12. Aproximačné algoritmy *(F. Haško, J. Menda, M. Mareš)*

Na minulých prednáškach sme sa zaoberali rôzne ťažkými rozhodovacími problémami. Táto sa zaoberá postupmi ako sa v praxi vysporiadať s riešením týchto problémov.

Prvý spôsob: Špeciálny prípad

Často si vystačíme s vyriešením špeciálneho prípadu NP-úplného problému, ktorý leží v P. Napríklad, ak riešime grafovú úlohu, tak nám môže stačiť riešenie pre špeciálny druh grafov (strom, bipartitný graf, ...). Farbenie grafu je ľahké pre nejaký malý počet farieb. 2SAT, ako špeciálny prípad SAT-u, sa dá riešiť v lineárnom čase.

Problém: Maximálna nezávislá množina v strome (nie rozhodovacia)

Vstup: zakorenený strom T

Výstup: Maximálna (čo do počtu vrcholov) nezávislá množina vrcholov M

BUNV môžeme predpokladať, že v M sú všetky listy T. Ak by nejaký list l nebol v M, tak sa pozrieme na jeho otca:

- Ak otec nie je v M, tak vytvoríme novú nezávislú množinu M' obsahujúcu aj l (veľkosť nezávislej množiny stúpla o 1).
- Ak tam otec je, tak ho z M vyjmeme a namiesto neho vložíme l (veľkosť nezávislej množiny sa nezmenšila).

Tieto listy aj ich otcov z T odstránime a postup opakujeme. T sa môže rozpadnúť na les; potom tento postup aplikujeme na všetky stromy v lese.

Algoritmus:

1. Položíme M_1:=$\{$listy stromu $T\}$.
2. Položíme M_2:=$\{$otcovia vrcholov z $M_1\}$.
3. Vrátime $M_1 \cup$ MaxNz($T \setminus (M_1 \cup M_2)$.

Poznámka: Toto dokážeme naprogramovať v $\mathcal{O}(n)$ (vrcholy máme vo fronte a prechádzame).

Problém: Batoh

Je daná množina n predmetov s hmotnosťami $h_1, \ldots, h_n$ a cenami $c_1, \ldots, c_n$ a batoh, ktorý unesie hmotnosť H. Nájdite takú podmnožinu predmetov, ktorých celková hmotnosť je najviac H a celková cena je maximálna možná.

Tento problém je zobecněním problému batohu z minulé přednášky dvěma směry: Jednak místo rozhodovacího problému řešíme optimalizační, jednak předměty mají ceny (předchozí verze odpovídala tomu, že ceny jsou rovny hmotnostem). Ukážeme si algoritmus pro řešení tohoto obecného problému, jehož časová složitost bude polynomiální v počtu předmětů n a součtu všech cen $C = \sum_i c_i$.

Použijeme dynamické programování. Představme si problém omezený na prvních k předmětů. Označme si $A_k(c)$ (kde $0 \le c \le C$) minimální hmotnost podmnožiny, jejíž cena je právě c. Tato A_k spočteme indukcí podle k: Pro $k = 0$ je určitě $A_0(0) = 0$,

$A_0(c) = \infty$ pro $c > 0$. Pokud již známe A_{k-1}, spočítáme A_k následovně: $A_k(c)$ odpovídá nějaké podmnožině předmětů z $1, \ldots, k$. V této podmnožině jsme buďto k-tý předmět nepoužili (a pak je $A_k(c) = A_{k-1}(c)$), nebo použili a tehdy bude $A_k(c) = A_{k-1}(c-c_k)+h_k$ (to samozřejmě jen pokud $c \geq c_k$). Z těchto dvou možností si vybereme tu, která dává množinu s menší hmotností. Tedy:

$$A_k(c) = \min(A_{k-1}(c), A_{k-1}(c - c_k) + h_k).$$

Tímto způsobem v čase $\mathcal{O}(C)$ spočteme jednu množinu, v čase $\mathcal{O}(nC)$ pak všechny.

