Optimalizační Úlohy (IA102 “OU”)

Doc. RNDr. Petr Hliněný, Ph.D.

hlineny@fi.muni.cz

22. prosince 2005

Verze 0.9.

Obsah

0.1 Předmluva

Vážení čtenáři,

dostává se vám do rukou výukový text pro předmět Optimalizační Úlohy, který má vysokoškolské studenty (především na magisterském stupni studia) uvést do problematiky kombinatorické, lineární a celočíselné optimalizace. Tento text je určen především studentům s informatickým nebo matematickým zaměřením. Přitom je látka rozvržena tak, aby si z ní odnesli hodnotné poznatky jak čtenáři hledající praktické návody k formulaci a řešení běžně se vyskytujících optimalizačních úloh, tak i studující s hlubším zájmem o matematickou teorii skrývající se za optimalizací.

Od čtenáře přepokládáme dobrou znalost lineární algebry na úrovni běžného obecného kurzu a alespoň základní znalosti kombinatoriky a úvodu topologie. Zároveň očekáváme zběhlost čtenáře ve všech běžných informatických pojmech, jak v oblasti formulace a popisu algoritmů, tak i v základech výpočetní složitosti. (Pro případné doplnění těchto pojmů odkazujeme třeba na [4])

Na rozdíl od některých (dle našeho mínění špatně) zažitých postupů výuky lineární optimalizace zde neklademe hlavní důraz na bezduché memorování postupů simplexové metody, nýbrž se především snažíme čtenáři ukázat, co to vlastně jsou "optimalizační úlohy" a jak je správně matematicky "uchopit" a formulovat. Zároveň ukážeme základní matematické principy a postupy, na kterých je založeno řešení lineárních optimalizačních úloh simplexovou metodou a celočíselných úloh metodou větvení a mezí (branch&bound).

- V úvodní části si přiblížíme na několika *kombinatorických problémech* (jako minimální kostra, matroidy, toky v sítích, rozvrhování) principy formulace a vyřešení optimalizační úlohy. Mimo jiné si řekneme o tzv. hladovém postupu řešení a o principu dobré charakterizace.
- V druhé části se zaměříme na úlohy spojité *lineární optimalizace*. Ukážeme si, jak se mnohé prakticky založené přílady formulují jako úlohy lineární optimalizace (také se říká "lineární programování" LP). Pak vysvětlíme některé důležité pojmy polyedrální kombinatoriky jako mnohostěny, konvexitu a dualitu úloh LP. Na ně navážeme popisem teoretických principů simplexové metody pro řešení těchto úloh LP i popisem základních praktických implementací *simplexové metody* (včetně běžných triků s umělými proměnnými a degenerovanými řešeními).
- Ve třetí části přejdeme k různým úlohám tzv. *diskrétní optimalizace*, ve kterých se řešení nechovají spojitě, nýbrž "po diskrétních skocích". Typickým představitelem jsou úlohy celočíselné lineární optimalizace (také zvané "celočíselné programování" IP). Jedná se o velmi bohatou oblast, ve které mnohdy ani není jasné, jak problém převést správně do matematického jazyka, natož jak jej efektivně vyřešit. Kromě mnohých ukázek formulace úloh bude vysvětleno řešení úloh *celočíselného programování* metodou větvení a mezí. Doplňkově bude naznačen výhled k některým dalším, složitějším typům úloh diskrétní optimalizace.

Při prezentaci látky vycházíme z autorových vlastních poznatků a zkušeností z dob studia na MFF UK a na Georgia Tech. Navazujeme látku na následující základní literaturu: Janáček [3], Nemhauser–Wolsey [7] a online dostupné Schrijver [10]. Pro praktické řešení běžných úloh LP a IP používáme volně dostupné programové prostředky, z nichž především odkazujeme na online rozhraní [12] a [15].

Výklad látky lze samozřejmě vhodně uzpůsobit zájmům studujících. Například čtenáři s primárním zájmem o praktické řešení optimalizačních úloh mohou přeskočit většinu matematické teorie prezentované v Lekcích 4, 5 a 12, 13. Na druhou stranu pro stu-

dující, kteří se zajímají i o hlubší matematickou teorii stojící za různými zde probranými oblastmi optimalizace, prezentujeme již zmíněné teoreticky zaměřené Lekce 4, 5, 12, 13 a na konci každé lekce přidáváme komentované odkazy na literaturu vhodnou k dalšímu studiu látky.

Ve stručnosti se zde zmíníme o formální struktuře našeho textu. Přednesený materiál je dělený do jednotlivých lekcí (kapitol), které zhruba odpovídají obsahu týdenních přednášek v semestru. Lekce jsou dále děleny tematicky na oddíly. Výklad je strukturován obvyklým matematickým stylem na definice, tvrzení, úlohy, algoritmy, případné důkazy, poznámky a neformální komentáře. Navíc je proložen řadou vzorově řešených příkladů navazujících na vykládanou látku a doplněn dalšími otázkami a úlohami. (Otázky a úlohy označené hvězdičkou patří k obtížnějším a nepředpokládáme, že by na ně všichni studující dokázali správně odpovědět bez pomoci.) Správné odpovědi k otázkám a úlohám jsou shrnuty na konci textu.

Přejeme vám mnoho úspěchů při studiu a budeme potěšeni, pokud se vám náš výukový text bude líbit. Jelikož nikdo nejsme neomylní, i v této publikaci zajisté jsou nejasnosti či chyby, a proto se za ně předem omlouváme. Pokud chyby objevíte, dejte nám prosím vědět e-mailem

mailto:hlineny@fi.muni.cz.

Petr Hliněný, autor

Pár slov o vzniku

Tento učební text vznikal v průběhu let 2004 až 2005 podle autorových přednášek a cvičení předmětu Optimalizační Úlohy na FEI VŠB – TUO a dále na FI MU. Vyšel z původních krátkých počítačových zápisků přednášek pořízených v roce 2004 studenty VŠB A. Danielem, M. Dudou, V. Michalcem, M. Ptáčkem, B. Rylkem, M. Strakošem, N. Ciprichem a M. Krucinou. Dále se na přípravě textu podíleli v roce 2005 studenti FI MU J. Mareček, M. Maška, R. Vágner, M. Vlk a Mgr. T. Brázdil, kteří zčásti opravovali stávající text a doplňovali nové úlohy či důkazy.

Významné poděkování za vznik tohoto textu a ostatních výukových pomůcek k diskrétní optimalizaci patří také Fondu rozvoje vysokých škol ČR, který nám na přípravu studijních materiálů v roce 2005 poskytl grant **FRVŠ 2270/2005**. V rámci tohoto projektu byly především dále vytvořeny jako výukové pomůcky následující dvě počítačové aplikace s veřejným přístupem z internetu: Web rozhraní [15] (N. Ciprich) k volnému projektu řešení lineární a celočíselné optimalizace, použitelné i na primitivních mobilních zařízeních (PDA). Dále komfortní java rozhraní [13] (M. Krucina) k našemu programu Macek pro strukturální výpočty matroidů.

Část I
Několik Úvodních Úloh

1 O kombinatorické optimalizaci

Úvod

Pod pojmem "optimalizace" se skrývá celá široká škála úloh, jejichž podstatu lze zhruba shrnout následovně: Jsou dány jisté omezující podmínky, které popisují obor přípustných řešení úlohy. Dále je dána tzv. účelová funkce, která jednotlivým řešením přiřazuje jejich hodnotu a vzhledem ke které pak hledáme optimální (minimální či maximální, dle kontextu) řešení úlohy.

Optimalizační úlohy se dále (zhruba) dělí na hlavní podoblasti podle charkteru oboru přípustných řešení – spojité × diskrétní, a podle charakteru účelové funkce (a omezujících podmínek) – lineární, kvadratické, či jiné... Celý náš výklad se bude týkat především oblastí diskrétní (také kombinatorické) a lineární optimalizace.

*Přibližme si nejprve význam slova "diskrétní". (Ne, nemá to nic společného s udržením něčeho v tajnosti, to je jen bezvýznamná shoda slov v češtině.) V diskrétních úlohách se obor přípustných řešení skládá z několika izolovaných bodů či oblastí, neboli je obvykle vyjádřený celými čísly. (Například při optimalizaci osobní dopravy nelze dost dobře poslat polovinu člověka jedním autobusem a druhou jeho polovinu jiným, že?) Úlohy takového typu se nejčastěji vyskytují v kombinatorice, a proto se také někdy používá spojení kombinatorická optimalizace. Na úvod si ukážeme několik úloh, které se dají dobře řešit tím asi nejjednodušším způsobem, tzv. hladovým postupem – **bereme vždy to nejlepší, co se zrovna nabízí**...*

Cíle

Tato úvodní lekce nám v prvé řadě přiblíží obecný pojem "optimalizační úlohy" jako takové. Dále demonstruje velice jednoduchý postup tzv. hladové optimalizace na řešení několika kombinatorických problémů. V souvislosti s hladovým algoritmem si také řekneme, co to jsou matroidy a proč na nich hladový algoritmus funguje vždy optimálně.

1.1 Hladový algoritmus

Asi nejprimitivnějším možným přístupem při řešení optimalizačních úloh v kombinatorice je postup stylem ***beru vždy to nejlepší, co se zrovna nabízí***... Tento postup obecně v češtině nazýváme *hladovým algoritmem*, i když lepší by bylo použít správnější překlad anglického "greedy", tedy nenasystný algoritmus. A ještě hezčí české spojení by bylo "algoritmus hamouna". Jednoduše bychom jej nastínili takto:

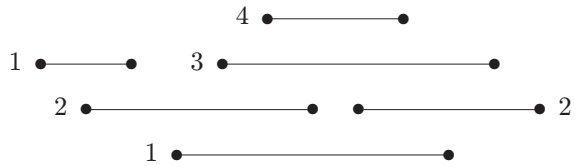
- Postupně v krocích vyber vždy to nejlepší, co se dá (nabízí).
- To vyžaduje zvolit *uspořádání na objektech*, ze kterých vybíráme.
- Průběh a úspěch algoritmu silně závisí na tomto zvoleném uspořádání (které již dále neměníme).

`Komentář:` Jak asi každý ví, nenasystnost či hamounství nebývá v životě tím nejlepším postupem, ale kupodivu tento princip perfektně funguje v mnoha kombinatorických úlohách! Jedním známým příkladem je třeba *hledání minimální kostry* uvedené v příští lekci. Jiným příkladem je třeba jednoduchý problém přidělování (uniformních) pracovních úkolů, na němž si nejprve hladový algoritmus přiblížíme.

Úloha 1.1. ***Přidělení pracovních úkolů***
Uvažujeme zadané pracovní úkoly, které mají přesně určený čas začátku i délku trvání. (Jednotlivé úkoly jsou tedy reprezentovány uzavřenými intervaly na časové ose.) Všichni pracovníci jsou si navzájem rovnocenní – uniformní, tj. každý zvládne všechno.

Vstup: Časové intervaly daných úkolů.
Výstup: Přidělení úkolů pracovníkům, aby celkově bylo potřeba co nejméně pracovníků.

`Komentář:` Pro příklad zadání takové úlohy si vezměme následující intervaly úkolů:



Kolik je k jejich splnění potřeba nejméně pracovníků? Asi sami snadno zjistíte, že 4 pracovníci stačí, viz zobrazené očíslování. Ale proč jich nemůže být méně?

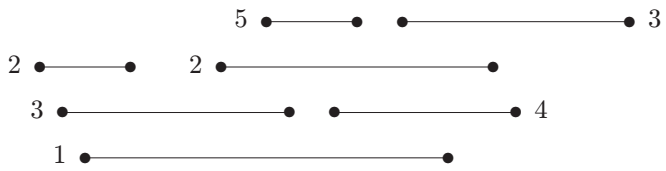
`Poznámka`: Uvedená úloha může být kombinatoricky popsána také jako problém optimálního obarvení daného intervalového grafu (vrcholy jsou intervaly úkolů a hrany znázorňují překrývání intervalů).

Algoritmus 1.2. ***Hladový algoritmus rozdělení pracovních úkolů.***
Úloha 1.1 je vyřešena následující aplikací hladového postupu:

1. *Úkoly nejprve seřadíme podle časů začátků.*
2. *Každému úkolu v pořadí přidělíme volného pracovníka s nejnižším číslem.*

`Důkaz`: Nechť náš algoritmus použije celkem k pracovníků. Dokážeme jednoduchou úvahou, že tento počet je optimální – nejlepší možný. V okamžiku, kdy začal pracovat pracovník číslo k, všichni $1, 2, \ldots, k-1$ také pracovali (jinak bychom vzali některého z nich). V tom okamžiku tedy máme k překrývajících se úkolů a každý z nich vyžaduje vlastního pracovníka. □

`Komentář:` Příklad neoptimálního přidělení pracovních úkolů dostaneme například tak, že na začátku úkoly seřadíme podle jejich časové délky. (Tj. čím delší úkol, tím dříve mu hladově přiřadíme pracovníka.)



Takový postup by se také mohl zdát rozumný, vždyť se v praxi často rozdělují nejprve ty velké úkoly a pak průběžně ty menší. Vidíme však na obrázku, že nalezené řešení není optimální – vyžaduje 5 místo 4 pracovníků.

Je tedy velmi důležité, podle jakého prinicipu seřadíme objekty (úkoly) na vstupu.

Doplňkové otázky

(1.1.1) *Proč tedy nestačí méně než 4 pracovníci pro splnění pracovních úkolů v zadání za Úlohou 1.1?*

(1.1.2) *Co kdybychom v hladovém řešení Úlohy 1.1 seřadili úkoly podle časů jejich ukončení?*

1.2 Problém minimální kostry

V dalším oddílu se podíváme na jeden problém z obasti grafů. Vzpomeňme si, že obecný graf bez kružnic je nazýván *les* a jeho souvislé komponenty jsou *stromy*. Blíže viz třeba [1, Lekce 9].

Definice 1.3. ***Kostrou grafu*** G rozumíme podgraf v G,
který je lesem, obsahuje všechny vrcholy grafu G a na každé souvislé komponentě G indukuje strom.

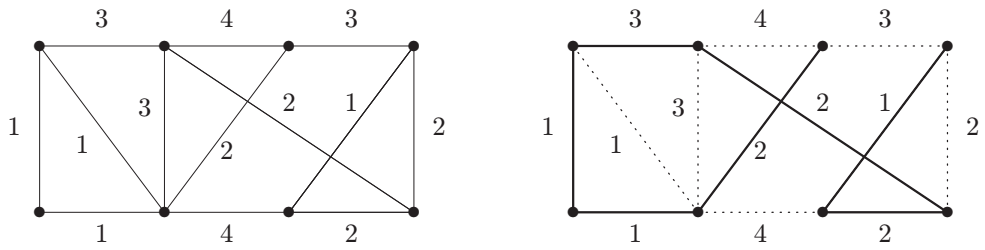
`Komentář:` Kostra daného grafu je minimální podgraf, který zachovává souvislost každé komponenty původního grafu. Proto nám vlastně ukazuje "minimální propojení" daných vrcholů, ve kterém ještě existují cesty mezi všemi dvojicemi, které byly propojeny i původně.

Úloha 1.4. ***Problém minimální kostry (MST)***
Je dán souvislý vážený graf G, w s nezáporným ohodnocením hran w. Otázkou je najít kostru T v grafu G, která má ze všech koster nejmenší celkové ohodnocení. Formálně:

Vstup: Souvislý graf G s nezáporným ohodnocením hran $w : E(G) \to \mathbb{R}^+$.
Výstup: Kostra v G s minimálním součtem hodnot hran

$$\min_{\text{kostra } T \subseteq G} \left(\sum_{e \in E(T)} w(e) \right).$$

Praktickou formulací úlohy je třeba propojení domů elektrickým rozvodem, propojení škol internetem, atd. Zde nás ani tak nezajímají délky cest mezi propojenými body, ale hlavně celková délka či cena vedení/spojení, které musíme postavit. Vstupní graf nám přitom udává všechny možné realizovatelné propojky s jejich cenami. Příklad je uveden na následujícím obrázku i s vyznačenou minimální kostrou vpravo.

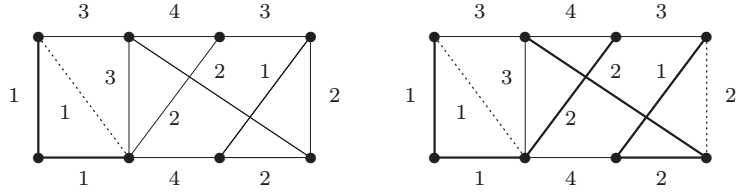


Algoritmus 1.5. ***Hladový algoritmus hledání minimální kostry v grafu.***
Úloha 1.4 je vyřešena následující aplikací hladového postupu:

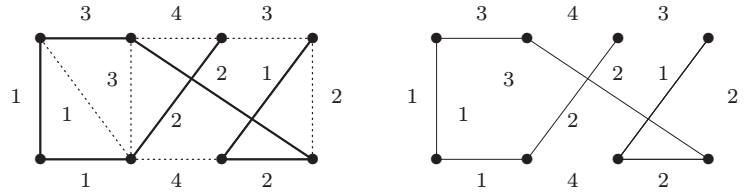
1. *Hrany grafu seřadíme podle jejich ohodnocení w od nejmenší.*
2. *Přidáváme do (budoucí) kostry postupně hrany, které nevytváří s dříve vybranými hranami kružnici. (Hrany uzavírající kružnice ignorujeme – "zahodíme".)*

`Komentář:` Pro ilustraci si podrobně ukážeme postup hladového algoritmu pro vyhledání kostry výše zakresleného grafu.

Hrany si nejprve seřadíme podle jejich vah $1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4$. (Na pořadí mezi hranami stejné váhy nezáleží, proto jej zvolíme libovolně.) Začneme s prázdnou množinou hran (budoucí) kostry. Pak hladovým postupem přidáme první dvě hrany váhy 1, v obrázku vlevo dole, které nevytvoří kružnici. Třetí hrana váhy 1 vlevo s nimi už tvoří trojúhelník, a proto ji přidat nelze, je zahozena. V obrázku průběhu algoritmu používáme tlusté čáry pro vybrané hrany kostry a tečkované čáry pro zahozené hrany:



Poté přejdeme na hrany s vahou 2, z nichž lze tři postupně přidat bez vytvoření kružnice a čtvrtá (úplně vpravo) již kružnici vytvoří a je proto zahozena. Viz. obrázek vpravo. Nakonec ještě přidáme hranu nejmenší vyšší váhy 3 vlevo nahoře a zbylé hrany již zahodíme, protože všechny tvoří kružnice.



Získáme tak minimální kostru velikosti $1+2+2+3+1+1+2=12$, která je v tomto případě (náhodou) cestou, na posledním obrázku vpravo.

Poznamenáváme, že při jiném seřazení hran stejné váhy by kostra mohla vyjít jinak, ale vždy bude mít stejnou velikost 12. (Například místo levé svislé hrany může obsahovat přilehlou úhlopříčku stejné váhy 1.)

Základní hladový algoritmus pro hledání minimální kostry byl popsán Kruskalem. Jiné (a mnohem starší) varianty výše popsaného algoritmu jsou následující:

- **Jarníkův algoritmus**
  Hrany na začátku neseřazujeme, ale začneme kostru vytvářet z jednoho vrcholu a v každém kroku přidáme nejmenší z hran, které vedou z již vytvořeného podstromu do zbytku grafu.

  `Komentář:` Toto je velmi vhodný algoritmus pro praktické výpočty a je dodnes široce používaný. Málokdo však ví, že pochází od Vojtěcha Jarníka, známého českého matematika — ve světové literatuře se obvykle připisuje američanu Primovi, který jej objevil skoro 30 let po Jarníkovi.

- **Borůvkův algoritmus**
  Toto je poněkud složitější algoritmus, chová se jako Jarníkův algoritmus spuštěný zároveň ze všech vrcholů grafu najednou. Viz popis v [6, Sekce 4.4]. Jedná se o historicky vůbec první algoritmus pro minimální kostru z roku 1928, který se pak stal inspirací i pro Jarníkův algoritmus.

Důkaz správnosti hladového algoritmu pro hledání minimální kostry bude dále podán v obecnosti u pojmu matroidu v příští sekci.

Doplňkové otázky

(1.2.1) *Co se stane, pokud v Algoritmu 1.5 seřadíme hrany naopak, tedy sestupně?*

(1.2.2) *Čím je Jarníkův algoritmus pro MST výhodnější než základní hladový postup?*

(1.2.3) *Promyslete si Jarníkův algoritmus, jaké datové struktury potřebujete pro jeho co nejrychlejší implementaci?*

1.3 Pojem matroidu

Pojem matroidu se často vyskytuje ve spojení kombinatorickou optimalizací, viz třeba mnohé práce Edmondse. Jedná se o pojem velice "obtížný k uchopení", a proto si o něm zde uvedeme jen několik základních poznatků v souvislosti s hladovým algoritmem (Věta 1.11). Více viz Lekce 13.

Definice 1.6. ***Matroid*** na množině X, značený $M=(X,\mathcal{N})$,
je takový systém $\mathcal{N}$ podmnožin nosné množiny X, ve kterém platí následující:

1. $\emptyset \in \mathcal{N}$
2. $A \in \mathcal{N}$ a $B \subseteq A \;\Rightarrow\; B \in \mathcal{N}$
3. $A, B \in \mathcal{N}$ a $|A| < |B| \;\Rightarrow\; \exists y \in B \setminus A : A \cup \{y\} \in \mathcal{N}$

Množinám ze systému $\mathcal{N}$ říkáme *nezávislé množiny*. Těm ostatním pak říkáme *závislé*. Nezávislým množinám, do kterých již nelze přidat žádný prvek tak, že zůstanou nezávislé, říkáme *báze matroidu*.

`Komentář:` Nejdůležitější částí definice matroidu je zvýrazněný třetí bod. Přímo ukázkový příklad matroidu nám dává lineární algebra – všechny lineárně nezávislé podmnožiny vektorů tvoří matroid. Odtud také pocházejí pojmy nezávislosti a báze matroidu, které přímo odpovídají příslušným pojmům vektorového prostoru.

Lema 1.7. *Všechny báze matroidu obsahují stejně mnoho prvků.*

`Důkaz`: Toto přímo vyplývá z třetí vlastnosti definice matroidu: Pokud nezávislá množina A má méně prvků než báze B, tak do A lze vždy přidat další prvek x tak, že zůstane $A \cup \{x\}$ nezávislá. □

Nyní uvedeme několik poznatků o stromech, které jsou relevantní pro zavedení "grafových" matroidů.

Lema 1.8. *Les na n vrcholech s c komponentami souvislosti má přesně $n - c$ hran.*

`Důkaz`: Každý vrchol lesa L náleží právě jedné komponentě souvislosti z definice. Jak známo, každý strom, tj. komponenta lesa L, má o jednu hranu méně než vrcholů. Ve sjednocení c komponent tak bude právě o c méně hran než vrcholů. □

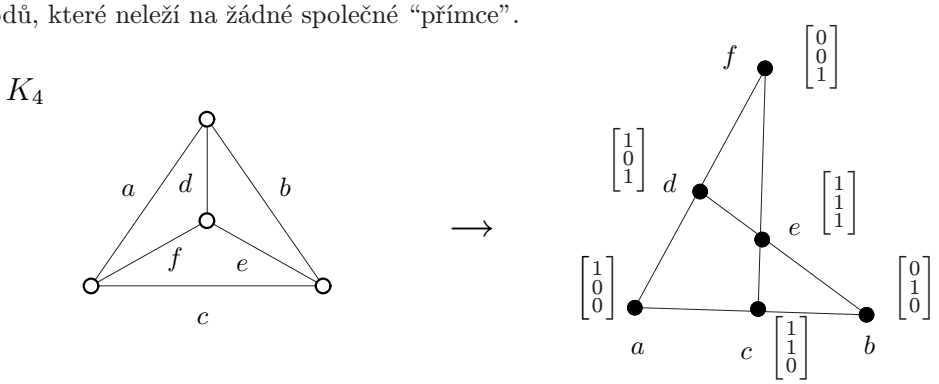
`Definice`: Řekneme, že podmnožina hran $F \subseteq E(G)$ je *acyklická*, pokud podgraf s vrcholy $V(G)$ a hranami z F nemá kružnici.

Lema 1.9. *Nechť F_1, F_2 jsou acyklické podmnožiny hran grafu G a $|F_1| < |F_2|$. Pak existuje hrana $f \in F_2 \setminus F_1$ taková, že $F_1 \cup \{f\}$ je také acyklická podmnožina.*

`Důkaz`: Jelikož $|F_1| < |F_2|$ a platí Lema 1.8, má podgraf G_1 tvořený hranami z F_1 více komponent než podgraf G_2 tvořený hranami z F_2. Potom však některá hrana $f \in F_2$ musí spojovat dvě různé komponenty podgrafu G_1, a tudíž přidáním f do F_1 nevznikne kružnice. □

`Definice`: Podle Lematu 1.9 tvoří systém všech acyklických podmnožin hran v (libovolném) grafu G matroid. Tento matroid nazýváme *matroidem kružnic* grafu G.
V analogii s grafy dále používáme název *kružnice* pro minimální závislé množiny matroidu.

`Komentář:` Tyto dva příklady jsou hezky ilustrovány v následujícím obrázku, který ukazuje, jak hrany grafu K_4 vlevo odpovídají vektorům v matroidu vpravo. Čáry (zvané "přímky") v pravém schématu vyznačují lineární závislosti mezi vektory; tj. nezávislé jsou ty trojice bodů, které neleží na žádné společné "přímce".



Abstraktní hladový algoritmus

V praxi se matroid obvykle nezadává výčtem všech nezávislých množin, protože těch je příliš mnoho (až 2^n pro n-prvkovou množinu X). Místo toho bývá dána externí funkce pro testování nezávislosti dané podmnožiny.

Algoritmus 1.10. ***Nalezení minim. báze matroidu – hladový algoritmus.***
`vstup:` *množina X s váhovou funkcí $w : X \to \mathbb{R}$,*
matroid na X určený externí funkcí `nezavisla(Y);`

`setřídit X=(x[1],x[2],...,x[n])` *tak, aby* `w[x[1]]<=...<=w[x[n]];`
`B = ∅;`
`for (i=1; i<=n; i++)`
`  if (nezavisla(B∪{x[i]}))`
`    B = B∪{x[i]};`

`výstup:` *báze B daného matroidu s minimálním součtem ohodnocení vzhledem k w.*

`Poznámka`: Pokud X v tomto algoritmu je množina hran grafu, w váhová funkce na hranách a nezávislost znamená acyklické podmnožiny hran (matroid kružnic grafu), pak Algoritmus 1.5 je přesně instancí Algoritmu 1.10.

Věta 1.11. *Algoritmus 1.10 (hladový algoritmus) pro danou nosnou množinu X s váhovou funkcí $w : X \to \mathbb{R}$ a pro daný matroid $\mathcal{N}$ na X správně najde bázi v $\mathcal{N}$ s nejmenším součtem vah.*

`Důkaz`: Z definice matroidu je jasné, že k výsledné množině B již nelze přidat další prvek, aby zůstala nezávislá, proto je B báze. Seřaďme si prvky X podle vah jako v algoritmu $w(x[1]) \leq \ldots \leq w(x[n])$. Nechť indexy $i_1, i_2, \ldots, i_k$ určují vybranou k-prvkovou bázi B v algoritmu a nechť indexy $j_1, j_2, \ldots, j_k$ vyznačují (třeba jinou?) bázi $\{x[j_1], \ldots, x[j_k]\}$ s nejmenším možným součtem vah.

Vezměme nejmenší $r \geq 1$ takové, že $w(x[i_r]) \neq w(x[j_r])$. Potom nutně $w(x[i_r]) < w(x[j_r])$, protože náš algoritmus je "hladový" a bral by menší $w(x[j_r])$ již dříve. Na druhou stranu, pokud by druhá báze $\{x[j_1], \ldots, x[j_k]\}$ dávala menší součet vah, muselo by existovat jiné $s \geq 1$ takové, že $w(x[i_s]) > w(x[j_s])$. Nyní vezměme nezávislé podmnožiny $A_1 = \{x[i_1], \ldots, x[i_{s-1}]\}$ a $A_2 = \{x[j_1], \ldots, x[j_s]\}$, kde A_2 má o jeden prvek více než A_1 a všechny prvky A_2 mají dle předpokladu menší váhu než $w(x[i_s])$.

Podle definice matroidu existuje $y \in A_2 \setminus A_1$ takové, že $A_1 \cup \{y\}$ je nezávislá. Přitom samozřejmě $y = x[\ell]$ pro nějaké ℓ. Ale to není možné, protože, jak je výše napsáno, $w(y) < w(x[i_s])$, takže by náš hladový algoritmus musel $y = x[\ell]$, $\ell < i_s$ vzít dříve do vytvářené báze B než vzal $x[i_s]$. Proto jiná báze s menším součtem vah než nalezená B neexistuje. □

Tím jsme zároveň dokončili důkaz správnosti Algoritmu 1.5, který je jen specifickou instancí Algoritmu 1.10.

`Poznámka`: Požadavek, že hladový Algoritmus 1.10 hledá bázi s minimálním součtem vah je v zásadě jen naší konvencí. Je jasné, že obrácením znamének u hodnot w se z minimalizace stává maximalizace a naopak.
Na druhou stranu je obecně podstatný požadavek, že výsledná množina má být bází, ne jen nezávislou množinou, neboť při kladných hodnotách w by minimální nezávislou množinou byla vždy $\emptyset$. (Naopak při maximalizaci s kladnými hodnotami w vychází automaticky báze jako ta nezávislá množina s maximálním ohodnocením.)

Kdy hladový algoritmus nepracuje správně

Čtenáře asi napadne, že hladový algoritmus nemůže fungovat vždy optimálně. My jsme dokonce schopni popsat všechny struktury, na kterých hladový postup funguje univerzálně – jsou to právě matroidy.

Věta 1.12. *Nechť X je nosná množina se systémem "nezávislých" podmnožin $\mathcal{N}$ splňující podmínky (1,2) Definice 1.6. Pokud pro jakoukoliv váhovou funkci $w : X \to \mathbb{R}$ najde Algoritmus 1.10 optimální nezávislou množinu z $\mathcal{N}$, tak $\mathcal{N}$ splňuje také podmínku (3), a tudíž tvoří matroid na X.*

Důkaz: Tvrzení dokazujeme sporem. Předpokládejme, že vlastnost (3) neplatí pro dvojici nezávislých množin A, B, tj. že $|A| < |B|$, ale pro žádný prvek $y \in B \setminus A$ není $A \cup \{y\}$ nezávislá. Nechť $|A| = a, |B| = b$, kde $2b > 2a+1$. Zvolíme následující ohodnocení

- $w(x) = -2b$ pro $x \in A$,
- $w(x) = -2a - 1$ pro $x \in B \setminus A$,
- $w(x) = 0$ jinak.

Hladový algoritmus přirozeně najde bázi B_1 obsahující A a disjunktní s $B \setminus A$ podle našeho předpokladu. Její ohodnocení je $w(B_1) = -2ab$. Avšak optimální bází je v tomto případě jiná B_2 obsahující celé B a mající ohodnocení nejvýše $w(B_2) \leq (-2a - 1)b = -2ab - b < w(B_1)$. To je ve sporu s dalším předpokladem, že i při námi zvoleném ohodnocení w nalezne hladový algoritmus optimální bázi. Proto je sporný náš předpoklad o množinách A, B a podmínka (3) je splněna. □

Nakonec uvádíme několik příkladů dobře známých kombinatorických úloh, ve kterých hladový algoritmus výrazně selže:

Obarvení grafu. Jak jsem již poznamenali, v Úloze 1.1 bylo přidělování úkolů pracovníkům vlastně barvením grafu. Obecně hladově barvíme graf tak, že ve zvoleném pořadí vrcholů každému následujícímu přidělíme první volnou barvu.

Třeba v nakreslené cestě délky 3 můžeme barvit hladově v pořadí od vyznačených krajních vrcholů, a pak musíme použít 3 barvy místo optimálních dvou.

Vrcholové pokrytí. Problém vrcholového pokrytí se ptá na co nejmenší podmnožinu C vrcholů daného grafu takovou, že každá hrana má alespoň jeden konec v C. Přirozeným hladovým postupem by bylo vybírat od vrcholů nejvyšších stupňů ty, které sousedí s doposud nepokrytými hranami. Bohužel tento postup také obecně nefunguje.

Poznámka: Zmíněná selhání hladového algoritmu se obecně vážou k nevhodně zvolenému pořadí kroků. Nemysleme si však, že by se tato selhání dala nějak snadno napravit volbou jiného pořadí – platí, že nalezení optimálního pořadí kroků pro použití hladového algoritmu může být (a bývá) stejně obtížné jako vyřešení úlohy samotné.

Doplňkové otázky

(1.3.1) *Jak špatně může dopadnout hladové barvení bipartitního grafu? (Bipartitní grafy jsou ty, které lze optimálně obarvit 2 barvami.)*
Přesně se otázkou myslí, kolik barev se hladově použije pro nejhorší bipartitní graf při

nejhorším uspořádání jeho vrcholů, když se v každém kroku pro nový vrchol vybírá první volná barva.

(1.3.2) *V jakém (jednoduše spočitatelném) pořadí barvit vrcholy bipartitního grafu, aby stačily 2 barvy?*

(1.3.3) *Jak lze (dobře) využít hladový algoritmus pro obarvení grafu se všemi vrcholy stupně k pomocí $k+1$ barev?*

(1.3.4) *Kdy selže hladový postup pro vrcholové pokrytí?*

1.4 Co je optimalizační úloha

Ke konci úvodní lekce si uvedeme obecný popis toho, co rozumíme pod pojmem optimalizační úlohy. Čtenář by měl mít na paměti, že se rozhodně nejedná o přesnou a exkluzivní definici, ale spíše o hrubý popis, který se může upřesňovat podle potřeby.

Definice 1.13. ***Optimalizační úloha***
je výpočetní problém (zhruba) určený následujícími atributy zadání:

1. *Univerzem* $\mathcal{U}$ všech potenciálních řešení, jež je obvykle dáno jako vektorový prostor s jednotlivými proměnnými jako souřadnicemi.
2. *Omezujícími podmínkami*, které určují podmnožinu $\mathcal{P} \subseteq \mathcal{U}$ všech *přípustných řešení* úlohy.
3. *Účelovou funkcí* $\eta : \mathcal{U} \to \mathbb{R}$, která přiřazuje každému možnému řešení jeho hodnotu – cenu. Podle kontextu úlohy hodnotu účelové funkce buď *maximalizujeme* nebo *minimalizujeme*.

Vyřešením optimalizační úlohy pak rozumíme následující:

4. Nalezení řešení $\vec{x} \in \mathcal{P}$ s *optimální hodnotou* účelové funkce, tj. dle konextu

$$\vec{x} \in \mathcal{P} : \quad \eta(\vec{x}) = \max_{\vec{z} \in \mathcal{P}} / \min_{\vec{z} \in \mathcal{P}} \ \eta(\vec{z}) \,.$$

5. Případné podání *důkazu optimality* nalezeného řešení $\vec{x}$. Tento krok je přitom relevantní jen v případech (poměrně častých), kdy takové zdůvodnění je snažší než samotné nalezení řešení $\vec{x}$.

`Komentář:` Podívejme se na konkrétní úlohy této lekce z pohledu Definice 1.13:

V Úloze 1.1 je univerzem prostor všech přiřazení čísel pracovníků jednotlivým úkolům (tj. celočíselný vektorový prostor $\mathbb{Z}^k$, kde k je počet úkolů na vstupu). Přípustnými řešeními jsou ta přiřazení, kdy čísla pracovníků jsou kladná celá a žádné překrývající se úkoly nemají stejného pracovníka. Účelovou funkcí pak je hodnota nejvyššího použitého čísla pracovníka, tuto funkci minimalizujeme. Formálně, jsou-li zadány intervaly $I_1, I_2, \ldots, I_k$ pracovních úkolů, pak $\mathcal{U} = \mathbb{Z}^k$ a složka z_i vektoru $\vec{z}$ určuje pracovníka pro úkol I_i, $\mathcal{P} = \{z \in \mathcal{U}, z > 0 : (i \neq j \wedge I_i \cap I_j \neq \emptyset) \Rightarrow z_i \neq z_j\}$ a $\eta(\vec{z}) = \max\{z_1, \ldots, z_k\}$. Promyslete si to sami!

V Úloze 1.4 je univerzem prostor m-složkových binárních vektorů, $\mathcal{U} = \mathbb{Z}_2^m$, kde m je počet hran daného grafu. Složka z_i vektoru říká, zda i-tou hranu grafu vybíráme do kostry. Přípustná řešení jsou tvořena acyklickými podmnožinami hran a účelová funkce je $\eta(\vec{z}) = \sum_{i=1}^{m} z_i \cdot w(e_i)$, tj. součet vah vybraných hran.

`Poznámka`: V některých případech jsou optimalizační úlohy tak obtížné, že nalezneme jen jejich *přibližné řešení* – takové, které je dostatečně blízko optimálnímu. Třeba v případě maximalizace účelové funkce je přibližným řešením s chybou 10% takové $\vec{y}$, že

$$\vec{y} \in \mathcal{P} : \quad \eta(\vec{y}) \geq 0.9 \cdot \max_{\vec{z} \in \mathcal{P}} \ \eta(\vec{z}) \,,$$

kdežto v případě minimalizace účelové funkce s chybou 10% je

$$\vec{y} \in \mathcal{P}: \quad \eta(\vec{y}) \leq 1.1 \cdot \min_{\vec{z} \in \mathcal{P}} \eta(\vec{z}) \,.$$

Zároveň pak místo zdůvodňování optimality nalezeného řešení zdůvodňujeme odhad procenta chyby od optima.

Doplňkové otázky

(1.4.1) *Jak se ve smyslu Definice 1.13 zformuluje problém barvení grafu?*

(1.4.2) *Má smysl povolit neceločíselné hodnoty jako barvy grafu?*

(1.4.3) *Jak se ve smyslu Definice 1.13 zformuluje problém vrcholového pokrytí?*

1.5 Další úlohy

Úplným závěrem lekce přidáváme několik dalších, víceméně náhodně zvolených, příkladů optimalizačních úloh k procvičení látky. Zdůrazňujeme, že naším cílem není úlohy vyřešit, jen matematicky pochopit a zapsat jejich zadání jako optimalizační úlohu dle Definice 1.13.

Příklad 1.14. *Ukázka optimalizace letecké dopravy.*

Letecká společnost má z města S přepravit najednou 500 lidí do okolních čtyř měst A, B, C, D. Pro jednoduchost uvažme, že z města S je do ostatních měst stejná vzdálenost a že náklady na jeden let nezávisí na počtu pasažérů, jen na typu letadla. Společnost má k dispozici čtyři typy letadel:

Typ stroje	Kapacita	Náklady na let	Počet strojů
(1)	250	210	1
(2)	100	140	2
(3)	150	170	1
(4)	30	50	5

Přitom do města A je potřeba přepravit 200 lidí, do města B 100 lidí, do města C 70 lidí a do města D 130 lidí. Navrhněte letový plán tak, aby byly minimalizovány náklady na přepravu.

Nejprve si ujasněme, co je univerzem všech řešení, neboli, zhruba řečeno, jakou máme možnost volby: Pro každý stroj můžeme volit jednu z destinací A,B,C,D,∅, přičemž ∅ znamená, že stroj nikam nepoletí. Jinak se dá říci, že pro každou destinaci volíme multimnožinu typů letadel, které tam poletí. (Multimnožina znamená, že jeden typ letadla může být ve více exemplářích, což je náš případ.)

Z tohoto druhého pohledu vidíme třeba přípustná řešení

1. A:(1), B:(2), C:(4)(4)(4), D:(3)(4),
2. A:(1), B:(2), C:(2), D:(3).

Přípustnost řešení je dána splněním dvou podmínek: Celková kapacita letadel do každé destinace musí dostačovat pro všechny pasažéry a nesmíme použít více strojů, než které jsou k dispozici. Snadno si spočítáme, že cena prvního řešení je 690, zatímco cena řešení druhého je pouze 660 (to by mělo být zároveň i optimální řešení, ale optimalitou se zatím zabývat nebudeme).

Univerzum všech řešení lze pro počítačové zpracování zapsat pomocí matice

$$\boldsymbol{L} = \begin{array}{c|cccc} & (1) & (2) & (3) & (4) \\ \hline \text{A} & 1 & 0 & 0 & 0 \\ \text{B} & 0 & 1 & 0 & 0 \\ \text{C} & 0 & 0 & 0 & 3 \\ \text{D} & 0 & 0 & 1 & 1 \end{array},$$

jejíž prvky udávají počty letadel jednotlivých typů do jednotlivých destinací. Zapíšeme-li vektorem požadované *počty pasažérů i kapacity letadel*, první podmínka se maticově zapíše

$$\boldsymbol{L} \cdot (250, 100, 150, 30)^T \geq (200, 100, 70, 130)^T .$$

Zapíšeme-li *počty strojů* jednotlivých typů vektorem, druhá podmínka pak maticově zní

$$(1, 1, 1, 1) \cdot \boldsymbol{L} \leq (1, 2, 1, 5) .$$

Dále nesmíme zapomenout na samozřejmé podmínky, že počty použitých strojů na linkách (tj. položky $\boldsymbol{L}$) musí být *přirozená čísla*. A nakonec *účelová funkce* se zapíše jako

$$\max \left[(1, 1, 1, 1) \cdot \boldsymbol{L} \cdot (210, 140, 170, 50)^T \right] .$$

Čtenář si zajisté sám odvodí, jak tyto podmínky zapsat pro jiná konkrétní zadání úlohy. □

Příklad 1.15. *Jednotlivé přerušitelné rozvrhování úloh.*

Mějme stroj, který zpracovává po jedné zadané úlohy. Každá úloha má danou prioritu, je připravena ke zpracování v určitém čase a její dokončení trvá určitou dobu. Přitom zpracování kterékoliv úlohy lze kdykoliv beze ztrát přerušit. Jakou navrhneme optimální strategii zpracování a jak bude určena její cena?

