Rozhledy matematicko-fyzikální

Petr Zahradník

Lineární optimalizace

Rozhledy matematicko-fyzikální, Vol. 94 (2019), No. 4, 1–8

Persistent URL: http://dml.cz/dmlcz/148010

Lineární optimalizace

Petr Zahradník

Abstrakt. Lineární optimalizace hraje v oboru matematické informatiky velmi významnou roli. Její jednoduchá formulace, deterministická řešitelnost a prokazatelná optimalita ji předurčují k široké aplikaci napříč všemi obory našeho každodenního života. Přestože se může tato disciplína zdát komplikovanou, pro její pochopení stačí středoškolská matematika a trochu prostorové představivosti. Aniž bychom se pouštěli do složitých vět a důkazů, získáme náhled do teorie i praktického využití.

1. Všude samé problémy

Žádná oblast matematiky se neobejde bez motivace k jejímu vzniku. Ani lineární optimalizace není výjimkou, praktické využití celou disciplínu hnalo kupředu. Z mnoha příkladů můžeme vybrat například řízení výroby v továrně, přidělování zakázek řidičům taxi, stavbu ropovodů, svoz odpadu na skládku, tvorbu rozvrhu, zefektivňování integrovaného záchranného systému, skládání jídelníčku… Zkrátka příkladů je nepřeberně a během čtení tohoto článku vás jistě napadnou mnohé další. My však začneme poněkud jednodušeji:

Příklad. Zahradník Petr by rád vysadil ovocné stromy: jabloně a hrušně. Jabloň stojí 100 peněz, hrušeň 300 peněz, nicméně z hrušně dostane zahradník dvakrát větší úrodu. Kolik kterých stromů má Petr vysadit, pokud má na zahrádce místo pro 7 stromů, jeho rozpočet je 1 300 peněz a rád by získal co možná největší úrodu?

Řešení není těžké uhodnout, zkusme se ale zamyslet nad obecnými postupy řešení. Oba stromy zabírají jedno místo na zahrádce, hrušeň je však dvakrát výnosnější, je však také třikrát dražší. Když nakoupíme co nejvíce stromů jednoho druhu, brzy narazíme na jedno z těchto omezení a přitom zůstane rezerva v druhém omezení. Zdá se tedy, že je nutné najít nějakou střední cestu, která vyváží obě kritéria a maximalizuje výnos.

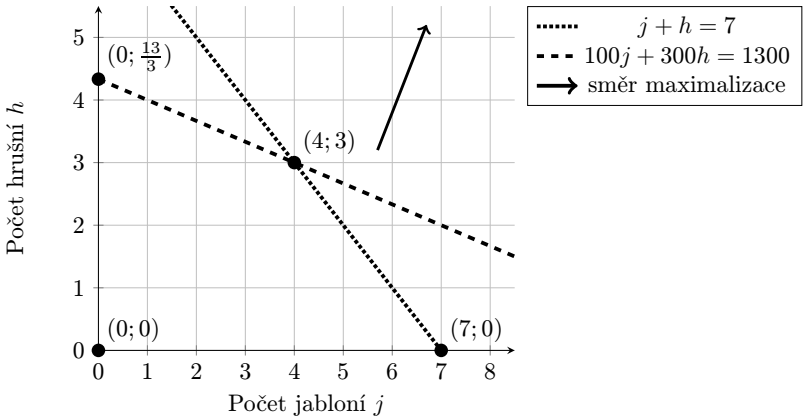
Zkusme tedy nejprve nakoupit 7 jabloní. Tím jsme zcela zaplnili zahrádku, zbyly nám ale nějaké peníze. Můžeme tedy vyměnit 3 jabloně za 3 hrušně. Tím vyčerpáme místo i peníze a máme vyhráno. Pokud zkusíme opačný postup, nakoupíme nejprve 4 hrušně. Za zbylou sto-bankovku dokoupíme jednu jabloň. Nyní můžeme opět vyměňovat pouze jedním

směrem – místo hrušně zasadíme jabloň a tím ušetříme dvě stě peněz. To je ale přesně tolik, kolik potřebujeme na doplnění zahrady jabloněmi.

