Lineární programování a aproximační algoritmy

Štěpán Šimsa

Abstrakt. Příspěvek se věnuje dvěma informatickým tématům, lineárnímu programování a aproximačním algoritmům, jejichž výsledky se často používají v praxi. Přestože je lineární programování důležité především pro informatiky, jedná se o matematický problém. Aproximační algoritmy jsou často založené právě na lineárním programování, a proto na sebe tato dvě témata navazují. Přestože aproximační algoritmy jsou tématem nesporně informatickým, často bývá obtížnější dokázat, že daný algoritmus je skutečně aproximační, než algoritmus vymyslet.

Lineární programování

Definice. *Úlohou lineárního programování* je maximalizovat jistou účelovou funkci na základě daných lineárních podmínek. Formálně mějme reálné proměnné $x_1, \ldots, x_n$ a konstanty $c_1, \ldots, c_n, a_{11}, \ldots, a_{mn}, b_1, \ldots, b_m$. Pak chceme najít maximum funkce $c_1x_1 + \cdots + c_nx_n$ přes všechny možné hodnoty $x_1, \ldots, x_n$ splňující podmínky $a_{i1}x_1 + \cdots + a_{in}x_n \leq b_i$ pro všechna $i \in \{1, \ldots, m\}$.

Cvičení. Ukaž, že se tato úloha také dá vyjádřit lineárním programem:

$$\begin{array}{ll} \text{minimalizovat} & x_1 - 3x_2 + \pi x_3 \\ \text{mezi všemi vektory } (x_1, x_2, x_3) \in \mathbb{R}^3 & \\ \text{za podmínek} & x_1 + 2x_2 = x_3 \\ & x_2 - x_3 \leq 0 \\ & x_3 \geq 1. \end{array}$$

Předchozí cvičení ukazuje, že si můžeme v lineárních podmínkách povolit i opačné nerovnosti, nebo dokonce i rovnosti, a podobně že můžeme účelovou funkci minimalizovat místo maximalizovat.

Příklad 1. Najdi optimální řešení následujícího lineárního programu:

$$\begin{array}{ll} \text{maximalizovat} & x_1 + x_2 \\ \text{mezi všemi vektory } (x_1, x_2) \in \mathbb{R}^2 & \\ \text{za podmínek} & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_2 - x_1 \leq 1 \\ & x_1 + 6x_2 \leq 15 \\ & 4x_1 - x_2 \leq 10. \end{array}$$

Definice. *Přípustné řešení* lineárního programu je libovolné řešení, které splňuje všechny zadané podmínky. *Optimální řešení* je takové přípustné řešení, jehož účelová funkce je největší, resp. nejmenší (pro maximalizační, resp. minimalizační problém).

Poznámka. Všimni si, že úloha nemusí mít žádné, jedno nebo i více přípustných řešení. To samé platí i pro optimální řešení úlohy.

Definice. Úloha je *nepřípustná*, pokud nemá žádné přípustné řešení, a *neomezená*, nabývá-li účelová funkce mezi přípustnými řešeními libovolně velkých (resp. malých) hodnot.

Na úlohu lineárního programování se také můžeme dívat geometricky. Máme-li n proměnných, tak nám každá podmínka určuje poloprostor v n-dimenzionálním prostoru. Přípustná řešení jsou pak ty body, které leží v průniku všech těchto poloprostorů, což je nějaký konvexní mnohostěn. Optimalizační funkce určuje směr a optimální řešení jsou ty body tohoto mnohostěnu, které jsou nejdále v tomto směru.

Cvičení. Rozmysli si, že má-li lineární program optimální řešení, pak je i nějaký z vrcholů příslušného mnohostěnu přípustných řešení optimálním řešením.

Předchozí cvičení naznačuje jeden možný způsob řešení lineárních programů. Nalezneme všechny vrcholy a vezmeme ten nejlepší.

Příklad 2. Najdi takový lineární program s n proměnnými a $2n$ podmínkami, aby konvexní mnohostěn představující přípustná řešení této úlohy měl 2^n vrcholů.

Příklad 3. Máme dán konvexní n-úhelník P v rovině a chceme najít největší kruh v něm obsažený.

Příklad 4. Máme daných několik modrých a červených bodů v rovině. Nalezni takovou přímku, že všechny modré, resp. červené body leží na jedné, resp. druhé straně přímky (žádný bod nesmí ležet na přímce samotné).

Poznámka. Úlohy lineárního programování se dají řešit v polynomiálním čase, například **elipsoidovou metodou**.

