, že délka poslední, a tedy **každé** úlohy je alespoň OPT/3
  - žádný počítač nebude mít 3 úlohy, jelikož by pak byl větší než optimum
  - chceme argumentovat, že LPT udělá stejné dvojice jako optimální rozvrh:
    * $p_m + p_{m+1} \leq$ OPT – uvážíme-li pouze prvních $m+1$ úloh, tak optimální rozvrh bude mít alespoň 2 na jednom počítači a ty rozhodně nebudou kratší než dvě nejkratší
      · tento rozvrh s méně úlohami je jistě $\leq$ OPT, proto nerovnost platí
    * $p_{m-1} + p_{m+2} \leq$ OPT – v optimálním rozvrhu budou mít alespoň 2 počítače dvojici úloh; v jedné z nich bude čtvrtá nejmenší (p_{m-1}), která tam bude s nejmenší (p_{m+2})
    * obdobně dostaneme všechny další dolní odhady…



Odbočka: online algoritmy

- vstup přichází postupně
- řešení musíme konstruovat postupně po krocích a pak už neměníme
- **hladový** je online, **lokální prohledávání** není
- pro $m = 2$: lepší než 3/2-aproximační neexistuje (úlohy délky $\{1, 1, 2\}$)
- pro $m = 3$ je opět hladový nejlepší
- pro $m > 3$ to už tak není (existují lepší)

Bin packing

- *Vstup:* $a_1, \ldots, a_n \geq 0$
- *Výstup:* rozklad $\{1, \ldots, n\} = I_1 \cup \ldots \cup I_m$ tak, že $\forall i : \sum_{j \in I_i} a_j \leq 1$
  - součet věcí v každém koši může být nejvýše 1
- *Cíl:* minimalizovat m (počet košů)

Algoritmus (first fit) dej a_j do prvního koše, do kterého se vejde.

Algoritmus (best fit) dej a_j do nejplnějšího koše, do kterého se vejde.

Věta: oba algoritmy jsou 1.7-aproximační (a je to těsný odhad).

Věta: Je NP těžké najít R-aproximační algoritmus pro $R \leq 3/2$

Důkaz: Je NP těžké rozhodnout, zda OPT $= 2$, jelikož pomocí něho můžeme přímočaře vyřešit problém dělení množiny na dvě části se stejným součtem (nastavíme velikost košů na tenhle součet), což je NP těžké.

Věta: Existuje asymptotické aproximační schéma, t. j.

$$(\forall \varepsilon)(\exists \text{ALG})(\forall I)\text{ALG}(I) \leq (1 + \varepsilon)\text{OPT}(I) + 1$$

Algoritmus (any fit) dej a_j do nějakého neprázdného koše; pokud nelze, dej ho do nového.

- zahrnuje jak best fit, tak first fit

Věta: každý any fit algoritmus má aproximační poměr ≤ 2.

Důkaz: Pro OPT $= 1$ triviální. Jinak nechť $B_i = \sum_{j \in I_i} a_j$. Musí platit, že $(\forall i, j, i \neq j) B_i + B_j > 1$ (jinak spor s během algoritmu). Posčítáním pro všechny dvojice dostáváme

$$\frac{m}{2} < \sum_{i=1}^{m} B_i = \sum_{j=1}^{n} a_j \leq \text{OPT}$$

Hledání disjunktních cest

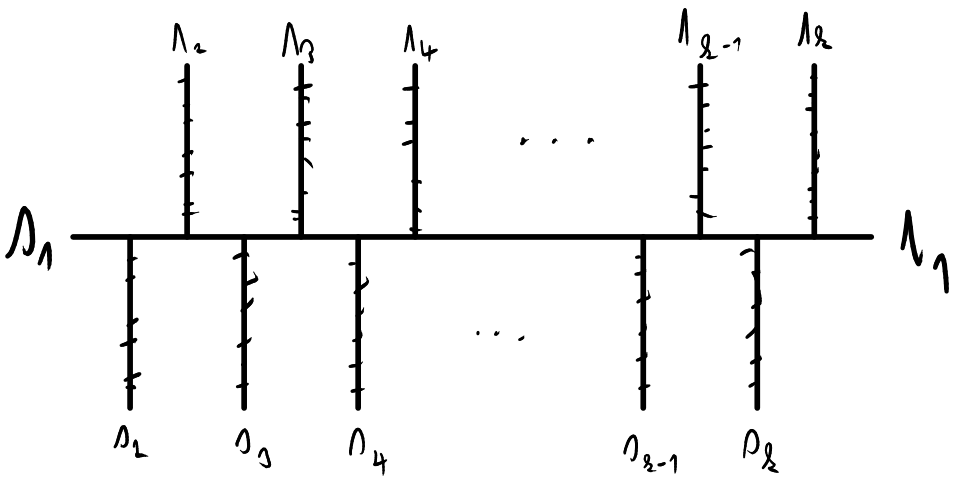
- *Vstup:*
  - graf $G = (V, E)$ (orientovaný/neorientovaný)
  - dvojice vrcholů $(s, t), \ldots, (s_k, t_k)$
  - kapacita hran c
- *Výstup:*
  - $I \subseteq \{1, \ldots, k\}$ (dvojice které spojíme cestou)
  - cesty $P_i, i \in I, P_i$ cesta z s_i do t_i tak, že každá hrana $e \in E$ leží na nejvýše c cestách P_i
- *Cíl:* minimalizovat $|I|$

Jednotkové kapacity

Algoritmus (hladový)

1. najdeme nejkratší cestu mezi nespojenou dvojicí (přes všechna i)
   - pokud neexistuje, tak vystoupíme
2. odebereme použité hrany

(👀): hladový algoritmus nemá aproximační poměr lepší než $\mathcal{O}(\sqrt{m})$:



Délka $s_1 \mapsto t_1$ je $\mathcal{O}(k)$. Abychom tuto cestu nezvolili, tak musejí mít ostatní alespoň $\mathcal{O}(k)$ hran a celkově jich tedy musí být řádově $m = \mathcal{O}(k^2)$. Naše řešení je tedy o $k = \mathcal{O}(\sqrt{m})$ horší.

Věta: Hladový algoritmus s $c = 1$ je $\mathcal{O}(\sqrt{m})$-aproximační.