V obou případech jsme došli ke stejnému řešení – 4 jabloně a 3 hrušně. Ekvivalent výnosu je 10 a je maximální možný. Pro důkaz, že je řešení správné, použijeme jednoduchý geometrický náhled. Nejprve ale trocha značení. Počet jabloní označíme j, počet hrušní h. Podmínku pro místo na zahrádce zapíšeme jako $j + h \leq 7$, podmínku pro peníze potom jako $100j + 300h \leq 1\,300$. Maximalizovat se snažíme funkci ve tvaru $Z = j + 2h$. To nám dohromady dává obvyklou formulaci problému z lineární optimalizace, tzv. *kanonický tvar*:

$$\begin{aligned}&\text{maximalizovat } Z = j + 2h,\\ &\text{pokud platí } j + h \leq 7\\ &\text{a zároveň } 100j + 300h \leq 1300,\\ &j,\ h \text{ jsou nezáporné.}\end{aligned}$$

Všechny podmínky a vlastně i cílová funkce jsou lineární ve dvou proměnných, můžeme si je proto zobrazit jako přímky a poloroviny v kartézské soustavě souřadnic. Maximalizovaná funkce potom určuje směr, ve kterém hodnota řešení roste:



Obr. 1: Grafické znázornění rovnic podmínek a jejich průsečíků

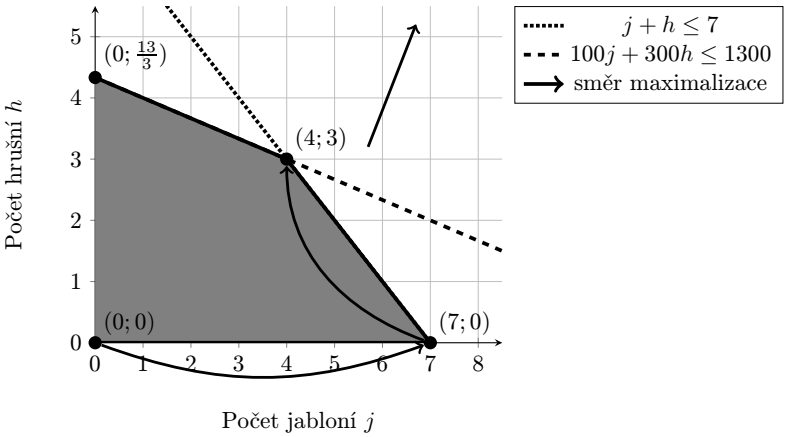
2. Konvexní skákání

Vidíme, že nám vznikly čtyři průsečíky, a souřadnice každého průsečíku vlastně označují kombinaci počtu nakoupených jabloní a hrušní.

Pokud dokážeme po průsečících *skákat*, dojdeme k nejlepšímu řešení. Označme oblast vymezenou podmínkami a souřadnicovými osami jako oblast řešení. Každý bod z oblasti řešení je tedy platným řešením, k hledání optimálního řešení se nám budou hodit tři pozorování:

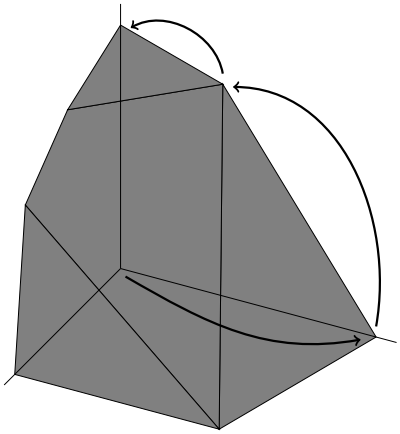
1. Oblast řešení je konvexní mnohoúhelník. To vyplývá z toho, že se vlastně jedná o průnik polorovin.
2. Nejvzdálenějším bodem oblasti řešení v každém směru je vrchol. Pro důkaz vezměme libovolnou stranu mnohoúhelníka. Pokud je kolmá na hledaný směr, pak jsou všechny její body (a tedy i oba vrcholy) stejně vzdálené. Pokud kolmá není, potom je jeden z vrcholů vzdálenější než všechny její ostatní body, neboť vzdálenost bodů je lineární, a tedy monotónní. Jelikož je mnohoúhelník konvexní, je extrémní bod právě jeden nebo právě jedna celá strana.
3. Do nejvzdálenějšího vrcholu se umíme dostat pomocí skoků po sousedních vrcholech. Skokem rozumíme přesun z nějakého vrcholu do sousedního vzdálenějšího vrcholu. Ať už začneme kdekoli, můžeme skákat po obvodu mnohoúhelníka a stále se v daném směru *vzdalovat* právě do chvíle, než skončíme v optimálním vrcholu (takové vrcholy mohou být nejvýše dva). Kdybychom po cestě k tomuto vrcholu museli skočit *dozadu* v daném směru, nebyl by mnohoúhelník konvexní.