Řešení: Toto je volněji zadaná úloha, u níž si teprve sami musíme upřesnit matematické zadání.

Vstup: Úloha J_i začne v čase t_i s délkou l_i a prioritou p_i, $i = 1, 2, \ldots, n$.
Výstup: Postup zpracování, kdy úloha J_i je ukončena v čase f_i.

Cena řešení je dána souhrnem čekání na dokončení jednotlivých úloh, váženým prioritami úloh

$$\min c = \sum_{i=1}^{n} (f_i - t_i) \cdot p_i .$$

Toto však není jediný možný pohled na ocenění našeho řešení. V jiném pohledu by nás místo celkových dob ukončení úloh mohly zajímat jen prostoje zpracování způsobeném čekáním na jiné úlohy. Pak by se cena zapsala matematicky takto

$$\min c = \sum_{i=1}^{n} (f_i - t_i - l_i) \cdot p_i .$$

Jen pro zajímavost, optimální řešení zadaného problému získáme snadno hladovým postupem, kdy v každém okamžiku zpracováváme čekající úlohu s nejvyšší prioritou. (Tj. když zrovna přijde úloha vyšší priority, stávající zpracování přerušíme a přejdeme k nové úloze.) □

Příklad 1.16. *Jednotlivé nepřerušitelné rozvrhování předem známých úloh.*

Zadání je jako v Příkladu 1.15, jen zpracování jedné úlohy nyní nelze přerušit.

Vstup: Úloha J_i začne v čase t_i s délkou l_i a prioritou p_i, $i = 1, 2, \ldots, n$. Tato data jsou předem známa.

Výstup: Postup zpracování, kdy úloha J_i je započata v čase s_i a ukončena v čase $f_i = s_i + l_i$.

Řešení: Smyslem tohoto příkladu je hlavně ukázat, jak drobná změna zadání (nemožnost přerušení úlohy) radikálně změní celý matematický optimalizační model. Již z letmého pohledu je jasné, že rozvrhování nyní bude obtížnější, neboť zařazenou úlohu již nemůžeme přerušit a ostatní úlohy (třebaže s vyšší prioritou) musí čekat na její dokončení. Zde je pěkně vidět, jak se *univerzum* všech řešení mění ze spojitého na diskrétní.

Jednoduchým *matematickým modelem postupu* zpracování úloh je permutace π množiny indexů $\{1, 2, \ldots, n\}$, která zpracovávání došlých úloh v pořadí $J_{\pi(i)}, \ldots, J_{\pi(n)}$. Permutace π navíc rekurzivně určuje čas s_i začátku zpracování každého úkolu J_i takto (za předpokladu hladové optimality):

- $s_{\pi(1)} = t_{\pi(1)}$,
- $s_{\pi(k+1)} = \max(t_{\pi(k+1)}, s_{\pi(k)} + l_{\pi(k)})$.

(Vzorce nám říkají, že každá další úloha musí počkat na čas svého začátku a také na dokončení předchozí úlohy.) *Cena řešení* pak je opět

$$\min c = \sum_{i=1}^{n} (f_i - t_i) \cdot p_i = \sum_{i=1}^{n} (s_i + l_i - t_i) \cdot p_i \,.$$

□

Doplňkové otázky

(1.5.1) *Co se stane, když v Příkladu 1.15 zadáme místo priorit jednotlivých úloh požadované termíny jejich dokončení (deadline)? Co by pak bylo vhodné optimalizovat?*

(1.5.2) *Proč se v Příkladu 1.16 zdůrazňuje, že zpracovávané úlohy jsou předem známé?*

(1.5.3) *Zamyslete se sami, jak se změní charakter výše popsaných rozvrhovacích úloh, pokud budeme v zadání kombinovat priority spolu s termíny dokončení.*

Rozšiřující studium

Pro bližší studium teorie grafů doporučujeme skripta [1] nebo výbornou moderní knihu [6]. Konkrétně hladovým algoritmům a minimální kostře se věnují [4, Oddíl 6.2] a [6, Kapitola 5]. Množství motivačních optimalizačních úloh se nachází v úvodních partiích prakticky orientované učebnice [3].

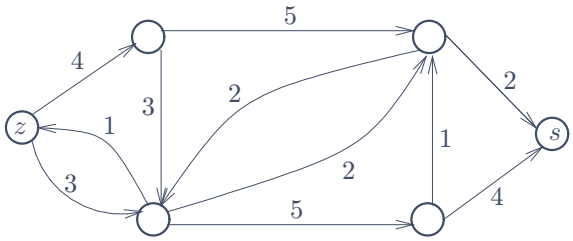
V oblasti teorie matroidů je jen poskrovnu české literatury, ale asi nejlepší anglickou monografií je [8]. Krátký čtivý úvod do matroidů je volně dostupný na [9]. My se dále optimalizaci na matroidech budeme věnovat v Lekci 13 a matroidům v optimalizaci je také věnováno [10, Chapter 10].

2 Toky v sítích

Úvod

Nyní se podíváme na jinou oblast úloh, kde našla kombinatorická optimalizace bohaté uplatnění. Jde o oblast tzv. "síťových" úloh: Pojem síť používáme jako souhrnné pojmenování pro matematické modely situací, ve kterých přepravujeme nějakou substanci (hmotnou či nehmotnou) po předem daných přepravních cestách, které navíc mají omezenou kapacitu.

Jedná se třeba o potrubní sítě přepravující vodu nebo plyn, o dopravní síť silnic s přepravou zboží, nebo třeba o internet přenášející data. Obvykle nás zajímá problém přenést z daného „zdroje“ do daného cíle čili „stoku“ co nejvíce této substance, za omezujících podmínek kapacit jednotlivých přepravních cest (případně i jejich uzlů). Obrázkem můžeme vyjádřit síť s danými kapacitami přepravy jako:



Problém maximálního toku v síti je snadno algoritmicky řešitelný, jak si také popíšeme. (Postup je dosti podobný hladovému algoritmu.) Jeho řešení má (kupodivu) i mnoho teoretických důsledků v teorii grafů, třeba pro souvislost či párování v grafech.

Cíle

Úkolem této lekce je teoreticky popsat problém toku v síti a vysvětlit základní algoritmus nenasycených cest pro jeho řešení. Dále jsou uvedeny některé důsledky vysvětlené látky (pro rozšířené sítě, pro bipartitní párování a výběr reprezentantů množin). Na příkladě toku v síti je vysvětlen důležitý princip tzv. dobré charakterizace úloh.

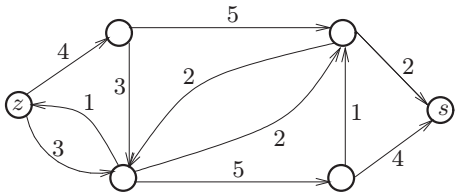
2.1 Definice sítě

Základní strukturou pro reprezentaci sítí je orientovaný graf [1, Lekce 6]. Vrcholy grafu modelují jednotlivé uzly sítě a hrany jejich spojnice. V orientovaných grafech je každá hrana tvořena uspořádanou dvojicí (u, v) vrcholů grafu, a tudíž taková hrana má *směr z vrcholu u do v*.

Definice 2.1. ***Síť*** je čtveřice $S = (G, z, s, w)$, kde

- G je orientovaný graf,
- vrcholy $z \in V(G)$, $s \in V(G)$ jsou *zdroj* a *stok*,
- $w : E(G) \to \mathbb{R}^+$ je kladné ohodnocení hran, zvané *kapacita hran*.

`Komentář:`



Na obrázku je zakreslena síť s vyznačeným zdrojem z a stokem s, jejíž kapacity hran jsou zapsány čísly u hran. Šipky udávají směr hran, tedy směr proudění uvažované substance po spojnicích. (Pokud směr proudění není důležitý, vedeme mezi vrcholy dvojici opačně orientovaných hran se stejnou kapacitou.) Kapacity hran pak omezují maximální množství přenášené substance.

`Poznámka`: V praxi může být zdrojů a stoků více, ale v definici stačí pouze jeden zdroj a stok, z něhož / do nějž vedou hrany do ostatních zdrojů / stoků. (Dokonce pak různé zdroje a stoky mohou mít své kapacity.)

Obvykle nás na síti nejvíce zajímá, kolik nejvíce substance můžeme (různými cestami) přenést ze zdroje do stoku. Pro to musíme definovat pojem toku, což je formální popis okamžitého stavu přenášení v síti.

`Značení`: Pro jednoduchost píšeme ve výrazech znak $e \to v$ pro hranu e přicházející do vrcholu v a $e \leftarrow v$ pro hranu e vycházející z v.

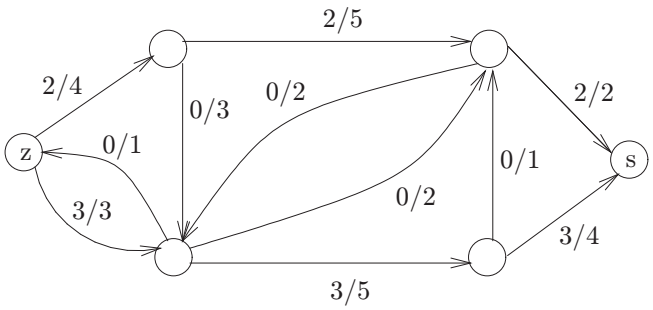
Definice 2.2. ***Tok*** v síti $S = (G, z, s, w)$
je funkce $f : E(G) \to \mathbb{R}_0^+$ splňující

- $\forall e \in E(G) : \ 0 \le f(e) \le w(e)$,
- $\forall v \in V(G), v \neq z, s : \ \sum\limits_{e \to v} f(e) = \sum\limits_{e \leftarrow v} f(e)$.

Velikost toku f je dána výrazem $\|f\| = \sum\limits_{e \leftarrow z} f(e) - \sum\limits_{e \to z} f(e)$.

`Značení`: Tok a kapacitu hran v obrázku sítě budeme zjednodušeně zapisovat ve formátu F/C, kde F je hodnota toku na hraně a C je její kapacita.

`Komentář:` Neformálně tok znamená, kolik substance je každou hranou zrovna přenášeno (ve směru této hrany, proto hrany musí být orientované). Tok je pochopitelně nezáporný a dosahuje nejvýše dané kapacity hrany.



Ve vyobrazeném příkladě vede ze zdroje vlevo do stoku vpravo tok o celkové velikosti 5.

`Poznámka`: Obdobně se dá velikost toku definovat u stoku, neboť

$$0 = \sum_e (f(e) - f(e)) = \sum_v \sum_{e \leftarrow v} f(e) - \sum_v \sum_{e \to v} f(e) = \sum_{v=z,s} \left(\sum_{e \leftarrow v} f(e) - \sum_{e \to v} f(e) \right) .$$

(Dvojité sumy uprostřed předchozího vztahu nabývají stejných hodnot pro všechny vrcholy kromě z a s dle definice toku.) Proto velikost toku počítaná u zdroje je rovna opačné velikosti toku počítaného u stoku

$$\left(\sum_{e \leftarrow z} f(e) - \sum_{e \to z} f(e) \right) = - \left(\sum_{e \leftarrow s} f(e) - \sum_{e \to s} f(e) \right) .$$

Doplňkové otázky

(2.1.1) *Dal by se nějak rozumně problém toku v síti přeformulovat tak, aby formálně součet toků u každého vrcholu (včetně z, s) byl 0?*

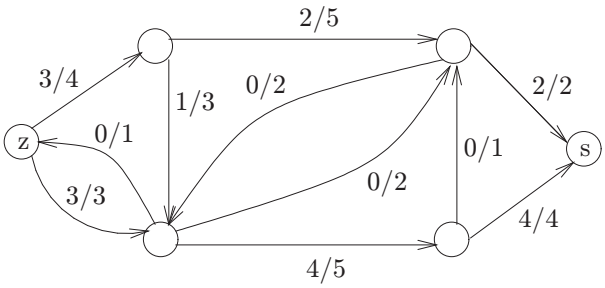
(2.1.2) *Nechť U je množina vrcholů sítě obsahující zdroj a neobsahující stok. Rozmyslete si, proč velikost toku mezi zdrojem a stokem lze spočítat také z rozdílu součtů toků na všech hranách vycházejících z U a hranách přicházejících do U.*

2.2 Hledání maximálního toku

Naším úkolem je najít co největší tok v dané síti. Pro jeho nalezení existují jednoduché a velmi rychlé algoritmy.

Úloha 2.3. ***O maximálním toku v síti***
Je dána síť $S = (G, z, s, w)$ a našim úkolem je pro ni najít co největší tok ze zdroje z do stoku s vzhledem k ohodnocení w.
Formálně hledáme $\max \|f\|$ *dle Definice 2.2.*

`Komentář:` Tok velikosti 5 uvedený v ukázce v předchozí části nebyl optimální, neboť v té síti najdeme i tok velikosti 6:

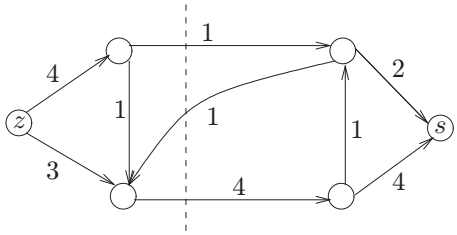


Jak však poznáme, že větší tok již v dané síti neexistuje? V této konkrétní ukázce to není obtížné, vidíme totiž, že obě dvě hrany přicházející do stoku mají součet kapacit $2 + 4 + 6$, takže více než 6 do stoku ani přitéct nemůže. V obecnosti lze použít obdobnou úvahu, kdy najdeme podmnožinu hran, které nelze tokem "obejít" a které v součtu kapacit dají velikost našeho toku. Existuje však taková množina hran vždy? Odpověď nám dá následující definice a věta.

Definice 2.4. ***Řez*** v síti $S = (G, z, s, w)$
je podmnožina hran $C \subseteq E(G)$ taková, že v podgrafu $G - C$ (tj. po odebrání hran C z G) nezbude žádná orientovaná cesta ze z do s.
Velikostí řezu C rozumíme součet kapacit hran z C, tj. $\|C\| = \sum_{e \in C} w(e)$.

Věta 2.5. *Maximální velikost toku v síti je rovna minimální velikosti řezu.*

`Komentář:` Na následujícím obrázku vidíme trochu jinou síť s ukázkou netriviálního minimálního řezu velikosti 5, naznačeného svislou čárkovanou čarou. Všimněte si dobře, že definice řezu mluví o přerušení *všech orientovaných cest ze z do s*, takže do řezu stačí započítat hrany jdoucí přes svislou čáru od z do s, ale ne hranu jdoucí zpět. Proto je velikost vyznačeného řezu $1 + 4 = 5$.



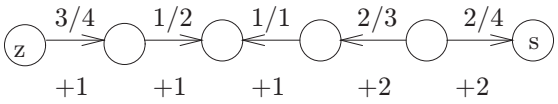
`Poznámka`: Tato věta poskytuje tzv. *dobrou charakterizaci* problému maximálního toku: Když už nalezneme maximální tok, tak je pro nás vždy snadné dokázat, že lepší tok není, nalezením příslušného řezu o stejné velikosti. Přitom toto zdůvodnění řezem můžeme směle ukázat i někomu, kdo se vůbec nevyzná v matematice.

Důkaz Věty 2.5 bude proveden následujícím algoritmem.

`Definice`: Mějme síť S a v ní tok f. *Nenasycená cesta* (v S vzhledem k f) je neorientovaná cesta v G z vrcholu u do vrcholu v (obvykle ze z do s), tj. posloupnost navazujících hran $e_1, e_2, \ldots, e_m$, kde $f(e_i) < w(e_i)$ pro e_i ve směru z u do v a $f(e_i) > 0$ pro e_i v opačném směru.

Hodnotě $w(e_i) - f(e_i)$ pro hrany e_i ve směru z u do v a hodnotě $f(e_i)$ pro hrany e_i v opačném směru říkáme *rezerva kapacity* hran e_i. Nenasycená cesta je tudíž cesta s kladnými rezervami kapacit všech hran.

`Komentář:` Zde vidíme příklad nenasycené cesty ze zdroje do stoku s minimální rezervou kapacity +1.

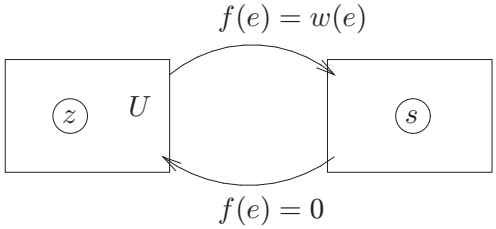


Všimněte si dobře, že cesta není orientovaná, takže hrany na ní jsou v obou směrech.

Algoritmus 2.6. ***Ford–Fulkersonův pro tok v síti***
`vstup` *síť* $S = (G, z, s, w)$`;`
tok $f \equiv 0$`;`
`do` {
prohledáváním grafu najdeme množinu U *vrcholů* G,
do kterých se dostaneme ze z *po nenasycených cestách*`;`
`if` ($s \in U$) {
`P =` *(výše nalezená) nenasycená cesta v* S *ze* z *do* s`;`
zvětšíme tok f *o minimální rezervu kapacity hran v* P`;`
}
`while` ($s \in U$)`;`
`výstup` *vypíšeme maximální tok* f`;`
`výstup` *vypíšeme min. řez jako množinu hran vedoucích z* U *do* $V(G) - U$.

`Důkaz` správnosti Algoritmu 2.6:
Pro každý tok f a každý řez C v síti S platí $\|f\| \leq \|C\|$. Jestliže po zastavení algoritmu s tokem f nalezneme v síti S řez o stejné velikosti $\|C\| = \|f\|$, je jasné, že jsme našli maximální možný tok v síti S. Zároveň tím dokážeme i platnost Věty 2.5.

Takže stačí dokázat, že po zastavení algoritmu nastane rovnost $\|f\| = \|C\|$, kde C je vypsaný řez mezi U a zbytkem grafu G. Vezměme tok f v S bez nenasycené cesty ze z do s. Pak množina U z algoritmu neobsahuje s. Schematicky vypadá situace takto:



Jelikož z U žádné nenasycené cesty dále nevedou, má každá hrana $e \leftarrow U$ (odcházející z U) plný tok $f(e) = w(e)$ a každá hrana $e \rightarrow U$ (přicházející do U) tok $f(e) = 0$. Velikost toku f ze z do s se také dá psát jako

$$\|f\| = \sum_{e \leftarrow U} f(e) - \sum_{e \rightarrow U} f(e) = \sum_{e \leftarrow U} f(e) = \sum_{e \in C} w(e) = \|C\| \ .$$

To je přesně, co jsme chtěli dokázat o výsledném toku. □

Z popisu Algoritmu 2.6 vyplývá ještě jeden důležitý důsledek:

Důsledek 2.7. *Pokud jsou kapacity hran sítě S celočíselné, optimální tok také vyjde celočíselně.*

`Poznámka`: Algoritmus pro celá čísla kapacit vždy skončí. Pro reálná čísla se ale dají najít extrémní případy, které nepovedou k řešení ani v limitě.

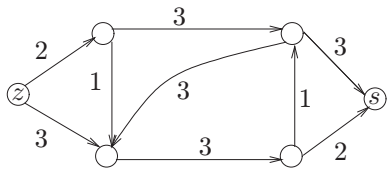
Pro rychlý běh algoritmu je vhodné hledat nejkratší nenasycenou cestu, tj. prohledáváním sítě do šířky. V takové implementaci algoritmus dobře a rychle funguje i s reálnými kapacitami hran. Viz [4].

Doplňkové otázky

(2.2.1) *Co by se stalo s úlohou o maximálním toku, pokud by graf sítě obsahoval násobné hrany?*

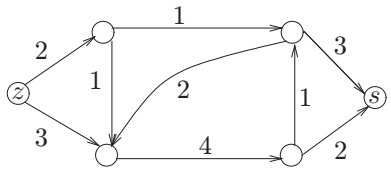
(2.2.2) *Co by se stalo s úlohou o maximálním toku a s Algoritmem 2.6, pokud bychom povolili i záporné kapacity hran?*

(2.2.3) *Jaký je maximální tok touto sítí ze z do s?*



(2.2.4) *Co obsahuje výsledná množina U Algoritmu 2.6 v předchozím příkladě?*

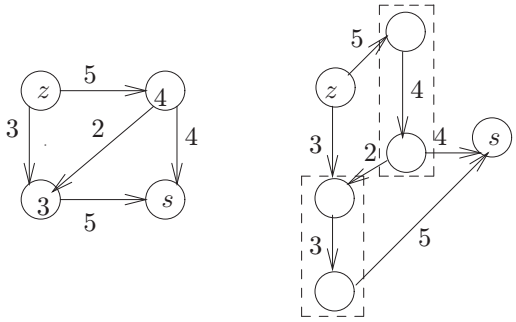
(2.2.5) *Kde je minimální řez v této síti mezi z a s?*



2.3 Zobecnění sítí a další aplikace

Pojmy sítě a toků v ní lze zobecnit v několika směrech. My si zde stručně uvedeme tři možnosti:

1. U sítě můžeme zadat i *kapacity vrcholů*.
   To znamená, že žádným vrcholem nemůže celkem protéct více než povolené množství substance. Takovou síť "zdvojením" vrcholů snadno převedeme na běžnou síť, ve které kapacity původních vrcholů budou uvedeny u nových hran spojujících zdvojené vrcholy. Viz neformální schéma:



2. Pro hrany sítě lze zadat také *minimální kapacity*, tedy dolní meze toku. (Například u potrubní sítě mohou minimální vyžadované průtoky vody garantovat, že nedojde k zanesení potrubí.) V této modifikaci úlohy již přípustný tok nemusí vůbec existovat. Takto zobecněná úloha je také snadno řešitelná, ale my se jí nebudeme zabývat.

3. V síti lze najednou přepravovat více substancí. To vede na problém tzv. *vícekomoditních toků* v síti. Tento problém je složitější a už není v obecnosti snadno řešitelný, a proto se jím nebudeme zabývat.

Kromě uvedených (a podobných) zobecnění toků v sítích jsou velmi zajímavé i některé speciální formulace problému toků, které se vyskytují v možná i nečekaných oblastech. Více o tom napíšeme v dalších částech tohoto oddílu.

Bipartitní párování

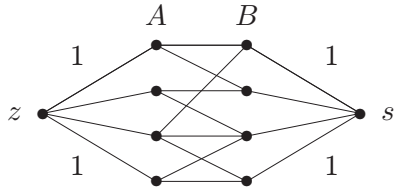
Biparitní grafy definujeme jako ty, které lze korektně obarvit dvěma barvami [1, Lekce 10]. Jinými slovy to jsou grafy, jejichž vrcholy lze rozdělit do dvou množin tak, že všechny hrany vedou mezi těmito množinami.

`Definice`: *Párování* v (biparitním) grafu G je podmnožina hran $M \subseteq E(G)$ taková, že žádné dvě hrany z M nesdílejí koncový vrchol.

> `Komentář:` Pojem (bipartitního) párování má přirozenou motivaci v mezilidských vztazích. Pokud neuvažujeme bigamii ani jisté menšiny, můžeme si partnerské vztahy představit jako párování v bipartitním grafu. Jednu stranu grafu tvoří muži a druhou ženy. Hrana mezi mužem a ženou znamená vzájemné sympatie (přitom jedinec může mít vzájemné sympatie s několika jinými opačného pohlaví). Pak skutečné partnerské vztahy představují párování v popsaném grafu.

Úlohu nalézt v daném bipartitním grafu co největší párování lze poměrně snadno vyřešit pomocí toků ve vhodně definované síti. Uvedená metoda použití toků v síti na řešení problému párování přitom hezky ilustruje obecný přístup, jakým toky v sítích pomohou řešit i úlohy, které na první pohled se sítěmi nemají nic společného.

Algoritmus 2.8. ***Nalezení bipartitního párování***
Pro daný bipartitní graf G s vrcholy rozdělenými do množin A, B sestrojíme síť S následovně:



Všechny hrany sítě S orientujeme od zdroje do stoku a přiřadíme jim kapacity 1. Nyní najdeme (celočíselný) maximální tok v S Algoritmem 2.6. Do párování vložíme ty hrany grafu G, které mají nenulový tok.

`Důkaz` správnosti Algoritmu 2.8: Podle Důsledku 2.7 bude maximální tok celočíselný, a proto každou hranou poteče buď 0 nebo 1. Jelikož však do každého vrcholu v A může ze zdroje přitéct jen tok 1, bude z každého vrcholu A vybrána do párování nejvýše jedna hrana. Stejně tak odvodíme, že z každého vrcholu B bude vybrána nejvýše jedna hrana, a proto vybraná množina skutečně bude párováním. Zároveň to bude největší možné párování, protože z každého párování lze naopak vytvořit tok příslušné velikosti a větší než nalezený tok v S neexistuje. □

`Poznámka`: Popsaná metoda je základem tzv. Maďarského algoritmu pro párování v bipartitních grafech. Úlohu nalezení maximálního párování lze definovat i pro obecné grafy a také ji efektivně algoritmicky vyřešit (viz Edmonds), ale příslušný algoritmus není jednoduchý.

Vyšší grafová souvislost

Představme si, že na libovolném grafu G definujeme zobecněnou síť tak, že kapacity všech hran a všech vrcholů položíme rovny 1 v obou směrech. Pak celočíselný tok (viz Důsledek 2.7) velikosti k mezi dvěma vrcholy u, v se skládá ze soustavy k disjunktních cest (mimo společné koncové vrcholy u, v). Naopak řez odděluje u a v do různých souvislých komponent zbylého grafu. Aplikace Věty 2.5 na tuto situaci přímo poskytne následující tvrzení.

Lema 2.9. *Nechť u, v jsou dva vrcholy grafu G a $k > 0$ je přirozené číslo. Pak mezi vrcholy u a v existuje v G aspoň k disjunktních cest, právě když po odebrání libovolných $k-1$ vrcholů různých od u, v z G zůstanou u a v ve stejné komponentě souvislosti zbylého grafu.*

Použitím tohoto tvrzení pro všechny dvojice vrcholů grafu snadno dokážeme důležitou Mengerovu větu:

Věta 2.10. *Graf G je vrcholově k-souvislý právě když mezi libovolnými dvěma vrcholy lze vést aspoň k disjunktních cest (různých až na ty dva spojované vrcholy).*

Ještě jiné použití si ukážeme na problému výběru reprezentantů množin.

Různí reprezentanti

`Definice`: Nechť $M_1, M_2, \ldots, M_k$ jsou neprázdné množiny. *Systémem různých reprezentantů* množin $M_1, M_2, \ldots, M_k$ nazýváme takovou posloupnost různých prvků $(x_1, x_2, \ldots, x_k)$, že $x_i \in M_i$ pro $i = 1, 2, \ldots, k$.

Důležitým a dobře známým výsledkem v této oblasti je Hallova věta plně popisující, kdy lze systém různých reprezentantů daných množin nalézt.

Věta 2.11. *Nechť $M_1, M_2, \ldots, M_k$ jsou neprázdné množiny. Pro tyto množiny existuje systém různých reprezentantů, právě když platí*

$$\forall J \subseteq \{1, 2, \ldots, k\} : \left|\bigcup_{j \in J} M_j\right| \geq |J|\,,$$

neboli pokud sjednocení libovolné skupiny z těchto množin má alespoň tolik prvků, kolik množin je sjednoceno.

`Důkaz`: Označme $x_1, x_2, \ldots, x_m$ po řadě všechny prvky ve sjednocení $M_1 \cup M_2 \cup \ldots \cup M_k$. Definujeme si bipartitní graf G na množině vrcholů $\{1, 2, \ldots, k\} \cup \{x_1, x_2, \ldots, x_m\} \cup \{u, v\}$, ve kterém jsou hrany $\{u, i\}$ pro $i = 1, 2, \ldots, k$, hrany $\{v, x_j\}$ pro $j = 1, 2, \ldots, m$ a hrany $\{i, x_j\}$ pro všechny dvojice i, j, pro které $x_j \in M_i$.

`Komentář`: Konstrukce našeho grafu G je obdobná konstrukci sítě v Algoritmu 2.8: Vrcholy u a v odpovídají zdroji a stoku, ostatní hrany přicházející do vrcholu x_j znázorňují všechny z daných množin, které obsahují prvek x_j.

Cesta mezi u a v má tvar u, i, x_j, v, a tudíž ukazuje na reprezentanta $x_j \in M_i$. Systém různých reprezentantů tak odpovídá k disjunktním cestám mezi u a v. Nechť X je nyní libovolná minimální množina vrcholů v G, po jejímž odebrání z grafu nezbude žádná

cesta mezi u a v. Podle Lematu 2.9 a této úvahy mají naše množiny systém různých reprezentantů, právě když každá taková oddělující množina X má aspoň k prvků.

Položme $J = \{1, 2, \ldots, k\} \setminus X$. Pak každá hrana z J (mimo u) vede do vrcholů z $X \cap \{x_1, \ldots, x_m\}$ (aby nevznikla cesta mezi u, v), a proto

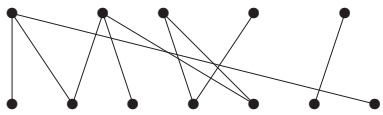
$$\left|\bigcup_{j\in J} M_j\right| = |X \cap \{x_1, \ldots, x_m\}| = |X| - |X \cap \{1, \ldots, k\}| = |X| - k + |J|\,.$$

Vidíme tedy, že $|X| \geq k$ pro všechny (minimální) volby oddělující X, právě když $\left|\bigcup_{j\in J} M_j\right| \geq |J|$ pro všechny volby J, což je dokazovaná podmínka naší věty. □

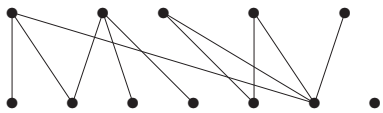
`Poznámka`: Předchozí důkaz nám také dává návod, jak systém různých reprezentantů pro dané množiny nalézt – stačí použít Algoritmus 2.6 na vhodně odvozenou síť.

Úlohy k řešení

(2.3.1) *Najděte maximální párování v tomto bipartitním grafu:*



(2.3.2) *Najděte maximální párování v tomto bipartitním grafu:*



(2.3.3) *Mají všechny tříprvkové podmnožiny množiny* $\{1, 2, 3, 4\}$ *systém různých reprezentantů?*

(2.3.4) *Mají všechny tříprvkové podmnožiny množiny* $\{1, 2, 3, 4, 5\}$ *systém různých reprezentantů?*

2.4 Dobrá charakterizace

V návaznosti na Definici 1.13, bod 5, se podíváme na tuto situaci: Dokazujeme optimalitu nalezeného řešení, tj. že lepší řešení našeho problému neexistuje, tím, že (názorně) ukážeme nějakou vhodnou "vylučovací" vlastnost. Tzn. něco, co jasně vylučuje existenci lepšího řešení způsobem pochopitelným i pro laika neobeznámeného hlouběji s problémem a naším řešením. (Poznamenáváme, že ne vždy je něco takového možné.)

> `Komentář:` Vzpomeňme si na Úlohu 1.1 o přidělování pracovních úkolů. Jak jsme u něj uměli zdůvodnit optimálnost nalezeného řešení? Velmi snadno, že? Stačilo najít okamžik, kdy všichni přiřazení pracovníci pracovali najednou. Stejně tak snadné zdůvodnění optimality řešení jsme dokázali podat v Úloze 2.3 o maximální kostře – stačilo ukázat minimální řez, kerý byl tokem zcela nasycený.

Přesněji řečeno, v Úloze 1.1 o přidělování pracovních úkolů platí, že existence přípustného přidělení k pracovníků na dané úkoly je ekvivalentní neexistenci okamžiku s více než k překrývajícími se úkoly. V přirozenějším jazyce totéž řekneme takto: k pracovníků na úkoly stačí, právě když nikdy není více než k současných úkolů.

`Fakt`: Nechť I označuje průnikový graf intervalů daných pracovních úkolů. Pak přípustného přidělení k pracovníků na tyto úkoly je modelováno jako grafový homomorfismus $p : I \to K_k$. (Vrcholy cílového úplného grafu K_k odpovídají pracovníkům, hrany ukazují možnost souběžné práce libovolných dvou z pracovníků a nepřítomnost smyček v cílovém grafu K_k určuje, že jeden pracovník nemůže vykonávat překrývající se úkoly.)

Naopak ℓ překrývajících se úkolů je v grafu I ukázáno jako grafový homomorfismus $q : K_\ell \to I$. Takže ve shrnutí můžeme naši úvahu formálně zapsat

$$\exists\, p : I \to K_k \quad \Longleftrightarrow \quad \neg\exists\, q : K_{k+1} \to I\,.$$

V případě toků v sítích platí, že tok velikosti t existuje, právě když není v síti žádný řez menší než t. Naopak pro mnohé jiné (i úspěšně řešené) optimalizační úlohy takovýto snadný a názorný důkaz optimality není znám. V obecnosti můžeme podat následující hrubý popis, kterému říkáme *princip dobré charakterizace*:

Konvence 2.12. Říkáme, že optimalizační problém má *dobrou charakterizaci*, pokud optimalitu nalezeného řešení můžeme vždy prokázat nalezením řešení jiné ("duální") úlohy, jehož ověření je "výrazně snažší" (či názornější) než bylo samotné vyřešení úlohy.

`Poznámka`: V obecnosti se princip dobré charakterizace neomezuje jen na optimalizační úlohy (kde je výrazně spojen s pracemi Edmondse), ale je to důležitý tzv. kategoriální pojem v matematice: Existence jednoho "morfismu" je dána neexistencí jiného "morfismu". Například v kombinatorice najdeme mnoho důležitých příkladů takových strukturálních charakterizací. To je však daleko za obsahem našeho předmětu.

Doplňkové otázky

*(2.4.1) *Jak byste popsali dobrou charakterizaci maximálního toku v síti pomocí minimálního řezu jako dvojici "morfismů" mezi strukturami?*

Rozšiřující studium

Problematika toků v sítích je běžnou základní součástí jak teorie grafů (i když není pokryta v učebnici [6]), tak i kombinatorické optimalizace. Podrobný popis algoritmu pro toky v sítích včetně jeho rozumně rychlé implementace najdeme třeba v [4, Oddíl 6.3,6.4]. Jinak jsou širšímu pohledu na síťové toky v kombinatorické optimalizaci věnovány [7, Chapter I.3] a [10, Chapter 4]. Posledně jmenovaný odkaz také vysvětluje blíže vztah toků s grafovou souvislostí a Mengerovou větou.

Část II
Lineární Optimalizace a Mnohostěny

3 Úloha lineární optimalizace

Úvod

Od této lekce se začneme zabývat jistou obsáhlou a dobře prozkoumanou třídou optimalizačních úloh zvanou úlohy lineární optimalizace, neboli lineární programování LP. Typickým znakem takových úloh je spojitost a "konvexita" jejich řešení.

Jak sám název naznačuje, úloha lineární optimalizace LP se skládá z lineárního (vektorového) univerza, omezujících podmínek vyjádřených jako lineární rovnosti a nerovnosti a z lineární účelové funkce. Úkolem pak je najít řešení splňující všechna omezení a maximalizující či minimalizující tuto účelovou funkci.

Nejprve si projdeme různé možné formy matematického zápisu úloh LP a jejich maticové formalizace. Ukážeme si, jak lze mezi různými zápisy LP snadno přecházet. (Problematika samotného řešení těchto úloh je prozatím odsunuta do pozadí.) Výklad je názorně doplněn množstvím praktických příkladů.

Cíle

Úkolem této lekce je přiblížit čtenáři na příkladech, jak se matematicky formulují úlohy lineární optimalizace a jak se převádí mezi různými způsoby zápisu. Čtenář by měl získat hlubší porozumění matematickému zápisu slovních úloh LP.

3.1 Příklad formulace úlohy

Pro názornost začneme rovnou rozebráním zadání konkrétního jednoduchého slovního příkladu. (Poznamenáváme, že mnohé naše ukázkové příklady byly inspirovány příklady z knihy [3]...)

Příklad 3.1. *Firma hodlá prodávat lupínky za* 120*Kč/kg a hranolky za* 76*Kč/kg. Na výrobu* 1*kg lupínků se spotřebují* 2*kg brambor a* 0.4*kg oleje. Na výrobu* 1*kg hranolek je zapotřebí* 1.5*kg brambor a* 0.2*kg oleje. Firma má nakoupeno* 100*kg brambor a* 16*kg oleje. Brambory stály* 12*Kč/kg a olej* 40*Kč/kg. Nalezněte takový plán výroby, při kterém firma nejvíce vydělá.*

Řešení: Nechť vyrobíme ℓ kg lupínků a h kg hranolků. Dané podmínky jsou

- máme k dispozici 100kg brambor
- a 16kg oleje,
- navíc lze (pochopitelně) vyrobit jen nezáporné množství od každého výrobku.

Tedy v matematickém zápise dle výše uvedených bodů

$$\begin{aligned} 2\ell + 1.5h &\leq 100 \\ 0.4\ell + 0.2h &\leq 16 \\ \ell, h &\geq 0\,. \end{aligned}$$

Tyto nerovnice nám určují množinu všech přípustných řešení úlohy ve vektorovém prostoru se dvěma souřadnicemi ℓ, h. Podívejte se na Obrázek 3.1.

Našim cílem je maximalizace zisku, na to však mohou být dva různé (i když podobné) pohledy:

- Můžeme být v situaci, kdy nakoupené suroviny již nelze jinak využít a jejich zbytky se vyhodí. V tom případě nás zajímá jen hrubý zisk z tržeb, tedy optimalizujeme funkci

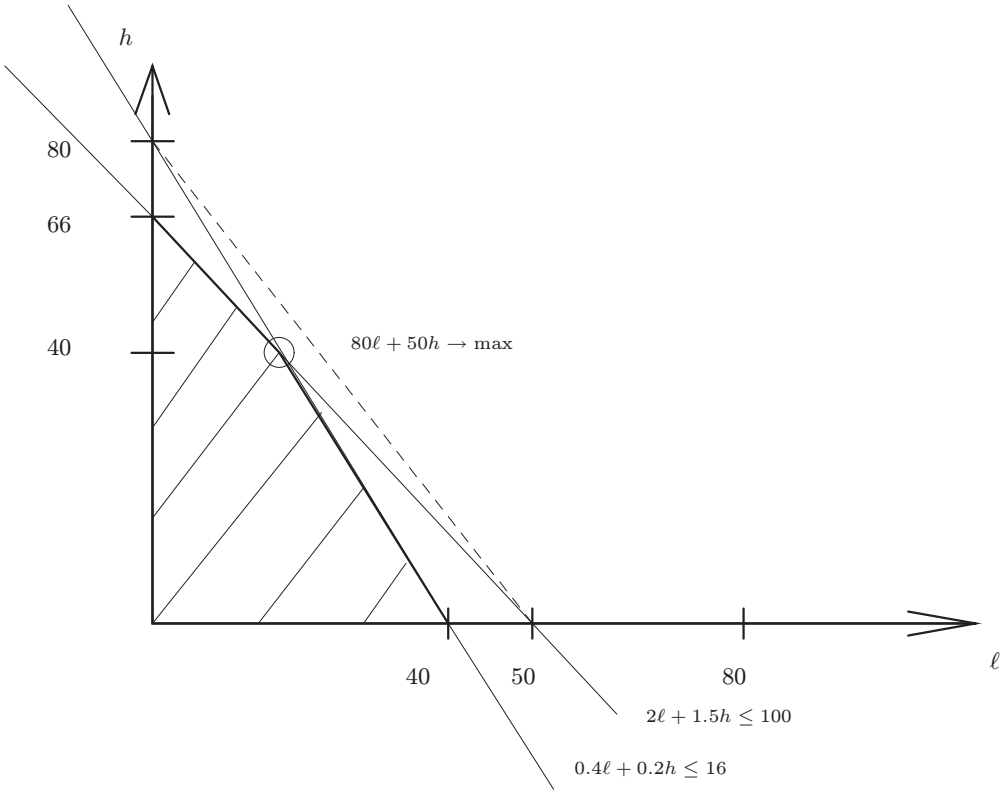
$$\max\ 120\ell + 76h\,.$$

(Náklady surovin jsou fixní a lze je nakonec od tržeb odečíst, nezávisle na řešení úlohy.)

- Můžeme také uvažovat situaci, kdy se zbytky z výroby dále využijí (na další výrobu či se vrátí dodavateli). Potom však musíme náklady surovin započítat již do účelové funkce, neboli počítat čistý zisk po odečtení nákladů, takže vyjde

$$\max\ (120 - 24 - 16)\ell + (76 - 18 - 8)h = 80\ell + 50h\,.$$

Optimálním řešením je v tomto příkladě, při obou formulacích účelové funkce, vyrobit 20kg lupínků a 40kg hranolků. Tržba firmy v první formulaci je 5440Kč, zisk z výroby v druhé formulaci je 3600Kč. Jak ale k těmto výsledkům dospějeme?



Obrázek 3.1: Grafický význam předchozí úlohy LP (Příklad 3.1): Množina přípustných řešení je šrafovaná, optimum je vyznačeno kroužkem.