Podle A_n snadno nalezneme maximální cenu množiny, která se vejde do batohu. To bude největší c^*, pro něž je $A_n(c^*) < \infty$. Jeho nalezení nás stojí čas $\mathcal{O}(C)$.

A jak zjistit, které předměty do nalezené množiny patří? Upravíme algoritmus, aby si pro každé $A_k(c)$ pamatoval $B_k(c)$, což bude index posledního předmětu, který jsme do příslušné množiny přidali. Pro nalezené c^* tedy bude $i = B_n(c^*)$ poslední předmět v nalezené množině, $i' = B_{i-1}(c^* - c_i)$ ten předposlední a tak dále. Takto v čase $\mathcal{O}(n)$ rekonstruujeme celou množinu od posledního prvku k prvnímu.

Ukázali jsme tedy algoritmus s časovou složitostí $\mathcal{O}(nC)$, který vyřeší problém batohu. Jeho složitost není polynomem ve velikosti vstupu (C muže být až exponenciálně velké vzhledem k velikosti vstupu), ale pouze ve velikosti čísel na vstupu. Takovým algoritmům se říká *pseudopolynomiální.*

Verze bez cen: Na verzi s cenami rovnými hmotnostem se dá použít i jiný algoritmus založený na dynamickém programování: počítáme množiny Z_k obsahující všechny hmotnosti menší než H, kterých nabývá nějaká podmnožina prvních k prvků. Přitom $Z_0 = \{0\}$, Z_k spočteme ze Z_{k-1} a ze Z_n vyčteme výsledek. Všechny tyto množiny mají nejvýše H prvků, takže celková časová složitost algoritmu je $\mathcal{O}(nH)$.

Druhý spôsob: Aproximácia

V predchádzajúcich problémoch sme sa zamerali na špeciálne prípady. Občas však také štastie nemáme a musíme vyriešiť celý NP-úplný problém. Možeme si však pomôcť tým, že sa nebudeme snažiť vyriešiť ho optimálne – iba v nejakom pomere k optimálnosti (*aproximácia*), t.j. budeme vedieť, o koľko maximálne je naše riešenie horšie ako optimálne.

Problém: Obchodný cestujúci

Vstup: neorientovaný graf G, popisujúci nejaku krajinu a každá hrana je ohodnotená funkciou $w : E(G) \to \mathbb{R}_0^+$

Vystup: Hamiltonovská kružnica (všetky vrcholy grafu), a to tá najkratšia (podľa ohodnotenia).

Tento problém je hneď na prvý pohľad náročný – už problém, či existuje Hamiltonovská kružnica, je NP-úplný. BUNV nech graf G je úplný (doplnime zvyšné hrany ohodnotené $max(w)+1$ alebo viac, nie však nekonečnom, lebo by neplatila trojuholníková nerovnosť, ktorú neskôr budeme potrebovať). Vyriešme tento problém najprv za predpokladu, že vrcholy grafu spĺňajú trojuholníkovú nerovnosť, potom bez nej.

a) trojuholníková nerovnosť: $\forall x, y, z \in V : w(xz) \leq w(xy) + w(yz)$

Existuje pekný algoritmus, ktory nájde Hamiltonovsku kružnicu, čo je maximálne dvakrát tak veľká ako optimálna.

Nájdeme najmenšiu kostru a obchodnému cestujúcemu poradíme, nech ide po nej (stačí zakoreniť a prejsť do hĺbky). Problémom však je, že daný sled obsahuje každý vrchol viackrát a preto musíme nahradiť nepovolené vracania sa, t.j. pre každý vrchol nájsť ešte nenavštívený vrchol v našom slede a ísť priamo naň. Keďže platí trojuholníková nerovnosť, tak si týmito skratkami neuškodíme. Nech minimálna kostra má váhu T. Váha obídeného sledu tak bude $2T$. Skrátenia určite nezväčšujú, takže váha nájdene Hamiltonovskej kružnice bude nanajvýš $2T$.

