Úvod do aproximačních a pravděpodobnostních algoritmů

Karel Velička

6. února 2025

Vyučující: *prof. RNDr. Jiří Sgall, DrSc., doc. Mgr. Petr Kolman, Ph.D.*

Obsah

1 Úvod a základní definice

Definice 1.1. ***(Optimalizační problém)*** definujeme jako $(\mathcal{I}, \mathcal{F}, f, g)$, kde:

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

Definice 1.2. ***(NP-Optimalizační problém)*** definujeme jako $(\mathcal{I}, \mathcal{F}, f, g)$, pro které platí stejné vlastnosti 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 je počitatelná v polynomiálním čase.

Definice 1.3. ***(R-aproximace)****:* Algoritmus A je R-aproximační, 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: $\forall I: \ f(A) \geq \text{OPT}(I)/R$.

Třídy jazyků a Pravděpodobnostní algoritmy

Definice 1.4. ***(Třída problémů NP)*** je třída rozhodovacích problémů, v níž problém L leží $\iff$ pokud $\exists K \in P$ problém a $\exists g$ polynom, přičemž $\forall x$ vstupy je $L(x) = 1 \iff$ pokud pro nějaký řetězec y délky nejvýše $g(|x|)$ platí $K(x, y) = 1$.

Definice 1.5. ***(NP-těžký problém)*** $L \equiv$ je-li na něj převoditelný každý problém z NP.

Definice 1.6. ***(NP-úplný problém)*** $L \equiv$ pokud je NP-těžký a zároveň L leží v NP.

Definice 1.7. ***(Zero-error Probabilistic Poly time - ZPP)****:* Algoritmus má vždy správný výstup a běží v poly čase.

Definice 1.8. ***(Randomized Poly time - RP)****:* Algoritmus může omylem odmítnout správnou odpověď, ale nikdy nemá falešný pozitivní výsledek. Pravděpodobnost chyby je nejvýše $1/2$. Běží v poly čase.

Definice 1.9. ***(Bounded-error Probabilistic Poly time - BPP)****:* Algoritmus může mít falešně pozitivní i negativní výsledky. Pokud je srávná odpověď je TRUE, algoritmus přijme s pravděpodobností alespoň $2/3$, naopak pokud je FALSE, tak s pravděpodobností $< 1/3$.

2 Metrický TSP

Definice 2.1. ***(Metrika)*** na množině V je funkce $f : V \times V \to \mathbb{R}_0^+$ taková, že:

(i) $\forall x, y \in V : d(x, y) = 0 \iff x = y$,

(ii) $\forall x, y \in V : d(x, y) = d(y, x)$,

(iii) $\forall x, y, z \in V : d(x, z) \leq d(x, y) + d(y, z)$.

Problém 2.1. ***(Obchodního cestujícího - TSP):*** Dostaneme úplný ohodnocený graf. Cílem je najít nejmenší Hamiltonovský cyklus. Nemůže být aproximována, za předpokladu $P \neq NP$.

V algoritmech budeme používat pojem **'zkracování'**, to znamená, že budeme vracet vždy jen první výskyt vrcholu v eulerovském tahu. Například $ET = (1, 3, 4, \cancel{3}, 2, \cancel{1}, 6) \rightsquigarrow HT = (1, 3, 4, 2, 6)$.
Cíl je minimalizovat $d(\mathcal{C})$.

Algorithm 1 Metric-TSP

Input: $G = (V, E)$, $d : V \times V \to \mathbb{R}_0^+$
Output: Cyklus $\mathcal{C} = V_1, \ldots, V_n$ zpermutovaných vrcholů

1: Najdi minimální kostru T.
2: Zdvojnásob každou hranu a najdi Eulerovský tah $\mathcal{T}$. ▷ *každá hrana právě jednou*
3: Vypiš Hamiltonovský cyklus $\mathcal{C}$ po zkracování.

Věta 2.1. *Algoritmus Metrický TSP je 2-aproximační*

Důkaz. Víme, že $\text{cost}(T) \leq OPT$. Protože jinak pokud vynecháme hranu z OPT, dostaneme kostru. Ta ale nemůže být menší než naše nejmenší kostra T. Výsledný cyklus je díky zkracování $\text{cost}(\mathcal{C}) \leq \text{cost}(\mathcal{T})$. Jelikož $\mathcal{T}$ obsahuje každou hranu T dvakrát, tak $\text{cost}(\mathcal{T}) = 2 \cdot \text{cost}(T)$.
Celkem tak dostaneme $\text{cost}(\mathcal{C}) \leq \text{cost}(\mathcal{T}) = 2 \cdot \text{cost}(T) \leq 2 \cdot OPT$. ■

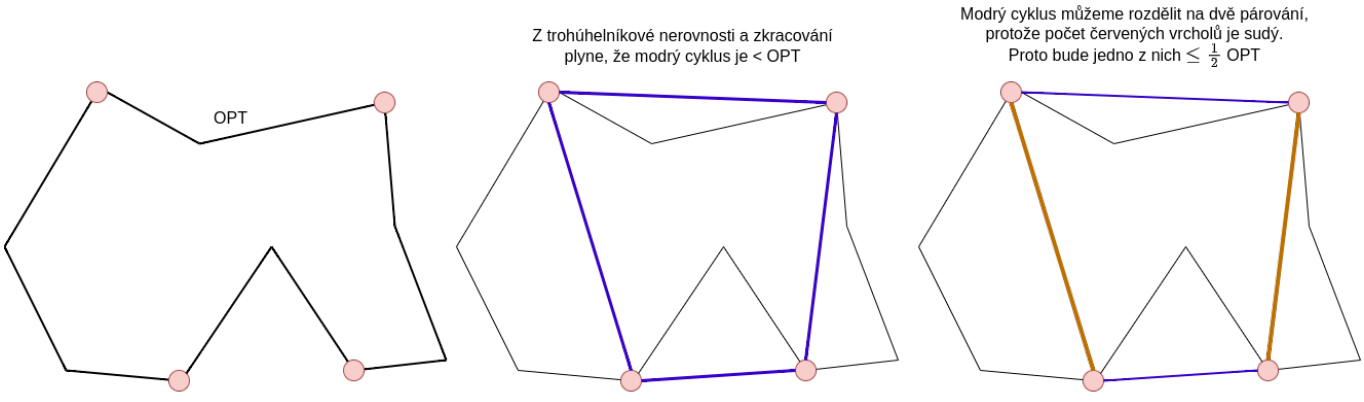
Algorithm 2 Christofidesův algoritmus

1: Najdi minimální kostru T a označme W vrcholy lichého stupně T.
2: Najdi minimální perfektní párování M na W
3: Najdi Eulerovský tah $T \cup M$ a proveď zkracování.
4: Vypiš vzniklý cyklus $\mathcal{C} = T \cup M$.

Věta 2.2. *Algoritmus Metric-TSP je 3/2-aproximační.*

Důkaz. Víme, že $\text{cost}(T) \leq OPT$. Zároveň víme, že $\text{cost}(M) \leq \frac{OPT}{2}$ (viz. Obr.), proto:

$$\text{cost}(T \cup M) \leq \text{cost}(T) + \text{cost}(M) \leq OPT + \frac{1}{2}OPT = \frac{3}{2}OPT.$$



(a) Původní graf (OPT... černá) (b) Provedli jsme zkracování (modrá) (c) Našli jsme perfektní párování (oranžová)

■

3 Diskrétní pravděpodobnost

Definice 3.1. (Diskrétní pravděpodobnostní prostor) je dvojice (Ω, P), kde Ω je konečná nebo spočetná množina elementárních jevů a P je pravděpodobnostní funkce $P : \Omega \to [0, 1]$ taková, že $\sum_{w\in\Omega} P[w] = 1$.

Definice 3.2. (Uniformní pravděpodobnostní prostor). *Uniformní pravděpodobnostní prostor* pro konečnou Ω je každý elementární jev $A \in \Omega : P[A] = \frac{1}{|\Omega|}$.

Definice 3.3. (Geometrický pravděpodobnostní prostor). *Geometrický pravděpodobnostní prostor* pro spočetnou Ω je je každý elementární jev $A \in \Omega : P[A] = \frac{1}{2^A}$.