Grafické vyjádření řešení: V jednoduchém případě se dvěma proměnnýma nám jednotlivé nerovnice určují poloroviny v běžné Euklidovské rovině, jejichž průnikem je množina všech *přípustných řešení*. Účelová funkce je vyjádřená systémem rovnoběžek odpovídajících jednotlivým hodnotám přípustných řešení. Optimálním řešením úlohy pak je ten bod množiny přípustných řešení, který je protnutý *rovnoběžkou účelové funkce s co nejvyšší / nejnižší hodnotou*. Názorně viz Obrázek 3.1... □

`Poznámka`: Ještě poznamenáme, že se často při matematickém přepisu slovní úlohy stává, že některou (často implicitní) podmínku zapomeneme zapsat nerovnicí. Například bychom snadno

mohli zapomenout na podmínku nezápornosti vyrobených množství hranolků a lupínků a nemuseli si toho všimnout, neboť nezápornost se neprojeví v optimálním řešení. Při jiných účelových funkcích to však může hrát roli, a pak bychom dostali například výsledek jako "vyrobit 50kg lupínků a −10kg hranolků", který je nesmyslný.
(Dobrou praxí tedy je se hluboce zamyslet nad významem dosaženého řešení úlohy, a pak doplnit případně zapomenuté další podmínky.)

Doplňkové otázky

(3.1.1) *Jaký faktický význam má výsledek "vyrobit −10kg hranolků"?*

(3.1.2) *Odhadněte, podle Obrázku 3.1, pro jakou minimální prodejní cenu hranolků se vůbec vyplatí vyrábět hranolky v Příkladu 3.1?*

(3.1.3) *Jak bychom řešení předchozí otázky vyjádřili formulací (jiného) LP?*

3.2 Složitější formulace

Pro zjednodušení budeme používat zkratku LP pro třídu úloh lineární optimalizace. Dále budeme pokračovat několika složitějšíma ukázkama, kde se již musíme více zamyslet nad formulací omezujících podmínek i účelové funkce. Zopakujte si Definici 1.13.

Začínáme typickou ukázkou tzv. dopravní úlohy.

Příklad 3.2. *Spotřebitelé požadují* 7, 8, 10 *a* 11 *tun cementu. Sklady cementu jsou tři, s kapacitami po řadě* 10, 15 *a* 11 *tun. Dopravní náklady mezi sklady (řádky) a spotřebiteli (sloupce) jsou dané tabulkou*

	sp 1	sp 2	sp 3	sp 4
sklad 1	4	5	5	3
sklad 2	6	6	7	8
sklad 3	5	7	7	5

na tunu nákladu. Minimalizujte dopravní náklady mezi spotřebiteli a sklady.

Řešení: Podmínky jsou:

- Dodání cementu pro i-tého spotřebitele z j-tého skladu

$$\begin{aligned} x_{11} + x_{12} + x_{13} &= 7, \\ x_{21} + x_{22} + x_{23} &= 8, \\ x_{31} + x_{32} + x_{33} &= 10, \\ x_{41} + x_{42} + x_{43} &= 11. \end{aligned}$$

- Dodržení kapacit skladů

$$\begin{aligned} x_{11} + x_{21} + x_{31} + x_{41} &\leq 10, \\ x_{12} + x_{22} + x_{32} + x_{42} &\leq 15, \\ x_{13} + x_{23} + x_{33} + x_{43} &\leq 11. \end{aligned}$$

- Podmínky nezápornosti

$$x_{ij} \geq 0 \qquad \text{pro } i = 1, 2, 3, 4 \; j = 1, 2, 3.$$

`Poznámka`: Podmínky nezápornosti nyní nejsou zcela nutné. Převoz záporného množství cementu ze skladu spotřebiteli má reálný význam – spotřebitel dostal z jiných skladů více cementu, než potřebuje, tak zbylé množství vrací do skladu. Hodnocení účelové funkce nám však zaručí, že takové zbytečné převozy tam a zpět se neuskuteční v optimálním řešení.

Cílem je minimalizace přepravních nákladů, účelová funkce tedy vypadá takto:

$$\min\ 4x_{11} + 5x_{21} + 5x_{31} + 3x_{41} + 6x_{12} + 6x_{22} + 7x_{32} + 8x_{42} + 5x_{13} + 7x_{23} + 7x_{33} + 5x_{43}.$$

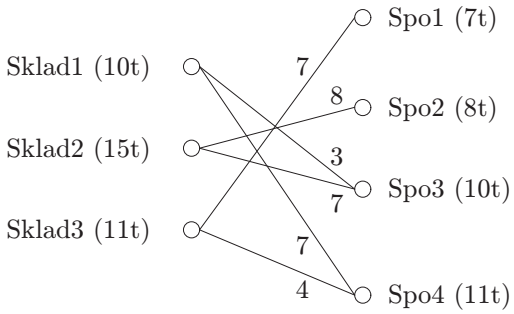
Optimální hodnota účelové funkce je

$$\eta(\vec{x}) \ = \ 188.0$$

a hodnoty jednotlivých nenulových proměnných:

$$\begin{array}{ll} x_{31} = 3.0 & x_{41} = 7.0 \\ x_{22} = 8.0 & x_{32} = 7.0 \\ x_{13} = 7.0 & x_{43} = 4.0 \end{array}$$

Optimálním řešením tedy je přepravit z prvního skladu 3t třetímu spotřebiteli a 7t čtvrtému spotřebiteli, z druhého skladu přepravit 8t materiálu druhému spotřebiteli a 7t třetímu spotřebiteli a ze třetího skladu přepravit 7t prvnímu spotřebiteli a 4t čtvrtému spotřebiteli. Náklady na přepravu činí 188 jednotek. Obrázkem: □



Přirozenou otázkou je, jak jsme k uvedenému číselnému řešení předchozího příkladu dospěli. Takové úlohy s více proměnnými již nelze řešit "pohledem na obrázek" jako Příklad 3.1. Pro řešení úloh LP však existuje poměrně jednoduchá *simplexová metoda* – viz Lekce 6, která má mnoho i volně dostupných implementací.

`Komentář:` Malé a vhodně formulované úlohy lineární optimalizace snadno vyřešíme různými aplety a prográmky volně dostupnými na internetu, například [12]. Další možností (zcela nenáročnou na stranu klienta, funguje i na PDA) je využít klientský přístup [15] k aplikaci WLPS řešící LP a IP úlohy. Viz příklady v Oddíle 3.4.

`Poznámka`: Čtenáře může napadnout, v jakém vztahu jsou celková kapacita skladů a celkové požadavky spotřebitelů. Pokud by celková kapacita skladů byla nižší, řešení by nemohlo existovat. V našem případě je kapacita právě dostatečná, takže každý sklad bude plně využit. To přináší jisté nebezpečí zaokrouhlovacích chyb, které mohou znemožnit nalezení řešení. (Představte si, že vinou zaokrouhlovací chyby se během výpočtů třeba jen velmi málo sníží kapacita jednoho skladu a řešení úlohy tak nebude možné.) Při matematické formulaci praktických úloh je dobré na toto nebezpečí myslet a vyhýbat se "hraničním" formulacím rovností v LP.

Další ukázkou je tzv. min-max úloha.

Příklad 3.3. *Armáda pro potřeby cvičení musí ze dvou skladů o kapacitách* 6 a 5 *tun střeliva přepravit* 3, 2 a 2 *tuny střeliva na tři její střelnice. Úkolem je, v rámci rozložení rizik, minimalizovat maximální množství střeliva přepravované po jednotlivých cestách od skladů ke střelnicím.*

Řešení: Podmínky nyní jsou následovné.

- Kapacity skladů

$$\begin{aligned} x_{11} + x_{12} + x_{13} &\leq 6, \\ x_{21} + x_{22} + x_{23} &\leq 5. \end{aligned}$$

- Požadavky střelnic — z i-tého skladu dodat j-té množství střeliva

$$\begin{aligned} x_{11} + x_{21} &= 3, \\ x_{12} + x_{22} &= 2, \\ x_{13} + x_{23} &= 2. \end{aligned}$$

- Podmínky nezápornosti $x_{ij} \geq 0, \quad i = 1, 2,\ j = 1, 2, 3.$

Cílem je nyní minimalizovat převážené množství střeliva na každé jedné silnici (aby nedocházelo k příliš vysoké koncentraci střeliva na jednom místě). Pokud podmínku přepíšeme do hodnocení účelové funkce, získáme zápis

$$\min\left(\eta(\vec{x}) \;=\; \max\{x11, x12, x13, x21, x22, x23\}\right).$$

Tuto formuli lze následně přidáním dodatečné proměnné z adodatečných podmínek upravit na

$$\min \eta(\vec{x}) = z: \text{ kde } z \geq x_{ij} \text{ pro } i = 1, 2,\ j = 1, 2, 3\,.$$

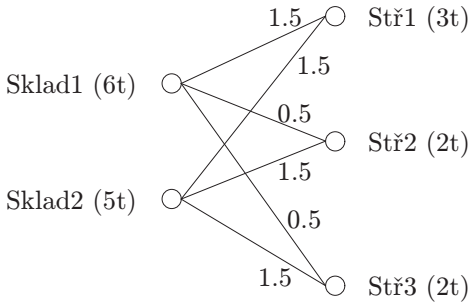
Optimální hodnota účelové funkce je

$$\eta(\vec{x}) = 1.5$$

a hodnoty jednotlivých složek / proměnných jsou:

$$\begin{array}{ccc} & z = 1.5 & \\ x_{11} = 1.5 & x_{12} = 0.5 & x_{13} = 0.5 \\ x_{21} = 1.5 & x_{22} = 1.5 & x_{23} = 1.5 \end{array}$$

Optimálním řešením je třeba přepravit z prvního skladu 1.5t k 1. střelnici, 0.5t k 2. střelnici a 0.5t k 3. střelnici. Z druhé skladiště se přepraví 1.5t k 1. střelnici, 1.5t k 2. střelnici a 1.5t k 3. střelnici. Obrázkem:



Všimněme si však, že uvedené řešení není jediné mezi optimálními – stejné hodnoty účelové funkce dosáhneme například pokud 2. i 3. střelnice dostávají po 1t střeliva z každého skladu. (Optimálních řešení lze takto sestrojit nekonečně mnoho. Je to však spíše vyjímečná situace.) □

Doplňkové otázky

(3.2.1) *Uměli byste podat jednoduché zdůvodnění, proč se po některé cestě v Příkladu 3.3 musí přepravit aspoň* 1.5*t střeliva?*

(3.2.2) *Jaké bude řešení Příkladu 3.3, pokud* 1. *střelnice bude požadovat* 2.5*t střeliva?*

3.3 Obecná úloha LP

Nyní přejdeme ke zobecnění matematické formalizace LP.

`Značení`: Pokud píšeme vektor $\vec{x}$, obvykle tím myslíme sloupcový vektor, avšak v jasných případech používáme stejné značení i pro řádkový vektor. Zápis $(\vec{x}, \vec{y})$ používáme pro zřetězení vektorů $\vec{x}$ a $\vec{y}$ do jednoho.

Pro zopakování uvádíme, že zápis $\vec{c} \cdot \vec{x}$ znamená běžný skalární součin (stejně dlouhých) vektorů $\vec{c} = (c_1, \ldots, c_k)$ a $\vec{x} = (x_1, \ldots, x_k)$, tj. výraz $c_1x_1 + \ldots c_kx_k$. Vektorová nerovnost $\vec{x} \leq \vec{c}$ vyjadřuje nerovnosti ve všech složkách vektorů zároveň, *tj.* $x_i \leq c_i$ pro $i = 1, \ldots, k$. Zápis $\boldsymbol{A} \cdot \vec{x}$ znamená běžný maticový součin – matice s vektorem příslušných rozměrů.

Definice 3.4. ***Úlohou lineárního programování*** (zkratkou LP) rozumíme úlohu nalezení vektoru $\vec{x}$, který maximalizuje, resp. minimalizuje, skalární součin $\vec{c} \cdot \vec{x}$ za podmínek $\boldsymbol{A} \cdot \vec{x} \leq \vec{b}$, $\vec{x} \geq 0$, kde $\boldsymbol{A}$ je daná *matice úlohy*, $\vec{b}$ je *vektor pravých stran* a $\vec{c}$ je *účelový vektor*.

Náš zkrácený maticový zápis podmínek úlohy

$$\begin{aligned} \boldsymbol{A} \cdot \vec{x} &\leq \vec{b}, \\ \vec{x} &\geq 0 \end{aligned}$$

ve skutečnosti znamená pro $\boldsymbol{A} = (a_{i,j})_{i,j=1}^{m,n}$ soustavu lineárních nerovností

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n &\leq b_1, \\ \vdots \qquad\qquad & \quad \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \ldots + a_{m,n}x_n &\leq b_m, \\ x_1, x_2, \ldots, x_n &\geq 0\,. \end{aligned}$$

`Značení`: Úlohu LP z Definice 3.4 zkráceně zapisujeme

$$\max \vec{c} \cdot \vec{x} \quad \text{pro} \quad \boldsymbol{A} \cdot \vec{x} \leq \vec{b} \quad \text{a} \quad \vec{x} \geq 0\,.$$

Pro lepší rozlišení také mluvíme o úloze *LP v základním tvaru*. Všimněme si, že v základním tvaru je každá složka vektoru $\vec{x}$ nezáporná.

`Definice`: *Zobecněnou úlohou LP* rozumíme úlohu $\max \vec{c} \cdot (\vec{x}, \vec{y})$ či $\min \vec{c} \cdot (\vec{x}, \vec{y})$ pro

$$\begin{aligned} \boldsymbol{A} \cdot (\vec{x}, \vec{y}) &\leq \vec{b}, \\ \boldsymbol{A'} \cdot (\vec{x}, \vec{y}) &\geq \vec{b'}, \\ \boldsymbol{A''} \cdot (\vec{x}, \vec{y}) &= \vec{b''}, \\ \vec{x} &\geq 0. \end{aligned}$$

O složkách vektoru $\vec{x}$ pak mluvíme jako o *omezených proměnných* a o složkách vektoru $\vec{y}$ jako o *neomezených proměnných*.

`Definice`: *Přípustné řešení* úlohy LP je vektor $(\vec{x}, \vec{y})$ splňující všechny podmínky (rovnosti a nerovnosti) úlohy.

Převody forem úloh LP

Pro co nejširší aplikovatelnost úlohy LP je samozřejmě vhodnější volit její zobecněný tvar, avšak z dobrých matematických důvodů je zase lepší zapisovat úlohy LP v základním tvaru. Že to neznamená žádné podstatné omezení, ukazuje následující tvrzení.

Věta 3.5. *Ke každé zobecněné úloze LP existuje ekvivalentní úloha v základním tvaru.*

`Důkaz`: Mějme zobecněnou úlohu danou maticemi a vektory $\boldsymbol{A}, \boldsymbol{A}', \boldsymbol{A}'', \vec{b}, \vec{b}', \vec{b}'', \vec{c}, \vec{x}, \vec{y}$ jako ve výše uvedené definici. Provedeme následující substituce:

- Náhrada proměnných $\vec{y}$ novými nezápornými proměnnými

$$y_i \longrightarrow (x_i' - x_i''), \quad x_i', x_i'' \geq 0.$$

- Obrácení nerovností z $\boldsymbol{A}'(\vec{x}, \vec{y}) \geq \vec{b}'$

$$a_j'(\vec{x}, \vec{y}) \geq b_j' \longrightarrow -a_j'(\vec{x}, \vec{y}) \leq -b_j'.$$

- Náhrada rovností dvojicema nerovností

$$a_j''(\vec{x}, \vec{y}) = b_j'' \longrightarrow a_j''(\vec{x}, \vec{y}) \leq b_j'', \ -a_j''(\vec{x}, \vec{y}) \leq -b_j''.$$

Nyní je úloha v základním tvaru. □

Doplňkové otázky

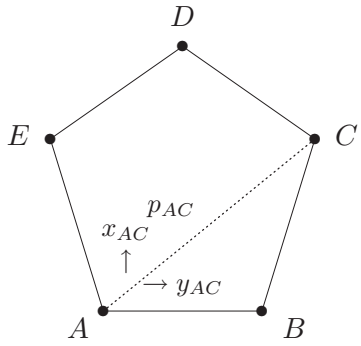
(3.3.1) *Převeďte do základního tvaru formulace úloh LP z této lekce.*

3.4 Další ukázky úloh

Pro bližší seznámení s dostupnými softwarovými prostředky pro řešení úloh LP si uvedeme následující, už méně triviální, optimalizační úlohu i s jejím vyřešením.

Příklad 3.6. *Minimalizace zátěže v síti s kruhovou topologií*

Uvažujme počítačovou síť $\mathcal{S}$ s kruhovou topologií, do které je zapojeno pět strojů $\mathcal{S} = \{A, B, C, D, E\}$. Data je v této síti možné po jednom kruhu přenášet oběma směry současně, požadavky na přenosovou kapacitu mezi dvojicemi počítačů jsou zadávány vždy v součtech pro oba směry. Vhodným rozložením přenosů do obou směrů na kružnici chceme minimalizovat zatížení nejzatíženějšího ze spojů mezi dvojicemi počítačů.



Vstup: Požadavky na přenosovou kapacitu mezi vybranými dvojicemi počítačů (vždy v součtech pro oba směry):

$$p_{AC} = 30,\ p_{BE} = 25,\ p_{DB} = 21,\ p_{CE} = 30,\ p_{AD} = 32\,.$$

Výstup: Distribuce přenosové kapacity $p_{ij} = x_{ij} + y_{ij}$ mezi dvojicemi počítačů $i, j \in \mathcal{S}$, kde x_{ij} udává tu část kapacity mezi stroji i a j směřovanou ve směru hodinových ručiček a y_{ij} udává část proti směru hodinových ručiček.

Účelovou funkcí přitom je minimalizovat síťovou zátěž z nejzatíženějšího ze spojů mezi dvojicemi počítačů.

Řešení: Nejprve za p_{ij} přímo dosadíme dané číselné hodnoty:

$$\begin{aligned} 30 &= x_{AC} + y_{AC} \\ 25 &= x_{BE} + y_{BE} \\ 21 &= x_{DB} + y_{DB} \\ 30 &= x_{CE} + y_{CE} \\ 32 &= x_{AD} + y_{AD} \end{aligned} \tag{1}$$

Zátěže jednotlivých spojů sítě (viz obrázek) vyjádříme jako:

$$\begin{aligned} z_{ED} &= y_{AC} + x_{BE} + x_{DB} + x_{CE} + y_{AD} \\ z_{DC} &= y_{AC} + x_{BE} + y_{DB} + x_{CE} + x_{AD} \\ z_{CB} &= x_{AC} + x_{BE} + y_{DB} + y_{CE} + x_{AD} \\ z_{BA} &= x_{AC} + y_{BE} + x_{DB} + y_{CE} + x_{AD} \\ z_{AE} &= y_{AC} + y_{BE} + x_{DB} + y_{CE} + y_{AD} \end{aligned} \tag{2}$$

Nyní zbývá jen minimalizovat maximum zátěží $z = \max\{z_{ED}, z_{DC}, z_{CB}, z_{BA}, z_{AE}\}$. Vztahy (2) spolu s operátorem maximalizace jednoduše přepíšeme (Příklad 3.3) na:

$$\begin{aligned} z &\geq y_{AC} + x_{BE} + x_{DB} + x_{CE} + y_{AD} \\ z &\geq y_{AC} + x_{BE} + y_{DB} + x_{CE} + x_{AD} \\ z &\geq x_{AC} + x_{BE} + y_{DB} + y_{CE} + x_{AD} \\ z &\geq x_{AC} + y_{BE} + x_{DB} + y_{CE} + x_{AD} \\ z &\geq y_{AC} + y_{BE} + x_{DB} + y_{CE} + y_{AD} \end{aligned} \tag{3}$$

Účelovou funkcí tak je

$$\min \eta = z\,.$$

Optimálním řešením $\min z$ za předpokladu vztahů (1) a (3) je $\boldsymbol{z = 58.5}$ a jednotlivé proměnné vyjdou:

$$\begin{aligned} x_{AC} = 30 &\quad , \quad y_{AC} = 0 \\ x_{BE} = 2 &\quad , \quad y_{BE} = 23 \\ x_{DB} = 0 &\quad , \quad y_{DB} = 21 \\ x_{CE} = 30 &\quad , \quad y_{CE} = 0 \\ x_{AD} = 5.5 &\quad , \quad y_{AD} = 26.5 \end{aligned}$$

Jak jsme k tomu došli? Viz následující text. □

Praktické řešení úloh LP

Závěrem následuje několik ukázek, kterak pomocí softwaru volně dostupného na internetu můžeme snadno řešit (nevelké) úlohy LP. Demonstrujeme přístup na řešení Příkladu 3.6.

V prvé řadě si ukážeme formulaci zadání pro aplet [12]:

```
min: z;
z >= yAC + xBE + xDB + xCE + yAD;
z >= yAC + xBE + yDB + xCE + xAD;
z >= xAC + xBE + yDB + yCE + xAD;
z >= xAC + yBE + xDB + yCE + xAD;
z >= yAC + yBE + xDB + yCE + yAD;
xAC + yAC = 30;
xBE + yBE = 25;
xDB + yDB = 21;
xCE + yCE = 30;
xAD + yAD = 32;
```

Další interaktiví web formulář pro LP a IP, vyvinutý speciálně pro účely našeho předmětu, je přístupný na adrese [15]. (Jeho výhodou jsou nicotné požadavky na klienta, takže na něj lze přistupovat i z PDA zařízení.) Příslušna formulace Příkladu 3.6 následuje:

```
    Účelová funkce:
min z
    Podmínky:
yAC + xBE + xDB + xCE + yAD < z
yAC + xBE + yDB + xCE + xAD < z
xAC + xBE + yDB + yCE + xAD < z
xAC + yBE + xDB + yCE + xAD < z
yAC + yBE + xDB + yCE + yAD < z
xAC + yAC = 30
xBE + yBE = 25
xDB + yDB = 21
xCE + yCE = 30
xAD + yAD = 32
```

Další možností je využít systém Maxima [14]

```
load("load_simplex")
minimize_sx(z, [yAC + xBE + xDB + xCE + yAD <= z,
            yAC + xBE + yDB + xCE + xAD <= z,
            xAC + xBE + yDB + yCE + xAD <= z,
            xAC + yBE + xDB + yCE + xAD <= z,
            yAC + yBE + xDB + yCE + yAD <= z,
            xAC + yAC = 30,
            xBE + yBE = 25,
            xDB + yDB = 21,
            xCE + yCE = 30,
            xAD + yAD = 32]), nonegative_sx=true;
```

Nakonec pro úplnost uvedeme postup řešení v komerčním systému Waterloo Maple (9.5): (Poznamenáváme však, že autor nepodporuje používání komerčního softwaru v situacích, kdy je dostupný dostačující volný produkt.)

```
with(simplex):
minimize(z, {yAC + xBE + xDB + xCE + yAD <= z,
         yAC + xBE + yDB + xCE + xAD <= z,
         xAC + xBE + yDB + yCE + xAD <= z,
         xAC + yBE + xDB + yCE + xAD <= z,
         yAC + yBE + xDB + yCE + yAD <= z,
         xAC + yAC = 30,
         xBE + yBE = 25,
         xDB + yDB = 21,
         xCE + yCE = 30,
         xAD + yAD = 32}, NONNEGATIVE);
```

Doplňkové otázky

(3.4.1) *Jaký vyjde výsledek (optimální zatížení spojů) Příkladu 3.6 pro vstup změněný na* $p_{AC} = 20$*?*

(3.4.2) *Jaký vyjde výsledek (optimální zatížení spojů) Příkladu 3.6 pro vstup změněný na* $p_{AD} = 22$*?*

Rozšiřující studium

Množství dalších příkladů formulací úloh LP čtenář najde v učebnici [3], ze které jsme převzali i některé naše příklady. Co se týče praktického řešení úloh LP na počítači, znovu připomínáme výše zmíněné volně dostupné programové prostředky [12, 15, 14]. Čtenáři zajímajícímu se o alternativní pohled na úlohy LP a jejich řešení doporučujeme další vynikající učebnici [2].

4 Konvexita a mnohostěny

Úvod

V této lekci si uvedeme či zopakujeme základní matematické pojmy potřebné pro pozdější pochopení a zdůvodnění principů lineární optimalizace a jejího řešení tzv. simplexovou metodou. Mimo připomenutí běžných pojmů algebry a topologie se jedná především o pojmy konvexity a mnohostěnu. V návaznosti na konvexní množiny si také uvedeme zajímavé a užitečné tzv. Farkasovo lema.

Samotný důležitý pojem mnohostěnu sice je intuitivní v dimenzích 2 *nebo* 3*, ale ve vyšších dimenzích již přináší netriviální komplikace. Proto v našem textu důsledně rozlišujeme mezi pojmem polyedru (popsaného svými stěnami) a polytopu (popsaného svými vrcholy).*

Jedná se o patrně matematicky nejnáročnější (a nejformálnější) lekci této části našeho učebního textu, která je zařazena pro matematickou úplnost a hlubší porozumnění přednesené látce. Pro samotné studium postupů optimalizace není nutno plně pochopit všechny uvedené důkazy, ale čtenář by měl nastudovat alespoň uvedené definice.

Cíle

Úkolem této lekce je především vybudovat matematický aparát nutný k formálnímu popisu a zdůvodnění řešení úloh LP simplexovou metodou. Zdejší teoretické poznatky budou dále využity při zavedení duality úloh LP i později při matematickém popisu řešení úloh IP. (V případě, že čtenáře nezajímá matematické pozadí lineární optimalizace, může zde uvedená formální matematická tvrzení přeskočit, avšak měl by se seznámit s pojmy konvexity a mnohostěnu a pochopit význam Farkasova lematu.)

4.1 Vybrané matematické pojmy

Tato část pro úplnost jen stručně opakuje některé pojmy, které budeme dále používat.

`Definice` některých základních pojmů lineární algebry:

- d-dimenzionální *euklidovský prostor* je vektorový prostor $\mathbb{R}^d$ s klasickým skalárním součinem.
- *Afinní podprostor* je vektorový podprostor posunutý o daný vektor (tj. nemusí procházet počátkem) .
- *Dimenze* podprostoru je počet prvků jeho báze. *Dimenze množiny* X je nejmenší dimenzí afinního podprostoru obsahujícího X.
- *Nadrovinou* v $\mathbb{R}^d$ rozumíme afinní podprostor dimenze $d-1$.
- *Poloprostor v* $\mathbb{R}^d$ je uzavřená podmnožina, jejíž hranicí je nadrovina, neformálními slovy množina všech bodů "na jedné straně nadroviny".
- Nechť $\|\vec{x}\| = \sqrt{x_1^2 + \ldots + x_d^2}$ značí *délku* vektoru $\vec{x} \in \mathbb{R}^d$.

Nadrovina v $\mathbb{R}^d$ je určena lineární rovnicí $\vec{a} \cdot \vec{x} = a_1x_1 + \ldots + a_dx_d = b$. Poloprostor v $\mathbb{R}^d$ je určen lineární nerovnicí $\vec{a} \cdot \vec{x} = a_1x_1 + \ldots + a_dx_d \leq b$.

`Značení`: Pokud H označuje nadrovinu, H^+ a H^- označuje příslušné dva poloprostory určené H (bez definovaného rozlišení, který ze dvou je H^+ a který H^-).

`Definice` několika běžných topologických pojmů:

- Mějme množinu $X \subseteq \mathbb{R}^d$. X je *omezená*, pokud existuje konstanta $k \in \mathbb{R}$ taková, že $\forall \vec{x} \in X$ platí $\|\vec{x}\| < k$.
- Množina $X \subseteq \mathbb{R}^d$ je *uzavřená*, pokud pro každou konvergentní posloupnost $(x_1, x_2, ...) \subseteq X$ platí $(\lim_{n\to\infty} x_n) \in X$. (Neformálně, že "hranice X patří" do X.)
- Množina $X \subseteq \mathbb{R}^d$ je *otevřená*, pokud její doplněk je uzavřený.
- Funkce f se je *spojitá*, je-li vzorem otevřené množiny (tj. $f^{-1}(U)$) opět otevřená množina. (Neformálně pro každé dva "dostatečně blízké" body jsou si blízké i jejich funkční hodnoty.)

`Fakt`: Poloprostor i nadrovina jsou zřejmě uzavřené množiny.
Průnikem i sjednocením konečně mnoha uzavřených (otevřených) množin je opět uzavřená (otevřená) množina.

Následující pokročilý analytický pojem kompaktnosti bude ve velké míře využíván v další teoretické části.

Definice 4.1. ***Kompaktní množina*** X je taková,
ve které každá nekonečná posloupnost $(x_1, x_2, ...) \subseteq X$ má podposloupnost konvergující uvnitř X.

Věta 4.2. *Nechť $X \subseteq \mathbb{R}^d$.*
a) Pak X je kompaktní právě když je X omezená a uzavřená.
b) Každá spojitá funkce $f : X \to \mathbb{R}$ má na kompaktní množině X globální maximum i minimum.

Doplňkové otázky

(4.1.1) *Jaká je dimenze prázdného podprostoru a jaké jednoho bodu?*

(4.1.2) *Jakou nejvyšší dimenzi může mít množina 7 bodů?*

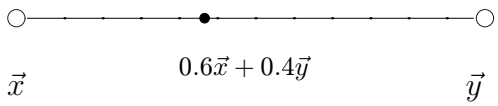
(4.1.3) *Co znamená, že v rovnici $a_1x_1 + \ldots + a_dx_d = b$ nadroviny je $b = 0$?*

(4.1.4) *Je celý Euklidovský prostor otevřená nebo uzavřená množina?*

4.2 Konvexita: definice a vlastnosti

Pojem konvexní kombinace a konvexní množiny bude potřebný při popisu množiny přípustných řešení úlohy LP a také pro pozdější zavedení simplexové metody.

Definice 4.3. ***Konvexní kombinací*** vektorů $\vec{x}, \vec{y} \in R^d$ rozumíme každý vektor tvaru $\alpha\vec{x} + (1-\alpha)\vec{y}$, $\alpha \in \langle 1, 0\rangle$. (Tj. všechny body na úsečce s konci $\vec{x}$ a $\vec{y}$.) Množina $X \subseteq \mathbb{R}^d$ je *konvexní*, pokud s každými dvěma body $x, y \in X$ patří do X i celá úsečka xy, tj. všechny konvexní kombinace vektorů $\vec{x}, \vec{y}$.



Obrázek 4.1: Příklad konvexní kombinace vektorů $\vec{x}, \vec{y}$.

Lema 4.4. *Průnik $X \cap Y$ dvou konvexních množin X, Y je konvexní.*

Důkaz: Nechť $x, y \in X \cap Y$. Pak $x, y \in X$ a dle konvexity X i celá úsečka xy náleží X. Stejně tak xy náleží Y a celkově náleží do $X \cap Y$. To je definice konvexnosti $X \cap Y$. □

Důsledek 4.5. *Množina všech přípustných řešení úlohy LP je konvexní.*

Důkaz: Množina všech řešení rovnice $a_1x_1 + \ldots + a_dx_d = b$ (nadrovina) je zřejmě konvexní a totéž platí pro řešení nerovnice $a_1x_1 + \ldots + a_dx_d \leq b$ (poloprostor). Jejich průnik je tudíž také konvexní. □

Při popisech konvexních množin jsou nejdůležitější jejich "okrajové" body, které, jak později uvidíme přesně, odpovídají tomu, co si představujeme pod pojmem "vrcholu" mnohostěnu.

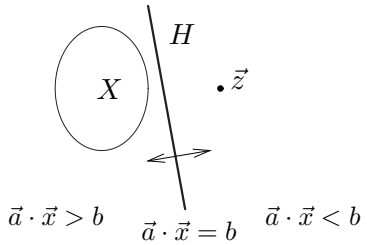
Definice: Nechť K je konvexní množina. Bod $v \in K$ je *krajním bodem* K, pokud neexistují body $x, y \in K \setminus \{v\}$ takové, že v by bylo konvexní kombinací x a y.

Konvexnost množiny má množství teoretických důsledků, z nichž je pro nás nejpotřebnější následující tzv. *věta o oddělující nadrovině*. (Obrázek 4.2)

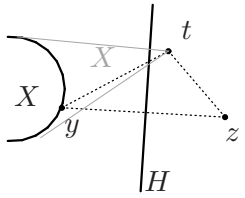
Věta 4.6. *Nechť $X \subseteq \mathbb{R}^d$ je uzavřená a konvexní množina a $\vec{z} \in \mathbb{R}^d$ je bod $\vec{z} \notin X$. Pak existuje nadrovina $H \subseteq \mathbb{R}^d$ oddělující $\vec{z}$ od X, jinými slovy existují $\vec{a} \in \mathbb{R}^d$, $b \in \mathbb{R}$ takové, že $\vec{a} \cdot \vec{z} < b$ a $\forall \vec{x} \in X\colon \vec{a} \cdot \vec{x} > b$.*

Důkaz: Nechť $\vec{x} \in X$ je libovolné a $l = \|\vec{x} - \vec{z}\|$. Pak vezmeme $X' \subseteq X$ podmnožinu všech bodů ve vzdálenosti $\leq l$ od $\vec{z}$. X' je uzavřená, omezená, tedy kompaktní, a tudíž podle Věty 4.2 má funkce vzdálenosti $f(\vec{x}) = \|\vec{x} - \vec{z}\|$ minimum na X' v některém bodě $\vec{y} \in X'$ ($\vec{y}$ je bod X nejbližší k $\vec{z}$).

Za oddělující nadrovinu H vezmeme kolmou nadrovinu k úsečce yz procházející jejím středem, tj. volíme $\vec{a} = \vec{y} - \vec{z}$ a $b = \frac{\vec{y}+\vec{z}}{2} \cdot \vec{a}$. Pokud by existoval bod $\vec{t} \in X$, pro který



Obrázek 4.2: Nadrovina H oddělující konvexní množinu $X \subseteq H^+$ od bodu $\vec{z} \in H^-$.



$\vec{a}\cdot\vec{t} < b$ (tzn. $\vec{t}$ leží na stejné straně H jako $\vec{z}$), pak by i celá úsečka yt patřila do X podle definice konvexity. Viz obrázek. Jelikož však úsečka yt protíná oddělující nadrovinu H, musí na ní podle trojúhelníkové nerovnosti ležet bod bližší k $\vec{z}$ než je $\vec{y}$, a to je spor. (Uvědomte si, že zmíněný bližší bod nemusí být $\vec{t}$ samotný.) □

Velmi známý důsledek předchozí věty je znám jako *Farkasovo lema*. V "tradiční" formulaci je toto tvrzení však dost obtížně uchopitelné (jak se sami můžete přesvědčit), proto si my Farkasovo lema doplníme ještě jinou jeho formulací coby Důsledek 4.8.

Důsledek 4.7. (Farkasovo lema) *Soustava lineárních rovnic $\vec{u}^T \cdot \boldsymbol{A} = \vec{c}^T$ má nezáporné řešení $\vec{u} \geq 0$ právě tehdy, když pro každé řešení soustavy $\boldsymbol{A} \cdot \vec{y} \leq 0$ platí $\vec{c} \cdot \vec{y} \leq 0$.*

`Důkaz` $\Rightarrow$: Předpokládejme, že $\vec{u} \geq 0$ a $\vec{y}$ jsou takové, že $\vec{u}^T \cdot A = \vec{c}^T$ a $A \cdot \vec{y} \leq 0$. Potom $\vec{c} \cdot \vec{y} = (\vec{u}^T \cdot A) \cdot \vec{y} = \vec{u}^T \cdot (A \cdot \vec{y}) \leq 0$.

`Důkaz` $\Leftarrow$ (sporem): Předpokládejme, že $\vec{u}^T \cdot A = \vec{c}^T$ nemá nezáporné řešení. Ukážeme, že potom existuje $\vec{y}$ takové, že $A \cdot \vec{y} \leq 0$ a $\vec{c} \cdot \vec{y} > 0$. Uvažme množinu $P = \{\vec{u}^T \cdot A : \vec{u} \geq 0\}$. Množina P je zřejmě uzavřená a konvexní a platí, že $\vec{c} \notin P$. Podle Věty 4.6 existuje oddělující nadrovina mezi $\vec{c}$ a množinou P, tj. existují $\vec{a}$ a b takové, že $\vec{a} \cdot \vec{c} < b$ a zároveň $\vec{a} \cdot \vec{x} \geq b$ pro každé $\vec{x} \in P$.

Dokážeme, že lze volit $b = 0$ (tj. že naše oddělující nadrovina může procházet počátkem). Zřejmě $\vec{0} \in P$ a tedy $\vec{a} \cdot \vec{c} < 0$, protože $0 = \vec{a} \cdot \vec{0} \geq b$. Přepokládejme, že existuje $\vec{x} \in P$ takové, že $\vec{a} \cdot \vec{x} < 0$. Z definice P plyne, že pro každé $\lambda \geq 0$ platí $\lambda \cdot \vec{x} \in P$. Avšak pro dostatečně velké λ dostaneme, že $\vec{a} \cdot (\lambda \cdot \vec{x}) = \lambda \cdot (\vec{a} \cdot \vec{x}) < b$, neboť $\lambda \cdot (\vec{a} \cdot \vec{x}) \to -\infty$ pro $\lambda \to \infty$, a tedy že P se nachází na obou stranách oddělující nadroviny, ale to je zřejmý spor. Z toho plyne, že $\vec{a} \cdot \vec{x} \geq 0$ pro každé $\vec{x} \in P$, tedy že můžeme volit $b = 0$.

Konečně zvolme $\vec{y} = -\vec{a}$. Potom $\vec{c} \cdot \vec{y} = -\vec{a} \cdot \vec{c} > 0$ a zároveň $A \cdot \vec{y} = -A \cdot \vec{a} \leq 0$, což plyne dosazením jednotlivých řádků matice A za $\vec{x}$ do nerovnice $\vec{a} \cdot \vec{x} \geq b = 0$ (všechny řádkové vektory A jsou zřejmě v P). □

Důsledek 4.8. (Farkasovo lema trochu jinak) *Nechť soustava $\boldsymbol{A} \cdot \vec{x} \leq \vec{c}$, $\vec{x} \geq 0$ nemá řešení. Pak existuje vektor $\vec{\lambda} \geq 0$ takový, že $\vec{\lambda} \cdot \boldsymbol{A} \geq 0$ a $\vec{\lambda} \cdot \vec{c} < 0$.*

`Důkaz`: Zřejmě soustava

$$(\boldsymbol{A}) \cdot (\vec{x}) \leq \vec{c},\ \vec{x} \geq 0$$

má řešení právě když má řešení následující soustava

$$(\boldsymbol{A} \mid \boldsymbol{I}) \cdot \begin{pmatrix} \vec{x} \\ \vec{z} \end{pmatrix} = \vec{c},\ \vec{x} \geq 0,\ \vec{z} \geq 0\,.$$

Nechť $\vec{u}^T = (\vec{x}^T, \vec{z}^T)$ a $\boldsymbol{A'} = \begin{pmatrix} \boldsymbol{A}^T \\ \boldsymbol{I} \end{pmatrix}$. Z Farkasova lematu 4.7 pro $\boldsymbol{A'}$ a $\vec{u}$ plyne, že existuje $\vec{y}$ takové, že $\boldsymbol{A'} \cdot \vec{y} \leq 0$ a $\vec{c} \cdot \vec{y} > 0$. Nyní zvolme $\vec{\lambda} = -\vec{y}^T$. Pak $\boldsymbol{A'} \cdot \vec{\lambda}^T = -\boldsymbol{A'} \cdot \vec{y} \geq 0$, což lze rozepsat jako $A^T \cdot \vec{\lambda}^T = (\vec{\lambda} \cdot A)^T \geq 0$ a navíc $\boldsymbol{I} \cdot \vec{\lambda} = \vec{\lambda} \geq 0$. Zároveň platí, že $\vec{\lambda} \cdot \vec{c} = -\vec{c} \cdot \vec{y} < 0$. □

`Komentář:` Alternativní formulace Farkasova lematu v Důsledku 4.8 má následující pěkný význam: Nemá-li soustava $\boldsymbol{A} \cdot \vec{x} \leq \vec{c}$, $\vec{x} \geq 0$ řešení, lze přímo získat nezápornou lineární kombinací řádků soustavy spor $0 \leq \vec{\lambda} \cdot \boldsymbol{A} \cdot \vec{x} \leq \vec{\lambda} \cdot \vec{c} < 0$.

Doplňkové otázky

(4.2.1) *Je prázdná množina konvexní?*

(4.2.2) *Je sjednocení dvou konvexních množin konvexní?*

(4.2.3) *Které (jednotlivé) body můžeme vypustit ze čtverce tak, že výsledný útvar je stále konvexní?*

(4.2.4) *Které jsou krajní body kruhu?*

(4.2.5) *Jak byste získali přímý spor z následující (neřešitelné) soustavy nerovností?*

$$\begin{aligned} x + y + z &\leq 2 \\ x + y - z &\leq 3 \\ x, y, z &\geq 0 \end{aligned}$$

(4.2.6) *Jak byste získali přímý spor z následující (neřešitelné) soustavy nerovností?*

$$\begin{aligned} x + y + z &\leq 5 \\ x - y - z &\leq 3 \\ 3x - y - z &\geq 15 \end{aligned}$$

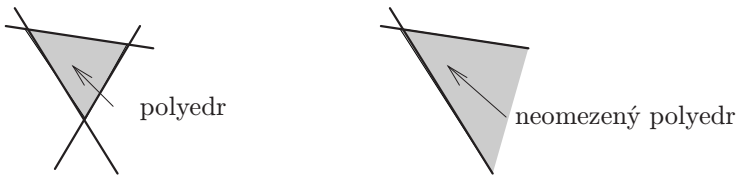
4.3 Mnohostěny

Pojem vícedimenzionálního *mnohostěnu* je klíčový pro budoucí řešení úloh LP simplexovou metodou. Je tomu tak proto, že množina přípustných řešení úlohy LP je právě mnohostěnem. Samotné zavedení tohoto pojmu ve vyšších dimenzích však přináší následující hluboký problém: Mnohostěn lze popsat jeho *vrcholy* nebo jeho stěnami (*facetami*), ale převod mezi těmito popisy není vůbec jednoduchý a může z výpočetního hlediska vést k exponenciálnímu nárustu složitosti.

