12. Aproximační algoritmy

(F. Haško, J. Menda, M. Mareš, Michal Kozák, Vojta Tůma)

Na minulých přednáškách jsme se zabývali různými těžkými rozhodovacími problémy. Tato se zabývá postupy, jak se v praxi vypořádat s řešením těchto problémů.

Co dělat, když potkáme NP-úplný problém

1. Nepanikařit.
2. Spokojit se s málem.
3. Rozmyslet, jestli opravdu potřebujeme obecný algoritmus. Mnohdy potřebujeme pouze speciálnější případy, které mohou být řešitelné v polynomiálním čase.
4. Spokojit se s přibližným řešením, (použít aproximační algoritmus).
5. Použít heuristiku – například genetické algoritmy nebo randomizované algoritmy. Velmi pomoci může i jen výhodnější pořadí při prohledávání či ořezávání některých napohled nesmyslných větví výpočtu.

První způsob: Speciální případ

Často si vystačíme s vyřešením speciálního případu NP-úplného problému, který leží v P. Například při řešení grafové úlohy nám může stačit řešení pro speciální druh grafů (stromy, bipartitní grafy, ...). Barvení grafu je lehké např. pro dvě barvy či pro intervalové grafy. 2-SAT, jako speciální případ SATu, se dá řešit v lineárním čase.

Ukážeme si dva takové případy (budeme řešení hledat, nejen rozhodovat, zda existuje)

Problém: Maximální nezávislá množina ve stromě (ne rozhodovací)

Vstup: Zakořeněný strom T.

Výstup: Maximální (co do počtu vrcholů) nezávislá množina vrcholů M v T.

BÚNO můžeme předpokládat, že v M jsou všechny listy T. Pokud by některý list l v M nebyl, tak se podíváme na jeho otce:

- Pokud otec není v M, tak list l přidáme do M, čímž se nezávislost množiny zachovala a velikost stoupla o 1.
- Pokud tam otec je, tak ho z M vyjmeme a na místo něho vložíme l. Nezávislost ani velikost M se nezměnily.

Nyní listy spolu s jejich otci z T odebereme a postup opakujeme. T se může rozpadnout na les, ale to nevadí → tentýž postup aplikujeme na všechny stromy v lese.

Algoritmus: MaxNz(T)

1. Položíme L:={listy stromu T}.
2. Položíme O:={otcové vrcholů z L}.
3. Vrátíme $L \cup$ MaxNz($T \setminus (O \cup L)$).

Poznámka: Toto dokážeme naprogramovat v $\mathcal{O}(n)$ (udržujeme si frontu listů).

Problém: Batoh

Je daná množina n předmětů s hmotnostmi $h_1, \ldots, h_n$ a cenami $c_1, \ldots, c_n$ a batoh, který unese hmostnost H. Najděte takovou podmnožinu předmětů, jejichž celková hmotnost je maximálně H a celková cena je maximální 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 $0 \leq c \leq C$) 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 $A_k(c)$ pro fixní k a všechna c, v čase $\mathcal{O}(nC)$ pak všechny $A_k(c)$.

Podle A_n snadno nalezneme maximální cenu množiny, která se vejde do batohu. To bude největší c^*, pro něž je $A_n(c^*) \leq H$. 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 může být až exponenciálně velké vzhledem k velikosti vstupu), ale pouze ve velikosti čísel na vstupu. Takovým algoritmům se říká *pseudopolynomiální.* Ani takové algoritmy ale nejsou k dispozici pro všechny problémy (např. u problému obchodního cestujícího nám vůbec nepomůže, že váhy hran budou malá čísla).

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} — udržujme si Z_{k-1} jako setříděný spojový seznam, výpočet dalšího seznamu uděláme slitím dvou seznamů Z_{k-1} a Z_{k-1} se všemi prvky zvýšenými o hmotnost k zahazujíce duplicitní a příliš velké hodnoty — a ze Z_n vyčteme výsledek. Všechny tyto množiny mají nejvýše H prvků, takže celková časová složitost algoritmu je $\mathcal{O}(nH)$.