Definice 3.4. (Jev). *Jev* je podmnožina $A \subseteq \Omega$.

Definice 3.5. (Pravděpodobnost). *Pravděpodobnost* jevu A je $P[A] = \sum_{w\in A} P[w]$.

Pozorování 3.1. *Pravděpodobnost pro disjunktní jevy $A, B \subseteq \Omega$ je $P[A \cup B] = P[A] + P[B]$.*

Definice 3.6. (Náhodná veličina). *Náhodná veličina* na (Ω, P) je libovolná funkce $X : \Omega \to \mathbb{R}$.

Definice 3.7. (Střední hodnota). *Střední hodnota* náhodné veličiny X je $\mathbb{E}(X) = \sum_{w\in\Omega} P[w] \cdot X(w)$.

Lemma 3.1. ***(Linearita střední hodnoty).*** *Nechť X, Y jsou nezávislé veličiny a $\alpha \in \mathbb{R}$, potom:*

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y] \quad a \quad \mathbb{E}[\alpha X] = \alpha\mathbb{E}[X].$$

Definice 3.8. (Indikátor náhodného jevu). *Indikátor náhodného jevu A* je náhodná veličina:

$$Y_A : \Omega \to \{0, 1\}, \qquad Y_A(w) = \begin{cases} 1 & \text{pokud } w \in A \text{ } \textit{(pokud jev nastal)} \\ 0 & \text{pokud } w \notin A \text{ } \textit{(pokud jev nenastal)} \end{cases}$$

Definice 3.9. (Podmíněná pravděpodobnost). *Podmíněná pravděpodobnost* je pravděpodobnost, že nastal jev A za podmínek, že nastal jev B.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Věta 3.1. ***(O úplné pravděpodobnosti).*** *Nechť A je jev a $B_1, ..., B_k$ je rozklad Ω na disjunktní jevy. Potom:*

$$P(A) = \sum_{i}^{k} P[A|B_i] \cdot P(B_i)$$

Důkaz.

$$P(A) = P\left[A \cap \bigcup_{i=1}^{k} B_i\right] = \sum_{i=1}^{k} P[A \cap B_i]$$

■

Definice 3.10. (Nezávislé jevy). Jevy A a B jsou *nezávislé* $\iff P(A \cap B) = P(A) \cdot P(B)$.

4 Randomized Quicksort

Problém 4.1. ***(Randomizovaný Quiksort)***

Algorithm 3 QS(S) , $\mathcal{O}(n \log n)$

Input: S množina n čísel
Output: Setříděná množina S

1: **if** $|S| \leq 1$ **then return** S
2: Vyber uniformě náhodně pivot $p \in S$
3: Rozděl S na $A = \{x \in S \mid x < p\}, B = \{x \in S \mid x > p\}$
4: **return** QS(A), p , QS(B)

Pozorování 4.1. *Quicksort běží v nejhorším případě* $\Omega(n^2)$.

Věta 4.1. *Pro každý vstup je očekávaný počet porovnání nejvýše* $2nH_n$.

Důkaz. Nechť X je náhodná veličina udávající počet porovnání.
Zafixujeme vstupní posloupnost a počítáme $A_{i,j} = \Pr$ [porovnáme i-tý a j-tý prvek].
To, že se dva prvky porovnají musí znamenat, že jeden z jich byl pivot:
Protože kdyby pivot $p > i, j$ respektive $p < i, j$, tak bychom postoupili do další úrovně rekurze.
Kdyby $i < p < j$, tak to prvky i, j rozdělí – i půjde do levé větve a j do pravé – takže se i, j nemůžou porovnat. Proto, aby se porovnaly, musí platit, že $p = i$, nebo $p = j$.
Možností voleb p, že prvky i, j nepostoupí do další úrovně rekurze je $j + 1 - i$. Z toho 2 volby vedou k porovnání. Dostaneme tak pro $i < j$, že

$$\Pr[A_{i,j}] = \frac{2}{j - i + 1}.$$

Nechť $X_{i,j} = \begin{cases} 1 & A_{i,j} \text{ nastane,} \\ 0 & \text{jinak} \end{cases}$ je indikátorové veličina. Střední hodnota počtu porovnání je pak:

$$\mathbb{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{i,j}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[A_{i,j}] =$$
$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1} \leq \sum_{i=1}^{n-1} \sum_{k=2}^{n} \frac{2}{k} \leq 2n \cdot \sum_{k=1}^{n} \frac{1}{k} = 2nH_n$$

■

Pozorování 4.2. *Jelikož Harmonická řada* $H_n \approx \ln n$, *dostáváme tak průměrný čas* $\mathcal{O}(n \log n)$.

5 Konflikty v distribuovaném systému

Mějme n synchronizovaných procesů $P_1, \ldots, P_n$ a jednu sdílenou paměť. Všechny tyto procesory chtějí zapisovat do jedné buňky. To se povede vždy jen jednomu procesoru. Zkouší to v každém cyklu. Cílem je navrhnout protokol tak, aby po t krocích s velkou pravděpodobností získal každý procesor přístup.

Algorithm 4 ConflictDistSystem

1: V každém cyklu zkus s pravděpodobností p přistoupit do databáze
2: Opakuj, dokud se ti to nepovede

Věta 5.1. *Pro algortimus s pravděpodobností $p = \frac{1}{n}$ platí, že s pravděpodobností alespoň $1 - \frac{1}{n}$ všechny procesy uspějí v prvních $t = 2en \ln n$ kolech.*

Důkaz. Označme $A_{i,t}$ jev, že i-tý proces uspěl v kole t. Potom platí [1]:

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

Označme $F_{i,t}$ jev, že i-tý proces neuspěl v žádném z $1 - t$ kol. Potom tedy:

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

Nyní dosadíme hodnotu $t = 2en \ln n$, aby nám to hezky vycházelo: $\left(\frac{1}{e}\right)^{\frac{t}{en}} = \left(\frac{1}{e}\right)^{\frac{2\cancel{en}\ln n}{\cancel{en}}} = \frac{1}{n^2}$.
Pravděpodobnost, že nějaký proces P_i neuspěje v prvních t kolech je $\leq \sum_{i=1}^{n} \Pr[F_{i,t}] \leq n \cdot \frac{1}{n^2} = \frac{1}{n}$. ■

6 Globální minimální řez

Na vstupu dostaneme neorientovaný multigraf $G = (V, E)$ a na výstupu bude řez $\emptyset \neq S \subseteq E$. Cílem je minimalizovat velikost řezu $|S|$.

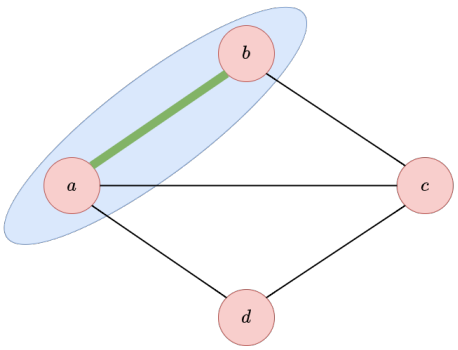
Algorithm 5 Tokový $\mathcal{O}(n^3)$

1: Převedeme graf na ohodnocený s jednotkovými kapacitami.
2: Zafixujeme vrchol s.
3: Pro všechny ostatní vrcholy t najdeme minimální st-řez.
4: **return** minimum z nich

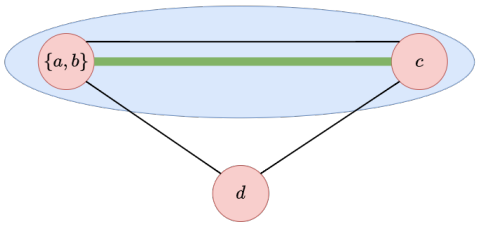
Algoritmus na $\binom{n}{2} \approx n^2$ párech počítá minimální st-řez (v $\mathcal{O}(n)$). Proto celkem $\mathcal{O}(n \cdot n^2) = \mathcal{O}(n^3)$.