Proto se na mnohostěny budeme dívat dvěma odlišnými formálními pohledy, nejprve pohledem na jeho stěny.

Definice 4.9. ***Polyedr*** v $\mathbb{R}^d$ je průnikem konečně mnoha poloprostorů.
(Je to konvexní a uzavřená množina podle Lemmatu 4.4.)

`Komentář:` Dobře si uvědomme, že z této definice vůbec nevyplývá omezenost polyedru.



Definice 4.10. ***Stěna*** F d-dimenzionálního polyedru P
je každá taková množina $F \subseteq P$, pro kterou existuje nadrovina H v $\mathbb{R}^d$ určující poloprostor H^+, že $P \subseteq H^+$ a $F = P \cap H$.

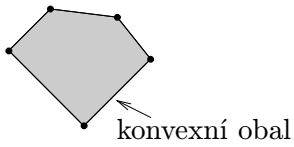
> `Komentář:` Všimněme si, že i prázdná množina může být stěnou polyedru. Obvykle se za stěnu bere navíc i celý polyedr P a pak se $\emptyset$ a celému P říká *nevlastní stěny*. Všechny stěny včetně nevlastních zřejmě tvoří svaz s nejmenším i největším prvkem, navíc se jedná u omezeného polyedru o svaz atomický (atomy jsou jednotlivé vrcholy).

`Značení`: *Dimenzí* stěny F přirozeně míníme afinní dimenzi množiny F. Stěny dimenze 0 nazýváme *vrcholy*, stěny dimenze 1 nazýváme *hranami* a stěny dimenze $d-1$ *facetami* d-dimenzionálního polyedru P.

`Fakt`: Přímo z definic plyne, že vrcholy polyedru dimenze větší než 0 jsou jeho krajními body a naopak. (Pro dimenzi 0 se o tento fakt definice rozšíří.)

Následuje druhý pohled na mnohostěny podle jejich vrcholů. Poznamenáváme, že obecně se v matematice mluví o *konvexních* mnohostěnech, ale jelikož zde uvažujeme jen konvexní množiny, tento přívlastek pro jednoduchost vynecháváme.

`Definice`: *Konvexním obalem* $conv(V)$ množiny $V \subseteq \mathbb{R}^d$ je průnik všech konvexních množin obsahujících V; tj. neformálně množina všech bodů, které lze získat z bodů V (násobnými) konvexními kombinacemi.



Definice 4.11. ***Polytop*** je konvexním obalem konečně mnoha bodů v $\mathbb{R}^d$.
(Opět se jedná o uzavřenou konvexní množinu, navíc vždy omezenou.)

`Poznámka`: Citovaná učebnice [3] nerozlišuje mezi pojmy polyedru a polytopu a používá jen slovo "polyédr" v obou našich významech. To rozhodně není matematicky přesné. My přejdeme ke zjednodušené záměně pojmů polyedru a polytopu až po důkazu následující Věty 4.12.

Ekvivalence pohledů na mnohostěny

Nakonec si ukážeme, že z matematického pohledu lze pojmy polyedru a polytopu dle potřeby zaměnit. Platí však, že složitost popisu téhož tělesa jako polytopu se může diametrálně lišit od jeho popisu coby polyedru.

`Fakt`: d-dimenzionální hyperkrychle má $2d$ stěn a 2^d vrcholů.

Věta 4.12. *Omezený polyedr je polytop a naopak.*

Důkaz této důležité věty bude podán následujícími tvrzeními.

Lema 4.13. *Nechť P je omezený polyedr a F jeho faceta. Pak existuje vrchol (krajní bod) v P který neleží na F.*

`Důkaz`: Uvědomme si, že v dimenzi 0 je tvrzení triviální – jakékoliv těleso dimenze 0 je tvořeno jediným bodem, který je zároveň krajním bodem, a jedinou facetou takového tělesa je prázdná množina. Dále postupujeme matematickou indukcí podle dimenze P.

Nechť H je nadrovina definující facetu F v P. Podle Věty 4.2 má funkce vzdálenosti bodu od H své globální maximum na kompaktním P, řekněme v bodě $x \in P \setminus F$. Označme H' nadrovinu rovnoběžnou s H a rocházející x. Pokud $H' \cap P = \{x\}$, máme požadovaný vrchol. Jinak je $H' \cap P$ polyedrem menší dimenze než P a podle indukčního předpokladu má $H' \cap P$ nějaký vrchol. □

Následující tvrzení pro zjednodušení nedokazujeme, ale odkazujeme čtenáře na argumenty uvedené v Oddíle 6.2.

Tvrzení 4.14. *Nechť x je krajním bodem polyedru P dimenze d. Pak existuje d facet $F_1, \ldots, F_d$ polyedru P takových, že $F_1 \cap \ldots \cap F_d = \{x\}$.*

Lema 4.15. *Nechť P je omezený polyedr a X množina jeho krajních bodů. Pak X je konečná a $\mathrm{conv}(X) = P$ (tj. X jsou vrcholy tohoto polytopu).*

`Důkaz`: V prvé řadě, jelikož P má konečně mnoho facet, řekněme ℓ, má podle Tvrzení 4.14 nejvýše $\binom{\ell}{d}$ krajních bodů. Proto je X konečná. Označme $T = \mathrm{conv}(X)$.

Zřejmě $T \subseteq P$, protože P je konvexní z definice. Předpokládejme nyní, že existuje $x \in P \setminus T$. Z Věty 4.6 plyne, že mezi T a x existuje oddělující nadrovina H. Nechť $x \in H^-$ (H^- je záporná strana H), tj. $T \subseteq H^+$. Uvažme polyedr $P' = P \cap H^-$ s facetou $F = P \cap H$. Podle Lematu 4.13 existuje další krajní bod $s \in P'$ nenáležející F. Proto je s zároveň krajním bodem původního P. Jelikož však podle naší volby $X \subseteq H^+$ a $s \in H^-$, máme $s \notin X$, spor. □

Tímto jsme dokázali, že z popisu polyedru odvodíme jeho popis (téhož tělesa) coby polytopu. V opačném směru důkazu Věty 4.12 potřebujeme popsat daný polytop coby polyedr. Využijeme následujícího užitečného pojmu:

`Definice`: Nechť $S \subseteq \mathbb{R}^n$. *Polární množinou* k S nazveme

$$S^* = \{\vec{y} \in \mathbb{R}^n : \vec{x} \cdot \vec{y} \leq 1 \text{ pro všechna } \vec{x} \in S\}\,. \tag{4}$$

`Fakt`: Pro každou $S \subseteq \mathbb{R}^n$ je $(S^*)^* \supseteq S$.

`Komentář:` Praktický význam polární operace pro nás tkví v tom, že "převrací" popis polytopu P na popis P^* jako polyedru, zvaného *polární polyedr* (a taky naopak). Blíže viz následující tvrzení.

Lema 4.16. *Nechť P je polytop. Pak P^* je omezený polyedr.*

`Důkaz`: Podle definice

$$P^* = \{\vec{y} \in \mathbb{R}^n : \vec{x} \cdot \vec{y} \leq 1 \text{ pro všechna } x \in X\}\,.$$

Označme X konečnou množinu vrcholů P. Tvrdíme, že

$$P^* = Q, \quad \text{kde} \quad Q = \{\vec{y} \in \mathbb{R}^n : \vec{x} \cdot \vec{y} \leq 1 \text{ pro všechna } x \in X\}\,,$$

z čehož je ihned vidět, že P^* je průnikem konečně mnoha poloprostorů $\vec{x} \cdot \vec{y} \leq 1$, tedy polyedrem. Zřejmě $P^* \subseteq Q$. Nechť naopak $y \in Q$, pak $\vec{x} \cdot \vec{y} \leq 1$ pro všechna $x \in X$ a tudíž i pro každé x, které je konvexní kombinací z X. Proto $q \in P^*$. Navíc je Q zřejmě omezené. □

Nyní pokud P je polytop, pak $Q = P^*$ je omezený polyedr, a tudíž podle Lematu 4.15 je Q i polytop. Uplatněním stejné úvahy ještě jednou dostáváme, že i Q^* je omezený polyedr. Pro dokončení důkazu Věty 4.12 zbývá zdůvodnit, že $Q^* = P$. Je zřejmé, že posunutím souřadnic můžeme předpokládat, že počátek souřadnic je vnitřním bodem P.

Lema 4.17. *Nechť P je polytop obsahující počátek souřadnic 0 jako svůj vnitřní bod. Pak $(P^*)^* = P$.*

Důkaz: Označme Y množinu vrcholů polyedru–polytopu P^*. Chceme dokázat

$$P = S, \text{ kde } S = \{\vec{x} \in \mathbb{R}^n : \vec{x} \cdot \vec{y} \leq 1 \text{ pro všechna } y \in Y\}.$$

Jelikož již víme $S \supseteq (P^*)^* \supseteq P$, stačí nám pro důkaz sporem předpokládat, že $\vec{x}^o \notin P$ pro nějaké $\vec{x}^o \in S$. Podle Věty 4.6 (o oddělující nadrovině) pak existují $\vec{a}, b$ takové, že $\vec{a} \cdot \vec{x}^o < b$ a přitom $\vec{a} \cdot \vec{x} > b$ pro všechna $\vec{x} \in P$. Jelikož $0 \in P$, platí $0\vec{x} = 0 > b$. Vydělením uvedených nerovností záporným b proto dostáváme

$$\vec{c} \cdot \vec{x}^o > 1\,, \quad \forall \vec{x} \in P : \vec{c} \cdot \vec{x} < 1\,,$$

kde $\vec{c} = \frac{1}{b}\vec{a}$. Proto podle definice (4) je $\vec{c} \in P^*$. Jelikož P^* je také polytop, je $\vec{c}$ konvexní kombinací jeho vrcholů z Y; $\vec{c} = \lambda_1 \vec{y}^1 + \ldots \lambda_k \vec{y}^k$. Nakonec získáme úpravou

$$1 < \vec{c} \cdot \vec{x}^o = \left(\sum_{i=1}^{k} \lambda_i \vec{y}^{\,i}\right) \cdot \vec{x}^o = \sum_{i=1}^{k} \lambda_i (\vec{x}^o \cdot \vec{y}^{\,i}) \leq \sum_{i=1}^{k} \lambda_i \cdot 1 = 1\,,$$

což je zřejmý spor. □

Doplňkové otázky

(4.3.1) *Který mnohostěn je polární k hyperkrychli? Jak byste jej jednoduše popsali?*

(4.3.2) *Jaký mnohostěn (například) má velmi mnoho stěn při relativně málo vrcholech?*

*(4.3.3)** *V jaké dimenzi lze konstruovat mnohostěn, jehož každé dva vrcholy jsou spojené některou hranou?*

(4.3.4) *Najdete příklad mnohostěnu, který nemá žádnou stěnu podle Definice 4.10? (Pozor, nejedná se o prázdný mnohostěn.)*

*(4.3.5)** *Dokažte podrobně Tvrzení 4.14.*

*(4.3.6)** *Dokažte, že pro každou množinu T platí $((T^*)^*)^* = T^*$. (Za pomoci faktu $(S^*)^* \supseteq S$.)*

*(4.3.7)** *Jak byste využili poznatek 4.3.6 k alternativnímu důkazu Lematu 4.17?*

Rozšiřující studium

Pro zopakování potřebných pojmů z Oddílu 4.1 by čtenáři měla dostačovat jakákoliv základní učebnice matematické analýzy či metrické topologie. Následující pojmy konvexity a mnohostěnů patří do oblasti tzv. polyedrální kombinatoriky, k jejímuž hlubšímu studiu lze doporučit například knihu [11]. Toto doporučení se týká skutečně jen zájemců o studium přímo této oblasti, neboť pro náš další výklad postačí znalosti mnohostěnů prezentované v našem textu.

Mnohostěnům v lineární optimalizaci a problematice kolem Farkasova lematu se dobře věnují třeba [7, Chapter I.4] nebo [10, Chapter 2]. Tam se čtenář může dozvědět i některé rozšiřující informace, které se do našeho textu nevešly, třeba jak se neomezené polyedry popisují pomocí svých vrcholů a tzv. "paprsků".

5 Dualita úloh LP

Úvod

V návaznosti na předchozí teoretickou lekci nyní zavedeme užitečný pojem duality úloh LP, který nám mimo jiné poskytne dobrou charakterizaci optimálních řešení LP. Platí totiž, že původní (primární) i duální úloha mají zároveň optimální řešení stejné účelové hodnoty. Zjednodušeně řečeno, duální úloha k úloze v základním tvaru se získá transpozicí matice úlohy a záměnou účelového vektory s vektorem pravých stran, při současné změně směru nerovností.

Cíle

Úkolem této lekce je seznámit čtenáře s pojmem duální úlohy LP a s jejím významem, jak ve formálním, tak i v neformálním podání na příkladech. (Opět platí poznámka, že čtenář nezajímající se o teoretické pozadí lineární optimalizace může matematická tvrzení této lekce přeskočit.)

5.1 Základní tvar duality

Běžná (a snadno zapamatovatelná) definice duální úlohy LP se vztahuje k formulaci původní úlohy (zvané *primární úloha*) v základním tvaru, ale později si ukážeme, jak lze převést na duální i úlohu LP v obecném tvaru.

Definice 5.1. ***Duální úlohou*** LP k úloze v základním tvaru

$$\max\ \vec{c}\cdot\vec{x}\ \text{ pro }\ \boldsymbol{A}\cdot\vec{x}\le\vec{b},\ \vec{x}\ge 0$$

je úloha

$$\min\ \vec{b}\cdot\vec{y}\ \text{ pro }\ \vec{y}\cdot\boldsymbol{A}\ge\vec{c},\ \vec{y}\ge 0\,.$$

Složky vektoru $\vec{y}$ jsou tzv. *duální proměnné*.

`Komentář:` Jinými slovy, při formulaci duální úlohy přiřadíme nové (duální) proměnné y_j jednotlivým nerovnicím primární úlohy, matici $\boldsymbol{A}$ transponujeme a vektory účelový $\vec{c}$ a pravých stran $\vec{b}$ vzájemně zaměníme. Nově získané duální nerovnosti budou v opačném směru a duální účelová funkce $\vec{b}\cdot\vec{y}$ se bude minimalizovat. I duální proměnné budou všechny nezáporné.

Příklad 3.1 o hranolkách a lupíncích vede na níže zapsanou duální úlohu LP.

$$\begin{array}{rcl|rcl}
\max\ 80x_1+50x_2: & & & \min\ 100y_1+16y_2: & & \\
2x_1+1.5x_2 & \le & 100 & 2y_1+0.4y_2 & \ge & 80 \\
0.4x_1+0.2x_2 & \le & 16 & 1.5y_1+0.2y_2 & \ge & 50 \\
x_1,x_2 & \ge & 0 & y_1,y_2 & \ge & 0
\end{array}$$

V obecnějším zápise (pro názornější pochopení a zapamatování) vypadá dualita úloh v základním tvaru takto:

Primární úloha:

$$\max\ c_1x_1+c_2x_2+\ldots+c_nx_n:$$

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}\cdot\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}\le\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

$$x_1,x_2,\ldots,x_n\ \ge\ 0$$

Duální úloha:

$$\min\ b_1y_1+b_2y_2+\ldots+b_my_m:$$

$$[\,y_1\ \ldots\ y_m\,]\cdot\begin{bmatrix} a_{11} & \ldots & a_{m1} \\ a_{12} & \ldots & a_{m2} \\ \vdots & & \vdots \\ a_{1n} & \ldots & a_{mn} \end{bmatrix}\ge\begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

$$y_1,y_2,\ldots,y_m\ \ge\ 0$$

Rozeberte si sami pro sebe, proč dvojí aplikací duality získáme zpět původní úlohu!

Důležitost duální úlohy při dobré charakterizaci optimálních řešení úloh LP vyplývá z následujících tvrzení o dualitě.

Věta 5.2. (Slabá věta o dualitě LP) *Nechť x^o je přípustným řešením (maximalizační) primární úlohy $\boldsymbol{A} \cdot \vec{x} \leq \vec{b}$, $\vec{x} \geq 0$ a y^o je přípustným řešením (minimalizační) duální úlohy $\vec{y} \cdot \boldsymbol{A} \geq \vec{c}$, $\vec{y} \geq 0$. Pak*

$$\vec{c} \cdot \vec{x}^o \leq \vec{b} \cdot \vec{y}^o \,.$$

`Důkaz`: $\vec{c} \cdot \vec{x}^o \leq (\vec{y}^o \cdot \boldsymbol{A}) \cdot \vec{x}^o = \vec{y}^o \cdot (\boldsymbol{A} \cdot \vec{x}^o) \leq \vec{y}^o \cdot \vec{b} = \vec{b} \cdot \vec{y}^o$. □

Důsledek 5.3. *Máme-li přípustná řešení $\vec{x}^o, \vec{y}^o$ jako ve Větě 5.2 a navíc $\vec{c} \cdot \vec{x}^o = \vec{b} \cdot \vec{y}^o$, pak $\vec{x}^o$ je primárním optimálním řešením a $\vec{y}^o$ duálním optimem.*

`Důkaz` sporem: Nechť existuje lepší řešení $\vec{x}^1$, tj. $\vec{c} \cdot \vec{x}^1 > \vec{c} \cdot \vec{x}^o$. Pak ale $\vec{c} \cdot \vec{x}^1 > \vec{c} \cdot \vec{x}^o = \vec{b} \cdot \vec{y}^o$, z čehož plyne spor s $\vec{c} \cdot \vec{x}^1 \leq \vec{b} \cdot \vec{y}^o$ podle Věty 5.2. □

5.2 Silná dualita

V souvislosti s užitečným Důsledkem 5.3 se přirozeně nabízí otázka, zda vždy nalezneme přípustná řešení primární a duální úlohy o stejných účelových hodnotách. Pochopitelně, pokud primární úloha nemá optimální řešení, ať už z důvodu nepřípustnosti nebo neomezenosti, tak odpovídající duální řešení nenajdeme. Ve všech ostatních případech však ano:

Věta 5.4. (Silná věta o dualitě LP) *Nechť $\vec{x}^o$ je optimálním řešením primární úlohy*

$$\max \ \vec{c} \cdot \vec{x} \text{ pro } \boldsymbol{A} \cdot \vec{x} \leq \vec{b}, \ \vec{x} \geq 0 \,.$$

Pak existuje přípustné řešení $\vec{y}^o$ příslušné duální úlohy takové, že

$$\vec{c} \cdot \vec{x}^o = \vec{b} \cdot \vec{y}^o \,.$$

`Poznámka`: Důsledek 5.3 spolu s Větou 5.4 dává dobrou charakterizaci pro všechny řešitelné úlohy LP – pro důkaz optimality našeho řešení vždy najdeme přípustné duální řešení stejné účelové hodnoty.

`Důkaz`: Fakt, že přípustný vektor $\vec{x}^o$ je optimálním řešením dané primární úlohy lze jinak vyjádřit tvrzením, že neexistuje její přípustné řešení s hodnotou $\vec{c} \cdot \vec{x} \geq \vec{c} \cdot \vec{x}^o + \varepsilon$ pro žádné $\varepsilon > 0$, tedy že rozšířená úloha

$$\begin{pmatrix} \boldsymbol{A} \\ -\vec{c} \end{pmatrix} \cdot \vec{x} \leq \begin{pmatrix} \vec{b} \\ -\vec{c} \cdot \vec{x}^o - \varepsilon \end{pmatrix}, \quad \vec{x} \geq 0$$

nemá žádné přípustné řešení. Nyní aplikujeme (Farkasovo lema) Důsledek 4.8. Podle něj existuje nezáporná kombinace nerovností rozšířené úlohy, která je sporná. Formálně, existuje $\vec{y} \geq 0$ takový, že

$$\vec{y} \cdot \begin{pmatrix} \boldsymbol{A} \\ -\vec{c} \end{pmatrix} \geq 0, \quad \vec{y} \cdot \begin{pmatrix} \vec{b} \\ -\vec{c} \cdot \vec{x}^o - \varepsilon \end{pmatrix} < 0 \,. \tag{5}$$

V prvé řadě si uvědomme, že poslední souřadnice $\vec{y}$ musí být kladná, neboť teprve poslední nerovnost rozšířené soustavy způsobuje její neřešitelnost. Takže lze (po normalizaci) psát $\vec{y} = (\vec{y}^o, 1)$. Rozepsáním vztahů (5) pak dostáváme

$$\vec{y}^o \cdot \boldsymbol{A} \geq \vec{c}, \quad \vec{y}^o \cdot \vec{b} < \vec{c} \cdot \vec{x}^o + \varepsilon \,.$$

Jelikož poslední vztah platí pro libovolně malé $\varepsilon > 0$, limitním přechodem $\varepsilon \to 0$ získáme

$$\vec{y}^o \cdot \boldsymbol{A} \geq \vec{c}, \quad \vec{y}^o \geq 0, \quad \vec{y}^o \cdot \vec{b} \leq \vec{c} \cdot \vec{x}^o \,.$$

Vidíme tedy, že y^o je přípustným řešením příslušné duální úlohy a navíc podle Důsledku 5.3 je $\vec{y}^o \cdot \vec{b} = \vec{c} \cdot \vec{x}^o$. □

Přímým důsledkem pak je následující tvrzení.

Důsledek 5.5. *Primární úloha LP má optimální řešení právě tehdy, když jej má i příslušná duální úloha.*

Doplňkové otázky

(5.2.1) *Sestavte duální úlohu k úloze LP*

$$\begin{aligned} \max\ & x_1 - x_3 : \\ x_1 + x_2 + x_3 &\leq 5 \\ x_1 + x_2 - x_3 &\leq 3 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

(5.2.2) *Sestavte duální úlohu k úloze LP*

$$\begin{aligned} \max\ & x_1 - x_3 : \\ x_1 + x_2 + x_3 &\geq 5 \\ x_1 + x_2 - x_3 &\geq 3 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

(5.2.3) *Co znamená pro primární úlohu fakt, že její duální úloha nemá přípustné řešení?*

*(5.2.4)** *Mohou být primární i duální úloha zároveň bez přípustného řešení?*

*(5.2.5)** *Kde přesně se v důkaze Věty 5.4 použil fakt, že primární úloha má optimální řešení? (Ta místa jsou dvě!)*

5.3 Obecný tvar duality LP

Duální úlohu LP lze pochopitelně sestavit i k úloze LP v obecném tvaru. Pro toto můžeme nakonec použít základní Definici 5.1 a přidat postup podle důkazu Věty 3.5 (nahrazovat vztahy a proměnné dvakrát – před i po duální konstrukci).

`Fakt`: Pro obecný tvar úlohy LP je *duální úloha* schematicky naznačena takto:

$$\begin{aligned} \max\ & \vec{c}\cdot\vec{x} + \vec{d}\cdot\vec{x}' : \\ \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} \cdot \begin{pmatrix} \vec{x} \\ \vec{x}' \end{pmatrix} &\begin{matrix} \leq \\ = \end{matrix} \begin{pmatrix} \vec{a} \\ \vec{b} \end{pmatrix} \\ \vec{x} &\geq 0 \end{aligned} \qquad \begin{aligned} \min\ & \vec{a}\cdot\vec{y} + \vec{b}\cdot\vec{y}' : \\ \begin{pmatrix} \boldsymbol{A}^T & \boldsymbol{C}^T \\ \boldsymbol{B}^T & \boldsymbol{D}^T \end{pmatrix} \cdot \begin{pmatrix} \vec{y} \\ \vec{y}' \end{pmatrix} &\begin{matrix} \geq \\ = \end{matrix} \begin{pmatrix} \vec{c} \\ \vec{d} \end{pmatrix} \\ \vec{y} &\geq 0 \end{aligned}$$

`Komentář`: Jinými slovy, při formulaci obecné duální úlohy přiřadíme nové (duální) proměnné y_j jednotlivým nerovnicím i rovnicím primární úlohy. Přitom duální proměnná odpovídající nerovnosti bude nezáporná, kdežto ta odpovídající rovnosti bude neomezená. Analogicky duální vztahy odpovídající sloupcům nezáporných primárních proměnných budou nerovnostmi, kdežto ty odpovídající neomezeným primárním proměnným budou vyjádřeny jako duální rovnosti. Opět vektory účelový a pravých stran vzájemně zaměníme. Nově získané duální nerovnosti budou v opačném směru a duální účelová funkce se bude minimalizovat.

Doplňkové otázky

(5.3.1) *Sestavte duální úlohu k úloze LP*

$$\begin{aligned} \max\ & x_1 - x_3 : \\ x_1 + x_2 + x_3 &\leq 5 \\ x_1 + x_2 - x_3 &= 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Rozšiřující studium

Teorie duality LP je důležitou součástí optimalizace a má úzký vztah s dobrými charakterizacemi. Náležitou pozornost jí věnuje většina učebnic optimalizace, například [2, Chapter I.5] nebo [3, Oddíl 3.2]. Upozorňujeme, že dualita úloh LP bývá v učebnicích optimalizace zmiňována na různých místech v různých souvislostech.

6 Simplexová metoda: Principy

Úvod

Po předchozí důkladné teoretické přípravě přikročíme (konečně) v této lekci k osvětlení principů, na kterých stojí simplexová metoda pro řešení úloh lineární optimalizace. Náš pohled je především geometrický: Zjednodušeně řečeno, simplexová metoda přechází po hranách polyedru všech přípustných řešení mezi jeho vrcholy, až najde vrchol optimálního řešení.

Zmiňované principy simplexové metody zahrnují přípravu kanonického tvaru úlohy, definici a vysvětlení bázických řešení, jejich vztahu k vrcholům a ukázání geometrických aspektů této metody. Zároveň si zavedeme tzv. simplexovou tabulku, která bude představovat základní datovou strukturu metody, a popíšeme její vlastnosti a interpretaci.

Cíle

Cílem této lekce je uvést a připravit simplexovou metodu pro řešení LP v jejím zjednodušeném podání, tj. se zanedbáním problematiky výchozího řešení a degenerace. Především je zavedena a vysvětlena simplexová tabulka. (Čtenáři, kteří případně přeskočili matematickou teorii předchozích lekcí, by měli opět zbystřit pozornost.)

6.1 Kanonický tvar úlohy

Před použitím simplexové metody musíme nejprve vhodně upravit tvar naší úlohy LP. Stručně řečeno, požadujeme tvar, kdy všechny proměnné jsou nezáporné a všechna další omezení jsou vyjádřena jako rovnosti. (Jak uvidíme, není to na újmu obecnosti.)

`Definice`: Úloha LP je v *kanonickém tvaru*, pokud je vyjádřena jako

$$\begin{aligned} \max\ \vec{c}\cdot\vec{x}\ \text{ pro}& \\ \boldsymbol{A}\cdot\vec{x} &= \vec{b}, \\ \vec{x} &\geq 0\,, \end{aligned}$$

kde matice $\boldsymbol{A}$ má plnou řádkovou hodnost.

Lema 6.1. *Každou úlohu LP lze převést do kanonického tvaru.*

`Důkaz`: Podle Věty 3.5 vyjádříme danou úlohu v základním tvaru

$$\max \vec{c}'\vec{x}'\ \text{ pro }\ \boldsymbol{A}'\cdot\vec{x}'\leq\vec{b},\ \vec{x}'\geq 0\,.$$

Ke každé (i-té) nerovnosti v $\boldsymbol{A}'\cdot\vec{x}\leq\vec{b}$ přidáme tzv. *doplňkovou* proměnnou $x_i''\geq 0$ následovně

$$a'_{i,1}\cdot x'_1+\ldots+a'_{i,n}\cdot x'_n\leq b_i\quad\longmapsto\quad a'_{i,1}\cdot x'_1+\ldots+a'_{i,n}\cdot x'_n+x''_i=b_i$$

Formálně zapsáno (s nulovou cenou doplňkových proměnných v účelovém vektoru)

$$\begin{aligned} \boldsymbol{A} &= (\boldsymbol{A}', \boldsymbol{I})\,, & \vec{b} &= \vec{b}'\,, \\ \vec{x} &= (\vec{x}', \vec{x}'')\,, & \vec{c} &= (\vec{c}', 0, \ldots, 0) \end{aligned}$$

a nový tvar úlohy zní

$$\max\ \vec{c}\cdot\vec{x}\ \text{ pro}$$
$$\boldsymbol{A}\cdot\vec{x} = \vec{b},\ \vec{x} \geq 0\,,$$

jak bylo našim cílem. Jelikož doplňkové proměnné $\vec{x}''$ jsou nezáporné, je snadné vidět jednoznačnou korespondenci mezi řešeními kanonické a základní úlohy. □

`Komentář:` Jak již dobře víme, množinou všech přípustných řešení úlohy LP je polyedr. Převodem na kanonický tvar se nijak podstatně nezmění množina přípustných řešení – bude to "podobný" (kombinatoricky stejný) polyedr jako pro základní tvar úlohy, který bude umístěn ve vyšší dimenzi. Přesněji řečeno, kolmou projekcí nového polyedru na původní proměnné získáme zpětně původní polyedr řešení. Proto tato transformace není na újmu formulaci úlohy.

Zároveň si uvědomme nepatrný význam podmínky plné hodnosti matice $\boldsymbol{A}$ v definici kanonického tvaru. Jak vidíme z důkazu Lematu 6.1, převedená (kanonická) matice $\boldsymbol{A} = (\boldsymbol{A}', \boldsymbol{I})$ přímo obsahuje jednotkovou podmatici plné hodnosti. Pokud by byla úloha náhodou zadána ve tvaru rovností se závislými řádky, přípustné řešení by buď vůbec neexistovalo, nebo by bylo možno závislé řádky vyloučit.

Doplňkové otázky

(6.1.1) *Co když úloha LP již je zadaná výhradně s rovnostmi. Je pak vždy v kanonickém tvaru?*

(6.1.2) *Převeďte do kanonického tvaru úlohu:*

$$\begin{aligned} \max\ x_1 - x_3\ : & \\ x_1 + x_2 + x_3 &\leq 5 \\ x_1 + x_2 - x_3 &= 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

6.2 Vrcholy, báze a bázická řešení

Důležitost vrcholů při řešení úloh LP je ukázána následujícím tvrzením. (Vzpomeňme si, že polyedr má jen konečně mnoho vrcholů, viz Věta 4.12.)

Věta 6.2. *Nechť daná úloha LP má optimální řešení a všechny její proměnné jsou nezáporné. Pak se toto optimální řešení nabývá také v některém krajním bodě (vrcholu) polyedru všech jejích přípustných řešení.*

`Důkaz`: Nechť množinou přípustných řešení je polyedr P, účelovou funkcí je $\vec{c}\cdot\vec{x}$ a optimem je c_o v bodě $\vec{x}^o$. Označme $Q = P \cap \{\vec{x} : \vec{c}\cdot\vec{x} = c_o\}$. Q je podle předpokladu optimality c_o stěnou P. Pokud $|Q| = 1$, jedná se o požadovaný vrchol. Jinak definujeme poloprostor $H^+ = \{\vec{x} : (1,\ldots,1)\cdot\vec{x} \leq 1 + (1,\ldots,1)\cdot\vec{x}^o\}$. Zřejmě je $Q' = Q \cap H^+$ omezený polyedr, a proto podle Lematu 4.13 má Q' nějaký vrchol $\vec{y}^o$ nenáležející facetě určené H^+. Potom $\vec{y}^o$ je vrcholem Q i vrcholem P. □

Ve skutečnosti místo vrcholů budeme při řešení úlohy LP mluvit o tzv. bázických řešeních, která budou hrát klíčovou roli v popisu simplexové metody.

`Definice`: *Bázickým řešením* kanonické úlohy $\boldsymbol{A} \cdot \vec{x} = \vec{b},\ \vec{x} \geq 0$ rozumíme vektor $\vec{x}^o$ splňující $\boldsymbol{A} \cdot \vec{x}^o = \vec{b}$ a mající $\leq m$ nenulových složek (m je počet rovností – řádků $\boldsymbol{A}$), kde nenulové složky $\vec{x}^o$ odpovídají nezávislým sloupcům $\boldsymbol{A}$ (tj. regulární podmatici).

Všimněme si dobře, že definice bázického řešení nevyžaduje nezápornost složek vektoru, proto ne každé bázické řešení je zároveň přípustné. Z elementární lineární algebry snadno plyne:

`Fakt`: Každé čtvercové regulární podmatici $\boldsymbol{A}_1$ v $\boldsymbol{A}$ jednoznačně odpovídá jedno bázické řešení, jehož složky odpovídající sloupcům $\boldsymbol{A}_1$ jsou určeny vztahem $\boldsymbol{A}_1^{-1} \cdot \vec{b}$.

`Značení`: Mějme úlohu $\boldsymbol{A} \cdot \vec{x} = \vec{b}$, $\vec{x} \geq 0$, kde matice $\boldsymbol{A}$ má m řádků a n sloupců. Každé bázické řešení $\vec{x}^o$ je určeno výběrem některé regulární čtvercové podmatice $\boldsymbol{A}_1$ v $\boldsymbol{A}$, neboli výběrem m nezávislých sloupců $\boldsymbol{A}$. Sloupce $\boldsymbol{A}_1$ budeme nazývat *bází řešení* $\vec{x}^o$. Zároveň složky vektoru $\vec{x}$ odpovídající sloupcům $\boldsymbol{A}_1$ budeme nazývat *bázickými proměnnými* a ty zbylé *nebázickými proměnnými*.

`Poznámka`: Pokud bázické řešení $\vec{x}^o$ má právě m nenulových složek, pak je báze pro $\vec{x}^o$ určena jednoznačně. Avšak pokud $\vec{x}^o$ má méně než m nenulových složek, pak všechny různé regulární podmatice pokrývající nenulové složky $\vec{x}^o$ jsou zřejmě bázemi pro totéž řešení $\vec{x}^o$. Takové bázické řešení se nazývá *degenerované*.

Lema 6.3. *Přípustná bázická řešení úlohy LP jednoznačně odpovídají krajním bodům (vrcholům) polyedru všech přípustných řešení.*

`Důkaz`: Mějme vrchol $\vec{v}$ polyedru. Ten je přípustným řešením $\boldsymbol{A} \cdot \vec{v} = \vec{b},\ \vec{v} \geq 0$ z definice. Předpokládejme pro spor, že $\vec{v}$ není bázické, tedy že $\vec{v}$ má více než m kladných složek, nazvěme $\vec{v}^1$ tyto kladné složky $\vec{v}$ a vyberme podmatici $\boldsymbol{A}_1$ složenou ze sloupců $\boldsymbol{A}$ odpovídajících $\vec{v}^1$. Pak soustava $\boldsymbol{A}_1 \cdot \vec{v}^1 = \vec{b}$ má více proměnných než rovnic, a proto množina jejích řešení obsahuje aspoň přímku p procházející $\vec{v}^1 \sim \vec{v}$. Jelikož $\vec{v}^1 > 0$, v dostatečně blízkém okolí $\vec{v}^1$ na p jsou řešení soustavy také kladná, tedy přípustná v naší úloze, a přitom $\vec{v}^1$, potažmo $\vec{v}$, je jejich konvexní kombinací. To je spor, $\vec{v}$ není krajním bodem.

Naopak nechť $\vec{v}$ je přípustným bázickým řešením a $\boldsymbol{A}_1$ je odpovídající báze, tj. regulární podmatice $\boldsymbol{A}$. Pak $\vec{v}$ leží v polyedru. Pokud by $\vec{v}$ bylo v konvexní kombinací, zjednodušeně $\vec{v} = \frac{1}{2}(\vec{u} + \vec{w})$, pak bychom si označili $\vec{v}^1, \vec{u}^1, \vec{w}^1$ příslušné podvektory odpovídající sloupcům báze $\boldsymbol{A}_1$. Vzhledem k jednoznačnosti řešení regulární soustavy rovnic $\boldsymbol{A}_1 \cdot \vec{v}^1 = \vec{b}$; pokud by $\vec{u}^1, \vec{w}^1$ byly také přípustná řešení, některá nebázická složka $\vec{u}$ i $\vec{w}$ by musela být nenulová. Ale jelikož i v této nebázické složce (kde $\vec{v}$ je nulové) platí $\vec{v} = \frac{1}{2}(\vec{u} + \vec{w})$, buď v $\vec{u}$ nebo v $\vec{w}$ by musela pak být tato složka záporná, spor s přípustností. Takže $\vec{v}$ skutečně nelze získat jako konvexní kombinaci přípustných řešení. Z definice je proto $\vec{v}$ vrcholem polyedru. □

`Značení`: Báze $\boldsymbol{A}_1$ a $\boldsymbol{A}_2$ bázických řešení jsou *sousední* pokud se liší právě v jednom sloupci.

`Komentář`: Sousední báze určují bázická řešení v sousedních vrcholech polyedru, nebo totožný vrchol v degenerovaném případě.

Doplňkové otázky

(6.2.1) *Čím a kde je významná podmínka nezápornosti proměnných při hledání optimálního řešení úlohy LP?*

(6.2.2) *Dobrá tedy, ve Větě 6.2 jsme ukázali existenci optimálního řešení ve vrcholu v případě nezáporných proměnných. Kde ale najdeme řešení v úloze s neomezenými proměnnými?*

V takovém případě klidně množina všech přípustných řešení může být celá přímka, která vůbec žádný vrchol nemá. Přitom my musíme řešení hledat jako bázická, tj. jako vrcholy. Zamyslete se a vysvětlete.

6.3 Geometrický princip metody

Nejprve si ukážeme všeobecný popis běhu simplexové metody, přitom pro zjednodušení na chvíli zanedbáme některé technické detaily, které by nyní jen zamlžovaly průzračnost a krásu myšlenky této metody. Popis si doplníme komentáři, které se již vztahují k detailním pojmům uvedeným v následující sekci.

Algoritmus 6.4. ***Princip simplexové metody***
V hrubých rysech algoritmus simplexové metody běží podle následujícího schématu. (Upozorňujeme, že je v tomto zjednodušeném znění zanedbán problém získání výchozího přípustného řešení i problém prevence zacyklení v degenerovaných řešeních.)

1. Začneme v některém (výchozím) přípustném bazickém řešení $\vec{x}^0$ s bází $\boldsymbol{A}_0$ v $\boldsymbol{A}$. Jinými slovy, jsme ve vrcholu $\vec{x}^0$ polyedru přípustných řešení.

   `Komentář:`

   – Jak $\vec{x}^0$ a $\boldsymbol{A}_0$ najdeme? V $\boldsymbol{A}$ vybereme jednotkovou podmatici $\boldsymbol{I} = \boldsymbol{A}_0$ velikosti $m \times m$, případně ji "vyrobíme" přidáním *umělých proměnných*.

   – Pozor na přípustnost výchozího řešení $\vec{x}^0 \geq 0$.

2. Krok i: Pokud žádný sousední vrchol k $\vec{x}^i$ nemá lepší hodnotu účelové funkce, je $\vec{x}^i$ optimální řešení.

   `Komentář:`

   – Pozor, musíme se dívat skutečně na sousední vrcholy, ne jen sousední báze!

   – Tuto situaci poznáme podle nekladných redukovaných cen všech nebázických proměnných, Tvrzení 6.8.

3. Pokud z vrcholu $\vec{x}^i$ vede neomezená hrana polyedru ve směru zlepšující se účelové funkce, optimální řešení neexistuje z *důvodu neomezenosti*.

   `Komentář:`

   – Tuto situaci poznáme podle nekladných všech koeficientů některé nebázické proměnné s kladnou redukovanou cenou, Tvrzení 6.9.

4. K bázi $\boldsymbol{A}_i$ najdeme sousední bázi $\boldsymbol{A}_{i+1}$, která zlepšuje naši účelovou funkci. Pokud vyloučíme degenerovanost, přejdeme tak do sousedního vrcholu $\vec{x}^{i+1}$ s lepší účelovou hodnotou.

   `Komentář:`

   – Zlepšující sousední bázi najdeme takto: Pomocí *sloupcového pravidla* najdeme nebázický sloupec matice $\boldsymbol{A}$, který má do báze vstoupit (třeba ten s největší kladnou redukovanou cenou). Pomocí *řádkového pravidla* pak najdeme bázický sloupec matice $\boldsymbol{A}_i$, který musí bázi opustit.

   – V degenerovaném případě může nastat $\vec{x}^{i+1} = \vec{x}^i$. Pak hrozí nebezpečí zacyklení metody, čemuž zabráníme (třeba) použitím dodatečného *lexikografického pravidla*.