Důkaz: BUNO OPT ≥ 1 (jinak bychom hned skončili)

- pak $|I| = \text{ALG} \geq 1$ (také nějakou najdeme)

Nechť $I^*, \{P_i^* \mid i \in I^*\}$ je optimum. Počítejme cesty:

- dlouhé cesty: $|P_i^*| > \sqrt{m}$
  - je jich $\leq \sqrt{m}$ (jinak bych měl více než m hran)
  - ty mi tedy nic nekazí, jelikož nám stačí, že algoritmus najde nějakou cestu

- krátké cesty:
  - $i \in I \ldots$ vše ok
  - $i \notin I \ldots$ P_i^* má nějakou společnou hranu s nějakou cestou P_j t. ž. $|P_j| \leq \sqrt{m}$
    - ve chvíli, kdy algoritmus poprvé vybral cestu delší než $\sqrt{m}$ už nemohl vybrat P_i^*, protože tu blokovala nějaká cesta, kterou již předtím zvolil (a ta musí být krátká)

Tedy počet krátkých cest $P_i^* \leq |I|\,(\sqrt{m}+1)$

- 1 – náš algoritmus a optimum vybrali stejnou cestu
- $\sqrt{m}$ – krátká cesta v našem řešení zablokuje nejvýše $\sqrt{m}$ ostatních krátkých

$$\mathrm{OPT} = |I^*| \leq \underbrace{\sqrt{m}}_{\text{dlouhé}} + \underbrace{|I|\left(\sqrt{m}+1\right)}_{\text{krátké}} \leq \mathcal{O}(\sqrt{m})|I| = \mathcal{O}(\sqrt{m})\mathrm{ALG}$$

Nejednotkové kapacity

Algoritmus (hladový pro nejednotkovou kapacitu)

1. zvolíme $\beta = \left\lceil m^{\frac{1}{c+1}} \right\rceil$
2. najdeme nejkratší cestu mezi nespojenou dvojicí (přes všechna i)
   - pokud neexistuje nebo $d(P_i) \geq \beta^c$, vystoupíme
3. přenásobíme délku hran nejkratší cesty faktorem β a opakujeme

(👀): algoritmus nepoužije e s $d(e) \geq m$

Důsledek: výsledné řešení je přípustné

- po c použitích hrany e je $d(e) = \beta^c \approx m^{\frac{c}{c+1}} < m$, dále algoritmus hranu nepoužívá

Důsledek: algoritmus je polynomiální.

Věta: Hladový algoritmus je $\mathcal{O}(\beta)$-aproximační.

- pro $c = 1$ máme $\beta = m^{\frac{1}{c+1}} = \sqrt{m}$, což odpovídá

Důkaz: BUNO optimum ≥ 1 (jinak bychom hned skončili)

- pak $|I| = \mathrm{ALG} \geq 1$ (také nějakou najdeme)

Opět si rozmyslíme to, když cesta je v našem algoritmu a když není:

- $i \in I \ldots$ vše ok
- $i \notin I \ldots$ na konci algoritmu je $d(P_i^*) \geq \beta^c$ (jinak by ji algoritmus použil)

Nyní nejprve zesdola odhadneme $d(E)$ na konci algoritmu:

- $\beta^c(|OPT| - |I|)$: dolní odhad na délku cest, které algoritmus nespojil ale optimální ano
  - každá cesta má na konci delku alespoň β^c a je jich alespoň $|\mathrm{OPT}| - |I|$
- $d(E) \geq \beta^c(|\mathrm{OPT}| - |I|)/c$: každou hranu můžeme použít c-krát

Poté odhadneme $d(E)$ zeshora (opět na konci algoritmu):

- na začátku $d(E) = m$ (délky hran jsou jednotkové)
- po výběru $P_i \ldots$ $d(P_i) \leq \beta^c \cdot \beta = \beta^{c+1}$
- na konci $d(E) \leq m + |I|\beta^{c+1} \leq (|I|+1)\,\beta^{c+1}$
  - $|I|$ je počet kroků, v každém jsme zvětšili délku vybrané cesty na β^{c+1}
  - druhá úprava je pouze z definice

Po spojení nerovnic dostáváme:

$$\begin{aligned} \beta^c\,(|\mathrm{OPT}| - |I|) &\leq c \cdot d(E) \leq c\,(|I|+1)\,\beta^{c+1} \\ |\mathrm{OPT}| - |I| &\leq \beta c(|I|+1) \\ |\mathrm{OPT}| &\leq \mathcal{O}(\beta)|I| \end{aligned}$$

Splnitelnost (MAX-SAT)

- *Vstup:* $C_1 \wedge \ldots \wedge C_m$, každá klauzule je disjunkcí $k_j \geq 1$ literálů
  - každá C_j má váhu w_j ($= 1$ by default)
- *Výstup:* ohodnocení $a \in \{0,1\}^n$
- *Cíl:* maximalizovat $\sum w_i$ (pro $w_j = 1$ je to počet splněných klauzulí)

Poznámka:

- MAX-3SAT: $k_j \leq 3$: NP těžké
- 2SAT: orientovaný graf, ve kterém různé literály implikují jiné
  - $x_1 \wedge x_2$ implikuje $\overline{x}_1 \implies x_2$ (a obrácené)
  - testujeme tedy, zda graf neobsahuje cyklus (protože by pak nešel splnit)
- MAX-2SAT: NP těžké

Předpokládáme:

- žádný literál se v klauzuli neopakuje
- nejvýše jeden z $x_i, \overline{x}_i$ se vyskytuje v klauzuli

RAND-SAT

Algoritmus (RAND-SAT)

1. vybereme nezávisle náhodně všechny literály ($p = 1/2$)
2. profit?

Věta: RAND-SAT je 2-aproximační algoritmus.

Důkaz: pro každou klauzuli zavedeme indikátorovou proměnnou Y_j.

- pravděpodobnost, že C_j není splňená je $\frac{1}{2^k}$

Díky tomu, že $k \geq 1$ máme $\mathbb{E}[Y_j] = \Pr[C_j \text{ is satistied}] = 1 - \frac{1}{2^k} \geq \frac{1}{2}$ a tedy:

$$\mathbb{E}\left[\sum_{j=1}^{m} w_j Y_j\right] \overset{\text{linearita}}{=} \frac{1}{2}\sum_{j=1}^{m} w_j \geq \frac{1}{2}\text{OPT}$$

Poznámka: pro $k = 3$ dostáváme po dosazení $\frac{8}{7}$-aproximaci

- $\forall \varepsilon > 0 : \left(\frac{8}{7} - \varepsilon\right)$-aproximace MAX-3SATu ne NP úplná

BIASED-SAT

- předpoklad: $\forall i : \sum_{j:C_j=x_i} w_j \geq \sum_{j:C_j=\overline{x}_i} w_j$ [4]
  - chceme preferovat splnění krátkých kladných před splněním krátkých záporných
  - pokud nevychází, tak literál všude znegujeme

Algoritmus (BIASED-SAT)

1. vybereme nezávisle náhodně všechny literály
   - true: $p > \frac{1}{2}$, jinak false; hodnotu p najdeme později

Věta: BIASED-SAT je $\left(\phi = \frac{\sqrt{5}-1}{2}\right)$-aproximační algoritmus.