Algorithm 6 Contract $\mathcal{O}(n^2 \log n)$

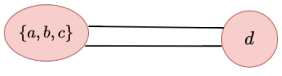
1: Vyber uniformě náhodně hranu a její vrcholy zkontrahujeme do jednoho
2: Opakujeme, dokud nemáme pouze dva vrcholy
3: Zbylé hrany na konci jsou náš řez



(a) Máme graf G



(b) Kontrahujeme b, takže $G_1 = G \setminus \{a, b\}$



(c) Kontrahujeme c, takže $G_2 = G \setminus \{a, b, c\}$

Na obrázku nám zůstaly pouze dva vrcholy, označme proto globální řez například $S = \{a, b, c\}$.

Pozorování 6.1. *Každý řez v $G \setminus \{e\}$ odpovídá řezu v G.*

[1] Geometrické rozdělení $p(1-p)^{n-1}$ - prvních $n-1$ pokusů neúspěšných, n-tý pokus úspěšný.

Pozorování 6.2. *Pokud C je řez v G a $e \notin C$, pak C je řez v $G \setminus \{e\}$.*

Lemma 6.1. *Multigraf s n vrcholy a minimálním řezem velikosti k má alespoň $\frac{1}{2}nk$ hran.*

Důkaz. Pro každý vrchol $v \in V$, kde $|V| = n$ hrany incidentní s v tvoří řez. A tedy $\forall v : \deg(v) \geq k$. Kdyby ne, tak existuje řez s velikostí menší než k.
Víme, že počet hran je roven polovině ze součtu stupňů vrcholů, tedy:

$$|E| = \frac{1}{2}\sum_{v} \deg(v) \geq \frac{1}{2}nk.$$

■

Věta 6.1. *Pravděpodobnost, že najdeme daný minimální řez C je alespoň $\binom{n}{2}^{-1} = \frac{2}{n\cdot(n-1)}$*

Důkaz. Zafixujme minimální řez C.
Označme A_i jev, že v prvních i iteracích jsme **ne**vybrali hranu z C.
V počátečním stavu zjevně platí:

$$\Pr[A_0] = 1,$$

protože jsme nevybrali žádnou hranu.
Nás ovšem zajímá $\Pr[A_{n-2}]$ – začal s n vrcholy a končí se dvěma – celkem udělá $n-2$ iterací.
$\Pr[A_1] \geq 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}$, neboli alespoň 1 mínus pravděpodobnost, že vybereme hranu z C.
$\Pr[A_2 \mid A_1] \geq 1 - \frac{k}{(n-1)k/2} = 1 - \frac{2}{n-1}$, je tam už pouze $n-1$ vrcholů a z podmíněné pravděpodobnosti dostaneme: $\Pr[A_2] = \Pr[A_2 \mid A_1] \cdot \Pr[A_1]$. Pro i-tý krok tak platí, že $\Pr[A_{i+1} \mid A_i] \geq 1 - \frac{2}{n-i}$.
Celkem tedy můžeme určit:

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

■

Důsledek 6.1. *Každý graf G má nejvýše $\binom{n}{2}$ globálních minimálních řezů.*

Věta 6.2. *Když budeme algoritmus opakovat $\binom{n}{2}\log n$-krát, tak dostaneme globální minimální řez s pravděpodobností $\geq 1 - \frac{1}{n}$.*

Důkaz.

$$\left(1 - \binom{n}{2}^{-1}\right)^{\binom{n}{2}\log n} = \left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}\log n} \leq \left(\frac{1}{e}\right)^{\log n} = \frac{1}{n}.$$

■

Důsledek 6.2. *Celkový počet globálních minimálních řezů je $\leq \frac{1}{\frac{2}{n(n-1)}} = \frac{n(n-1)}{2} = \binom{n}{2}$.*

7 Hladové algoritmy

- *Vstup:* m strojů, n úloh, každá trvá p_i.
- *Výstup:* Rozvržení úloh stroje. Rozklad množiny $\{1, \ldots, n\} = I_1, \ldots, I_m$. *(na disjunktní množiny)*
- *Cíl:* $\max\limits_{i=1,\ldots,m} \sum\limits_{j \in I_i} p_j$, neboli maximalizovat délku rozvrhu stroje

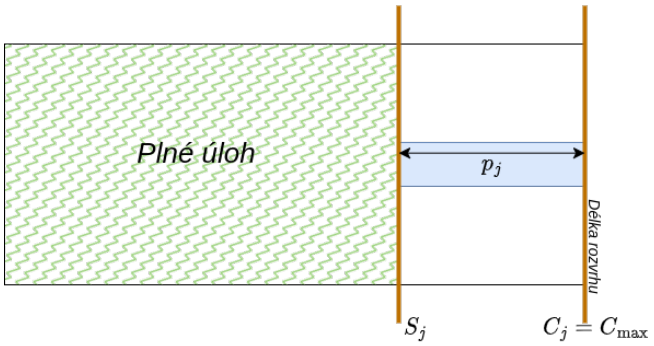
Každá úloha může být zpracována jen na jednom ze strojů a každý stroj může zpracovávat nejvýše jednu úlohu najednou.

Notace: Pro $I_1, \ldots, I_m$ a úlohu $j \in I_i$:

- $S_j = \sum_{k \in I_i;\ k<j} p_k$ je čas začátku úlohy j
- $C_j = S_j + p_j$ je čas konce úlohy j
- $C_{\max} = \max_j C_j$, neboli délka rozvrhu
- $C_{\min} = \min_i \max_{j \in I_i} C_j$

Algorithm 7 Lokální prohledávání rozvrhu

1: Začni s nějakým rozvrhem
2: **if** $\exists j : C_j = C_{\max}$ & $S_j > C_{\min}$ **then**
3: Přeřaď úlohu j na stroj s minimální dobou dokončení.
4: **else return** aktuální rozvrh
5: Opakuj krok 2.



Pozorování 7.1. *Platí* $\forall j \in [n] : OPT \geq p_j$. *Protože* p_j *je součástí* OPT.

Pozorování 7.2. *Platí* $OPT \geq \frac{1}{m} \sum_{i=1}^{n} p_i \geq C_{\min} \geq S_j$

Věta 7.1. *Algoritmus je* $(2 - \frac{1}{m})$*-aproximační.*

Důkaz. Odhad zlepšíme na $S_j \leq \frac{1}{m} \left(\sum_{i=1}^{n} p_i - p_j \right)$, protože alg před časem S_j nepracuje na úloze j.

$$C_{\max} = S_j + p_j \leq \frac{\sum_{i=1}^{n} p_i}{m} - \frac{p_j}{m} + p_j = \frac{\sum_{i=1}^{n} p_i}{m} + \left(1 - \frac{1}{m}\right) p_j \overset{Poz.7.1.+7.2.}{\leq} \tag{1}$$

$$\leq OPT + \left(1 - \frac{1}{m}\right) OPT = \left(2 - \frac{1}{m}\right) OPT. \tag{2}$$

■

Algorithm 8 List scheduling - Hladový

1: Libovolně uspořádej úlohy
2: Zpracovávej úlohy jednu po druhé a přiřaď úlohu na nejméně načtený stroj

Věta 7.2. *Algoritmus je* $(2 - \frac{1}{m})$*-aproximační.*

Online algoritmy Vstup přichází postupně. Řešení musíme konstruovat také po krocích a pak už ho nesmíme měnit. Hladový algoritmus je online, ale Lokální prohledávání není.

7.1 LPT - Largest Processing Time first

Algorithm 9 LPT

1: Úlohy uspořádáme tak, že $p_1 \geq p_2 \geq \ldots \geq p_n$.
2: Použijeme hladový List Scheduling algoritmus.

Věta 7.3. *Algoritmus je $\frac{4}{3}$-aproximační.*

Důkaz. BÚNO předpokládejme, že p_n skončil jako poslední a určuje tak délku rozvrhu. Nechť:

(1) $p_n \leq \frac{1}{3}OPT$: Jelikož $S_n \leq C_{\min} \leq OPT$, tak $C_{\max} = S_n + p_n \leq OPT + \frac{1}{3}OPT = \frac{4}{3}OPT$.

(2) $p_n > \frac{1}{3}OPT$: V optimu rozvrhu jsou nejvýše 2 úkoly na každém stroji.