5. Jdeme zpět na bod 2 v iteraci $i + 1$.

Lema 6.5. *Nechť daná úloha LP má přípustné řešení. Pokud vhodnými pravidly výběru sousední báze zabráníme zacyklení v degenerovaném bázickém řešení, tak Algoritmus 6.4 simplexové metody nalezne v konečném počtu kroků optimální řešení úlohy, nebo případně potvrdí neexistenci řešení z důvodu neomezenosti.*

`Důkaz`: Spojením Lematu 6.3 a Věty 6.2 plyne, že pro některý výběr báze v matici dané úlohy LP příslušné bázické řešení nabývá optima. Všech bází je jen konečně mnoho, neboť vybíráme z konečně mnoha sloupců matice úlohy. Dle předpokladu nebudeme opakovat báze stejného degenerovaného řešení a báze různých řešení vždy striktně zlepšují hodnotu účelové funkce. Proto po konečně mnoha krocích algoritmus musí skončit v bodě 2 nebo 3. V bodě 3 máme neomezenou úlohu a v bodě 2 se dostaneme do bázického řešení – vrcholu $\vec{v}^o$, jehož žádný soused nemá lepší hodnotu účelové funkce.

Pro spor v druhém případě předpokládejme, že některý přípustný bod $\vec{x}$ úlohy má lepší hodnotu účelové funkce, tj. $\vec{c}\cdot\vec{x} > \vec{c}\cdot\vec{v}^o$. Pak i pro jakoukoliv jejich vlastní konvexní kombinaci $\vec{y} = \alpha\vec{x} + (1-\alpha)\vec{v}^o$ (vnitřní bod úsečky xv^o) platí $\vec{c}\cdot\alpha\vec{x} + \vec{c}(1-\alpha)\vec{v}^o > \vec{c}\cdot\vec{v}^o$, neboli v bezprostřední blízkosti $\vec{v}^o$ ($\alpha \to 0$) nemůže takové $\vec{y}$ být přípustným řešením podle předpokládané lokální optimality $\vec{v}^o$. To je však ve sporu s konvexností množiny všech přípustných řešení úlohy LP (polyedru). □

`Poznámka`: Třebaže nám předchozí tvrzení zaručuje skončení simplexové metody po konečném počtu kroků, horní odhad tohoto počtu kroků nevychází příliš příznivě – počet kroků je nejvýše rovný počtu všech čtvercových podmatic (bází) dané matice úlohy, což je číslo exponenciální ve velikosti úlohy.

Bohužel se jedná o dosti hluboký problém, neboť přes velkou snahu se doposud nikomu nepodařilo dokázat polynomiální (tedy efektivní) horní odhad počtu kroků simplexové metody. Dokonce pro běžná řádková a sloupcová pravidla jsou známy skutečné příklady exponenciálně dlouhých výpočtů. Přesto je v praktických případech simplexová metoda velice rychlá a většina uživatelů se nad jejím možným dlouhým průběhem ani nezamýšlí.

Doplňkové otázky

6.4 Simplexová tabulka

Pro pohodlný (počítačově-orientovaný) zápis jak úlohy LP, tak i průběhu simplexové metody se nejčastěji používá tzv. simplexová tabulka. Úlohu LP v kanonickém tvaru

$$\begin{aligned} \max\ \vec{c}\cdot\vec{x} \quad &: \\ \boldsymbol{A}\cdot\vec{x} \quad &= \quad \vec{b} \\ \vec{x} \quad &\geq \quad 0 \end{aligned}$$

si přepíšeme do ekvivaletního *redukovaného* zápisu

$$\begin{aligned} \min\ x_0 & \\ x_0 + \vec{c}\cdot\vec{x} \quad &= \quad 0 \\ \boldsymbol{A}\cdot\vec{x} \quad &= \quad \vec{b} \\ \vec{x} \quad &\geq \quad 0\,. \end{aligned} \tag{6}$$

Zde jsme zavedli novou proměnnou x_0 vystupující pouze v novém nultém, tzv. účelovém řádku podmínek úlohy. Všimněte si, že x_0 je vždy jednoznačně určeno hodnotami ostatních proměnných a udává opačnou hodnotu účelové funkce příslušné k řešení $\vec{x}$. Soustava (6) se pak převede do následující formy (na počátku s hodnotou $b_0 = 0$):

`Značení`: Redukovaný zápis kanonické úlohy LP se vyjádří *simplexovou tabulkou*

1	c_1	c_2	c_3	$\dots$	c_n	$-b_0$
0	a_{11}	a_{12}	a_{13}	$\dots$	a_{1n}	b_1
0	a_{21}	a_{22}	a_{23}	$\dots$	a_{2n}	b_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$	$\vdots$
0	a_{m1}	a_{m2}	a_{m3}	$\dots$	a_{mn}	b_m

zapisující koeficienty soustavy lineárních rovnic

$$\begin{pmatrix} 1 & \vec{c} \\ \vec{0} & \boldsymbol{A} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ \vec{x} \end{pmatrix} = \begin{pmatrix} -b_0 \\ \vec{b} \end{pmatrix} . \tag{7}$$

Nultý řádek a sloupec tabulky se nazývají *účelový řádek a sloupec*, přitom účelový sloupec se do tabulky většinou vůbec nezapisuje. Hodnoty v účelovém řádku (mimo nultého a posledního sloupce) se nazývají *redukované ceny* proměnných (složek) $\vec{x}$.

`Komentář:` Vzpomeňme si, že pro každou volbu regulární podmatice $\boldsymbol{A}_1$ plné řádkové hodnosti v matici $\boldsymbol{A}$ úlohy LP máme jednoznačně určeno příslušné bázické řešení úlohy (ne nutně přípustné), jehož bázické proměnné dle elementárních poznatků lineární algebry vyjádříme jako $\vec{x}^B = \boldsymbol{A}_1^{-1} \cdot \vec{b}$. Tento princip přirozeně rozšíříme i na nultý řádek simplexové tabulky, který v redukovaném zápise vlastně také je lineární rovnicí.

Mějme matici $\boldsymbol{C} = \begin{pmatrix} 1 & \vec{c} \\ \vec{0} & \boldsymbol{A} \end{pmatrix}$ a v ní regulární čtvercovou podmatici $\boldsymbol{C}_1$ obsahující sloupce podmatice $\boldsymbol{A}_1$ a navíc nultý sloupec. Pak vztah $\begin{pmatrix} x_0 \\ \vec{x}^B \end{pmatrix} = \boldsymbol{C}_1^{-1} \cdot \begin{pmatrix} -b_0 \\ \vec{b} \end{pmatrix}$ určuje hodnoty bázických proměnných $\vec{x}^B$ v příslušném bázickém řešení (odpovídajícím bázi $\boldsymbol{A}_1$) a zároveň i jeho účelovou hodnotu $-x_0$. Ve spojení s faktem, že standardní řádkové maticové úpravy nemění množinu řešení soustavy (úlohy), dostáváme klíčové pozorování, že řádkovými operacemi na simplexové tabulce můžeme postupně vyjadřovat jednotlivá bázická řešení úlohy LP včetně jejich účelových hodnot.

`Definice`: Simplexová tabulka $\boldsymbol{T}$ je v *jednotkovém tvaru*, pokud v ní je vyznačena jednotková podmatice $\boldsymbol{I}_{m+1}$ obsahující nultý účelový sloupec a vybrané sloupce matice $\boldsymbol{A}$. Zároveň je simplexová tabulka v jednotkovém tvaru *přípustná*, pokud poslední sloupec má mimo nultého řádku nezáporné hodnoty.

Jelikož báze určená jednotkovou podmaticí nám přímo umožňuje číst hodnoty jednotlivých proměnných, z lineární algebry vyplývá:

Tvrzení 6.6. *Mějme pro danou úlohu LP zápis simplexovou tabulkou*

$$\boldsymbol{T} = \left[\begin{array}{cc|c} 1 & \vec{c}\,' & -b_0' \\ \vec{0} & \boldsymbol{A}' & \vec{b}' \end{array}\right]$$

v jednotkovém tvaru. Pak složky vektoru $\vec{b}'$ vyjadřují hodnoty bázických proměnných $\vec{x}^B$ příslušného bázického řešení (odpovídajícího vyznačené jednotkové podmatici v $\boldsymbol{A}'$) a b_0' je účelovou hodnotou tohoto řešení.

`Komentář:` Řekneme-li si, že rozšířenou simplexovou tabulkou rozumíme jednu celou třídu ekvivalence tabulek vzhledem k maticovým řádkovým operacím, pak různé jednotkové tvary rozšířené tabulky "vyobrazují" jednotlivá bázická řešení úlohy. V obecnosti lze dokonce tvrdit následující.

Lema 6.7. *Mějme pro danou úlohu LP zápis simplexovou tabulkou*

$$\boldsymbol{T} = \left[\begin{array}{cc|c} 1 & \vec{c}\,' & -b_0' \\ \vec{0} & \boldsymbol{A}' & \vec{b}' \end{array}\right]$$

v jednotkovém tvaru. Rozdělme si vektor $\vec{x}$ proměnných na bázické složky $\vec{x}^B$ a nebázické $\vec{x}^N$ a podobně pro $\boldsymbol{A}'$ a $\vec{c}'$. Pak pro zvolené nebázické hodnoty $\vec{x}^N \geq 0$ jsou odpovídající bázické proměnné určeny vztahy

$$\vec{x}^B = \vec{b}' - \boldsymbol{A}'^N \cdot \vec{x}^N \tag{8}$$

a účelová hodnota řešení je

$$b'_0 + \vec{c}'^N \cdot \vec{x}^N \,. \tag{9}$$

(Nezapomeňme na dodatečnou podmínku nezápornosti i pro $\vec{x}^B \geq 0$.)

`Důkaz`: Z významu simplexové tabulky (7) plyne přepisem $\boldsymbol{A}'^N \cdot \vec{x}^N + \boldsymbol{A}'^B \cdot \vec{x}^B = \vec{b}'$ a podle předpokladu je $\boldsymbol{A}'^B = I$ jednotková matice. Proto $\boldsymbol{A}'^B \cdot \vec{x}^B = \vec{x}^B$ a (8) platí.

Obdobně je $x_0 + \vec{c}'^N \cdot \vec{x}^N + \vec{c}'^B \cdot \vec{x}^B = -b'_0$ a podle předpokladu je $\vec{c}'^B = 0$ a $-x_0$ je účelovou hodnotou, takže i (9) platí. □

Toto přímočaré tvrzení má dva klíčové důsledky pro chápání významu simplexové tabulky. Z pohledu $\vec{x}^N \geq 0$ na vztah (9) ihned plyne:

Tvrzení 6.8. *Pokud pro danou úlohu LP má zápis simplexovou tabulkou v přípustném jednotkovém tvaru všechny redukované ceny nekladné $\vec{c}' \geq 0$, pak vyjádřené bázické řešení $\vec{x}^B = \vec{b}'$, $\vec{x}^N = 0$ je optimálním řešením úlohy.*

Dále si všimněme, že pokud některá nebázická proměnná $\vec{x}_s$ ve vztahu (8) má všechny koeficienty nekladné, pak pro libovolně velkou hodnotu $\vec{x}_s \to \infty$ se zachová přípustnost řešení $\vec{x}^B \geq 0$. Z toho již s použitím (9) plyne:

Tvrzení 6.9. *Mějme pro danou úlohu LP zápis simplexovou tabulkou v přípustném jednotkovém tvaru. Pokud v některém sloupci s tabulky je redukovaná cena kladná $c'_s > 0$ a zároveň zbytek sloupce je nekladný, tj. $a'_{js} \leq 0$, $j = 1, \ldots, m$, pak optimální řešení úlohy neexistuje z důvodu neomezenosti.*

`Komentář:` Pro dobré pochopení simplexové tabulky je třeba si ujasnit praktickou roli redukovaných cen proměnných v účelovém řádku. Za prvé, redukované ceny bázických proměnných jsou vždy nulové. Pro každou nebázickou proměnnou, dle vztahu (9), její redukovaná cena vyjadřuje změnu výsledné účelové hodnoty při změně hodnoty této proměnné o 1. (Tato změna je celková, již bere do úvahy indukované změny (8) bázických proměnných!)

Na závěr si demonstrujeme sestavení simplexové tabulky na krátkém příkladě.

Příklad 6.10. *Hranolky, slané a paprikové lupínky.*

Firma chce vyrobit hranolky, slané lupínky a paprikové lupínky. K dispozici má přitom 1000 kg brambor a může nakupovat neomezeně sůl za 6 Kč/kg, olej za 30 Kč/kg a paprikové koření za 200 Kč/kg. Bohužel má firma celkem na nákup surovin pouze 5000 Kč. Spotřeby surovin a prodejní ceny na 10 kg jsou následující:

Produkt (10kg)	*Brambory*	*Olej*	*Paprika*	*Sůl*	*Prodejní cena*
hranolky	15	2	0	0.5	450
slané lupínky	20	4	0	1	650
paprikové lupínky	20	3	0.2	0.5	800

Sestavte počáteční simplexovou tabulku úlohy.

Řešení: Ze zadaných hodnot se snadno spočítá zisk a cena surovin na 10 kg produktu:

Produkt (10kg)	*Zisk*	*Cena surovin*
hranolky	387	63
slané lupínky	524	126
paprikové lupínky	667	133

Za proměnné zvolíme vyrobená množství produktů: $10x_1$ hranolků, $10x_2$ slaných lupínků a $10x_3$ paprikových lupínků. Základní tvar úlohy

$$\begin{aligned} \max\ 387x_1 + 524x_2 + 667x_3 : & \\ 15x_1 + 20x_2 + 20x_3 &\leq 1000 \\ 63x_1 + 126x_2 + 133x_3 &\leq 5000 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

převedeme na kanonický tvar přidáním dvou doplňkových proměnných

$$\begin{aligned} \min\ x_0 & \\ x_0 + 387x_1 + 524x_2 + 667x_3 & = 0 \\ 15x_1 + 20x_2 + 20x_3 \quad +x_4 & = 1000 \\ 63x_1 + 126x_2 + 133x_3 \quad +x_5 & = 5000 \\ x_1, x_2, x_3, \quad x_4, x_5 & \geq 0 \end{aligned}$$

a z toho již vypíšeme výchozí simplexovou tabulku (bez nultého řádku)

387	524	667	0	0	0
15	20	20	1	0	1000
63	126	133	0	1	5000

.

Umíte najít řešení této úlohy? □

Doplňkové otázky

Rozšiřující studium

Náš pohled na zápis úlohy LP i zápis průběhu simplexové metody je "tabulkový" (anglicky simplex tableau) a následuje učebnici [3, Kapitola 2]. Podobný pohled na úlohy LP je prezentován i v [7, Chapter I.2]. Poněkud jinak, přes simplexový lexikon (anglicky simplex dictionary) se na úlohy LP dívá jiná klasická učebnice Chvátala [2]. Je třeba zdůraznit, že i přes jiné úhly pohledu se stále jedná o tu stejnou úlohu LP a simplexovou metodu a je snadné mezi různými formami zápisu přecházet. Čtenáři, který by rád získal nadhled na problematikou simplexové metody, bychom doporučovali se seznámit i s pohledem [2].

7 Výpočet simplexové metody

Úvod

V této lekci si ukážeme krok za krokem základní způsob implementace simplexové metody pomocí "pivotování" simplexové tabulky (Oddíl 6.4). To znamená, že od teorie přejdeme úplně k praktickým dovednostem a zkušenostem s řešením úloh LP.

Některé pokročilejší úkony, jako přidávání umělých proměnných pro získání výchozího řešení, nebo použití lexikografického pravidla pro prevenci možného zacyklení, budou doplněny a probrány hlouběji v příští lekci.

Cíle

V návaznosti na teoretickou přípravu předchozích lekcí si prakticky ukážeme a procvičíme zjednodušený výpočet simplexové metody postupem pivotování simplexové tabulky.

7.1 Ilustrační výpočet

Vraťme se k základnímu Příkladu 3.1 s bramborovými lupínky a hranolky. Po vynásobení 10 (jen pro zbavení se necelých čísel) zadání zní:

$$\begin{aligned} \max\ & 80x_1 + 50x_2 & & \\ & 20x_1 + 15x_2 & \leq & \ 1000 \\ & 4x_1 + 2x_2 & \leq & \ 160 \\ & x_1, x_2 & \geq & \ 0 \end{aligned}$$

Tuto snadnu úlohu si nyní vyřešíme v podrobných komentovaných krocích.

- Převedeme úlohu na kanonický tvar

$$\begin{aligned} \max\ & 80x_1 + 50x_2 & & & & \\ & 20x_1 + 15x_2 & +x_3 & & = & \ 1000 \\ & 4x_1 + 2x_2 & & +x_4 = & \ 160 \\ & x_1, x_2, & x_3, x_4 & \geq & \ 0 \end{aligned}$$

- Chceme maximalizovat funkci $80x_1 + 50x_2$, neboli
$$\min\ x_0\ ,\ \text{ kde }\ x_0 + 80x_1 + 50x_2 = 0\,.$$
Máme (s implicitní nezáporností proměnných) tedy soustavu rovnic

$$\begin{aligned} & x_0 + 80x_1 + 50x_2 & & & = & \ 0 \\ & 20x_1 + 15x_2 & +x_3 & & = & \ 1000 \\ & 4x_1 + 2x_2 & & +x_4 = & \ 160\,, \end{aligned}$$

zapsáno maticově

$$\begin{bmatrix} 1 & 80 & 50 & 0 & 0 \\ 0 & 20 & 15 & 1 & 0 \\ 0 & 4 & 2 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x_0 \\ \vec{x} \end{pmatrix} = \begin{bmatrix} 0 \\ 1000 \\ 160 \end{bmatrix}.$$

- V simplexové tabulce (*účelový řádek* proměnné x_0 nahoře) a s vyznačenou jednotkovou bází vypadá naše úloha takto:

1	80	50	**0**	**0**	0
0	20	15	**1**	**0**	1000
0	4	2	**0**	**1**	160

Tato tabulka ukazuje výchozí bázické řešení $x_3 = 1000, x_4 = 160$, $x_1, x_2 = 0$ ($x_0 = 0$), které je (naštěstí) přípustné. Pro jednoduchost už dále nebudeme účelový (nultý) řádek do tabulky zapisovat.

`Komentář:` Pokud soustava výchozí jednotkovou bázi vůbec nemá, nebo ta není přípustná, použijí se dodatečné *umělé proměnné*. Viz příští lekce...

- Přejdeme k sousední bázi, čili vyměníme jeden sloupec ve vyznačené bázi. Přesněji nejprve vybereme nový sloupec do báze pomocí *sloupcového pravidla* a pak vybereme stávající sloupec báze k odebrání pomocí *řádkového pravidla*. Nakonec použijeme běžné maticové operace k tomu, abychom novou bázi ukázali jako jednotkovou.

– Sloupcové pravidlo: Vybereme sloupec s, který má v nultém řádku největší z kladných redukovaných cen. Zde $s = 1$.
– Řádkové pravidlo: Vybereme řádek $i > 0$, který splňuje $a_{is} > 0$ a zároveň dosahuje nejmenší z podílů b_i/a_{is}. (Uvědomme si, že vždy $b_i \geq 0$.) Zde $i = 2$.
– Provedeme *pivot* na prvku $a_{i,s}$ matice: Řádkovými úpravami matice (jako v Gaussově eliminaci) sloupec s upravíme tak, aby obsahoval samé nuly (včetně nultého řádku) mimo $a_{i,s} = 1$.

80	50	**0**	**0**	0
20	15	**1**	**0**	1000
4	2	**0**	**1**	160

$\longrightarrow$

0	10	**0**	−20	−3200
0	5	**1**	−5	200
1	0.5	**0**	0.25	40

`Komentář:` Co nám nová tabulka ukazuje o současném bázickém řešení $\vec{x}$?

$x_0 = -3200 \quad \Rightarrow \quad \vec{c} \cdot \vec{x} = 3200,$
$x_1 = 40,\ x_3 = 200,$
$x_2 = x_4 = 0 \quad$ (protože nejsou v bázi).

- Opět vybereme podle téhož sloupcového pravidla sloupec $s = 2$ a podle řádkového pravidla řádek $i = 1$. Novým pivotem získáme:

0	10	**0**	−20	−3200
0	5	**1**	−5	200
1	0.5	**0**	0.25	40

$\longrightarrow$

0	**0**	−2	−30	−3600
0	**1**	0.2	−1	40
1	**0**	−0.1	0.75	20

Nyní již sloupcové pravidlo nelze použít, neboť redukované ceny v nultém řádku jsou všechny nekladné. Tvrzení 6.8. To znamená, že jsme našli *optimální řešení* $\vec{x}^o$ úlohy

$x_0 = -3600 \quad \Rightarrow \quad \vec{c} \cdot \vec{x}^o = 3600,$
$x_1^o = 40,\ x_2^o = 20,$
$x_3^o = x_4^o = 0 \quad$ (protože nejsou v bázi). □

`Poznámka` k řádkovému pravidlu: Pokud bychom ve sloupci s nenalezli kladnou složku mimo nultý řádek, znamená to, že úloha není omezená, neboli lze získat řešení s libovolně dobrou účelovou funkcí. Tvrzení 6.9. Říkáme, že pak optimální řešení neexistuje z důvodu *neomezenosti*.

Naopak, co by se stalo kdybychom vybrali jiný řádek než s nejmenším podílem b_i/a_{is}? Zkusme si to v předchozí úloze:

0	10	**0**	−20	−3200
0	5	**1**	−5	200
1	0.5	**0**	0.25	40

$\longrightarrow$

−20	**0**	**0**	−25	−4000
−10	**0**	**1**	−7.5	−200
2	**1**	**0**	0.5	80

Jak vidíme, nové bázické řešení obsahuje $x_3 = -200 < 0$, takže není přípustné. Výběr řádku i je volen dle uvedeného řádkového pravidla právě proto, abychom dostali ve všech složkách pravé strany $\vec{b}$ nezáporné, tj. přípustné hodnoty.

7.2 Popis implementace

Před samotným podrobným rozpisem algoritmu simplexové metody si uvedeme ještě jeden důležitý fakt přímo navazující na Tvrzení 6.8,6.9. Z Lematu 6.7(9) plyne, že pokud hodnotu nebázické proměnné x_i s kladnou redukovanou cenou $c_i > 0$ zvětšíme (a změníme hodnoty bázických proměnných odpovídajícím způsobem), tak hodnota účelové funkce řešení stoupne. Důležitým důsledkem je následovné:

Tvrzení 7.1. *Mějme pro danou úlohu LP zápis simplexovou tabulkou v přípustném jednotkovém tvaru. Pokud přejdeme k nové bázi tabulky získané ze stávající báze záměnou některého bázického sloupce sloupcem s kladnou redukovanou cenou, tak získáme buď stejné degenerované bázické řešení nebo řešení se striktně lepší účelovou hodnotou.*

Algoritmus 7.2. ***Implemenace Algoritmu 6.4 simplexové metody***
Následuje jednofázová metoda v zápise simplexovou tabulkou bez prevence zacyklení v degenerovaných řešeních.

0. Úlohu převedeme do základního a pak do kanonického tvaru

$$\max\ \vec{c}\cdot\vec{x} \quad \text{pro} \quad \boldsymbol{A}\cdot\vec{x}\ =\ \vec{b},\quad \vec{x}\geq 0$$

přidáním *doplňkových proměnných* ke každé nerovnici (Lemma 6.1). Dále přidáme pro redukovaný zápis novou *účelovou* proměnnou x_0 vztahem

$$\min\ x_0 \quad \text{pro} \quad x_0+\vec{c}\cdot\vec{x}=0\,.$$

1. Zapíšeme maticově předchozí vztahy a odpovídající *simplexovou tabulkou*

$$\begin{pmatrix}1 & \vec{c}\\ \vec{0} & \boldsymbol{A}\end{pmatrix}\cdot\begin{pmatrix}x_0\\ \vec{x}\end{pmatrix}=\begin{pmatrix}0\\ \vec{b}\end{pmatrix} \qquad\longrightarrow\qquad \left[\begin{array}{c|c}1 & \vec{c}\\ \vec{0} & \boldsymbol{A}\end{array}\right|\left.\begin{array}{c}0\\ \vec{b}\end{array}\right].$$

Horní (nultý) řádek tabulky nazýváme *účelovým řádkem*, jeho prvky jsou *redukované ceny proměnných*, sloupec $\vec{b}$ nazýváme *pravou stranou* a zbytek tabulky nazýváme *levou stranou*. V dalším textu již budeme tabulku zapisovat bez nultého účelového sloupce.

Dále získáme *výchozí přípustné bázické řešení*. K tomu potřebujeme tabulku v přípustném jednotkovém tvaru. Často tabulka již v takovém tvaru je (doplňkové proměnné nám přidají jednotkovou podmatici a pravé strany obvykle bývají kladné), ale jinak uplatníme následující postup:

a) Všechny rovnice se zápornou pravou stranou vynásobíme -1. (Aby bylo výchozí bázické řešení přípustné.)

b) Pro každý chybějící jednotkový vektor $\vec{e}^{\,i}$ v tabulce na levé straně přidáme *umělou proměnnou* $u_i\geq 0$ s cenou $-\nu$, kde ν je velmi velká konstanta, formálně $\nu=\infty$. Zapíšeme vzniklou výchozí ("umělou") tabulku v jednotkovém tvaru

$$\left[\begin{array}{cccc|c}\vec{c} & \dots & 0 & -\nu & 0\\ \hline \multicolumn{4}{c|}{\underbrace{\boldsymbol{A}'\ \big|\ \boldsymbol{I}}_{\boldsymbol{A}^u}} & \vec{b}'\end{array}\right],$$

která vyjadřuje vztahy:

$$\begin{aligned} x_0+\vec{c}\cdot\vec{x}-\nu\cdot\vec{u} &= 0\\ \boldsymbol{A}^u\cdot(\vec{x},\vec{u})^{\,T} &= \vec{b}\\ \vec{x},\vec{u} &\geq 0\end{aligned}$$

c) Upravíme tabulku tak, aby i nultý účelový řádek obsahoval redukované ceny 0 ve sloupcích jednotkové podmatice $\boldsymbol{I}$: Přičteme násobky řádků příslušných k jednotlivým jednotkovým vektorům $\boldsymbol{I}$ k účelovému řádku.

`Komentář:` Všimněme si, že úpravou (c) se naše konstanta ν objeví v redukovaných cenách některých nebázických proměnných zároveň s jejich původními cenami. Z toho vyplývá zásadní problém praktické implementace – zaokrouhlovací chyby vynucené velkou konstantou ν mohou "vymazat" původní ceny proměnných a zcela tak změnit řešení úlohy.

Mimo uvedený přístup k umělým proměnným oceněním $-\nu$ si ještě v příští přednášce uvedeme tzv. dvoufázovou simplexovou metodu, která se nejprve "zbaví" umělých proměnných a teprve pak řeší původní úlohu.

2. Jsou-li všechna čísla v účelovém řádku tabulky nekladná, máme optimální řešení, viz Tvrzení 6.8. (Nekladné redukované ceny všech proměnných znamenají, že zvětšováním žádné z nebázických proměnných již nelze zvýšit účelovou funkci.)
Pokud však je v bázi stále ještě zůstává některá umělá proměnná, původní úloha nemá *žádné přípustné řešení*.

3. Je-li v tabulce sloupec s takový, že $c_s = a_{0s} > 0$ a $a_{is} \leq 0$ pro všechna $i > 0$, úloha nemá optimální řešení z *důvodu neomezenosti*.

4. Přejdeme k sousední bázi:

a) *Sloupcové pravidlo* vybere sloupce s s nejvyšší kladnou hodnotou redukované ceny v účelovém řádku, tj. ve směru nejvyššího přírustku účelové funkce.

b) *Řádkové pravidlo* vybere řádek $i > 0$ s nejmenší hodnotou podílu b_i/a_{is} pro $a_{is} > 0$, což je nutné pro zachování nezápornosti pravé strany.

c) Přejdeme k *sousední bázi* aplikací tzv. *pivotu* na prvku a_{is}: Pivotace prvku je vlastně jednou fází dobře známé Gaussovy eliminace v matici, která řádkovými operacemi "vytvoří" jednotkový (sloupcový) vektor na této pozici.

- Pro každé $j \neq i$, od j-tého řádku odečteme a_{js}/a_{is}-násobek i-tého,
- i-tý řádek vydělíme číslem a_{is}.

(Nyní v nové bázi sloupec s nahradí sloupec původního jednotkového vektoru $\vec{e}^{\,i}$.)

Jak jsme již výše zmínili, aplikací řádkových maticových operací na simplexové tabulce zřejmě nezměníme zapsanou optimalizační úlohu, a nová báze tabulky jistě nemá horší účelovou hodnotu podle Tvrzení 7.1.

5. Vrátíme se v cyklu do bodu **2**.

`Komentář:` Výše uvedený popis Algoritmu 7.2 ponechává jistý stupeň nedeterminismu jeho běhu. Za prvé není určeno, který sloupec vybrat v bodě 4 (a), pokud více sloupců dosahuje nejvyšší hodnoty redukované ceny. Nejedná se však o kritický problém, prostě si vybereme libovolný z nich. (Dokonce by bylo možné algoritmus zobecnit tak, aby mohl vybírat kterýkoliv sloupec s kladnou redukovanou cenou.)

Za druhé v bodě 4 (b) může být více řádků i se stejnou minimální hodnotou b_i/a_{is}. (Toto typicky nastává v degenerovaných bázických řešeních.) Zde se již jedná o kritický problém, neboť nevhodná volba mezi těmito řádky může způsobit nekonečné zacyklení Algoritmu 7.2 v degenerovaném řešení. Blíže o tomto problému a jeho řešení pojednáme v Části 8.2.

Doplňkové otázky

(7.2.1) *Jak máme volit "velmi velkou" konstantu ν ve výše popsané implementaci umělých proměnných v rámci reálné počítačové aritmetiky (řekněme typ 'double')? Je možné volit třeba $1e50$?*

(7.2.2) *Jak implementace "velmi velké" konstantu ν ovlivní numerickou přesnost simplexové metody?*

7.3 Různé příklady výpočtů

Příklad 7.3. *Dokončení výpočtu Příkladu 6.10.*

$$\begin{aligned} \max\ & 387x_1 + 524x_2 + 667x_3 & & \\ & 15x_1 + 20x_2 + 20x_3 & \leq & \ 1000 \\ & 63x_1 + 126x_2 + 133x_3 & \leq & \ 5000 \\ & x_1, x_2, x_3 & \geq & \ 0 \end{aligned}$$

Zmíněný příklad vedl na následující výchozí simplexovou tabulku, která je v přípustném jednotkovém tvaru.

387	524	667	0	0	0
15	20	20	**1**	**0**	1000
63	126	133	**0**	**1**	5000

Jsme v Algoritmu 7.2 v bodě 4. Užitím sloupcového pravidla vybereme třetí sloupec pro vstup do báze a pak řádkovým pravidlem druhý řádek pro opuštění báze. Výsledkem je:

71.05	−107.9	0	0	−5.015	−25075
5.526	1.053	0	1	−0.15	248.1
0.4737	0.9474	1	0	0.00752	37.6

V další iteraci vybereme první sloupec a první řádek a výsledkem je tabulka:

0	−120.9	0	−12.84	−3.08	−28265
1	0.19	0	0.18	−0.027	44.88
0	0.8571	1	−0.0857	−0.02	16.33

Z poslední tabulky vidíme (viz Algoritmus 7.2 v bodě 2), že optimálním řešením je výroba (zhruba) 448.8 kg hranolků, 0 kg slaných lupínků a 163.3 kg paprikových lupínků se ziskem 28265 Kč. □

Příklad 7.4. *Ukázka výpočtu neomezené úlohy LP:*

$$\begin{aligned} \max\ & x_1 + x_2 + x_3 & & \\ & -x_1 - x_2 + 2x_3 & \leq & \ 2 \\ & -x_1 + 2x_2 - x_3 & \leq & \ 4 \\ & 2x_1 - x_2 - x_3 & \leq & \ 6 \\ & x_1, x_2, x_3 & \geq & \ 0 \end{aligned}$$

Již dobře známým postupem přes kanonický tvar úlohy zapíšeme výchozí simplexovou tabulku v jednotkovém přípustném tvaru

1	1	1	0	0	0	0
-1	-1	2	**1**	**0**	**0**	2
-1	2	-1	**0**	**1**	**0**	4
2	-1	-1	**0**	**0**	**1**	6

.

Dále postupujeme v intencích Algoritmu 7.2 následovně:

0	1.5	1.5	0	0	-0.5	-3
0	-1.5	1.5	1	0	0.5	5
0	1.5	-1.5	0	1	0.5	7
1	-0.5	-0.5	0	0	0.5	3

0	0	**3**	0	-1	-1	-10
0	0	0	1	1	1	12
0	1	-1	0	0.66	0.33	4.66
1	0	-1	0	0.33	0.66	5.33

Dobře si nyní všimněme poslední tabulky. V ní se stále ještě neuplatní bod 2 Algoritmu 7.2, stále nemáme optimální řešení díky kladné redukované ceně ve třetím sloupci. Poprvé mezi našimi příklady však vidíme uplatnění bodu 3 algoritmu – ve třetím sloupci jsou všechny další koeficienty nekladné, takže optimální řešení naší úlohy neexistuje z důvodu neomezenosti. (Čtenář si sám může nezávisle zkontrolovat, že volbou $x_1 = x_2 = x_3 = t$, $t \to \infty$ získá libovolně dobré přípustné řešení úlohy.) □

Příklad 7.5. *Vyřešme náš starý známý Příklad 3.1 o lupíncích a hranolcích s dodatečným požadavkem na výrobu alespoň 30 kg lupínků.*

Podmínky jsou zapsány

$$\begin{aligned} \max\ & 80x_1 + 50x_2 & & \\ & 20x_1 + 15x_2 & \leq\ & 1000 \\ & 4x_1 + 2x_2 & \leq\ & 160 \\ & x_1 & \geq\ & 30\,. \end{aligned}$$

V kanonickém tvaru pak úloha zní

$$\begin{aligned} \max\ & 80x_1 + 50x_2 & & & & & & \\ & 20x_1 + 15x_2 & +x_3 & & & =\ & 1000 \\ & 4x_1 + 2x_2 & & +x_4 & & =\ & 160 \\ & x_1 & & & -x_5 & =\ & 30 \\ & x_1, x_2, & & x_3, x_4, x_5 & \geq\ & & 0\,, \end{aligned}$$

což bohužel nedává tabulku v jednotkovém tvaru. Jsme Algoritmu 7.2 v bodě 1 (a,b). (Pokud by čtenáře napadlo obrátit znaménka poslední rovnosti, tak by získaná tabulka zase nebyla přípustná na posledním řádku.) Budeme proto muset přidat jednu umělou proměnnou $u = x_6$ následovně:

$$\begin{aligned} \max\ & 80x_1 + 50x_2 & & & -\nu\, x_6 & & \\ & 20x_1 + 15x_2 & +x_3 & & & =\ & 1000 \\ & 4x_1 + 2x_2 & & +x_4 & & =\ & 160 \\ & x_1 & & & -x_5 + x_6 & =\ & 30 \\ & x_1, x_2, & & x_3, x_4, x_5, x_6 & & \geq\ & 0\,, \end{aligned}$$

Zavedení umělé proměnné x_6 pochopitelně mění–rozšiřuje množinu přípustných řešení úlohy, takže v konečném důsledku musíme vliv x_6 z úlohy vyloučit. Toho dosáhneme určením "obrovské" záporné ceny $-\nu \sim -\infty$ pro x_6.

Nyní již můžeme sestavit výchozí tabulku v přípustném jednotkovém tvaru. Nezapomeňme však podle Algoritmu 7.2 v bodě 1 (c) nejprve upravit nenulovou hodnotu redukované ceny x_6 přičtením vhodného násobku posledního řádku.

80	50	0	0	0	$-\nu$	0
20	15	1	0	0	0	1000
4	2	0	1	0	0	160
1	0	0	0	-1	1	30

→

$80+\nu$	50	0	0	$-\nu$	0	30
20	15	**1**	**0**	0	**0**	1000
4	2	**0**	**1**	0	**0**	160
1	0	**0**	**0**	-1	**1**	30

Dále již postupujeme běžným způsobem simplexové metody.

0	50	0	0	80	$-80-\nu$	-2400
0	15	**1**	**0**	20	-20	400
0	2	**0**	**1**	4	-4	40
1	0	**0**	**0**	-1	1	30

0	0	0	-25	-20	$100-\nu$	-3400
0	**0**	**1**	-7.5	-10	10	100
0	**1**	**0**	0.5	2	-2	20
1	**0**	**0**	0	-1	1	30

Z poslední tabulky vidíme optimální řešení – vyrobit $x_1 = 30\,\text{kg}$ lupínků a $x_2 = 20\,\text{kg}$ hranolků. Hodnota $x_3 > 0$ nám říká, že v první nerovnosti úlohy ještě zbývá do rovnosti rezerva x_3, neboli v konkrétní formulaci nám zbývá 10 kg brambor. □

Doplňkové otázky

(7.3.1) *Vyřešte popsanou metodou (ručně) úlohy Lekce 6. Ve kterých budete potřebovat umělé proměnné?*

Rozšiřující studium

Co se týče rozšiřujícího studia, platí i zde poznámky z konce Lekce 6. Dále bychom chtěli explicitně připomenout, že simplexová metoda, zde prezentovaná ve své základní verzi, má mnoho variant (využívajících i duální úlohu LP) vhodných v různých podmínkách použití. Mimo jiné byla a jsou zkoumána různá pivotní pravidla, tj. způsoby výběru prvku tabulky k příštímu pivotu (u nás zastoupeno jen základním sloupcovým a řádkovým pravidlem).

8 Podrobnosti a variace metody

Úvod

Dostáváme se k poslední lekci našeho textu týkající se lineárního programování, která pojednává o třech doposud opomíjených (a problematických) detailech simplexové metody:

- *Jak efektivně získat výchozí přípustné bázické řešení úlohy (třeba pomocí umělých proměnných).*
- *Jak účinně zamezit zacyklení simplexové metody v degenerovaných bázických řešeních (například tzv. lexikografické pravidlo).*
- *Jak odhadnout celkový počet iterací simplexové metody.*

Zatímco pro první dva body si ukážeme dostačující řešení, ten třetí je už po dlouhá léta těžkým teoretickým problémem v optimalizaci a uspokojující odpověď na něj není známa.

Cíle

Ukážeme si v detailech, jak se u výchozího bázického řešení zbavíme umělých proměnných pomocí tzv. dvoufázové simplexové metody (která je dobře prakticky použitelná). Dále si zavedeme lexikografickou simplexovou metodu jako variantu dříve popsaného obecného algoritmu, která spolehlivě zabrání zacyklení v degenerovaných vrcholech. Nakonec stručně zmíníme problém extrémně dlouhých výpočtů simplexové metody.

8.1 Umělé proměnné; dvě fáze

Jak bylo popsáno v Algoritmu 7.2, umělé proměnné přidáváme do výchozí tabulky, abychom snadno získali výchozí přípustné bázické řešení. Pochopitelně však požadujeme, aby ve výsledném řešení měly tyto umělé proměnné nulovou hodnotu.

K odstranění umělých proměnných se přirozeně nabízejí dva postupy:

- *"Nekonečná" cena:*

  Umělým proměnným $\vec{u}$ přiřadíme cenu $-\nu$, kde ν je velmi velké číslo (nepřesně bychom mohli zapsat, že $\nu = \infty$). Pokud některá umělá proměnná vyjde ve výsledném řešení větší než nula, pak původní úloha nemá přípustné řešení.

  `Komentář:` Pozor, nezapomeňme, že ceny $-\nu$ v nultém řádku se musíme zbavit už ve výchozí tabulce, a to přičtením ν-násobku i-tého řádku k nultému řádku.

- *Dvoufázová simplexová metoda:*

  Ve dvoufázové metodě všem umělým proměnným $\vec{u}$ nejprve přiřadíme cenu -1 a původním proměnným cenu 0. (Součet všech umělých proměnných tak minimalizujeme.) Jestliže optimum této *první fáze* vyjde nenulové, původní úloha nemá přípustné řešení. Naopak pro nulové optimum umělé proměnné následně vypustíme a pokračujeme *druhou fází* v řešení původní úlohy.

Dvoufázová metoda

Druhý popsaný způsob odstranění umělých proměnných si nyní rozepíšeme.

Algoritmus 8.1. ***Implementace dvoufázové simplexové metody***
Dvoufázová simplexová metoda (zapsaná simplexovou tabulkou) je založena na postupu Algoritmu 6.4 jednofázové metody s následujícími vylepšeními.

0. Vyjdeme z (už známého) kanonického tvaru úlohy v redukovaném zápise:

$$\begin{aligned} \min\ & x_0 \\ x_0 + \vec{c}\cdot\vec{x} &= 0 \\ \boldsymbol{A}\cdot\vec{x} &= \vec{b} \\ \vec{x} &\geq 0 \end{aligned}$$

1. Jako v Algoritmu 7.2, bod 1, po případném přidání umělých proměnných zapíšeme výchozí simplexovou tabulku v jednotkovém tvaru

$$\left[\begin{array}{cc|c} \vec{c} & 0\ldots0 & 0 \\ \hline \underbrace{\boldsymbol{A}' \mid \boldsymbol{I}}_{\boldsymbol{A}^u} & & \vec{b}' \end{array}\right],$$

která vyjadřuje vztahy

$$\begin{aligned} x_0 + \vec{c}\cdot\vec{x} &= 0 \\ \boldsymbol{A}^u\cdot(\vec{x},\vec{u})^T &= \vec{b}' \\ \vec{x},\vec{u} &\geq 0\,. \end{aligned}$$

Nyní však pro *první fázi* k ocenění umělých proměnných zavedeme novou účelovou funkci a pro $\vec{c}^u = (1, \vec{c}, \vec{0})$ novou úlohu zapíšeme jako

$$\begin{aligned} \max \quad & -u_1 - \ldots - u_k & & \\ & \vec{c}^u \cdot (x_0, \vec{x}, \vec{u})^T & = & \ 0 \\ & \boldsymbol{A}^u \cdot (\vec{x}, \vec{u})^T & = & \ \vec{b}' \\ & \vec{x}, \vec{u} & \geq & \ 0\,. \end{aligned}$$

V zápise simplexovou tabulkou tato úloha zní

$$\left[\begin{array}{c|c} 0 \ldots 0 \quad -1 \ldots -1 & 0 \\ \hline \vec{c} \quad 0 \ \ldots \ 0 & 0 \\ \hline \boldsymbol{A}^u & \vec{b}' \end{array}\right]. \tag{10}$$

Nakonec ještě musíme tabulku upravit tak, aby účelový řádek obsahoval 0 ve sloupcích umělých proměnných.