Uvažme C_j (a opět indikátorové veličiny Y_j):

- kladný literál: $Y_j = p$
- $k_j \geq 2$: $Y_j = 1 - p^a(1-p)^b \overset{p>\frac{1}{2}}{\geq} 1 - p^{a+b} \overset{k_j \geq 2}{\geq} 1 - p^2$
  - a je počet kladných a b počet záporných literálů

[4] Předchozí algoritmus měl problémy s krátkými klauzulemi, jelikož je menší šance, že nějakou splní. Zkusíme to napravit tím, že jim budeme dávat preferenci.

Po vyřešení $p = 1 - p^2$ dostáváme $p = \phi = \frac{\sqrt{5}-1}{2}$.

Nechť U množina klauzulí bez záporných jednotkových.

(👀): $\text{OPT} \leq \sum_{j \in U} w_j$

- zde používáme předpoklad – kladné literály je splňovat lepší než záporné

$$\begin{aligned}
\mathbb{E}\left[\sum_{j=1}^{m} w_j Y_j\right] &= \sum_{j=1}^{m} w_j \mathbb{E}\left[Y_j\right] \\
&\geq \sum_{j \in U} w_j \cdot \Pr\left[C_j \text{ je splněná}\right] \\
&\geq \sum_{j \in U} w_j \cdot p \\
&\geq p \cdot \text{OPT}
\end{aligned}$$

LP-SAT

Algoritmus (LP-SAT)

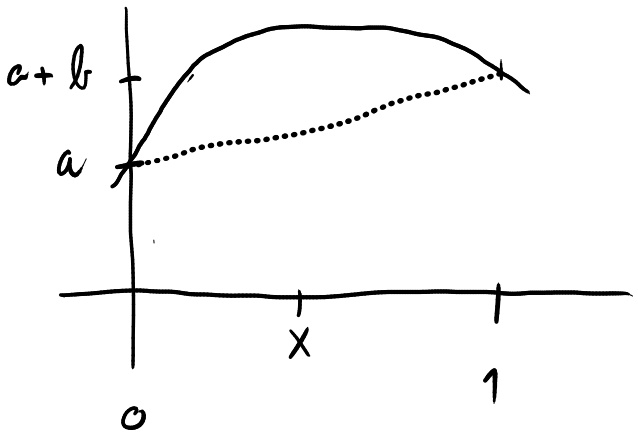
1. pro každou proměnnou si pořídíme binární proměnnou y_i, pro každou klauzuli binární proměnnou z_j
2. postavíme lineární program s těmito proměnnými
   - negaci zachytíme jako $1 - y_i$
   - pro každou klauzuli chceme $z_j \leq \sum_{\text{kladné}} y_i + \sum_{\text{záporné}} (1 - y_i)$
   - maximalizujeme $\sum z_j$
3. zrelaxujeme program a vyřešíme ho (dostaneme optimum y^*, z^*)
4. nastavíme proměnné x_i na true s pravděpodobností y_i^*

Věta: LP-SAT je $\left(1 - \frac{1}{e}\right)$-aproximační algoritmus.

Fakt (A - A/G nerovnost) $\prod_{i=1}^{n} a_i^{\frac{1}{n}} \leq \frac{1}{n} \sum_{i=1}^{n} a_i$

Fakt (B - konvexní funkce) pokud je funkce na $[0, 1]$ konkávní a $f(0) = a, f(1) = a + b$, pak

$$\forall x \in [0, 1] : f(x) \geq a + bx$$



Fakt (C - odhad na 1/e) $\left(1 - \frac{1}{n}\right)^n \leq \frac{1}{e}$

Důkaz: uvažme y^*, z^* a C_j s délkou k_j; pak

$$\begin{aligned}
\Pr\left[C_j \text{ není splněná}\right] &= \overbrace{\prod_{i:x_i\in C_j}(1-y_i^*)}^{\text{kladné}}\overbrace{\prod_{i:\overline{x}_i\in C_j} y_i^*}^{\text{záporné}} \\
&\overset{A}{=} \left[\frac{1}{k_j}\left(\sum_{i:x_i\in C_j}(1-y_i^*)+\sum_{i:\overline{x}_i\in C_j} y_i^*\right)\right]^{k_j} \\
&= \left[1-\frac{1}{k_j}\left(\sum_{i:x_i\in C_j} y_i^*+\sum_{i:\overline{x}_i\in C_j}(1-y_i^*)\right)\right]^{k_j} \\
&\leq \left(1-\frac{z_j^*}{k_j}\right)^{k_j} \qquad \text{//definice LP}
\end{aligned}$$

Nás zajímá splnění, tedy:

$$\begin{aligned}
\Pr\left[C_j \text{ je splněná}\right] &\geq \overbrace{1-\left(1-\frac{z_j^*}{k_j}\right)^{k_j}}^{f(z_j^*)} \\
&\overset{B}{\geq} \left[1-\left(1-\frac{1}{k_j}\right)^{k_j}\right] z_j^* \quad \overset{C}{\geq} \left(1-\frac{1}{e}\right) z_j^*
\end{aligned}$$

Pro fakt B jsme pozorovali, že $a = f(0) = 0$ a také že druhá derivace je nekladná. Pak:

$$\begin{aligned}
\mathbb{E}\left[\sum_{j=1}^{m} w_j Y_j\right] &= \sum_{j=1}^{m} w_j \mathbb{E}\left[Y_j\right] \\
&\geq \sum_{j\in U} w_j \cdot \Pr\left[C_j \text{ je splněná}\right] \\
&\geq \sum_{j\in U} w_j \cdot \left(1-\frac{1}{e}\right) z_j^* \\
&= \left(1-\frac{1}{e}\right) \text{OPT}
\end{aligned}$$

BEST-SAT

Algoritmus (BEST-SAT)

1. při přizazení s pravděpodobností 1/2 použijeme RAND-SAT, jinak použijeme BEST-SAT
2. zažijeme existenční krizi z toho, že takovýhle algoritmus funguje a je asymptoticky optimální

Věta: BEST-SAT je $\frac{3}{4}$-aproximační.