Druhý způsob: Aproximace

V předcházejících problémech jsme se zaměřili na speciální případy. Občas však takové štěstí nemáme a musíme vyřešit celý NP-úplný problém. Můžeme si však pomoct tím, že se ho nebudeme snažit vyřešit optimálně – namísto optimálního řešení najdeme nějaké, které je nejvýše c-krát horší pro nějakou konstantu c.

Problém: Obchodní cestující

Vstup: Neorientovaný graf G, každá hrana je ohodnocená funkcí $w : E(G) \to \mathbb{R}_0^+$.

Výstup: Hamiltonovská kružnice (obsahující všechny vrcholy grafu), a to ta nejkratší (podle ohodnocení).

Tento problém je hned na první pohled náročný – už sama existence hamiltonovské kružnice je NP-úplná. Najdeme aproximační algoritmus nejprve za předpokladu, že vrcholy splňují trojúhelníkovou nerovnost (tj. $\forall x, y, z \in V : w(xz) \leq w(xy)+w(yz)$), potom ukážeme, že v úplně obecném případě by samotná existence aproximačního algoritmu implikovala P = NP.

a) trojúhelníková nerovnost:

Existuje pěkný algoritmus, který najde hamiltonovskou kružnici o délce $\leq 2 \cdot opt$, kde opt je délka nejkratší hamiltonovské kružnice. Vedle předpokladu trojúhelníkové nerovnosti budeme potřebovat, aby náš graf byl úplný. Souhrnně můžeme předpokládat, že úlohu řešíme v nějakém metrickém protoru, ve kterém jsou obě podmínky podle definice splněny.

Najdeme nejmenší kostru grafu a obchodnímu cestujícímu poradíme, ať jde po ní – kostru zakořeníme a projdeme jako strom do hloubky, přičemž se zastavíme až v kořeni po projití všech vrcholů. Problém však je, že průchod po kostře obsahuje některé vrcholy i hrany vícekrát, a proto musíme nahradit nepovolené vracení se. Máme-li na nějaký vrchol vstoupit podruhé, prostě ho ignorujeme a přesuneme se rovnou na další nenavštívený – dovolit si to můžeme, graf je úplný a obsahuje hrany mezi všemi dvojicemi vrcholů (jinak řečeno, pořadí vrcholů kružnice bude preorder výpis průchodem do hloubky). Pokud platí trojúhelníková nerovnost, tak si těmito zkratkami neuškodíme. Nechť minimální kostra má váhu T. Pokud bychom prošli celou kostru, bude mít sled váhu $2T$ (každou hranou kostry jsme šli tam a zpátky), a přeskakování vrcholů celkovou váhu nezvětšuje (při přeskoku nahradíme cestu xyz jedinou hranou xz, přičemž z trojúhelníkové nerovnosti máme $xz \leq xy + xz$), takže váha nalezené hamiltonovské kružnice bude také nanejvýš $2T$.

Když máme hamiltonovskou kružnici C a z ní vyškrtneme hranu, dostaneme kostru grafu G s váhou menší než C – ale každá kostra je alespoň tak těžká jako minimální kostra T. Tedy optimální hamiltonovská kružnice je určitě těžší než minimální kostra T. Když tyto dvě nerovnosti složíme dohromady, algoritmus nám vrátí hamiltonovskou kružnici T' s váhou nanejvýš dvojnásobnou vzhledem k optimální hamiltonovské kružnici ($T' \leq 2T < 2C$). Takovéto algoritmy se nazývají *2-aproximační*, když řešení je maximálně dvojnásobné od optimálního.⟨1⟩

⟨1⟩ Hezkým trikem se v obecných metrických prostorech umí 1,5-aproximace. Ve ně-

b) bez trojúhelníkové nerovnosti:

Zde se budeme naopak snažit ukázat, že žádný polynomiální aproximační algoritmus neexistuje.