Pokud $n \geq m+1 : OPT \geq p_m + p_{m+1}$, protože OPT pro prvních $m+1$ úloh má 2 úlohy na stejném stroji a ty jsou velké alespoň jako p_m, p_{m+1}. Pokud $n \geq m+2 : OPT \geq p_{m-1} + p_{m+2}$, protože OPT pro prvních $m+2$ úloh má 2 dvojice na stejném stroji. Alespoň jedna ve dvojici je velikosti alespoň p_{m-1}.

Tedy pro $\begin{cases} n = m+i & OPT \geq p_{m-i+1} + p_{m+i} \\ n = m+(n-m) & OPT \geq p_{m-(n-m)+1} + p_{m+(n-m)} = p_{1-m} + p_n \rightsquigarrow \frac{2}{3}OPT > p_{1-m} \end{cases}$

Můžeme si tedy všimnout, že alespoň $n-m$ úkolů z $p_1, \ldots, p_m$ má délku $< \frac{2}{3}OPT$.

■

7.2 Bin packing

- *Vstup:* n věcí $a_1, \ldots, a_n \in [0,1]$
- *Výstup:* Rozvržení $\{1, \ldots, n\}$ na $I_1, \ldots, I_m$, že $\forall i : \sum_{j \in I_i} a_j \leq 1$. ($\sum$ věcí v každém koši ≤ 1)
- *Cíl:* minimalizovat m, tedy počet košů.

Algorithm 10 Hladový - First fit *"dej a_j do prvního koše, do kterého se vejde."*

1: **for all** $j = 1 \ldots n$ **do**
2: Nechť i je první koš, do kterého se věc j vejde.
3: Vložme věc j do koše i.
4: **if** není takový koš **then**
5: Přidáme nový koš.

Best fit: Říká, dej a_j do nejplnějšího koše, do kterého se vejde.

Věta 7.4. *Každý* Any Fit *algoritmus je 2-aproximační.*

Důkaz. Předpokládejme $I_1, \ldots, I_m$ zkonstruované algoritmem. Jelikož pro jeden koš je $OPT = 1$, tak nechť máme alespoň 2 koše. Označme $B_i = \sum_{j \in I_i} a_j$ jako velikost i-tého koše.

Musí platit, že $\forall i, j$, že $i \neq j : B_i + B_j > 1$ a $B_1 + B_m > 1$. Sečteme tedy všechny dvojice:

$$(B_1 + B_2) + (B_2 + B_3) + \ldots + (B_{m-1} + B_m) + (B_m + B_1) = 2(B_1 + \ldots + B_m) = 2\sum_{i=1}^{m} B_i$$

A jelikož pro jednotlivý součet dvojice platí, že je > 1, tak:

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

■

7.3 Hledání disjunktních cest

Bez kapacity:

- *Vstup:* $G = (V, E)$, k párů $(s_1, t_1), \ldots, (s_k, t_k) \in V^2$
- *Výstup:* $I \subseteq \{1, \ldots, k\}$ spolu s cestou P_i pro $i \in I$ takovou, že P_i spojuje s_i a t_i a každá hrana je použita nejvýše jednou cestou.
- *Cíl:* $\max |I|$, neboli počet dvojic, které pospojujeme

Algorithm 11 Hladový

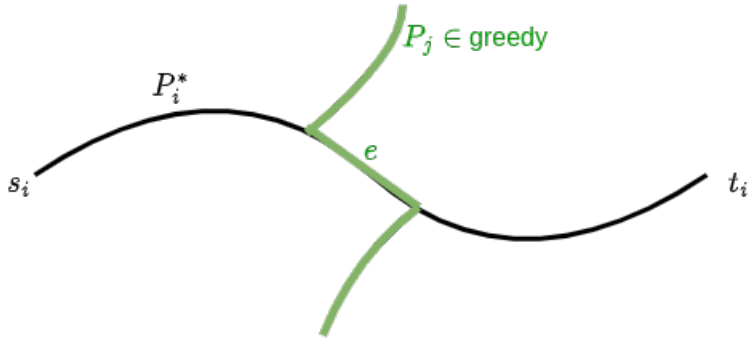
$I \leftarrow \emptyset$
for all $i \in [k] \setminus I$ **do**
  Nechť P_i je nejkratší $s_i t_i$-cesta v G
  Najdi $j \in [k] \setminus I$, že $|P_j| \leq |P_i|$ pro každé $i \in [k] \setminus I$
  if neexistuje taková cesta **then return** I a P_i $(\forall i \in I)$
  $I \leftarrow I \cup \{j\}$, $G \leftarrow G \setminus P_j$
return I a P_i $(\forall i \in I)$

Věta 7.5. *Algoritmus je* $2\sqrt{m} + 1$*-aproximační, kde* $m = |E| = \Omega(k^2)$ *pro* $OPT = k$.

Důkaz. Předpokládejme optimální řešení $OPT \geq 1$ a $|I| = ALG \geq 1$. Nechť P_i^* je optimální $s_i t_i$-cesta pro $i \in OPT$. Počítejme cesty a rozdělme je na dva druhy – dlouhé OPT_l a krátké OPT_s:

$$OPT_l = \{|P_i^*| \geq \sqrt{m} \mid i \in OPT\}, \quad OPT_s = OPT \setminus OPT_l.$$

Můžeme si všimnout, že $|OPT_l| \leq \sqrt{m}$. Protože máme $\sqrt{m} \cdot \sqrt{m} = m$ neopakujících se hran. Kdyby jich bylo více než $\sqrt{m}$, tak bychom použili více hran, než je v grafu.
Dlouhé cesty tedy nic nekazí, protože najdou alespoň jednu cestu. Což nám stačí.
Krátké cesty, kde $i \in OPT_s$ jsou také v pořádku, protože tuto $s_i t_i$-cestu alg spojil. Je jich $|I|$.
Předpokládejme ale P_i^* pro $i \in OPT_s \setminus I$.
Potom existuje společná hrana $e \in P_i^*$ s nějakou cestou P_j (zvolena hladově), že $|P_j| \leq |P_i^*| \leq \sqrt{m}$.



Cesta P_j blokuje P_i^* a nemůžeme tak vybrat cestu delší než $\sqrt{m}$.
Počet krátkých cest P_i^* je proto $|OPT_s \setminus I| \leq \sqrt{m} \cdot |I|$ a tedy:

$$|OPT| = |OPT_l| + \underbrace{|OPT_s|}_{|OPT_s \setminus I| + |I|} \leq \underbrace{\sqrt{m}}_{\text{dlouhé}} + \underbrace{\sqrt{m} \cdot |I| + |I|}_{\text{krátké}} \leq (2\sqrt{m} + 1)|I|.$$

■

S kapacitami:

Každá hrana má celočíselnou kapacitu $c \geq 1$. Pro $c = 1$ bychom měli $\beta = \sqrt{m}$.

Algorithm 12 HLADOVÝ S KAPACITAMI

$I \leftarrow \emptyset$, $\beta := \left\lceil m^{\frac{1}{c+1}} \right\rceil$, $\forall e \in E : d(e) = 1$.
for all $i \in [k] \setminus I$ **do**
  Nechť P_i je d-nejkratší $s_i t_i$-cesta v G
  Najdi $j \in [k] \setminus I$, že $d(P_j) \leq d(P_i)$ pro každé $i \in [k] \setminus I$
  if neexistuje takové j nebo $d(P_j) \geq \beta^c$ **then return** I a P_i $(\forall i \in I)$
  $I_i \leftarrow I \cup \{j\}$, $\forall e \in P_j : d(e) := d(e) \cdot \beta$
return I a P_i $(\forall i \in I)$

Věta 7.6. *Algoritmus je* $3c \cdot m^{\frac{1}{c+1}} + 1$*-aproximační, tedy* $\mathcal{O}(m^{\frac{1}{c+1}}) = \mathcal{O}(\beta)$.