`Komentář:` Dobře si všimněte, že výchozí simplexová tabulka (10) dvoufázové metody obsahuje dva účelové řádky – jeden pro první fázi metody (kdy se minimalizují umělé proměnné před jejich vyloučením) a druhý, vlastně původní, účelový řádek pro pozdější druhou fázi. Původní účelový řádek je sice zbytečný pro výpočet první fáze, ale musíme jej upravovat spolu se zbytkem tabulky, aby na začátku druhé fáze obsahoval správné redukované ceny pro původní úlohu.

Během první fáze se tak na původní účelový řádek díváme jako na další omezující podmínku úlohy (která jen definuje hodnotu $-x_0$ původní účelové funkce), ale řádek této podmínky se nemůže účastnit výběru prvku pro pivotování ze zřejmých důvodů. Tabulkové úpravy tohoto řádku spolu se zbylými podmínkami nám zaručují, že v každém bázickém řešení vyjádřeném naší tabulkou bude původní účelový řádek udávat správné redukované ceny i účelovou hodnotu.

2.–5. Postupujeme v těchto bodech podle Algoritmu 6.4, ale máme na paměti, že i původní účelový řádek je vyloučen z působnosti řádkového pravidla, třebaže jinak je pokládán za běžný řádek tabulky.

`Komentář:` Je jasné, že vzhledem k podmínce $\vec{u} \geq 0$ nemůže být optimalizace $\max -u_1 - \ldots - u_k$ neomezená. Ve skutečnosti předpokládáme optimální řešení první fáze jako $\vec{u} = \vec{0}$. Takové řešení nám umožní všechny umělé proměnné následně vypustit.

6. Pokud optimální řešení první fáze má kladnou hodnotu (tedy $\vec{u} \neq \vec{0}$), pak původní úloha nemá žádné přípustné řešení.

7. Jinak z výsledné tabulky první fáze vypustíme umělý účelový řádek i všechny sloupce umělých proměnných. Pokračujeme ve výpočtu *druhé fáze* metody už známým způsobem od bodu **2** Algoritmu 7.2.

Věta 8.2. *Pokud první fáze Algoritmu 8.1 skončí s optimálním řešením s hodnotou* 0*, pak po vypuštění umělých proměnných v bodě 7 získáme správnou výchozí simlexovou tabulku v jednotkovém tvaru pro původní úlohu LP. Pokud naopak první fáze skončí s řešením s kladnou hodnotou, pak původní úloha LP nemá žádné přípustné řešení.*

`Důkaz`:
.....
□

Důsledek 8.3. *Algoritmus 8.1 správně řeší úlohy LP.*

Pro lepší pochopení dvoufázové simplexové metody je nejlepší se podívat na malý názorný příklad jejího použití.

Příklad 8.4. *Vyřešme Příklad 7.5 za pomocí dvoufázové simplexové metody.*

Připomeneme, že v kanonickém tvaru úloha zní

$$\begin{array}{rrrrrrrl} \max & 80x_1 & + 50x_2 & & & & & \\ & 20x_1 & + 15x_2 & +x_3 & & & = & 1000 \\ & 4x_1 & + 2x_2 & & +x_4 & & = & 160 \\ & x_1 & & & & -x_5 & = & 30 \\ & & x_1, x_2, & & x_3, x_4, x_5 & \geq & 0\,, & \end{array}$$

což bohužel nedává tabulku v jednotkovém tvaru. Opět tak budeme muset nejprve zavést umělou proměnnou do poslední rovnice.

Podstatou dvoufázové simplexové metody je, že optimalizace probíhá ve dvou stupních, nejprve vyloučením umělých proměnných a pak teprve původní účelovou funkcí. Proto nejprve musíme původní účelovou funkci převést na rovnost s x_0 v redukovaném tvaru.

$$\begin{array}{rrrrrrrl} & -x_0 + 80x_1 & + 50x_2 & & & & = & 0 \\ & 20x_1 & + 15x_2 & +x_3 & & & = & 1000 \\ & 4x_1 & + 2x_2 & & +x_4 & & = & 160 \\ & x_1 & & & & -x_5 & = & 30 \\ & & x_1, x_2, & & x_3, x_4, x_5 & \geq & 0\,, & \end{array}$$

Nyní je čas zavést umělou proměnnou x_6, jejíž hodnotu budeme v první fázi minimalizovat.

$$\begin{array}{rrrrrrrl} & & & & \max & -x_6 & & \\ & -x_0 + 80x_1 & + 50x_2 & & & & = & 0 \\ & 20x_1 & + 15x_2 & +x_3 & & & = & 1000 \\ & 4x_1 & + 2x_2 & & +x_4 & & = & 160 \\ & x_1 & & & & -x_5 + x_6 & = & 30 \\ & & x_1, x_2, & & x_3, x_4, x_5, x_6 & \geq & 0\,, & \end{array}$$

Z poslední formulace teď sestavíme výchozí simplexovou tabulku, která již po úpravě nultého řádku bude v jednotkovém přípustném tvaru.

0	0	0	0	0	**-1**	0
80	50	0	0	0	0	0
20	15	1	0	0	0	1000
4	2	0	1	0	0	160
1	0	0	0	-1	1	30

$\longrightarrow$

1	0	0	0	-1	0	30
80	50	0	0	0	0	0
20	15	**1**	**0**	0	**0**	1000
4	2	**0**	**1**	0	**0**	160
1	0	**0**	**0**	-1	**1**	30

Všimněme si dobře, že naše tabulka obsahuje dva účelové řádky. To je přirozené, neboť si potřebujeme pamatovat původní účelovou funkci, tj. vztah proměnné x_0, i novou účelovou funkci max $-x_6$. Zároveň by v plném zápise tabulky měly být nalevo přítomny příslušné dva účelové sloupce, ale ty, jak známo, pro jednoduchost nevypisujeme.

Dále budeme postupovat pivotováním stejně jako v Algoritmu 7.2. Účelový řádek proměnné x_0 budeme zpracovávat stejně jako jiné řádky tabulky, jen jej vyjmeme z

působnosti řádkového pravidla a z požadavku přípustnosti pravé strany. To znamená, že ve výchozí tabulce vybereme podle kroku 4 algoritmu první sloupec a z něj poslední řádek (třebaže řádkové pravidlo by jinak vybíralo účelový řádek x_0).

1	0	0	0	-1	0	30
80	50	0	0	0	0	0
20	15	**1**	**0**	0	**0**	1000
4	2	**0**	**1**	0	**0**	160
1	0	**0**	**0**	-1	**1**	30

0	0	0	0	0	-1	0
0	50	0	0	80	-80	-2400
0	15	1	0	20	-20	400
0	2	0	1	4	-4	40
1	0	0	0	-1	1	30

Jak vidíme z poslední tabulky, dosáhli jsme optimálního řešení v první fázi metody, tj. máme minimální přípustnou hodnotu umělé proměnné x_6. Velmi důležité přitom je, že skutečně $x_6 = 0$ a my dále můžeme tuto umělou proměnnou z úlohy vyloučit. (Pokud by se stalo, že v optimálním řešení první fáze má některá umělá proměnná kladnou hodnotu, pak původní úloha nemá žádné přípustné řešení.)

Dalším důležitým bodem je role druhého účelového řádku, tj. vztahu proměnné x_0. Tím, že jsme na něj aplikovali stejné operace jako na celou tabulku, máme zaručeno, že jím vyjádřená hodnota $-x_0$ skutečně udává účelovou hodnotu aktuálního bázického řešení v původní úloze. Proto vymazáním nultého řádku a sloupce umělé proměnné x_6 z poslední tabulky získáme správnou výchozí tabulku původní úlohy v přípustném jednotkovém tvaru. Dále již postupujeme známým způsobem.

0	50	0	0	80	-2400
0	15	**1**	**0**	20	400
0	2	**0**	**1**	4	40
1	0	**0**	**0**	-1	30

0	10	0	-20	0	-3200
0	5	1	-5	0	200
0	0.5	0	0.25	1	10
1	0.5	0	0.25	0	40

0	0	0	-25	-20	-3400
0	0	1	-7.5	-10	100
0	1	0	0.5	2	20
1	0	0	0	-1	30

Výsledek výpočtu je $x_1 = 30$, $x_2 = 20$, $x_3 = 100$, $x_4 = x_5 = 0$. Účelová funkce má hodnotu 3400. Porovnejte si tento výsledek s výsledkem v Příkladu 7.5. □

Doplňkové otázky

(8.1.1) *Vyřešte úlohy z Lekce 7 obsahující umělé proměnné pomocí dvoufázové metody.*

(8.1.2)

8.2 Degenerace a prevence zacyklení

Jak plyne z definice, degenerovaná řešení se v průběhu simplexové metody objevují, pokud pravá strana tabulky obsahuje 0. Pak nová báze získaná iterací simplexové metody může odpovídat totožnému řešení (vrcholu). Jak pak ale u degenerovaných řešení zabránit zacyklení bází ve stejném vrcholu?

Příklad 8.5. *Ukázka zacyklení Algoritmu 7.2 simplexové metody na úloze LP.*

Použijme k řešení následující úlohy LP Algoritmus 7.2 s tím, že při nejednoznačné volbě sloupcového a řádkového pravidla vybereme podle nejnižšího indexu.

$$\begin{array}{rrrrrrrrl} \max & -20x_1 & +53x_2 & +41x_3 & -204x_4 & & & & \\ & 2x_1 & -11x_2 & -5x_3 & +18x_4 & +x_5 & & = & 0 \\ & -x_1 & +4x_2 & +2x_3 & -8x_4 & & +x_6 & = & 0 \\ & -2x_1 & +11x_2 & +5x_3 & -18x_4 & & +x_7 & = & 1 \\ & & & & x_1, x_2, x_3, x_4 & & x_5, x_6, x_7 & \geq & 0 \end{array}$$

Čtenář nechť si sám zkusí spočítat několik iterací simplexové metody podle popsané implementace. (Po šesté iteraci uvidí, že získal zpět jednu z předchozích tabulek.) □

Definice 8.6. ***Lexikografické porovnání vektorů*** ("zprava"):
Vektor $\vec{x}$ je lexikograficky menší než $\vec{y}$, píšeme $\vec{x} \prec \vec{y}$, pokud

$$\exists i : x_i < y_i \ \wedge \ \forall j > i : x_j = y_j \, .$$

Vektor $\vec{x}$ je *lexikograficky kladný*, pokud $\vec{x} \succ \vec{0}$.

`Komentář:` Lexikografické porovnávání již asi čtenář zná jako způsob řazení slov ve slovníku (lexikon) – jednotlivá písmena slov odpovídají složkám vektorů. Je třeba si však uvědomit, že naše definice porovnává zprava, tedy od konců vektorů.

Význam lexikografického porovnání vektorů pro vylepšení Algoritmu 7.2 simplexové metody tkví v následujících poznatcích.

Tvrzení 8.7. *Mějme simplexovou tabulku, jejíž každý řádek mimo účelového je lexikograficky kladný. Zvolme její libovolný sloupec s s kladnou redukovanou cenou.*

a) Je-li v tabulce $a_{i,s} > 0$, pak aplikace pivotu na $a_{i,s}$ lexikograficky zmenší vektor účelového řádku.

b) Je-li $i > 0$ index řádku tabulky s lexikograficky nejmenším vektorem $\frac{1}{a_{i,s}}(\vec{a}_i, b_i)$, pak po aplikaci pivotu na $a_{i,s}$ zůstane každý řádek mimo účelového lexikograficky kladný.

`Důkaz`: Oba závěry přímočaře plynou z definice lexikografického uspořádání a z významu pivotu. Detaily ponecháme čtenáři. □

Algoritmus 8.8. ***Lexikografické simplexové metody***
Algoritmus 7.2 simplexové metody zpřesníme v následujících dvou bodech.

- V kroku 1 u výchozí tabulky přesuneme sloupce vybrané jednotkové podmatice na konec. (Tím zajistíme lexikografickou kladnost výchozí tabulky.)
- V kroku 4.b řádkového pravidla vybíráme řádek $i > 0$ tabulky s lexikograficky nejmenším vektorem $\frac{1}{a_{i,s}}(\vec{a}_i, b_i)$.

Věta 8.9. *Algoritmus 8.8 lexikografické simplexové metody vždy skončí se správným výsledkem.*

`Důkaz`: Jelikož podle Tvrzení 8.7 se každou iterací Algoritmu 8.8 lexikograficky zmenší účelový řádek tabulky, nikdy se v průběhu algoritmu stejná tabulka nezopakuje. Jak víme (Oddíl 6.4), přípustná bázická řešení jednoznačně odpovídají přípustným zápisům tabulky úlohy v jednotkovém tvaru. Proto se nezopakuje v průběhu algoritmu ani stejné bázické řešení úlohy. Takže algoritmus musí skončit v konečném čase.

Správnost výsledného řešení úlohy pak plyne z Lematu 6.5. □

Doplňkové otázky

(8.2.1) *Kde ve výpočtu Příkladu 8.5 lexikografické pravidlo změní průběh výpočtu?*

(8.2.2) *Dokážete nalézt jiný (podstatně) příklad zacykleného výpočtu základní simplexové metody než je v Příkladě 8.5?*

8.3 Poznámky k simplexové metodě

Z předchozích Vět 8.2,8.9 plyne hlavní závěr našeho výkladu:

Důsledek 8.10. *Dvoufázová lexikografická simplexová metoda vždy skončí a správně nalezne optimální řešení úlohy, nebo případně potvrdí jeho neexistenci.*

Jak jsme již uvedli, exponenciální horní odhad počtu kroků simplexové metody prostou enumerací všech možných bázických řešení zatím neumíme nijak podstatně vylepšit. Co se týče složitosti nejhoršího případu výpočtu, je známo následující:

`Fakt`: Pro každá běžná pravidla výběru pivota v simplexové metodě jsou známy protipříklady vyžadující exponenciální počet kroků výpočtu.

Pro řešení úloh LP jsou známy algoritmy s nejhorší polynomální časovou složitostí (viz Khachiyan, Karmarkar), avšak pro svou jednoduchost a rychlost v běžných praktických výpočtech se stejně nejčastěji používá simplexová metoda.

Na závěr si uděláme malý přehled (nástin) některých dalších užitečných variant implementace simplexové metody. Pro jejich podrobné nastudování čtenáře odkazujeme na doplňkovou literaturu.

- *Redukovaná simplexová metoda*

  Jedná se o paměťově úspornější variantu simplexové metody, ve které samotná jednotková podmatice $\boldsymbol{I}$ v matici $\boldsymbol{A}$ není uložena v tabulce a místo toho bázické proměnné přiřazujeme řádkům.

- *Revidovaná simplexová metoda*

  Její použití je výhodné, když má úloha mnohem více proměnných než nerovností. Pak vůbec nemodifikujeme výchozí tabulku – matici $\boldsymbol{A}$, ale místo toho řádkové úpravy (pivoty) provádíme na $(n+1) \times (n+1)$ matici $\boldsymbol{D}$, která zleva násobí $\boldsymbol{A}$. Tabulka simplexové metody je tedy v každém kroku dána maticí $\boldsymbol{D} \cdot \boldsymbol{A}$.

  `Komentář:` Výhodou je, že upravujeme pouze "malou" matici $\boldsymbol{D}$ a celá tabulka se ani nemusí držet v pracovní paměti.

- *Duální simplexová metoda*

  Tato metoda zjednodušeně řečeno prochází přípustná řešení duální úlohy, až najde přípustné primární řešení.

Doplňkové otázky

(8.3.1) *V jaké situaci má největší přínos redukovaná simplexová metoda?*

(8.3.2) *Čemu geometricky odpovídá exponenciálně dlouhý výpočet simplexové metody? Máme na mysli vyjádření jako "procházky po vrcholech a hranách"...*

Rozšiřující studium

Co se týče možností rozšiřujícího studia, platí i zde poznámky z konce Lekce 6. Na tomto místě bychom chtěli přidat několik referencí týkajících se především výpočetní složitosti úloh lineární optimalizace.

Příklad úlohy LP, jejíž výpočet běžnou simplexovou metodou vyžaduje exponenciálně mnoho iterací, podali [Klee a Minty, 1972]. Čtenář může nalézt jeho popis například v [H.J. Greenberg,
`http://carbon.cudenver.edu/~hgreenbe/glossary/notes/Klee-Minty.pdf`*]. V dnešní době jsou již známy varianty metody, které randomizovaně garantují ukončení výpočty v subexponenciálním čase, ale polynomiální horní odhad ještě nebyl dosažen.*

Pro polynomiální řešení úloh LP jsou známy následující dvě metody, které ale již přesahují rámec našeho textu: ellipsoid method [Khachiyan, 1979] a interior point method [Karmarkar, 1984]. Přitom pouze druhá z nich se ukazuje jako prakticky použitelná. (Ale v praktických výpočtech stejně většinou vítězí simplexová metoda.) Blíže viz [7, Chapter I.6].

Část III
Diskrétní a Celočíselná Optimalizace

9 Úloha celočíselné optimalizace

Úvod

Úlohy lineárního programování teď již ponecháme za námi a podíváme se na novou zajímavou třídu optimalizačních úloh: V praxi se totiž často vyskytují okolnosti formulace úloh, ve kterých některé (či všechny) vystupující proměnné mohou nabývat pouze celočíselných hodnot. (Například nemůžeme jednoho pracovníka "přepůlit" na dva úkoly, započatý pracovní úkon třeba nelze přerušit, nemůžeme poslat na trasu linky pouze čtvrt autobusu, a podobně.) Přitom v takových úlohách jinak omezení vypadají obdobně jako v už známých úlohách LP.

Zmíněné okolnosti pak vedou na úlohy celočíselné lineární optimalizace, neboli celočíselné programování IP. V zásadě se dá říci, že úloha IP je úlohou LP s dodatečnými podmínkami celočíselnosti proměnných. Tato analogie je však také zavádějící, neboť úlohy IP jsou nesrovnatelně komplikovanější při formulaci i obtížnější při řešení.

V této úvodní lekci poslední části si řekneme o způsobech formulace celočíselných lineárních optimalizačních úloh IP a o základním postupu jejich řešení pomocí "větvení" a odhadu lineární relaxace úlohy.

Cíle

Úkolem této lekce je nejprve přiblížit čtenáři na příkladech, jak se matematicky formulují jednoduché úlohy celočíselné lineární optimalizace IP. (Obtížnější příklady formulací úloh IP jsou zařazeny do následujících lekcí.) Dále bude stručně přiblížena základní metoda větvení a mezí pro řešení úloh IP.

9.1 Úvodní příklady IP

Opět pro názornost výkladu začneme hned s jednoduchými příklady úloh celočíselné optimalizace. Poznamenáváme, že pro tuto třídu úloh se používají zkratky IP nebo obecněji MIP. (Z anglického *Integer Programming*.)

Příklad 9.1. *Opět se vraťme ke starému známému Příkladu 3.1 o lupíncích a hranolcích s dodatečným požadavkem, že musíme vyrábět (kvůli balení produktů) lupínky i hranolky v celočíselných násobcích množství* 15*kg*.

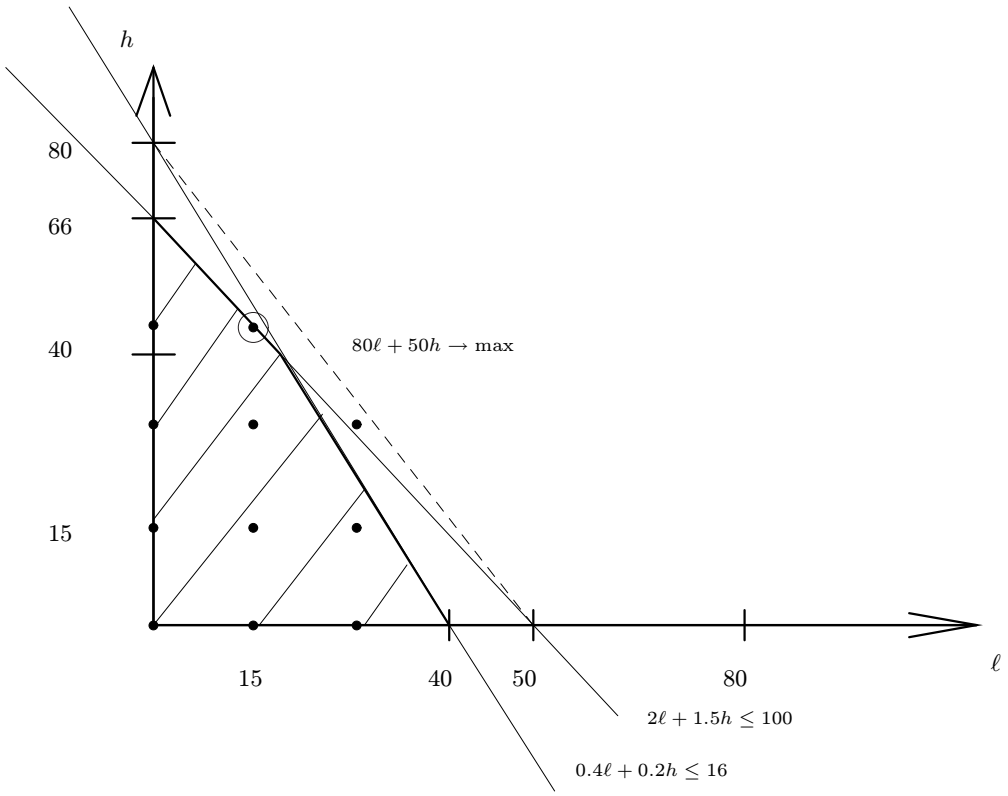
Řešení: Použijeme původní LP formulaci úlohy

$$\begin{aligned} 2\ell + 1.5h &\leq 100 \\ 0.4\ell + 0.2h &\leq 16 \\ \ell, h &\geq 0\,, \end{aligned}$$

ale přidáme dodatečnou podmínku

$$\ell = 15z_1,\ h = 15z_2,\quad \text{kde } z_1, z_2 \in \mathbb{N}\,. \tag{11}$$

Graficky si tuto formulaci vyznačíme na Obrázku 9.1. Z náhledu na obrázek vidíme, že omezení úlohy nám popisují polyedr jako v úloze LP (šrafovaný) a dodatečná podmínka celočíselnosti definuje jisté "mřížové body" (značené tečkami). Přitom přípustnými řešeními jsou ty mřížové body, které leží v polyedru, tj. množinou přípustných řešení je



Obrázek 9.1: Grafický význam formulace a řešení úlohy IP (Příklad 9.1): Množina řešení původní úlohy LP je šrafovaná, možná řešení v celočíselných násobcích 15kg jsou značená tečkami a celočíselné optimum je vyznačeno kroužkem.

průnik onoho polyedru s "celočíselnou mříží". Role účelové funkce přitom zůstává stejná jako v úlohách LP. V tomto jednoduchém případě přímo z obrázku vyčteme optimální řešení úlohy $\ell = 15\text{kg}$, $h = 45\text{kg}$.

`Komentář:` Všimněme si, že je nově nalezené řešení $\ell = 15\text{kg}$, $h = 45\text{kg}$ mírně horší než v Příkladě 3.1 bez podmínky celočíselnosti $\ell = 20\text{kg}$, $h = 40\text{kg}$. Je přirozené, že přidáním dalších podmínek se kvalita řešení může zhoršit.

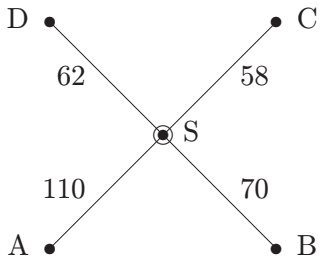
Nakonec si zadání naší úlohy můžeme po přímém dosazení z (11) zapsat jako

$$\begin{aligned} 2 \cdot 15z_1 + 1.5 \cdot 15z_2 &\leq 100 \\ 0.4 \cdot 15z_1 + 0.2 \cdot 15z_2 &\leq 16 \\ z_1, z_2 &\in \mathbb{N}\,. \end{aligned}$$

□

Příklad 9.2. *Letecká společnost přepravuje cestující z města S do čtyř sousedních měst A,B,C,D. Na dnešní den jsou požadavky na přepravu do města A* 110 *cestujících, do B* 70*, do C* 58 *a do D* 62 *cestujících. Naše společnost přitom má k dispozici dva typy letadel: Prvního typu má 6 letadel s kapacitou po* 33 *místech a s fixní cenou letu* 120*. Druhého typu má 4 letadel s kapacitou po* 60 *místech a s fixní cenou letu* 190*. Navrhněte, kolik letadel kterého typu má dnes společnost vypravit do kterého města, aby pokryla požadavky přepravy a zároveň minimalizovala cenu letů.*

Schematickým obrázkem si zadání zakreslíme takto:



typ let.	počet	kapacita	cena
1	6	33	120
2	4	60	190

Řešení: Jednoduše použijeme vědomosti, které jsme již získali při formulaci úloh LP. Nechť n_{1X} značí počet letadel prvního typu letících do města $X = \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}$. Podmínky celkového počtu letadel jednotlivých typů zformulujeme:

$$\begin{aligned} n_{1A} + n_{1B} + n_{1C} + n_{1D} &\leq 6 \\ n_{2A} + n_{2B} + n_{2C} + n_{2D} &\leq 4 \end{aligned}$$

Podmínky přepravy cestujících zní:

$$\begin{aligned} 33n_{1A} + 60n_{2A} &\geq 110 \\ 33n_{1B} + 60n_{2B} &\geq 70 \\ 33n_{1C} + 60n_{2C} &\geq 58 \\ 33n_{1D} + 60n_{2D} &\geq 62 \end{aligned}$$

Účelová funkce je taktéž zřejmá

$$\min\ 120(n_{1A} + n_{1B} + n_{1C} + n_{1D}) + 190(n_{2A} + n_{2B} + n_{2C} + n_{2D}).$$

Co nám ještě ve formulaci úlohy chybí?

Vyřešíme-li takto zadanou úlohu, výsledkem budou následující nenulové hodnoty:

$$n_{1A} = 1.8182,\ n_{2A} = 0.8333,\ n_{2B} = 1.1667,\ n_{2C} = 0.9667,\ n_{2D} = 1.0333$$

To je samozřejmě nesmyslné – neuvedli jsme podmínky celočíselnosti

$$n_{1A}, n_{1B}, n_{1C}, n_{1D}, n_{2A}, n_{2B}, n_{2C}, n_{2D} \in \mathbb{N}\,.$$

S dodatečnými celočíselnými podmínkami již optimální řešení vyjde

$$n_{1B} = 1,\ n_{1D} = 2,\ n_{2A} = 2,\ n_{2B} = 1,\ n_{2C} = 1,\ n_{2D} = 0\,,$$

neboli do města A poletí dvě velká letadla, do B malé a velké, do C jedno velké a do D dvě malá. O způsobech obecného řešení úloh IP si řekneme za chvíli. □

Doplňkové otázky

(9.1.1) *Jak byste Příklad 9.2 zformulovali s binárními proměnnými?*

(9.1.2) *Rozmyslete si další omezující požadavky v Příkladě 9.2, například co když požadujeme stejný typ letadla do jedné destinace. Jak je zformulujete matematicky?*

*(9.1.3)** *Představme si, že bychom úlohy IP řešili jako úlohy LP a následně bychom výsledek zkusili zaokrouhlit nahoru nebo dolů. Pomineme-li možnost, že "blízká" celá řešení prostě nejsou přípustná, co ještě dělá takový přístup prakticky nepoužitelným? (Napovíme, že to souvisí s velkým množstvím celočíselných proměnných.)*

9.2 Formulace úlohy (M)IP

Při matematické formulaci úloh celočíselné optimalizace postupujeme podobně jako při úlohách LP.

Definice 9.3. ***Úloha celočíselné (lineární) optimalizace*** je úlohou najít max $\vec{c} \cdot (\vec{x}, \vec{z})$ za podmínek

$$\begin{aligned} \boldsymbol{A} \cdot (\vec{x}, \vec{z})^T &\leq \vec{b} \\ \vec{x}, \vec{z} &\geq 0 \\ \vec{z} &\in \mathbb{Z}^* \end{aligned}$$

Složky vektoru $\vec{x}$ jsou *reálné proměnné*, složky $\vec{z}$ jsou *celočíselné proměnné*, oba typy se volně mísí v lineárních nerovnostech vyjařujících podmínky úlohy. Hodnoty složek $\vec{z}$ obvykle mohou nabývat jen konečně mnoha hodnot (říkáme, že pak je $\vec{z}$ *omezené*).

`Značení`: Takto zadané úloze se obvykle říká *smíšená* se zkratkou *MIP* (mixed IP), vyjadřujíce fakt, že ve formulaci se mísí celočíselné i reálné proměnné. Zkratka *IP* pak označuje úlohu celočíselné optimalizace bez reálných proměnných, což je poměrně častý praktický případ.

`Definice`: Úloha MIP je v *základním tvaru*, pokud je formulována jako

$$\begin{aligned} \max\ \vec{c} \cdot (\vec{x}, \vec{z}) &: \\ \boldsymbol{A} \cdot (\vec{x}, \vec{z})^T &\leq \vec{b} \\ \vec{x} &\geq 0 \\ \vec{z} &\in \{0,1\}^* \end{aligned}$$

Proměnným $z_i \in \{0,1\}$ se říká *binární proměnné*.

V praxi se ukazuje, že obvykle mohou celočíselné proměnné nabývat jen konečně mnoha hodnot (například $0, 1, \ldots, k$) a v jiných případech omezení hodnot celočíselných proměnných lze snadno do úlohy doplnit. Proto se nadále budeme zabývat jen úlohami MIP s omezenými celočíselnými proměnnými.

Věta 9.4. *Každou úlohu MIP s omezenými celočíselnými proměnnými lze převést na základní tvar.*

`Důkaz`: Každou celočíselnou proměnnou $z_i \in \{0, 1, ..., k_i\}$ (dle předpokladů omezená) nahradíme $l = \lceil \log_2(k_i + 1) \rceil$ proměnnými $z_i^0, z_i^1, ..., z_i^{l-1}$ a píšeme $z_i = z_i^0 + 2z_i^1 + ... + 2^{l-1} z_1^{l-1}$ (jako v binárním zápise hodnoty z_i). Je zřejmé, že každá přípustná hodnota z_i jednoznačně odpovídá jisté přípustné posloupnosti hodnot $z_i^0, z_i^1, ..., z_i^{l-1} \in \{0,1\}^l$, která vyjadřuje číslice binárního zápisu čísla z_i. Nakonec případně rovnosti a nerovnosti podmínek úlohy převedeme do základního LP tvaru podle Věty 3.5. □

Zobecněné formulace úloh MIP

Jako v případě úloh LP, i v podmínkách úloh MIP se mohou kromě levých nerovností vyskytovat i pravé a případně rovnosti a neomezené reálné proměnné. Všechny tyto případy snadno dokážeme převést na základní tvar podle dřívější Věty 3.5.

V případě celočíselných úloh jsou však možná i následující netriviální zobecnění celočíselných proměnných. Poznamenáváme, že potom budeme obecněji mluvit o úlohách *diskrétní optimalizace*, kde slovo diskrétní nevyjadřuje žádné utajení, ale nespojitost množiny hodnot proměnné.

Lema 9.5. *Do základního tvaru úlohy MIP lze převést také úlohy diskrétní optimalizace, ve kterých se vyskytují diskrétní proměnné $\vec{t}$ následujících typů:*

- *Proměnné s více diskrétními reálnými hodnotami, tj. proměnná $t_i \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, kde $\alpha_1, \ldots, \alpha_k$ jsou libovolná reálná čísla.*
- *Semikontinuální proměnné, tj. proměnná $t_i \in \{0\} \cup [\alpha, \beta]$ nabývající nulové hodnoty nebo hodnoty z intervalu $[\alpha, \beta]$ (neobsahujícího nulu).*
- *Proměnné s více intervalovými hodnotami, tj. proměnná $t_i \in [\alpha_1, \beta_1] \cup [\alpha_2, \beta_2] \cup \ldots$ s hodnotami ze sjednocení několika disjunktních intervalů.*

`Důkaz`: Je vidět, že stačí dokázat třetí bod, jelikož první dva jsou jeho speciálními případy. Nechť se množina přípustných hodnot pro t_i skládá z l intervalů $[\alpha_1, \beta_1], [\alpha_2, \beta_2], \ldots, [\alpha_l, \beta_l]$. Použijeme $l-1$ binárních proměnných $z_{i2}, \ldots, z_{il}$ a dodatečné podmínky

$$z_{i,2} + \ldots + z_{i,l} \leq 1$$

$$\alpha_1 + \sum_{j=2}^{l} z_{ij}(\alpha_j - \alpha_1) \;\leq\; t_i \;\leq\; \beta_1 + \sum_{j=2}^{l} z_{ij}(\beta_j - \beta_1)\,.$$

Proměnné z_{ij} nám tak "vyberou", ve kterém z intervalů bude ležet hodnota t_i. (Všimněte si, že první nerovnost zaručuje, že bude vybrán jen jeden interval.) □

Doplňkové otázky

(9.2.1) *Rozmyslete si, jak třeba v rámci formulace úlohy MIP získáte hodnotu, která je zaokrouhlením proměnné x_1?*

(9.2.2) *Jak v rámci formulace úlohy MIP získáte hodnotu, která je zlomkovou částí proměnné x_1? (Předpokládejte, že x_1 nemá celou hodnotu.)*

(9.2.3) *Podívejte se znovu na řešení předchozí otázky. Proč, ať jakkoliv zadefinujete zlomkovou část y_1 proměnné x_1, bude v případě celé hodnoty x_1 její zlomková část neurčena jednoznačně $(y_1 = 0, 1)$?*

(9.2.4) *Představte si úlohu IP s celými proměnnými z_1, z_2. Jak vyjádříme požadavek $z_1 \neq z_2$?*

9.3 Řešení úloh MIP relaxací a větvením

Nyní si ve zjednodušeném podání uvedeme jednu ze základních metod pro řešení úloh MIP. Ta je založena na myšlence "rozvětvení" původní úlohy podle voleb hodnot jednotlivých celočíselných proměnných do množství podúloh umístěných v "binárním stromu", a následném prohledávání všech uzlů takového "stromu" a řešení podúloh pomocí lineární optimalizace.

Definice 9.6. ***LP - relaxací*** úlohy MIP základního tvaru

$$\begin{aligned} \max\ \vec{c}\cdot(\vec{x},\vec{z}) \quad &: \\ \boldsymbol{A}\cdot(\vec{x},\vec{z})^T \quad &\leq \quad \vec{b} \\ \vec{x} \quad &\geq \quad 0 \\ \vec{z} \quad &\in \quad \{0,1\}^* \end{aligned}$$

je úloha LP ve tvaru:

$$\begin{aligned} \max\ \vec{c}\cdot(\vec{x},\vec{z}) \quad &: \\ \boldsymbol{A}\cdot(\vec{x},\vec{z})^T \quad &\leq \quad \vec{b} \\ \vec{z} \quad &\leq \quad 1 \\ \vec{x},\vec{z} \quad &\geq \quad 0 \end{aligned}$$

`Komentář:` LP relaxaci základní úlohy MIP vlastně získáme rozšířením přípustného oboru pro proměnné $\vec{z}$ z hodnot $0, 1$ na celý interval $[0, 1]$. Geometricky si lze představit polyedr všech přípustných řešení LP-relaxace a v něm obsažené "mřížové" body odpovídající přípustným celým hodnotám $\vec{z} \in \{0, 1\}^*$, podobně Obrázku 9.1. Podrobněji viz příští lekce.

`Fakt:` Množina přípustných řešení úlohy MIP je právě průnikem množiny přípustných řešení její LP-relaxace s mřížovými body $\{\vec{z} \in \{0, 1\}^*\}$.

`Definice:` Hodnotu optimálního řešení LP-relaxace úlohy MIP nazýváme *(LP-)relaxační mezí* pro původní úlohu MIP.

`Fakt:` Hodnota optimálního řešení úlohy MIP není nikdy lepší než hodnota její LP-relaxační meze. Proto pokud najdeme přípustné řešení úlohy MIP dosahující hodnoty její relaxační meze, máme optimální řešení.

Algoritmus 9.7. ***Jednoduchá metoda větvení a mezí s lineární relaxací***
Mějme danou úlohu MIP v základním tvaru, tj. s binárními proměnnými $\vec{z}$.
Nechť f je globální proměnná inicializovaná $f = -\infty$.

Procedura `branch&bound` ($\mathcal{U}$: úloha MIP)

1. $\mathcal{L}$ = LP-relaxace $\mathcal{U}$, $\vec{s}^o$ = optimální řešení $\mathcal{L}$, r^o = relaxační mez.
2. Pokud $\mathcal{L}$ nemá přípustné řešení nebo $r^o \leq f$, vrátíme se bez řešení `return` $\emptyset$.
3. Pokud $\mathcal{L}$ nemá optimální řešení z důvodu neomezenosti a zároveň v některém přípustném řešení $\mathcal{L}$ jsou složky $\vec{z}$ celé, položíme $f = \infty$ a vrátíme `return` $\emptyset$.
4. Pokud $\vec{s}^o$ je přípustné (celočíselné ve složkách $\vec{z}$) řešení $\mathcal{U}$, položíme $f = r^o$ a vrátíme `return` $\vec{s}^o$.
5. Zvolíme binární proměnnou z_i (*nedeterministicky*) v $\mathcal{U}$ a vytvoříme jejím dosazením podúlohy
$$\mathcal{U}_0 := (\mathcal{U} \restriction z_i = 0)\,, \qquad \mathcal{U}_1 := (\mathcal{U} \restriction z_i = 1)\,.$$
Zavoláme rekurzivně
$$\vec{s}^1 = \texttt{branch\&bound}(\mathcal{U}_1) \text{ a } \vec{s}^2 = \texttt{branch\&bound}(\mathcal{U}_2)$$
zároveň *nedeterministicky*. Vracíme lepší z obou řešení `return` max $(\vec{s}^1, \vec{s}^2)$.

Návratová hodnota procedury je optimálním řešením úlohy $\mathcal{U}$ s účelovou funkcí f. Pokud $f = -\infty$, úloha nemá přípustné řešení, pokud $f = \infty$, úloha nemá optimální řešení z důvodu neomezenosti.

`Poznámka:` Pro správné porozumění algoritmu je třeba upřesnit význam *nedeterministických* kroků v bodě 5. Za prvé, algoritmus bude fungovat s jakoukoliv volbou "větvící" proměnné z_i, takže je zde ponechána uživateli volba pro co nejlepší implementaci. Za druhé, obě větve podúloh se musí zavolat v rekurzi, ale nikde není určeno, v jakém pořadí, takže opět má uživatel možnost volby. Dokonce je možno kdykoliv zpracování jedné podúlohy odložit a pokračovat mezitím jinými větvemi.

Tvrzení 9.8. *Algoritmus 9.7 vždy (pro jakoukoliv volbu vykonání a pořadí nedeterministických kroků) skončí a správně nalezne optimální řešení.*

`Důkaz:` Nechť d je počet binárních proměnných – složek $\vec{z}$. Pak zřejmě žádná větev rekurze není hlubší než d, neboť každým zanořením se sníží počet binárních proměnných v úloze. Nakonec pokud úloha $\mathcal{U}$ neobsahuje binární proměnné ($\vec{z}$ nemá složky), tak se vždy aplikuje jeden z kroků 2,3,4 před 5 a rekurze se ukončí. Takže algoritmus skončí po $O(2^d)$ iteracích procedury `branch&bound`.

Předpokládejme, že optimální řešení úlohy $\mathcal{U}$ má binární složky rovny $\vec{z}^o$. Pak ve větvi rekurze, která odpovídá volbám hodnot $\vec{z}$ jako v $\vec{z}^o$ bude toto řešení nalezeno jako přípustná relaxační mez úlohy $\mathcal{L}'$. (Úloha $\mathcal{U} \restriction \vec{z} = \vec{z}^o$ je již úlohou LP, kterou umíme přesně vyřešit.) Toto přípustné řešení pak jako nejlepší možné (případně jedno z několika stejné hodnoty) bude vráceno procedurou `branch&bound`. □

`Komentář:` Různými implementačními způsoby volby proměnné z_i pro "větvení" se budeme zabývat v příští kapitole. (Algoritmus korektně funguje pro jakoukoliv volbu, ale vhodnou volbou jej lze výrazně urychlit.)