Důkaz: chceme dokázat, že $\Pr\left[C_j \text{ je splněná}\right] \geq \frac{3}{4} z_j^*$.

Podívejme se, s jakou pravděpodobností splní klauzuli algoritmy:

- RAND-SAT: $1 - \frac{1}{2^k}$ (alespoň jedna musí být splněná a volíme s $p = 1$)
- LP-SAT: $\left[1-\left(1-\frac{1}{k}\right)^k\right] z_j^*$ (formulka z minulého důkazu těsně před odhadem)

k_j	RAND-SAT	LP-SAT	BEST-SAT
1	$\frac{1}{2} \geq \frac{1}{2} z_j^*$	$1 \cdot z_j^*$	$\frac{1}{2}\frac{1}{2} + \frac{1}{2} z_j^* \geq \frac{3}{4} z_j^*$
2	$\geq \frac{3}{4} z_j^*$	$\frac{3}{4} \cdot z_j^*$	$\geq \frac{3}{4} z_j^*$
≥ 3	$\geq \frac{7}{8} z_j^*$	$\geq \left(1-\frac{1}{e}\right) \cdot z_j^*$	$> \frac{3}{4} z_j^*$

Derandomizace metodou podmíněných pravděpodobností

Neformálně: postupně plníme klauzule tak, že náhodně vybíráme pravděpodobnosti. Jelikož počet splnění určuje to, jak hodnoty vybereme, tak je můžeme vybírat (*podmíněně* se rozhodujeme, jak to dopadne, když $x_i = 0$ nebo $x_i = 1$ a vybereme si to lepší). Aproximační poměr si neshoršíme, jelikož vždy vybírám větší z pravděpodobností.

Pokrývací problémy

Vrcholové pokrytí

- *Vstup:* graf G, ceny vrcholů $c_v \geq 0$
- *Výstup:* $W \subseteq V$ tak, že $\forall e \in E : e \cap W \neq 0$
- *Cíl:* minimalizovat $c(W) = \sum_{v \in W} c_v$

Algoritmus (LP relaxace)

1. vytvoř celočíselný lineární program:
   - proměnné jsou binární podle vrcholů, které vybíráme
   - podmínky jsou $\forall (u, v) \in E : x_u + x_v \geq 1$ (chceme pokrýt všechny hrany)
   - minimalizujeme $\sum_{v \in V} x_v c_v$
2. zrelaxuj lineární program (proměnné jsou teď reálné)
3. použij ho při řešení – zvol v když $x_v \geq \frac{1}{2}$
   - dává správné řešení, jelikož pro splnění podmínek je vždy alespoň jeden z $(x_u, x_v) \geq \frac{1}{2}$

Věta: algoritmus je 2-aproximační.

Důkaz: proměnné jsme z $\geq \frac{1}{2}$ zaokrlouhlovali na 1, čímž jsme řešení max. zdvojnásobili.

Množinové pokrytí

- *Vstup:* množiny $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$, ceny $c_1, \ldots, c_m \geq 0$
- *Výstup:* $I \subseteq \{1, \ldots, m\}$ t. ž. $\bigcup_{i \in I} S_i = \{1, \ldots, n\}$
- *Cíl:* minimalizovat $\sum_{i \in I} c_i$

Pro rozbor budeme potřebovat ještě dva parametry:

- $f = \max_{e=1}^{n} |\{j \mid e \in S_j\}|$ (v kolika nejvíce množinách je nějaký prvek)
- $g = \max_{j=1}^{m} |S_j| \leq n$ (velikost největší množiny)

(👀): vrcholové pokrytí je množinové pokrytí s $f \leq 2$

f-aproximační algoritmy

Algoritmus (LP relaxace) [5]

1. vytvoř celočíselný lineární program:
   - proměnné jsou $x_1, \ldots, x_m \geq 0$ podle **množin**
   - podmínky jsou $\forall e \in \{1, \ldots, n\} : \sum_{j \mid e \in S_i} x_j \geq 1$ (chceme pokrýt všechny prvky univerza)
   - minimalizujeme $\sum_{i \in \{1, \ldots, m\}} x_i c_i$
2. zrelaxuj lineární program (proměnné jsou teď reálné)
3. použij ho při řešení – zvol v když $x_v \geq \frac{1}{f}$
   - dává správné řešení – argument je stejný jako u vrcholového pokrytí

Věta: algoritmus je f-aproximační.

Důkaz: proměnné opět zvětšuji z $\frac{1}{f}$ na 1, řešení tedy zhorším nejvýše f-krát.

---

[5] Program pro vrcholové pokrytí: - proměnné jsou binární podle vrcholů, které vybíráme - podmínky jsou $\forall (u, v) \in E : x_u + x_v \geq 1$ (chceme pokrýt všechny hrany) - minimalizujeme $\sum_{v \in V} x_v c_v$

[6] **Význam primáru (sběratel):** jak můžu nejlevněji nakoupit balíčky známek tak, abych měl všechny známky. **Význam duálu (procejce):** kolik můžu nejvíce účtovat za každou známku, aby byl obchod ochotný kupovat známky a tvořit z nich balíčky.

(👀): duál programu vypadá následně: [6]

- proměnné jsou $y_1, \ldots, y_n \geq 0$ pro každý **prvek**
- podmínky jsou $\forall e \in \{1, \ldots, m\} : \sum_{e \in S_j} y_e \leq c_j$
- maximalizujeme $\sum_{e \in S_j} y_e$

(👀): podmínky komplementarity:

- $\forall j : x_j^* = 0 \vee \sum y_e^* = c_j$
  - pokud by prodejce na obálce vydělal, tak ji sběratel nekoupí
  - pokud prodejce na známce nevydělává, tak ji sběratel koupí
- $\forall e : y_e^* = 0 \vee \sum_{j|e \in S_j} x_j^* = 1$
  - pokud prodejce prodává známku zdarma, tak jí sběratel nakoupí trochu více
  - pokud prodejce známku zdarma neprodává, tak jí sběratel nekoupí víc než potřeba

Algoritmus (primárně-duální algoritmus)

1. $y_1, \ldots, y_n = 0; I = \emptyset, E = \emptyset$
2. dokud existuje nepokryté $e \notin E$, tak zvyšíme y_e „co nejvíce“:
   - $\delta = \min_{j|e \in S_j} \left(c_j - \sum_{e \in S_j} y_e \right)$
     - zvyšujeme tak, abychom splnili tu nejpřísnější duální podmínku
   - $y_e = y_e + \delta$
   - $\forall j : e \in S_j$ a $\sum_{e \in S_j} y_e = c_j$ přidám j do pokrytí ($I = I \cup \{j\}$) a $E = E \cup S_j$
     - do algoritmu přidáme ty množiny, jejichž podmínky komplementarity jsme naplnili

(👀): po přidání do algoritmu se y_e prvku nezmění (ostře splníme nějakou rovnost)

Věta: algoritmus je f-aproximační.