Pokud tedy známe oblast řešení a průsečíky na jejím obvodu, můžeme je proskákat až k optimálnímu vrcholu. A to je přesně to, co jsme udělali v ukázkovém příkladě!



Obr. 2: Grafické znázornění úlohy s vyznačenými skoky po průsečících

Můžeme si jednoduše rozmyslet, že tento postup funguje i pro problémy s více proměnnými a více podmínkami. Jenom se nám pak z 2D oblasti řešení stane konvexní mnohodimenzionální mnohostěn. V angličtině se používá výraz simplex, proto byl algoritmus pojmenován *Simplex algorithm*. Jeho tvůrcem je americký matematik minulého století George Bernard Dantzig [2]. Stejně tak můžeme místo maximalizace minimalizovat, pokud řešení posuzujeme například podle trestných bodů či ztrát zisku.



Obr. 3: Grafické znázornění simplexového algoritmu ve 3D

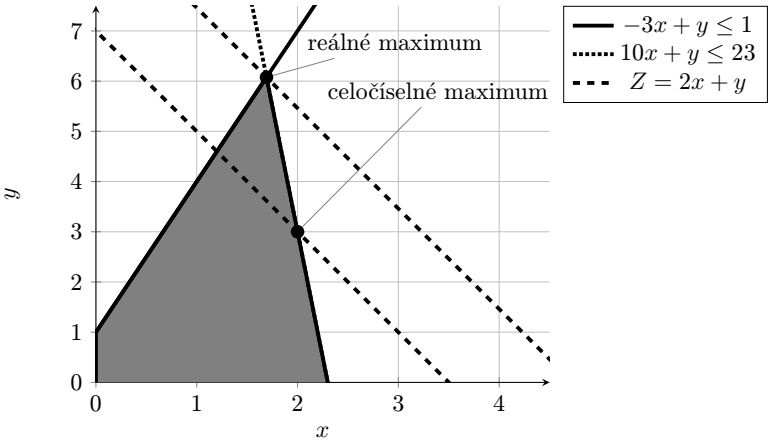
Představa skákání po vrcholech je výborná pro nahlédnutí, nicméně ne zrovna matematicky korektní. Jak se hledají průsečíky? Které průsečíky leží na obvodu oblasti řešení? Pokud máme více možností, kterým směrem skočit? Co když úloha nemá řešení? Co když má úloha řešení v nekonečnu? Pro odpověď na tyto otázky je nutné zavést formálně správný algoritmus tak, jak to kdysi udělal Dantzig [1].

On si totiž všiml, že pokud budeme uvažovat o seznamu podmínek jako o soustavě lineárních rovnic, skok do vrcholu odpovídá Gaussově eliminaci jedné proměnné. Pokud postupně vyeliminujeme všechny proměnné, skončíme v optimálním vrcholu. Podrobnější popis přesahuje rozsah tohoto článku, v literatuře i na internetu je však mnohokrát popsán. Jelikož se až na drobné úpravy jedná o klasické středoškolské řešení soustavy, je algoritmus hledání řešení velmi rychlý a jednoduchý, v průměrném případě běží kvadraticky dlouho vzhledem k počtu proměnných a podmínek [7].

3. Polynomiálně či nepolynomiálně?

Máme tedy k dispozici velmi mocný nástroj pro řešení lineární optimalizace a zdaleka ne jediný, podobných algoritmů je hned několik. *Metoda vnitřních bodů* (angl. *Interior-point method*) [4] nebo *Metoda elipsoidů* (angl. *Ellipsoid method*) [5] jsou jen některé z nich. V čem je tedy háček? Ti bystřejší z vás si jistě všimli bodu o souřadnicích 0 a $\frac{13}{3}$ na předchozích obrázcích. Přestože je matematicky zcela v pořádku nakupovat stromy po třetinách, zahradník by nás za takové řešení nepochválil. Pokud například zvýšíme rozpočet na 1 400 peněz, je optimální nakoupit od každého druhu tři a půl stromu. Zkuste si promyslet, jak naše řešení opravit.