Duální program

Definice. *Duální program* k lineárnímu programu ve tvaru:

$$\begin{array}{lll} \text{maximalizovat} & \sum_{i=1}^{n} c_i x_i & \\ \text{za podmínek} & \sum_{i=1}^{n} a_{ji} x_i \leq b_j & \forall j \in \{1, 2, \ldots, m\} \\ & x_i \geq 0 & \forall i \in \{1, 2, \ldots, n\} \end{array} \qquad \text{(P)}$$

je lineární program:

$$\begin{array}{lll} \text{minimalizovat} & \sum_{i=1}^{m} c_i y_i & \\ \text{za podmínek} & \sum_{i=1}^{m} a_{ij} y_i \geq c_j & \forall j \in \{1, 2, \ldots, n\} \\ & y_i \geq 0 & \forall i \in \{1, 2, \ldots, m\} \end{array} \qquad \text{(D)}$$

Věta. (Slabá věta o dualitě) *Pro každé přípustné řešení x primární úlohy* (P) *a každé přípustné řešení y duální úlohy* (D) *platí*

$$\sum_{i=1}^{n} c_i x_i \leq \sum_{i=1}^{m} b_i y_i.$$

Speciálně, je-li (P) *neomezená, musí* (D) *být nepřípustná, a je-li* (D) *neomezená, pak je nepřípustná* (P).

Věta. (Silná věta o dualitě) *Pro úlohy* (P) *a* (D) *nastane právě jedna z následujících možností:*

(1) *Ani* (P), *ani* (D) *nemá přípustné řešení.*
(2) (P) *je neomezená a* (D) *nemá přípustné řešení.*
(3) (P) *nemá přípustné řešení a* (D) *je neomezená.*
(4) *Jak* (P), *tak* (D) *mají přípustné řešení. Pak existuje optimální řešení x^* úlohy* (P) *a optimální řešení y^* úlohy* (D) *a platí*

$$\sum_{i=1}^{n} c_i x_i^* = \sum_{i=1}^{m} b_i y^* k_i.$$

Těžké problémy

Zde následuje několik problémů, kterým říkejme *těžké*. Za předpokladu, že $P \neq NP$ (což se považuje za rozumný předpoklad) pro tyto algoritmy neexistuje žádný polynomiální algoritmus.

Problém. (Minimální vrcholové pokrytí) Úlohou je nalézt co nejmenší podmnožinu vrcholů takovou, že každá hrana má alespoň jeden ze svých krajních vrcholů v této množině.

Problém. (Set cover problem) Mějme množinu $U = \{1, 2, \ldots, n\}$ a její podmnožiny $S_1, \ldots, S_m$, jejichž sjednocením je U. Cílem je najít co nejmenší indexovou množinu $I \subset U$, že sjednocení přes příslušné podmnožiny $S_i, i \in I$, je celá množina U, tedy:

$$\bigcup_{i \in I} S_i = U.$$

Problém. (Největší nezávislá množina) Úlohou je najít největší *nezávislou množinu* v grafu, tedy podmnožinu vrcholů takovou, že mezi nimi nevede žádná hrana.

Problém. (Problém obchodního cestujícího) Máme daná města a vzdálenost mezi každými dvěma těmito městy. Cílem je najít nejkratší možnou trasu, která projde přes všechna tato města a vrátí se zpět do původního města.

Všimněme si, že je-li nějaký problém těžký a povede se nám ho vyřešit převedením na jiný problém (aniž bychom tím příliš zvýšili velikost vstupu), tak i tento problém je těžký.

Celočíselné programování

Definice. *Úlohou celočíselného programování* je úloha lineárního programování, kde navíc přidáme podmínky na celočíselnost proměnných.

Příklad 5. Napiš celočíselný program, který řeší *set cover problem*.

Důsledek. *Za předpokladu $P \neq NP$ se úloha celočíselného programování nedá řešit v polynomiálním čase.*

Definice. *Relaxace* celočíselného lineárního programu je lineární program, ve kterém se vynechají podmínky na celočíselnost. Řešení této úlohy lze pak často upravit na řešení původní celočíselné úlohy.

Příklad 6. (Párování maximální váhy, těžký) Navrhni algoritmus pro následující problém. Máme daný bipartitní graf se stejně velkými komponentami a s ohodnocenými hranami. Chceme nalézt perfektní párování (každý vrchol bude spojen s právě jedním ze svých sousedů), aby součet vah na hranách v párování byl co největší.

Aproximační algoritmy

Definice. Algoritmus je *α-aproximační*, pokud pro všechny možné vstupy vydá v polynomiálním čase výstup, jehož hodnota je alespoň $1/\alpha$-násobek pro maximalizační problém, resp. nejvýše α-násobek pro minimalizační problém.