Důkaz:

$$\begin{aligned}
\text{ALG} &= \sum_{j \in I} c_j && \text{//definice} \\
&= \sum_{j \in I} \sum_{e \in S_j} y_e && \text{//definice} \\
&\leq \sum_{e=1}^{n} f \cdot y_e && \text{//prohození sumy + definice } f \\
&\leq f \cdot \text{OPT} && \text{//hodnota duálního řešení}
\end{aligned}$$

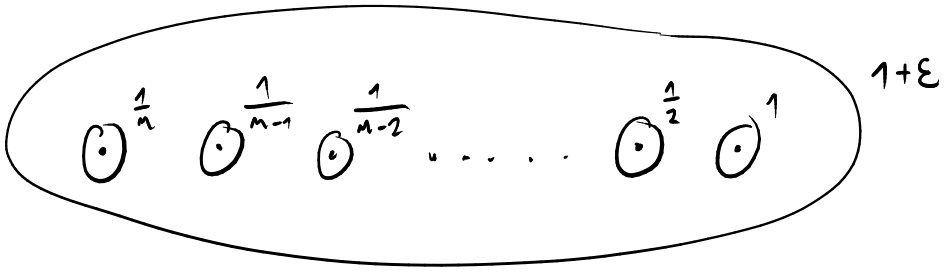
g-aproximační algoritmy

Algoritmus (hladový)

1. $I = \emptyset, E = \emptyset, q_e = 0$
   - q je vektor indexovaný prvky, pomůže nám při analýze algoritmu
     - odpovídá ceně za pokrytí daného prvku
2. opakovaně ber „nejlepší“ množinu: přidáme množinu s minimálním $\left(p_j = \frac{c_j}{|S_j \setminus E|} \right)$
   - p_j odpovídá tomu, kolik zaplatíme za pokrytí nového prvku
   - $\forall e \in S_j \setminus E : q_e = p_j$ (uložíme cenu nově pokrytých prvků)
   - $I = I \cup \{j\}, E = E \cup S_j$ (přidáme tuto množinu a pokryté prvky)

Věta: algoritmus je $(H_g \approx \ln g \leq \ln n)$-aproximační

(👀): algoritmus nemůže být lepší (viz následující protipříklad):



(👀): ALG $= \sum_{e=1}^{n} q_e$

- vyplývá z toho, že jsme cenu p_j při přidávání rozdělili do q_e

Lemma: $\overline{q} = \frac{1}{H_g} \cdot q$ je přípustné řešení duálního LP

Důkaz: chceme dokázat, že $\sum_{e \in S_j} \overline{q}_e \leq c_j$ (přímo podmínka v duálu). Nechť $S_j = \{e_1, \ldots, e_k\}$

- očíslujeme tak, že e_k je první pokrytý, e_{k-1} druhý, až e_1 poslední

(👀): $q_{e_i} \leq \frac{c_j}{i}$

- v i-tém kroku ještě nejsou pokryté prvky $1, \ldots, i$
- z definice vybíráme nejlevnější možnou množinu

Nyní dostáváme

$$\sum_{e \in S_j} q_e = \sum_{1}^{k} q_{e_i} \leq \frac{c_j}{1} + \frac{c_j}{2} + \ldots = H_k \cdot c_j$$

$$\sum_{e \in S_j} \overline{q}_e = \frac{1}{H_g} \sum_{e \in S_j} q_e \leq \frac{1}{H_g} \cdot H_k \cdot c_j \leq c_j$$

Maximální nezávislá množina

- *Vstup:* graf $G = (V, E)$
- *Výstup:* $I \subseteq V$ nezávislá množina, maximální **vzhledem k inkluzi**
  - největší moc dobře řešit nejde (ani aproximovat)

Nás zajímá najít rychlý paralelní algoritmus:

- operací chceme udělat řádově $\mathcal{O}(\log n)$
- k dispozici máme řádově m procesorů (každý vrchol/hrana má jeden)
- povolujeme procesorům najednou šahat na data a najednou měnit data **na stejnou věc**

Algoritmus (rychlý paralelní)

1. $I = \emptyset$
2. dokud $V \neq \emptyset$, tak každý následující krok děláme paralelně:
   1. $\forall v \in E$ pokud je stupeň 0, pak přidáme do I a vymažeme z V
   2. $\forall v \in E$ označ v (přidej do S) s pravděpodobností $\frac{1}{2d_v}$ (nezávisle)
   3. $\forall u, v \in E$ pokud u i v jsou označené, odeber značku nižšího stupně
      - nižší stupeň proto, abychom odebírali hran co nejvíce
   4. přidej označené vrcholy do I a odeber je **a jejich sousedy** (a odpovídající hrany) z V
      - sousedy množiny S značíme $N(S)$

Definice: vrchol je **dobrý**, jestliže má $\geq \frac{d_v}{3}$ sousedů stupně $\leq d_v$

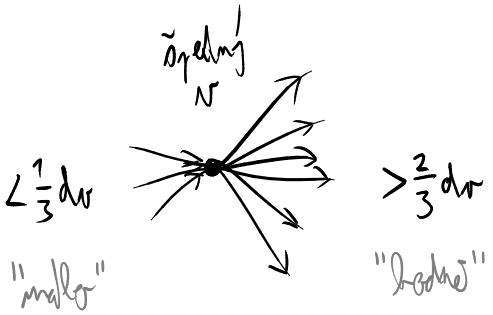
- má velkou pravděpodobnost, že ho vyřešíme výběrem souseda, protože má hodně sousedů malého stupně [7]
- analogicky špatný vrchol a dobrá (obsahuje dobrý vrchol) a špatná hrana

[7] Chceme, aby se graf v každé iteraci zmenšil o nějakou část a iterací bylo tedy logaritmicky. Uděláme to počítání toho, že máme hodně dobrých hran a že jich hodně zmizí.