Možná vás napadly různé způsoby řešení zaokrouhlit nebo prohledat nejbližší celočíselné body. Musím vás však zklamat. Neexistuje žádný jednoznačný postup, jak obecně *opravit* reálné řešení na celočíselné. Jednoduchým příkladem budiž obr. 4:



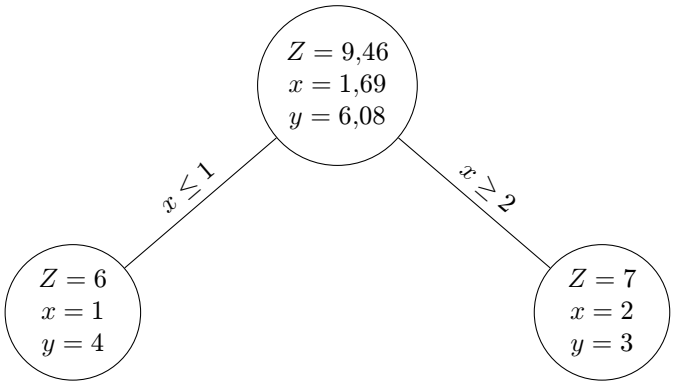
Obr. 4: Grafické znázornění možné vzdálenosti celočíselného a reálného maxima. Zatímco reálné maximum v bodě $(1{,}69; 6{,}08)$ je $Z = 9{,}46$, hledané celočíselné maximum $Z = 7$ je až v bodě $(2; 3)$.

Problém *hledání celočíselného řešení* (angl. *Integer linear programming*, ILP) je tak složitý, že jej neumíme řešit efektivně. Formálně řečeno asymptotický odhad doby strávené deterministickým řešením roste exponenciálně vzhledem k počtu proměnných. Pro takovéto problémy se

ujalo označení třída složitosti NP, aby se odlišily od problémů, jejichž složitost umíme odhadnout polynomem (mnohočlenem) závislým na počtu proměnných. Pokud tedy máme problém velkého rozsahu (mnoho proměnných a podmínek), algoritmy z třídy NP budou zpravidla mnohem pomalejší a odhad výpočetní doby přesahuje stáří vesmíru. Není však třeba házet flintu do žita, i nepolynomiální algoritmus může s velkou pravděpodobností nalézt řešení během několika sekund. Pojďme si jeden takový ukázat.

4. Řežba

S jednou z prvních metod řešení ILP přišli pánové A. H. Land a A. Doig už v roce 1960 [6]. Její název *Metoda větví a mezí* (angl. *Branch and bound method*) je naprosto vypovídající. Pro tuto metodu je důležité, abychom uměli hledat reálná řešení, třeba právě simplexovou metodou. Po vyřešení vybereme libovolnou neceločíselnou proměnnou a rozvětvíme problém na dva případy zaokrouhlením proměnné nahoru a dolů, jak ilustruje obr. 5.



Obr. 5: Prohledávací strom vytvořený metodou větví a mezí. Celočíselné řešení je zde nalezeno již po větvení na první proměnné a není třeba dále větvit

Větvení na proměnné $x_i = k$ je formálně vlastně přidání podmínky $x_i \leq \lfloor k \rfloor$ resp. $x_i \geq \lceil k \rceil$. Taková podmínka *vyřízne* z oblasti řešení pás neceločíselných bodů. Je jasné, že takto můžeme postupně větvit na všech n proměnných a získat řešení v čase 2^n. Řešení ale můžeme zrychlit pomocí tzv. mezí. Víme, že celá čísla jsou podmnožinou reálných čísel. Pokud

tedy úloha nemá reálné řešení, pak nemá ani celočíselné řešení. Nebo pokud je reálné řešení horší než nejlepší doposud nalezené, celočíselné nemůže být lepší. Tímto způsobem můžeme prohledávání většiny větví zastavit už velmi brzy a ušetřit si spoustu práce. Jedná se sice o heuristický postup (tedy ne obecně platný), pro většinu problémů však velmi rychlý.

Tento poměrně jednoduchý nápad stojí společně s *Chvátalovými–Gomoryovými řezy* a heuristikou *Branch and price* v základu téměř všech ILP algoritmů.