Příklad 7. (Minimální vrcholové pokrytí) Nalezni 2-aproximační algoritmus pro problém *minimální vrcholové pokrytí*.

Příklad 8. (Set cover problem) Nalezni nějaký aproximační algoritmus na *set cover problem*.

Příklad 9. (Balancování grafu) Navrhni 2-aproximační algoritmus pro následující problém. Na vstupu je neorientovaný graf s kladnými váhami na hranách. Cílem je zorientovat všechny hrany, aby ten nejtěžší vrchol (vrchol s nějvětší součtem vah hran orientovaných k němu) měl co nejmenší možnou váhu.

Příklad 10. (Problém batohu) Najdi 2-aproximační algoritmus pro problém batohu, který je zadán následovně. Máme batoh s danou nosností a několik předmětů, každý se svojí váhou a cenou. Cílem je najít takovou podmnožinu předmětů, které se vejdou do batohu a jejichž součet cen je největší možný.

Příklad 11. (Metrické TSP) Nalezni aproximační algoritmus pro *problém obchodního cestujícího*, kde navíc jednotlivé vzdálenosti splňují trojúhelníkovou nerovnost.

Příklad 12. (Plánování) Nalezni 2-aproximační algoritmus pro následující plánovací problém. Máme několik počítačů a několik úkolů. Každý úkol má svoji délku a může být závislý na jiných úkolech, což znamená, že ho může počítač začít provádět, až když jsou všechny předchozí úkoly (na libovolném počítači) dokončené. Cílem je minimalizovat čas, kdy se dokončí všechny úkoly.

Příklad 13. (Problém k dodavatelů) Nalezni 3-aproximační algoritmus pro následující problém. Na vstupu máme $m+n$ bodů v rovině, kde m z nich jsou dodavatelé a zbylí jsou zákazníci. Úkolem je vybrat k dodavatelů tak, že minimalizujeme nejdelší vzdálenost mezi zákazníkem a jeho nejbližším vybraným dodavatelem.

Návody

1. Nakresli si příklad do roviny.

2. Zobecni krychli do více dimenzí.

3. Jako proměnné zvol poloměr, který se bude maximalizovat, a souřadnice středu. Využij vzoreček pro vzdálenost bodu od přímky.

4. Zvlášť vyřeš svislou přímku. Poté napiš program s ostrými nerovnostmi, kterých se následně zbav zavedením nové proměnné.

5. Vezmi proměnnou pro každou množinu a dej jí hodnotu 1 nebo 0 podle toho, jestli je mezi vybranými, nebo ne.

6. Napiš celočíselný program, který tuto úlohu řeší, a uvaž jeho relaxaci. Pak najdi způsob, jak zachovat optimalitu, ale z neceločíselných proměnných udělat celočíselné.

7. Sestav lineární program řešící tuto úlohu, uvaž jeho relaxaci a optimální řešení vhodně zaokrouhli.

8. Sestav celočíselný lineární program, který tuto úlohu řeší, a vezmi jeho relaxaci. Také můžeš uvážit relaxaci duálního programu. Pro důkaz, že jde o aproximační algoritmus, uvaž číslo odpovídající největšímu počtu množin, ve kterých se některý prvek vyskytuje.

9. Napiš lineární program, kde proměnná odpovídá každé orientované hraně (hodnota 1/0 znamená, že hrana je/není orientovaná tímto směrem), uvaž relaxaci a řešení zaokrouhli.

10. Setřiď předměty podle „hustoty“ (poměr cena/hmotnost).

11. Pro 2-aproximační algoritmus obejdi dvakrát minimální kostru grafu. Pro $1,5$-aproximační algoritmus nejprve sestav minimální kostru a pak na vrcholech s lichým stupněm nalezni minimální perfektní párování.

12. Posouvej se v čase a vždy umísti všechny možné úkoly. Využij toho, že optimum je alespoň součet všech délek úkolů podělený počtem počítačů a také alespoň tolik jako součet délek úkolů na libovolné cestě v závislostním grafu.

13. Vždy vezmi zákazníka, který je na tom nejhůře, a přidej jeho nejbližšího dodavatele.

Literatura a zdroje

[1] Jiří Matoušek: *Lineární programování a lineární algebra pro informatiky*, ITI Series 2006-311.

[2] D. P. Williamson, D. B. Shmoys: *The Design of Approximation Algorithms*, Cambridge University Press, 2011.

[3] Martin Böhm: *cvičení k Úvodu do aproximačních a pravděpodobnostních algoritmů*, *http://iuuk.mff.cuni.cz/ bohm/15-16/apxintro/*.