Lemma: alespoň polovina hran je dobrá.

Důkaz: hrany zorientujeme od menšího k většímu stupni (rovnost řešíme libovolně)

- v špatný $\implies d_v^{\text{in}} < \frac{d_v}{3}$
  - z definice – vstupující jsou stejného nebo menšího stupně, takže jich má málo, jinak by byl dobrý
  - $> \frac{2d_v}{3}$ vstupuje a platí $d_v^{\text{in}} \leq \frac{1}{2} d_v^{\text{out}}$
    - „za každou špatnou hranu nejvýše dvě dobré“



Nyní počítáme

$$\begin{aligned}
|\text{špatné hrany}| &\leq \sum_{v \text{ špatný}} d_v^{\text{in}} && \text{//špatná hrana jde do špatného vrcholu} \\
&\leq \sum_{v \text{ špatný}} \frac{1}{2} d_v^{\text{out}} && \text{//nerovnost výše} \\
&\leq \sum_{v \in E} \frac{1}{2} d_v^{\text{out}} \\
&\leq \frac{1}{2}|E|
\end{aligned}$$

Tedy dobrých je $\geq \frac{1}{2}$.

Lemma: existuje $\alpha > 0$ t. ž. $\forall v$ **dobrý** platí

$$\Pr\left[v \in S \cup N(S)\right] \geq \alpha \tag{8}$$

- přímo z toho plyne to, co chceme, jelikož dobré hrany jsou pouze u dobrých vrcholů

Důkaz: Pro dobrý vrchol v platí následující:

$$\begin{aligned}
\Pr\left[v \text{ má souseda označeného v kroku 2}\right] &\geq 1 - \overbrace{\prod_{w \in N(v)} \left(1 - \frac{1}{2d_w}\right)}^{\text{neoznačíme žádného souseda}} \\
&\geq 1 - \left(1 - \frac{1}{2d_v}\right)^{\frac{d_v}{3}} && \text{//lemma výše} \\
&= \text{konstanta}
\end{aligned}$$

Může být špatné, když by se hodně ze sousedů dobrého vrcholu odstranilo. To dokážeme, že se nestane tím, že ukážeme, že u libovolného vrcholu v odstraníme značku s $\leq$ konstantní pravděpodobností (jen pozor, v Pr

[8] Pravděpodobnost, že dobrý vrchol odstraním (buď označením toho vrcholu samotného nebo nějakého jeho souseda) je $\alpha > 0$.

používáme podmíněně, že v byl označený):

$$\begin{aligned}\Pr[\text{odstraníme značku}] &= \Pr[\text{je označený soused s větším stupněm}] \\ &= \Pr[\exists u \in N(v) : d_u \geq d_w \wedge u \text{ byl označený}] \\ &\leq \sum_{u \in N(v) | d_u \geq d_v} \Pr[u \text{ byl označený}] \\ &\leq \sum_{w \in N(v)} d_w \cdot \frac{1}{2d_w} \\ &\leq \frac{1}{2}\end{aligned}$$

Nikde v důkazu nepočítáme s pravděpodobností označení dobrého vrcholu, což nepotřebujeme.

Věta: očekávaný počet fází algoritmu je $\leq \mathcal{O}(\log n)$

Důkaz: nechť M_i = počet hran po i fázích. Platí, že $\mathbb{E}[|M_{i+1}|] \leq \left(1 - \frac{\alpha}{2}\right) \mathbb{E}[|M_i|]$

- podle lemmatu je $\geq M_i/2$ hran dobrých a dobrá hrana je odebrána s $p = \alpha$

Tedy po logaritmicky mnoho krocích (v m nebo n) odstraníme všechny hrany pravděpodobností alespoň $\frac{1}{2}$.

TODO: derandomizace pomocí 2-nezávislých proměnných

Hashovací funkce

Definice: nechť $M, |M| = m, N, |N| = n, H \subseteq \{f \mid f : M \mapsto N\}$

- systém H je 2-univerzální, jestliže [9]

$$(\forall x_1, x_2 \in M, x_1 \neq x_2) \Pr_{h \in H}[h(x_1) = h(x_2)] \leq 1/n$$

- systém H je **silně** 2-univerzální, jestliže

$$(\forall x_1, x_2 \in M, x_1 \neq x_2)(\forall y_1, y_2 \in N) \Pr_{h \in H}[h(x_1) = y_1 \wedge h(x_2) = y_2] = 1/n^2$$

Příklad: pro $M = N$ je těleso máme silně 2-univerzální systém

$$H = \{h_{a,b} \mid a, b \in N\} \quad h_{a,b} : x \mapsto ax + b$$

Příklad: pro $|M| \gg |N|$ můžeme vzít $\overline{H} \subseteq \{f \mid f : M \mapsto M\}$ a vytvořit z něho $H \subseteq \{f \mid f : M \mapsto N\}$ tím, že budeme brát funkce mod$n$

- $\overline{H}$ silně 2-univerzální $\implies$ H univerzální
  - pokud $n \mid m$, tak máme silnou univerzalitu

Dynamický slovník

Příklad (dynamický slovník) universum M, $|M| = 2^d$, slovník $S \subseteq M, |S| = s$

- reprezentujeme S tabulkou $N, |N| = n = \mathcal{O}(s)$
- operace (trvá průměrně $\mathcal{O}(1)$):
  - vložení do S
  - vyhledávání x v S
  - vymazání x z S

[9] **2-Univerzalita:** pro dva rozdílné prvky máme pro náhodnou hashovací funkci z rodiny omezenou pravděpodobnost, že se namatchují na stejnou hodnotu. **Silná 2-univerzalita:** zahashované hodnoty x_1, x_2 tvoří dvě náhodné po dvou nezávislé veličiny. Takže kromě toho, že jsou univerzální (když zafixuju jeden, tak se tím druhým trefím s pravděpodobností $\frac{1}{n}$ to platí i pro libovolnou dvojici, na kterou prvky mapuju.