Důkaz. Předpokládejme optimální řešení $OPT \geq 1$ a $|I| = ALG \geq 1$. Nechť P_i^* je optimální $s_i t_i$-cesta pro $i \in OPT$. Nechť d_i je délka funkce na konci iterace i.
Opět rozdělíme cesty na krátké a dlouhé:
Říkáme, že cesta je *krátká*, pokud $d(P) = \sum_{e \in P} d(e) \leq \beta^c = m^{\frac{c}{c+1}}$.
Nechť l je poslední iterace, ve které hladový algoritmus vybal krátkou cestu a označme $\bar{d} = d_l$.
Z *pozorování 7.3.* a *pozorování 7.4.* vyplývá, že:

$$|OPT \setminus I| \leq \frac{c \cdot \bar{d}(E)}{\beta^c} = \frac{c \cdot m(1 + 2|I|)}{m^{\frac{c}{c+1}}} = \\ = c \cdot m^{\frac{1}{c+1}}(1 + 2|I|) \leq c \cdot m^{\frac{1}{c+1}} \cdot 3|I|.$$

Takže $|OPT| \leq |OPT \setminus I| + |I| \leq \left(3c \cdot m^{\frac{1}{c+1}} + 1\right)|I|$. ■

Pozorování 7.3. $\bar{d}(E) \leq m(1 + 2|I|)$ ´

Důkaz. Každá hrana má na počátku algoritmu $d(e) = 1$, proto i $d(E) = m$. Kdyby byla delší, tak by to značilo, že už byla vybána (byla by přenásobena faktorem β). Takže pro krátkou cestu po výběru:

$$\bar{d}(E) \leq m + \sum_{i \in I} \beta^c \cdot \beta \leq m + \sum_{i \in I} 2m = m + 2m|I|.$$

■

Pozorování 7.4. *Pro každou cestu* $P \in OPT \setminus I$ *je* $\bar{d}(P) \geq \beta^c$.

Důkaz. Pro spor předpokládejme, že pro nějakou cestu P je její délka $d(P) < \beta^c$.
Každá hrana na P je použita $\leq c-1$ cestami z hladového algoritmu $\implies$ P je dostupná pro hladový algoritmus, ale ten ji nepoužil $\implies$ spor. ■

8 SAT

- *Vstup:* n boolovských proměnných $x_1, \ldots, x_n$ a m klauzulí $C_1, \ldots, C_m$ s vahou w_j.
- *Výstup:* Hodnota True/False vzhledem k $x_1, \ldots, x_n$
- *Cíl:* maximalizovat váhy příslušných klauzulí, tedy $\max \sum_{j=1}^{m} w_j$

Předpokládáme, že se žádný literál v klauzuli neopakuje a že nejvýše jeden z $x_i, \bar{x}_i$ se vyskytuje v C_i

8.1 RAND-SAT

Algorithm 13 RAND-SAT

1: Pro $\forall i \in [n]$ nezávisle náhodně nastav $x_i = \begin{cases} \text{True} & \Pr = \frac{1}{2}, \\ \text{False} & \Pr = \frac{1}{2}. \end{cases}$

Věta 8.1. *Algoritmus je 2-aproximační.*

Důkaz. Pro každou klauzuli C_j zavedeme indikátorovou proměnnou Y_j s 1 =nesplněna, 0 =splněna.
Pro k literálů je $\Pr[C \text{ není splněna}] = \frac{1}{2^k} = \sum Y_j$.
Víme, že $k \geq 1$ a že platí $\mathbb{E}[Y_j] = \Pr[C \textit{ je splněna}] = 1 - \frac{1}{2^k} \geq \frac{1}{2}$. A tedy:

$$W = \sum_{j=1}^{m} w_j Y_j, \qquad \mathbb{E}[W] \overset{\text{linearita}}{=} \sum_{j=1}^{m} \mathbb{E}[Y_j] w_j \geq \frac{1}{2} \sum_{j=1}^{m} w_j \geq \frac{1}{2} OPT.$$

■

8.2 BIASED-SAT

Předpokládáme, že $\forall i : \underbrace{\sum_{j:C_j = x_i} w_j}_{\Sigma_+} \geq \underbrace{\sum_{j:C_j = \bar{x}_i} w_j}_{\Sigma_-}$.

Algorithm 14 BIASED-SAT

1: Pro $\forall i \in [n]$ nezávisle náhodně nastav $x_i = \begin{cases} \text{True} & \text{s pravděpodobností } p > \frac{1}{2}, \\ \text{False} & 1 - p. \end{cases}$

Nechť U je množina klauzulí bez záporných jednotkových klauzulí.

Pozorování 8.1. $OPT \leq \sum_{j \in U} w_j$.

Důkaz. Používáme předpoklad, že $\sum_+ w_j \geq \sum_- w_j$

$$\mathbb{E}\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j \mathbb{E}[Y_j] \geq \sum_{j=1}^{m} w_j \Pr[C_j \text{ je splněná}] \geq$$
$$\geq \sum_{j=1}^{m} w_j \cdot p \geq p \cdot OPT.$$

■

Věta 8.2. *Algoritmus je φ-aproximační.*

Důkaz. Uvažme klauzuli C_j délky k_j a indikátorovou veličinu Y_j.
Pokud je $k = 1$, tak dostaneme kladný literál: $Y_j = p$. Pro $k_j \geq 2$:
Označme a počet záporných literálů a b počet kladných literálů. $\Pr[C \text{ nesplněná}] = p^a (1-p)^b$.

$$Y_j = 1 - p^a (1-p)^b \overset{p > \frac{1}{2}}{\geq} 1 - p^{a+b} \overset{k_j \geq 2}{\geq} 1 - p^2.$$

Máme tedy $p = 1 - p^2$ a dostáváme tak $p = \varphi = \frac{\sqrt{5}-1}{2}$. ■

8.3 LP-SAT

Algorithm 15 LP-SAT

1: Pro každou proměnnou x_i si pořídíme binární proměnnou $y_i \in \{0,1\}$ a pro každou klauzuli C_j binární proměnnou $z_j \in \{0,1\}$.
2: Vytvoř lineární program *(viz. LP níže)*
3: Relaxuj program a vyřeš ho (dostaneme optimum y^*, z^*)
4: Nastav proměnné x_i na True s pravděpodobností y_i^* a False s $(1 - y_i^*)$.

Lineární Program:

- *Účelová funkce:* $\max \sum_{j=1}^{m} z_j$
- *Proměnné:* $y_i \in \{0,1\}$, $z_j \in \{0,1\}$, (negaci zapisujeme jako $1 - y_i$)
- *Podmínky:* $z_j \leq \sum_{+} y_i + \sum_{-} (1 - y_i)$

Fakt A. (A/G nerovnost). Pro každé nezáporné $a_1, \ldots, a_n$: $\quad \prod_{i=1}^{n} a_i^{\frac{1}{n}} \leq \frac{1}{n} \sum_{i=1}^{n} a_i$

Fakt B. (Konvexní funkce). Pokud je funkce f 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). $\quad \left(1 - \frac{1}{n}\right)^n \leq \frac{1}{e}$

Věta 8.3. *Algoritmus je* $(1 - \frac{1}{e})$*-aproximační*

Důkaz. Uvažme y^*, z^* a C_j s délkou k_j; potom:

$$\Pr[C_j \text{ není splněná}] = \overbrace{\prod_{i:x_i \in C_j} (1 - y_i^*)}^{\text{kladné}} \overbrace{\prod_{i:\overline{x}_i \in C_j} y_i^*}^{\text{záporné}} \overset{A}{=}$$

$$\overset{A}{=} \left[\frac{1}{k_j}\left(\sum_{i:x_i \in C_j} (1 - y_i^*) + \sum_{i:\overline{x}_i \in C_j} y_i^*\right)\right]^{k_j} =$$

$$= \left[1 - \frac{1}{k_j}\left(\sum_{i:x_i \in C_j} y_i^* + \sum_{i:\overline{x}_i \in C_j} (1 - y_i^*)\right)\right]^{k_j} \leq \qquad \text{// definice } z^* \text{ v LP}$$

$$\leq \left(1 - \frac{z_j^*}{k_j}\right)^{k_j}$$

Nás zajímá splnění, tedy:

$$\Pr[C_j \text{ je splněná}] \geq \overbrace{1 - \left(1 - \frac{z_j^*}{k_j}\right)^{k_j}}^{f(z_j^*)} \overset{B}{\geq}$$

$$\overset{B}{\geq} \left[1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right] \cdot z_j^* \overset{C}{\geq} \left(1 - \frac{1}{e}\right) z_j^*$$

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

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

■

8.4 BEST-SAT

Algorithm 16 BEST-SAT

1: S pravděpodobností $\frac{1}{2}$ spusť `RAND-SAT`, s pravděpodobností $\frac{1}{2}$ spusť `LP-SAT`.