Věta: Pokud pro libovolné $\varepsilon > 0$ existuje polynomiální $(1+\varepsilon)$-aproximační algoritmus pro problém obchodního cestujícího bez trojúhelníkové nerovnosti, tak P = NP.

Důkaz: Ukážeme, že v takovém případě dokážeme v polynomiálním čase zjistit, zda v grafu existuje hamiltonovská kružnice.

Dostali jsme graf G, ve kterém hledáme hamiltonovskou kružnici. Doplníme G na úplný graf G' a váhy hran G' nastavíme takto:

- $w(e) = 1$, když $e \in E(G)$
- $w(e) = c \gg 1$, když $e \notin E(G)$

Konstantu c potřebujeme zvolit tak velkou, abychom jasně poznali, jestli je každá hrana z nalezené hamiltonovské kružnice hranou grafu G (pokud by nebyla, bude kružnice obsahovat aspoň jednu hranu s váhou c, která vyžene součet poznatelně vysoko). Pokud existuje hamiltonovská kružnice v G' složená jen z hran, které byly původně v G, pak optimální řešení bude mít váhu n, jinak bude určitě minimálně $n - 1 + c$. Když máme aproximační algoritmus s poměrem $1 + \varepsilon$, musí tedy být

$$(1 + \varepsilon) \cdot n < n - 1 + c$$
$$\varepsilon n + 1 < c$$

Kdyby takový algoritmus existoval, máme polynomiální algoritmus na hamiltonovskou kružnici. ♡

Poznámka: O existenci pseudopolynomiálního algoritmu platí analogická věta, a dokáže se analogicky – existující hrany budou mít váhu 1, neexistující váhu 2.

Aproximační schéma pro problém batohu

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:

---

kterých metrických prostorech (třeba v euklidovské rovině) se aproximační poměr dá dokonce srazit na libovolně blízko k 1. Zaplatíme ale na čase – čím přesnější výsledek po algoritmu chceme, tím déle to bude trvat.

Označíme si c_{max} maximum z cen c_i. Zvolíme si nějaké přirozené číslo $M < c_{max}$ a zobrazíme interval cen $[0, c_{max}]$ na $[0, M]$ (tedy každou cenu znásobíme M/c_{max}). Jak jsme tím zkreslili výsledek? Všimněme si, že efekt je stejný, jako kdybychom jednotlivé ceny zaokrouhlili na násobky čísla c_{max}/M (prvky z intervalu $[i \cdot c_{max}/M, (i+1) \cdot c_{max}/M)$ se zobrazí na stejný prvek). Každé c_i jsme tím tedy změnili o nejvýše c_{max}/M, celkovou cenu libovolné podmnožiny předmětů pak 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 $M \geq n/\varepsilon$.⟨2⟩

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: $\forall i : \hat{c}_i \leftarrow \lfloor c_i \cdot M/c_{max} \rfloor$.
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 $\hat{C} \leq nM = \mathcal{O}(n^2/\varepsilon)$, 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, jakou cenu budou mít předměty které daly optimální řešení v původním zadání (tedy mají v původním zadání dohromady cenu OPT), když jejich ceny 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 spočítejme, jak dopadne optimální 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

⟨2⟩ Připoměňme, že toto ještě není důkaz, neboť velkoryse přehlížíme chyby dané zaokrouhlováním. Důkaz provedeme níže.

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⟨3⟩). V našem případě je dokonce složitost polynomiální i v závislosti na $1/\varepsilon$, takže schéma je *plně polynomiální* (řečené též FPTAS⟨4⟩). U některých problémů se stává, že aproximační schéma závisí na $1/\varepsilon$ exponenciálně, což tak příjemné není. Shrňme, co jsme zjistili, do následující věty:

Věta: Existuje algoritmus, který pro každé $\varepsilon > 0$ nalezne $(1-\varepsilon)$-*aproximaci* problému batohu s n předměty v čase $\mathcal{O}(n^3/\varepsilon)$.

⟨3⟩ Polynomial-Time Approximation Scheme

⟨4⟩ Fully Polynomial-Time Approximation Scheme