Zvolíme $n \in [s, 2s], H, h \in H$ náhodně uniformně:

- $h(x)$ je očekávaná pozice v poli
- kolize se přidávají do spojového seznamu pole (n_i je počet prvků)

Lemma: pokud $n = \mathcal{O}(s)$, tak průměrná doba operace je $\mathcal{O}(1)$

Důkaz: chceme $(\forall x \in S)\ \mathbb{E}\left[n_{h(x)}\right] = \mathcal{O}(1)$. Budeme počítat počet kolizí na jeden prvek:

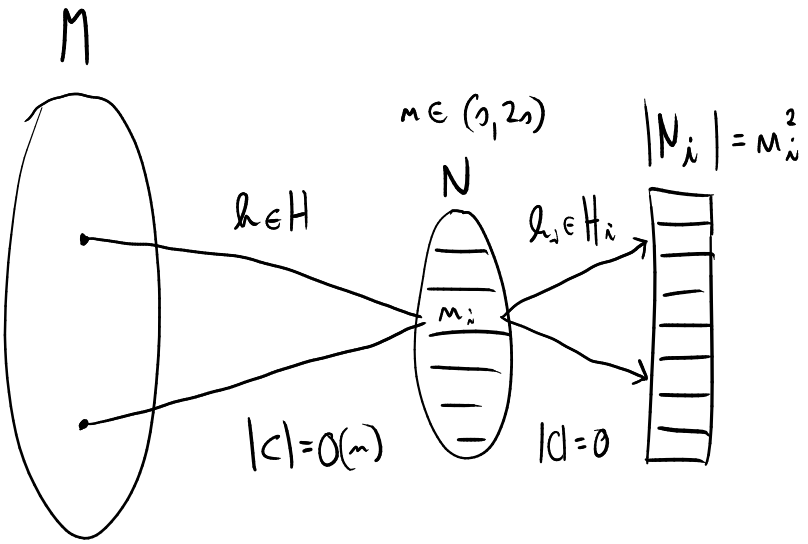
- nechť $X_y = \begin{cases} 1 & h(y) = h(x) \\ 0 & \text{jinak} \end{cases}$
- jelikož $(\forall x, y, x \neq y)\ h(x), h(y)$ jsou nezávislé, tak $\mathbb{E}[X_y] = \frac{1}{n}$:

$$\mathbb{E}\left[n_{h(x)}\right] = \overset{\text{prvek } x}{1} + \overbrace{\sum_{y \neq x} \mathbb{E}[X_y]}^{\#\text{ kolizí=délka } n_{h(x)}} = 1 + \frac{s-1}{n} = \mathcal{O}(1)$$

Statický slovník

Příklad (statický slovník) S je dáno předem

- vytvoříme datastrukturu v polynomiálním čase
- chceme, aby operace vyhledání běžela v **maximálním** čase $\mathcal{O}(1)$
  - to jsme předtím neměli – seznam mohl být dlouhý a maximální počet operací velký
- použijeme tabulky dvě:
  - hashujeme dvakrát – jednou pro index do první tabulky, ta určí funkci pro druhé hashování
  - v první budeme chtít $\leq n$ kolizí, ve druhé $= 0$



- vybereme $h \in H$ tak, že má $\leq n$ kolizí
  - kolize $C = \{\{x, y\} \mid x, y \in M, x \neq y, h(x) = h(y)\}$

Lemma: existuje $h \in H$ s počtem kolizí $\leq n$.

Důkaz: $\mathbb{E}[|C|] \overset{2-\text{univ}}{\leq} \binom{s}{2}\frac{1}{n} \overset{s \leq n}{\leq} \binom{n}{2} \cdot \frac{1}{n} \leq \frac{n}{2}$

- jelikož je průměrný počet kolizí $\leq \frac{n}{2}$, tak musí existovat hodně takových, že $\leq \frac{n}{2}$

Lemma: existuje $h_i \in H$ s počtem kolizí 0.

Důkaz: $\mathbb{E}[|C_{n_i}|] \leq \binom{n_i}{2} \cdot \frac{1}{n_i^2} \leq \frac{1}{2}$

- jelikož je průměrný počet kolizí $\leq \frac{1}{2}$, tak musí existovat hodně takových, že 0

(👀): $|C| = \sum_{i=1}^{n} \binom{n_i}{2} = \sum \frac{n_i^2}{2} - \sum \frac{n_i}{2}$

- počet prvků kolidující do daného políčka je n_i, počet dvojic je tedy výraz nahoře

Výpočtem dostáváme $\sum n_i^2 \leq 2|C| + \sum n_i \leq 2s + s = \mathcal{O}(s)$

Testování

Násobení matic

- pomalé násobení: $\mathcal{O}(n^3)$
- nejlepší známé: $\mathcal{O}(n^\omega)$
  - $\omega = 2.37$
- *Vstup:* $A, B, C \subseteq K^{n \times n}$ (pro těleso K)
- *Výstup:* ANO, pokud $A \cdot B = C$, jinak NE

Lemma: nechť $\vec{a} \in K^n, \vec{a} \neq 0$ a $\vec{x} \in \{0,1\}^n$ uniformně náhodný. Pak

$$\Pr_{\vec{x}} \left[\vec{a}^T \cdot \vec{x} \neq 0\right] \geq \frac{1}{2}$$

Důkaz: uvažme poslední nenulovou souřadnici $\vec{a}_k$. Ta má hodnotu 0 nebo $\vec{a}_k$, podle vybraného bitu. 0 bude tedy právě tehdy, když součet předchozích vyšel a_k (a opakujeme s $k-1, \ldots$).

Věta: existuje pravděpodobnostní algoritmus s jednostrannou chybou pro testování maticového násobení v čase $\mathcal{O}\left(n^2\right)$

- když platí, tak vždy řekne že platí
- když neplatí, tak se netrefí s nějakou pravděpodobností (konkrétně $\geq \frac{1}{2}$)

Algoritmus:

1. vezmi náhodný $\vec{x} \in \{0,1\}^n$
2. vstup ANO, jestliže $A \cdot B \cdot \vec{x} = C \cdot \vec{x}$, jinak NE

(👀): algoritmus trvá $\mathcal{O}(n^2)$ kroků

(👀): algoritmus řekne ano $\iff (A \cdot B - C) \cdot \vec{x} = D \cdot \vec{x} = \vec{0}$

- D je nenulová matice (jelikož $A \cdot B \neq C$), má tedy **nenulový řádek**
  - podle lemmatu platí $\Pr_{\vec{x}} [D \cdot \vec{x} \neq 0] \geq \frac{1}{2}$

Nulovost polynomů (Polynomial Identity Testing)

- nezajímá nás, jestli je identicky nulový, ale zda je nulový **v tělese**, ve kterém pracujeme
- uvažujeme více proměnných
  - $d \ldots$ celkový stupeň (t. j. součet stupňů v nějakém nenulovém monomu)
- převeditelné na to, zda je výrok tautologie (jdou na sebe převést) $\implies$ NP těžké