Věta 8.4. *Algoritmus je $\frac{3}{4}$-aproximační.*

Důkaz. Nechť W je součet všech splnitelných klauzulí v `BEST-SAT`, W_1 v `RAND-SAT` a W_2 v `LP-SAT`.

$$\mathbb{E}[W] = \mathbb{E}\left[\frac{1}{2}W_1 + \frac{1}{2}W_2\right] \geq \frac{1}{2}\sum_{j=1}^{m} w_j \left(1 - \frac{1}{2^{k_j}}\right) + \frac{1}{2}\sum_{j=1}^{m} w_j \left[1 - \left(1 - \frac{1}{k_j}\right)^{k_j}\right] z_j^* \geq \underbrace{\frac{3}{4}\sum_{j=1}^{m} w_j z_j^*}_{\texttt{LP-OPT}} \geq \frac{3}{4} OPT$$

k_j	Rand-SAT	LP-SAT	BEST-SAT
1	$\frac{1}{2}$	$1 \cdot z_j^*$	$\geq \frac{3}{4} z_j^*$
2	$\frac{3}{4} = k_1 + \frac{1}{4}$	$\frac{3}{4} = k_1 + (1 - \frac{1}{4}) z_j^*$	$\geq \frac{3}{4} z_j^*$
≥ 3	$\geq \frac{7}{8}$	$(1 - \frac{1}{e}) z_j^*$	$> \frac{3}{4} z_j^*$

■

9 Pokrývací problémy

9.1 Vrcholové pokrytí

- *Vstup:* Graf $G = (V, E)$ a ceny vrcholů $c : V \to \mathbb{R}^+$
- *Výstup:* $W \subseteq V$, že $\forall e \in E : e \cap W \neq 0$.
- *Cíl:* $\min \sum_{w\in W} c(w)$, minimalizujeme ceny, označme $C(W)$

Pozorování 9.1. *W je vrcholové pokrytí $\iff$ $V \setminus W$ je nezávislá množina.*

Pozorování 9.2. *Vrcholové pokrytí je speciálním případem množinového pokrytí, kde $f = 2$ a $g \leq n$ je maximální stupeň grafu. (f, g si uvedeme později).*

Algorithm 17 VERTEX-COVER

1: Vytvoř celočíselný lineární program *(viz. LP níže).*
2: Zrelaxuj program a vyřeš ho: $I = \{i \mid x_i^* \geq \frac{1}{2}\}$, tedy $x_v = 1$, když $x_v \geq \frac{1}{2}$.

Lineární program:

- *Proměnné:* $x_v \in \{0, 1\} \quad (\forall v \in V)$
- *Účelová funkce:* $\min \sum_{v \in V} c(v)\, x_v$
- *Podmínky:* $x_u + x_v \geq 1 \quad (\forall uv \in E)$

Věta 9.1. *Algoritmus je 2-aproximační.*

Důkaz. Proměnné jsme během relaxace zaokrouhlili z $x_v^* \geq \frac{1}{2}$ na 1. Tím jsme řešení max zdvojnásobili.

$$ALG = \sum_{v \in V} c(v) \leq 2 \sum_{v \in V} c(v) x_v^* \leq 2OPT.$$

■

9.2 Množinové pokrytí

- *Vstup:* Systém množin $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$, každý má cenu $c_1, \ldots, c_m \geq 0$, kde $c : V \to \mathbb{R}^+$
- *Výstup:* Podsystém $I \subseteq \{1, \ldots, m\}$, že $\bigcup_{j \in I} S_j = \{1, \ldots, n\}$.
- *Cíl:* $\min \sum_{j \in I} c(j)$, tedy minimalizovat cenu I.

Parametry: Můžeme si to představit jako bipartitní graf, kde levá partita obsahuje $\{1, \ldots, n\}$ a pravá $\{S_1, \ldots, S_m\}$. Potom máme parametry:

- $f = \max_{e \in [n]} |\{j \mid e \in S_j\}|$, tedy maximální stupeň na levé partitě,
- $g = \max_{j \in [m]} |S_j| \leq n$, tedy maximální stupeň na pravé partitě.

Algorithm 18 LP-SET COVER

1: Vytvoř celočíselný lineární program *(viz. LP níže).*
2: Zrelaxuj program a vyřeš ho: $I = \{i \mid x_i^* \geq \frac{1}{f}\}$, tedy zvol v, když $x_v \geq \frac{1}{f}$.

Lineární program

- *Účelová funkce:* $\min \sum_{i=1}^{m} c(i)\, x_i$
- *Proměnné:* $x_1, \ldots, x_m \geq 0$
- *Podmínky:* $\sum_{j : e \in S_j} x_j \geq 1$, pro $e = 1, \ldots, m$

Duální program

- *Účelová funkce:* $\max \sum_{e=1}^{m} y_e$
- *Proměnné:* $y_1, \ldots, y_n \geq 0$
- *Podmínky:* $\sum_{e \in S_j} y_e \leq c(j)$, pro $j = 1, \ldots, m$

Věta 9.2. *Algoritmus je f-aproximační.*

Důkaz. Proměnné jsme během relaxace zaokrouhlili z $x_v^* \geq \frac{1}{f}$ na 1. Řešení jsme násobili max f-krát:

$$ALG = \sum_{i \in I} c(i) \leq f \sum_{i \in I} c(i) x_i^* \leq f \sum_{i=1}^{m} c(i) x_i^* = f \cdot OPT.$$

■

Podmínky komplementarity Pokud jsou x^* a y^* optima, pak $\begin{cases} \forall j : x_j^* = 0 \vee \sum y_e = c(j), \text{ a} \\ \forall e : y_e^* = 0 \vee \sum_{j:e \in S_j} x_j = 1 \end{cases}$

Algorithm 19 Primárně-Duální (PDA)

$I \leftarrow \emptyset, \quad E \leftarrow \emptyset, \quad y_1, \ldots, y_n = 0$
while $\exists e \notin E$ **do**
  $\delta = \min_{j:e \in S_j} \left(c(j) - \sum_{e \in S_j} y_e \right)$ ▷ *říká o kolik můžeme zvýšit*
  $y_e \leftarrow y_e + \delta$ ▷ *zvýšíme y_e "co nejvíc"*
  for all $j : e \in S_j$ a $\sum_{e \in S_j} y_e = c(j)$ **do** ▷ *přidáme množiny splňující podm. kompl.*
    $I \leftarrow I \cup \{j\}, \quad E \leftarrow E \cup S_j$
return I

Věta 9.3. *Algoritmus je f-aproximační.*

Důkaz.

$$\text{ALG} \overset{\text{def.}}{=} \sum_{j \in I} c(j) \overset{\text{PDA 5.}}{=} \sum_{j \in I} \sum_{e \in S_j} y_e \overset{\text{def } f}{\leq} \sum_{e=1}^{n} f \cdot y_e \overset{DP}{\leq} f \cdot \text{OPT}$$

■

Vybereme na začátku množinu pokrývající nejvíc prvků. A pak vybíráme takovou, která pokrývá nejvíce nových prvků. Tedy abychom si za stejnou cenu koupili co nejvíc.