Podívejme se blíže na analýzu složitosti v důkaze 9.8 – časový odhad $O(2^d)$ je velmi "špatný" již pro poměrně malé hodnoty d. Co však způsobuje tento exponenciální nárust složitosti algoritmu? Je to prohledávání mnoha větví až do úplného konce, do hloubky d. Naši snahou při implementaci algoritmu tedy musí být co nejrychleji "zabít" všechny neperspektivní větve rekurze. V praxi se s největším potenciálem k vyloučení větví rekurze ukazuje krok 2, když relaxační mez vyjde horší než nejlepší dosud nalezené řešení. K aplikaci tohoto kroku však již nějaké přípustné řešení potřebujeme mít nalezené, proto by naší snahou mělo zpočátku být co nejdříve nalézt (jedno, jakým způsobem) nějaká dobrá přípustná celočíselná řešení úlohy.

Doplňkové otázky

(9.3.1) *Mějme úlohu IP s binárními proměnnými. Co se geometricky stane s množinou přípustných řešení, pokud přejdeme k LP-relaxaci?*

(9.3.2) *Co přesně nastává, pokud úloha MIP (s omezenými celočíselnými proměnnými) je neomezená?*

(9.3.3) *Najděte příklad, kdy jedna větev v Algoritmu 9.7, krok 5, je neomezená a zároveň druhá je nepřípustná.*

9.4 Jednoduché ukázky řešení IP

Příklad 9.9. *Letecká společnost má přepravit 141 lidí z města A do B. K dispozici má čtyři letadla:*

	kapacita	*cena letu*	*posádka*
L1	*90*	*300*	*7*
L2	*60*	*210*	*4*
L3	*50*	*200*	*3*
L4	*33*	*130*	*2*

Která letadla má zvolit, aby dosáhla nejlevnějšího řešení?

Řešení:

$$\max \; -300z_1 - 210z_2 - 200z_3 - 130z_4$$

U: pro $90z_1 + 60z_2 + 50z_3 + 33z_4$

$$z_i \in \{0, 1\}$$

$f = -\infty$ U LP relaxace

$$\max \; -300z_1 - 210z_2 - 200z_3 - 130z_4$$

$$90z_1 + 60z_2 + 50z_3 + 33z_4 \geq 141$$

$$0 \leq z_i \leq 1$$

Řešení...

Výsledek: poletí L1 a L2. □

Doplňkové otázky

Rozšiřující studium

Pro dobrý obsáhlý úvod do problematiky formulace úloh MIP doporučujeme následující učební texty [3, Kapitoly 2,4.1] a [7, Chapter I.1]. Některé pokročilé způsoby formulace úloh MIP budou také ukázány v Lekci 12. Konkrétně jednoduchou a snadno aplikovatelnou formulaci metody větvení a mezí může čtenář nalézt v [3, Oddíl 4.3].

10 Význam a řešení úloh MIP

Úvod

Předchozí neformální uvedení způsobu řešení úloh celočíselné optimalizace je v následující lekci postaveno na pevné matematické základy. Náš náhled na přípustná řešení úloh MIP je přes (celočíselné) "mřížové" body v prostoru:

Omezení (nerovnosti) daná úlohou MIP geometricky definují polyedr jako v případě úloh LP, avšak nyní za přípustná řešení bereme jen ty body, keré jsou v průniku onoho polyedru s tzv. celočíselnou mříží danou podmínkami celočíselnosti na vybraných proměnných. Naše optimalizace se tak týká pouze tohoto průniku nebo jeho konvexního obalu. (Ano, konvexní obal přípustných mřížových bodů je polytop, tedy i polyedr, ale jeho popis nerovnostmi povětšinou efektivně získat neumíme pro jeho rozsáhlost.)

Řešení pomocí metody větvení postupně dělí a prohledává celočíselnou mříž úlohy, přičemž si pomáhá například lineárními relaxacemi úlohy pro vyloučení neperspektivních větví hledání. Pochopitelně nejpozději po rozdělení mříže na jednotlivé body je úloha vyřešena opakovanou aplikací lineární optimalizace. Jedná se však o dlouhý, až exponenciálně, proces.

Cíle

V lekci si osvojíme pojem celočíselné mříže, která popisuje přípustná (celočíselná) řešení úloh MIP, a ukážeme význam konvexního obalu celočíselných řešení úlohy. Dále si popíšeme obecné schéma metody větvení a její implementaci pomocí mezí lineárních relaxací úlohy. Na příkladě rozvrhování si ukážeme i alternativní způsob relaxace úlohy v této metodě.

10.1 Celočíselná mříž

Jak jsme již psali, přípustná řešení úloh IP se získají jako průnik všech řešení odpovídající úlohy LP (tj. její LP-relaxace) s jistou "celočíselnou mříží". Nyní je na čase tento pojem a související poznatky matematicky zformalizovat.

`Definice`: *Celočíselná mříž* v $\mathbb{R}^d$ (dimenze d) je množina všech bodů s celočíselnými souřadnicemi

$$\mathbb{M}^d = \{(z_1, \ldots, z_d) : z_1, \ldots, z_d \in \mathbb{Z}^d\}\,.$$

Taková definice však stačí postihnout pouze úlohy IP, ale co s obecnějšími úlohami MIP, které obsahují i reálné (neboli kontinuální) proměnné?

`Definice`: *Rozšířená celočíselná mříž* dimenze $d + r$ je množina všech bodů

$$\mathbb{M}^{d,r} = \{(x_1, \ldots, x_r, z_1, \ldots, z_d) : (z_1, \ldots, z_d) \in \mathbb{M}^d,\, x_1, \ldots, x_r \in \mathbb{R}\}\,.$$

Je to tedy sjednocení všech disjunktních podprostorů dimenze r, jejichž posledních d souřadnic jsou fixní celočíselné hodnoty.

Důležitost mříže pro úlohy IP ukazuje následující zřejmý poznatek.

`Fakt`: Mějme úlohu IP $\mathcal{U}$ s omezenými celočíselnými proměnnými. Pak množina přípustných řešení $\mathcal{U}$ (jako podmnožina mříže $\mathbb{M}^d$) je konečná a její konvexní obal je polytop. Podle Věty 4.12 lze tudíž konvexní obal Q přípustných řešení úlohy $\mathcal{U}$ zapsat jako polyedr Q, na kterém pak můžeme použít běžnou lineární optimalizaci: Jelikož řešením LP je některý krajní bod polyedru, je výsledkem lineární optimalizace na Q jedno z přípustných řešení $\mathcal{U}$.

Pozor, uvedený fakt však neznamená, že bychom rázem měli efektivní metodu řešení úloh IP, neboť v obvyklých příkladech složitost zápisu polyedru Q obrovským způsobem vzroste oproti původní úloze (oproti původnímu polyedru lineární relaxace).

O některých důležitých příkladech, kdy konvexní obal přípustných řešení úlohy IP umíme zapsat rozumným způsobem (jako tzv. celočíselný polyedr) se zmíníme v Oddíle 12.3. Nyní se ještě uvedeme rozšíření tohoto faktu:

Tvrzení 10.1. *Mějme úlohu MIP $\mathcal{U}$ s omezenými proměnnými. Pak konvexní obal K množiny přípustných řešení $\mathcal{U}$ (jako podmnožiny mříže $\mathbb{M}^{d,r}$) je polyedrem.*

`Důkaz`: Nechť P je (omezený) polyedr definovaný nerovnostma úlohy $\mathcal{U}$, tedy že $P \cap \mathbb{M}^{d,r}$ jsou přípustná řešení $\mathcal{U}$. Nechť $\vec{z}$ je vektor celočíselných proměnných. Pak díky omezenosti může $\vec{z}$ nabývat jen konečně mnoha hodnot z podmříže $\vec{z} \in M$. Z definice polyedru je $P \cap \{\vec{z} : \vec{z} = \vec{z}^1\}$ omezeným polyedrem pro každou volbu $\vec{z}^1 \in M$, tedy i polytopem s konečnou množinou vrcholů $V(\vec{z}^1)$. Položíme $V = \bigcup_{\vec{z}^1 \in M} V(\vec{z}^1)$, což je také konečná množina, a K je konvexním obalem V, tudíž je K polytopem i polyedrem. □

`Poznámka`: Idea hledání polyedru, který je konvexním obalem přípustných řešení úlohy MIP, stojí za obecným schématem tzv. *"cutting-plane" algoritmů* pro řešení MIP.

Základním krokem je postupné přidávání *platných nerovností* k původní formulaci úlohy, tj. přidávání takových nerovností, které nevyplývají z lineárních kombinací ostatních nerovností úlohy, ale které zachovají množinu přípustných celočíselných řešení. (Toto si lze představit jako "uříznutí" neceločíselných vrcholů polyedru.) Důležitým původním příkladem takových nerovností jsou *Gomoryho řezy*, které v konečném počtu kroků konvergují k optimálnímu řešení úlohy IP.

Doplňkové otázky

(10.1.1) *Proč asi úlohy IP umožňují mnohem větší variabilitu formulací než úlohy LP? (Podívejte se na otázku geometricky.)*

(10.1.2) *Je pravdou, že konvexní obal množiny přípustných řešení úlohy IP (bez omezení proměnných) je vždy polyedrem?*

10.2 Obecná metoda větvení

Níže popsaná obecná metoda větvení rozšiřuje pojetí Algoritmu 9.7 na různé typy relaxací úlohy MIP, které nemusí být lineární ani nemusí být řešeny simplexovou metodou.

`Definice`: Nechť $\mathcal{U}$ je úlohou MIP s reálnými proměnnými $\vec{x} = (x_1, \ldots, x_r)$ a celočíselnými proměnnými $\vec{z} = (z_1, \ldots, z_d)$, které patří do rozšířené celočíselné mříže $(\vec{x}, \vec{z}) \in \mathbb{M}^{d,r}$. Označme P množinu přípustných řešení $\mathcal{U}$. Množinu R (libovolnou) nazveme *množinou relaxovaných řešení* úlohy $\mathcal{U}$, pokud $P = R \cap \mathbb{M}^{d,r}$ je průnikem R s celočíselnou mříží $\mathbb{M}^{d,r}$.

Algoritmus 10.2. ***Obecná metoda větvení pro řešení úloh MIP***
Mějme dánu obecnou úlohu MIP s reálnými proměnnými $\vec{x} = (x_1, \ldots, x_r)$ a celočíselnými proměnnými $\vec{z} = (z_1, \ldots, z_d)$, ve které je účelovou funkcí $\max \vec{c} \cdot (\vec{x}, \vec{z})$.
Nechť f je globální proměnná inicializovaná $f = -\infty$ a nechť M je podmnožina celočíselné mříže obsahující všechny přípustné hodnoty $\vec{z}$.

Procedura `branch&bound` ($\mathcal{U}$: úloha MIP, $M \subseteq \mathbb{M}^d$)

1. Pokud M obsahuje jediný prvek, pak fixujeme $\vec{z}^o \in M$ a řešíme úlohu LP $\mathcal{U} \restriction \vec{z} = \vec{z}^o$ (třeba simplexovou metodou). Vrátíme její optimální řešení `return` (x^o, z^o).
2. Zvolíme $R \subseteq \mathbb{R}^{r+d}$: libovolná množina relaxovaných řešení úlohy $\mathcal{U} \restriction \vec{z} \in M$.
3. $r^o = \max\{\vec{c} \cdot \vec{t} : \vec{t} \in R\}$, $\vec{t}^o \in R : r^o = \vec{c} \cdot \vec{t}^o$ (optimální řešení relaxované úlohy).
4. Pokud $r^o \leq f$ nebo $R = \emptyset$, vrátíme se bez řešení `return` $\emptyset$.
5. Pokud $r^o = \infty$ pro nějaké přípustné $\vec{z} \in M$, položíme $f = \infty$ a vrátíme `return` $\emptyset$.
6. Pokud je $\vec{t}^o \in \mathbb{M}^{d,r}$, tj. přípustné v $\mathcal{U}$, položíme $f = r^o$ a vratíme řešení `return` $\vec{t}^o$.
7. [Případně heuristicky vyhledáme přípustné řešení $\vec{s} \in \mathbb{M}^{d,r}$ "blízké" (zaokrouhlením) k $\vec{t}^o$ a upravíme podle něj f.]
8. [Případně formulaci úlohy $\mathcal{U}$ rozšíříme ("vylepšíme") na $\mathcal{U}'$ o další *platné podmínky*, tj. zachovávající $\mathcal{U} \cap M = \mathcal{U}' \cap M$, zjištěné z nepřípustnosti řešení $\vec{t}^o$. Zavoláme `branch&bound` $(\mathcal{U}', M)$.]
9. *Krok větvení*: S pomocí $\vec{t}^o$ nalezneme (libovolný) rozklad $M = M_1 \cup M_2$, kde $M_1 \cap M_2 = \emptyset$ a $M_1, M_2 \neq \emptyset$. Zavoláme rekurzivně

$$\vec{t}^1 = \texttt{branch\&bound}(\mathcal{U}, M_1) \quad \text{a} \quad \vec{t}^2 = \texttt{branch\&bound}(\mathcal{U}, M_2)$$

zároveň *nedeterministicky*. Vracíme lepší z obou řešení `return` max $(\vec{t}^1, \vec{t}^2)$.

Body (7,8) jsou nepovinné. Návratová hodnota procedury je optimálním řešením úlohy $\mathcal{U}$ s účelovou hodnotou f. Pokud $f = -\infty$, úloha nemá přípustné řešení, pokud $f = \infty$, úloha nemá optimální řešení z důvodu neomezenosti.

`Komentář:` U tohoto algoritmu se jedná o velmi obecné schéma s různými možnými implementacemi. Hlavní možnost volby nám poskytuje nedeterminismus v bodě (9); a to jak při volbě rozkladu podmříže M, tak i při pořadí volání větví M_1, M_2. Dále je také možno libovolně volit implementace bodů (7,8), nebo je jednoduše neimplementovat vůbec.

Jak vidíme v bodě (4), pro efektivní implementaci metody potřebujeme rychle nalézt co nejlepší přípustné řešení úlohy (abychom měli k dispozici dobrou hranici f pro ukončení neperspektivních větví výpočtu, kterých je většina). Proto se v bodě (7) také snažíme jakýmkoliv heuristickým způsobem přípustná řešení odhadnout.

Co se týče bodu (8), kdy "vylepšovat" formulaci úlohy, budeme se problematice více věnovat v Oddíle 12.4.

`Poznámka`: Důležitým aspektem implementace uvedeného schématu algoritmu je způsob zpracování volaného větvení v bodě (9). Naše schéma je formulováno tak, že větvení je rekurzivním voláním, avšak to je z hlediska praktické programové implementace sice snadné, ale neefektivní. Pro vylešování implementace totiž potřebujeme mít programovou kontrolu nad pořadím a způsobem zpracování jednotlivých větví. Proto je vhodné si vytvořit vlastní datovou strukturu – úložiště nezpracovaných větví metody.

Věta 10.3. *Mějme dánu obecnou úlohu MIP $\mathcal{U}$ s omezenými celočíselnými proměnnými z konečné podmříže M. Pokud se nepoužije bod (8), nebo pokud je zabráněno jeho zacyklení, tak Algoritmus 10.2 na* `branch&bound` $(\mathcal{U}, M)$ *vždy skončí a nalezne optimální řešení.*

`Důkaz`(náznak): Důkaz jen přímočaře zobecňuje argumenty Tvrzení 9.8. Za prvé, hloubka rekurze je omezená počtem bodů $|M|$ – nejpozději v této hloubce nastane $|M| = 1$ a uplatní se bod (1). Proto je algoritmus konečný, ale délka výpočtu je až exponenciální. Optimální řešení úlohy zřejmě náleží do jedné z prohledávaných větví a jako takové bude nalezeno a vráceno. □

Doplňkové otázky

(10.2.1) *Lze v Algoritmu 10.2 dosáhnout lepšího odhadu času výpočtu, pokud budeme v bodě (9) dělit podmříže zhruba napůl? (Dosáhneme tak hloubku rekurze logaritmickou v $|M|$.)*

(10.2.2) *Co se stane s Algoritmem 10.2 v případě, že počáteční mříž M je nekonečná? Skončí jeho běh?*

10.3 Metoda větvení a mezí s lineárními relaxacemi

V následujícím oddíle si především vysvětlíme, jak popis konkrétního Algoritmu 9.7 (pro větvení a meze základní úlohy MIP) zapadá do rámce schématu Algoritmu 10.2.

Algoritmus 10.4. ***Postup větvení a mezí s lineárními relaxacemi***
Nechť $\mathcal{U}$ je obecná úloha MIP s omezenými celočíselnými proměnnými $\vec{z} \leq \vec{d}$.
Přirozeně položíme $M = \{\vec{z} : 0 \leq \vec{z} \leq \vec{d}\}$. Při implementaci `branch&bound`$(\mathcal{U}, M)$ postupujeme dle Algoritmu 10.2 s následujícími implementačními detaily:

- V kroku 2 za relaxovaná řešení volíme lineární relaxaci $\mathcal{U}$.
- V kroku 3 nalezneme r^o, $\vec{t}^o$ simplexovou metodou.
  Případně pokud $\mathcal{U}$ je úloha IP a M obsahuje jen málo mřížových bodů, je občas lepší přeskočit relaxaci úlohy úplně a jen dál větvit M až na jednotlivé body.
- V kroku 7 se pro neceločíselné $\vec{z}$-složky řešení $\vec{t}^o$ pokusíme získat přípustné řešení zaokrouhlením (třeba náhodným) těchto necelých proměnných v $\vec{t}^o$.
- Větvení kroku 9 určíme takto:
  - Zvolíme libovolnou celočíselnou proměnnou z_i, která má necelou hodnotu z_i^o v $\vec{t}^o$ (z_i zde nazveme *větvící proměnnou*).
  - Definujeme

$$M_1 = M \cap \{z_i \leq \lfloor z_i^o \rfloor\} \ , \qquad M_2 = M \cap \{z_i \geq \lceil z_i^o \rceil\} \ .$$

  Konkrétně *výběr větvící proměnné* je možný pro příklad následujícími postupy:

- Uživatelem specifikované priority (obvykle vycházející z povahy řešeného problému).
- Maximální *celočíselná nepřípustnost* – tam vybíráme proměnnou z_i, jejíž necelá část $(z_i^0 - \lfloor z_i^0 \rfloor)$ je nejblíže číslu 0.5. Ve větvení se pak doporučuje nejprve řešit větev M_1, pokud necelá část je < 0.5, jinak M_2.
- Zobecněním předchozího je volba podle *celočíselné degradace*, kdy jsou (nějak) specifikovány koeficienty p_i^+, p_i^- (mohou se měnit během výpočtu) a vybíráme proměnnou z_i s necelou hodnotou, pro kterou je maximalizováno $\min((p_i^-(z_i^0 - \lfloor z_i^0 \rfloor), p_i^+(\lceil z_i^0 \rceil - z_i^0))$.
- Jiné, chytřejší metody se v praxi obvykle ukazují jako příliš zdlouhavé.

Doplňkové otázky

10.4 Řešené příklady IP

V návaznosti na Oddíl 9.4 rozšíříme formulaci úlohy:

Příklad 10.5. *Letecká společnost má přepravit* 141 *lidí z města A do B. K dispozici má čtyři letadla:*

	kapacita	*cena letu*	*posádka*
L1	*90*	*300*	*7*
L2	*60*	*210*	*4*
L3	*50*	*200*	*3*
L4	*33*	*130*	*2*

Která letadla má zvolit, aby dosáhla nejlevnějšího řešení?

Řešení: Řešíme tutéž úlohu, ale nyní doplníme omezení počtu členů posádky, kterých má společnost celkem k dispozici 10.

$$7z_1 + 4z_2 + 3z_3 + 2z_4 \leq 10$$

Výsledek: poletí L2, L3 a L4.

Následně doplníme další vlastnost, a to možnost použití aerotaxi s kapacitou 5 a cenou 35 (bez potřeby vlastní posádky).

$$\max \quad -300z_1 - 210z_2 - 200z_3 - 130z_4 - 35z_5$$

$$90z_1 + 60z_2 + 50z_3 + 33z_4 + 5z_5 \geq 141$$

$$7z_1 + 4z_2 + 3z_3 + 2z_4 \geq 10$$

$$z_i \in \{0, 1\}$$

Výsledek: poletí L1, L3 a aerotaxi. □

Doplňkové otázky

(10.4.1) *Vymyslete a vyřešte další možná přirozená omezení v Příkladě 10.5.*

10.5 Větvení a meze trochu jinak

V návaznosti na Oddíl 1.5 si ukážeme, jak se úloha *nepřerušitelného rozvrhování* na jednom stroji (Příklad 1.16) řeší postupem Algoritmu 10.2 s relaxací úlohou *přerušitelného rozvrhování* (Příklad 1.15).

Příklad 10.6. *Jednotlivé nepřerušitelné rozvrhování úloh II.*

Mějme stroj, který zpracovává po jedné zadané úlohy. Každá úloha má danou prioritu, je připravena ke zpracování v určitém čase a její dokončení trvá určitou dobu. Přitom postup zpracování jedné úlohy nelze přerušit (ostatní úlohy musí čekat na její dokončení).

Vstup: Úloha J_i začne v čase t_i s délkou l_i a prioritou p_i, $i = 1, 2, \ldots, n$.
Výstup: Pořadí zpracování, ve kterém úloha J_i je ukončena v čase f_i.

Cena řešení je dána souhrnem čekání na dokončení jednotlivých úloh, váženým prioritami úloh

$$\min c = \sum_{i=1}^{n} (f_i - t_i) \cdot p_i \,.$$

Řešení: Úlohu budeme řešit specifickou implementací metody větvení a mezí, přičemž jako relaxační mez nám bude sloužit řešení obdobné úlohy, ve které lze zpracování úloh libovolně přerušovat.

Jako v Příkladě 1.16 použijeme formulaci, kdy pořadí zpracování úloh bude určeno jednou z $n!$ permutací. Na rozdíl od běžných úloh MIP však nebudeme sestavovat soustavu lineárních nerovnic pro formulaci omezení, ale budeme kombinovat účelovou hodnotu řešení nepřerušitelného rozvrhování pro danou permuaci s relaxací úlohy přerušitelným rozvrhováním.

. . . . . □

`Komentář:` Výhodou ukázaného řešení je, že hladová relaxace přerušitelným rozvrhováním (Příklad 1.15) je mnohem rychlejší než simplexová metoda a navíc má snadnější implementaci.

Doplňkové otázky

Rozšiřující studium

O problematice celočíselných mříží a hlavně o platných nerovnostech obsáhle pojednává [7, Chapter II.1]. Jelikož se náš text této problematice věnuje jen povrchně (pro nedostatek času), doporučujeme čtenáři s hlubším zájmem studium další literatury. Více o řezných nadrovinách, jejich algoritmickém využití i důkazu správnosti tohoto přístupu lze číst v [3, Oddíl 4.2] a v [7, Section II.4.3].

Obecně je metoda větvení a mezí (branch&bound) popsána také v [7, Section II.4.2] a v [3, Oddíl 4.3]. Čtenář si pro rozšíření svých obzorů může nastudovat více o způsobech větvení, zpracování větví apod.

11 Kombinatorické optimalizační problémy

Úvod

Velmi mnoho známých kombinatorických úloh má přirozené formulace v lineární či celočíselné optimalizaci. Předpokládáme, že čtenáři, zvláště ti mající informatický základ, znají různé běžné kombinatorické problémy na grafech, nebo třeba problémy splnitelnosti formulí. Ukázkami jejich formulace zároveň čtenáři ukážeme některé obecné principy formulace úloh IP. Jedním z asi (laikům) nejznámějších těžkých optimalizačních problémů je tzv. problém obchodního cestujícího, kterému se budeme hlouběji věnovat.

Cíle

V této lekci si ukážeme, jak jsou úlohy IP ve spojení s běžnými známými kombinatorickými úlohami. Mimo jiné tak formulací problému splnitelnosti logických formulí ukážeme, že úlohy IP nejspíše nemají efektivní řešení (jsou NP-těžké). Dále si konkrétně uvedeme IP formulace problémů nezávislosti, dominující množiny a barevnosti grafu a také popis známého optimalizačního problému obchodního cestujícího. Tyto příklady mají čtenáři také ukázat základní principy formulace úloh IP.

11.1 Formulace problému SAT

Začneme s IP formulací jednoho ze základních obtížných algoritmických problémů a jeho optimalizačního rozšíření.

`Definice`: Logická formule Φ s proměnnými $x_1, x_2, \ldots, x_n$ je v *konjunktivním normálním tvaru* pokud je zapsaná jako

$$\Phi = c_1 \wedge c_2 \wedge \ldots \wedge c_k \,,$$

kde každé c_i se nazývá *klauzule* a představuje zkratku pro

$$c_i = (\ell_{i1} \vee \ell_{i2} \vee \ldots \vee \ell_{im_i})$$

a kde ℓ_{ij} je *literál* mající jednu ze dvou podob pro některé $p \in \{1, \ldots, n\}$

$$\ell_{ij} \equiv x_p \quad \text{nebo} \quad \ell_{ij} \equiv \neg x_p \,.$$

`Komentář:` Příklady logických formulí v konjunktivním normálním tvaru jsou:

$$\Phi_1 = (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4)$$
$$\Phi_2 = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_4 \vee \neg x_5) \wedge (x_2 \vee x_5)$$

U takovýchto formulí si budeme všímat, zda některé logické ohodnocení vstupních proměnných má za výsledek hodnotu T (pravda) celé formule.

`Definice`: Problém *splnitelnosti SAT* se pro danou formuli v konjunktivním normálním tvaru ptá, zda je pro nějaké ohodnocení proměnných tato formule pravdivá.

Problém *MAX-SAT* se pro danou formuli v konjunktivním normálním tvaru ptá, kolik nejvíce klauzulí v ní lze splnit vhodným ohodnocením proměnných.

Věta 11.1. *Problém MAX-SAT lze formulovat v IP s polynomiálním počtem nerovností a proměnných.*

`Důkaz`: Použijeme binární proměnné z_i pro logické proměnné x_i dané formule Φ a další binární proměnné t_l pro klauzule c_l ve Φ. Účelovou funkcí bude

$$\max t_1 + t_2 + \ldots + t_k$$

za podmínek, ve kterých pro každou klauzuli $c_l = x_i \vee \neg x_j \vee \ldots$ píšeme

$$t_l \leq z_i + (1 - z_j) + \ldots \,.$$

Snadno vidíme, že hodnota $t_l = 1$ pro klauzuli c_l je přípustná právě tehdy, pokud aspoň jedna pozitivní proměnná v c_l má hodnotu 1 nebo pokud aspoň jedna negovaná proměnná v c_l má hodnotu 0. To jest právě když je klauzule c_l splněná v ohodnocení. □

`Komentář:` Pro výše uvedený příklad formule

$$\Phi_1 = (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4)$$

důkaz Věty 11.1 vyprodukuje úlohu IP ve tvaru

$$\begin{aligned} \max\ t_1 + t_2 + t_3 \quad & : \\ t_1 \quad & \leq \quad z_1 + 1 - z_2 + z_4 \\ t_2 \quad & \leq \quad 1 - z_1 + z_3 + z_4 \\ t_3 \quad & \leq \quad 1 - z_2 + 1 - z_3 + 1 - z_4 \,. \end{aligned}$$

Důsledek 11.2. *Problém SAT, a tudíž všechny problémy ve třídě NP, lze efektivně vyjádřit jako problém existence přípustného řešení pro úlohu IP.*

Důsledek 11.3. *Otázka existence přípustného řešení úlohy IP je NP-úplná a řešení úloh IP je NP-těžké.*

Uvědomte si, jak silné jsou uvedené důsledky. Nejen, že víme, že řešení úloh IP je v obecnosti efektivně nezvládnutelné, ale víme i, že každý problém ve třídě NP je nějakým způsobem zformulovatelný jako problém přípustnosti IP. Něco takového u mnoha příkladů vůbec není zřejmé, jak sami z vlastní zkušenosti poznáte...

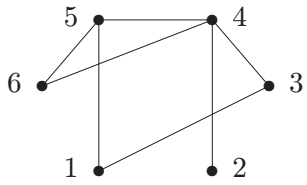
Doplňkové otázky

11.2 Některé grafové problémy

Pro ukázku se podíváme na IP formulace několika známých grafových problémů.

Příklad 11.4. *Zformulujme problém párování v grafu jako problém IP.*

Množina hran M je *párováním*, pokud žádné dvě z nich nesdílí vrchol. Pro ilustraci, následující graf má největší párování velikosti 3.



Řešení: V příkladech jako tento, kdy hledáme podmnožinu jistých vlastností, je přirozenou volbou přiřadit této podmnožině její charakteristický vektor (složený z binárních proměnných). V tomto případě tedy hraně $\{i, j\}$ přiřadíme binární proměnnou $z_{i,j}$ s významem, že $z_{i,j} = 1$ právě když $\{i, j\} \in M$. Aby vybraná podmnožina M byla párováním, může každý vrchol grafu náležet nejvýše jedné hraně M. Pro vrchol i se sousedy $j_1, \ldots, j_k$ tedy zformulujeme podmínku

$$z_{i,j_1} + z_{i,j_2} + \ldots + z_{i,j_k} \leq 1\,.$$

Účelovou funkcí – velikostí párování, je potom součet všech proměnných $z_{i,j}$ pro hrany grafu, případně i vážený součet.

Například pro graf na obrázku zformulujeme úlohu

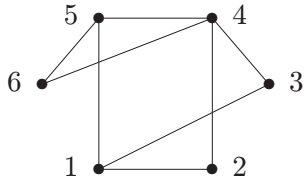
$$\begin{aligned}
\max\ z_{13} + z_{15} + z_{24} + z_{34} + z_{45} + z_{56} + z_{46} \quad &: \\
z_{13} + z_{15} \quad &\leq \quad 1 \\
z_{13} + z_{34} \quad &\leq \quad 1 \\
z_{24} + z_{34} + z_{45} + z_{46} \quad &\leq \quad 1 \\
z_{15} + z_{45} + z_{56} \quad &\leq \quad 1 \\
z_{46} + z_{56} \quad &\leq \quad 1 \\
z_{13}, z_{15}, z_{24}, z_{34}, z_{45}, z_{56}, z_{46} \quad &\in \quad \{0, 1\}\,.
\end{aligned}$$

□

V následujícím se podíváme na IP formulace grafových problémů, které jsou NP-těžké.

Příklad 11.5. *Zformulujme problém nezávislé množiny v grafu jako problém IP.*

Množina vrcholů I je *nezávislá*, pokud žádné dva z nich nejsou spojené hranou. Pro ilustraci, následující graf má největší nezávislou množinu velikosti 3. Dokážete ji najít?



Řešení: V tomto příkladě hledáme podmnožinu vrcholů, takže vrcholu i přiřadíme binární proměnnou z_i s významem, že $z_i = 1$ právě když $i \in I$. Podmínka nezávislosti se pak formuluje jako nerovnost $z_i + z_j \leq 1$ pro každou hranu $\{i, j\}$ daného grafu.

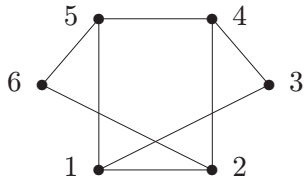
Například pro graf na obrázku zformulujeme úlohu

$$\begin{aligned}
\max\ z_1+z_2+z_3+\ z_4+z_5+z_6 \quad &: \\
z_1+z_2 \le 1, \quad z_1+z_3 &\le 1 \\
z_1+z_5 \le 1, \quad z_2+z_4 &\le 1 \\
z_3+z_4 \le 1, \quad z_6+z_4 &\le 1 \\
z_5+z_4 \le 1, \quad z_6+z_5 &\le 1 \\
z_1,z_2,z_3, \quad z_4,z_5,z_6 \quad &\in \{0,1\}\,.
\end{aligned}$$

□

Příklad 11.6. *Zformulujme problém dominující množiny v grafu jako problém IP.*

Dominující množina D v grafu G je taková $D \subseteq V(G)$, že každý vrchol G mimo D je spojený hranou s některým vrcholem v D. Najdete v následujícím grafu dominující množinu velikosti 2?



Řešení: Opět každému vrcholu i přiřadíme binární proměnnou z_i s významem, že $z_i = 1$ právě když $i \in D$. Podmínka dominující množiny se pak formuluje pro každý vrchol i se sousedy $j_1, \ldots, j_k$ jako nerovnost $z_i + z_{j_1} + \ldots + z_{j_k} \ge 1$ vyjadřující, že vrchol i je v D nebo některý jeho soused je v D.

Například pro graf na obrázku zformulujeme úlohu

$$\begin{aligned}
\min\ z_1+z_2+z_3+z_4+z_5+z_6 \quad &: \\
z_1+z_2+z_3+z_5 \quad &\ge \quad 1 \\
z_2+z_1+z_4+z_6 \quad &\ge \quad 1 \\
z_3+z_1+z_4 \quad &\ge \quad 1 \\
z_4+z_2+z_3+z_5 \quad &\ge \quad 1 \\
z_5+z_1+z_4+z_6 \quad &\ge \quad 1 \\
z_6+z_2+z_5 \quad &\ge \quad 1 \\
z_1,z_2,z_3,z_4,z_5,z_6 \quad &\in \quad \{0,1\}\,.
\end{aligned}$$

□

Předchozí příklady byly docela jednoduché a přímočaré, že? Co ale třeba s podobně jednoduše vypadajícím problémem barvení grafu? *Obarvením* grafu rozumíme přiřazení přirozených čísel (*barev*) vrcholům grafu tak, aby sousední vrcholy dostaly různé barvy. *Barevností grafu* pak je nejmenší počet barev, kterým lze daný graf obarvit.

Příklad 11.7. *Zformulujme problém barevnosti grafu jako problém IP.*

Přirozenou první myšlenkou většiny čtenářů asi bude použít celočíselné proměnné pro barvy jednotlivých vrcholů. (Ponechme stranou otázku omezenosti proměnných, neboť počet barev lze vždy omezit počtem vrcholů.) Jak ale vyjádříme, že sousední vrcholy mají různé barvy? Rámec formulace úloh IP nám totiž nedává možnost přímo napsat $z_i \ne z_j$.

Pokud bychom pro hranu $\{i,j\}$ chtěli například napsat $z_i \ge z_j + 1$, museli bychom připustit i možnost $z_i \le z_j - 1$, ale my neumíme přímo vyjádřit disjunkci podmínek. (Blíže viz Oddíl 12.2.) Využijeme raději podobnost problému barvení s nezávislou množinou – vrcholy stejné barvy musí být nezávislé. Obarvení grafu k barvami tudíž znamená nalezení rozkladu množiny vrcholů na k nezávislých podmnožin.

Řešení: Pro každou potenciální barvu c (je třeba počítat s počtem barev až rovným počtu vrcholů n) zavedeme sadu binárních proměnných $z_{1,c}, \ldots, z_{n,c}$, kde $z_{i,c} = 1$ znamená, že vrchol i může být obarven barvou c. Pro každé c pak přidáme na $z_{1,c}, \ldots, z_{n,c}$ podmínky vyjadřující nezávislou množinu, jako v Příkladě 11.5

$$\forall c = 1, 2, \ldots, n, \ \forall \{i, j\} \in E(G) : \quad z_{i,c} + z_{j,c} \leq 1 \,.$$

Dále musíme vyjádřit požadavek, že každý vrchol dostane přidělenu jednu barvu

$$\forall k = 1, 2, \ldots, n : \quad z_{k,1} + z_{k,2} + \ldots + z_{k,n} = 1 \,.$$

Posledním poněkud trikovým bodem je vyjádření počtu barev, které celkem použijeme. Pro to musíme zavést další binární proměnné $t_1, \ldots, t_n$ indikující ty z barev, které byly skutečně použity, a vyjádříme účel

$$\begin{aligned} & \min \ t_1 + \ldots + t_n \\ \forall c = 1, 2, \ldots, n : \quad & z_{1,c} + z_{2,c} + \ldots + z_{n,c} \quad \leq \quad n \cdot t_c \,. \end{aligned}$$

`Komentář:` Všimněte si dobře, jakým způsobem jsme v posledních vztazích "spočítali", kolik barev c je vlastně při barvení našeho grafu použito: Pokud $t_c = 0$, barva c nemůže být použita vůbec, pokud naopak $t_c = 1$, lze c použít až na všech n vrcholech. Jedná se o docela užitečný trik k zapamatování. □

Doplňkové otázky

(11.2.1) *Jak v IP zformulujete problém největší kliky v grafu?*

(11.2.2) *Jak v IP zformulujete problém nezávislé dominující množiny? (Například v Příkladě 11.6 jsou různé závislé i jedna nezávislá dominující množina velikosti 2.)*

*(11.2.3)** *Jak byste v IP zformulovali problém MST (minimální kostry)? (Třebaže je MST velmi snadno řešitelný problém, popsat acyklickou podmnožinu hran je v IP obtížné, že?)*

11.3 Problém obchodního cestujícího

Klasickou dobře známou úlohou diskrétní optimalizace je tzv. problém obchodního cestujícího (TSP), motivovaný následovně: Úkolem cestujícího je obejít daných n měst (v libovolném pořadí, s návratem do výchozího) tak, aby byla celková délka či cena cesty minimální. Tento tradiční problém neopomineme ani v našem textu.

`Definice`: (Orientovaný) sled v grafu G je posloupnost $v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$, kde vždy hrana e_i, $1 \leq i \leq k$ vede z vrcholu v_{i-1} do v_i.

Definice 11.8. ***Obecný problém TSP*** (Obchodního cestujícího v grafu)
V nejobecnější grafové formulaci se problém obchodního cestujícího zadá následovně:

Vstup: Orientovaný ohodnocený souvislý graf G, s ohodnocením hran $w : E(G) \to \mathbb{R}^+$.
Výstup: Najít uzavřený orientovaný sled v G, ve kterém je součet ohodnocení (*délek*) hran nejmenší možný.

`Poznámka`: Uvědomme si, že optimální řešení obecného problému TSP může opakovat jeden vrchol vícekrát, například pokud graf G je stromem. Obvykle se však při formulaci problému TSP explicitně požaduje, aby se žádný vrchol ve sledu neopakoval, neboli se hledá tzv. Hamiltonovský cyklus grafu. V této "neopakovací" formulaci také graf G obvykle bývá úplný.

Příklad 11.9. *Zformulujte problém TSP na orientovaném grafu G bez povoleného opakování vrcholů ve sledu jako úlohu IP.*

Nechť $V(G) = \{1, 2, ..., n\}$, pak každé orientované hraně $(i, j) \in E(G)$ přiřadíme proměnnou $z_{i,j} \in \{0, 1\}$. Jelikož opakování vrcholů je zakázáno, hledáme formulaci popisující Hamiltonovský cyklus v G.

- Do každého vrcholu k jednou přijdeme a jednou z něj odejdeme

$$\sum\nolimits_{i:(i,k)\in E(G)} z_{i,k} = 1, \qquad \sum\nolimits_{i:(k,i)\in E(G)} z_{k,i} = 1\,.$$

  Pokud však má G dostatek vrcholů, tyto podmínky splňuje i kolekce více disjunktních cyklů, že?

- Navíc tedy musíme vyloučit případ, kdy by se náš cyklus "uzavřel" ještě před navštívením všech vrcholů. To znamená, že na každé podmnožině vrcholů $W \in V(G)$, $2 \leq |W| \leq n - 2$ zabráníme vytvoření *podcyklu* nerovnostmi

$$\sum\nolimits_{(i,j)\in E(G),\, i,j\in W} z_{i,j} \leq |W| - 1\,, \tag{12}$$

  které zaručí pokračování cyklu mimo W.

Takto sestavených nerovností pro TSP je bohužel zhruba 2^n, takže je nelze ani rozumně zapsat do paměti. Přesto se tato formulace TSP v praxi používá tak, že nerovnosti v (12) se vygenerují "na žádost", tj. například pro jeden konkrétní podcyklus v částečném řešení přidáme příslušnou vylučující nerovnost. Ukazuje se, že nakonec je třeba využít obvykle jen rozumně malou část ze všech těchto podcyklových nerovností. □

`Poznámka` V poválečné době bylo velkým matematickým úspěchem vyřešit problém TSP na 49 hlavních městech kontinentálních států USA. S dnešními vylepšenými algoritmy a výkonnými počítači jsme schopni řešit obecné problémy TSP až na tisících vrcholech. Přesto tento optimalizační problém zůstává velmi obtížný a také velmi populární.

Aproximace Euklidovského TSP

Původní motivace problému TSP nese oproti abstraktní formulaci jednu podstatnou informaci navíc – délky hran mezi městy splňují trojúhelníkovou nerovnost. Toho lze s výhodou použít při aproximaci řešení TSP.

`Definice`: Problém TSP se nazývá *Euklidovský*, pokud délky hran grafu splňují trojúhelníkovou nerovnost (graf musí být úplný). Speciálně tedy pokud jsou délky dány Euklidovskou metrikou.

Příklad 11.10. *Ukážeme, jak lze problém Euklidovského neorientovaného TSP přibližně vyřešit v polynomiálním čase s cenou nejvýše dvojnásobnou od optimálního řešení.*

Řešení: Na (neorientovaném) grafu G najdeme minimální kostru $T \subseteq G$ (například hladovým algoritmem v čase $O(m \cdot \log m)$, $m = |E(G)|$). Pak "zdvojením" každé hrany T dostaneme uzavřený sled délky dvojnásobku T.