Budeme používat trochu divný vstup:

- *Vstup:* matice polynomů proměnných, determinant určuje náš polynom
- *Výstup:* ANO, jestliže je polynom identicky nulový, jinak NE

Lemma: nechť $P(x_1, \ldots, x_n)$ je **nenulový** polynom nad K stupně $\leq d_i$ a $S \subseteq K$ konečná. Nechť $x_1, \ldots, x_n \in S$ unif. náhodně. Pak

$$\Pr_{\vec{x}} [P(\vec{x}) = 0] \leq \frac{d}{|S|}$$

- $n = 1 \ldots$ polynom má nejvýše d kořenů, ať zvolíme s jakkoliv
- je to dost šikovné, protože podle $|S|$ si volíme přesnost algoritmu (pro $|S| \geq 2d$ máme $\geq \frac{1}{2}$)

Důkaz: pro $n = 1$ platí. Nyní indukcí podle n. Rozdělíme polynom na A a B, kde stupeň v B je ostře menší k. To umíme tím, že vytkneme nějakou proměnnou:

- $P(\vec{x}) = x_1^k \cdot A(x_2, \ldots, x_n) + B(\vec{x})$
  - A je identicky nulový (podle IP) s pravděpodobností $\leq \frac{d-k}{|S|}$
  - chci dokázat, že $\Pr[P(\vec{x}) = 0 \mid A(x_2, \ldots, x_n) \neq 0] \leq \frac{k}{|S|}$
    - při konkrétních hodnotách $x_2, \ldots, x_n$ se mi polynom vyhodnotí na nějaké číslo a zbytek polynomu $P(\vec{x})$ bude $\alpha x_1^k + \beta$, což nebude mít více než k kořenů

Nyní si uvědomíme, že

$$\Pr[P(\vec{x}) = 0] \leq \alpha + \beta \leq \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}$$

Perfektní párování

Nechť (U, V, E) je bipartitní graf, $n = |U| = |V|$. Pak Edmondsova matice grafu je $n \times n$ matice B s

$$B_{u,v} = \begin{cases} x_{u,v} & uv \in E \\ 0 & uv \notin E \end{cases}$$

- za každou hranu bude v matici jedna proměnná

(👀): $\det(B)$ je polynom, jehož monomy vzájemně jednoznačně odpovídají perfektním párovaním.

- sčítáme součin permutace matice a když se zrovna trefíme do párovaní, tak máme monom

Algoritmus (test existence PP)

1. zvolme uniformně náhodně nezávisle $x_{u,v} \in \{1, \ldots, 2n\}$
   - $2n$ kvůli tomu, aby nám vyšlo NE správně s pravděpodobností $\geq \frac{1}{2}$
2. spočítáme determinant
   - pokud je nenulový, párování určitě existuje
   - pokud je nulový, tak párování neexistuje s pravděpodobností $\geq \frac{1}{2}$

Izolující lemma

Věta: Nechť máme systém množin $S_1, \ldots, S_n \subseteq \{a_1, \ldots, a_m\}$ s náhodně zvolenými vahami $w(a_1), \ldots, w(a_m) \in R, |R| = r$. Pak [10]

$$\Pr[\exists \text{ právě jedinná } S_j \text{ s minimální } w(S_j)] \geq 1 - \frac{m}{r}$$

- pro naše použití budeme chtít $r = 2m$

Důkaz: $A_i \ldots$ jev, že existují S_k, S_l tak, ze $w(S_k) = w(S_l) = \min_j w(S_j)$ a $a_i \notin S_k, a_i \in S_l$

- existují dvě minimální množiny, které se liší v prvku i (špatný jev)
- když nenastane žádný s jevů A_i, pak máme vyhráno, jelikož dvě minimální neexistují

Ukážeme, že $\Pr[A_i] \leq \frac{1}{r}$. $S_1, \ldots, S_n$ rozdělíme na dvě množiny podle i:

- $\mathcal{S}_0 = \{j \mid a_i \notin S_j\}$
- $\mathcal{S}_1 = \{j \mid a_i \in S_j\}$

Pokud A_i nastane, pak platí

- pro S_k: $k \in \mathcal{S}_0, w(S_k) = \min_{j \in \mathcal{S}_0} w(S_j)$
- pro S_l: $l \in \mathcal{S}_1, w(S_l) = \min_{j \in \mathcal{S}_1} w(S_j)$

Pak (když zafixujeme všechny váhy a vybíráme váhu a_i) platí

$$\Pr_{w(a_i) \in R}[w(S_k) = w(S_l) \mid w(a_i'), i' \neq i \text{ vybrána}] \leq \frac{1}{r}$$

[10] Prvky a_i budou hrany v grafu a množiny S_j budou perfektní párování. Chceme nějak zvolit váhy a ukázat, že nám nějak jednoznačně identifikují nějakou z množin (tedy perfektních párování).

Součtem pro všechny množiny a dostáním opačného jevu dostáváme hledanou nerovnost.

Algoritmus (rychlý paralelní algoritmus pro PP)

1. zvolíme rovnoměrně náhodně váhy $w(uv) \in \{1, \ldots, 2m\}$ pro každou hranu
2. zasubstituujeme do Edmondsovy matice následně: $x_{uv} = 2^{w(uv)}$
   - $\det(C) \ldots$ příspěvek PP je $\pm 2^{w(M)} = \pm \prod_{uv \in M} 2^{w(uv)}$
     - z definice determinantu (permutace nějakých indexů matice)
3. najdeme W tak, že 2^W je maximální číslo tvaru 2^α dělící $\det(C)$
   - zajímá nás **poslední index, kde má determinant jedničku**, jelikož to odpovídá unikátnímu PP (všechny PP jsou ve tvaru $0b1\underbrace{0000}_{w(uv)}$)
4. pro $uv \in E$ spočítáme $d = \det(C^{uv})$
   - jestliže $2^{W-w(uv)}$ je max. číslo tvaru 2^α dělící d, pak přidáme uv do M
     - odpovídá tomu, zda párování přežilo odstranění hrany – pokud ne, tak ho přidáme
5. zkontrolujeme, že M je PP (mohli jsme vygenerovat nesmysl)

TODO: algoritmus pro obecné grafy přes Tutteho matici

Odkazy

- Webová stránka předmětu
- Odkaz na skripta (pozor, jsou vcelku nedopsaná)