Ak máme Hamiltonovskú kružnicu C a z nej vyškrtneme hranu, tak máme kostru grafu G s váhou najviac $w(C)$, teda to aspoň takú, aká je váha minimálnej kostry – T. To je optimálny prípad Hamiltonovskej kružnice. Ak to teda zložíme dohromady, algoritmus nám vráti Hamiltonovskú kružnicu s váhou najviac dvojnásobnou od optimálnej Hamiltonovskej kružnice. Takéto algoritmy sa nazývajú *2-aproximačné*, keďže riešenie je maximálne dvojnásobné od optimálneho.

b) bez trojuholníkovej nerovnosti:

Tu sa budeme naopak snažiť ukázať, že žiaden polynomiálny aproximačný algoritmus neexistuje.

Veta: Ak existuje polynomiálny $(1 + \varepsilon)$-aproximačný algoritmus pre algoritmus obchodného cestujúceho bez trojuholníkovej nerovnosti pre ľubovoľné $\varepsilon > 0$, tak potom $P = NP$.

Důkaz: Ukážeme, že v tom prípade dokážeme v polynomiálnom čase nájsť Hamiltonovskú kružnicu.

Dostali sme graf G, v ktorom hľadáme Hamiltonovskú kružnicu. Doplníme G na uplný graf G' a váhy hrán G' nastavíme takto:

- $w(e) = 1$, ak $e \in E(G)$
- $w(e) = c \ll 1$, ak $e \notin E(G)$

Ak existuje Hamiltonovská kružnica v G' zložená iba z hrán, ktoré boli pôvodne v G, tak optimálné riešenie bude mať váhu n, inak bude určite minimálne $n - 1 + c$. Ak máme aproximačný algoritmus s pomerom $1 + \varepsilon$, musí byť

$$\begin{aligned}(1 + \varepsilon).n &< n - 1 + c \\ \varepsilon n + 1 &< c\end{aligned}$$

Ak by taký algoritmus existoval, tak na neho máme polynomiálny algoritmus na Hamiltonovsku kružnicu. Inak neexistuje ani pseudo-polynomialny algoritmus. ♡

Aproximační schéma pro problém batohu

POZOR: Verze algoritmu, kterou jsem říkal na přednášce, obsahovala jednu poměrně zásadní chybu, které jsem si nevšiml: Verze se zaokrouhlováním dolů mohla produkovat nepřípustná (příliš těžká) řešení, verze se zaokrouhlováním nahoru pro změnu někdy spočítala řešení příliš daleká od optima. Algoritmus lze opravit (budeme-li zvlášť zpracovávat lehké a těžké předměty), ale raději budeme místo hmotností kvantovat ceny. Tak dojdeme k následujícímu aproximačnímu algoritmu. –M.M.

Již víme, jak optimalizační verzi problému batohu vyřešit v čase $\mathcal{O}(nC)$, pokud jsou hmotnosti i ceny na vstupu přirozená čísla a C je součet všech cen. Jak si poradit, pokud je C obrovské? Kdybychom měli štěstí a všechny ceny byly dělitelné nějakým číslem p, mohli bychom je tímto číslem vydělit. Tím bychom dostali zadání s menšími čísly, jehož řešením by byla stejná množina předmětů jako u zadání původního.

Když nám štěstí přát nebude, můžeme přesto zkusit ceny vydělit a výsledky nějak zaokrouhlit. Řešení nové úlohy pak sice nebude přesně odpovídat optimálnímu řešení té původní, ale když nastavíme parametry správně, bude alespoň jeho dobrou aproximací.

Základní myšlenka:

Označíme si c_{max} maximum z cen c_i. Zvolíme si nějaké přirozené číslo M a zobrazíme interval cen $[0, c_{max}]$ na $[0, M]$. Jak jsme tím zkreslili výsledek? Všimněme si, že efekt je stejný, jako kdybychom jednotlivé ceny zaokrouhlili na násobky čísla c_{max}/M. Každé c_i jsme tím změnili o nejvýše c_{max}/M, celkovou cenu libovolné podmnožiny předmětů tedy nejvýše o $n \cdot c_{max}/M$. Teď si ještě všimněme, že pokud ze zadání odstraníme předměty, které se samy nevejdou do batohu, má optimální řešení původní úlohy cenu $OPT \geq c_{max}$, takže chyba v součtu je nejvýše $n \cdot OPT/M$. Má-li tato chyba být shora omezena $\varepsilon \cdot OPT$, musíme zvolit $M \geq n/\varepsilon$.