Algorithm 20 Hladový Set-Cover

$I \leftarrow \emptyset, \quad E \leftarrow \emptyset, \quad q_e = 0$
while $E \neq \{1, \ldots, n\}$ **do**
  Pro $j \in \{1, \ldots, m\}$ & $S_j \not\subseteq E$ polož $p_j := \frac{c(j)}{|S_j \setminus E|}$. ▷ *kolik zaplatíme za pokrytí nového prvku*
  Buď j_0, že p_{j_0} je minimální.
  Buď $q_e = p_{j_0} \quad (\forall e \in S_{j_0} \setminus E)$. ▷ *uložíme cenu nově pokrytých prvků*
  $I \leftarrow I \cup \{j_0\}, \quad E \leftarrow E \cup S_{j_0}$.
return I

Věta 9.4. *Algoritmus je H_g aproximační.*

Důkaz. Máme, že $\text{ALG} = \sum_{e=1}^{m} q_e$.
Označme $\bar{q} = \frac{1}{H_g} q$ jako přípustné řešení duálního LP. Chceme $\sum_{e \in S_j} q_e \leq c(j)$.
Nechť $S_j = \{e_1, \ldots, e_i, \ldots, e_k\}$ a očíslujeme prvky tak, že e_k je první a e_1 je poslední pokrytý prvek. Uvědomme si, že $q_{e_i} \leq \frac{c(j)}{i}$, protože pro e_i máme prvních i prvků ještě nepokrytých. Z definice vybíráme vždy nejlevnější možnou množinu. Dostaneme tak:

$$\sum_{e \in S_j} q_e = \sum_{i=1}^{k} q_{e_i} \leq \frac{c(j)}{1} + \frac{c(j)}{2} + \ldots + \frac{c(j)}{k} = H_k \cdot c(j)$$

$$\sum_{e \in S_j} \overline{q}_e = \frac{1}{H_g} \sum_{e \in S_j} q_e \leq \frac{1}{H_g} \cdot H_k \cdot c(j) \overset{k \leq g}{\leq} c(j)$$

A tedy $\bar{q}$ je přípustné řešení. ■

Pozorování 9.3. *Jelikož platí, že $H_g \approx \ln g \leq \ln n$, tak je algoritmus $\ln(n)$-aproximační.*

10 Paralelní maximální nezávislá množina - PARA-MIS

- *Vstup:* Graf $G = (V, E)$
- *Výstup:* $I \subseteq V$ maximální nezávislá množina vzhledem k inkluzi

Vybereme velkou nezávislou množinu S. Z grafu odebereme $S \cup N(S)$ spolu se všemi incidentními hranami.[2]

Algorithm 21 PARA-MIS

1: $I \leftarrow \emptyset$
2: **while** $V \neq 0$ **do**
3: **for all** $v \in V$ **do** pokud je $d_v = 0$ pak $I := I \cup \{v\}$, $V := V \setminus \{v\}$. ▷ $d_v := \deg(v)$
4: **for all** $v \in V$ **do** označ v s pravděpodobností $\frac{1}{2d_v}$ (nezávisle).
5: **for all** $uv \in E$ **do** pokud u i v jsou označeny, smaž označení nižšího stupně.
6: $S \leftarrow \{v \in V \mid v \text{ označený}\}$
7: $I \leftarrow I \cup S$, $V \leftarrow V \setminus (S \cup N(S))$, $E \leftarrow$ odebereme hrany incidentní s vrcholem v $S \cup N(S)$.
return I, pokud $V \neq 0$.

Definice 10.1. Vrchol $v \in V$ je **dobrý**, pokud má alespoň $\frac{d_v}{3}$ sousedů stupně $\leq d_v$. Jinak **špatný**.

Definice 10.2. Hrana $uv \in E$ je **špatná**, pokud jsou oba její vrcholy špatné.

Lemma 10.1. *Existuje konstanta $\alpha > 0$, že $\forall v$ dobrý platí v jedné iteraci* $\Pr[v \in S \cup N(S)] \geq \alpha$. [3]

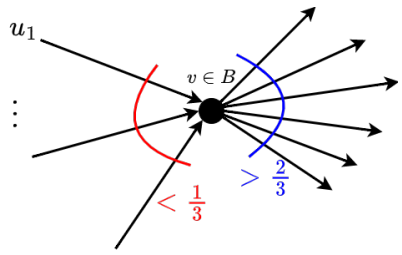
Důkaz. Nechť v je dobrý vrchol. Platí:

$$\Pr[v \text{ má souseda označeného v (4)}] \geq 1 - \overbrace{\prod_{w \in N(v)} \left(1 - \frac{1}{2d_w}\right)}^{\text{neoznačíme žádného souseda}} \geq 1 - \left(1 - \frac{1}{2d_v}\right)^{\frac{d_v}{3}} = \text{konst.} > 0$$

Jestliže je libovolný vrchol w označen, ukážeme, že s pravděpodobností alespoň $\frac{1}{2}$ zůstane označen i po kroku (5). Označení můžeme odebrat jedině když má označeného souseda stejného nebo vyššího stupně. Ten je však označen s pravděpodobností nejvýše $\frac{1}{2dw}$.
Vzhledem k tomu, že w má nejvýše dw takových sousedů, celková pravděpodobnost odebrání označení je nejvýše $d_w \cdot \frac{1}{2dw} = \frac{1}{2}$. ■

Lemma 10.2. *Špatných hran je nejvýše $\frac{|E|}{2}$. (Respektive, alespoň polovina hran je dobrá.)*

Důkaz. Nechť jsou B špatné vrcholy, E_B špatné hrany a d^{in}, d^{out} je počet (vy)stupujících hran. Hrany zorientujeme tak, že $u \to v$, pokud $d_u < d_v$, tedy od menšího k většímu stupni.



[2] $N(S) = \{u \mid \exists v \in S : uv \in E\}$ je množina sousedů
[3] Pravděpodobnost, že daný vrchol odstraníme je $> \alpha$

Podíváme se na špatný vrchol $v \in B$, z definice máme $d_v^{\text{in}} \leq \frac{d_v}{3}$. Proto pro vystupující $d_v^{\text{out}} > \frac{2}{3}d_v$ a tedy dostaneme $d_v^{\text{in}} < \frac{1}{2}d_v^{\text{out}}$. To platí i celkově při sečtení:

$$|E_B| \leq \sum_{v \in B} d_v^{\text{in}} \leq \frac{1}{2} \sum_{v \in B} d_v^{\text{out}} \leq \frac{1}{2}|E|.$$

■

Věta 10.1. *Průměrný počet fází algoritmu je* $\leq \mathcal{O}(\log n)$.

Důkaz. Nechť M_i je počet hran E po i-fázích.
Na počátku máme $M_0 = m = |E|$ počet hran na vstupu. Tvrdíme, že platí:

$$\mathbb{E}[M_{i+1}] \leq \left(1 - \frac{\alpha}{2}\right) \mathbb{E}[M_i].$$

Pokud máme v nějaké fázi M_i hran, které vstupují do i-té fáze, tak podle lemmatu 10.2. je alespoň $\frac{M_i}{2}$ dobrých a dobrá hrana je odebrána s pravděpodobností α.
"Polovina hran je dobrých a dobrou hranu odstraníme s pravděpodobností α*, proto* $1 - \frac{\alpha}{2}$*".*
Po t fázích, kde $t = c \log m$, dostaneme, že algoritmus skončí s pravděpodobností alespoň $\frac{1}{2}$:

$$\mathbb{E}[M_t] \leq \left(1 - \frac{\alpha}{2}\right)^t m \overset{c \log m}{\leq} \frac{1}{2}.$$

■

11 Hashovací funkce

Definice 11.1. Nechť $M, |M| = m, N, |N| = n, H \subseteq \{f \mid f : M \mapsto N\}$. Systém H je

- 2-**univerzální**, jestliže:

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

- **silně** 2-**univerzální**, jestliže:

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

Pozorování 11.1. *Silná* 2*-univerzalita* $\implies$ *slabá* 2*-univerzalita.*

Pozorování 11.2. *Silná* 2*-univerzalita* $\implies \{h(x) \mid x \in N\}$ *po* 2 *nezávislé náhodné proměnné.*

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

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

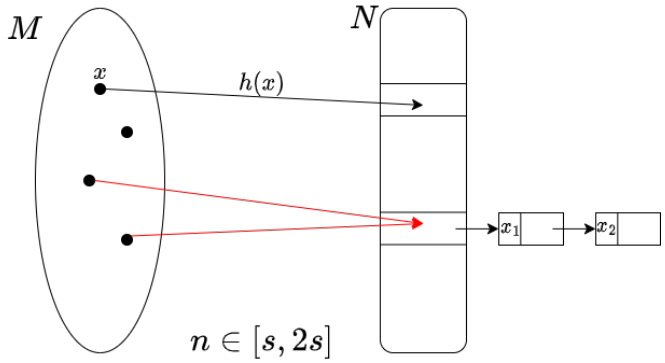
11.1 Dynamický slovník

Máme univerzum M velikosti $|M| = m = 2^d$, slovník $S \subseteq M$ velikosti $|S| = s$ a cílem je reprezentovat S tabulkou N velikosti $n = |N| = \mathcal{O}(s)$. H je silně 2-univerzální, $h \in H$ volíme uniformě náhodně.