Pokud je vyloučeno opakování vrcholů, tak nakonec tento sled "narovnáme" přeskočením opakovaných vrcholů. Podle trojúhelníkové nerovnosti si narovnáním sled neprodloužíme, ale v praxi většinou zkrátíme. □

Doplňkové otázky

(11.3.1) *Kdy řešení problému TSP není grafovou kružnicí, přestože se vrcholy neopakují?*

(11.3.2) *Kdy sled v řešení problému TSP je nucen opakovat vrcholy?*

(11.3.3) *Kdy problém TSP nemá žádné přípustné řešení, třebaže je opakování vrcholů povoleno?*

(11.3.4) *Navrhněte, jak řešení Příkladu 11.10 vylepšit, aby zaručený aproximační faktor byl $\frac{3}{2}$ místo 2.*

Rozšiřující studium

Problém SAT a běžné třídy výpočetní složitosti (téma, do nějž patří i třída NP a fenomén NP-úplnosti) čtenář najde podrobně popsané v téměř kterékoliv učebnici teoretické informatiky či výpočetní složitosti. My doporučujeme v češtině například [4, Kapitoly 2,7,8]. Konkrétně složitosti grafových problémů se věnuje [4, Kapitola 8] a třeba [10, Chapters 3,4,7].

Problém obchodního cestujícího se ukazuje jako výrazně odlišný od ostatních úloh MIP při praktickém řešení a většinou vyžaduje speciální algoritmy. Takovým je věnována například [7, Section II.6.3], kde lze nalézt i různé jeho vhodné relaxace a přibližná řešení. Více o historii i současném stavu problematiky řešení TSP je možno najít na hezkých stránkách [W. Cook, http://www.tsp.gatech.edu/].

12 Umění formulace úloh MIP

Úvod

Již několikrát jsme viděli, že formulace úloh diskrétní optimalizace jako úloh MIP nebývá tak jednoduchá a skrývá v sobě někdy nečekaná úskalí. Jedním z nich je třeba nutnost formulovat exponenciálně mnoho omezení, jiným třeba požadavek formulace "A nebo B", což přímo nejde. V této lekci si ukážeme hlubší umění správné formulace diskrétních úloh v MIP na několika hezkých příkladech.

Cíle

Cílem je nyní ukázat čtenáři několik užitečných triků pokročilého umění formulace úloh MIP. Dále jsou ukázány celočíselné polyedry a vysvětlen jejich význam. Obsah lekce bude ještě obohacen na základě budoucí výuky autora.

12.1 Více o barevnosti grafu

Pro ilustraci některých pokročilých postupů formulace úloh MIP si zkusme nejprve zapsat problém barevnosti grafu (Příklad 11.7) úplně jinou formulací:

Příklad 12.1. *Zformulujme problém barevnosti grafu jako problém MIP, kde barvy přiřazené vrcholům budou přímo (jako čísla) uloženy v reálných proměnných. Zachováváme přitom požadavek, aby se hodnoty sousedních barev lišili alespoň o* 1.

Řešení: V souladu se zadáním každému vrcholu $i = 1, 2, \ldots, n$ grafu G přiřadíme reálnou proměnnou x_i. Nechť $\{i, j\}$ je hranou G. Jak pak v rámci formulace MIP vyjádříme požadavek $|x_i - x_j| \geq 1$? To je, zdá se být, hlavním problémem našeho příkladu.

Trik je následovný – zavedeme dodatečnou binární proměnnou $z_{i,j}$ pro každou hranu $\{i, j\}$ a požadavek na barvy rozepíšeme jako

$$\begin{aligned} x_i &\leq x_j - 1 + nz_{i,j}\,, \\ x_i &\geq x_j + 1 - n(1 - z_{i,j})\,. \end{aligned}$$

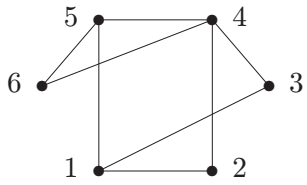
(Hodnota proměnné $z_{i,j}$ nám takto určuje, která z obou barev na hraně $\{i, j\}$ bude větší.) Skutečně, pokud $z_{i,j} = 0$, tak z první nerovnosti plyne $x_i - x_j \leq -1$, a pokud

$z_{i,j} = 1$, tak z druhé nerovnosti plyne $x_i - x_j \geq 1$. Opačný směr ekvivalence je také zřejmý.

Nakonec vyjádříme celkový počet použitých barev jako maximum z hodnot nabytých proměnnými $\vec{x}$ plus 1 (nezapomeňme, že hodnoty $\vec{x}$ začínají od 0). Takže zapíšeme

$$\begin{aligned} & \min\ y \quad : \\ \forall i = 1, 2, \ldots, n : \ & x_i + 1 \ \leq \ y\,. \end{aligned}$$

Například pro graf na obrázku zapíšeme úlohu MIP



$$\begin{aligned}
& \min\ y : \\
& x_1, x_2, x_3, x_4, x_5, x_6 \quad \leq y - 1 \\
x_1 \ & \leq x_2 - 1 + 6z_{12}, \qquad x_1 \ \geq x_2 + 1 - 6(1 - z_{12}) \\
x_1 \ & \leq x_3 - 1 + 6z_{13}, \qquad x_1 \ \geq x_3 + 1 - 6(1 - z_{13}) \\
x_2 \ & \leq x_4 - 1 + 6z_{24}, \qquad x_2 \ \geq x_4 + 1 - 6(1 - z_{24}) \\
x_3 \ & \leq x_3 - 1 + 6z_{34}, \qquad x_3 \ \geq x_4 + 1 - 6(1 - z_{34}) \\
x_4 \ & \leq x_5 - 1 + 6z_{45}, \qquad x_4 \ \geq x_5 + 1 - 6(1 - z_{45}) \\
x_1 \ & \leq x_5 - 1 + 6z_{15}, \qquad x_1 \ \geq x_5 + 1 - 6(1 - z_{15}) \\
x_6 \ & \leq x_5 - 1 + 6z_{65}, \qquad x_6 \ \geq x_5 + 1 - 6(1 - z_{65}) \\
x_4 \ & \leq x_6 - 1 + 6z_{46}, \qquad x_4 \ \geq x_6 + 1 - 6(1 - z_{46}) \\
& x_1, x_2, x_3, x_4, x_5, x_6 \quad \geq 0 \\
& z_1, z_2, z_3, z_4, z_5, z_6 \quad \in \{0, 1\}\,.
\end{aligned}$$

Poněkud stranou zatím zůstala důležitá otázka, zda jsme naší úlohou MIP zapsali skutečně problém klasické barevnosti grafu. Vždyť jsme výrazně rozšířili množinu přípustných řešení na reálná čísla. Přesto lze dokázat, že optimálním výsledkem naší úlohy je skutečně klasická (celočíselná) barevnost. □

Důležitým přínosem nových lineárních formulací kombinatorických problémů je také možnost jejich zobecňování (či relaxace) umožněné novým pohledem. Například výše uvedená formulace grafové barevnosti s reálnými "barvami" přímo motivuje následující (již dříve známou) neceločíselnou relaxaci barevnosti grafu.

`Definice`: *Cirkulární barevností* grafu G rozumíme nejmenší reálné číslo $f > 0$ takové, že vrcholy G lze zobrazit na kruhový interval délky (tj. obvodu) f tak, že sousední vrcholy G mají na obvodu kruhu vzdálenost alespoň 1.

Příklad 12.2. *Zformulujme problém cirkulární barevnosti grafu jako problém MIP, kde barvy přiřazené vrcholům budou přímo uloženy v reálných proměnných.*

Na začátek si blíže osvětlíme význam kruhového intervalu. Dvě čísla (body) s, t jsou na kružnici o obvodu f ve vzdálenosti alespoň d pokud zároveň $|s - t| \geq d$ a $|s - t| \leq f - d$ (vzdálenosti měřené v obou směrech na kružnici). Podmínka vzdálenosti hodnot sousedních vrcholů G $|x_i - x_j| \geq 1$ byla zformulována již v Příkladě 12.1. Pro

druhou podmínku vzdálenosti na kruhu se v tomto případě ukazuje jako lepší ji přímo neformulovat, ale místo toho implicitně definovat obvod f kruhového intervalu jako minimum z hodnot $|x_i - x_j| + 1$ přes všechny hrany G.

Řešení: S výhodou využijeme již zapsané podmínky z Příkladu 12.1

$$\begin{aligned} \forall \{i,j\} \in E(G): \; x_i &\leq x_j - 1 + n z_{i,j} \\ x_i &\geq x_j + 1 - n(1 - z_{i,j}) \\ z_{i,j} &\in \{0,1\} \end{aligned}$$

a přidáme účelovou funkci spolu s podmínkami

$$\begin{aligned} & & \min\; y &: \\ \forall i = 1,2,\ldots,n: \; & x_i - x_j + 1 &\leq\; y & \\ & x_j - x_i + 1 &\leq\; y\,. & \end{aligned} \tag{13}$$

Pokud cirkulární barevnost G je f, tak výše zapsaný systém podmínek má přípustné řešení s $y = f$ dané tímto cirkulárním obarvením.

Naopak pokud vezmeme přípustné řešení výše zapsaného systému podmínek a položíme $f = y$, pak snadno přiřadíme vrcholu i grafu G barvu $c_i = x_i \bmod (f+1)$. Potom bude splněno $|c_i - c_j| \geq 1$ a také $|c_i - c_j| \leq y - 1 = f - 1$ pro každou hranu $\{i,j\} \in E(G)$, neboli se bude jednat o platné cirkulární obarvení. □

Doplňkové otázky

(12.1.1) *Proč formulace Příkladu 12.1 určuje přesně klasicku barevnost?*

(12.1.2) *Zformuluje slovně, co vztah (13) v Příkladě 12.2 říkají o optimálním y.*

(12.1.3) *Jakou úlohu dostaneme, pokud v Příkladě 12.2 budeme požadovat celočíselnost barev $\vec{x}$?*

12.2 Užitečné triky formulace

Modelování disjunkce

Trik použitý v Příkladě 12.1 lze zobecnit na modelování jakéhokoliv počtu *disjunktivních podmínek* v úlohách MIP. Stručně řečeno, dokážeme vyjádřit podmínky ve stylu "platí toto nebo toto nebo toto..." (nebo zde není exkluzivní).

Tvrzení 12.3. *Mějme úlohu MIP s omezenou množinou přípustných řešení Q. Nechť $P_1, \ldots, P_k$ jsou polyedry určené nerovnostmi $\boldsymbol{C}^j \cdot \vec{x} \leq \vec{d^j}$ pro $j = 1, \ldots, k$ (kde vektor $\vec{x}$ zahrnuje všechny proměnné úlohy). Pak úlohu s množinou přípustných řešení $Q \cap (P_1 \cup \ldots \cup P_k)$ lze vyjádřit jako úlohu MIP.*

Důkaz: Zvolme nejprve dostatečně velkou konstantu K. Zavedeme nové binární proměnné $z_1, \ldots, z_k$ spolu s přidanými vztahy

$$\begin{aligned} z_1 + z_2 + \ldots + z_k &= k - 1 \\ \forall j = 1,2,\ldots,k: \; \boldsymbol{C}^j \cdot \vec{x} &\leq \vec{d^j} + K \cdot \vec{1} \cdot z_j \end{aligned} \tag{14}$$

Pokud vektory $\vec{x} \in Q$ a $\vec{z}$ splňují vztahy (14), tak jedno $z_i = 0$, neboli $\vec{x}$ splňuje $\boldsymbol{C}^i \cdot \vec{x} \leq \vec{d^i}$ a $\vec{x} \in Q \cap P_i$.

Na druhou stranu pokud $\vec{x} \in Q \cap (P_1 \cup \ldots \cup P_k)$, tak pro některé i je $\vec{x} \in Q \cap P_i$. Zvolíme $z_i = 0$ a $z_j = 1$ pro $j \neq i$. Pak z předpokladu $\vec{x} \in P_i$ bude platit $\boldsymbol{C}^i \cdot \vec{x} \leq \vec{d^i}$ a navíc pro vhodnou volbu konstany K bude $\boldsymbol{C}^j \cdot \vec{x} \leq \vec{d^j} + K \cdot \vec{1}$ platit pro všechna $j = 1, \ldots, k$. Tudíž (14) bude splněno. □

Krátká formulace TSP

Další moc hezký trik formulace úloh MIP je ukázán na zápise problému TSP polynomiálně mnoha nerovnostmi. Přirozená formulace v Příkladu 11.9 vyžaduje exponenciální počet nerovností a pozdější objevení výrazně kratšího zápisu úlohy bylo docela velkým překvapením.

Tvrzení 12.4. *Problém obchodního cestujícího na orientovaném grafu G s n vrcholy a ohodnocením hran $w : E(G) \to \mathbb{R}^+$ lze vyjádřit následujícími polynomiálně mnoha nerovnostmi jako úlohu MIP s binárními proměnnými $z_{i,j}$ a reálnými u_i:*

$$\begin{aligned} \forall k = 1, 2, \ldots, n: \quad & \textstyle\sum_{i:(i,k)\in E(G)} z_{i,k} && = 1 \\ \forall k = 1, 2, \ldots, n: \quad & \textstyle\sum_{i:(k,i)\in E(G)} z_{k,i} && = 1 \\ \forall (i,j) \in E(G),\ i,j > 1: \quad & u_i - u_j + n \cdot z_{i,j} && \leq n - 1 \\ & \vec{u} && \geq 0 \\ & \vec{z} && \in \{0,1\}^* \end{aligned} \tag{15}$$

Důkaz: Předpokládejme pro spor, že přípustné hodnoty proměnných $z_{i,j} = 1$ nevyjadřují v grafu G Hamiltonovský cyklus. Jelikož do každého vrcholu přichází jedna hrana se $z_{i,j} = 1$ a jedna taková odchází, množina hran se $z_{i,j} = 1$ je tvořena více cykly a alespoň jeden z nich, označme jej C, neprochází vrcholem 1. Sečteme-li nerovnosti (15) přes hrany cyklu C, proměnné u_i se vyruší a vyjde

$$|V(C)| \cdot n \cdot 1 \leq (n-1) \cdot |V(C)| \ ,$$

což je spor.

Naopak máme dokázat, že každý Hamiltonovský cyklus C v G (tj. procházející přes všechny vrcholy) je přípustný. Nechť vrcholy G následují v cyklu C v pořadí $v_1 = 1$, v_2, $v_3, ..., v_n$. Položíme $u_i = l$, kde $i = v_l$ (tj. podle pořadí na cyklu). Pak pro každou hranu $(i,j) \in E(C)$, $i,j > 1$ platí $u_i - u_j = l - (l+1) = -1$, takže

$$u_i - u_j + n \cdot 1 \leq -1 + n \cdot 1 = n - 1.$$

Pro kteroukoliv jinou hranu přirozeně platí $u_i - u_j + n \cdot 0 \leq n - 1$. Tím jsme dokončili důkaz správnosti formulace (15) pro problém TSP. □

Poznámka: Naneštěstí se uvedená krátká formulace ukazuje jako velmi nevhodná k praktickému řešení problému TSP, neboť její lineární relaxace jen velmi hrubě aproximuje množinu přípustných celočíselných řešení. Proto má tento výsledek povětšinou jen teoretický význam.

Doplňkové otázky

(12.2.1) *Jak byste v rámci MIP formulovali podmínku exkluzivní disjunkce? (Tj. nastane jen právě jedna z možností.))*

(12.2.2) *Jak byste využili triku Tvrzení 12.4 k MIP formulaci problému MST?*

12.3 Celočíselné polyedry

Pro úvod začneme s přirozenou (a trochu upravenou) formulací problému toků v síti jako úlohy LP.

Definice: $\boldsymbol{A}$ je *vrcholově–hranová incidenční matice* orientovaného grafu G, pokud její složky (indexované vrcholy × hranami) jsou

$$a_{i,(i,j)} = 1 \text{ pro } (i,j) \in E(G), \quad a_{i,(j,i)} = -1 \text{ pro } (j,i) \in E(G), \quad a_{i,e} = 0 \text{ jinak.}$$

Mějme síť $(G, z, s, \vec{c})$, kde G je orientovaný graf, z zdroj, s stok a $\vec{c}$ udává kapacity jednotlivých hran. (Definice 2.1) Doplňme graf G o "zpětnou" hranu ze stoku s do zdroje z bez omezení kapacity a přiřaďme proměnné $x_{(i,j)}$ tokům na jednotlivých hranách. Pak úlohu hledání maximálního toku zapíšeme jako:

$$\begin{array}{rcl} \max\ x_{(s,z)} & : & \\ \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{I} \end{pmatrix} \cdot \vec{x} & = & \begin{pmatrix} 0 \\ \vec{c} \end{pmatrix} \\ & \leq & \\ \vec{x} & \geq & 0 \end{array}$$

`Definice`: Polyedr P je *celočíselný*, pokud každá jeho neprázdná stěna obsahuje vektor se všemi celočíselnými složkami.

`Komentář:` Omezený polyedr P je celočíselný podle této definice, pokud každý jeho vrchol je vektorem s celočíselnými složkami.

`Fakt`: Z celočíselnosti řešení Algoritmu 2.6 pro tok v síti v případě celočíselných kapacit $\vec{c}$ (nepřímo) vyplývá, že polyedr přípustných toků v dané síti je celočíselný.

Uvedený fakt není náhodný, je ve skutečnosti jen speciálním případem širšího fenoménu popsaného následně.

Totálně unimodulární matice

`Definice`: Řekneme, že matice $\boldsymbol{A}$ je *totálně unimodulární* (TU) pokud každá její čtvercová podmatice má determinant 0, 1 nebo -1.

`Komentář:` Není těžké zjistit, že složky totálně unimodulární matice mohou být jen 0, 1 nebo -1, ale to není dostačující. Například jednotková matice $\boldsymbol{I}$ je totálně unimodulární a taková je třeba i matice

$$\begin{pmatrix} -1 & 1 & 0 & 0 & 1 \\ 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 1 \\ 1 & 0 & 0 & 1 & -1 \end{pmatrix},$$

která má mezi TU maticemi výsadní postavení, jehož význam přesahuje rámec našeho textu.

Význam totálně unimodulárních matic v oblasti celočíselných mnohostěnů je naznačen následující vlastností, která přímo plyne ze známého "determinantového" vztahu pro inverzní matici.

`Fakt`: Inverze regulární totálně unimodulární matice je celočíselná matice.

Z uvedeného faktu a znalosti souvislosti bázických řešení LP s vrcholy polyedru (Lema 6.3) není těžké odvodit, že pro TU matici $\boldsymbol{A}$ a celočíselné $\vec{b}$ je polyedr určený vztahy $\boldsymbol{A} \cdot \vec{x} \leq \vec{b}$, $\vec{x} \geq 0$ celočíselný. V obecnosti dokonce lze tvrdit (zde bez důkazu):

Věta 12.5. *Matice $\boldsymbol{A}$ je totálně unimodulární, právě když pro každé $\vec{b} \in \mathbb{Z}^m$ je polyedr určený vztahy $\boldsymbol{A} \cdot \vec{x} \leq \vec{b}$, $\vec{x} \geq 0$ celočíselný.*

Souvislost totálně unimodulárních matic s toky v sítích se ukáže následovně.

Tvrzení 12.6. *Nechť $\boldsymbol{A}$ je vrcholově–hranová incidenční matice orientovaného grafu G. Pak $\boldsymbol{A}$ je totálně unimodulární.*

`Důkaz`: Vezměme čtvercovou podmatici $\boldsymbol{C}$ v $\boldsymbol{A}$. Tvrzení je jasné, pokud $\boldsymbol{C}$ má jediný sloupec. Dále postupujeme indukcí podle velikosti $\boldsymbol{C}$.

Pokud $\boldsymbol{C}$ je singulární, má determinant 0. Tato možnost mimo jiné nastává, pokud každý sloupec $\boldsymbol{C}$ obsahuje obě nenulové souřadnice (1 a -1) z příslušného sloupce matice $\boldsymbol{A}$. Jinak najdeme sloupec v $\boldsymbol{C}$ mající právě jednu nenulovou souřadnici. Podle tohoto sloupce determinant rozvineme na jediný poddeterminant, jehož hodnota je ± 1 z indukčního předpokladu, tudíž to samé platí pro $|\boldsymbol{C}|$. □

Párování v grafu

Mimo TU matice existují i další zajímavé případy celočíselných polyedrů vycházejících z grafových problémů. Jedním z nich je tzv. mnohostěn párování. Připomínáme, že *párování* v grafu je taková podmnožina hran, ve které žádné dvě hrany nesdílejí vrchol. Párování je *perfektní*, pokud obsahuje všechny vrcholy grafu. (Mimo jiné musí mít takový graf sudý počet vrcholů.)

`Definice`: Nechť $\vec{x} \in \{0,1\}^*$ je charakteristický vektor podmnožin hran grafu G. *Mnohostěn párování* grafu G je konvexní obal všech takových charakteristických vektorů $\vec{x}$, které odpovídají párováním v G. *Mnohostěn perfektních párování* grafu G je konvexní obal charakteristických vektorů perfektních párování v G.

Vzpomeňme si na Příklad 11.4 ukazující IP formulaci problému párování v grafu. Je zřejmé, že pokud bychom k oné formulaci udělali LP-relaxaci, dostaneme mnohem více řešení než jen ta z mnohostěnu párování.

> `Komentář:` Například ohodnocení všech proměnných hran na liché kružnici číslem $\frac{1}{2}$ vyhovuje nerovnostem Příkladu 11.4, ale takové ohodnocení zřejmě nemá nic společného se skutečnými párováními. Jak vidíme při bližším pohledu, problémy dělají liché podmnožiny vrcholů indukující podgrafy s plně "nasycenými" ohodnocenými hranami, což je věc nemožná pro skutečné párování grafu.

Věta 12.7. *Mnohostěn perfektních párování grafu G s n vrcholy je vyjádřen následujícími vztahy:*

$$\forall i = 1,2,\ldots,n: \quad \sum_{e\in E(G):\, e=\{i,j\}} x_e = 1 \tag{16}$$

$$\forall\, W \subseteq V(G), |W| \text{ liché}: \quad \sum_{e\in E(G):\, e=\{i,j\}\subseteq W} x_e \leq \lfloor |W|/2 \rfloor \tag{17}$$

$$\vec{x} \geq 0$$

Mnohostěn párování grafu G je vyjádřen stejnými vztahy, jen vztah (16) je zapsán s nerovností $\leq$ místo $=$.

`Důkaz` (náznak): Nejprve si krátce ukažme, jak z platnosti věty pro perfektní párování plyne její platnost pro všechna párování:

.....

Nyní dokážeme, že vztahy (16),(17) plně popisují mnohostěn perfektních párování grafu G. V jednom směru potřebujeme dokázat, že každé perfektní párování splňuje vztahy (16) a (17). Pro (16) to platí z definice perfektního párování. Pokud W je lichá podmnožina vrcholů, tak aspoň jeden z vrcholů musí zůstat uvnitř W nespárovaný, a proto plyne (17).

Naopak vezměme vektor $\vec{x}$, který splňuje výše uvedené vztahy a zároveň je krajním bodem mnohostěnu perfektních párování. Předpokládejme pro spor, že $\vec{x}$ není celočíselný. Hranám e grafu G, které mají necelé hodnoty x_e, říkejme *šedé hrany*. Pro šedou kružnici C sudé délky v G nazveme ε-alternací $\vec{x}$ takové ohodnocení $\vec{x}'$, které na i-té hraně v pořadí na kružnici C nabývá $x'_{e_i} = x_{e_i} + (-1)^i\varepsilon$ a mimo C zůstává stejné jako $\vec{x}$.

Tvrzení 12.8. *Za výše uvedených předpokladů v G existuje sudá šedá kružnice C a konstanta $\delta > 0$ takové, že všechny ε-alternace C jsou přípustné ve vztazích (16),(17) pro $-\delta < \varepsilon < \delta$.*

Tvrzení dokazujeme indukcí podle n, použitím kontrakcí lichých "nasycených" podmnožin...

Nyní již snadno dokončíme důkaz celé věty. Vektor $\vec{x}$ je totiž konvexní kombinací svých ε-alternace a $-\varepsilon$-alternace, které jsou přípustné v úloze pro dostatečně malé $\varepsilon > 0$. To je ale ve sporu s předpokladem, že $\vec{x}$ je krajním bodem polyedru. □

Mnohostěn perfektního grafu

Jiný známý příklad kombinatorických celočíselných mnohostěnů vychází z popisu klik v jistých grafech. Připomínáme, že *klika* v grafu je jiný název pro inkluzí maximální úplný podgraf.

`Definice`: Graf G je *perfektní*, pokud v každém jeho indukovaném podgrafu je barevnost rovna velikosti největší kliky.

> `Komentář:` Nepleťme si pojem perfektního grafu s perfektním párováním, není mezi nimi žádná souvislost. Příklady perfektních grafů jsou bipartitní grafy (barevnost 2), intervalové grafy (určené průniky intervalů na přímce), nebo tzv. chordální grafy (bez indukovaných kružnic delších než 3).

`Definice`: Nechť $\boldsymbol{K}$ je incidenční matice klik a vrcholů perfektního grafu G. (Tj. $K_{i,j} = 1$ právě když vrchol j náleží do i-té kliky.) Pak *mnohostěn perfektního grafu* G je určený vztahy

$$\begin{aligned} \boldsymbol{K} \cdot \vec{x} &\leq 1 \\ \vec{x} &\geq 0\,. \end{aligned}$$

Věta 12.9. *Mnohostěn perfektního grafu je vždy celočíselný.*

Doplňkové otázky

(12.3.1) *Proč pro definici celočíselného polyedru musíme mluvit o všech jeho stěnách, nestačí jen definovat vrcholy jako celočíselné?*

(12.3.2) *Zdůvodněte podrobně, jak z celočíselnosti řešení Algoritmu 2.6 plyne celočíselnost polyedru toků.*

(12.3.3) *Pokuste se vysvětlit, proč jsou bipartitní, intervalové a chordální grafy perfektními.*

(12.3.4)

12.4 Neúplné formulace úloh MIP

Iterativní vylepšování formulace MIP novými podmínkami na základě předchozích relaxovaných řešení.....

Doplňkové otázky

Rozšiřující studium

Vlastně všechny námi citované učebnice optimalizace obsahují množství různých příkladů a triků formulace úloh MIP, takže čtenář má mnoho materiálu k hlubšímu studiu. Konkrétně k problematice celočíselných mnohostěnů doporučujeme: párování [10, Chapter 3,5], perfektní grafy [10, Chapter 7], totálně unimodulární matice [10, Chapter 8]. Více různých poznatků lze také nalézt v [7, Chapters III.1,2].

13 Pokročilá kombinatorická optimalizace

Úvod

Na závěr textu si přiblížíme ukázky řešení hlubších úloh kombinatorické optimalizace.....

matroid intersection, submodular function minimization

Cíle

.....

Tato lekce bude připravena na základě budoucí výuky autora, podle látky [7, Part III].

13.1 Průnik matroidů

Návaznost na hladový algoritmus, Oddíl 1.3, rozšíření problému na hledání optimální množiny, která je zároveň nezávislá ve více matroidech – tzv. problém *průniku matroidů*.

`Fakt`: Problém průniku tří matroidů je NP-úplný.

`Fakt`: Edmondsův algoritmus řeší problém průniku dvou matroidů v polynomiálním čase.

.....

Doplňkové otázky

13.2 Minimalizace submodulární funkce

`Definice`: Funkce $f : 2^E \to \mathbb{Z}$ je *submodulární*, pokud pro všechna $X, Y \subseteq E$ platí

$$f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y)\,.$$

`Fakt`: Pro problém minimalizace symetrické submodulární funkce existuje polynomiální algoritmus.

Doplňkové otázky

Rozšiřující studium

.....

matroidy a jejich průnik [10, Chapter 10], [7, Chapter III.3],
submodulární funkce [7, Chapter III.3],

Část IV

Klíč k řešení úloh

(1.1.1) V jistém časovém okamžiku vidíme, že se zpracovávají 4 úkoly najednou, a proto méně než 4 pracovníci nestačí.

(1.1.2) ...

(1.2.1) Nalezneme kostru s největším celkovým ohodnocením.

(1.2.2) Není potřeba na začátku všechny (mnoho) hrany seřadit, seřazují se jen některé hrany potřebné v průběhu algoritmu.

(1.2.3) Hlavně nějaký rychlý typ haldy pro ukládání seznamu hran vycházejících ze současné komponenty ven a vybírání nejmenší z nich.

(1.3.1) Jakkoliv špatně, pro každý zvolený počet barev se dá nalézt špatný příklad, zkuste si to...

(1.3.2) V pořadí jakéhokoliv souvislého prohledávání našeho grafu.

(1.3.3) Jednoduše, prostě hladový algoritmus spustíme a on nikdy nepoužije barvu vyšší než $k+1$, neboť každý vrchol má jen k (možná) obarvených sousedů.

(1.3.4)* Třeba pro graf, který je hvězdou s cestami délky 2 jako paprsky. Hladově se totiž nejprve vybere centrální vrchol velkého stupně, a pak také všichni jeho sousedé stupňů 2. Ten centrální vrchol tudíž bude přebývat, nebude optimální.

(1.4.1) Univerzem jsou vektory s přirozenými koeficienty (hodnoty nejvýše rovny počtu vrcholů) a indexované jednotlivými vrcholy. Přípustné jsou ty vektory, kde sousední vrcholy mají různá čísla. Účelovou funkcí je počet použitých hodnot barev.

(1.4.2)* Ano, ve smyslu předchozí formulace prostě nevyžadujeme celočíselnost hodnot, ale sousední vrcholy musí mít hodnoty odlišné aspoň o 1 (tzv. zlomková barevnost grafu). Zlomková barevnost vychází někdy o trochu menší.

(1.4.3) Univerzem jsou binární vektory indexované vrcholy grafu. Omezujícími podmínkami je, že součet hodnot vrcholů u každé hrany je aspoň 1, a účelová funkce jen sčítá hodnoty vrcholů.

(1.5.1) Optimalizujeme třeba počet zpožděných úloh. Hladový postup již není v takovém příkladě moc dobře použitelný. Jak by se mohl řešit?

(1.5.2) Pokud bychom se o parametrech úlohy J_i dozvěděli teprve na jejím začátku t_i, nemohli bychom nic plánovat dopředu a pouze bychom aplikovali primitivní hladový algoritmus.

(1.5.3)* ...

(2.1.1) Dal, přidáme novou hranu zpět ze z do s s neomezenou kapacitou, kterou se tok bude "vracet".

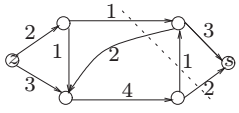
(2.1.2) Je to přirozené, neboť množství přenášené substance se dle definice ve všech ostatních vrcholech zachovává.

(2.2.1) Nic zvláštního, stačilo by všechny (stejně orientované!) násobné hrany nahradit jednou a sečíst jejich kapacity.

(2.2.2)* Byl by to jen formální problém – zápornou hranu lze nahradit opačně orientovanou hranou s opačnou (tj. už kladnou) kapacitou. I pokud bychom záporné hrany neobraceli, můžeme použít náš algoritmus, ale záporné hrany by se "sytily" ze záporných hodnot toku na nulu.

(2.2.3) Tok 5.

(2.2.4) Jen zdroj z.

(2.2.5) Řez velikosti 4 zde: 

(2.3.1) Velikosti 5: 

(2.3.2) Velikosti 4: 

(2.3.3) Ano, reprezentanti jsou zapsaní první: $(1,2,3)$, $(2,1,4)$, $(3,1,4)$, $(4,2,3)$.

(2.3.4) Ne, všech podmnožin je $\binom{5}{3} = 10$, ale prvků k reprezentaci jen 5.

(2.4.1)* …

(3.1.1) Je to, jako bychom 10kg hranolků rozložili zpět na výchozí suroviny. Nesmyslné, ale v jiných úlohách se něco podobného může reálně objevit.

(3.1.2) Je to tehdy, když účelová funkce má přímky rovnoběžné s přímkou $0.4\ell + 0.2h = 16$. To v první formulaci účelové funkce je pro cenu 60Kč, v druhé formulaci pro 66Kč.

(3.1.3)* Obtížné, ale mělo by to jít…

(3.2.1) Protože první střelnice požaduje 3t střeliva ze dvou cest.

(3.2.2) 1.25

(3.3.1)

(3.4.1) 54

(3.4.2) 53.5

(4.1.1) Neformální -1 pro $\emptyset$ a 0 pro bod.

(4.1.2) 6

(4.1.3) Že prochází počátkem souřadnic.

(4.1.4) Kupodivu obojí – zkontrolujte si definice.

(4.2.1) Ano

(4.2.2) Ne

(4.2.3) Jen jeho čtyři vrcholy.

(4.2.4) Celý obvod.

(4.2.5) První mínus druhá, výsledek $2z \leq 0 \leq -1$.

(4.2.6) První plus $2\times$ druhá mínus třetí, výsledek $0 \leq -4$.

(4.3.1) Je to tzv. crosspolytop, zobecňující pravidelný osmistěn.

(4.3.2) Právě crosspolytop.

(4.3.3)* Již v dimenzi 4 – tzv. cyklický polytop.

(4.3.4) Prázdný mnohostěn má alespoň prázdnou stěnu. Zcela beze stěn je "plný" mnohostěn určený žádným poloprostorem.

(4.3.5)* …

(4.3.6)* Jeden směr; pro $S = T^*$ plyne $((T^*)^*)^* \supseteq T^*$. Druhý směrů $U \supseteq V$ zřejmě implikuje $U^* \subseteq V^*$, takže $((T^*)^*)^* \subseteq T^*$.

(4.3.7)* Vyjádřete P jako $P = Q^*$…

(5.2.1) $\min 5y_1 + 3y_3$ pro $y_1 + y_2 \geq 1$, $y_1 + y_2 \geq 0$, $y_1 - y_2 \geq -1$, $y_1, y_2 \geq 0$.

(5.2.2) $\max 5y_1 + 3y_3$ pro $y_1 + y_2 \leq -1$, $y_1 + y_2 \leq 0$, $y_1 - y_2 \leq 1$, $y_1, y_2 \geq 0$.

(5.2.3) Primární nemá optimální řešení.

(5.2.4)* Ano, třeba $\max x_1 + 2x_2$ pro $x_1 + x_2 = 1$, $x_1 + x_2 = 2$ s neomezenými proměnnými je sama sobě duální, přitom je zřejmě bez přípustného řešení.

(5.2.5)* Jednak na začátku – zavedení $\vec{x}^o$, pak po vztahu (5) pro argumentaci, že $\vec{y}$ končí kladnou souřadnicí.

(5.3.1) $\min 5y_1 + 3y_3$ pro $y_1 + y_2 \geq 1$, $y_1 + y_2 \geq 0$, $y_1 - y_2 = -1$, $y_1 \geq 0$.

(6.1.1) Je třeba ještě zkontrolovat nezápornost proměnných a hodnost matice (případně vyloučit závislé řádky).

(6.1.2) $\max\ x_1 - x_3 + x_4 : x_1 + x_2 + x_3 - x_4 + x_5 = 5$, $x_1 + x_2 - x_3 + x_4 = 3$, $x_1, x_2, x_3, x_4, x_5 \geq 0$

(6.2.1) Ve Větě 6.2 nám teprve nezápornost proměnných zaručuje existenci vrcholu s optimálním řešením úlohy.

(6.2.2)* Není to tak jednoduché, ale je třeba se podívat na důkaz Věty 3.5, kde se neomezené proměnné nahrazovaly rozdílem $x = x' - x''$ pro $x', x'' \geq 0$ – pak již bude úloha mít definovány vrcholy (ve vyšší dimenzi) položením těchto proměnných rovno 0.

(7.2.1) Volba $\nu = 1e50$ by byla velmi neuvážená, neboť, jak lze vidět v Příkladě 7.5, konstanta ν se dostane do běžných redukovaných cen a takto velká hodnota by působením zaokrouhlovacích chyb zcela "vymazala" skutečné ceny. Je nutno volit jen tak velkou hodnotu, která se "vejde" do rozsahu přesnosti použité aritmetiky zároveň s běžnými hodnotami. Třeba $1e9$ pro aritmetiku s přesností na 16 míst.

(7.2.2) ...

(7.3.1) ...

(8.1.1) ...

(8.1.2)

(8.2.1) ...

(8.2.2)* Asi je to těžké, když stený příklad se opakuje v různých učebnicích...

(8.3.1) Pokud podmínek–nerovností úlohy velmi mnoho v porovnání s proměnnými.

(8.3.2)* Je to situace, kdy z jednoho vrcholu polyedru do druhého vede exponenciálně dlouhá cesta po hranách, stále ve směru gradientu účelové funkce.

(9.1.1) Za každé letadlo $l = 1, \ldots, 10$ a každou destinaci $X = A, B, C, D$ bude proměnná $z_{l,X}$ indikující, zda letadlo l poletí do X. Podmínky jsou už zřejmé...

(9.1.2) ...

(9.1.3)* Jestliže máme, řekněme, d binárních proměnných v úloze a řešení LP vyjde zhruba kolem vektoru $(\frac{1}{2}, \ldots, \frac{1}{2})$, pak máme 2^d celkem možností tento vektor zaokrouhlit – každou složku nahoru či dolů. To je pro větší d (stačí $d = 30$) nemožné prakticky probrat.

(9.2.1) Celočíselné z_1 spolu se vztahem $x_1 - 0.5 \leq z_1 \leq x_1 + 0.5$.

(9.2.2) Celočíselné z_1 a reálné y_1 spolu se vztahy $0 \leq y_1 \leq 1$ a $x_1 = z_1 + y_1$.

(9.2.3)* Vezměme posloupnost necelých hodnot $x_1 \to 1$. Pak v každém bodě musí být $y_1 = x_1$. Z faktu uzavřenosti množiny přípustných řešení úloh LP a MIP pak plyne, že pro $x_1 = 1$ musí být $y_1 = 1$ přípustným řešením (navíc ke správné hodnotě $y_1 = 0$).

(9.2.4)* ...

(9.3.1) Z vrcholů jednotkové krychle se stane přípustnou celá tato krychle.

(9.3.2)* Úloha MIP U je neomezená, právě když pro některé $\vec{z}^1$ je úloha LP daná $U \restriction \vec{z} = \vec{z}^1$ neomezená.

(9.3.3)* Třeba $\max x_1 : x_1 - x_2 \geq 1,\ x_1 - x_2 + 2z_1 \leq 2$.

(10.1.1) Protože může být mnoho (relaxačních) polyedrů určujících stejnou přípustnou podmnožinu mřížových bodů.

(10.1.2)* Není, vezměte podmínky $y \leq \sqrt{2}x$, $x, y \in \mathbb{N}$. Zde se přípustní řešení mimo 0 nacházejí libovolně blízko přímky $y = \sqrt{2}x$, ale žádné ne na ní, takže konvexní obal nebude ani uzavřený, natož polyedrem.

(10.2.1) Bohužel ne, počet uzlů stromu větvení bude stále proporcionální k $|M|$.

(10.2.2)* ...

(10.4.1) ...

(11.2.1) Stejně jako nezávislé množiny, jen pro doplněk hran grafu.

(11.2.2) Prostě spojte nerovnosti pro nezávislou a pro dominující množinu.

(11.2.3)* ...

(11.3.1) Pokud má graf pouze jeden nebo dva vrcholy.

(11.3.2) Třeba pokud je graf stromem, nebo pokud obsahuje vrchol rozdělující jej na dvě komponenty. Obecně pokud graf nemá Hamiltonovskou kružnici, což je však těžké rozhodnout.

(11.3.3) Pokud graf není souvislý.

(11.3.4)* ...

(12.1.1) Pokud uděláme z každé necelé hodnoty barvy dolní celou část, dostaneme běžné obarvení grafu.

(12.1.2) Že $y - 1$ je rovno největšímu z absolutních rozdílů barev $|x_j - x_i|$ přes hrany grafu.

(12.1.3) Zpět klasickou barevnost.

(12.2.1) ...

(12.2.2)* ???
(12.3.1) Ne, neomezený polyedr totiž nemusí mít vrcholy!
(12.3.2)* ...
(12.3.3) ...
(12.3.4)

Reference

[1] P. Hliněný, Diskrétní Matematika, výukový text FEI VŠB (2005), http://www.cs.vsb.cz/hlineny/vyuka/DIM-slides/.

[2] V. Chvátal, Linear Programming, W.H. Freeman and Co., USA, 1983.

[3] J. Janáček, Matematické Programování, EDIS Žilinská Univerzita, 2003.

[4] L. Kučera, Kombinatorické Algoritmy, SNTL, Praha, 1983.

[5] L. Kučera, Algovize, http://kam.mff.cuni.cz/~ludek/Algovision/Algovision.html.

[6] J. Matoušek a J. Nešetřil, Kapitoly z Diskrétní Matematiky, Karolinum, UK Praha, 2000.

[7] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization. John Wiley, USA, 1988.

[8] J. Oxley, Matroid Theory, Oxford University Press, 1992, ISBN 0-19-853563-5.

[9] J. Oxley, What is a Matroid?, http://www.math.lsu.edu/~preprint/2002/jgo2002e.pdf, LSU, USA.

[10] A. Schrijver, A Course in Combinatorial Optimization. http://homepages.cwi.nl/~lex/files/dict.pdf, CWI, Amsterdam.

[11] G.M. Ziegler. Lectures on Polytopes. Volume 152 of Graduate Texts in Math., Springer-Verlag, New York, 1995.

[12] Jednoduchý aplet pro simplexovou metodu, http://algos.inesc.pt/lp/.

[13] MACEK - strukturální výpočty s matroidy, online přístup http://linux456.vsb.cz/~macek (klient / server služba).

[14] GNU Maxima (5.9) s balíčkem Simplex (1.01, Andrej Vodopivec), http://maxima.sourceforge.net/, http://wxmaxima.sourceforge.net/.

[15] WLPS - řešení úloh lineární a celočíselné optimalizace, online přístup http://mipsolve.cs.vsb.cz/ (klient / server služba).