Algoritmus:

1. Odstraníme ze vstupu všechny předměty těžší než H.
2. Spočítáme $c_{max} = \max_i c_i$ a zvolíme $M = \lceil n/\varepsilon \rceil$.
3. Kvantujeme ceny: $\hat{c}_i = \lfloor c_i \cdot M/c_{max} \rfloor$.
4. Vyřešíme dynamickým programováním problém batohu pro upravené ceny $\hat{c}_1, \ldots, \hat{c}_n$ a původní hmotnosti i kapacitu batohu.
5. Vybereme stejné předměty, jaké použilo optimální řešení kvantovaného zadání.

Kroky 1–3 a 5 jistě zvládneme v čase $\mathcal{O}(n)$. Krok 4 řeší problém batohu se součtem cen $\hat{C} \leq nM \leq n^2/\varepsilon$, což stihne v čase $\mathcal{O}(n\hat{C}) = \mathcal{O}(n^3/\varepsilon)$. Zbývá dokázat, že výsledek našeho algoritmu má opravdu relativní chybu nejvýše ε.

Nejprve si rozmyslíme, jak dopadne optimální řešení OPT původního zadání, když ceny v něm použitých předmětů nakvantujeme (množinu indexů těchto předmětů si označíme Y):

$$\widehat{OPT} = \sum_{i \in Y} \hat{c}_i = \sum_i \left\lfloor c_i \cdot \frac{M}{c_{max}} \right\rfloor \geq \sum_i \left(c_i \cdot \frac{M}{c_{max}} - 1 \right) \geq$$
$$\geq \left(\sum_i c_i \cdot \frac{M}{c_{max}} \right) - n = OPT \cdot \frac{M}{c_{max}} - n.$$

Nyní naopak spočítejme, jak dopadne řešení Q nakvantovaného problému při přepočtu na původní ceny (to je výsledek našeho algoritmu):

$$ALG = \sum_{i \in Q} c_i \geq \sum_i \hat{c}_i \cdot \frac{c_{max}}{M} = \left(\sum_i \hat{c}_i \right) \cdot \frac{c_{max}}{M} \geq^* \widehat{OPT} \cdot \frac{c_{max}}{M}.$$

Nerovnost $\geq^*$ platí proto, že $\sum_{i \in Q} \hat{c}_i$ je optimální řešení kvantované úlohy, zatímco $\sum_{i \in Y} \hat{c}_i$ je nějaké další řešení téže úlohy, které nemůže být lepší. Teď už stačí složit obě nerovnosti a dosadit za M:

$$ALG \geq \left(\frac{OPT \cdot M}{c_{max}} - n \right) \cdot \frac{c_{max}}{M} \geq OPT - \frac{n \cdot c_{max}}{n/\varepsilon} \geq OPT - \varepsilon c_{max} \geq$$
$$\geq OPT - \varepsilon OPT = (1 - \varepsilon) \cdot OPT.$$

Algoritmus tedy vždy vydá řešení, které je nejvýše $(1 - \varepsilon)$-krát horší než optimum, a dokáže to pro libovolné ε v čase polynomiálním v n. Takovému algoritmu říkáme *polynomiální aproximační schéma* (jinak též PTAS⟨1⟩). V našem případě je dokonce složitost polynomiální i v závislosti na $1/\varepsilon$, takže schéma je *plně polynomiální* (řečené též FPTAS⟨2⟩).

⟨1⟩ Polynomial-Time Approximation Scheme
⟨2⟩ Fully Polynomial-Time Approximation Scheme