Kolize Řešíme za pomoci spojového seznamu. Řetězíme za sebe.

Operace: (trvá průměrně $\mathcal{O}(1)$):

- vložení do S
- vyhledávání x v S
- vymazání x z S



Lemma 11.1. *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 $\forall x, y, x \neq y$ jsou $h(x), h(y)$ nezávislé, tak $\mathbb{E}[X_y] = \frac{1}{n}$:

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

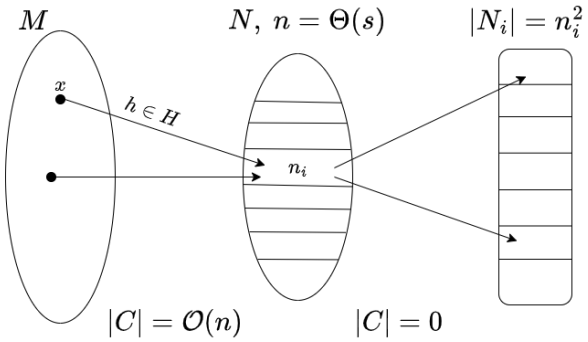
■

Lemma 11.2. $\mathbb{E}[|C|] = \binom{s}{2}\frac{1}{n}$, *pokud je H silně 2-univerzální, kde C je množina kolizí pro h, S.*

11.2 Statický slovník

S je dáno předem. Vytvoříme datastrukturu v polynomiálním čase. Chceme, aby prostor byl velikosti $\mathcal{O}(|S|)$ a aby operace vyhledání běžela nejhůř v čase $\mathcal{O}(1)$.
(to jsme předtím neměli – seznam mohl být dlouhý a maximální počet operací velký)

1. Najdeme $h \in H : U \to T$ tak, že $|C| \leq n$. [4]
2. Vytvoříme dvě tabulky a hashujeme dvakrát – jednou pro index do první tabulky ($|C| \leq n$) a ta určí funkci pro druhé hashování ($|C| = 0$).



Lemma 11.3. *Existuje $h \in H$ s $|C| \leq n$.*

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

Lemma 11.4. *Existuje $h \in H$ s $|C| = 0$.*

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

Lemma 11.5. $\sum n_i^2 \in \mathcal{O}(|C| + s)$

Důkaz. $|C| = \sum_{i=1}^{n} \binom{n_i}{2} = \sum \left(\frac{n_i^2}{2} - \frac{n_i}{2}\right) = \sum \frac{n_i^2}{2} - \frac{n}{2}$ ■

[4] $C = \{\{x, y\} \mid x, y \in M, x \neq y, h(x) = h(y)\}$, neboli kolize

12 Testování

12.1 Násobení matic

- *Vstup:* Matice $A, B, C \subseteq K^{n \times n}$
- *Výstup:* ANO, pokud $A \cdot B = C$, jinak NE.

Algorithm 22 MATRIX, $\mathcal{O}(n^2)$

1: Vezmi náhodný $\vec{x} \in \{0,1\}^n$
2: **if** $A \cdot B \cdot \vec{x} = C \cdot \vec{x}$ **then return** ANO
3: **else return** NE

Věta 12.1. *Pokud $A \cdot B \neq C$, pak* $\Pr[\text{'NE'}] \geq \frac{1}{2}$.

Důkaz. Nechť $D = AB - C$ je nenulová matice. Pokud $D \neq 0$, pak $\Pr[Dx \neq 0] \geq \frac{1}{2}$. ■

12.2 Nulovost polynomů (PIT)

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

Nezajímá nás, jestli je identicky nulový, ale zda je nulový *v tělese*, ve kterém pracujeme. Budeme pracovat s polynomy více proměnných a d bude označovat celkový stupeň (součet stupňů v nějakém nenulovém monomu).
Převeditelné na to, zda je výrok tautologie (jdou na sebe převést) $\implies$ NP těžké.

Lemma 12.1. *Nechť $P(x_1, \ldots, x_n)$ je nenulový polynom nad K stupně $\leq d_i$ a $S \subseteq K$ konečná množina. Nechť $x_1, \ldots, x_n \in S$ uniformě náhodně. Pak*

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

Pro $n = 1$ má polynom 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 $\leq \frac{d-k}{|S|}$.
Chceme 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 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

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

■

13 Perfektní párování

Definice 13.1. (Edmondsova matice). Nechť (U, V, E) je bipartitní graf, $n = |U| = |V|$. Pak *Edmondsova matice* grafu je $n \times n$ matice B definována předpisem $B_{u,v} = \begin{cases} x_{u,v} & uv \in E, \\ 0 & \text{jinak.} \end{cases}$

Za každou hranu bude v matici jedna proměnná.

Pozorování 13.1. $\det(B)$ *je polynom, jehož monomy bijektivně odpovídají perfektním párováním. (sčítáme součin permutace matice a když se zrovna trefíme do párovaní, tak máme monom)*

Algorithm 23 Test existence perfektního párování

1: Zvol uniformně náhodně nezávisle $x_{u,v} \in \{1, \ldots, 2n\}$, ▷ $2n$, aby nám vyšlo 'NE' s $p \geq \frac{1}{2}$
2: Spočítej determinant.
3: **if** determinant $\neq 0$ **then** párování určitě existuje.
4: **else** párování neexistuje s pravděpodobností $\geq \frac{1}{2}$

13.1 Izolující lemma

Věta 13.1. *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*[5]

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

Pro naše použití budeme chtít $r = 2m$

Důkaz. Nechť A_i je jev, že existují S_k, S_l tak, že $w(S_k) = w(S_l) = \min_j w(S_j)$ 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 z jevů A_i, pak máme vyhráno, jelikož dvě minimální neexistují
Ukážeme, že $\Pr[A_i] \leq \frac{1}{r}$. Systémy $S_1, \ldots, S_n$ rozdělíme na dvě množiny podle i:

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

Pokud A_i nastane, pak platí: $\begin{cases} \text{pro } S_k & k \in \mathcal{S}_0, w(S_k) = \min\limits_{j \in \mathcal{S}_0} w(S_j), \\ \text{pro } S_l & l \in \mathcal{S}_1, w(S_l) = \min\limits_{j \in \mathcal{S}_1} w(S_j). \end{cases}$
Pak, když zafixujeme všechny váhy a vybíráme váhu a_i, platí:

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

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

Algorithm 24 Rychlý paralelní algoritmus pro perfektní párování

1: Zvol rovnoměrně náhodně váhy $w(uv) \in \{1, \ldots, 2m\}$ pro každou hranu.
2: Zasubstituuj do Edmondsovy matice $x_{uv} = 2^{w(uv)}$. ▷ (*)
3: Najdi W tak, že 2^W je maximální číslo tvaru 2^α, že $2^\alpha \mid \det(C)$. ▷ (**)
4: **for** $uv \in E$ **do**
5: Spočítej $d = \det(C^{uv})$.
6: **if** $2^{W - w(uv)}$ je max. číslo tvaru $2^\alpha \mid d$ **then** přidej uv do M ▷ *přežilo párov. smazání uv?*
7: Zkontroluj, že M je PP ▷ *mohli jsme vygenerovat nesmysl*

(*) V algoritmu příspěvek PP je $\pm 2^{w(M)} = \pm \prod_{uv \in M} 2^{w(uv)}$
(**) Zajímá nás *poslední index, kde* $\det = 1$, jelikož to odpovídá unikátnímu perfektnímu párování. (všechny PP jsou ve tvaru $0b10000$, kde $w(uv)$ jsou nuly za 1)

[5] 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).

Zdroje

Čerpal jsem z vlastních poznámek z hodin plus:

- skripta prof. Jiřího Sgalla
- poznámky z hodin – Sláma