5. Binární nesplnitelnost

Existuje jedna skupina problémů, které mají osud ještě krutější. Jedná se o binární lineární optimalizaci (angl. *Binary linear programming* – BIP), problémy, jejichž proměnné mohou nabývat pouze hodnot 0 a 1. Tato úloha je zapsána na slavném Karpově seznamu 21 NP-úplných problémů [3]. To je seznam problémů z třídy složitosti NP, které umíme mezi sebou převádět, žádný však neumíme řešit v polynomiálním čase. Kdo vyřeší jeden problém, vyřeší všechny. Není však jasné, zda efektivní algoritmus existuje, nebo existovat nemůže. Za vyřešení této otázky (známé jako *P vs NP*) je slíbeno mnoho finančních ocenění a věčná sláva.

Nezbývá nám tedy nic jiného než spokojit se s řešením v nejhůře exponenciálním čase a hledat heuristiky (postupy rychle hledající nekompletní nebo nepřesná řešení) a metaheuristiky (postupy hledající pouze dostatečně dobrá řešení). Často nám například stačí pouze lokální maximum (či minimum), například pokud je naším cílem pouze překonat nějakou mez nebo splnit alespoň polovinu podmínek. Výzkum na poli lineární optimalizace stále pokračuje kupředu a možnost řešit i rozsáhlé problémy (řádově tisíce proměnných a miliony podmínek) se stává dostupnou.

Literatura

[1] Dantzig, G. B.: *Linear Programming and Extensions.* United States Air Force Project RAND, R-366-PR, Princeton University Press, Princeton, 1963.

[2] Dantzig, G. B.: *A History of Scientific Computing.* Origins of the Simplex Method, Nash, S. G. (ed.), ACM, New York, NY, 1990, s. 141–151.

[3] Karp, R. M.: Complexity of Computer Computations, *Reducibility among Combinatorial Problems.* Miller, R. E., Thatcher, J. W., Bohlinger, J. D. (eds.), Springer, Boston, MA, 1972, s. 85–103.

[4] Karmarkar, N.: A new polynomial-time algorithm for linear programming. *Combinatorica*, roč. 4 (1984), č. 4, s. 373–395.

[5] Khachiyan, L. G.: Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, roč. 20 (1980), č. 1, s. 53–72.

[6] Land, A. H., Doig, A. G.: An automatic method of solving discrete programming problems. *Econometrica*, roč. 28 (1960), č. 3, s. 497–520.

[7] Spielman, D. A., Teng, S.-H.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, roč. 51 (2004), č. 3, s. 385–463.

Krátký příběh o obecném tvaru Pythagorovy věty

Vlastimil Dlab, Bzí u Železného Brodu

Abstrakt. This article presents the following simple property of a general triangle ABC: Let D be an arbitrary point and A_0, B_0, C_0 the feet of the perpendiculars from D on the (possibly extended) sides BC, CA, AB, respectively. Then

$$|AC_0|^2 + |BA_0|^2 + |CB_0|^2 = |C_0B|^2 + |A_0C|^2 + |B_0A|^2.$$

This statement is a proper generalization of the Pythagorean theorem. Surprisingly, it does not appear in textbooks or other publications.

Dobří učitelé vědí, že každá hodina matematiky by měla být zajímavým a přitažlivým příběhem. To platí pro výuku všeobecně, ale pro výuku na školách především. Pro matematiku je to o to důležitější, že takový přístup pomyslné matematické záhady polidšťuje a tím jakékoliv obavy, či dokonce strach z tohoto předmětu zmírňuje. Ano, každá zajímavá matematická historka sdělená s pohlazením, úsměvem a nadšením zahání strašáka symbolů a zbytečného biflování.

Dnešním úkolem je odhalit tajemství tohoto tvrzení:

V rovině uvažujeme trojúhelník ABC a libovolný bod D. Nechť A_0 je pata kolmice z bodu D na přímku určenou vrcholy B a C, B_0 pata kolmice z bodu D na přímku určenou vrcholy C a A a C_0 pata kolmice z bodu D na přímku určenou vrcholy A a B. Potom

$$|AC_0|^2 + |BA_0|^2 + |CB_0|^2 = |C_0B|^2 + |A_0C|^2 + |B_0A|^2. \qquad (*)$$